

Preface

This book is about filter banks and wavelets. Those are new ways to see and represent a signal. They are alternatives to the Fourier transform, using short wavelets instead of long waves. We will explain the advantages (and disadvantages!) of the new methods. The final decision will depend on the application itself, the actual signal, and its bandwidth.

To design and understand wavelets we still use Fourier techniques—the connections between time and frequency. This idea remains at the center of signal processing. Wavelets are “alternatives” rather than replacements. The classical transforms will survive very well. But other ideas have come quickly forward, to be understood and applied.

In a word, the new transforms are much more *local*. An event stays connected to the time when it occurs. Instead of transforming a pure “time description” into a pure “frequency description”, the new methods find a good compromise—a *time-frequency description*. This is like a musical score, with specified frequencies at specified times. Remember that the Heisenberg uncertainty principle stands in the way of perfection! We lose accuracy in time when we gain accuracy in frequency. The musician cannot and would not change frequencies instantly. But our eyes and ears succeed to give location as well as frequency, and the new wavelet transforms have the same purpose.

The extreme case is an instantaneous impulse, with all frequencies in equal amounts. Its Fourier transform has constant magnitude over the whole spectrum. By contrast, a wavelet transform will involve only a small fraction of the wavelets—those that overlap the impulse. Figure 0.1 shows a sum of two extreme cases—an impulse and a pure wave $2 \cos \omega n$. The Fourier transform spreads the impulse while it concentrates the wave (at frequencies ω and $-\omega$ of $e^{i\omega n} + e^{-i\omega n}$). The wavelet transform in the third figure is large at the *time* of the impulse and also large at the *frequency* of the wave.

Purpose of the Book

There are already good books on this subject. The ideas and applications are beautiful, and the word has spread. The bibliography lists many of those books, which have special strengths. Our purpose is different. We believe that a *textbook* is needed. Our text explains filter banks and wavelets from the beginning—in several ways and at least two languages. The examples and exercises come from our courses at M.I.T. and Wisconsin, which brought students from all over engineering and science.

Whether you are working individually or in class, we hope this book clarifies the central ideas. Implicit in that goal is the recognition that we cannot describe every filter and prove every theorem. The book concentrates on the underpinning of the subject, which is now stable. There is a special “*glossary*” to organize and define the terms that are constantly used, some from signal processing and others from mathematics. The central idea is a *perfect reconstruction filter bank*, with properties and purposes selected from this list:

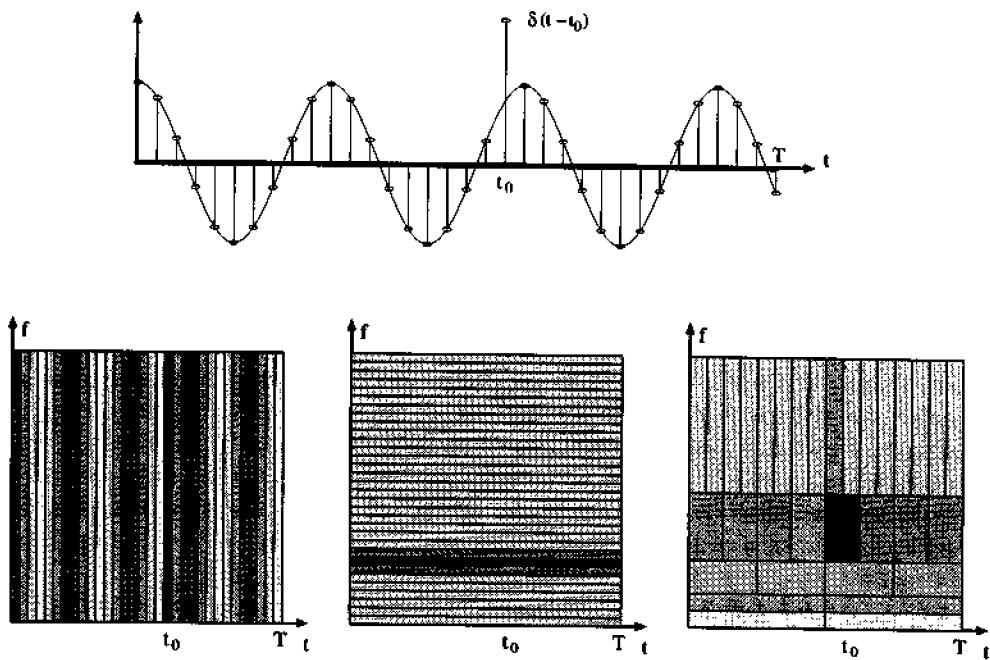


Figure 0.1: Impulse plus sinusoid, in time and frequency and time-frequency(time-scale).

Properties: Orthogonality, symmetry, short length, good attenuation.

Purposes: Audio/video compression, echo cancellation, radar, image analysis, communications, medical imaging,

Design and analysis techniques: Time domain, frequency domain, z -transform.

If the reader is willing, we would like to develop specific examples in this preface. Traditionally, the opening pages thank those who helped to create the book. Our debt to friends will soon be very gratefully acknowledged. But first, we go directly to transforms.

Transforms

Start with the basic idea and its purpose. The transform of a signal (a vector) is a *new representation* of that signal. The components $x(0), x(1), x(2), x(3)$ of a four-dimensional signal are replaced by four other numbers. Those numbers $y(0), y(1), y(2), y(3)$ are combinations of the x 's. Our transforms are linear, so these are *linear combinations* — for example sums and differences:

$$\begin{aligned} y(0) &= x(0) + x(1) & y(2) &= x(2) + x(3) \\ y(1) &= x(0) - x(1) & y(3) &= x(2) - x(3). \end{aligned}$$

What is the purpose of the y 's? And can we get back to the x 's? The second question is easier and we answer it first.

This transform can be inverted. If you add the equations for $y(0)$ and $y(1)$, you get $2x(0)$. Subtracting those equations yields $2x(1)$. The *inverse transform* uses addition and subtraction (like the original!), and then division by 2:

$$\begin{aligned} x(0) &= 0.5(y(0) + y(1)) & x(2) &= 0.5(y(2) + y(3)) \\ x(1) &= 0.5(y(0) - y(1)) & x(3) &= 0.5(y(2) - y(3)). \end{aligned}$$

The y 's allow perfect reconstruction of the x 's, by the inverse transform. Looking ahead for a brief moment, we divide transforms into three groups:

- (a) **Lossless (orthogonal) transforms** (orthogonal and unitary matrices)
- (b) **Invertible (biorthogonal) transforms** (invertible matrices)
- (c) **Lossy transforms** (not invertible)

A lossless unitary transform is like a rotation. The transformed signal has the same length as the original. This is true of the Fourier transform and all its real versions (DCT = discrete cosine transform, DST = discrete sine transform, HT = Hartley transform). The same signal is measured along new perpendicular axes.

For biorthogonal transforms, lengths and angles may change. The new axes are not necessarily perpendicular, but no information is lost. Perfect reconstruction is still available. We just invert. *Orthogonal wavelets give orthogonal matrices and unitary transforms, biorthogonal wavelets give invertible matrices and perfect reconstruction.* These transforms don't remove any information (or any noise), they just move it around — aiming to separate out the noise and decorrelate the signal. The irreversible step is to destroy small components, as we do below in "compression." Then invertibility is lost.

Matrices will appear very early, and we make no apology. A matrix displays the details of the transform. (The key texts in signal processing are amazingly empty of matrices. With the importance of systems like MATLAB, this must change.) Our book maintains a time-domain description by matrices, in parallel with the frequency-domain description by functions of ω . When the inputs $x(n)$ and outputs $y(n)$ are seen as vectors, the transform from x to y is executed by a matrix:

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}.$$

The sum $x(0) + x(1)$ comes from the first row. The difference $x(0) - x(1)$ comes from the second row. Readers may recognize this as the 2-point DFT, and now comes its inverse. The matrix that recovers the x 's from the y 's is changed only by the factor $\frac{1}{2}$:

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix}.$$

If you multiply these matrices, the result is the 4×4 identity matrix.

Small note When the first matrix is divided by $\sqrt{2}$, it becomes an orthogonal matrix. The rows of length $\sqrt{1+1} = \sqrt{2}$ will become unit vectors. The inner products between rows remain at zero. The transform becomes unitary (lossless) when the perpendicular axes — the four rows — are correctly normalized. *The inverse is the transpose.*

The length squared of the transform is now

$$\left(\frac{x(0) + x(1)}{\sqrt{2}}\right)^2 + \left(\frac{x(0) - x(1)}{\sqrt{2}}\right)^2 + \left(\frac{x(2) + x(3)}{\sqrt{2}}\right)^2 + \left(\frac{x(2) - x(3)}{\sqrt{2}}\right)^2.$$

This simplifies to $x(0)^2 + x(1)^2 + x(2)^2 + x(3)^2$. Thus $\|y\|^2 = \|x\|^2$ and the transform is unitary. For real matrices we also use the word orthogonal.

Purpose of the Transform

One important purpose is to see patterns in the y 's that are not so clear in the x 's. This means that the computer should see the pattern. What it sees best is *large versus small*. The computer will notice nothing special about the four numbers

$$x(0) = 1.2 \quad x(1) = 1.0 \quad x(2) = -1.0 \quad x(3) = -1.2.$$

Your eye notices various things in Figure 0.2 (I hope). Maybe the small movement and then the jump. Maybe the antisymmetry. To see this signal in the y representation, compute sums and differences:

$$y(0) = 2.2 \quad y(1) = 0.2 \quad y(2) = -2.2 \quad y(3) = 0.2.$$

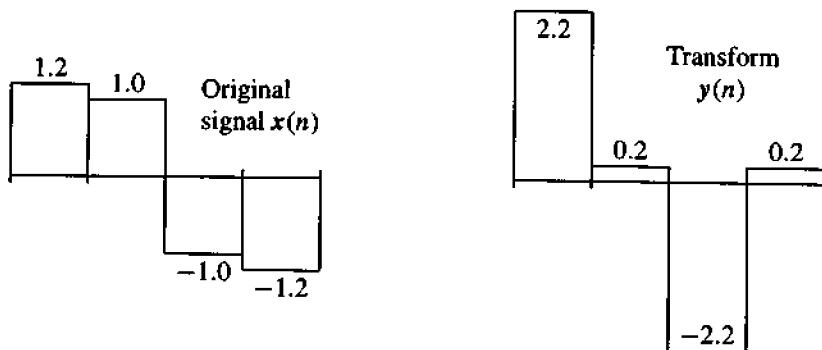


Figure 0.2: The x 's transform to the y 's by sums and differences.

First point: $y(1)$ and $y(3)$ are much smaller than $y(0)$ and $y(2)$. The differences are an order of magnitude smaller than the sums. Our transform has *almost lined up the signal* in the $y(0)-y(2)$ plane. Those two components of y do most of the work of four. It is true that we need all four y 's to reconstruct perfectly all four x 's. But if we change the small numbers $y(1)$ and $y(3)$ —just cancel them!—the compressed signal y_c is

$$y_c(0) = 2.2 \quad y_c(1) = 0 \quad y_c(2) = -2.2 \quad y_c(3) = 0.$$

Those numbers 0.2 were below our threshold. In the compressed y_c they are gone. Figure 0.3 shows the signal x_c reconstructed from y_c —the small difference between $x(0)$ and $x(1)$ is lost.

For comparison we compute the 4-point DFT, expecting and seeing the imaginary number $i = j = \sqrt{-1}$:

$$\begin{bmatrix} \widehat{x}(0) \\ \widehat{x}(1) \\ \widehat{x}(2) \\ \widehat{x}(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} \begin{bmatrix} 1.2 \\ 1.0 \\ -1.0 \\ -1.2 \end{bmatrix} = \begin{bmatrix} 0 \\ 2.2(1+i) \\ 0.4 \\ 2.2(1-i) \end{bmatrix}.$$

That zero in \widehat{x} reflects the antisymmetry. To see it in the sum-difference transform, we must go to the next level.

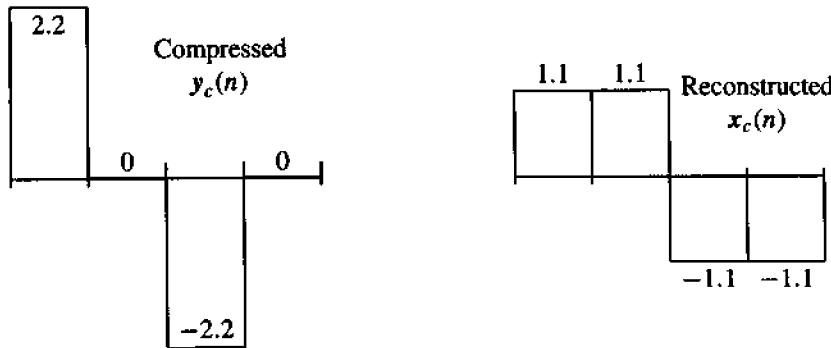


Figure 0.3: y_c and x_c are close to y and x , when small becomes zero.

Multilevel Transforms by Recursion

A key idea for wavelets is the concept of “*scale*.” Sums and differences of neighbors are at the finest scale. We move to a larger picture by taking sums and differences again. This is recursion—the same transform at a new scale. It leads to a *multiresolution* of the original signal $x(0), x(1), x(2), x(3)$. Averages and details will appear at different scales.

The wavelet formulation keeps the differences $y(1)$ and $y(3)$ at the finest level, and *iterates only on $y(0)$ and $y(2)$* . Iteration means sum and difference of the transform:

$$z(0) = y(0) + y(2) = 0 \quad \text{and} \quad z(2) = y(0) - y(2) = 4.4.$$

At this level there is extra compression. One component $z(2)$ now does most of the work of the original four. The wavelet transform $z(0), y(1), z(2), y(3)$ is in Figure 0.4.

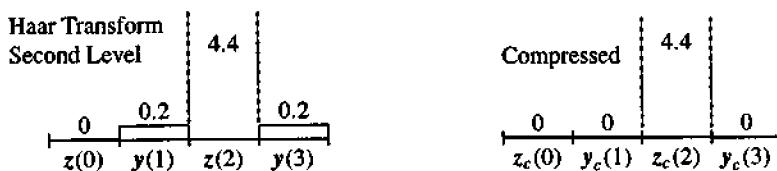
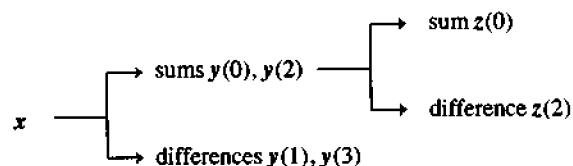


Figure 0.4: $z(0)$ and $z(2)$ are the sum and difference of $y(0)$ and $y(2)$. Compress to $z(2)$.

The key point is the multilevel construction, which is clear in a flow-graph:



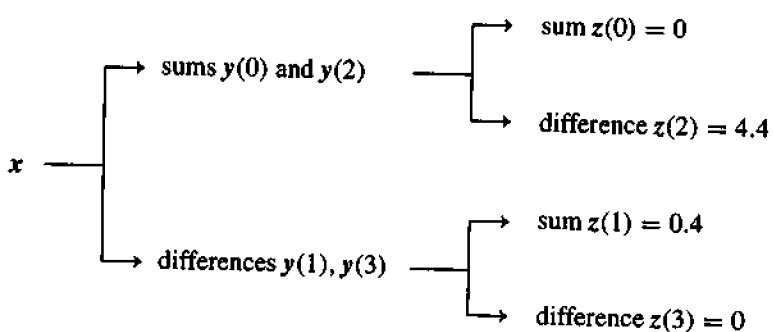
This two-step wavelet transform is executed by a product of two matrices:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 1 & -1 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$

That is invertible! To draw the flow-graph of the inverse, just reverse the arrows. To invert the matrix on the right, multiply the inverses of the matrices on the left (in reverse order of course).

The wavelet transform becomes unitary as before, when we divide sums and differences by $\sqrt{2}$. You will see this number appearing throughout the book, to compensate for scale changes.

Perhaps one more transform could be mentioned. It is the Walsh-Hadamard transform, which iterates *also on the differences* $y(1)$ and $y(3)$. Instead of the “logarithmic tree” for wavelets, we have a complete binary tree for Walsh-Hadamard:



This also counts as a *wavelet packet* (those include every binary tree). You would not want to miss the “Hadamard matrix” for this transform, which is exceptional. All entries are 1 and -1 and all rows are orthogonal:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

When divided twice by $\sqrt{2}$ this matrix is orthogonal and also symmetric. The original x 's can be reconstructed from the y 's or the w 's or the wavelet outputs $y(1), y(3), z(0), z(2)$. From compressed outputs we get approximate (but good) reconstruction.

Don't forget the disadvantage compared to wavelets! All sixteen entries are nonzero. The complete tree takes more computation than the wavelet tree. For signals of length n (a power of two), the Hadamard matrix has n^2 nonzeros and the wavelet matrix has only $n + n \log_2 n$. Those are the costs *without recursion*, jumping from the original x 's directly to the final z 's.

When the computations are recursive, as they always should be, we have a product of very sparse matrices. The flow-graph gives $2n \log_2 n$ for the complete Walsh-Hadamard transform (*exactly matching the Fast Fourier Transform*). This is good but wavelets are slightly better. The transform to $z(0), y(1), z(2), y(3)$ computes the sum and difference of $n - 1$ pairs. This needs only $2n - 2$ calls to memory.

The wavelet transform achieves the Holy Grail of complexity theory (or simplicity theory). *The transform is an $O(n)$ computation*. But does it separate the true signal from noise? Does it allow compression of 4:1 or 8:1 or 20:1? This sum-difference transform is analyzed further in Chapter 1—it is such a good example—but we admit here that it is too simple. Better filters are needed, and they lead to better wavelets (still with $O(n)$ operations). That is the subject of our book.

Applications

The choice of waves or wavelets, Fourier analysis or filter analysis, depends on the signal. It must. Signals coming from different applications have different characteristics. It is helpful to see a broad picture:

Audio: Use many subband filters or a windowed (short time) Fourier transform.

Speech: The time variation is irregular and requires nonuniform intervals.

Images: Finite length signals need special treatment to reduce blocking. Symmetric extension goes with symmetric filters.

Video: Use motion prediction (optical flow of images) or space-time filtering.

The best choice for medical imaging is not clear. The legal questions arising from lossy compression are not clear either. Identification from synthetic aperture radar (SAR images) is an enormous problem. So is de-noising, which is at the heart of signal representation. Ultimately we are trying to choose a good basis.

The problem is to represent typical signals with a small number of conveniently computable functions. The traditional bases (Fourier, Bessel, Legendre, ...) come from differential equations. Wavelets do not come from differential equations! One reason is, those equations don't include dilation. A dilation equation $\phi(t) = \sum 2h(k)\phi(2t - k)$ involves two time scales. Its solution $\phi(t)$ is nonzero only on a finite interval. Then $\phi(2t)$ is nonzero on half of that interval. The basis is localized. It is quickly (and recursively) computed from the numbers $h(k)$. Those are the coefficients in the corresponding lowpass filter.

This relation of filters to functions is the heart of the book.

Acknowledgments

Many friends have created this theory. A lot of them also helped us to write about it. The bibliography points to their original work (we wish it were possible to be complete — that is better attempted electronically). Our personal thanks must begin with Vasily Strela and Chris Heil. They read most of the words and discussed all the ideas. Their unselfish help is deeply appreciated.

Other friends will see, at specific places in the book, exactly where their ideas and suggestions made a difference. We are enormously grateful to Kevin Amaratunga, Ross Barmish, Christopher Brislawn, Charles Chui, Albert Cohen, Adriannus Djohan, Dave Donoho, Jeff Geronimo, Doug Hardin, Peter Heller, Tom Hopper, Angela Kunoth, Seng Luan Lee, Michael Lightstone, Eric Majani, Stéphane Mallat, Henrique Malvar, Ricardo de Queiroz, Jianhong Shen, Wim Sweldens, Pankaj Topiwala, Steffen Trautmann, Chi Wah Kok, Victor Wickerhauser, and Zhifeng Zhang. There is no way to thank you enough for such generous help.

We want to recognize separately the outstanding leadership of four authors, P. P. Vaidyanathan and Martin Vetterli in signal processing and Ingrid Daubechies and Yves Meyer in mathematics. The two subjects are thoroughly mixed, thanks to these four! Their work has been a model in every way, personal (to us) as well as scientific (for all).

Our writing of the book was shared in a natural way — words mostly from the first author, filter designs and applications from the second author, ideas from both. Vasily Strela and Robert Becker helped to get Versions 1.0 to 9.9 into TeX, and Martin Stock made it more beautiful. The whole project grew out of our courses and workshops (which will continue). Those were tremendous sources of inspiration, thanks to the participants.

This subject is not just theory. The ideas have to be implemented, and MATLAB is the outstanding way to do that. It is a pleasure to have this book closely linked to the *Wavelet Toolbox* offered by MathWorks. Exercises that use this Toolbox are in the book; the reader can see the wavelets and the multiresolution they produce. A more extensive Wavelet Manual will come from the same sources: Wellesley-Cambridge Press and MathWorks.

For correspondence about this whole subject, and for comments and corrections of any kind, the authors thank the readers. Especially we thank Jill and Thuy Duong for their patience and support. It is finished at last! We hope you enjoy this book.

Gilbert Strang and Truong Nguyen

M.I.T. and Wisconsin, November 1995

gs@math.mit.edu and nguyen@ece.wisc.edu

<http://saigon.ece.wisc.edu/~waveweb/Tutorials/book.html>

Guide to the Book

This book has a two-part subject. One part is discrete, the other is continuous. In discrete time we develop the idea and applications of *filter banks*. In continuous time we have *scaling functions* $\phi(t)$ and *wavelets* $w(t)$. By a natural limiting process, iteration of the lowpass filter leads to the scaling function. One highpass filter then produces a wavelet. Our goal is to make this connection clear. We find the conditions on the discrete coefficients that lead to good filter banks and good wavelets.

Historically and mathematically, the filters come first. Perfect reconstruction filter banks were developed in the early 1980's. The excitement around wavelets started later (and grew quickly). This excitement was not universal—designers of filter banks naturally asked what was new. Part of the answer is precisely in that process of *iteration*. For a filter to behave well in practice, when it is combined with subsampling and repeated five times, it must have an extra property—not built into earlier designs. This property expresses itself in the frequency domain by a sufficient number of “zeros at π ”. Then the frequency band can be successfully separated into five octaves.

The underlying problem is to choose a good basis. We want to represent a signal well, by a small number of basic signals. These can be sinusoids and they can be wavelets. On a discrete grid, $\omega = \pi$ is the highest frequency at which a signal can oscillate. Those oscillations $x(n) = e^{inx} = (-1)^n$ are stopped by the lowpass filter with a “zero at π ”. The highpass filter lets fast oscillations through, and the synthesis filters can reconstruct the exact input. But *compression* may come between analysis and synthesis. Frequencies that are barely represented will be intentionally lost. That mostly means high frequencies but the filter bank is impartial—it keeps the basis functions that are important to the specific signal. We want to show when, and why, filter banks and wavelets are effective in reconstruction and signal representation and compression.

Filter Banks

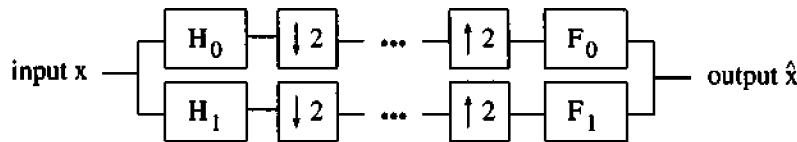
Some readers will begin this book with Chapter 1. Others will jump forward to a topic of particular interest. This brief guide is for both, especially to tell the first readers what is coming and the second group where to look. We are pointing to places where preparation and explanation come together, to design and study new structures.

For filter banks, that place is Section 4.1. There we identify the two conditions for perfect reconstruction (in the absence of lossy compression). One condition removes distortion, the other condition removes aliasing. The anti-distortion condition applies to the products $F_0 H_0$ and $F_1 H_1$ along the channels of the filter bank. Then the anti-aliasing condition controls how those products can be separated into the four filters.

The design of a perfect reconstruction filter bank is a choice of $F_0 H_0$ and then a factorization. To understand the conditions on distortion and aliasing, we apply the techniques of multirate filtering. Those techniques are explained in Chapters 1–3, with examples throughout. Of course

filters are to be defined! But we can go forward even now, to illustrate a filter bank that gives perfect reconstruction.

The analysis bank is on the left. It has a lowpass filter H_0 , and a highpass filter H_1 , and decimation by ($\downarrow 2$) — which removes the odd-numbered components after filtering. The analysis bank yields two “half-length” outputs. Then the synthesis bank on the right begins with the up-sampling operation ($\uparrow 2$) — which inserts zeros in those odd components:

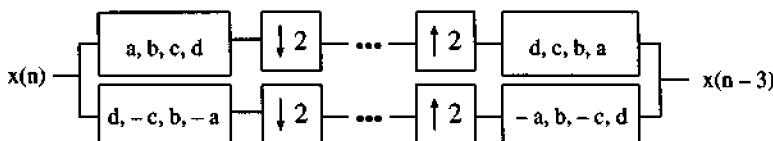


Two-channel filter bank: Separate the input into frequency bands (filter and downsample).
Then reassemble (upsample and filter).

The gap in the center indicates where the subband signals are compressed or enhanced. The applications of this structure are extremely widespread. We believe that any reader interested in signal processing (and image processing) will find that filter bank analysis is extremely useful.

The filters H_0 , H_1 , F_0 , and F_1 are linear and time-invariant. The operators ($\downarrow 2$) and ($\uparrow 2$) are *not* time-invariant. These multirate operations are responsible for *aliasing* and for *imaging* — they create undesirable and extraneous signals that the filters must cancel. To understand how that happens, and to design good filters, we use the tools developed in Chapters 1–3: especially transformation to the frequency domain and z -domain. We try to explain the analysis of multirate filtering, with ($\downarrow 2$) and ($\uparrow 2$), clearly and memorably.

The theory and the design of filter banks and wavelets will dominate Chapters 4–6. This is the heart of the book. The structure of an *orthogonal* bank is very special, and the next figure shows how the filters are related. For length 4 all filters use the four coefficients a, b, c, d that Daubechies derived:



The form of an orthogonal filter bank with four coefficients.

How did she choose a, b, c, d ? Part of the answer will have to wait, but here is the essential idea. The product along the top channel gives a particular “halfband filter”:

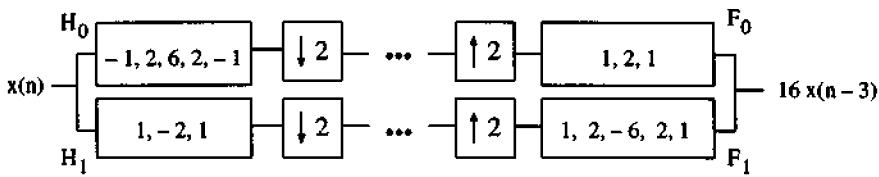
$$(a, b, c, d) * (d, c, b, a) = (-1, 0, 9, 16, 9, 0, -1)/16.$$

This convolution is a multiplication of two polynomials, when a, b, c, d are the coefficients:

$$\begin{aligned} (a + bz^{-1} + cz^{-2} + dz^{-3})(d + cz^{-1} + bz^{-2} + az^{-3}) = \\ (-1 + 9z^{-2} + 16z^{-3} + 9z^{-4} - z^{-6})/16. \end{aligned}$$

The four coefficients are pleasant to calculate. The serious job in Chapter 4 is to explain what is special about that 6th-degree polynomial in which z^{-1} and z^{-5} are missing.

A filter bank also gives perfect reconstruction if it is *biorthogonal*. This design is less restricted. The product $F_0 H_0$ must skip the same odd powers of z^{-1} , but F_0 does not have to be the transpose (the flip) of H_0 . Here are specific numbers for the filter coefficients — not the only choice and maybe not the best. They show how the filters F_0 and F_1 on the synthesis side are related to the analysis filters H_1 and H_0 (by alternating signs):



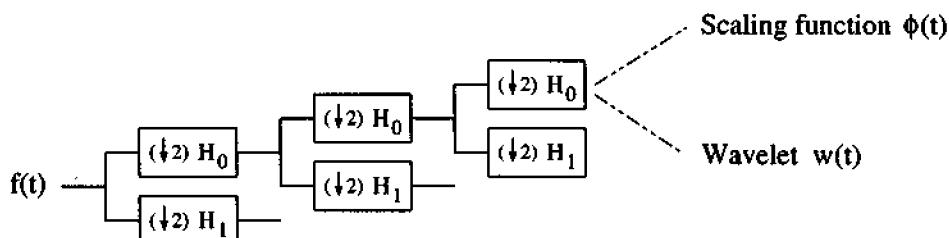
A biorthogonal filter bank: Perfect reconstruction with 3 delays.

For filters, we stop here. This time $F_0(z) = 1 + 2z^{-1} + z^{-2}$. Multiplied by $H_0(z)$ it gives the same important 6th-degree polynomial as before. To understand why the zero coefficients are necessary in that polynomial, and why $-\frac{1}{16}$ and $\frac{9}{16}$ are desirable, I am afraid that you have to read the book!

Our discussion went this far (further than we intended) so as to make a basic and encouraging point: *The construction of new filter banks need not be complicated*. This subject is accessible to new ideas and experiments.

Wavelets

Wavelets are localized waves. Instead of oscillating forever, they drop to zero. They come from the *iteration* of filters (with rescaling). The link between discrete-time filters and continuous-time wavelets is in the limit of a logarithmic filter tree:



Scaling function and wavelets from iteration of the lowpass filter.

Scaling functions and wavelets have remarkable properties. They inherit orthogonality, or biorthogonality, from the filter bank. Because of the repeated rescaling that produces them, wavelets decompose a signal into details at all scales. The wavelet $w(t)$ and its shifts $w(t-k)$ are at unit scale. The wavelets $w(2^j t)$ and $w(2^j t - k)$ are at scale 2^{-j} . The biorthogonal functions $\tilde{\phi}(t)$ and $\tilde{w}(t)$ come from iterating the synthesis bank.

Wavelets produce a natural *multiresolution* of every image, including the all-important edges. Where the low frequency part of the Fourier transform is often a blur, the output from the lowpass channel is a useful compression.

Sections 5.5 and 6.2 study the particular wavelets created by Ingrid Daubechies. They are orthogonal, with the advantages and limitations that this property brings. Sections 4.1 and 6.5 study biorthogonal alternatives, which come from different factorizations of the same polynomial (as above). This polynomial corresponds to a “maxflat halfband filter”, and we hope you will like the connections.

More than that, we hope you enjoy the whole book. This subject is a beautiful combination of mathematical analysis and signal processing applications. The analysis and the applications are based on designs that give perfect reconstruction. To explain both sides of this subject, we need words from mathematics and words from digital signal processing. The Glossary at the end is a dictionary of their meanings. Above all we need *ideas* from both sides, and from a tremendous range of application areas. It is to the understanding of filters and wavelets, and the growth of successful applications, that this book is dedicated.

Summary of the Theory

There are four conditions that play a central part in this book. Because of their importance we highlight them here. They apply directly to the coefficients in the filter banks—and the consequences are felt (after iteration!) in the scaling functions and wavelets. Here are the four conditions—some might say in decreasing order of importance:

PR Condition *Perfect reconstruction.*

The synthesis bank inverts the analysis bank, with ℓ delays.
Biorthogonal banks with no aliasing and no distortion.

Condition O *Orthogonality.*

The analysis bank is inverted by its transpose.
The wavelets are orthogonal to all their dilates and translates.

Condition A_p *Accuracy of order p for approximation by scaling functions $\phi(t - k)$.*

p vanishing moments in the wavelets.
 p th order decay of wavelet coefficients for smooth $f(t)$.

Condition E *Eigenvalue condition on the cascade algorithm.*

Determines convergence to $\phi(t)$ and smoothness of wavelets.
Equivalent to stability of the wavelet basis.

One step further and this Guide is ended. The four fundamental conditions will be stated explicitly for a two-channel filter bank, with the sections in which they appear. We continue to use the polynomials $H_0(z)$, $H_1(z)$, $F_0(z)$, and $F_1(z)$, whose coefficients come directly from the filters. By convention, these are polynomials in z^{-1} , and the lowpass analysis filter is represented by $H_0(z) = h(0) + h(1)z^{-1} + \dots + h(N)z^{-N}$. Here are the conditions that give filters and wavelets with good properties:

1. Perfect Reconstruction (*PR condition in Section 4.1*)

$$F_0(z)H_0(z) + F_1(z)H_1(z) = 2z^{-\ell} \quad \text{and} \quad F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0.$$

The second equation gives the anti-aliasing choices $F_0(z) = H_1(-z)$ and $F_1(z) = -H_0(-z)$.

2. Orthogonality (*Condition O in Section 5.3*)

The filter coefficients are reversed by $F_0(z) = z^{-N} H_0(z^{-1})$ and $F_1(z) = z^{-N} H_1(z^{-1})$. Then perfect reconstruction depends on the “double-shift orthogonality” of the lowpass coefficients $h(k)$:

$$\sum h(k) h(k + 2n) = \delta(n).$$

In terms of the polynomials this is $H_0(z)H_0(z^{-1}) + H_0(-z)H_0(-z^{-1}) = 2$.

3. Accuracy of order p (*Condition A_p in Sections 5.5 and 7.1*)

The lowpass filter has a zero of order p at $z = -1$:

$$H_0(z) = \left(\frac{1+z^{-1}}{2}\right)^p Q(z).$$

4. Convergence and Stability (*Condition E in Section 7.2*)

The transition matrix T has $\lambda = 1$ as simple eigenvalue and all other $|\lambda(T)| < 1$.

Final note: The sixth degree polynomial in the examples above has *four zeros* at $z = -1$:

$$-1 + 9z^{-2} + 16z^{-3} + 9z^{-4} - z^{-6} = (1 + z^{-1})^4 (-1 + 4z^{-1} - z^{-2}).$$

These zeros give flat responses near $\omega = \pi$ and also $\omega = 0$. The absence of z^{-1} and z^{-5} is the key to perfect reconstruction. Polynomials of higher degree, also with zeros at $z = -1$ and also with only one odd power, factor into $F_0(z)H_0(z)$ to give the best filters for iteration. In the limit of the iterations, these filters give good wavelets.

Filters and Matrices

- $\mathbf{h} = (\mathbf{h}(0), \dots, \mathbf{h}(N))$ Causal FIR lowpass filter (impulse response) with $\sum \mathbf{h}(k) = 1$
- \mathbf{H} Toeplitz matrix with $\mathbf{h}(k)$ on the k th diagonal: $\mathbf{H}_{ij} = \mathbf{h}(i - j)$
- $H(\omega) = \sum \mathbf{h}(k) e^{-ik\omega}$ (frequency response in reduced notation)
- $H(e^{j\omega}) = \sum \mathbf{h}(k) e^{-jk\omega}$ (signal processing notation)
- $H(z) = \sum \mathbf{h}(k) z^{-k}$ Transfer function in the z -domain with $z = e^{j\omega}$
- $y = \mathbf{Hx} = \mathbf{h} * \mathbf{x}$ corresponds to $Y(z) = H(z)X(z)$ **Convolution Rule**
- \mathbf{S} Delay gives $(\mathbf{Sh})(k) = \mathbf{h}(k - 1)$ **Time Invariance is $SH = HS$**
- \mathbf{S}^{-1} Advance gives $(\mathbf{S}^{-1}\mathbf{h})(k) = \mathbf{h}(k + 1)$ Multiply transform by $z = e^{-j\omega}$
- $(\downarrow 2)$ Downsampling operator $(\downarrow 2)\mathbf{h} = \mathbf{h}_{\text{even}} = (\mathbf{h}(0), \mathbf{h}(2), \mathbf{h}(4), \dots)$ = even phase of \mathbf{h}
- $(\downarrow 2)\mathbf{S}^{-1}\mathbf{h} = \mathbf{h}_{\text{odd}} = (\mathbf{h}(1), \mathbf{h}(3), \mathbf{h}(5), \dots)$ Odd phase of \mathbf{h}
- $\mathbf{H}_p(z) = [H_{\text{even}}(z) \quad H_{\text{odd}}(z)] = [\sum \mathbf{h}(2k)z^{-k} \quad \sum \mathbf{h}(2k+1)z^{-k}]$ Polyphase representation
- $(\uparrow 2)$ Upsampling operator $(\uparrow 2)\mathbf{v} = [\mathbf{v}(0) \ 0 \ \mathbf{v}(1) \ 0 \ \mathbf{v}(2) \ 0 \ \dots]$ has z -transform $V(z^2)$
- $(\uparrow 2)(\downarrow 2)\mathbf{x} = [\mathbf{x}(0) \ 0 \ \mathbf{x}(2) \ 0 \ \mathbf{x}(4) \ 0]$ has z -transform $\frac{1}{2}[X(z) + X(-z)] = X_{\text{even}}(z^2)$
- $H(z^{-1})$ Transpose filter with coefficients $\mathbf{h}(-k)$: anticausal matrix \mathbf{H}^T
- $z^{-N}H(z^{-1})$ Flip to $\mathbf{h}(N - k)$ produces $(\mathbf{h}(N), \dots, \mathbf{h}(2), \mathbf{h}(1), \mathbf{h}(0))$
- $H(-z)$ Alternating sign $(-1)^k\mathbf{h}(k)$ produces $(\mathbf{h}(0), -\mathbf{h}(1), \mathbf{h}(2), -\mathbf{h}(3), \dots)$
- $z^{-N}H(-z^{-1})$ Alternating flip $(-1)^k\mathbf{h}(N - k)$ produces $(\mathbf{h}(N), \dots, (-1)^N\mathbf{h}(0))$
- $\mathbf{h}(k) = \mathbf{h}(N - k)$ Symmetric filter (**W** symmetry for odd N and **H** symmetry for even N)

Filter Banks

- $L = (\downarrow 2)C = (\downarrow 2)\sqrt{2}H_0$ Lowpass analysis channel: Filter and downsample
- $B = (\downarrow 2)D = (\downarrow 2)\sqrt{2}H_1$ Highpass analysis channel: Filter has $\sum h_1(k) = 0$
- $\sqrt{2}F_0(\uparrow 2)$ Lowpass synthesis channel: Upsample and filter with $F_0(z) = H_1(-z)$
- $\sqrt{2}F_1(\uparrow 2)$ Highpass synthesis channel: Upsample and filter with $F_1(z) = -H_0(-z)$
- $F_0(z)H_0(z) - F_0(-z)H_0(-z) = P_0(z) - P_0(-z) = 2z^{-\ell}$ No distortion: ℓ delays (odd ℓ)
- $H_p(z) = \begin{bmatrix} H_{0,\text{even}}(z) & H_{0,\text{odd}}(z) \\ H_{1,\text{even}}(z) & H_{1,\text{odd}}(z) \end{bmatrix}$ and $F_p(z) = \begin{bmatrix} F_{0,\text{even}}(z) & F_{1,\text{even}}(z) \\ F_{0,\text{odd}}(z) & F_{1,\text{odd}}(z) \end{bmatrix}$ Polyphase matrices
- $H_m(z) = \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix}$ and $F_m = \begin{bmatrix} F_0(z) & F_1(z) \\ F_0(-z) & F_1(-z) \end{bmatrix}$ Modulation matrices
- $F_p(z)H_p(z) = I$ Perfect Reconstruction with no delays
- $F_m(z)H_m(z) = \begin{bmatrix} 2z^{-\ell} & 0 \\ 0 & -2z^{-\ell} \end{bmatrix}$ Perfect Reconstruction with ℓ delays (odd ℓ)
- $H_p^T(z^{-1})H_p(z) = I$ $H_p(z)$ is paraunitary and the filter bank is orthogonal
- $\sum h_0(n)h_0(n+2k) = \delta(k)$ and $H_0(z)H_0(z^{-1}) + H_0(-z)H_0(-z^{-1}) = 2$ Orthogonal lowpass
- $\sum (-1)^k k^m h_0(k) = 0$ for $0 \leq m < p$ Condition A_p gives p zeros of $H_0(e^{j\omega})$ at $\omega = \pi$

List of Scaling Functions and Wavelets

- Haar (box function and up-down square wave)
- Daubechies (maxflat orthogonal with $2p$ filter coefficients)
- Splines and biorthogonal wavelets ($F(z) = (1+z^{-1})^p$ gives spline of degree $p-1$)
- Splines and semiorthogonal wavelets (perpendicular to splines, IIR duals)
- Maxflat biorthogonal ($F(z)H(z)$ is Daubechies maxflat product filter)
- Binary biorthogonal (coefficients are integers times 2^{-n} ; Section 6.5)
- Shannon (ideal filter, sinc scaling function, box function in ω)
- Meyer (IIR but smooth in ω and t)
- Coifman (zero moments $H^{(k)}(0)$ and $\int t^k \phi(t) dt = 0$ for $0 < k < p$)
- Morlet ($e^{-iat} e^{-t^2/2}$ with $a = \pi\sqrt{2/\ln 2}$)

Multiresolution and Wavelets

- $\phi(t)$ from the dilation equation (refinement equation) $\phi(t) = \sum 2\mathbf{h}(k)\phi(2t - k)$
- $w(t)$ from the wavelet equation $w(t) = \sum 2\mathbf{h}_1(k)\phi(2t - k)$
- $w_{jk}(t) = 2^{j/2}w(2^j t - k)$: Normalized wavelet on $[k\Delta t, (k + N)\Delta t]$
- Scale parameter j for stepsize $\Delta t = 2^{-j}$ ($\Delta t = 2^j$ in [D] and MATLAB Toolbox)
- Shift-invariant subspaces $V_j \subset V_{j+1}$ with $V_j \oplus W_j = V_{j+1}$: $f(t) \in V_0 \Leftrightarrow f(2^j t) \in V_j$
- $\{\phi(t - k)\}$ An orthonormal basis (or only a Riesz basis) for V_0
- $\{w(t - k)\}$ An orthonormal basis (or only a Riesz basis) for W_0
- $\{2^{j/2}\phi(2^j t - k)\}$ and $\{2^{j/2}w(2^j t - k)\}$ Bases for V_j and W_j . Joint basis for V_{j+1}
- Orthogonal multiresolution Orthonormal bases with V_j perpendicular to W_j
- Semiorthogonal multiresolution Riesz bases with V_j perpendicular to W_j
- Biorthogonal multiresolution Biorthogonal bases with $V_j \perp \tilde{W}_j$ and $W_j \perp \tilde{V}_j$
- $\tilde{\phi}(t)$ and $\tilde{w}(t)$ from the analysis filters \mathbf{h}_0 and \mathbf{h}_1 generate $\tilde{V}_0 \oplus \tilde{W}_0 = \tilde{V}_1$
- $\phi(t)$ and $w(t)$ from the synthesis filters f_0 and f_1 generate $V_0 \oplus W_0 = V_1$
- $\langle \phi(t - k), \tilde{\phi}(t - \ell) \rangle = \delta(k - \ell)$ Biorthogonal (dual) bases for V_0 and \tilde{V}_0
- $\langle w(t - k), \tilde{w}(t - \ell) \rangle = \delta(k - \ell)$ Biorthogonal (dual) bases for W_0 and \tilde{W}_0
- $a_{jk} = \langle f(t), \tilde{\phi}_{jk}(t) \rangle$ Coefficients in $f_j(t) = \sum_k a_{jk}\phi_{jk}(t)$ = projection of $f(t)$ onto V_j
- $b_{jk} = \langle f(t), \tilde{w}_{jk}(t) \rangle$ Wavelet coefficients in $f(t) = \sum \sum b_{jk}w_{jk}(t)$
- $a_{jk} = \sum \mathbf{h}_0(\ell - 2k) a_{j+1,\ell}$ and $b_{jk} = \sum \mathbf{h}_1(\ell - 2k) a_{j+1,\ell}$ Mallat Fast Wavelet Transform
- $a_{j+1,\ell} = \sum f_0(\ell - 2k) a_{jk} + \sum f_1(\ell - 2k) b_{jk}$ Fast Inverse Wavelet Transform

Dilation Equation — Solution and Smoothness

- $\widehat{\phi}(\omega) = H\left(\frac{\omega}{2}\right) \widehat{\phi}\left(\frac{\omega}{2}\right)$ Fourier transform of the dilation equation
- $\widehat{\phi}(\omega) = \prod_{j=1}^{\infty} H\left(\frac{\omega}{2^j}\right)$ Fourier transform of the scaling function $\phi(t)$
- $\phi^{(i+1)}(t) = \sum 2h(k)\phi^{(i)}(2t - k)$ Cascade algorithm with $\phi^{(0)}(t) = \text{box function}$
- $M = (\downarrow 2) 2 H$ Cascade matrix with double-shifted rows: $M = \sqrt{2}L$
- $m(0)$ and $m(1)$ N by N submatrices of M Entries $2h(2i - j)$ and $2h(2i - j + 1)$
- $m(0)\Phi = \Phi$ Eigenvector with $\lambda = 1$ gives $\Phi = (\phi(0), \phi(1), \dots)$ at the integers
- $\Phi(. t_1 t_2 \dots) = m(t_1)\Phi (. t_2 t_3 \dots)$ Recursion for $\Phi(t) = (\phi(t), \phi(t+1), \dots)$ at dyad
- $m(t_1) m(t_2) \dots m(t_k)$ All products uniformly bounded \leftrightarrow Bounded recursion for $\phi(t)$
- $T = (\downarrow 2) 2 H H^T$ Transition matrix from autocorrelation $h * h^T$ corresponding to $|H(\omega)|^2$
- $Ta = a$ Eigenvector of T gives the inner products $a(k) = \langle \phi(t), \phi(t+k) \rangle$
- $A(z) = \sum a(k)z^{-k}$ and $A(\omega) = \sum |\widehat{\phi}(\omega + 2\pi k)|^2$ Euler-Frobenius polynomial from $h(k)$
- $A(\omega) \equiv 1$ and $A(z) \equiv 1$ and $a = \delta$ Orthonormal basis $\{\phi(t-k)\}$
- $0 < A \leq A(\omega) \leq B$ and $A \sum a_k^2 \leq \|\sum a_k \phi(t-k)\|^2 \leq B \sum a_k^2 \leftrightarrow$ Riesz basis $\{\phi(t-k)\}$
- Condition E for Riesz bases $\{\phi(t-k)\}$ and $\{w_{jk}(t)\}$ and L^2 convergence of $\phi^{(i)}(t)$ to $\phi(t)$:
 - $\lambda = 1$ is a simple eigenvalue of T and all other $|\lambda(T)| < 1$
 - Special eigenvalues $\lambda = 1, \dots, (\frac{1}{2})^{p-1}$ of M and $\lambda = 1, \dots, (\frac{1}{2})^{2p-1}$ of T from p zeros
 - $s_{\max} = -\frac{\log \rho}{\log 4}$ Smoothness of $\phi(t)$ with $\rho = |\lambda_{\max}(T)|$ excluding $\lambda = 1, \frac{1}{2}, \dots, (\frac{1}{2})^{2p-1}$
 - $s_{\max} = p - \frac{1}{2}$ Smoothness in L^2 of splines of degree $p-1$ from $H(z) = \left(\frac{1+z^{-1}}{2}\right)^p$
 - $s_{\max} \leq p - \frac{1}{2}$ Bound on derivatives of $\phi(t)$ when the filter has p zeros at π

Chapter 1

Introduction

1.1 Overview and Notation

We begin with an overview of *filters*, *filter banks*, and *wavelets*. We want to indicate, first in rough outline and then in detail, the connections between these three topics. Our immediate purpose is to open up the problem and the language — starting with the filter coefficients $\mathbf{h}(n)$. The choice of those coefficients is the crucial decision. Their properties govern all that follows.

Each step is a natural development from the one before:

(1) A *filter* is a linear time-invariant operator. It acts on input vectors \mathbf{x} . The output vector \mathbf{y} is the convolution of \mathbf{x} with a fixed vector \mathbf{h} . The vector \mathbf{h} contains the filter coefficients $\mathbf{h}(0), \mathbf{h}(1), \mathbf{h}(2), \dots$. Our filters are digital, not analog, so the coefficients $\mathbf{h}(n)$ come at discrete times $t = nT$. The sampling period T is assumed to be 1 here. The inputs $\mathbf{x}(n)$ and outputs $\mathbf{y}(n)$ come at all times $t = 0, \pm 1, \pm 2, \dots$:

$$\mathbf{y}(n) = \sum_k \mathbf{h}(k) \mathbf{x}(n - k) = \text{convolution } \mathbf{h} * \mathbf{x} \text{ in the time domain.}$$

One input $\mathbf{x} = (\dots, 0, 1, 0, \dots)$ has special importance — a unit impulse at time zero. The input has $\mathbf{x}(n - k) = 0$ except when $n = k$. The sum in the convolution has only one term, and that term is $\mathbf{h}(n)$. This output $\mathbf{y}(n) = \mathbf{h}(n)$ is the response at time n to the unit impulse $\mathbf{x}(0) = 1$. It is the *impulse response* $\mathbf{h}(0), \mathbf{h}(1), \dots, \mathbf{h}(N)$.

In a moment the same filter will be described in the frequency domain. Convolution with the vector \mathbf{h} will become *multiplication* by a function H . It is the simplicity of multiplication that makes this subject a success. The action of a filter in time and frequency is the foundation on which signal processing is built.

(2) A *filter bank* is a set of filters. The analysis bank often has two filters, lowpass and highpass. They separate the input signal into frequency bands. Those subsignals can be compressed much more efficiently than the original signal. Then they can be transmitted or stored. We are describing “subband coding” and its applications. At any time the signals can be recombined (by the *synthesis bank*).

It is not necessary to preserve the full outputs from the analysis filters. Normally they are *downsampled*. *We keep only the even components of the lowpass and highpass filter outputs.*

If there are M filters, then keeping every M th component of each output gives a total of the same length as the input. Critical sampling is the key to subband coding.

This book explains how two or more filters, with downsampling, can jointly achieve properties that are impossible for a single filter. We are particularly interested in “perfect reconstruction FIR filter banks”. In this case the reconstructed output $\hat{x}(n)$ from the synthesis bank is identical to the original input x to the analysis bank (with only a time delay). In matrix language, a banded matrix (for the analysis bank) has a banded inverse (the synthesis bank).

In the frequency domain, each filter leads to a multiplication. But downsampling is *not a time-invariant operation*. If we delay all components of y by one time unit, the output from downsampling is totally different. The new samples $y(-1), y(1), y(3)$ are entirely separate and independent from the original samples $y(0), y(2), y(4)$. Those two subsampled signals are two “phases” of y , not connected. Therefore downsampling alters the multiplication picture in the frequency domain. In fact it introduces *aliasing*.

Chapter 4 will show how the simplicity of multiplication can be rescued by looking at each phase separately. Each phase of y comes from filtering the phases of x (using phases of H). These separate pieces are multiplications in the frequency domain. The whole operation together, filtering followed by downsampling, becomes a matrix multiplication — by the *polyphase matrix*.

This is the foundation of filter bank theory (still to be explained in detail!). The analysis polyphase matrix H_p will reveal the correct synthesis bank for perfect reconstruction. That synthesis filter bank uses H_p^{-1} .

(3) Wavelets are basis functions $w_{jk}(t)$ in continuous time. A basis is a set of linearly independent functions that can be used to produce all admissible functions $f(t)$:

$$f(t) = \text{combination of basis functions} = \sum_{j,k} b_{jk} w_{jk}(t). \quad (1.1)$$

The special feature of the wavelet basis is that all functions $w_{jk}(t)$ are constructed from a single mother wavelet $w(t)$. This wavelet is a small wave (a pulse). Normally it starts at time $t = 0$ and ends at time $t = N$.

The shifted wavelets w_{0k} start at time $t = k$ and end at time $t = k + N$. The rescaled wavelets w_{j0} start at time $t = 0$ and end at time $t = N/2^j$. Their graphs are compressed by the factor 2^j , where the graphs of w_{0k} are translated (shifted to the right) by k :

$$\text{compressed: } w_{j0} = w(2^j t) \quad \text{shifted: } w_{0k}(t) = w(t - k).$$

A typical wavelet w_{jk} is compressed j times and shifted k times. Its formula is

$$w_{jk}(t) = w(2^j t - k).$$

The remarkable property that is achieved by many wavelets is *orthogonality*. The wavelets are orthogonal when their “inner products” are zero:

$$\int_{-\infty}^{\infty} w_{jk}(t) w_{JK}(t) dt = \text{inner product of } w_{jk} \text{ and } w_{JK} = 0. \quad (1.2)$$

In this case the wavelets form an *orthogonal basis* for the space of admissible functions. This basis corresponds to a set of axes that meet at 90° angles — as most good axes do. Orthogonality leads to a simple formula for each coefficient b_{JK} in the expansion for $f(t)$. Multiply the

expansion displayed in equation (1.1) by $w_{JK}(t)$ and integrate:

$$\int_{-\infty}^{\infty} f(t) w_{JK}(t) dt = b_{JK} \int_{-\infty}^{\infty} (w_{JK}(t))^2 dt. \quad (1.3)$$

All other terms in the sum disappear because of orthogonality. Equation (1.2) eliminates all integrals of w_{jk} times w_{JK} , except the one term that has $j = J$ and $k = K$. That term produces $(w_{JK}(t))^2$. Then b_{JK} is the ratio of the two integrals in equation (1.3).

As we describe the connection between filter banks and wavelets, you will see that it is the “*highpass filter*” that leads to $w(t)$. The “*lowpass filter*” leads to a scaling function $\phi(t)$. In most constructions the lowpass filter comes first — ***the scaling function is obtained before the wavelet***. In fact the scaling function (in continuous time) comes from infinite repetition $L L \dots L$ of the lowpass filter, with rescaling at each iteration. The wavelet follows from $\phi(t)$ by just *one* application of the highpass filter.

Multiresolution

At a given resolution of a signal or an image, the scaling functions $\phi(2^j t - k)$ are a basis for the set of signals. The level is set by j , and the time steps at that level are 2^{-j} . The new details at level j are represented by the wavelets $w(2^j t - k)$. Then the smooth signal plus the details, the ϕ 's plus the w 's, combine into a ***multiresolution*** of the signal at the finer level $j + 1$. Averages come from the scaling functions, details come from the wavelets:

$$\begin{array}{c} \text{signal at level } j \text{ (local averages)} \\ + \\ \text{details at level } j \text{ (local differences)} \end{array} \begin{array}{c} \searrow \\ \oplus \\ \nearrow \end{array} \begin{array}{c} \text{signal at level } j + 1 \end{array}$$

That is multiresolution for one signal. When we apply it to all signals, we have multiresolution for *spaces* of functions:

$$\begin{array}{c} V_j = \text{scaling space at level } j \\ \oplus \\ W_j = \text{wavelet space at level } j \end{array} \begin{array}{c} \searrow \\ \nearrow \end{array} \begin{array}{c} V_{j+1} = \text{scaling space at level } j + 1 \end{array}$$

This idea of multiresolution is absolutely basic to wavelet analysis. Again, we are only introducing it. We are sending a coarse signal to the reader, not the details. You only have the input at level 1.

Thus the signal is divided into different ***scales*** of resolution, rather than different frequencies. The “time-scale plane” takes the place for wavelets that the “time-frequency plane” takes for filters. Multiresolution divides the frequencies into *octave bands*, from ω to 2ω , instead of uniform bands from ω to $\omega + \Delta\omega$. The compression of a graph, when $f(t)$ is replaced by $f(2t)$, means expansion of its Fourier transform from $F(\omega)$ to $\frac{1}{2}F(\frac{\omega}{2})$. Frequencies shift upward by an octave, when time is rescaled by two. You will see how the time-frequency plane is partitioned naturally into *rectangles of constant area* (Figure 1.1).

This matching of long time with low frequency and short time with high frequency occurs in a natural way for wavelets. It is one of the attractions of a wavelet decomposition.

To the reader: We have reprinted in Appendix A an article on wavelets published in the American Scientist of May 1994. This article introduces wavelet notation through its correspondence with *musical notation*. In music, each note specifies a frequency and a position in time.

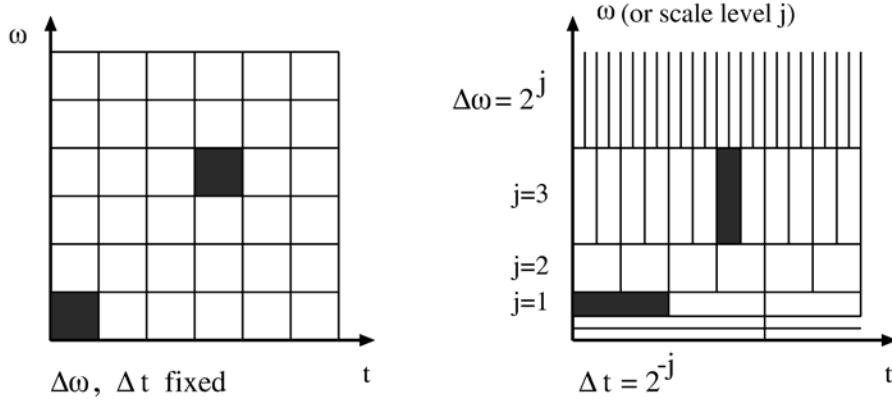


Figure 1.1: Time-frequency squares for Fourier decompositions become rectangles for wavelets.
Short time intervals are natural for high frequencies.

Its vertical placement gives frequency, its horizontal placement indicates time. A musical score is almost a wavelet decomposition — except that it has fractional jumps in frequency. There are notes between middle C and high C, while wavelets jump by octaves. The shortest note I have seen is a 32nd note, corresponding to level $j = 5$ (because $2^{-5} = \frac{1}{32}$). Wavelets often stop there too, in practice. But in principle the scale level j goes to infinity.

That article on wavelets was written for a nontechnical audience, but it aims to explain the essential ideas. In the wavelet decomposition, all instruments play the same tune! They have different amplitudes and they play at different speeds and different times. The basses contribute the coarsest signals $b_{0k}w(t - k)$, starting at integer times $t = k$ and overlapping. The cellos play an identical tune but twice as fast. They contribute $b_{1k}w(2t - k)$, starting at half-integer times $t = k/2$ and again overlapping. The violas and violins add details at levels $j = 2$ and $j = 3$. Those details are wavelets $w(4t)$ and $w(8t)$ and their translates. It is the *orthogonality* of all these tunes, and especially the *localization* of each tune into a short time interval, that makes it possible to decompose the symphony efficiently.

Similarly it is orthogonality (or biorthogonality) and localization that make wavelet decompositions attractive for other signals.

Frequency Domain and Notation

To see a filter as a multiplication, we must take Fourier transforms. This will be the *discrete-time Fourier transform*, since the vectors $\mathbf{x}(n)$ and $\mathbf{h}(n)$ and $\mathbf{y}(n)$ are discrete. The time index n goes from $-\infty$ to ∞ . (A vector with zero components at all negative times is called *causal*.) The transform of \mathbf{x} has two reasonable notations. They both stand for the same transform, which we denote by X :

$$\begin{aligned} X(e^{j\omega}) &= \sum_{-\infty}^{\infty} \mathbf{x}(n) e^{-jn\omega} \quad (\text{signal processing notation}) \\ X(\omega) &= \sum_{-\infty}^{\infty} \mathbf{x}(n) e^{-in\omega} \quad (\text{reduced notation}). \end{aligned}$$

You see two differences. On the right side, one uses j and the other uses i . Both represent $\sqrt{-1}$. On the left side, the standard signal processing notation uses $e^{j\omega}$ while the reduced notation writes only ω . (Each has advantages and we are prepared to print the book both ways!)

The standard notation allows a direct conversion of the Fourier transform to the *z-transform*. The transform is still X but the variable becomes z :

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

We simply replace $e^{j\omega}$ by z , extending the formal definition of X from $e^{j\omega}$ on the unit circle to z in the whole complex plane. (Remember: $e^{j\omega}$ has magnitude 1.) The Fourier transform will dominate the first part of the book, but the *z*-transform appears more frequently in the end.

The reduced notation has the advantage of outstanding simplicity. There will be many, many occasions to write $X(\omega)$ and $X(\omega + \pi)$ or to write $X(e^{j\omega})$ and $X(e^{j(\omega+\pi)})$. The first occasion is the most important right now, and we want to express the action of a filter both ways:

Convolution by \mathbf{h} in time becomes multiplication by H in frequency:

$$\begin{aligned} Y(e^{j\omega}) &= H(e^{j\omega}) X(e^{j\omega}) && \text{in signal processing notation} \\ Y(\omega) &= H(\omega) X(\omega) && \text{in reduced notation.} \end{aligned}$$

This is the transform of $y(n) = \sum \mathbf{h}(k) \mathbf{x}(n - k)$. Exercise 10 will ask you to verify this fundamental fact. It is the “convolution rule”. In the *z*-domain it becomes $Y(z) = H(z) X(z)$. The only inputs to the proof are the definition of convolution and of the transform.

Decision on notation. Simplicity often wins. We keep the freedom to write $X(\omega)$ rather than $X(e^{j\omega})$. In reduced notation, the *frequency response* is $H(\omega)$:

$$H(\omega) = \sum \mathbf{h}(n) e^{-in\omega}.$$

This is the response at frequency ω to a unit input at that frequency. When the input at *each* frequency is $X(\omega) = 1$, the output at each frequency is $H(\omega)$. Those inputs are coming from an impulse (all $X(\omega)$ are equal). **Then the frequency response is the transform of the impulse response:**

When the input is a unit impulse $\mathbf{x}(0) = 1$, the output is $\mathbf{y}(n) = \mathbf{h}(n)$.

When the input is a unit impulse $X(\omega) \equiv 1$, the output is $Y(\omega) = H(\omega)$.

Please note the difference between the vector $\mathbf{x} = (\dots, 0, 0, 1, 0, 0, \dots)$ in discrete time and the function $X(\omega) \equiv 1$ in continuous frequency. If the impulse is delayed to time $t = 1$, the input is $(\dots, 0, 0, 0, 1, 0, \dots)$. The output is equally delayed to $\mathbf{y}(n) = \mathbf{h}(n - 1)$. This is **time-invariance: A delay of the input just produces a delay of the output**.

In the frequency domain, the delayed impulse is $X(\omega) = e^{-i\omega}$. The corresponding output is $Y(\omega) = e^{-i\omega} H(\omega)$. In $e^{j\omega}$ notation this is $Y(e^{j\omega}) = e^{-j\omega} H(e^{j\omega})$. It is the transform of the delayed impulse response $\mathbf{h}(n - 1)$.

Convolution by Hand

A good way to compute $\mathbf{y} = \mathbf{h} * \mathbf{x}$ is to arrange it as an ordinary multiplication — but don't carry digits from one column to the next:

$$\begin{array}{r}
 \begin{array}{rrr} \mathbf{x}(2) & \mathbf{x}(1) & \mathbf{x}(0) \\ \mathbf{h}(2) & \mathbf{h}(1) & \mathbf{h}(0) \\ \hline (2) & (1) & (0) \\ (3) & (2) & (1) \\ (4) & (3) & (2) \\ \hline \mathbf{y}(4) & \mathbf{y}(3) & \mathbf{y}(2) & \mathbf{y}(1) & \mathbf{y}(0)
 \end{array}
 \end{array}
 \quad
 \begin{array}{r}
 \begin{array}{rrr} 3 & 2 & 4 = \mathbf{x} \\ 1 & 5 & 2 = \mathbf{h} \\ \hline 6 & 4 & 8 \\ 15 & 10 & 20 \\ 3 & 2 & 4 \\ \hline 3 & 17 & 20 & 24 & 8 = \mathbf{y}
 \end{array}
 \end{array}$$

The coefficients $\mathbf{x}(n) = 4, 2, 3$ add to $X(0) = 9$. The sum of $\mathbf{h}(n) = 2, 5, 1$ is $H(0) = 8$. Then notice that the sum for $\mathbf{y} = \mathbf{h} * \mathbf{x}$ is $H(0)X(0) = 72$.

Another check is at $\omega = \pi$. The alternating sum $4 - 2 + 3$ gives $X(\pi) = 5$. Similarly $H(\pi) = -2$. Then necessarily $Y(\pi) = (5)(-2)$. This agrees with the alternating sum $3 - 17 + 20 - 24 + 8 = -10$.

Problem Set 1.1

1. Suppose the only nonzero components of \mathbf{x} and \mathbf{h} are $\mathbf{x}(0) = 1$, $\mathbf{x}(1) = 3$, and $\mathbf{h}(0) = \frac{1}{2}$, $\mathbf{h}(1) = \frac{1}{2}$. Compute the outputs $\mathbf{y}(n)$. Verify in the frequency domain that $Y(\omega) = H(\omega)X(\omega)$.
2. What are the components $\mathbf{h}(n)$ for the filter to become a simple advance? For any input vector $\mathbf{x}(n)$, the output is $\mathbf{y}(n) = \mathbf{x}(n+1)$. Find $H(\omega)$ for this filter and verify that $Y(\omega) = H(\omega)X(\omega)$.
3. If the input filter vector \mathbf{x} and the vector \mathbf{h} are both causal, explain why the output \mathbf{y} is also causal (meaning that $\mathbf{y}(n) = 0$ for negative n). If \mathbf{h} is causal and if $\mathbf{x}(n) = 0$ for all $n < 8$, what can you conclude about $\mathbf{y}(n)$?
4. If the output vector \mathbf{y} is causal whenever the input \mathbf{x} is causal, explain why the vector \mathbf{h} must be causal.
5. If $- = (\dots, 0, 0, 1, 0, 0, \dots)$ is the unit impulse at time zero, show that convolution with any vector \mathbf{v} leaves that vector unchanged. Translate this statement $\mathbf{v} * - = \mathbf{v}$ into the frequency domain.
6. What are the seven components of $\mathbf{h} * \mathbf{h}$, if $\mathbf{h}(0) = \mathbf{h}(1) = \mathbf{h}(2) = \mathbf{h}(3) = 1$ (all other $\mathbf{h}(n) = 0$)? Use the long multiplication format in the text.
7. The long multiplication format corresponds to thinking of \mathbf{h} and \mathbf{x} as $\mathbf{h}(0) + 10\mathbf{h}(1) + 100\mathbf{h}(2) + \dots$ and $\mathbf{x}(0) + 10\mathbf{x}(1) + 100\mathbf{x}(2) + \dots$. The product begins with $\mathbf{h}(0)\mathbf{x}(0) + 10(_) + 100(_)$.

This is the convolution rule $Y(z) = H(z)X(z)$ with $z = 10$.

8. Verify the convolution rule $Y = HX$ in the important special case when $\mathbf{h}(1) = 1$ and all other $\mathbf{h}(n) = 0$. Thus, $\mathbf{h} = (\dots, 0, 0, 0, 1, 0, \dots)$.

1. What is $\mathbf{y} = \mathbf{h} * \mathbf{x} = \mathbf{h} * (\dots, \mathbf{x}(-1), \mathbf{x}(0), \mathbf{x}(1), \dots)$?
2. What is $H(e^{j\omega})$, also written $H(\omega)$?
3. In this special case the filter H represents a _____.
4. Verify that $\sum \mathbf{y}(n) e^{-jn\omega}$ agrees with $H(e^{j\omega}) X(e^{j\omega})$.

9. Repeating the previous exercise k times shows that $Y = HX$ is still correct when H is a k -step delay: $\mathbf{h}(k) = 1$ and all other $\mathbf{h}(n) = 0$. The frequency response for this delay is $H = \underline{\hspace{2cm}}$. An arbitrary vector \mathbf{h} is a linear combination of delays (and also advances!). By linearity, $Y = HX$ is true in general.
10. (Important) A direct approach to the convolution rule $Y = HX$. What is the coefficient of z^{-n} in $(\sum \mathbf{h}(k) z^{-k}) (\sum \mathbf{x}(\ell) z^{-\ell})$? Show that your answer agrees with $\sum \mathbf{h}(k) \mathbf{x}(n-k)$. The exponent $-n$ appears in H times X when $k+l$ equals $\underline{\hspace{2cm}}$.
11. The *autocorrelation* $\mathbf{p} = \mathbf{h} * \mathbf{h}^T$ is the convolution of \mathbf{h} with its time reversal (or transpose): $\mathbf{h}^T(n) = \mathbf{h}(-n)$. Express the k th component $\mathbf{p}(k)$ as a sum of terms.

1.2 Lowpass Filter = Moving Average

We go forward with this introduction by studying the simplest lowpass filter. Its output at time $t = n$ is the average of the input $\mathbf{x}(n)$ at that time and the input $\mathbf{x}(n-1)$ at the previous time:

$$y(n) = \frac{1}{2}\mathbf{x}(n) + \frac{1}{2}\mathbf{x}(n-1). \quad (1.4)$$

The filter coefficients are $\mathbf{h}(0) = \frac{1}{2}$ and $\mathbf{h}(1) = \frac{1}{2}$. Equation (1.4) fits the standard form $\sum \mathbf{h}(k) \mathbf{x}(n-k)$, with only two terms $k = 0$ and $k = 1$ in the sum. This is a convolution $\mathbf{y} = \mathbf{h} * \mathbf{x}$. It is a *moving average*, because the output averages the current component $\mathbf{x}(n)$ with the previous one. Old components drop away as the average moves forward with the time.

Suppose the input is the unit impulse $\mathbf{x} = (\dots, 0, 0, 1, 0, 0, \dots)$. Then there are only two nonzero components in the output. The input vector has $\mathbf{x}(0) = 1$; all other input components are zero. The output vector has $y(0) = \frac{1}{2}$, from equation (1.4) with $n = 0$. It also has $y(1) = \frac{1}{2}$, from equation (1.4) with $n = 1$. All other outputs, or moving averages, are zero. Thus the impulse response is the vector $\mathbf{y} = (\dots, 0, 0, \frac{1}{2}, \frac{1}{2}, 0, \dots)$. Its components agree with the filter coefficients $\mathbf{h}(n)$ as they should.

We want to see this filter as a linear time-invariant operator. It is a combination of two special operators, the “identity” which yields output = input and the “delay” whose output is the input one time earlier:

$$\text{averaging filter} = \frac{1}{2}(\text{identity}) + \frac{1}{2}(\text{delay}).$$

Every linear operator acting on the signal vector \mathbf{x} can be represented by a matrix. Since the vectors are infinite, so are the matrices. Infinitely many components in \mathbf{x} and \mathbf{y} mean infinitely many entries in the filter matrix \mathbf{H} . The matrix has a special structure which you see immediately in $\mathbf{y} = \mathbf{Hx}$:

$$\begin{bmatrix} \cdot \\ y(-1) \\ y(0) \\ y(1) \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ & \frac{1}{2} & \frac{1}{2} & & \\ & & \frac{1}{2} & \frac{1}{2} & \\ & & & \ddots & \end{bmatrix} \begin{bmatrix} \cdot \\ \mathbf{x}(-1) \\ \mathbf{x}(0) \\ \mathbf{x}(1) \\ \cdot \end{bmatrix}$$

The numbers $\frac{1}{2}$ on the main diagonal come from $\frac{1}{2}$ (identity). The numbers $\frac{1}{2}$ on the subdiagonal come from $\frac{1}{2}$ (delay). Substitute the unit impulse for \mathbf{x} , with $\mathbf{x}(0) = 1$ as its only nonzero component. Matrix multiplication produces the impulse response \mathbf{y} . This vector has components $\frac{1}{2}$

and $\frac{1}{2}$. The response \mathbf{y} is the filter vector \mathbf{h} , in the middle column of the matrix, when \mathbf{x} is the unit impulse.

This is a first occasion to see a filter as a ***constant-diagonal matrix***. The coefficient $\mathbf{h}(0)$ appears constantly down the main diagonal. It represents $\mathbf{h}(0)$ times the identity matrix and it yields $\mathbf{h}(0)\mathbf{x}(n)$. The coefficient $\mathbf{h}(1)$ appears down the first subdiagonal, to represent $\mathbf{h}(1)$ times a delay and to yield $\mathbf{h}(1)\mathbf{x}(n-1)$. If there is a coefficient $\mathbf{h}(2)$ down the next diagonal, it multiplies a two-step delay and yields $\mathbf{h}(2)\mathbf{x}(n-2)$. The total output $\mathbf{y}(n)$ is the sum of these special outputs:

$$\begin{aligned}\mathbf{y}(n) &= \mathbf{h}(0)\mathbf{x}(n) + \mathbf{h}(1)\mathbf{x}(n-1) + \mathbf{h}(2)\mathbf{x}(n-2) + \dots \\ &= \sum_k \mathbf{h}(k)\mathbf{x}(n-k).\end{aligned}$$

The reader notices that $\mathbf{h}(-1)$ is not being allowed. We are dealing with ***causal filters***; the output cannot come earlier than the input. This makes $\mathbf{h}(n) = 0$ for negative n , and it makes the filter matrix *lower triangular*.

Our example has only a finite number (two) of nonzero filter coefficients $\mathbf{h}(n)$. The filter has a ***finite impulse response***. It is an “FIR filter”. For large n , the coefficients $\mathbf{h}(n)$ that give distant responses to the unit impulse are all zero.

To repeat: A causal FIR filter has $\mathbf{h}(n) = 0$ for all negative n and for large positive n . The matrix is banded and lower triangular. Only a finite number of coefficients $\mathbf{h}(0), \mathbf{h}(1), \dots, \mathbf{h}(N)$ can be nonzero. The filter has $N + 1$ “taps”. The matrix has bandwidth N ; all other diagonals contain zeros. We concentrate almost exclusively on causal FIR filters.

Frequency Response

To find the frequency response to the filter, we change the input vector. Instead of an impulse, which combines all frequencies, the vector \mathbf{x} will have pure frequency ω . Its components are

$$\mathbf{x}(n) = e^{in\omega}. \quad (1.5)$$

This vector extends over all time, $-\infty < n < \infty$. Its special feature is that *the output vector \mathbf{y} is a multiple (depending on ω) of the input vector \mathbf{x}* . A linear time-invariant operator, which is represented by a constant-diagonal matrix, has a pure frequency response to a pure frequency input. For our moving average this response is

$$\begin{aligned}\mathbf{y}(n) &= \frac{1}{2}\mathbf{x}(n) + \frac{1}{2}\mathbf{x}(n-1) \\ &= \frac{1}{2}e^{in\omega} + \frac{1}{2}e^{i(n-1)\omega} \\ &= \left(\frac{1}{2} + \frac{1}{2}e^{-i\omega}\right)e^{in\omega}.\end{aligned} \quad (1.6)$$

You see in parentheses the ***frequency response function*** $H(\omega) = \frac{1}{2} + \frac{1}{2}e^{-i\omega}$. Notice that $H = \frac{1}{2} + \frac{1}{2} = 1$ when $\omega = 0$. At zero frequency (direct current) the signal $\mathbf{x} = (\dots, 1, 1, 1, \dots)$ comes out unchanged as $\mathbf{y} = (\dots, 1, 1, 1, \dots)$. The low frequencies, near $\omega = 0$, have $H(\omega)$ near 1. Thus the name “lowpass filter”.

There will be a similar $H(\omega)$ for other filters. When the filter coefficients are $\mathbf{h}(n)$, the response to $\mathbf{x}(n) = e^{in\omega}$ is $\mathbf{y}(n) = \mathbf{h}(0)e^{in\omega} + \mathbf{h}(1)e^{i(n-1)\omega} + \dots$. The input is multiplied by the frequency response $H(\omega)$:

$$\begin{aligned}H(\omega) &= \mathbf{h}(0) + \mathbf{h}(1)e^{-i\omega} + \mathbf{h}(2)e^{-2i\omega} + \dots \\ &= \sum \mathbf{h}(n)e^{-in\omega}.\end{aligned} \quad (1.7)$$

This function is always periodic: $H(\omega + 2\pi) = H(\omega)$. When we add 2π to the frequency ω , we add $2\pi n$ to the angle $n\omega$. The cosine, sine and complex exponential $e^{-in\omega} = \cos n\omega - i \sin n\omega$ are not changed.

Note that the response function $H(\omega)$ involves complex numbers. We strongly prefer $e^{i\omega}$ or $e^{j\omega}$ to its rectangular form $\cos \omega + i \sin \omega$. Separate formulas for the real and imaginary parts are much more complicated than a single formula for $H(\omega)$. But a single graph cannot so easily represent this complex function. One way to do it is to plot the magnitude $|H(\omega)|$ separately from the phase angle $\phi(\omega)$, recalling that

$$H(\omega) = |H(\omega)| e^{i\phi(\omega)}.$$

Our example has $H(\omega) = \frac{1}{2} + \frac{1}{2}e^{-i\omega}$. We factor out $e^{-i\omega/2}$ to leave a symmetric quantity $\frac{1}{2}(e^{i\omega/2} + e^{-i\omega/2})$. This quantity is a perfect cosine:

$$H(\omega) = \left(\cos \frac{\omega}{2} \right) e^{-i\omega/2}. \quad (1.8)$$

This displays the magnitude and also the phase:

$$|H(\omega)| = \cos \frac{\omega}{2} \quad \text{and} \quad \phi(\omega) = -\frac{\omega}{2}. \quad (1.9)$$

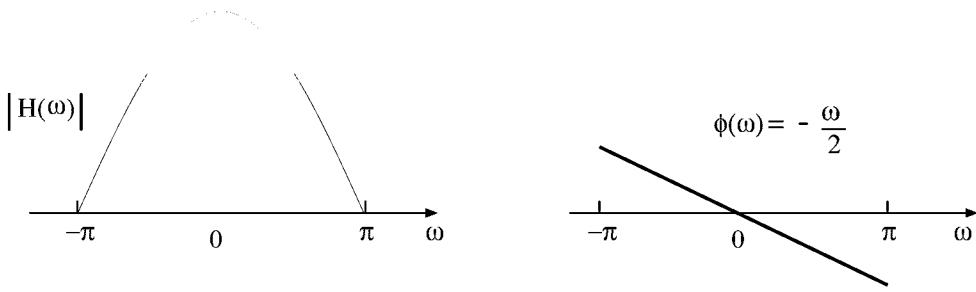


Figure 1.2: The magnitude $|H(\omega)| = \cos(\frac{\omega}{2})$ and the phase of $H(\omega)$.

Figure 1.2 shows the plot of the magnitude $|H(\omega)|$ against the frequency ω . The cosine of $\frac{\omega}{2}$ drops to zero at $\omega = \pi$. This high frequency is wiped out, when the filter takes a moving average. In the time domain, $\omega = \pi$ corresponds to the input vector $\mathbf{x}(n) = e^{i\pi n}$ with components $(-1)^n$. This input vector is

$$\mathbf{x} = (\dots, 1, -1, 1, -1, 1, \dots).$$

The moving average of these components is constantly zero! This confirms $H(\pi) = 0$ as the correct response to the frequency $\omega = \pi$.

This is a lowpass filter. The lowest frequency $\omega = 0$, which is the DC term (direct current), is exactly preserved because $\cos 0 = 1$. The input vector $(\dots, 1, 1, 1, \dots)$ is equal to its moving average.

The phase $\phi(\omega)$ is the angle from the horizontal when the complex number $H(\omega)$ is plotted in the complex plane. For this particular response, those points $H(\omega) = \frac{1}{2} + \frac{1}{2}e^{-i\omega}$ lie along a circle. The constant term $\frac{1}{2}$ is the center. Then $\frac{1}{2}e^{-i\omega}$ produces a circle of radius $\frac{1}{2}$ around that center. Figure 1.3 shows this “Nyquist diagram”.

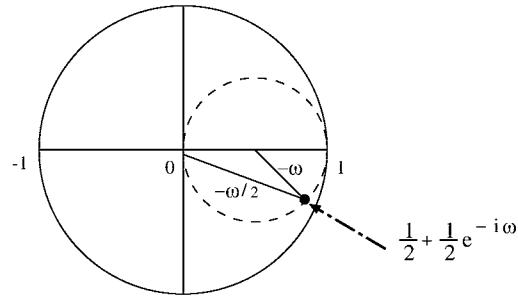


Figure 1.3: Nyquist diagram of the points $H(\omega)$ in the complex plane.

The graph of $\phi(\omega) = -\frac{\omega}{2}$ is a straight line. This is an example of “*linear phase*”, an important property that some special filters possess. It reflects the fact that the filter coefficients $\frac{1}{2}$ and $\frac{1}{2}$ are symmetric. Reverse the order of coefficients and nothing changes.

Note that if the filter coefficients were symmetric *around zero*, so that $\mathbf{h}(-1) = \mathbf{h}(1)$ and $\mathbf{h}(-2) = \mathbf{h}(2)$, the frequency response would be *real*:

$$\begin{aligned} H(\omega) &= \mathbf{h}(0) + \mathbf{h}(1) (e^{i\omega} + e^{-i\omega}) + \dots && \text{(symmetric coefficients)} \\ &= \mathbf{h}(0) + \mathbf{h}(1) (2 \cos \omega) + \dots && \text{(real response function)} \end{aligned}$$

In this case the phase angle is $\phi = 0$. The response has “zero phase”. Similarly, coefficients that are *antisymmetric* around zero produce a pure imaginary $H(\omega)$. The phase is $\frac{\pi}{2}$ or $-\frac{\pi}{2}$. If $\mathbf{h}(-n) = -\mathbf{h}(n)$ then $\mathbf{h}(0) = 0$:

$$\begin{aligned} H(\omega) &= \mathbf{h}(1) (-e^{i\omega} + e^{-i\omega}) + \mathbf{h}(2) (-e^{2i\omega} + e^{-2i\omega}) + \dots && \text{(antisymmetry)} \\ &= -2i [\mathbf{h}(1) \sin \omega + \mathbf{h}(2) \sin 2\omega + \dots] && \text{(imaginary } H(\omega)) \end{aligned}$$

Zero phase is ruled out for a causal filter. The coefficients can be symmetric or antisymmetric, but not around $n = 0$. Causal filters have *linear phase* when their coefficients are symmetric or antisymmetric around the central coefficient:

$$\begin{array}{lll} \text{Linear phase} & \mathbf{h}(k) & = \mathbf{h}(N-k) \quad (\text{symmetry}) \\ & \mathbf{h}(k) & = -\mathbf{h}(N-k) \quad (\text{antisymmetry}) \end{array}$$

Moving the center of the symmetry from 0 to $N/2$ produces a factor $e^{-iN\omega/2}$ in $H(\omega)$. This means a *linear* term $-N\omega/2$ in the phase. The magnitude $|H(\omega)|$ is an even (symmetric) function because $H(-\omega)$ is the complex conjugate of $H(\omega)$. The graph of $|H(\omega)|$ for $0 \leq \omega \leq \pi$ displays complete information.

The exercises ask you to compute $|H(\omega)|^2 = \frac{1}{2}(1 + \cos \omega)$. Then a trigonometric identity produces our special form $\cos \frac{\omega}{2}$:

$$|H(\omega)|^2 = \frac{1}{2}(1 + \cos \omega) = \cos^2 \frac{\omega}{2} \quad \text{so that} \quad |H(\omega)| = \left| \cos \frac{\omega}{2} \right|.$$

For other filters, $|H(\omega)|^2$ is a cosine series but its square root $|H(\omega)|$ has no simple formula.

Problem Set 1.2

1. The magnitude squared is $H(\omega)$ times its complex conjugate $\overline{H(\omega)}$. Show that $|H(\omega)|^2 = \frac{1}{2}(1 + \cos \omega)$ for the moving average filter.
2. Obtain the same result from $|H(\omega)|^2 = (\text{real part})^2 + (\text{imaginary part})^2$, with $H(\omega) = \frac{1}{2} + \frac{1}{2}(\cos \omega - i \sin \omega)$.
3. Why is the formula $|H(\omega)| = \cos \frac{\omega}{2}$ wrong beyond the frequency $\omega = \pi$? Draw the graph of $|H(\omega)|$ from -2π to 2π .
4. In the complex plane, draw $\tan \phi(\omega)$ as the ratio of the imaginary part $\text{Im } H(\omega)$ to the real part $\text{Re } H(\omega)$. Simplify this ratio to $-\tan \frac{\omega}{2}$.

Solution. The phase $\phi(\omega)$ is the angle from the horizontal, so the tangent of $\phi(\omega)$ is the ratio of imaginary part to real part:

$$\tan \phi(\omega) = \frac{\text{Im } H(\omega)}{\text{Re } H(\omega)} = \frac{-\frac{1}{2} \sin \omega}{\frac{1}{2} + \frac{1}{2} \cos \omega}.$$

Again trigonometric identities make this unusually simple:

$$\tan \phi(\omega) = \frac{-\sin \frac{\omega}{2} \cos \frac{\omega}{2}}{\cos^2 \frac{\omega}{2}} = -\tan \frac{\omega}{2} \quad \text{and} \quad \phi(\omega) = -\frac{\omega}{2}.$$

5. Find the frequency response $H(\omega)$ and its magnitude and phase, for the 3-point moving average filter with $h(0) = h(1) = h(2) = \frac{1}{3}$. Does it have zero phase, constant phase, or linear phase?
6. What infinite matrix \mathbf{H} represents the filter with $\mathbf{h}(0) = \mathbf{h}(1) = \mathbf{h}(2) = \frac{1}{3}\mathbf{I}$? Find two input vectors $\mathbf{x}(n)$ for which the output is $\mathbf{Hx} = \mathbf{0}$.

Solution. $\mathbf{x} = (\dots, 2, -1, -1, 2, -1, -1, \dots)$ and $\mathbf{x}' = \text{delay of } \mathbf{x}$.

7. What is the phase $\phi(\omega)$ of a symmetric filter with $\mathbf{h}(k) = \mathbf{h}(8-k)$?
8. What constant-diagonal matrix represents the anticausal filter \mathbf{H}' with coefficients $\mathbf{h}'(0) = \frac{1}{2}$ and $\mathbf{h}'(-1) = \frac{1}{2}$? What matrix represents the symmetric filter $\mathbf{F} = \mathbf{H}'\mathbf{H}$ and what is the frequency response of \mathbf{F} ?
9. Find causal filters whose magnitude responses are $|H(\omega)| = |\cos \omega|$ and $|H(\omega)| = (\cos \frac{\omega}{2})^2$. Are they unique?
10. Iterate the averaging filter four times to get $\mathbf{K} = \mathbf{H}^4$. What is $K(\omega)$ and what is the impulse response $\mathbf{k}(n)$?
11. Why is an antisymmetric filter, with $\mathbf{h}(k) = -\mathbf{h}(N-k)$, never lowpass?
12. Consider an averaging filter with four coefficients $\mathbf{h}(0) = \mathbf{h}(1) = \mathbf{h}(2) = \mathbf{h}(3) = \frac{1}{4}\mathbf{I}$ and the input $\mathbf{x}(n) = (-1)^n\mathbf{I}$. What is the output $\mathbf{y}(n) = \mathbf{x}(n) * \mathbf{h}(n)$? Without computing $H(\omega)$, explain why $\mathbf{h}(n)$ is lowpass.
13. Let $H(z)$ be a lowpass filter with $N+1$ coefficients and let $G(z) = z^{-N}H(z^{-1})$. Find $\mathbf{g}(n)$ in terms of $\mathbf{h}(n)$. What is the phase of $G(\omega)$ if H is symmetric? Is $G(z)$ lowpass or highpass?

1.3 Highpass Filter = Moving Difference

A lowpass filter takes “averages”. It smoothes out the bumps in the signal. A bump is a high-frequency component, which the lowpass filter reduces or removes. The response is small or zero near the highest discrete-time frequency $\omega = \pi$.

A *highpass filter* takes “differences”. It picks out the bumps in the signal. The smooth parts are low-frequency components, which the highpass filter reduces or removes. Now the frequency response is small or zero for frequencies near $\omega = 0$ (which is direct current and has no bumps).

The lowpass filter that outputs the moving average $\frac{1}{2}(\mathbf{x}(n) + \mathbf{x}(n - 1))$ has a twin (or mirror) filter. This is the highpass filter that computes *moving differences*:

$$\text{Highpass: } \mathbf{y}(n) = \frac{1}{2}\mathbf{x}(n) - \frac{1}{2}\mathbf{x}(n - 1). \quad (1.10)$$

The new filter coefficients are $\mathbf{h}(0) = \frac{1}{2}$ and $\mathbf{h}(1) = -\frac{1}{2}$. Equation (1.10) is a convolution $\mathbf{y} = \mathbf{h} * \mathbf{x}$, and the vector \mathbf{h} for this highpass filter is

$$\mathbf{h} = (\dots, 0, 0, \frac{1}{2}, -\frac{1}{2}, 0, \dots). \quad (1.11)$$

This \mathbf{h} is exactly the response to the impulse $\mathbf{x} = (\dots, 0, 0, 1, 0, 0, \dots)$. At time zero, the difference $\frac{1}{2}(\mathbf{x}(0) - \mathbf{x}(-1))$ is $\frac{1}{2}(1 - 0)$. The next difference $\frac{1}{2}(\mathbf{x}(1) - \mathbf{x}(0))$ is $\frac{1}{2}(0 - 1)$. Those numbers $\frac{1}{2}$ and $-\frac{1}{2}$ are the coefficients in \mathbf{h} .

This unit impulse at time zero we will denote by δ . In the language of convolution, we are saying that $\delta = (\dots, 0, 0, 1, 0, 0, \dots)$ always yields

$$\mathbf{h} * \delta = \mathbf{h}. \quad (1.12)$$

In the language of matrices, multiplying any matrix times the special column vector δ always picks out the *zeroth column of the matrix*. The matrix for our highpass filter does have $\frac{1}{2}$ and $-\frac{1}{2}$ in its zeroth column:

$$\begin{bmatrix} \cdot \\ \mathbf{y}(-1) \\ \mathbf{y}(0) \\ \mathbf{y}(1) \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot \\ -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} \mathbf{x}(-1) \\ \mathbf{x}(0) \\ \mathbf{x}(1) \\ \cdot \end{bmatrix}. \quad (1.13)$$

This is again a constant-diagonal matrix because the filter is again time-invariant. (We define and discuss time-invariance in the next chapter.) The main diagonal entries produce $\frac{1}{2}$ (identity). The subdiagonal entries give $-\frac{1}{2}$ (delay). The filter as a whole is

$$\text{highpass filter} = \frac{1}{2}(\text{identity}) - \frac{1}{2}(\text{delay}) = \frac{1}{2}\mathbf{I} - \frac{1}{2}\mathbf{S}. \quad (1.14)$$

This is another causal FIR filter with two taps, but its frequency response is completely different from the lowpass function $H_0(\omega) = \frac{1}{2}(1 + e^{-i\omega})$.

Frequency Response

At frequency ω , the input vector is $x(n) = e^{in\omega}$. The highpass output is

$$\begin{aligned} y(n) &= \frac{1}{2} e^{in\omega} - \frac{1}{2} e^{i(n-1)\omega} \\ &= \left(\frac{1}{2} - \frac{1}{2} e^{-i\omega} \right) e^{in\omega} \\ &= H_1(\omega) e^{in\omega}. \end{aligned} \quad (1.15)$$

This quantity $H_1(\omega) = \frac{1}{2} - \frac{1}{2} e^{-i\omega}$ is the highpass response. We want to graph it and compare it with $H_0(\omega)$ — the lowpass response. *We are introducing the subscripts 0 and 1 for lowpass and highpass.*

As before, we take out a factor $e^{-i\omega/2}$. This leaves a sine not a cosine:

$$H_1(\omega) = \frac{1}{2} (e^{i\omega/2} - e^{-i\omega/2}) e^{-i\omega/2} = \sin\left(\frac{\omega}{2}\right) i e^{-i\omega/2}. \quad (1.16)$$

The magnitude is $|H_1(\omega)| = |\sin \frac{\omega}{2}|$. Since the sine function can be negative, we must take its absolute value.

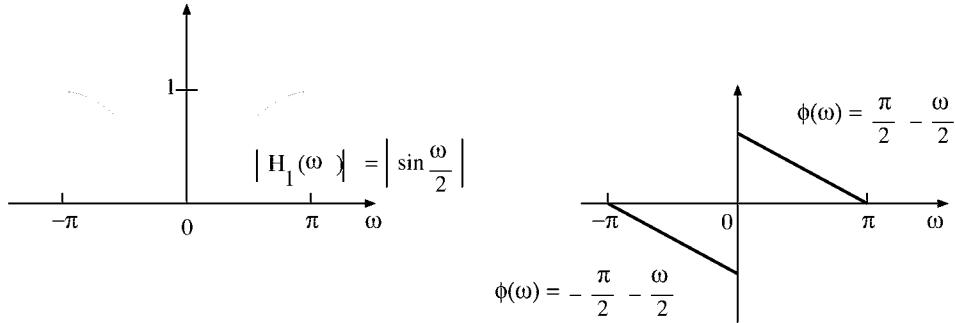


Figure 1.4: The magnitude and phase of the highpass filter $\frac{1}{2} - \frac{1}{2} e^{-i\omega}$.

The graph of $|H_1(\omega)|$ is in Figure 1.4a. It shows zero response to direct current, because $\sin 0 = 0$. It shows unit response at the highest frequency $\omega = \pi$, because $\sin \frac{\pi}{2} = 1$. In the time domain, these numbers come from taking differences of the lowest and highest frequency inputs:

$$x_{\text{low}} = (\dots, 1, 1, 1, 1, 1, \dots) \quad \text{and} \quad x_{\text{high}} = (\dots, 1, -1, 1, -1, 1, \dots).$$

The response to x_{low} is $0x_{\text{low}}$. The response to x_{high} is $1x_{\text{high}}$. The first vector has no bumps and the second vector is all bumps.

The phase factor in $H_1(\omega)$ is a little tricky. When the magnitude $|\sin \frac{\omega}{2}|$ is removed, we are left with a sign change at $\omega = 0$:

$$e^{i\phi(\omega)} = \begin{cases} -i e^{-i\omega/2} & \text{for } -\pi < \omega < 0 \\ +i e^{-i\omega/2} & \text{for } 0 < \omega < \pi. \end{cases}$$

Changing i to $e^{i\pi/2}$, we see a *discontinuity in the phase*. Figure 1.4b shows how $\phi(\omega)$ jumps from $-\frac{\pi}{2}$ to $+\frac{\pi}{2}$ at $\omega = 0$. At other points the graph is linear, so we turn a blind eye to this discontinuity and say that the filter is still *linear phase*. It is the zero at $\omega = 0$ that causes this discontinuity in phase.

Invertibility and Noninvertibility

In any reasonable sense, the averaging and differencing filters are *not invertible*. The constant signal $\mathbf{x} = (\dots, 1, 1, 1, 1, 1, \dots)$ is wiped out by \mathbf{H}_1 . There cannot exist a filter \mathbf{H}_1^{-1} that recovers this \mathbf{x} from the zero vector. A linear operator cannot raise the dead — it only recovers $\mathbf{0}$ from $\mathbf{0}$. If a filter \mathbf{H} has an inverse, the only vector in its nullspace is that zero vector:

$$\text{If } \mathbf{H} \text{ is invertible and } \mathbf{H}\mathbf{x} = \mathbf{0}, \text{ then } \mathbf{H}^{-1}\mathbf{H}\mathbf{x} = \mathbf{0} \text{ and thus } \mathbf{x} = \mathbf{0}.$$

The frequency response of an invertible filter must have $H(\omega) \neq 0$ at all frequencies. Our filters are not invertible because $H_0(\pi) = 0$ and $H_1(0) = 0$.

The lowpass filter wipes out the alternating signal $\mathbf{x} = (\dots, 1, -1, 1, -1, \dots)$. We cannot recover this \mathbf{x} from zero output.

Note. When the inverse filter exists, it has frequency response $1/H(\omega)$. Multiplication will recover the input, because $H(\omega)/H(\omega) = 1$. We are safe as long as $H(\omega)$ is nonzero.

Suppose we attempt this inversion for the moving average. It is doomed to failure because $H_0(\pi) = 0$, but we can compute the filter coefficients that come from $1/H_0(\omega)$:

$$\frac{1}{\frac{1}{2} + \frac{1}{2}e^{-i\omega}} = 2(1 - e^{-i\omega} + e^{-2i\omega} - e^{-3i\omega} + \dots). \quad (1.17)$$

Those coefficients $2, -2, 2, -2, \dots$ go down the diagonals of the inverse filter matrix:

$$\mathbf{H}^{-1}\mathbf{H} = \begin{bmatrix} \cdot & & & & \\ -2 & 2 & & & \\ 2 & -2 & 2 & & \\ -2 & 2 & -2 & 2 & \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \end{bmatrix} = \mathbf{I}.$$

We seem to have found an inverse, but it is not a legal filter. This \mathbf{H}^{-1} is not stable. The bounded input $\mathbf{y} = (\dots, 1, -1, 1, -1, 1, \dots)$ would produce an unbounded output $\mathbf{H}^{-1}\mathbf{y}$. The series expansion in (1.17) breaks down completely at $\omega = \pi$, where $1 + e^{-i\pi} = 0$:

$$\frac{1}{0} = \frac{1}{\frac{1}{2} + \frac{1}{2}e^{-i\pi}} = 2(1 + 1 + 1 + 1 + \dots).$$

But also note: If the input signal contains no frequencies near $\omega = \pi$, then we could reconstruct \mathbf{x} from \mathbf{y} . The output \mathbf{y} would also have no high frequencies. Dividing by $H(\omega)$ would not involve dividing by zero.

The lowpass filter with coefficients $\mathbf{h}(0) = \frac{2}{3}$ and $\mathbf{h}(1) = \frac{1}{3}$ is safely invertible. Its frequency response $\frac{2}{3} + \frac{1}{3}e^{-i\omega}$ is never zero. The response at $\omega = 0$ is $\frac{2}{3} + \frac{1}{3} = 1$. The response at $\omega = \pi$ is $\frac{2}{3} - \frac{1}{3} = \frac{1}{3}$. The convolution with $\mathbf{h} = (\dots, 0, \frac{2}{3}, \frac{1}{3}, 0, \dots)$ can be undone by *deconvolution*. To find the deconvolution coefficients, which enter the inverse filter, divide by $H(\omega)$:

$$\frac{1}{\frac{2}{3} + \frac{1}{3}e^{-i\omega}} = \frac{3}{2} \left(1 - \frac{1}{2}e^{-i\omega} + \frac{1}{4}e^{-2i\omega} - \frac{1}{8}e^{-3i\omega} + \dots \right). \quad (1.18)$$

Those coefficients $\frac{3}{2}, -\frac{3}{4}, \frac{3}{8}, -\frac{3}{16}, \dots$ go on the diagonals of the inverse filter matrix. It is an IIR filter, with infinitely many coefficients.

The main point is analysis in the frequency domain. Dividing by $\frac{2}{3} + \frac{1}{3}e^{-i\omega}$ inverts the filter (and the constant-diagonal infinite matrix) for deconvolution.

Final remark. The inverse filter is very rarely FIR, because $1/H(\omega)$ is not a polynomial. To invert a finite length filter with a finite length filter, we need to go to a *filter bank*. That comes next.

Problem Set 1.3

1. Can a symmetric filter, with $\mathbf{h}(k) = \mathbf{h}(N - k)$, be a highpass filter?
2. Draw the graph for $|\omega| \leq \pi$ of “ideal” lowpass and highpass filters $H_0(\omega)$ and $H_1(\omega)$ with $H_0(\omega) + H_1(\omega) \equiv 1$. Why don’t we use these filters exclusively?
3. Which of the following filters are invertible? Find the inverse filters:

- (a) $\mathbf{h}(0) = \frac{2}{3}$ and $\mathbf{h}(1) = -\frac{1}{3}$
- (b) $\mathbf{h}(0) = 2$ and $\mathbf{h}(2) = 1$
- (c) $\mathbf{h}(n) = \frac{1}{n!}$ ($n = 0, 1, 2, \dots$)

4. Invent a highpass filter K with three or four taps (coefficients) that is better than the moving difference H_1 ; the goal is

$$|K(\omega)| < |H_1(\omega)| \text{ for } 0 < |\omega| < \frac{\pi}{2}$$

and

$$|H_1(\omega)| < |K(\omega)| < 1 \text{ for } \frac{\pi}{2} < |\omega| < \pi.$$

5. Find all input signals $\mathbf{x}(n)$ that are cut in half by the moving difference H_1 , so that $H_1\mathbf{x}(n) = \frac{1}{2}\mathbf{x}(n)$. Answer in the frequency domain.
6. **Important** If $H_0(\omega)$ is the response of a lowpass filter, what is the response $H_1(\omega)$ of a corresponding highpass filter? If $\mathbf{h}(0), \dots, \mathbf{h}(N)$ are the coefficients of H_0 , what are the coefficients of your H_1 ?
7. Let $H(z)$ be symmetric lowpass with $2K + 1$ coefficients. What is the phase of $G(z) = z^{-K} - H(z)$? Sketch the amplitude response if $H(\omega)$ is $\frac{1}{3}$ -band (near 1 for $|\omega| \leq \frac{\pi}{3}$). Is G lowpass or highpass?
8. A simple highpass filter design is $G(z) = H(-z)$. What is the relation between $\mathbf{g}(n)$ and $\mathbf{h}(n)$? With $H(\omega)$ as in Problem 7, sketch $G(\omega)$.
9. If $G(z) = H(-z^{-1})$ find $\mathbf{g}(n)$ in terms of $\mathbf{h}(n)$. If $H(z)$ is highpass show that $G(z)$ is lowpass.

1.4 Filter Bank = Lowpass and Highpass

Separately, the lowpass and highpass filters are not invertible. H_0 removes the highest frequency $\omega = \pi$, and H_1 removes the lowest frequency $\omega = 0$. Together, these filters do something very desirable. *They separate the signal into frequency bands*. The filtered output $H_0\mathbf{x}$ is weighted towards low frequencies, and $H_1\mathbf{x}$ is in some way the complement. The cutoff is not sharp, because these filters are so crude. But they make up the beginning of a *filter bank*.

One difficulty: the signal length has doubled. If the input \mathbf{x} is nonzero over a time T , so are the outputs from both filters. If the first nonzero input is $\mathbf{x}(0)$ and the last is $\mathbf{x}(T)$, then $H_0\mathbf{x}$ and $H_1\mathbf{x}$ end at time $T + 1$ (because of the final average and difference). One extra component is no problem, but doubling the storage by keeping two full-length outputs is unacceptable.

The solution is to downsample (or decimate).

Downsampling

We can keep *half* of $\mathbf{H}_0\mathbf{x}$ and $\mathbf{H}_1\mathbf{x}$, and still recover \mathbf{x} . This is an essential part of the filter bank, to *downsample* the outputs from the separate filters. We shall save only the *even-numbered components* of the two outputs. The odd-numbered components are removed:

$$(\downarrow 2)\mathbf{y} = (\dots, \mathbf{y}(-4), \mathbf{y}(-2), \mathbf{y}(0), \mathbf{y}(2), \mathbf{y}(4), \dots). \quad (1.19)$$

The symbol $\downarrow 2$ indicates downsampling or decimation. This is a linear operation (of course). But it is not time-invariant and it is not invertible. We study it closely in Chapter 3, and already this first filter bank indicates how it works in practice.

One note about normalization. To compensate for losing half the components in $(\downarrow 2)$, we multiply the surviving $\mathbf{y}(2n)$ by $\sqrt{2}$. This normalizing factor (explained below) is usually included with the filter bank, so that

$$\begin{aligned} \text{lowpass: } & H_0(\omega) \text{ changes to } C(\omega) = \sqrt{2} H_0(\omega) \\ \text{highpass: } & H_1(\omega) \text{ changes to } D(\omega) = \sqrt{2} H_1(\omega). \end{aligned}$$

In the averaging filter, the coefficients increase to $\frac{\sqrt{2}}{2}$. We keep the notation $\mathbf{h}(0), \mathbf{h}(1)$ for the original coefficients, and we introduce $\mathbf{c}(0), \mathbf{c}(1)$ for the new (renormalized) coefficients of \mathbf{C} :

$$\mathbf{c}(0) = \mathbf{c}(1) = \frac{\sqrt{2}}{2} \quad (\text{which is } \frac{1}{\sqrt{2}}).$$

Similarly the highpass coefficients $\frac{1}{2}$ and $-\frac{1}{2}$ are multiplied by $\sqrt{2}$, in \mathbf{D} :

$$\mathbf{d}(0) = \frac{\sqrt{2}}{2} = \frac{1}{\sqrt{2}} \quad \text{and} \quad \mathbf{d}(1) = -\frac{\sqrt{2}}{2} = -\frac{1}{\sqrt{2}}.$$

Systematically in this book, we will write \mathbf{C} and \mathbf{D} for $\sqrt{2}\mathbf{H}_0$ and $\sqrt{2}\mathbf{H}_1$. The response at frequency $\omega = 0$ is $C = \sqrt{2}$ rather than $H_0 = 1$.

Decimation Filters in the Time Domain

Downsampling follows the filter \mathbf{C} , in operating on \mathbf{x} . These two steps, filtering and decimation, can be done with *one matrix* \mathbf{L} . Decimation removes the odd-numbered components. To obtain $(\downarrow 2)\mathbf{Cx}$ in one step, *remove the odd-numbered rows of the filter matrix \mathbf{C}* . The combination of filtering by \mathbf{C} and decimation by $(\downarrow 2)$ is represented by a *rectangular matrix* \mathbf{L} that no longer has constant diagonals. It has 1×2 blocks:

$$\mathbf{L} = (\downarrow 2)\mathbf{C} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & & \\ & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \\ & & & \ddots & \ddots \end{bmatrix}.$$

The entries are $\mathbf{c}(0)$ and $\mathbf{c}(1)$ but half the rows of \mathbf{C} have disappeared. Similarly the decimated highpass filter is represented by a rectangular matrix $\mathbf{B} = (\downarrow 2)\mathbf{D}$. Removing half the rows of \mathbf{D} leaves the matrix \mathbf{B} with a *double-shift*:

$$\mathbf{B} = (\downarrow 2)\mathbf{D} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & & \\ & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \\ & & & \ddots & \ddots \end{bmatrix}.$$

When the lowpass \mathbf{L} and the highpass \mathbf{B} go into one matrix, you will see why the normalization by $\sqrt{2}$ is desirable. The rectangular \mathbf{L} and \mathbf{B} fit into a square matrix:

$$\begin{bmatrix} (\downarrow 2)\mathbf{C} \\ (\downarrow 2)\mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{L} \\ \mathbf{B} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ -1 & 1 & -1 & 1 & \\ & & & \ddots & \ddots \\ & & & & \ddots \end{bmatrix}.$$

This matrix represents the whole analysis bank. It executes the lowpass channel and the highpass channel (both decimated). All rows are unit vectors (because of the division by $\sqrt{2}$). Those row vectors are mutually orthogonal. At the same time, the columns are also *orthogonal unit vectors*.

The combined square matrix is invertible. *The inverse is the transpose*:

$$\begin{bmatrix} \mathbf{L} \\ \mathbf{B} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{L}^T & \mathbf{B}^T \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & & & \\ 1 & 1 & & & \\ & 1 & -1 & & \\ & 1 & 1 & & \\ & & & \ddots & \ddots \\ & & & & \ddots \end{bmatrix}.$$

Multiplying in either order yields the identity matrix. *The second matrix $[\mathbf{L}^T \quad \mathbf{B}^T]$ represents the synthesis bank*. This is an **orthogonal filter bank**, because *inverse = transpose*. We pause to summarize what you have seen in this example.

The channels $\mathbf{L} = (\downarrow 2)\mathbf{C}$ and $\mathbf{B} = (\downarrow 2)\mathbf{D}$ of an orthogonal filter bank are represented in the time domain by a combined orthogonal matrix:

$$[\mathbf{L}^T \quad \mathbf{B}^T] \begin{bmatrix} \mathbf{L} \\ \mathbf{B} \end{bmatrix} = \mathbf{L}^T \mathbf{L} + \mathbf{B}^T \mathbf{B} = \mathbf{I}.$$

The synthesis bank is the transpose of the analysis bank. When one follows the other we have perfect reconstruction. For causality we add a delay.

That summary is a foretaste of later chapters. We will construct longer and better filters, but the underlying problem will be close to this one. There is an analysis bank and a synthesis bank. When they are transposes as well as inverses, the whole filter bank is called **orthogonal**. When they are inverses but not necessarily transposes, the filter bank is **biorthogonal**.

In the biorthogonal case, the coefficients in the synthesis bank are different from $\mathbf{c}(n)$ and $\mathbf{d}(n)$ in the analysis bank. Our **Haar filter bank** has orthogonal filters. This chapter pursues it further, all the way to Haar wavelets.

Block Form of a Filter Bank

The best way to represent a two-channel filter bank is by a block diagram (Figure 1.5). The input is a vector \mathbf{x} . The blocks are linear operators and the output is two half-length vectors: $\mathbf{Lx} = (\downarrow 2)\mathbf{Cx}$ and $\mathbf{Bx} = (\downarrow 2)\mathbf{Dx}$.

For this special filter bank, we want to display all vectors in detail. Later we could supply only the essential information: the coefficients $\mathbf{c}(n)$ and $\mathbf{d}(n)$. There will be several ways to display those coefficients — the filter bank form, the modulation form, and the polyphase form. We

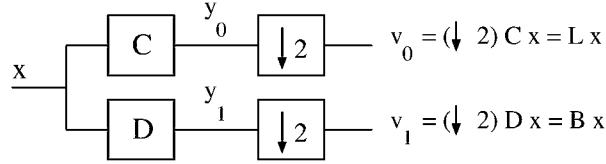


Figure 1.5: Schematic of the analysis half of a two-channel filter bank.

always need the coefficients! Here we use the direct filter bank form, and write the components of each output vector:

$$\mathbf{Lx} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{x}(0) & + & \mathbf{x}(-1) \\ \mathbf{x}(2) & + & \mathbf{x}(1) \\ \mathbf{x}(4) & + & \mathbf{x}(3) \end{bmatrix} \quad \mathbf{Bx} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{x}(0) & - & \mathbf{x}(-1) \\ \mathbf{x}(2) & - & \mathbf{x}(1) \\ \mathbf{x}(4) & - & \mathbf{x}(3) \end{bmatrix}.$$

The odd-numbered components of \mathbf{Cx} involved $\mathbf{x}(1) + \mathbf{x}(0)$ and $\mathbf{x}(3) + \mathbf{x}(2)$. Those are gone in \mathbf{Lx} . Similarly the odd-numbered components of \mathbf{Dx} are gone in \mathbf{Bx} . We are left with two half-length signals. Nevertheless we have enough information to recover the full-length input \mathbf{x} —and you see how.

To recover the zeroth component, add $\mathbf{x}(0) + \mathbf{x}(-1)$ to the difference $\mathbf{x}(0) - \mathbf{x}(-1)$. That gives $2\mathbf{x}(0)$. Since our sums and differences are already divided by $\sqrt{2}$ in the analysis bank, we need another $\sqrt{2}$ in synthesis:

$$\mathbf{x}(0) = \frac{1}{\sqrt{2}} \left(\frac{\mathbf{x}(0) + \mathbf{x}(-1)}{\sqrt{2}} + \frac{\mathbf{x}(0) - \mathbf{x}(-1)}{\sqrt{2}} \right). \quad (1.20)$$

For $\mathbf{x}(-1)$, use the same components of \mathbf{Lx} and \mathbf{Bx} , and *subtract*:

$$\mathbf{x}(-1) = \frac{1}{\sqrt{2}} \left(\frac{\mathbf{x}(0) + \mathbf{x}(-1)}{\sqrt{2}} - \frac{\mathbf{x}(0) - \mathbf{x}(-1)}{\sqrt{2}} \right).$$

Addition and subtraction are simple for this example, but the synthesis steps must be organized in a way that extends to other examples. At the end of the reconstruction, we want to reach these two vectors \mathbf{w}_0 and \mathbf{w}_1 :

$$\mathbf{w}_0 = \frac{1}{2} \begin{bmatrix} \mathbf{x}(0) & + & \mathbf{x}(-1) \\ \mathbf{x}(0) & + & \mathbf{x}(-1) \\ \mathbf{x}(2) & + & \mathbf{x}(1) \\ \mathbf{x}(2) & + & \mathbf{x}(1) \\ \vdots & & \end{bmatrix} \quad \text{and} \quad \mathbf{w}_1 = \frac{1}{2} \begin{bmatrix} -\mathbf{x}(0) & + & \mathbf{x}(-1) \\ \mathbf{x}(0) & - & \mathbf{x}(-1) \\ -\mathbf{x}(2) & + & \mathbf{x}(1) \\ \mathbf{x}(2) & - & \mathbf{x}(1) \\ \vdots & & \end{bmatrix}. \quad (1.21)$$

Notice how the signs in \mathbf{w}_1 are adjusted so that $\mathbf{w}_0 + \mathbf{w}_1$ recovers the input vector. Actually we are getting $\mathbf{x}(n-1)$ instead of $\mathbf{x}(n)$. *The total effect of the whole filter bank is a delay*. The input is $\mathbf{x}(n)$ and the output is the delayed $\mathbf{x}(n-1)$. The sum $\mathbf{w}_0 + \mathbf{w}_1$ is almost, but not quite, the original \mathbf{x} .

A delay is built in because all filters are *causal*. We analyzed \mathbf{x} into low and high frequencies. Now we synthesize to reach \mathbf{w}_0 and \mathbf{w}_1 and recover \mathbf{x} .

The Synthesis Bank

A well-organized synthesis bank is the inverse of the analysis bank. The analysis bank had two steps, *filtering* and *downsampling*. The synthesis bank also has two steps, *upsampling and filtering*. Notice how the order is reversed—as it always is for inverses.

The first step is to bring back full-length vectors. The downsampling operation ($\downarrow 2$) is not invertible, but *upsampling* is as close as we can come. The odd-numbered components are *returned as zeros by upsampling*. Applied to a half-length vector v , upsampling inserts zeros:

$$\text{Upsampling} \quad (\uparrow 2) \begin{bmatrix} \cdot \\ v(0) \\ v(1) \\ v(2) \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot \\ v(0) \\ 0 \\ v(1) \\ 0 \\ v(2) \\ \cdot \end{bmatrix}. \quad (1.22)$$

Upsampling is denoted by ($\uparrow 2$). To understand it, look at the result of downsampling to get $v = (\downarrow 2)y$ and upsampling to get $u = (\uparrow 2)(\downarrow 2)y$:

$$y = \begin{bmatrix} \cdot \\ y(0) \\ y(1) \\ y(2) \\ y(3) \\ y(4) \\ \cdot \end{bmatrix} \quad (\downarrow 2)y = \begin{bmatrix} \cdot \\ y(0) \\ y(2) \\ y(4) \\ \cdot \end{bmatrix} \quad (\uparrow 2)(\downarrow 2)y = \begin{bmatrix} \cdot \\ y(0) \\ 0 \\ y(2) \\ 0 \\ y(4) \\ \cdot \end{bmatrix}. \quad (1.23)$$

The odd-numbered components of y are replaced by zeros. We will see that ($\uparrow 2$) is the *transpose* of ($\downarrow 2$). Fortunately, transposes come in reverse order exactly as inverses do. So synthesis can be the *transpose* of analysis—apart from our ever-present delay.

Small note: Also ($\uparrow 2$) is a *right-inverse* of ($\downarrow 2$). If we put in zeros and remove them, we recover y . Thus $(\downarrow 2)(\uparrow 2) = I$. The order ($\uparrow 2$)($\downarrow 2$) that we actually use is displayed above, and it inserts zeros. Properly speaking, ($\uparrow 2$) is the “pseudoinverse” of ($\downarrow 2$)—which has no inverse.

The vectors reached by upsampling have zeros in their odd components:

$$\mathbf{u}_0 = \frac{1}{\sqrt{2}} \begin{bmatrix} x(0) + x(-1) \\ 0 \\ x(2) + x(1) \\ 0 \\ x(4) + x(3) \\ \cdot \end{bmatrix} \quad \text{and} \quad \mathbf{u}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} x(0) - x(-1) \\ 0 \\ x(2) - x(1) \\ 0 \\ x(4) - x(3) \\ \cdot \end{bmatrix}. \quad (1.24)$$

The second step in the synthesis bank, after upsampling, is *filtering*. The two vectors \mathbf{u}_0 and \mathbf{u}_1 are the inputs to the two filters. The vectors w_0 and w_1 are the desired outputs. Schematically, the structure of the synthesis bank is in Figure 1.6.

Normally we would construct the synthesis filters F and G based on the analysis filters C and D . That will be our procedure in the rest of the book. For this example, when we know the

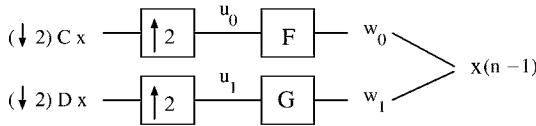


Figure 1.6: The synthesis half of a filter bank: upsample, filter, and add.

desired outputs, we proceed more directly. The filter \mathbf{F} that produces w_0 from u_0 is an addition filter:

$$\mathbf{F} \text{ filters } \frac{1}{\sqrt{2}} \begin{bmatrix} x(0) + x(-1) \\ 0 \\ x(2) + x(1) \\ 0 \\ \vdots \end{bmatrix} \text{ to give } \frac{1}{2} \begin{bmatrix} x(0) + x(-1) \\ x(0) + x(-1) \\ x(2) + x(1) \\ x(2) + x(1) \\ \vdots \end{bmatrix} = w_0.$$

This is the output we want. It comes from a time-invariant causal filter \mathbf{F} . There is no separate treatment of even and odd components! When the input to \mathbf{F} has components $u(n)$, the output has components

$$\mathbf{F}u(n) = \frac{1}{\sqrt{2}}(u(n) + u(n-1)). \quad (1.25)$$

The filter coefficients are $\mathbf{G}(0) = f(1) = \frac{1}{\sqrt{2}}$.

The second synthesis filter \mathbf{G} is a subtraction filter. Its coefficients are $\frac{-1}{\sqrt{2}}$ and $\frac{1}{\sqrt{2}}$. For an arbitrary input vector u , the output has components $\frac{1}{\sqrt{2}}(-u(n) + u(n-1))$. Notice especially how \mathbf{G} acts on $u_1 = (\uparrow 2)(\downarrow 2)\mathbf{D}x$:

$$\mathbf{G} \text{ filters } \frac{1}{\sqrt{2}} \begin{bmatrix} x(0) - x(-1) \\ 0 \\ x(2) - x(1) \\ 0 \\ \vdots \end{bmatrix} \text{ to give } \frac{1}{2} \begin{bmatrix} -x(0) & + & x(-1) \\ x(0) & - & x(-1) \\ -x(2) & + & x(1) \\ x(2) & - & x(1) \\ \vdots \end{bmatrix} = w_1.$$

The filter gives the right result, again without treating even and odd components differently. We caution that the highpass coefficients are in the order $-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$. When you transpose \mathbf{D} , the order of coefficients is reversed. Then a delay makes \mathbf{G} a causal filter (and the same for \mathbf{F}).

The only other caution concerns the factors $\frac{1}{\sqrt{2}}$. They are present in all four filters. A less perfect symmetry would have $\frac{1}{2}$ in one bank and 1 in the other bank. This is exactly like the two-point discrete Fourier transform, where we often allow $\frac{1}{2}$ and 1 in the matrix and its inverse. The orthogonal matrix has $\frac{1}{\sqrt{2}}$ in both:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^{-1} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{inverse matrices with 1 and } \frac{1}{2}$$

$$\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}^{-1} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \quad \text{orthogonal matrix with } 1/\sqrt{2}.$$

An orthogonal (or unitary) matrix has orthogonal rows and orthogonal columns normalized to be unit vectors. *The inverse is the transpose* (or conjugate transpose). It is an accident for this

example that the matrix is real and symmetric. You can't see that it was transposed and conjugated.

Important. The connection between this Haar filter bank and the 2-point DFT is no accident. The simplest M -band filter bank comes from an M -point DFT. It is called a *uniform DFT filter bank*. We are seeing the case $M = 2$, written as an ordinary filter bank (and made causal by a delay).

We summarize this section with a schematic of the whole filter bank (Figure 1.7).

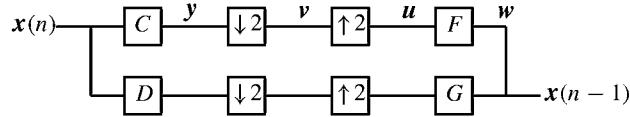


Figure 1.7: The analysis bank followed by the synthesis bank.

This allows us to indicate the symbols y, u, v, w for the outputs at the four stages. For a two-channel bank, we mark those vectors in the lowpass and highpass channels by subscripts 0 and 1. When there are M channels, we need subscripts $0, 1, \dots, M - 1$.

From the filter bank, the next step is to wavelets.

Problem Set 1.4

1. Write down the matrix $(\downarrow 2)$ that executes downsampling: $(\downarrow 2)x(n) = x(2n)$.
2. Write down the transpose matrix $(\uparrow 2) = (\downarrow 2)^T$. Multiply the matrices $(\uparrow 2)$ $(\downarrow 2)$ and $(\downarrow 2)(\uparrow 2)$. Describe the output from $(\uparrow 2)y(n)$.
3. Describe the output from $(\downarrow 2)^2x(n)$ and $(\uparrow 2)^2y(n)$.
4. Send the signal with $x(0) = x(1) = x(2) = 1$ through the whole filter bank, and give the output at every step.
5. Put each row $\begin{bmatrix} -1 & 1 \end{bmatrix}/\sqrt{2}$ of $B = (\downarrow 2)\mathbf{D}$ after the row $\begin{bmatrix} 1 & 1 \end{bmatrix}/\sqrt{2}$ of $L = (\downarrow 2)\mathbf{C}$. In this order we see a *block transform*. What is the inverse transform?
6. Show how to delay an anticausal (= upper triangular) filter matrix so it becomes causal. If the coefficients $h(0), \dots, h(N)$ are on the diagonals of the anticausal matrix, what are the diagonals after the delay? Give the two frequency responses, anticausal and causal.
7. The 4-channel bank analogous to Haar is based on the 4-point DFT matrix F_4 . Find F_4 and the four analysis filters and the outputs after downsampling by $(\downarrow 4)$.
8. Suppose a matrix has the property that $Q^T Q = I$. Show that the columns of Q are mutually orthogonal unit vectors. Does it follow that $Q Q^T = I$?
9. In a transmultiplexer, the synthesis bank comes *before* the analysis bank. Compute LL^T and LB^T and BB^T to verify that the Haar transmultiplexer still gives perfect reconstruction:

$$\begin{bmatrix} L \\ B \end{bmatrix} \begin{bmatrix} L^T & B^T \end{bmatrix} = \begin{bmatrix} LL^T & LB^T \\ BL^T & BB^T \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}.$$

10. Compute the subband outputs $v_0(n)$ and $v_1(n)$ for the average-difference filter bank with input $x(n) = 0, 1, -1, 2, 5, 1, 7, 0$. Reconstruct the signal by feeding $v_k(n)$ into the synthesis bank in Figure 1.6. Verify that the output is $x(n-1)$.
11. If $H_0(z) = 1$ and $H_1(z) = z^{-1}$ (no filtering) write the entries of $\begin{bmatrix} L \\ B \end{bmatrix} \begin{bmatrix} L^T & B^T \end{bmatrix} = I$.

1.5 Scaling Function and Wavelets

Corresponding to the lowpass filter, with $\mathbf{h}(0) = \frac{1}{2}$ and $\mathbf{h}(1) = \frac{1}{2}$, there is a continuous-time scaling function $\phi(t)$. Corresponding to the highpass filter, with coefficients $\frac{1}{2}$ and $-\frac{1}{2}$, there is a wavelet $w(t)$. We now describe the *dilation equation* that produces $\phi(t)$ and the *wavelet equation* for $w(t)$.

You will see how the filter coefficients (the \mathbf{c} 's and \mathbf{d} 's) enter these equations. Two time scales also appear. This joint appearance of t and $2t$ is the novelty of the dilation equation. It is also the source of difficulty! We have a “two-scale difference equation”. Later we develop the background of these equations and a general method for solving them. Here we quickly recognize the box function as $\phi(t)$. Then we construct the wavelet $w(t)$ and use it.

The dilation equation for the scaling function $\phi(t)$ is

$$\phi(t) = \sqrt{2} \sum_{k=0}^N \mathbf{c}(k) \phi(2t - k). \quad (1.26)$$

In terms of the original lowpass coefficients $\mathbf{h}(k)$, the extra factor is 2:

$$\phi(t) = 2 \sum_{k=0}^N \mathbf{h}(k) \phi(2t - k). \quad (1.27)$$

This involves a function $\phi(t)$ in continuous time, and a set of coefficients $\mathbf{c}(k)$ or $\mathbf{h}(k)$ from discrete time. The presence of t and $2t$ is the key. Without the 2 we would have an ordinary constant-coefficient equation (look for exponential solutions $e^{\lambda t}$). With two time scales, there are major changes:

1. There may or may not be a solution $\phi(t)$.
2. The solution is zero outside the interval $0 \leq t < N$.
3. The solution seldom has an elementary formula.
4. The solution is not likely to be a smooth function.

Formally, we can find an expression for the Fourier transform of $\phi(t)$. It is an infinite product (Section 6.4). The inverse Fourier transform yields $\phi(t)$ as a “distribution”—not necessarily continuous, possibly involving delta functions. (Those are impulses. Their integrals are jumps. More cautious people call them steps.) In our Haar example, the solution $\phi(t)$ lies just outside the class of continuous functions—it has a jump.

For the coefficients $2\mathbf{h}(0) = 1$ and $2\mathbf{h}(1) = 1$, the dilation equation is

$$\phi(t) = \phi(2t) + \phi(2t - 1). \quad (1.28)$$

The graph of $\phi(t)$ is compressed by 2, to give the graph of $\phi(2t)$. When that is shifted to the right by $\frac{1}{2}$, it becomes the graph of $\phi(2t - 1)$. We ask the two compressed graphs to combine into the original graph. Figure 1.8 shows that this occurs when $\phi(t)$ is the *box function*:

$$\phi(t) = \begin{cases} 1 & \text{for } 0 \leq t < 1 \\ 0 & \text{otherwise.} \end{cases}$$

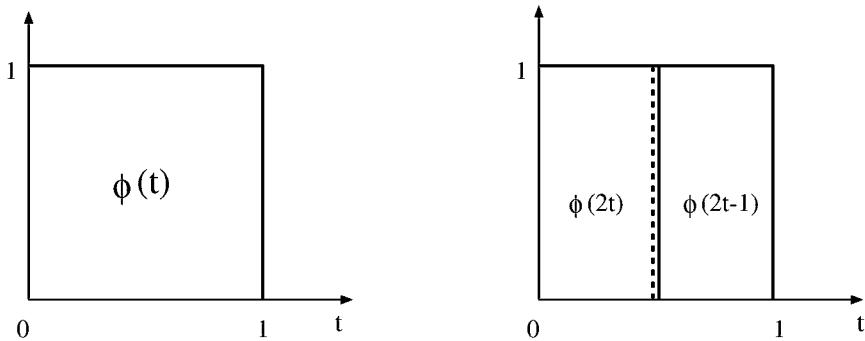


Figure 1.8: The box function with dilation and translation.

The graphs of $\phi(2t)$ and $\phi(2t - 1)$ are half-size boxes. Their sum is the full-size box $\phi(t)$.

As planned, we wrote down the solution rather than deriving it. The dilation equation is linear, so any multiple of the box function is also a solution. It is convenient to normalize so that *the integral of $\phi(t)$ from $-\infty$ to ∞ equals one*. Note that a solution $\phi(t)$ covering unit area is only possible when the coefficients in the dilation equation add up to 2.

Theorem 1.1 If $\int_{-\infty}^{\infty} \phi(t) dt = 1$ then $2\mathbf{h}(0) + 2\mathbf{h}(1) + \dots + 2\mathbf{h}(N) = 2$.

Proof: The graph of $\phi(2t)$ and every $\phi(2t - k)$ is compressed to area $\frac{1}{2}$:

$$2 \int_{-\infty}^{\infty} \phi(2t - k) dt = \int_{-\infty}^{\infty} \phi(u) du = 1 \quad (\text{set } u = 2t - k). \quad (1.29)$$

So integrating both sides of the dilation equation $\phi(t) = 2 \sum \mathbf{h}(k)\phi(2t - k)$ gives $1 = \mathbf{h}(0) + \mathbf{h}(1) + \dots + \mathbf{h}(N)$. This is our lowpass filter convention.

Important. For the filter, then the filter bank, and finally the dilation equation, *the normalization is different*. This is clear from the actual numbers in our lowpass filter. The sum of coefficients is 1 or $\sqrt{2}$ or 2:

- \mathbf{h} : $\frac{1}{2}$ and $\frac{1}{2}$ in the single filter, adding to 1
- \mathbf{c} : $\frac{1}{\sqrt{2}}$ and $\frac{1}{\sqrt{2}}$ in the filter bank, adding to $\sqrt{2}$
- $2\mathbf{h}$: 1 and 1 in the dilation equation, adding to 2.

The single lowpass filter has sum $H(0) = 1$. That preserves the zero frequency DC term: output $Y(0) = \text{input } X(0)$. The filter bank has a factor $\sqrt{2}$ to account for the downsampling step. There are only half as many components and half as many terms in the energy, when $\sum (y(n))^2$ is replaced by $\sum (y(2n))^2$. To compensate we must multiply $y(2n)$ by $\sqrt{2}$. That gives the normalization $C(0) = \sqrt{2}$ in place of $H(0) = 1$.

For the dilation equation, we have just seen why rescaling the time requires renormalizing the coefficients by 2 — to preserve area.

There are certainly filters in which $H(0)$ is not exactly 1. If $H(\omega)$ stays very near 1 over an interval around $\omega = 0$, this still deserves the name “lowpass filter”. *Such filters do not lead to*

wavelets! The requirements for wavelets are very strict, at $\omega = 0$ and $\omega = \pi$. Only a subset of special filters can pass the test, which starts with $H(0) = 1$ and $H(\pi) = 0$.

The box function is like the averaging filter—it smoothes the input. Convolution with the box function gives a moving average in continuous time, just as the filter coefficients $\mathbf{h} = \frac{1}{2}, \frac{1}{2}$ did in discrete time:

$$\mathbf{h} * (\dots, x(0), x(1), \dots) = \left(\dots, \frac{x(0) + x(-1)}{2}, \frac{x(1) + x(0)}{2}, \dots \right)$$

$$\phi(t) * x(t) = \int_{t-1}^t x(s) ds = \text{average over moving interval.} \quad (1.30)$$

There is a similar convolution with $w(t)$. Instead of picking up the smooth low frequency part of the function, the wavelet will lead to the high-frequency details. The coefficients for the Haar wavelet are 1 and -1.

The Wavelet Equation

The equation for the wavelet involves the highpass coefficients $\mathbf{d}(k)$. It is a direct equation that gives $w(t)$ immediately and explicitly from $\phi(t)$:

$$w(t) = \sqrt{2} \sum \mathbf{d}(k) \phi(2t - k). \quad (1.31)$$

In terms of the original coefficients $\mathbf{h}_1(k)$, the factor $\sqrt{2}$ becomes 2:

$$\text{Wavelet equation} \quad w(t) = 2 \sum \mathbf{h}_1(k) \phi(2t - k). \quad (1.32)$$

In our example, $\phi(t)$ is a box function and its dilations $\phi(2t - k)$ are half-boxes. Then the wavelet is a *difference of half-boxes*:

$$w(t) = \phi(2t) - \phi(2t - 1). \quad (1.33)$$

Explicitly, $w(t) = 1$ for $0 \leq t < \frac{1}{2}$ and $w(t) = -1$ for $\frac{1}{2} \leq t < 1$. This is the **Haar wavelet**. Its graph is in Figure 1.9 along with the graphs of $w(2t)$ and $w(2t - 1)$. Those are wavelets at scale 2t; their graphs are compressed and shifted. They join the original $w(t)$, and all its other dilations and translations, in the *wavelet basis*.

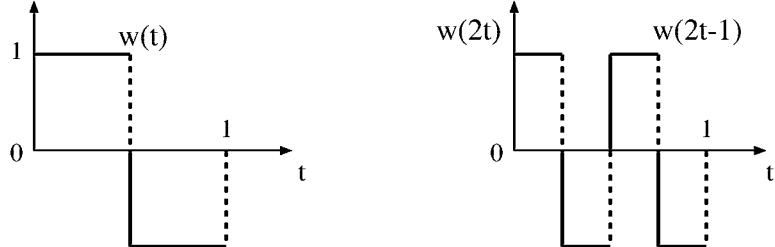


Figure 1.9: The Haar wavelet $w(t)$; the rescaled wavelets $w(2t)$ and $w(2t - 1)$.

It goes without saying that Alfred Haar did not call his function a “wavelet”. He was writing in 1910 about this particular function; all other wavelets came later. That name emerged from

the literature on geophysics, by a route through France. The word *onde* led to *ondelette*. In translation, the word wave led to wavelet. **A wavelet is a small wave.** In Haar's case it happened to be a square wave.

Comment. The square wave $w(t)$ has compact support. It comes from an FIR filter, with finite length. We use the word "support" for the closed interval in continuous time, here $[0, 1]$, outside which $w(t)$ is zero. When we close the set where it is nonzero, to include the jump location $t = \frac{1}{2}$ as well as the endpoints, we have found the support of the function $w(t)$. The words "compact support" mean that this closed set is bounded. The wavelet is zero outside a bounded interval: **compact support corresponds to FIR.**

Wavelets need not have compact support! They can come from IIR filters instead of FIR filters. Historically, since the connection of wavelets to filters was not immediately recognized, the wavelets after Haar were constructed in other ways (with considerable difficulty). They oscillated above and below zero along the whole line $\infty < t < \infty$, decaying as $|t| \rightarrow \infty$. This still qualifies as a wavelet. It is a localized pulse that decreases to zero and *has integral zero*. Ingrid Daubechies showed in 1988 that compact support was possible for other wavelets than Haar's.

We will tell that story more completely in Chapter 6. From Haar onwards, one property that most designers hoped for was **orthogonality**. This means: $w(t)$ is orthogonal to all its dilations and translations. The wavelet basis, containing all these functions $w(2^j t - k)$, is an *orthogonal basis*. For the first few Haar wavelets that is easy to verify:

$$\begin{aligned} \text{inner product} &= \int_{-\infty}^{\infty} w(t)w(2t) dt = 0 \\ \text{inner product} &= \int_{-\infty}^{\infty} w(t)w(2t - 1) dt = 0 \\ \text{inner product} &= \int_{-\infty}^{\infty} w(2t)w(2t - 1) dt = 0. \end{aligned}$$

In the first integral, $w(t) = 1$ in Figure 1.9 where $w(2t)$ is positive and then negative. The integral is zero. Similarly $w(t) = -1$ on the second half-interval where $w(2t - 1)$ is plus and minus. The second integral is therefore zero. The third integral vanishes for a different reason—the functions $w(2t)$ and $w(2t - 1)$ do not overlap. One is zero where the other is nonzero. So the product $w(2t)w(2t - 1)$ is zero everywhere.

The pattern continues for all translations by k and dilations by 2^j . Haar wavelets at the same scaling level (same j) do not overlap. They are orthogonal in the strictest way. When Haar wavelets at different levels j and J do overlap, the coarse one is constant where the fine one goes up and down. All integrals are zero, giving an orthogonal basis $w_{jk}(t) = w(2^j t - k)$:

$$\text{inner product} = \int_{-\infty}^{\infty} w(2^j t - k) w(2^J t - K) dt = 0. \quad (1.34)$$

A perfect basis is not only orthogonal but orthonormal. The functions have length 1. Like a unit vector, the inner product $\langle w(t), w(t) \rangle$ is normalized to 1:

$$\text{length squared} = \int_{-\infty}^{\infty} (w(t))^2 dt = 1.$$

This is true for the Haar wavelet. To make it true for the dilations of that wavelet, we multiply by $2^{j/2}$ —otherwise the compressed graphs cover less area. The same factor will apply to other wavelets, and we record it now:

Theorem 1.2 *The rescaled Haar wavelets $w_{jk}(t) = 2^{j/2}w(2^jt - k)$ form an orthonormal basis:*

$$\int_{-\infty}^{\infty} w_{jk}(t) w_{JK}(t) dt = \delta(j - J) \delta(k - K). \quad (1.35)$$

This Kronecker delta symbol equals zero except when $j = J$ and $k = K$. In that case the integral of $(w_{JK}(t))^2$ equals one. We need two indices j and k because there are two operations (dilation and translation) in constructing the basis.

Important note. For these Haar wavelets, orthogonality was verified by direct integration. For future wavelets, this integration is not desirable and not possible. We will not have elementary formulas for $\phi(t)$ and $w(t)$. Instead, we will know the coefficients $c(k)$ in the dilation equation and $d(k)$ in the wavelet equation. *All information about $\phi(t)$ and $w(t)$ —their support interval, their orthogonality, their smoothness, and their vanishing moments—will be determined by and from the c 's and d 's.*

Second note. We have not said *which space of functions* has the wavelets $w_{jk}(t)$ as a basis. Actually there are many choices. The space starts with *finite* combinations $g(t) = \sum b_{jk} w_{jk}(t)$. Those functions are piecewise constant on binary intervals—length $1/2^j$ and endpoints $m/2^j$. Other functions $f(t)$ are in the space if they are *limits of these piecewise constant* g 's:

$$\|f(t) - g_n(t)\| \rightarrow 0 \quad \text{for some sequence } g_n(t).$$

The choice of function space is decided by the choice of the norm $\|f - g_n\|$. A function $f(t)$ might be a limit of piecewise constants in the maximum norm but not in an integral norm, or vice versa. The most frequent choice is the L^2 norm, where the superscript 2 signals that we integrate the *square*:

$$\|f(t) - g(t)\| = \left(\int_{-\infty}^{\infty} |f(t) - g(t)|^2 dt \right)^{1/2}.$$

This choice of the L^2 norm is popular for four major reasons:

1. The norm is directly connected to the inner product:

$$\begin{aligned} \text{By definition} \quad \|f(t)\|^2 &= \langle f(t), f(t) \rangle \\ \text{By the Schwarz inequality} \quad \|f(t)\| \|g(t)\| &\geq |\langle f(t), g(t) \rangle|. \end{aligned}$$

2. Minimizing the L^2 norm leads to linear equations. This is familiar from ordinary least squares problems. The word “squares” introduces the L^2 norm. When the functions $g(t)$ are restricted to a subspace, the closest one to $f(t)$ is $g(t) = \text{projection of } f(t) \text{ onto the subspace}$. Projection leads to right angles and linear equations.

3. The Fourier transform *preserves the L^2 norm and the inner product* $\langle f, g \rangle$. This is the Parseval identity:

$$\|f\|^2 = \int_{-\infty}^{\infty} |f(t)|^2 dt = \int_{-\infty}^{\infty} |\widehat{f}(\omega)|^2 d\omega = \|\widehat{f}\|^2 \quad (1.36)$$

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(t) \overline{g(t)} dt = \int_{-\infty}^{\infty} \widehat{f}(\omega) \overline{\widehat{g}(\omega)} d\omega = \langle \widehat{f}, \widehat{g} \rangle. \quad (1.37)$$

The reader will know that irrelevant but necessary factors of 2π should enter these equations. They depend on how we define the Fourier transforms. Since $\widehat{f}(\omega)$ and $\widehat{g}(\omega)$ can be complex, we introduced absolute values in (1.36) and complex conjugates in (1.37). Of course (1.37) becomes (1.36) if $g(t) = f(t)$.

4. For an orthonormal basis (like the Haar wavelet basis), the L^2 norm $\|\sum b_{jk} w_{jk}(t)\|^2$ equals the sum of squares of the coefficients b_{jk} :

$$\begin{aligned}\int (\sum b_{jk} w_{jk}(t))^2 dt &= \sum \sum b_{jk} b_{JK} \int w_{jk}(t) w_{JK}(t) dt \\ &= \sum (b_{jk})^2.\end{aligned}$$

For all these reasons, and more, the L^2 norm is our choice. But we must mention that wavelets provide an unusually convenient basis for other norms and other function spaces. This makes them popular in functional analysis and harmonic analysis, which deal with the properties of functions. If we change the exponent from 2 to p , the space L^p contains all functions for which $(\int |f(t)|^p)^{1/p}$ is finite. This norm $\|f\|_p$ is not quite equal to $C (\sum |b_{jk}|^p)^{1/p}$, as it was for $p = 2$. But if the b_{jk} are wavelet coefficients, rather than Fourier coefficients, this sum lies between fixed bounds $A_p \|f\|_p$ and $B_p \|f\|_p$. Thus the absolute values $|b_{jk}|$ still indicate which functions have finite norm and belong to L^p .

In shorthand: The wavelets are an *unconditional basis* for L^p (no condition on the signs of the b_{jk}). The complex exponentials are conditional; phase information is needed on the b_{jk} .

We recognize that these last comments are “pure mathematics”. The reader is invited to start learning that language too—if desired and not already achieved. It is rewarding and not difficult. The *wavelet transform* that connects $f(t)$ to its wavelet coefficients b_{jk} is absolutely central—to theory and also to applications.

We turn to the practical problem: *How to compute the coefficients b_{jk} quickly?* This has a very good answer for wavelet transforms. There is a recursive Fast Wavelet Transform comparable in speed and stability to the FFT for Fourier transforms.

Problem Set 1.5

1. If $w(t)$ has unit norm, so that $\int (w(t))^2 dt = 1$, show that the function $w_{jk}(t) = 2^{j/2} w(2^j t - k)$ also has unit norm.
2. What combination of the half-boxes $\phi(2t)$ and $\phi(2t - 1)$ is closest in L^2 (least squares) to $f(t) = t^2$ for $0 \leq t \leq 1$?
3. The box function $\phi_s(t)$, shifted to the interval $[3, 4]$, is what combination of the functions $\phi_s(2t - k)$?
4. Show that the convolution of the box function with a continuous-time signal is the continuous average in (1.30). The convolution formula is

$$\phi(t) * x(t) = \int_{-\infty}^{\infty} \phi(t-s) x(s) ds.$$

5. Given a combination $a_0\phi(2t) + a_1\phi(2t - 1)$, express it as $A_0\phi(t) + B_0w(t)$. Then invert to find a_0 and a_1 from A_0 and B_0 .
6. What scaling function satisfies the three-scale equation $\phi(t) = \phi(3t) + \phi(3t - 1) + \phi(3t - 2)$? Find *two wavelets* that are combinations of $\phi(3t - k)$, orthogonal to $\phi(t)$ and to each other.

1.6 Wavelet Transforms by Multiresolution

The wavelet transform operates in continuous time (on functions) and in discrete time (on vectors). The input is $f(t)$ or $x(n)$. The output is the set of coefficients b_{jk} , which express the input in the wavelet basis. For functions and infinite signals, this basis is necessarily infinite. For finite length vectors with L components, there will be L basis vectors and L coefficients. The discrete wavelet transform, from L components of the signal to L wavelet coefficients, is expressed by an L by L matrix.

To begin, we derive formulas for the coefficients b_{jk} . In continuous time they involve integrals of $f(t)$ times $w_{jk}(t)$. In discrete time we are solving a linear system. The inverse transform involves the inverse matrix.

Then we show how the b_{jk} can be found recursively. Levels j and $j - 1$ are connected. This reorganizes the matrix multiplication. The discrete wavelet transform (DWT) becomes the *fast* wavelet transform (FWT). The central idea in this fast recursion is *multiresolution*.

First come the two directions, synthesis and analysis, for an orthonormal basis:

$$\begin{aligned} \text{Synthesis of a function: } f(t) &= \sum_{j,k} b_{jk} w_{jk}(t) \\ \text{Analysis of a function: } b_{JK} &= \int_{-\infty}^{\infty} f(t) w_{JK}(t) dt. \end{aligned} \quad (1.38)$$

In the matrix case, the wavelets are ordinary vectors. They go into the columns of the *wavelet matrix* S . To maintain the parallel with the continuous case, we use a double index jk for the column number. A single index would go from 1 to L , but the double index is more natural. Each wavelet vector has a position in time given by k and a position in frequency (better to say, in *scale*) given by j . The columns of the L by L matrix S are the discrete wavelets:

$$\text{Synthesis in discrete time: } x = Sb.$$

The rows of the L by L matrix A contain the “analyzing” wavelets:

$$\text{Analysis in discrete time: } b = Ax.$$

For Haar and all orthonormal wavelets, the columns of S are the same as the rows of A . Analysis and synthesis are related by $A = S^T$. In general they are related by $A = S^{-1}$.

The synthesis equation $x = Sb$ multiplies each coefficient b_{jk} by the basis vector in column jk of S , and adds. This is just matrix multiplication: Sb is a combination of the columns of S .

The analysis equation $Ax = ASb = b$ is completely parallel to the continuous formula $\int w_{jk}(t) f(t) dt = b_{jk}$. The left side is the inner product of x with each analyzing vector. The wavelet basis consists of *unit vectors*, so all formulas are simple. The basis is orthonormal. In the continuous case, $\int (w_{jk}(t))^2 dt = 1$. In the discrete case $S^T S = I$. Then $b = S^T x$ and no division by length squared is required.

Analysis is the *inverse* of synthesis. Why did we multiply by the *transpose matrix*, when we should have introduced the *inverse matrix*? For an orthonormal basis, the reason is fundamental: The inverse is the same as the transpose!

$$\text{Orthonormal columns: } S^T S = I \text{ means } S^{-1} = S^T.$$

When the basis is only orthogonal, and not normalized to unit vectors, $S^T S$ is diagonal. By using the word *basis*, we ensured that all these matrices are square. In the rectangular case S would

not have an inverse. There are too many columns to be independent; instead of a basis we have a *frame* (Section 2.6). Our formulas would give the “pseudoinverse” S^+ instead of S^{-1} .

Without orthogonality, the rows of $A = S^{-1}$ are *biorthogonal* to the columns of S :

$$(\text{row } i \text{ of } A) \cdot (\text{column } j \text{ of } S) = \delta(i - j).$$

Each row of S^{-1} is orthogonal to $L - 1$ columns of S . This is *biorthogonality*. The columns of S are the *synthesis basis*, and the rows of $A = S^{-1}$ are the *analysis basis*.

Tree-Structured Filter Bank

We move from transforms toward *fast* transforms. The wavelet basis and the Fourier basis have special properties, beyond orthogonality. The scales j and $j - 1$ are closely related, just as the frequencies ω and $\omega/2$ are closely related. By taking advantage of these relations, the multiplications by S and S^T can be reorganized. We will explain the special properties, and then derive fast transforms (starting in this section with Haar wavelets).

In the Fourier case, the matrices become F and F^T — or actually \bar{F}^T , since the Fourier vectors are complex exponentials. These are the L -point DFT matrix and its inverse. The Fast Fourier Transform is summarized in Section 8.1 (on periodic problems). You will see that the FWT is asymptotically faster than the FFT, requiring only $O(L)$ steps instead of $O(L \ln L)$. (The Walsh basis, which is not local, brings back $L \ln L$. The components are ± 1 , with no zeros.) All these transforms are so important and successful that we do not overemphasize the comparison. Our goal is to understand wavelets and Fourier both.

The recursive nature of wavelets is clearest when we construct a tree of filter banks (Figure 1.10). The highpass filter D computes differences of the input. The downsampling step symbolized by $(\downarrow 2)$ keeps the even-numbered differences $(x(2k) - x(2k - 1)) / \sqrt{2}$. These are final outputs because they are not transformed again. They are at the end of their branch, in this “logarithmic tree”. These outputs b_{jk} are at the fine-mesh level. The factor $r = 1/\sqrt{2}$ is included to produce unit vectors in the matrix S , below.

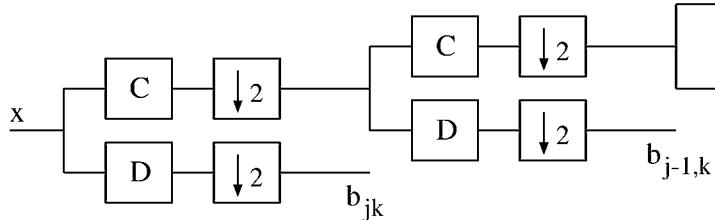


Figure 1.10: The logarithmic tree that leads to wavelets.

The lowpass filter C computes averages. Again the $(\downarrow 2)$ step keeps the even samples. These averages are not final outputs, because they will be filtered again by D and C . The averages and differences of all levels follow the fundamental recursion:

$$\begin{aligned} \text{Averages (lowpass filter)} \quad a_{j-1,k} &= \frac{1}{\sqrt{2}} (a_{j,2k} + a_{j,2k+1}) \\ \text{Differences (highpass filter)} \quad b_{j-1,k} &= \frac{1}{\sqrt{2}} (a_{j,2k} - a_{j,2k+1}). \end{aligned} \tag{1.39}$$

These filters are anticausal. ***There is a time-reversal here.*** That requirement is built in to convolutions and inner products. We will discuss it again below, for functions in continuous time. The essential thing is to see *filters with downsampling* in equation (1.39). Each step takes us from a finer level j to a coarser level $j - 1$, with half as many outputs.

You can see this logarithmic tree as a **pyramid** of averages and differences. The averages are sent up the pyramid, to be averaged again (and also differenced). Whenever a difference is computed, it is final. The sum of $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots$ gives 1, representing 100% of the output in the limit.

Figure 1.11 shows a finite pyramid. The input vector x at the bottom has length $L = 2^J$. It is at level J , where we find 2^{J-1} differences and averages. The averages are the inputs at the next level $J - 1$. Eventually we reach level 0 with an overall average and an overall difference (second half average – first half average). Keep this overall average as the final component—you might say the zeroth component—of the wavelet transform. Starting at level $J = 3$, where the input x has $L = 2^J = 8$ components, the count of wavelet coefficients is

$$4 \text{ differences} + 2 \text{ differences} + 1 \text{ difference} + \text{overall average} = 8.$$

The seven differences are wavelet coefficients b_{jk} . The overall average can be denoted for convenience by a_{00} . In the model case of infinite length, the iteration of the lowpass filter can go on forever.

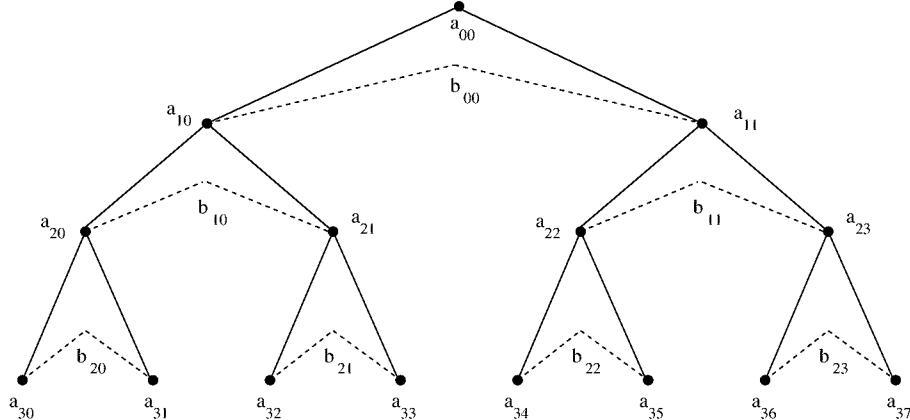


Figure 1.11: Averages a_{jk} go up the pyramid. Differences b_{jk} stop.

You must see the finite case in terms of matrices. With length $L = 4$, there are two fine differences, one coarse difference, and the overall average. The columns of S are the basis vectors for the Haar wavelets:

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \text{ is scaled by } r = \frac{1}{\sqrt{2}} \text{ to } S = \begin{bmatrix} r^2 & r^2 & r & 0 \\ r^2 & r^2 & -r & 0 \\ r^2 & -r^2 & 0 & r \\ r^2 & -r^2 & 0 & -r \end{bmatrix}. \quad (1.40)$$

The scaling gives unit vectors in the columns and rows, because $2r^2 = 1$ and $4r^4 = 1$. The inverse matrix appears in analysis, as we create the tree of averages and differences:

$$\mathbf{A} = \mathbf{S}^{-1} = \mathbf{S}^T = \begin{bmatrix} r^2 & r^2 & r^2 & r^2 \\ r^2 & r^2 & -r^2 & -r^2 \\ r & -r & 0 & 0 \\ 0 & 0 & r & -r \end{bmatrix}. \quad (1.41)$$

The first row gives the overall average a_{00} . The second row gives b_{00} . The third and fourth rows give the finer differences, coming earlier in the tree of filters. The transform $\mathbf{b} = \mathbf{Ax}$ contains these four numbers.

Fast Wavelet Transform (FWT)

The tree of filters is the fast wavelet transform!! The FWT expresses the analysis matrix \mathbf{A} as a product of simple average-difference matrices, coming from the downsampled filters in the tree:

$$\mathbf{A} = \begin{bmatrix} r & r & & \\ r & -r & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} r & r & r & r \\ r & -r & r & -r \\ & & r & -r \end{bmatrix} \quad (1.42)$$

The matrix on the right is first in the tree. The coefficients r, r are in the lowpass filter, and downsampling leaves a matrix \mathbf{L} in the top two rows. The rows of \mathbf{L} have a *double shift*. The coefficients $r, -r$ come from the highpass (or bandpass) filter, downsampled to leave \mathbf{B} . ***There is again a time-reversal from $-r, r$.*** The pattern that you see in the 4 by 4 Haar matrix is the product in (1.42):

$$\text{Analysis tree: } \mathbf{A} = \left[\begin{array}{c|c} \mathbf{L} & \\ \hline \mathbf{B} & \mathbf{I} \end{array} \right] \left[\begin{array}{c} \mathbf{L} \\ \mathbf{B} \end{array} \right].$$

This pattern applies to transforms of all lengths $L = 2^J$. ***It will apply to all wavelets!*** The fast transform expresses \mathbf{A} (and later \mathbf{S}) using *matrices with many zeros*. It is the matrix form of the *pyramid algorithm*.

For length $L = 2^J$, there will be J levels in the tree and J matrices in \mathbf{A} . The matrix on the right, from the start of the tree, has two nonzeros in each row. Filters with T coefficients will produce T nonzeros in each row. Then the finest factor has TL nonzero entries.

The next factor has $TL/2$ coefficients in the top half, processing a shorter input. (The identity matrix in the lower right costs nothing. It leaves the differences alone.) The third stage of the tree has $TL/4$ coefficients. The total for the fast wavelet transform (= factored form of \mathbf{W}^{-1}) comes from J factors:

$$TL\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{J-1}}\right) < 2TL. \quad (1.43)$$

Theorem 1.3 *The fast wavelet transform computes the L coefficients $\mathbf{b} = \mathbf{Ax}$ in less than $2TL$ multiplications.*

The same count applies to the synthesis step $\mathbf{x} = \mathbf{Sb}$. The factors of \mathbf{S} give the inverse of equation (1.42). The synthesis tree has the inverse matrices in opposite order:

$$\mathbf{S} = \begin{bmatrix} r & r & & \\ r & -r & & \\ r & r & r & \\ r & & -r & \end{bmatrix} \begin{bmatrix} r & r & & \\ r & -r & & \\ & & 1 & \\ & & & 1 \end{bmatrix}. \quad (1.44)$$

The matrix on the right inverts the last analysis filter (still with time-reversal). It is the first step in the synthesis tree. The product of J matrices is the whole synthesis bank, which reconstructs x in Figure 1.12.

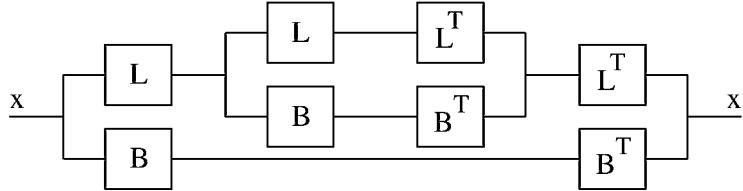


Figure 1.12: Analysis bank followed by synthesis bank: $SAx = x$. The matrices are factored by the tree, producing the FWT.

Haar Wavelets and Recursion

We move now to continuous time. The input $f(t)$ is a function instead of a vector. The output is the set of coefficients b_{jk} that multiply the wavelet basis functions $w_{jk}(t)$. These coefficients are inner products of $f(t)$ with $w_{jk}(t)$. The basis functions are normalized to unit length by the factor $2^{j/2}$:

$$b_{jk} = \langle f, w_{jk} \rangle = \int_{-\infty}^{\infty} f(t) 2^{j/2} w(2^j t - k) dt. \quad (1.45)$$

For Haar, the wavelets are piecewise constant. The original wavelet $w(t) = w_{00}(t)$ is +1 on the interval $[0, \frac{1}{2})$ and -1 on the interval $[\frac{1}{2}, 1)$. The basis function w_{jk} is $2^{j/2}$ on a subinterval of length $\frac{1}{2}2^{-j}$ and $-2^{j/2}$ on the next subinterval. There are four wavelets at level $j = 2$, when we start on the unit interval $[0, 1)$. There are 2^j wavelets at level j .

To compute all the inner product integrals at levels $j = 0$, $j = 1$, and $j = 2$, we can integrate $f(t)$ over all eight subintervals of length $\frac{1}{8}$. This gives eight numbers. *How do those numbers produce b_{2k} and b_{1k} and b_{00} and a_{00} ?*

The answer is beautiful. Those eight numbers are exactly the level 3 averages $a_{30}, a_{31}, a_{32}, \dots, a_{37}$. We act on them exactly as in discrete time: filter and downsample! *The numbers at level $j - 1$ come directly from the numbers at level j .* This is because the functions at level $j - 1$ come from the functions at level j . The box function is a sum of half-boxes and the wavelet is a difference of half-boxes:

$$\phi(t) = \phi(2t) + \phi(2t - 1) \text{ gives } \phi_{j-1,k}(t) = \frac{1}{\sqrt{2}} [\phi_{j,2k}(t) + \phi_{j,2k+1}(t)] \quad (1.46)$$

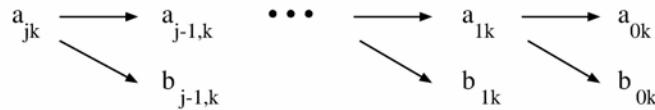
$$w(t) = \phi(2t) - \phi(2t - 1) \text{ gives } w_{j-1,k}(t) = \frac{1}{\sqrt{2}} [\phi_{j,2k}(t) - \phi_{j,2k+1}(t)]. \quad (1.47)$$

Please look at these equations. They are the dilation equation and wavelet equation. The scaling function $\phi_{jk}(t) = 2^{j/2} \phi(2^j t - k)$ is constant on the interval of length 2^{-j} starting at $t = k2^{-j}$. When we take its inner product with $f(t)$, we are integrating $f(t)$ over this subinterval. Multiply the two equations by $f(t)$ and integrate:

$$\begin{aligned} \text{Scaling coefficient: } a_{j-1,k} &= \frac{1}{\sqrt{2}}(a_{j,2k} + a_{j,2k+1}) \\ \text{Wavelet coefficient: } b_{j-1,k} &= \frac{1}{\sqrt{2}}(a_{j,2k} - a_{j,2k+1}). \end{aligned} \quad (1.48)$$

The coefficients follow the pyramid algorithm. Notice again the time-reversal in which $2k + 1$ appears. Our coefficients $a_{jk} = \langle f, \phi_{jk}(t) \rangle$ and $b_{jk} = \langle f, w_{jk}(t) \rangle$ are inner products, and a filter (a convolution $\sum h(k) x(n-k)$) has this reversal.

The main point is the pyramid. This beautiful connection between wavelets and filter banks was discovered by Stéphane Mallat. The pyramid algorithm is also called the Mallat algorithm (don't pronounce the 't'). It is a tree of butterflies in the Preface, it is a tree of filters in Figure 1.11, and a third form of the tree shows the trunk of averages a_{jk} and the branches of differences b_{jk} :



Wavelet coefficients at level $j + 1$ are differences of scaling coefficients at level j .

This pyramid is also an equality of functions. A function at fine resolution j is equal to a combination of "average plus detail" at coarse resolution $j - 1$:

$$\sum_k a_{jk} \phi_{jk}(t) = \sum_k a_{j-1,k} \phi_{j-1,k}(t) + \sum_k b_{j-1,k} w_{j-1,k}(t). \quad (1.49)$$

Multiresolution in Continuous Time

The equality of functions in (1.49) is also an equality of *function spaces*. On the left side is a combination of ϕ 's at level j . Let V_j denote the space of all such combinations. On the right side is a combination of ϕ 's at level $j - 1$. This is a function in the scaling space V_{j-1} . Also on the right is a combination of w 's at level $j - 1$. This is a function in the wavelet space W_{j-1} . The key statement of multiresolution is

$$V_j = V_{j-1} \oplus W_{j-1} \quad (1.50)$$

The symbol $+$ for vector spaces means that every function in V_j is a sum of functions in V_{j-1} and W_{j-1} , as in equation (1.49). The symbol \oplus for "direct sum" means that those smaller spaces meet only in the zero function. This is guaranteed when ***the two subspaces are orthogonal***. In that case the direct sum \oplus becomes an "orthogonal sum". For emphasis we restate the definition of the subspaces:

$$\begin{aligned} V_j &= \text{all combinations } \sum_k a_{jk} \phi_{jk}(t) \text{ of scaling functions at level } j \\ W_j &= \text{all combinations } \sum_k b_{jk} w_{jk}(t) \text{ of wavelets at level } j. \end{aligned}$$

V_j is spanned by translates of $\phi(2^j t)$ and W_j is spanned by translates of $w(2^j t)$. The time scale is 2^{-j} . At each level, all inner products are zero. We have two orthogonal bases for V_j , either the ϕ 's at level j or the ϕ 's and w 's at level $j - 1$.

Notice how this multiresolution grows to three levels or more:

$$V_3 = V_2 \oplus W_2 = V_1 \oplus W_1 \oplus W_2 = V_0 \oplus W_0 \oplus W_1 \oplus W_2. \quad (1.51)$$

On the left side are all piecewise constant functions on intervals of length $\frac{1}{8}$. On the right side is the same space of functions, differently expressed. The functions in V_0 are constant on $[0, 1]$. The functions in W_0, W_1, W_2 are combinations of wavelets. The function $f(t)$ in V_3 has a piece $f_j(t)$ in each wavelet subspace W_j (plus V_0):

$$f(t) = \sum_k a_{0k} \phi_{0k}(t) + \sum_k b_{0k} w_{0k}(t) + \sum_k b_{1k} w_{1k}(t) + \sum_k b_{2k} w_{2k}(t). \quad (1.52)$$

Note to the reader. The parallels between a filter tree in discrete time and multiresolution in continuous time are almost perfect. The filter bank separates lower and lower frequencies, as we iterate. Multiresolution uses longer and longer wavelets, as we climb the pyramid. We are speaking of the analysis half, where inputs are separated by *scale*.

<i>Discrete time</i>	<i>Continuous time</i>
filter bank tree	multiresolution
downsampling $\omega \rightarrow \frac{\omega}{2}$	rescaling $t \rightarrow 2t$
lowpass filter	averaging with $\phi(t)$
highpass filter	detailling with $w(t)$
orthogonal matrices	orthogonal bases
analysis bank output	wavelet coefficients
synthesis bank output	sum of wavelet series
product of filter matrices	fast wavelet transform

The reader will understand that in writing about Haar wavelets, we are writing about all wavelets. The pattern is fundamental, the pieces in the pattern can change. The actual $c(k)$ and $d(k)$ and $\phi(t)$ and $w(t)$ are at our disposal. The next chapters move to *filter design*, where we make choices. Those choices determine the wavelet design. Some filters and wavelets are better than others. We will not allow ourselves to forget the pattern that makes all of them succeed.

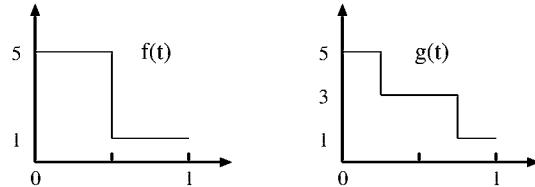
Caution. The Haar wavelets are orthogonal. Thus they are biorthogonal to *themselves*. We are not seeing a clear difference between analysis and synthesis filters, \mathbf{C} and \mathbf{D} versus \mathbf{F} and \mathbf{G} . For the same reason we are not seeing the dual functions $\tilde{\phi}(t)$ and $\tilde{w}(t)$. A clue to this shadow world (or tilde world) is in the time-reversals, which involve the *next* sample $2k+1$ instead of the previous sample $2k-1$. This suggests filters \mathbf{C}^T and \mathbf{D}^T rather than \mathbf{C} and \mathbf{D} .

The biorthogonal case comes in Section 6.5. It has *two* multiresolutions, $\tilde{V}_{j+1} = \tilde{V}_j \oplus \tilde{W}_j$ in parallel with $V_{j+1} = V_j \oplus W_j$. The pyramid and the fast wavelet transform go one way in analysis (with tilde). The inverse transform goes the other way in synthesis (without tilde).

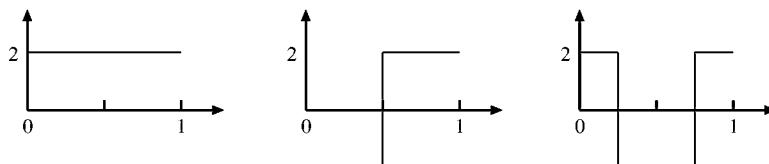
Problem Set 1.6

1. (a) Show that the exact sum in equation (1.43) is $2T(N-1)$.
- (b) How many matches are needed to decide the winner in a knockout tournament with N players? The average is the winner that goes to the next round (next filter). The difference is the loser that stops.

2. Write out the factorization of Haar's A for $N = 8$ (following 1.42).
3. For $N = 8$, write out the factorization of S corresponding to (1.44).
4. Draw the synthesis tree, the reverse of Mallat's analysis tree. A similar pyramid algorithm was proposed by Burt and Adelson before wavelets were named.
5. Split the function $f(t)$ into a scaling function plus a wavelet.



6. Split the function $g(t)$ into its pieces in V_0 , W_0 , and W_1 (box plus up-down coarse wavelet plus two fine wavelets).
7. The function $g(t)$ is in V_2 (its scale is $\frac{1}{4}$). The pyramid splits it first into a function in V_1 (two half-size boxes) plus a function in W_1 (two half-size wavelets). Find those pieces. Then split the piece in V_1 into a full-size box plus a full-size wavelet.
8. These three pieces are in V_0 and W_0 and W_1 . Synthesize $f_1(t)$ in V_1 from the first two pieces. Then add the details in the third piece to synthesize $f_2(t)$ in V_2 .



9. Suppose $H(t - \frac{1}{3})$ is the unit step function with jump at $t = \frac{1}{3}$. Its inner products a_{jk} with the boxes $\phi_{jk}(t)$ will be nonzero for about two-thirds of the 2^j boxes on $[0, 1]$. How many inner products b_{jk} with the wavelets $w_{jk}(t)$ will be nonzero?

This example shows the compression of step functions by Haar wavelets.

Chapter 2

Filters

This chapter comes between the Haar example of Chapter 1 and the full development of filter banks (leading to wavelets). We decided to collect other definitions that belong to this circle of ideas. Here is an indication of our plan:

Basic filters: Ideal filters and then FIR filter design

Basic tools: Fourier methods and functional analysis

Bases and frames: A matrix T has a two-sided or only a one-sided inverse

Integral transforms: windows in time-frequency, wavelets in time-scale.

Signal processing is an enormous subject. The input signal can arrive in many forms: continuous time, discrete time, finite time. It can be processed in many ways. Our greatest interest is a signal $x(n)$ in discrete time that is processed by a linear time-invariant operator. *If the input is shifted in time then the output is equally shifted.* These operators are **filters** —the fundamental actors in signal processing.

Filters can be expressed in three domains: n , ω , z . In each domain the filter is a multiplication:

(n) : Multiplication by a **Toeplitz matrix** with $\mathbf{h}(n)$ on the n th diagonal.

(ω) : Multiplication by the **frequency response** $H(e^{j\omega}) = \sum \mathbf{h}(n)e^{-j\omega n}$.

(z) : Multiplication by the **transfer function** $H(z) = \sum \mathbf{h}(n)z^{-n}$.

We want to explain these three forms and the connections between them. *If we emphasize the matrix form more than usual, it is because that form is less well known.* We believe that the matrix formulation must become familiar, as the teaching and practice of signal processing rely increasingly on computer systems like MATLAB.

Our goal is to understand filters, through impulse responses and frequency responses and transfer functions. If you know signal processing as in the Oppenheim–Schafer text, go past the first sections. If you are learning the whole subject from scratch, these sections can help. Do not hesitate to use this chapter for reference, as you reach Chapter 4 and the heart of the book.

We begin with signals.

2.1 Signals, Samples, and Time-invariance

A discrete-time signal is a sequence of numbers. The sequence could be finite or infinite. Most signals in this book are *doubly infinite*; the index n goes from $-\infty$ to $+\infty$. Time has no start and no finish. The signals look like

$$\mathbf{x} = (\dots, x_{-1}, x_0, x_1, x_2, \dots) \quad \text{or} \quad \mathbf{x} = \begin{bmatrix} \dots \\ x(-1) \\ x(0) \\ x(1) \\ x(2) \\ \dots \end{bmatrix}.$$

Those components are real or complex numbers (usually real). One particular signal is of tremendous value. It is the *unit impulse* $\mathbf{x} = \delta$:

$$\delta = (\dots, 0, 0, 1, 0, 0, \dots) \quad \text{has components} \quad \delta(n) = \begin{cases} 0, & n \neq 0 \\ 1, & n = 0. \end{cases} \quad (2.1)$$

The continuous-time analogue is the “delta function” $\delta(t)$ —also called a Dirac impulse. In one case n is an integer, in the other t is a real number. The standard notations are $n \in \mathbf{Z}$ and $t \in \mathbf{R}$.

Together with the special vector δ goes the delayed impulse $S\delta$, where the unit component appears one sample later at $n = 1$. The whole vector is shifted by one time step. The symbol S stands for the *shift* or *delay* that has this effect on the vector δ :

$$S\delta = (\dots, 0, 0, 0, 1, 0, \dots) \quad \text{has components} \quad \delta(n-1) = \begin{cases} 0, & n \neq 1 \\ 1, & n = 1. \end{cases}$$

It is worth emphasizing that a shift *to the right* (a delay) produces the *minus sign* in the expression $n - 1$. It is the same in continuous time. The graph of $f(t)$, when it is shifted one unit to the right, is the graph of $f(t - 1)$. The delayed function at $t = 1$ equals the original function at $t = 0$.

When the components are shifted to the left, the impulse comes sooner (at $n = -1$). This shift is an *advance* instead of a delay. The symbol is S^{-1} . The operator “ S inverse” has an effect opposite to S :

$$S^{-1}\delta = (\dots, 0, 1, 0, 0, 0, \dots) \quad \text{has components} \quad \delta(n+1) = \begin{cases} 0, & n \neq -1 \\ 1, & n = -1. \end{cases}$$

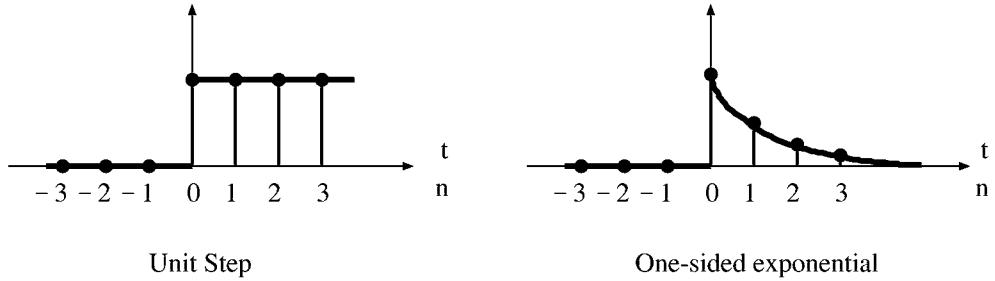
Of course $S^{-1}S\delta = \delta$. The impulse is delayed by S and then advanced by S^{-1} . Also $SS^{-1}\delta = \delta$. The operator S^{-1} is a “two-sided inverse” of S .

The vector δ could be defined, and Dirac’s delta function should be defined, by what happens for inner products. The inner product (dot product) with any vector $\mathbf{x}(n)$ or any continuous function $x(t)$ picks out $\mathbf{x}(0)$:

$$\mathbf{x}^T \delta = \sum_{-\infty}^{\infty} \mathbf{x}(n) \delta(n) = \mathbf{x}(0) \quad \text{and} \quad \langle x(t), \delta(t) \rangle = \int_{-\infty}^{\infty} x(t) \delta(t) dt = \mathbf{x}(0).$$

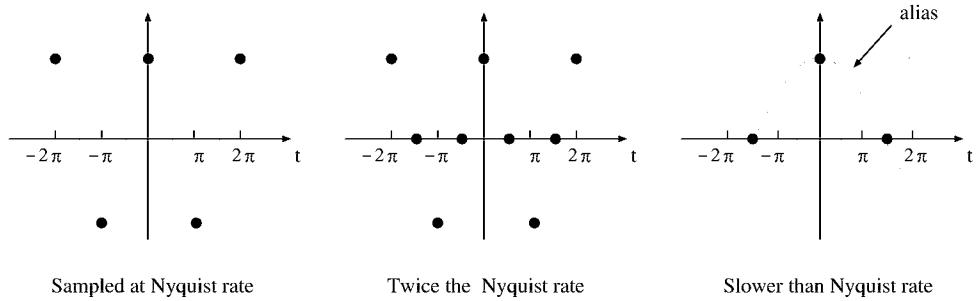
The number $\mathbf{x}(0)$ is a sample of the function $x(t)$. Many discrete signals come from sampling continuous signals. This *analog to digital* (A/D) conversion is a central part of communications

technology. We sketch two continuous-time signals $x(t)$, a step and an exponential, and their discrete-time samples $x(n)$.



The samples of $\cos \omega t$ are of special importance. The continuous signal has frequency ω (often normalized as $f = \frac{\omega}{2\pi}$). Everything depends on the sampling period T and the sampling rate $f_s = \frac{1}{T}$. The sampling may be fast enough to catch the oscillations in $\cos \omega t$, or it may be too slow. We may catch the oscillations in $\cos \omega t$, or miss some. The borderline is the **Nyquist rate**. The sampling rate is exactly the Nyquist rate when $f_s = 2f$ and $\frac{1}{T} = \frac{\omega}{\pi}$ and $\omega T = \pi$. This is the rate (two samples per oscillation) in the first figure. Those samples have the fastest oscillation that a discrete vector can achieve: $x(n) = (-1)^n$.

To repeat: The Nyquist rate gives the highest possible frequency $\omega T = \pi$.



The second sampling rate is *faster than Nyquist*. So the digital frequency ωT is less than π . In this figure, the sampling frequency f_s is twice the Nyquist rate, four samples (*four bullets*) per oscillation. The sampled signal is $x(n) = \cos \omega nT = \cos \frac{n\pi}{2}$. Therefore the samples are $1, 0, -1, 0, 1, 0, \dots$

The third sampling rate is *slower than Nyquist*. The sample after $\cos 0$ is $\cos \frac{3\pi}{2}$. The sampling rate is $\frac{2}{3}$ of the Nyquist rate ($\frac{1}{T} = \frac{2}{3}\frac{\omega}{\pi}$). At this slow rate, the samples are the same $1, 0, -1, 0, 1, 0, \dots$ as in the second figure! We cannot tell whether the true frequency of the continuous signal is ω (as drawn with solid line) or a slower frequency $\frac{\omega}{3}$ (as drawn by dotted line). **This is aliasing**. The slow frequency $\frac{\omega}{3}$ is an alias for the true frequency ω , because the discrete samples at this rate will be exactly the same.

An extreme case of slow sampling is when $\omega T = 2\pi$. All the samples are $\cos \omega nT = \cos 2\pi n = 1$. Every sample is at the top of the wave. The digital frequency 2π looks identical to frequency zero (which is the alias of 2π).

When the continuous signal is a combination of many frequencies ω (or $f = \frac{\omega}{2\pi}$), the largest one ω_{max} sets the Nyquist rate $2f_{max}$. The corresponding Nyquist period T has $\omega_{max}T = \pi$. As

long as the sampling period is smaller than this T , the sampling rate is faster than the Nyquist rate. Then there is no aliasing (by a lower frequency than the true frequency). *The continuous signal $x(t)$ can be recovered from its samples $\mathbf{x}(n)$.* The Shannon sampling formula, to achieve that recovery, is in Section 2.2.

Impulses and Delays in Three Domains

Impulses are the building blocks for all signals. Delays are the building blocks for all filters. We will present signals and filters in three ways—with time variable n , and frequency variable ω , and complex variable z .

$$\begin{array}{lll} \text{Signal in the time domain} & \mathbf{x}(n) & = (\dots, \mathbf{x}(-1), \mathbf{x}(0), \mathbf{x}(1), \dots) \\ \text{Signal in the frequency domain} & X(e^{j\omega}) & = \sum \mathbf{x}(n)e^{-jn\omega} \text{ (standard)} \\ & X(\omega) & = \sum \mathbf{x}(n)e^{-i\omega n} \text{ (reduced)} \\ \text{Signal in the } z\text{-domain} & X(z) & = \sum \mathbf{x}(n)z^{-n}. \end{array}$$

The standard notation and reduced notation were compared in Section 1.1. We use both! Sometimes the standard notation is clearer; it allows direct replacement of $e^{j\omega}$ by z . Sometimes the reduced notation is simpler, as in $Y(\omega) = H(\omega)X(\omega)$.

The impulse $\delta = (\dots, 0, 0, 1, 0, 0, \dots)$ becomes the constant function “1” in the frequency domain and z -domain. The only nonzero component $\delta(0) = 1$ is in the constant term. The delayed impulse $\mathbf{y} = (\dots, 0, 0, 0, 1, 0, \dots)$ looks more interesting. In the other domains this $\mathbf{y}(n) = \delta(n - 1)$ is $Y(e^{j\omega}) = e^{-j\omega}$ and $Y(z) = z^{-1}$.

Note that $\mathbf{y}(1) = 1$ multiplies the *negative power* of z , by the signal processing convention. If the impulse is advanced instead of delayed, the nonzero occurs at $n = -1$. The transform is z instead of z^{-1} . This signal is no longer causal. The advance operator S^{-1} is not a causal filter.

Now we define filters in general and study delays in particular.

A ***digital filter*** is a combination $\mathbf{H} = \sum \mathbf{h}(n)S^n$ of delays S and advances S^{-1} .

The filter is completely determined by its coefficients $\mathbf{h}(n)$. When this sequence is finite, we have an “FIR filter”. When $\mathbf{h}(n) = 0$ for negative n , we have a “causal filter”. Our greatest interest is in causal FIR filters like

$$\mathbf{H} = \frac{1 + \sqrt{3}}{8}\mathbf{I} + \frac{3 + \sqrt{3}}{8}\mathbf{S} + \frac{3 - \sqrt{3}}{8}\mathbf{S}^2 + \frac{1 - \sqrt{3}}{8}\mathbf{S}^3 \quad (\text{Daubechies } D_4 \text{ filter}).$$

Suppose this filter acts on the impulse δ . The output is a combination of δ and its delays $S\delta$ and $S^2\delta$ and $S^3\delta$:

$$\mathbf{H}\delta = (\dots, \frac{1 + \sqrt{3}}{8}, \frac{3 + \sqrt{3}}{8}, \frac{3 - \sqrt{3}}{8}, \frac{1 - \sqrt{3}}{8}, 0, \dots)$$

This is the “impulse response.” It is equal to the vector \mathbf{h} of filter coefficients:

The ***impulse response*** (causal and FIR) is $\mathbf{h} = (\mathbf{h}(0), \mathbf{h}(1), \dots, \mathbf{h}(N))$.

You see why $\mathbf{H} = \sum \mathbf{h}(n)S^n$ acting on δ produces this output \mathbf{h} . Each term $\mathbf{h}(n)S^n$ produces one response $\mathbf{h}(n)$ at time n .

As displayed, the filter is FIR and causal with $N + 1$ “taps”. The filter length is even when N is odd! Some authors end at $\mathbf{h}(N - 1)$, so the length is N — but then the power $z^{-(N-1)}$ enters into a large number of formulas. We prefer to have sums from 0 to N , and scaling functions and wavelets on the interval $0 \leq t < N$, and z^{-N} in all those formulas. The Daubechies filter has length 4 because $N = 3$.

The delay \mathbf{S} takes $\mathbf{x} = (\dots, \mathbf{x}(0), \mathbf{x}(1), \dots)$ into $\mathbf{y} = (\dots, \mathbf{x}(-1), \mathbf{x}(0), \dots)$. Every linear operator like \mathbf{S} is represented by a matrix. Since \mathbf{x} and $\mathbf{y} = \mathbf{S}\mathbf{x}$ have infinitely many components, the shift matrix \mathbf{S} has infinitely many rows and columns. This causal operator becomes a *lower triangular* matrix:

$$\mathbf{S}\mathbf{x} = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & 0 & 0 & 0 \\ \cdot & 1 & 0 & 0 & 0 \\ \cdot & 0 & 1 & 0 & 0 \\ \cdot & 0 & 0 & 1 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \mathbf{x}(-1) \\ \mathbf{x}(0) \\ \mathbf{x}(1) \\ \mathbf{x}(2) \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot \\ \mathbf{x}(-2) \\ \mathbf{x}(-1) \\ \mathbf{x}(0) \\ \mathbf{x}(1) \\ \cdot \end{bmatrix}.$$

The only nonzero coefficient for this filter is $\mathbf{h}(1) = 1$. This coefficient goes along diagonal *one*. In general $\mathbf{h}(n)$ goes on diagonal n .

What does the delay do in the z -domain? The input $X(z)$ and output $Y(z)$ are

$$X(z) = \dots + \mathbf{x}(0) + \mathbf{x}(1)z^{-1} + \dots \quad \text{and} \quad Y(z) = \dots + \mathbf{x}(0)z^{-1} + \mathbf{x}(1)z^{-2} + \dots$$

The delay has multiplied $X(z)$ by z^{-1} to produce $Y(z)$. This transfer function z^{-1} is exactly the z -transform of the vector $(\dots, 0, 1, 0, 0, \dots)$ of filter coefficients. The pattern is always $Y(z) = H(z)X(z)$. Here is the special result for this particular filter, a delay $\mathbf{H} = \mathbf{S}$:

$$X(e^{j\omega}) \text{ is multiplied by } e^{-j\omega} \text{ and } X(z) \text{ is multiplied by } z^{-1}.$$

Time-invariant Filters

Our filters \mathbf{H} are *linear*. This means in particular that “zero in produces zero out”. If $\mathbf{x} = 0$ then necessarily $\mathbf{y} = 0$. The output from $2\mathbf{x}$ is $2\mathbf{Hx}$. The output from $\mathbf{x} + \mathbf{z}$ is $\mathbf{Hx} + \mathbf{Hz}$. This has an important consequence: \mathbf{H} is represented by a matrix.

Our filters are also *time-invariant* (meaning shift-invariant). This leads to a special constant-diagonal property of the matrix:

$$\mathbf{H}(\mathbf{S}\mathbf{x}) = \mathbf{S}(\mathbf{Hx}) : \text{A shift of the input produces a shift of the output.}$$

Each column of \mathbf{H} is a delay of the previous column.

Each diagonal of \mathbf{H} is constant and the n th diagonal contains $\mathbf{h}(n)$.

Those are different statements of time-invariance. They imply that \mathbf{H} is a combination of shift operators: Every filter has the form $\mathbf{H} = \sum \mathbf{h}(n)\mathbf{S}^n$. The filter $\mathbf{H} = \mathbf{I} + 4\mathbf{S} + 3\mathbf{S}^2$ has coefficients $\mathbf{h} = (1, 4, 3)$. These are the entries down every column of the Toeplitz matrix.

$$H = \left[\begin{array}{cccccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & 0 & 0 & 0 & \cdot \\ \cdot & 4 & 1 & 0 & 0 & \cdot \\ \cdot & 3 & 4 & 1 & 0 & \cdot \\ \cdot & 0 & 3 & 4 & 1 & \cdot \\ \cdot & \cdot & \cdot & \bullet & \bullet & \bullet \\ \end{array} \right] \quad \begin{array}{l} \text{row 0} \\ \text{row 1} \end{array}$$

column 0 1 2 1 0

diagonal number -1

Toepplitz matrix = constant-diagonal matrix with entries $\mathbf{H}_{ij} = \mathbf{h}(i - j)$.

The numbers 1, 4, 3 also appear in every row — *but the order is reversed*. This is a causal FIR filter. It is time-invariant, because \mathbf{HS} or \mathbf{SH} (they are the same!) is $S + 4S^2 + 3S^3$. All columns and all diagonals shift down by one, from the delay.

The difference between the row number i and the column number j is the diagonal number $k = i - j$. The entries of \mathbf{H} depend only on k . This is a constant-diagonal matrix and a convolution matrix and a *Toeplitz matrix*.

Matrix Multiplication and Vector Convolution

The j th column of \mathbf{H} is $\mathbf{S}^j \mathbf{h}$. We can compute \mathbf{Hx} as a combination $\sum x(j)(\mathbf{S}^j \mathbf{h})$ of those columns. We can also compute $\sum \mathbf{h}(k)(\mathbf{S}^k \mathbf{x})$. Best to see the n th component $\mathbf{Hx}(n)$:

$$\sum_j x(j)h(n-j) = \sum_k h(k)x(n-k).$$

\uparrow \uparrow
nth component of delayed \mathbf{h} nth component of delayed \mathbf{x}

The equality comes by changing j to $n - k$. The sums are over all integers, so the change is allowed. The finite sum to $j = N$ would not equal the sum to $k = N$, unless \mathbf{h} has period N . (Then the N by N matrix \mathbf{H} would be a *circulant matrix*. This “wraparound” is the easiest way to deal with finite length signals, but not generally the best way.)

In both formulas for $\mathbf{H}\mathbf{x}(n)$, the indices add to n . The zeroth component is row zero of \mathbf{H} times \mathbf{x} :

$$y(0) = Hx(0) = h(N)x(-N) + \cdots + h(1)x(-1) + h(0)x(0), \quad (2.2)$$

Each pair of indices on the right adds to zero—which is the index on the left. The numbers $h(N), \dots, h(0)$ are like a *moving window* that multiplies x . The n th component of the output has indices adding to n :

$$\mathbf{H}\mathbf{x}(n) = \mathbf{h}(N)\mathbf{x}(n-N) + \cdots + \mathbf{h}(1)\mathbf{x}(n-1) + \mathbf{h}(0)\mathbf{x}(n). \quad (2.3)$$

This is the pattern that produces ***convolution***:

The output vector is $Hx = h * x =$ convolution of h with x .

Convolution Rule Indices automatically add when they are the exponents in a polynomial. Multiply $H(z)X(z)$ and their coefficients undergo a convolution:

$$H(z)X(z) = (\mathbf{h}(0) + \mathbf{h}(1)z^{-1} + \cdots + \mathbf{h}(N)z^{-N})(\cdots x(-1)z + x(0) + x(1)z^{-1} + \cdots).$$

The coefficient of z^0 is $\mathbf{h}(0)x(0) + \mathbf{h}(1)x(-1) + \cdots + \mathbf{h}(N)x(-N)$. This is $\mathbf{Hx}(0)$.

The coefficient of z^{-n} in the product $H(z)X(z)$ is the n th component of \mathbf{Hx} . Multiplying polynomials means collecting terms with the same exponent. This is convolution.

Example 2.1. The filter matrix has $\mathbf{h}(0) = 1$, $\mathbf{h}(1) = 4$, and $\mathbf{h}(2) = 3$. Suppose the input has $x(0) = 1$ and $x(1) = 1$. Then we multiply polynomials or we take convolution of vectors:

$$\begin{array}{r} 3 \ 4 \ 1 \\ 1 \ 1 \\ \hline 3 \ 4 \ 1 \\ 3 \ 4 \ 1 \\ \hline 3 \ 7 \ 5 \ 1 \end{array} \quad (1 + 4z^{-1} + 3z^{-2})(1 + z^{-1}) = 1 + 5z^{-1} + 7z^{-2} + 3z^{-3}$$

$$(1, 4, 3) * (1, 1) = (1, 5, 7, 3).$$

At $z = 1$ this is 8 times 2 equals 16. At $z = -1$ we check 0 times 0 equals 0.

Example 2.2. Multiplying two filter matrices (Toeplitz matrices) is also a convolution. The product \mathbf{FH} is another time-invariant filter, and its coefficients are in $f * \mathbf{h}$. This is just like multiplying polynomials, with the shift S in place of the complex variable z^{-1} :

$$\mathbf{FH} = (\mathbf{I} + S)(\mathbf{I} + 4S + 3S^2) = \mathbf{I} + 5S + 7S^2 + 3S^3.$$

Note again that the order is not important: $\mathbf{FH} = \mathbf{HF}$. This will change when there are sampling operators ($\downarrow 2$) and ($\uparrow 2$) between the filters.

Example 2.3. The convolution of $(1, a, a^2, \dots)$ with $(1, b, b^2, \dots)$ is

$$(1 + bz^{-1} + b^2z^{-2} + \cdots)(1 + az^{-1} + a^2z^{-2} + \cdots) = 1 + (a+b)z^{-1} + (a^2 + ab + b^2)z^{-2} + \cdots$$

The power z^{-2} in the product comes from z^0 times z^{-2} , and from z^{-1} times z^{-1} , and from z^{-2} times z^0 . The sum of exponents is -2 , to give z^{-2} in the answer. This is the pattern for the indices k and $n - k$. Their sum is always n , to give $y(n)$ in the convolution.

The Inverse of a Time-invariant H

The filter H is *invertible* if and only if

$$\begin{aligned} H(\omega) &\neq 0 & \text{for all frequencies } \omega \\ H(z) &\neq 0 & \text{for all } |z| = |e^{j\omega}| = 1. \end{aligned} \tag{2.4}$$

Then H^{-1} is also a constant-diagonal matrix. Its frequency response is $1/H(\omega)$.

Invertibility is the first of many properties that become infinitely simpler by transforming convolution to $H(\omega)X(\omega)$. The inverse of multiplication is division! We recover $X(\omega)$ from $Y(\omega)/H(\omega)$. The requirement is $H(\omega) \neq 0$.

To emphasize: If we know a frequency ω_0 for which $H(\omega_0) = 0$, then we know an input \mathbf{x} for which $\mathbf{Hx} = \mathbf{0}$. That input has the pure frequency ω_0 . It is the vector with components $\mathbf{x}(n) = e^{-j\omega_0 n}$. The pure frequency is selectively killed by $H(\omega_0) = 0$. Then $H(\omega_0)\mathbf{X}(\omega_0) = \mathbf{0}$ and \mathbf{H}^{-1} fails.

A moving average with equal weights $\mathbf{h}(0) = \mathbf{h}(1) = \frac{1}{2}$ is not invertible. The frequency response $H(\omega) = \frac{1}{2}(1 + e^{-j\omega})$ is zero at $\omega = \pi$. The vector with components $\mathbf{x}(n) = e^{-j\pi n} = (-1)^n$ is exactly the vector that has $\mathbf{Hx} = \mathbf{0}$. By changing to two unequal weights *the system becomes invertible*.

Example 2.4. Suppose $\mathbf{h}(0) = 1$ and $\mathbf{h}(1) = -\beta$. The frequency response is $H(\omega) = 1 - \beta e^{-i\omega}$. If we select β smaller than one, then $1 \neq \beta e^{-i\omega}$. Thus $H(\omega) \neq 0$. The matrix \mathbf{H} has 1 on the main diagonal and $-\beta$ on the diagonal below. To invert in the frequency domain, divide by $H(\omega)$. To invert in the time domain, practice with a 4 by 4 matrix:

$$\begin{bmatrix} 1 & & & \\ -\beta & 1 & & \\ & -\beta & 1 & \\ & & -\beta & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & & & \\ \beta & 1 & & \\ \beta^2 & \beta & 1 & \\ \beta^3 & \beta^2 & \beta & 1 \end{bmatrix}. \quad (2.5)$$

This suggests the correct diagonals $1, \beta, \beta^2, \dots$ for the infinite matrix \mathbf{H}^{-1} . If \mathbf{H} is $\mathbf{I} - \beta\mathbf{S}$, its inverse $\mathbf{I} + \beta\mathbf{S} + \beta^2\mathbf{S}^2 + \dots$ has the frequency response $1/H(\omega)$:

$$\frac{1}{1 - \beta e^{-i\omega}} = 1 + \beta e^{-i\omega} + (\beta e^{-i\omega})^2 + (\beta e^{-i\omega})^3 + \dots$$

The most important of all infinite series (the *geometric series*) gives us this inverse: $\frac{1}{1 - \beta} = 1 + \beta + \beta^2 + \dots$. The sum is restricted to $|\beta| < 1$. Otherwise the series diverges.

Example 2.5. What if β is larger than 1? For a finite matrix we don't notice the difference. The 4 by 4 inverse above is still correct. But the infinite series has to be written *in powers of $1/\beta$* . The inverse matrix changes from causal to anticausal. Look first at the frequency response $1/H(\omega)$:

$$\frac{1}{1 - \beta e^{-i\omega}} = \frac{e^{i\omega}/\beta}{(e^{i\omega}/\beta) - 1} = -\frac{e^{i\omega}}{\beta} - \frac{e^{2i\omega}}{\beta^2} - \frac{e^{3i\omega}}{\beta^3} - \dots. \quad (2.6)$$

This involves positive powers. We have *advances instead of delays* in the inverse.

The difference between $|\beta| < 1$ and $|\beta| > 1$ is the difference between a zero *inside* the unit circle and a zero *outside* that circle. $H(z) = 1 - \beta z^{-1}$ has its only zero at $z = \beta$. Here is the general rule for inverses of causal FIR systems:

No inverse when a zero is *on* the unit circle : $\mathbf{I} + \mathbf{S}$ has no inverse.

Causal inverse when all zeros are *inside* the unit circle: $1 - \beta z^{-1}$ has $|\beta| < 1$.

Anticausal inverse when all zeros are *outside* $|z| = 1$: $1 - \beta z^{-1}$ has $|\beta| > 1$.

A zero on the unit circle gives a particular frequency ω_0 at which $H = 0$. Then $1/H$ breaks down and the system has no inverse.

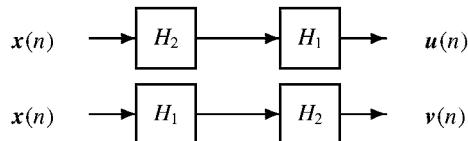
Problem Set 2.1

1. What matrix represents the inverse shift $y = S^{-1}x$? In the z -domain, the input is $X(z) = \cdots + x(0) + x(1)z^{-1} + \cdots$ and the output is $Y(z) = \underline{\hspace{2cm}}$. The advance S^{-1} multiplies the z -transform by $\underline{\hspace{2cm}}$.
2. Express the filter $H = S + S^{-1}$ with coefficients $h(-1) = 1$ and $h(1) = 1$ in all three domains. Write the matrix H with two nonzero diagonals. Write the transfer function $H(z)$ with two terms. Write the frequency response $H(e^{j\omega})$ or $H(\omega)$, and check that this is the transform of the impulse response $h = H\delta$.
3. Show that $H = S + S^{-1}$ is not invertible in three ways. Find a nonzero input x such that $Hx = 0$. Find a frequency ω that has response $H(e^{j\omega}) = 0$. Find a number with $|z| = 1$ such that $H(z) = 0$.
4. What are the matrix H and coefficient vector h for the 3-term moving average $Hx(n) = \frac{1}{3}(x(n) + x(n-1) + x(n-2))$? This is not invertible. Find two vectors x for which $Hx = 0$. Find two numbers with $|z| = 1$ such that $H(z) = 0$. Find two frequencies such that $H(\omega) = 0$.
5. Express this 3-term moving average in the form $H = \sum h(n)S^n$. What is N ? Find the output y when the input has $x(0) = x(1) = 1$. In the z -domain what are $X(z)$ and $Y(z)$, and how are they related?
6. For matrices show that $SS^{-1} = I$. What is the corresponding statement in the z -domain, about the transfer functions of S and S^{-1} ?
7. Multiply the matrix S by itself. The product $H = S^2$ corresponds to what coefficient vector h and what transfer function $H(z)$?
8. Every filter $\sum h(n)S^n$ commutes with a delay: $\sum h(n)S^{n+1}$ is HS and also SH . Why does every filter commute with every other filter?
9. If the continuous-time signal is $x(t) = \cos t$, what is the period T that gives sampling exactly at the Nyquist rate? What samples $x(nT)$ do you get at this rate? What samples do you get from $x(t) = \sin t$?
10. If the sampling period is $T = 1$ and the continuous signal is $x(t) = e^{2\pi it/5}$, describe the discrete signal $x(n)$. Is it periodic? Find two other frequencies ω such that $x(t) = e^{j\omega t}$ would give the same samples.
11. If the signal $x(t)$ has bandwidth 3 KHz, then the sampling rate must be at least $\underline{\hspace{2cm}}$ to avoid aliasing.
Note that the sampling period is generally normalized to $T = 1$. Then the largest digital frequency is $\omega = \pi$. Our graphs of $H(\omega)$ do not extend beyond π .
12. Why is the downsampling operator $(\downarrow 2)x(n) = x(2n)$ not time-invariant? Give an example with $(\downarrow 2)Sx \neq S(\downarrow 2)x$.
13. When are these filters invertible? Which has a causal inverse? Which has an FIR inverse? Which is allpass with $|H(e^{j\omega})| = 1$?

$$\begin{aligned} H_1(z) &= (1 - \alpha z^{-1})(1 - \beta z^{-1}) \\ H_2(z) &= 1 + \beta z^{-1} + \beta^2 z^{-2} + \beta^3 z^{-3} + \dots \\ H_3(z) &= (z - \beta)/(1 - \beta z^{-1}) \\ H_4(z) &= 1 - \beta z^{-1} + z^{-2} \end{aligned}$$

14. Determine the range of α and β for which the LTI system with impulse response $h(n) = \begin{cases} \alpha^n; & n \geq 0 \\ \beta^n; & n < 0 \end{cases}$ is stable. Find the output $y(n)$ when $x(n) = (-1)^n$.
15. Determine the impulse response for the cascade of the two LTI systems having impulse responses $h_1(n) = (\frac{1}{2})^n u(n)$ and $h_2(n) = (\frac{1}{4})^n u(n)$ using the convolution formula $h(n) = \sum h_1(k)h_2(n-k)$. Here $u(n)$ is the unit-step sequence.

16. Let H_1 be a system that throws away odd-indexed samples: $y(n) = x(2n)$. Is H_1 linear and time-invariant? H_1 is the downsampling block and its operation is discussed in Chapter 3.
17. Suppose H_2 inserts an extra zero between samples of the input: $y(n) = \begin{cases} x(n/2), & \text{even } n \\ 0, & \text{odd } n \end{cases}$. Is H_2 a linear time-invariant system? H_2 is the upsampling block and its operation is discussed in Chapter 3.
18. Which cascade of downsampling and upsampling is time-invariant and what is its impulse response?



19. Give two examples of LTI systems, two examples of linear time-varying systems, and two examples of nonlinear systems.

2.2 Ideal Filters, Shannon Sampling, Sinc Wavelets

The word *filter* suggests that H selects a band of frequencies. It rejects another band. For ω in the **passband**, the frequency response is near to $H(\omega) = 1$. For ω in the **stopband**, the response is near to $H(\omega) = 0$. Any realizable non-ideal filter has a **transition band** in between, where $H(\omega)$ changes from pass to stop (from near 1 to near 0).

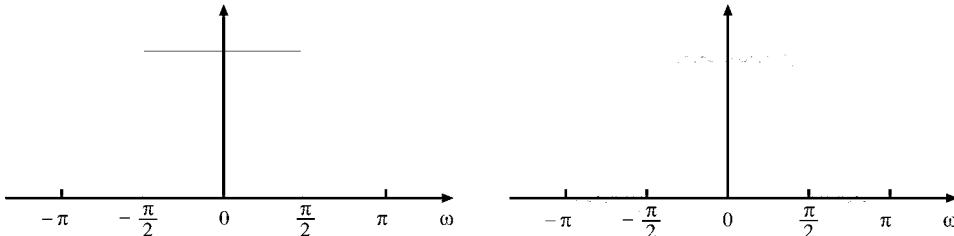


Figure 2.1: Ideal lowpass filter and best mean-square approximation with 20 terms.

We begin with an ideal filter, which has no transition band. Its responses are exactly $H(\omega) = 1$ and $H(\omega) = 0$. This is often called a *brick wall filter*, because of the step function in its graph. The response from an ideal lowpass filter is shown in Figure 2.1. This is a halfband filter, with sharp cutoff at $\omega = \frac{\pi}{2}$. The response $H(\omega) = \sum h(k)e^{-ik\omega}$ is 2π -periodic, and the ideal response is zero in the high frequency band from $\frac{\pi}{2}$ to π :

$$\text{Ideal lowpass} \quad H(\omega) = \sum_{-\infty}^{\infty} h(k)e^{-ik\omega} = \begin{cases} 1, & 0 \leq |\omega| < \frac{\pi}{2} \\ 0, & \frac{\pi}{2} \leq |\omega| < \pi. \end{cases} \quad (2.7)$$

What filter coefficients $h(k)$ produce this response? Multiply the equation for $H(\omega)$ by $e^{in\omega}$ and integrate from $-\pi$ to π :

$$\int_{-\pi}^{\pi} H(\omega)e^{in\omega} d\omega = \int_{-\pi}^{\pi} \left(\sum_{k=-\infty}^{\infty} h(k)e^{-ik\omega} \right) e^{in\omega} d\omega. \quad (2.8)$$

On the right side, there is an integral of $e^{-ik\omega}e^{in\omega}$ for each k . The great property of complex exponentials is that this integral is zero except when $k = n$:

$$\int_{-\pi}^{\pi} e^{-ik\omega}e^{in\omega} d\omega = \left[\frac{e^{i(n-k)\omega}}{i(n-k)} \right]_{-\pi}^{\pi} = 0 \quad \text{if } k \neq n. \quad (2.9)$$

The function in brackets is periodic. It has the same value at $-\pi$ and π . After substituting those limits, the definite integral is zero. ***The complex exponentials are orthogonal.***

Now equation (2.8) has only one term on the right, from $k = n$. Integrating this constant from $-\pi$ to π gives the result $2\pi h(n)$. This equals the left side:

$$\int_{-\pi}^{\pi} H(\omega)e^{in\omega} d\omega = 2\pi h(n). \quad (2.10)$$

The brick wall filter has $H(\omega) = 1$ on the part $|\omega| < \frac{\pi}{2}$:

$$2\pi h(n) = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} e^{in\omega} d\omega = \left[\frac{e^{in\omega}}{in} \right]_{-\frac{\pi}{2}}^{\frac{\pi}{2}} = \frac{2}{n} \sin \frac{\pi n}{2}. \quad (2.11)$$

The coefficients in the ideal lowpass filter are samples of a sinc function:

$$h(n) = \frac{\sin \frac{\pi n}{2}}{\pi n} = \begin{cases} \frac{1}{2}, & n = 0 \\ \pm \frac{1}{\pi n}, & n \text{ odd} \\ 0, & n \text{ even, } n \neq 0. \end{cases} \quad (2.12)$$

The halfband cutoff has produced a halfband filter! The coefficient $h(0) = \frac{1}{2}$ is the “DC term” = average value of $H(\omega)$. All other even-numbered coefficients are $h(n) = 0$. When $H(\omega)$ is antisymmetric around the halfband frequency $\omega = \frac{\pi}{2}$, the filter is always halfband.

For odd n , the numbers $h(n)$ alternate sign and decay slowly:

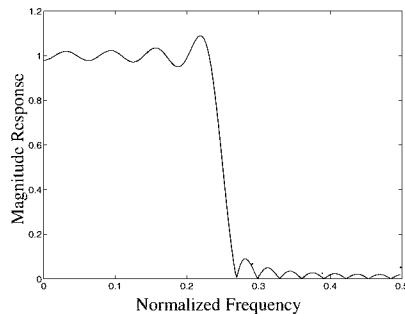
$$h(1) = h(-1) = -\frac{1}{\pi}, \quad h(3) = h(-3) = \frac{1}{3\pi}, \quad h(5) = h(-5) = -\frac{1}{5\pi}, \dots$$

The series that adds up to the brick wall (= square wave = ideal lowpass response) is

$$H(\omega) = \frac{1}{2} - \frac{e^{i\omega} + e^{-i\omega}}{\pi} + \frac{e^{i3\omega} + e^{-i3\omega}}{3\pi} - \frac{e^{i5\omega} + e^{-i5\omega}}{5\pi} + \dots \quad (2.13)$$

At $\omega = \frac{\pi}{2}$, the only nonzero term is halfway down the brick wall: $H\left(\frac{\pi}{2}\right) = \frac{1}{2}$. Most important is the behavior close to this jump at $\omega = \frac{\pi}{2}$, as shown in Figure 2.1 and below. Suppose we chop off the series after N terms:

The ripple at $\omega = \frac{\pi}{2}$ gets narrower as $N \rightarrow \infty$ but its height approaches a constant (about 0.09). This is the ***Gibbs phenomenon***.



This Gibbs phenomenon can be a disaster numerically. The ripple represents error. It is expensive to take a large number of terms and impossible to take all terms. A finite N gives the best approximation in the *mean square sense* — but the tall ripple remains. This “sidelobe” shows up as an echo in audio filtering and as a ghost in image processing. Practical design turns toward **equiripple filters**, which have many ripples of equal height. This design minimizes the maximum ripple height instead of the total ripple energy.

Note however that equiripple filters do not behave well in iteration. *They do not lead to good wavelets.*

Minimax filter design is implemented by the Parks-McClellan algorithm, which computes best approximations to ideal filters. Those filters have a passband, a stopband, and a “don’t care” transition band. The ripple heights (maximum errors) decay exponentially with the filter length N . If the acceptable error is specified, there is a formula for N (see *equiripple* in the Glossary). An alternative is *eigenfilter design*. For many problems this allows a simple mean square calculation, but without the big sidelobe from the Gibbs phenomenon.

Historical note. It is surprising to read the original paper by Gibbs. He completely missed the Gibbs phenomenon. His correction published later was even shorter — about three important lines. This correction must have the highest signal to noise ratio in the history of science.

Fourier’s Series by J. Willard Gibbs [*Nature*, vol. LIX, p. 200, December 29, 1898.]

... Let us write $f_n(x)$ for the sum of the first n terms of the series

$$\sin x - \frac{1}{2} \sin 2x + \frac{1}{3} \sin 3x - \frac{1}{4} \sin 4x + \text{etc.}$$

As n increases without limit, the curve defined by $y = 2f_n(x)$ approaches a limiting form, which may be thus described. Let a point move from the origin in a straight line at an angle of 45° with the axis of X to the point (π, π) , thence vertically in a straight line to the point $(\pi, -\pi)$, thence obliquely in a straight line to the point $(3\pi, \pi)$. The broken line thus described (continued indefinitely forwards and backwards) is the limiting form of the curve as the number of terms increases indefinitely....

Correction [in *Nature*, vol. LIX, p. 606, April 27, 1899.]

I should like to correct a careless error which I made in describing the limiting form of the family of curves represented in the equation

$$y = 2 \left(\sin x - \frac{1}{2} \sin 2x \cdots \pm \frac{1}{n} \sin nx \right) \quad (2.14)$$

as a zigzag line consisting of alternate inclined and vertical portions. The inclined portions were correctly given, but the vertical portions, which are bisected by the axis of X , extend beyond the points where they meet the inclined portions, their total lengths being expressed by four times the definite integral $\int_0^\pi \frac{\sin u}{u} du \dots$. But this limiting form of the graphs of the functions expressed by the sum is different from the graph of the function expressed by the limit of that sum.

I think this distinction important, for (with exception of what relates to my unfortunate blunder described above) whatever differences of opinion have been expressed on this subject seem due, for the most part, to the fact that some writers have had in mind the *limit of the graphs*, and others the *graph of the limit* of the sum.

The Gibbs phenomenon means that convergence to a brick wall is *not uniform*, as N increases. The coefficients $h(n)$ approach zero but the sum of absolute values $1 + \frac{1}{3} + \frac{1}{5} + \dots$

is infinite. These are multiplied by $\pm \sin n\omega$, so the actual series for $H(\omega)$ does not blow up. But the slow $1/n$ decay prevents uniform convergence and allows the large sidelobe. This ripple always appears in the Fourier series near a jump discontinuity.

Ideal Filter with Downsampling

In the time domain, $\mathbf{h}(n)$ is on the n th diagonal of the filter matrix \mathbf{H} . Writing a and b in place of $\mathbf{h}(1) = \mathbf{h}(-1)$ and $\mathbf{h}(3) = \mathbf{h}(-3)$, three rows are

$$\mathbf{H} = \begin{array}{ccccccccccccc} \cdots & 0 & b & 0 & a & 0.5 & a & 0 & b & 0 & \cdots \\ & \cdots & 0 & b & 0 & a & 0.5 & a & 0 & b & 0 & \cdots \\ & & \cdots & 0 & b & 0 & a & 0.5 & a & 0 & b & 0 & \cdots \end{array}$$

Those rows are not orthogonal! The dot product of the first two rows is a . Only an allpass filter has orthonormal rows (and columns). Then $|H(\omega)| = 1$ for all ω .

What is fundamental for this book is that *the even-numbered rows of \mathbf{H} are orthogonal*. When the downsampling operator $(\downarrow 2)$ removes half of the rows, this leaves a **double shift** in the remaining rows — the rows of $(\downarrow 2)\mathbf{H}$:

$$(\downarrow 2)\mathbf{H} = \begin{array}{ccccccccccccc} \cdots & 0 & b & 0 & a & 0.5 & a & 0 & b & 0 & \cdots \\ & \cdots & 0 & b & 0 & a & 0.5 & a & 0 & b & 0 & \cdots \\ & & \cdots & 0 & b & 0 & a & 0.5 & a & 0 & b & 0 & \cdots \end{array}$$

Orthogonality is not so clear in this time domain. Moving into the frequency domain, the double shift is a multiplication by $e^{-i2\omega}$ and row 0 of \mathbf{H} is orthogonal to row 2:

$$(\text{row 0}) \cdot (\text{row 2}) = \int_{-\pi}^{\pi} H(\omega) \overline{e^{-i2\omega} H(\omega)} d\omega = \int_{-\pi/2}^{\pi/2} e^{i2\omega} d\omega = 0.$$

Similarly row 0 is orthogonal to row 4, because the integral of $e^{i4\omega}$ is zero. This integral is over the half-period where $H(\omega) = 1$. The integral of $e^{i\omega}$ is not zero over this half-period, and row 0 of \mathbf{H} is not orthogonal to row 1.

The Ideal Highpass Filter

Haar's lowpass filter has coefficients $\frac{1}{2}$ and $\frac{1}{2}$, where the highpass filter has $\frac{1}{2}$ and $-\frac{1}{2}$. Those are clearly orthogonal. Now the ideal lowpass filter has infinitely many coefficients. We want to construct a highpass filter \mathbf{H}_1 , so that the rows of $(\downarrow 2)\mathbf{H}_1$ will be orthogonal to the rows of $(\downarrow 2)\mathbf{H}$.

It is easy to make $H_1(\omega)$ orthogonal to the ideal lowpass $H(\omega)$. We set $H_1 = 1$ in the intervals where $H = 0$:

$$\text{The ideal } H_1(\omega) \text{ can be } \begin{cases} 0 & \text{when } 0 \leq |\omega| < \frac{\pi}{2} \\ 1 & \text{when } \frac{\pi}{2} \leq |\omega| < \pi. \end{cases}$$

Shifted by π , the ideal $H(\omega)$ produces $H_1(\omega)$. In the time domain, that shift by π reverses the signs of the *odd-numbered* components (because $\mathbf{h}(k)$ changes to $\mathbf{h}(k)e^{ik\pi}$):

$$\mathbf{h}_1(0) = \frac{1}{2}, \quad \mathbf{h}_1(1) = \mathbf{h}_1(-1) = +\frac{1}{\pi}, \quad \mathbf{h}_1(3) = \mathbf{h}_1(-3) = -\frac{1}{3\pi}, \dots$$

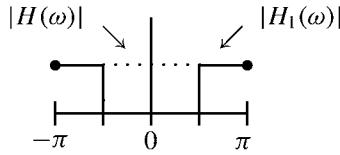


Figure 2.2: Ideal magnitude frequency responses: low $|H(\omega)|$ and high $|H_1(\omega)|$.

The graphs of $|H(\omega)|$ and $|H_1(\omega)|$ are in Figure 2.2. We explain below why absolute values suddenly appeared. Whatever the phase, orthogonality is sure because of no overlap.

We note that $|H(\omega)| + |H_1(\omega)| = 1$. This is not the identity that really matters. It is the sum of squares that applies not only in this case but in all orthogonal filter banks:

$$|H(\omega)|^2 + |H(\omega + \pi)|^2 \equiv 1. \quad (2.15)$$

For the Haar example this is the identity $\cos^2 \frac{\omega}{2} + \sin^2 \frac{\omega}{2} = 1$. The ideal case is deceptive because $1 = 1^2$ and $0 = 0^2$. The orthogonality requirement (2.15) will be established in Section 5.2.

The Alternating Flip (with Odd Shift)

There is an unusual point about the step from \mathbf{H} to \mathbf{H}_1 . In the time domain this usually comes from three operations on the coefficients $\mathbf{h}(n)$: *reverse the order, alternate the signs, and shift by 1* (or any odd N). This takes the lowpass coefficients $\mathbf{h}(n)$ into an orthogonal highpass sequence:

$$\text{Alternating flip } \mathbf{h}_1(n) = (-1)^n \mathbf{h}(N-n). \quad (2.16)$$

For a finite sequence $\mathbf{h}(0), \mathbf{h}(1), \dots, \mathbf{h}(N)$ — assuming N is odd! — you immediately see the flip in the next figure and the orthogonality between rows:

$$\begin{array}{ccccccc} \text{low} & \mathbf{h}(0) & \mathbf{h}(1) & \cdots & \mathbf{h}(N-1) & \mathbf{h}(N) \\ \text{high} & \mathbf{h}(N) & -\mathbf{h}(N-1) & \cdots & \mathbf{h}(1) & -\mathbf{h}(0) \end{array}$$

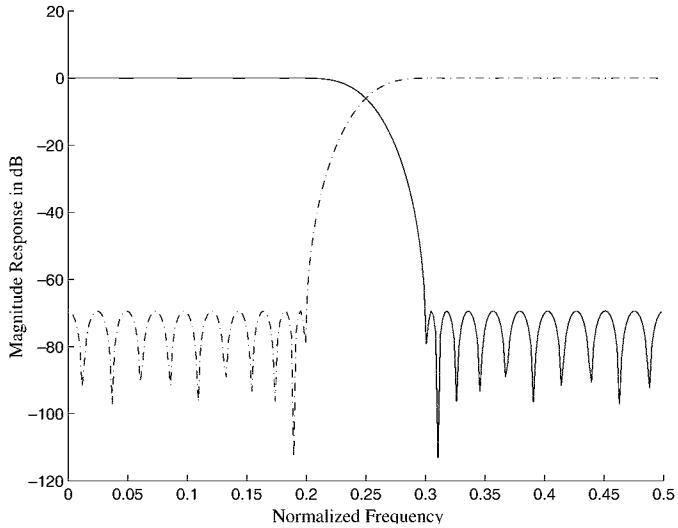
Important: There is also orthogonality of double shifts, as in

$$\begin{array}{ccccccc} \text{shift low by 2} & \mathbf{h}(2) & \mathbf{h}(3) & \cdots & \mathbf{h}(N-1) & \mathbf{h}(N) \\ \text{high} & \mathbf{h}(N) & -\mathbf{h}(N-1) & \cdots & \mathbf{h}(3) & -\mathbf{h}(2) \end{array}$$

$\mathbf{h}(2)\mathbf{h}(N)$ cancels $-\mathbf{h}(N)\mathbf{h}(2)$. This happens for all double shifts. It does not usually happen for single shifts! A single shift of Haar to $0, \frac{1}{2}, \frac{1}{2}$ is not orthogonal to the highpass $\frac{1}{2}, -\frac{1}{2}, 0$. Double shifts are all we care about, because it is double-shifted rows in $(\downarrow 2)\mathbf{H}$ that are orthogonal — and now the highpass rows in $(\downarrow 2)\mathbf{H}_1$ complete the filter bank.

The alternating flip is the key to orthogonality. In the frequency domain it has three steps: *Multiply by $e^{i\omega}$ to shift, take complex conjugates to flip, shift by π to alternate signs*:

$$H_1(\omega) = e^{-i\omega} \overline{H(\omega + \pi)}. \quad (2.17)$$



Theorem 2.1 *The alternating flip makes the rows of $(\downarrow 2)\mathbf{H}$ orthogonal to the rows of $(\downarrow 2)\mathbf{H}_1$.*

This was verified by eye, in the low and high rows above. You can verify it again for the ideal filters with infinitely many \mathbf{h} 's:

$$\begin{array}{ccccccccccccc} \text{row 0 of } \mathbf{H} & \cdots & b & 0 & a & 0.5 & a & 0 & b & 0 & \cdots \\ \text{row 0 of } \mathbf{H}_1 & \cdots & 0 & -b & 0 & -a & 0.5 & -a & 0 & -b & \cdots \end{array}$$

In the frequency domain we will see orthogonality in another form:

$$H(\omega)\overline{H_1(\omega)} + H(\omega + \pi)\overline{H_1(\omega + \pi)} = H(\omega)e^{i\omega}H(\omega + \pi) + H(\omega + \pi)e^{i(\omega+\pi)}H(\omega) \equiv 0. \quad (2.18)$$

All great, but for the ideal filters there is a very strange point. There was no odd shift! From the brick wall $H(\omega)$ on $[-\frac{\pi}{2}, \frac{\pi}{2}]$, we shifted by π to build the highpass brick wall. The wall is real, so conjugation has no effect. Still there should have been a phase shift from $e^{i\omega}$ and there wasn't.

Apparently row 0 of \mathbf{H} is orthogonal to $[-b \ 0 \ -a \ .5 \ -a \ 0 \ -b \ 0 \ \dots]$ without the shift. This unusual point occurs because $H_1(\omega)$ does not overlap $H(\omega)$. We can give $H_1(\omega)$ any phase we desire. It was natural to make it real. It is more consistent to include the phase shift $e^{i\omega}$.

This oddity (actually it is a lack of oddity) will reappear below for ideal wavelets. The sinc wavelets should have a shift in time, from the phase factor $e^{i\omega}$ —but generally they are taken from the unshifted \mathbf{H}_1 . Before turning to scaling functions and wavelets, we include a short discussion of the Sampling Theorem—to recover a band-limited function $x(t)$ from its samples. This famous theorem appears everywhere, so we focus on a particular aspect involving $(\downarrow 2)$.

Shannon (Down-)Sampling Theorem

Our main theme for filter banks is perfect reconstruction of all signals. With two filters this will be achieved in spite of downsampling. The Sampling Theorem restricts the input to a subspace of band-limited signals. Then *one* downsampled output is enough to recover the input.

The signal lies in the lower halfband $|\omega| < \frac{\pi}{2}$; no higher frequencies are allowed. In an ideal filter bank, nothing comes out of the highpass channel. Full information must be in $(\downarrow 2)\mathbf{H}\mathbf{x}$. For a half-range of *input frequencies*, we only need a half-range of *output samples*.

Suppose the output $(\downarrow 2)\mathbf{x}$ is an impulse $\delta = (\dots, 0, 1, 0, \dots)$. What was \mathbf{x} ? A first suggestion is $\mathbf{x} = \delta$, since $(\downarrow 2)\delta$ is equal to δ . But this is wrong — because δ is not band-limited. The impulse has all frequencies in equal amounts.

The correct input to yield $(\downarrow 2)\mathbf{x} = \delta$ has a *halfband of frequencies in equal amounts*. The graph of $X(\omega)$ is a square wave:

$$X(\omega) = \begin{cases} 2 & \text{for } 0 \leq |\omega| < \frac{\pi}{2} \\ 0 & \text{for } \frac{\pi}{2} \leq |\omega| < \pi. \end{cases} \quad (2.19)$$

Downsampling doubles every frequency, so $(\downarrow 2)\mathbf{x}$ has a full band of frequencies in equal amounts. It equals δ as required.

What signal \mathbf{x} has the transform $X(\omega) = \text{square wave}$? The inverse transform is

$$\mathbf{x}(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{in\omega} d\omega = \frac{1}{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} e^{in\omega} d\omega = \frac{2}{n\pi} \sin \frac{n\pi}{2}. \quad (2.20)$$

The input before downsampling has been recovered as the **sinc vector**:

$$\mathbf{x}_{\text{sinc}}(n) = \frac{\sin \frac{n\pi}{2}}{\frac{n\pi}{2}} \quad \text{with the convention } \mathbf{x}_{\text{sinc}}(0) = 1. \quad (2.21)$$

Downsampling gives δ because if n is even then $\sin \frac{n\pi}{2} = 0$.

The recovery problem is now solved when $(\downarrow 2)\mathbf{x}$ is δ . The band-limited input was \mathbf{x}_{sinc} . But every output is a combination of impulses at different times:

$$(\downarrow 2)\mathbf{x} = (\dots, \mathbf{x}(0), \mathbf{x}(2), \dots) = \dots + \mathbf{x}(0)\delta + \mathbf{x}(2)\delta + \dots$$

Delaying the input by 2 delays $(\downarrow 2)\mathbf{x}$ by 1. This leads us to the correct input:

Downsampling Theorem The halfband signal that produces $(\downarrow 2)\mathbf{x} = (\dots, \mathbf{x}(0), \mathbf{x}(2), \dots)$ is

$$\mathbf{x}(n) = \dots + \mathbf{x}(0)\mathbf{x}_{\text{sinc}}(n) + \mathbf{x}(2)\mathbf{x}_{\text{sinc}}(n-2) + \dots = \sum_{-\infty}^{\infty} \mathbf{x}(2k) \frac{\sin((n-2k)\frac{\pi}{2})}{(n-2k)\frac{\pi}{2}}.$$

For even n all terms are zero, except $2k = n$ which yields $\mathbf{x}(n)$. The input signal \mathbf{x} is halfband because \mathbf{x}_{sinc} and its shifts are halfband.

By changing $2k$ to k we would get the ordinary Shannon Sampling Theorem.

Sinc Wavelets (Shannon Wavelets)

In the Haar example, the lowpass \mathbf{H} is the averaging filter (coefficients $\frac{1}{2}$ and $\frac{1}{2}$). By iteration we reached a continuous-time box function. That function satisfies the dilation equation with coefficients 1 and 1. The box is the “scaling function,” and there is a corresponding up-down Haar wavelet.

Now \mathbf{H} is the ideal lowpass filter. Its scaling function is the sinc function $\phi(t) = \frac{\sin \pi t}{\pi t}$. Since the ideal filter is IIR, the steps in our discussion will go a little more steeply. After this page, our main theme is FIR filter banks.

Section 6.4 has an infinite product formula for the Fourier transform of $\phi(t)$:

$$\widehat{\phi}(\omega) = \prod_{j=1}^{\infty} H\left(\frac{\omega}{2^j}\right). \quad (2.22)$$

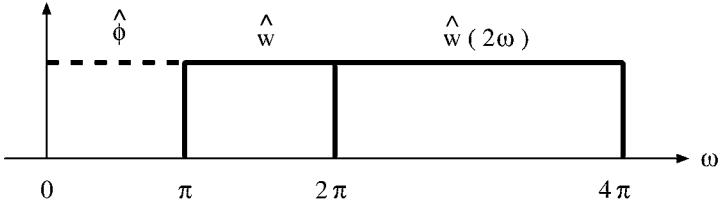
In the ideal case, every factor $H(\omega/2^j)$ is one for $|\omega| < \pi$. The j th factor is zero for $2^{j-1}\pi \leq |\omega| < 2^j\pi$. The infinite product gives a box function for $\widehat{\phi}(\omega)$, stretching from $-\pi$ to π . (Note that $H(\omega)$ is 2π -periodic, but the infinite product $\widehat{\phi}(\omega)$ is not.) The inverse transform $\phi(t)$ of this box is a sinc function:

$$\text{The ideal scaling function is } \phi(t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\omega t} d\omega = \frac{\sin \pi t}{\pi t}.$$

The wavelet $w(t) = \sum 2h_1(k)\phi(2t - k)$ comes from one application of the highpass filter (with downsampling) to $\phi(t)$. In the frequency domain this is

$$\widehat{w}(\omega) = H_1\left(\frac{\omega}{2}\right)\widehat{\phi}\left(\frac{\omega}{2}\right) = \begin{cases} 1 & \text{for } \pi \leq |\omega| < 2\pi \\ 0 & \text{otherwise.} \end{cases} \quad (2.23)$$

The ideal filter bank cuts the frequency band in half. The upper half of the band goes through the highpass filter (discrete time). In continuous time it is a combination of wavelets. The lower half of the frequency band goes through the lowpass filter (discrete time). In continuous time this half is a combination of scaling functions ϕ —ready to be split again into wavelets and scaling functions at the next finer scale. We have an *octave decomposition = logarithmic decomposition = “constant-Q decomposition”* of the line of frequencies:



Note 1 Meyer smoothed out this ideal picture to produce band-limited wavelets with fast decay (IIR of course). The key is to keep $\sum |\widehat{w}_{\text{Meyer}}(2^n\pi\omega)|^2 \equiv 1$. This can be done smoothly with overlap of nearest neighbors only [D]. All band-limited wavelets have two bumps ($\pm\omega$) like the Shannon and Meyer wavelets.

Problem Set 2.2

1. Show that the inverse transform of \widehat{w} in (2.23) is the *sinc wavelet* $w(t) = 2\text{sinc}(2t) - \text{sinc}(t)$.
2. Find the shifted sinc wavelet by inverse transform when the factor $e^{i\omega}$ is included in $H_1(\omega)$. This is the odd shift that we normally need for orthogonality of w to ϕ .
3. What are the coefficients $h(n)$ for the ideal quarterband filter, with $H(\omega) = 1$ on $[-\frac{\pi}{4}, \frac{\pi}{4}]$? What scaling function $\widehat{\phi}(\omega)$ comes from the infinite product formula? Is $\phi(t)$ orthogonal to its translates?
4. (**Important**) Show that $H(\omega)$ has halfband symmetry (odd around its value at $\omega = \frac{\pi}{2}$) when $h(n)$ is a **halfband filter**. This means $(\downarrow 2)h = -$.
5. Let $h_{LP}(n)$ denote an ideal lowpass filter with cutoff frequency ω_c .

- (a) Compute $H_{LP}(e^{j\omega})$ and normalized $\mathbf{h}_{LP}(n)$ such that $H_{LP}(e^{j0}) = 1$.
- (b) The ideal highpass filter $\mathbf{h}_{HP}(n)$ can be designed by
- $$\mathbf{h}_{HP}(n) = \begin{cases} 1 - \mathbf{h}_{LP}(n); & n = 0, \\ -\mathbf{h}_{LP}(n); & \text{otherwise.} \end{cases}$$
- Find $H_{HP}(e^{j\omega})$.
- (c) Design a highpass filter with cutoff frequency at $\pi - \omega_c$, using $\mathbf{h}_{LP}(n)$.
6. Let $H(z) = (1 - 2z^{-1} + 3z^{-2} - 3z^{-3} + 2z^{-4} - z^{-5})$. Compute $|H(e^{j\omega})|$ and the phase response $\phi(\omega)$ and the group delay $\phi'(\omega)$.
7. Let $H(z)$ be an FIR lowpass filter of length $(N + 1)$. Define $G_1(z) = z^{-N} H(z^{-1})$, $G_2(z) = H(-z)$ and $G_3(z) = z^{-N} H(-z^{-1})$.
- (a) What are $\mathbf{g}_1(n)$, $\mathbf{g}_2(n)$ and $\mathbf{g}_3(n)$ in terms of $\mathbf{h}(n)$?
 - (b) If $H(z)$ is an even-length symmetric filter, what is the symmetry or antisymmetry of $G_1(z)$, $G_2(z)$ and $G_3(z)$?
 - (c) If z_0 is a zero of $H(z)$, what are the corresponding zeros of $G_1(z)$, $G_2(z)$ and $G_3(z)$?
 - (d) What are the relations of $|H(e^{j\omega})|$, $|G_1(e^{j\omega})|$, $|G_2(e^{j\omega})|$, and $|G_3(e^{j\omega})|$?
8. Show that $G(z) = H(z)H(z^{-1})$ is a symmetric filter. What type of filter is $G(z)$ (lowpass, bandpass, highpass) if $H(z)$ is highpass? What is the (constant) phase of $G(z)$?
9. We have stated that $\delta_p = \delta_s$ in a halfband filter. Prove this.

2.3 Lowpass and Highpass Filter Design

The previous section dealt with ideal filters (necessarily IIR). This section deals with real FIR filters—often symmetric or antisymmetric (thus linear phase). We indicate the goals of filter design and we briefly discuss design methods.

For ideal brick walls, the transition from $H(\omega) = 1$ to $H(\omega) = 0$ happens instantly. For FIR filters this is not possible. It is important to see an actual magnitude response graph (Figure 2.3). In normal scale, we can observe details in the passband but not in the stopband. In logarithmic scale (dB scale, plotting $20 \log_{10} |H|$), it is the other way around. The stopband details are visible in dB scale but the passband details are lost.

Before moving to good lowpass filters, we review two very short and rather poor filters. This gives us a chance to emphasize the four types of linear phase filters—odd and even length, symmetric and antisymmetric.

Example 2.6. The impulse response is $\mathbf{h} = (\frac{1}{2}, \frac{1}{2})$. The frequency response is

$$H(\omega) = \frac{1}{2}(1 + e^{-i\omega}) = \left(\frac{e^{i\omega/2} + e^{-i\omega/2}}{2} \right) e^{-i\omega/2} = \left(\cos \frac{\omega}{2} \right) e^{-i\omega/2}. \quad (2.24)$$

In that last form you see the magnitude $|H(\omega)| = \cos \frac{\omega}{2}$ and the phase $\phi(\omega) = -\frac{\omega}{2}$. The magnitude is $\cos 0 = 1$ at zero frequency—this is a lowpass filter. The magnitude drops to zero at $\omega = \pi$. The phase $\phi(\omega)$ is the angle $-\omega/2$ in the polar form $re^{i\phi}$. This phase function $-\frac{\omega}{2}$ is linear in ω . The noninteger 1/2 reflects the fact that the coefficients in \mathbf{h} are symmetric about the “1/2” position.

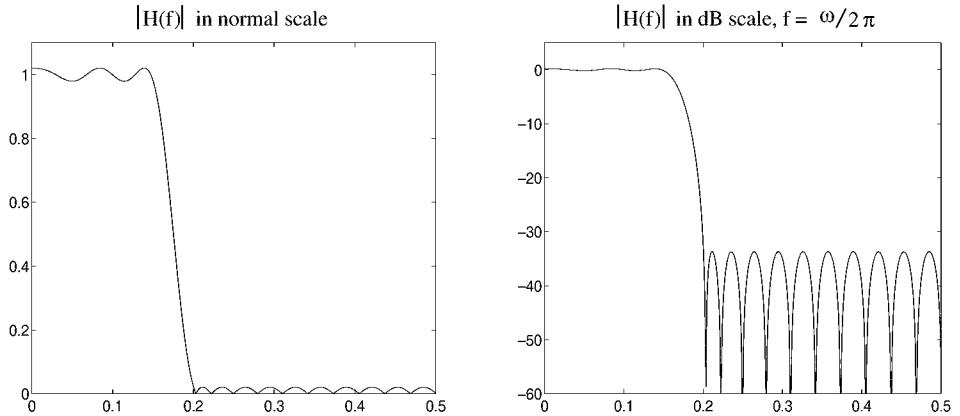


Figure 2.3: Magnitude response of a lowpass filter in normal and dB scale.

We will compute the phase $\phi(\omega)$ for other filters before analyzing its significance. Here we only mention: linear phase is a desirable property.

Example 2.7. By cascading the previous example we square its matrix H and we square its frequency response. The new filter coefficients are in

$$\begin{bmatrix} 0.5 & & \\ 0.5 & 0.5 & \\ 0 & 0.5 & 0.5 \\ \ddots & \ddots & \ddots \end{bmatrix}^2 = \begin{bmatrix} 0.25 & & \\ 0.50 & 0.25 & \\ 0.25 & 0.50 & 0.25 \\ \ddots & \ddots & \ddots \end{bmatrix} = H_{\text{new}}.$$

The new impulse response is $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$. It is the convolution of $(\frac{1}{2}, \frac{1}{2})$ with itself. Cascading filters means convolution of impulse responses (time domain). In the frequency domain we multiply responses $H(\omega)$:

$$H_{\text{new}}(\omega) = [H_{\text{old}}(\omega)]^2 = \frac{1}{4}(1 + e^{i\omega})^2 = \left(\cos^2 \frac{\omega}{2}\right)e^{-i\omega}. \quad (2.25)$$

This squares the magnitude and doubles the phase. The new phase $-\omega$ is still linear. It corresponds to a time shift of 1. The impulse response is a unit delay of a symmetric \mathbf{h} . H_{new} is a shift (= delay) times a symmetric matrix.

Every FIR matrix can be made causal, by sufficiently many delays. A symmetric matrix (not causal) corresponds to phase = zero because $H(\omega)$ is *real*:

$$H_{\text{sym}}(\omega) = \frac{1}{4}e^{i\omega} + \frac{1}{2} + \frac{1}{4}e^{-i\omega} = \frac{1}{2}(1 + \cos \omega). \quad (2.26)$$

Was this cascade desirable? Neither H or H_{new} is very impressive. The frequency responses are far from ideal. $H_{\text{new}}(\omega)$ has better attenuation in the stopband, because the cosine is squared. But it is also smaller in the passband — farther away from the ideal $H \equiv 1$.

By alternating signs in the lowpass coefficients they become *highpass*: $\mathbf{h}_1 = (0.5, -0.5)$ and $\mathbf{h}_1 * \mathbf{h}_1 = (0.25, -0.50, 0.25)$. These are still linear phase. The alternation of signs is a phase shift (a *modulation*) by π . The first is antisymmetric but the second is still symmetric.

It is useful to tabulate the four types of linear phase filters with real coefficients. N can be odd or even. The coefficients can satisfy

$$\mathbf{h}(n) = \mathbf{h}(N - n) \text{ for symmetric, } \mathbf{h}(n) = -\mathbf{h}(N - n) \text{ for antisymmetric.}$$

The linear phase $\phi(\omega) = -\omega N/2$ and the (real) amplitude response $H_R(\omega)$ are seen in $H(e^{j\omega}) = ce^{-j\omega N/2} H_R(\omega)$. The table shows that with odd N , symmetry guarantees a zero at $\omega = \pi$ and antisymmetry guarantees a zero at $\omega = 0$. The responses $H_R(\omega)$ in the table have factors $\cos \frac{\omega}{2}$ and $\sin \frac{\omega}{2}$. Remember that the filter length (number of taps) is $N + 1$. Here $c = 1$ for symmetric and $c = j$ for antisymmetric filters.

Type 1	Type 2	Type 3	Type 4
even $N = 2K$ symmetric	odd $N = 2K + 1$ symmetric	even $N = 2K$ antisymmetric	odd $N = 2K + 1$ antisymmetric
$H_R = \sum_0^K b_n \cos n\omega$	$\cos \frac{\omega}{2} \sum_0^K b_n \cos n\omega$ zero at $\omega = \pi$	$\sin \omega \sum_0^{K-1} b_n \cos n\omega$ zeros at $\omega = 0, \pi$	$\sin \frac{\omega}{2} \sum_0^K b_n \cos n\omega$ zero at $\omega = 0$

Now consider the ideal lowpass filter with cutoff frequency ω_p . This is a band-limited filter in the frequency domain, therefore its support in the time domain is infinite. It must be IIR. Its impulse response is $\mathbf{h}_I(n) = \pi \frac{\sin(\omega_p n)}{\omega_p n}$. Since the time-support is infinite, one needs to approximate it by a finite impulse response. The sequence $\mathbf{h}(n)$ becomes time-limited, therefore not band-limited. The magnitude response $|H(e^{j\omega})|$ typically has errors δ_p and δ_s in the passband and stopband (Figure 2.4). Those bands have cutoff frequencies ω_p and ω_s .

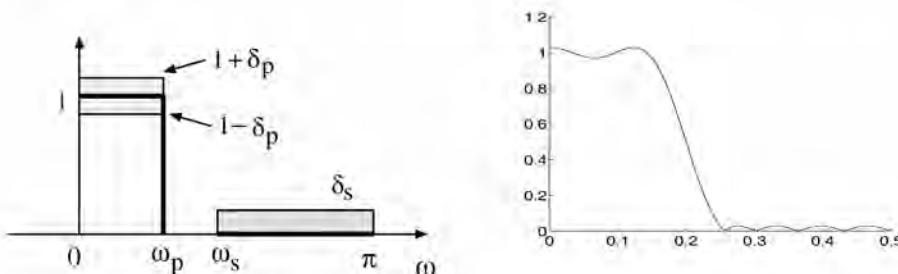


Figure 2.4: Idealized and typical magnitude responses of a lowpass filter.

Given ω_p , ω_s , and N , the errors δ_p and δ_s cannot both be small. Let $W = b/a$ be the relative error weighting. Increasing W in the design algorithm will decrease δ_s and increase δ_p . Figure 2.4 shows a magnitude response plot for a lowpass filter with length 19 and $\omega_p = 0.2\pi$ and $\omega_s = 0.3\pi$. The error weighting W is large (and thus yields a small stopband error).

In the sections below, several methods for the design of FIR digital filters are reviewed. These are based on windowing, minimax criteria, and weighted least squares.

Design by Windowing

The simplest way to truncate the ideal response \mathbf{h}_I is by a rectangular window:

$$\mathbf{h}(n) = \mathbf{h}_I(n)w(n) \quad \text{where} \quad w(n) = \begin{cases} 1; & |n| \leq N/2 \\ 0; & \text{otherwise.} \end{cases}$$

This $H(e^{j\omega})$ is the best least squares approximation to $H_I(e^{j\omega})$. But chopping off the impulse response manifests as passband and stopband ripples in the frequency response. As the window size increases, the ripples get closer to the cutoff frequency ω_c , but these error heights do not decrease. This is the notorious Gibbs phenomenon. (Section 2.2 showed the magnitude response for halfband filters, when $\omega_c = \frac{\pi}{2}$. The solid line has $N = 30$, the dotted line has $N = 10$.) To design FIR filters with better error characteristics, we can smooth out the window $w(n)$:

$$\begin{aligned} \text{Hamming window} \quad w(n) &= \alpha + (1 - \alpha) \cos\left(\frac{2\pi n}{N}\right) & |n| \leq \frac{N-1}{2} \\ \text{Hanning window} \quad w(n) &= \frac{1}{2} + \frac{1}{2} \cos\left(\frac{2\pi n}{N}\right) & |n| \leq \frac{N-1}{2} \\ \text{Kaiser window} \quad w(n) &= \frac{1}{2} I_0\left[\beta \sqrt{1 - \left(\frac{2n}{N}\right)^2}\right] / I_0(\beta) & |n| \leq \frac{N}{2} \end{aligned}$$

The parameter β in the Kaiser window controls the attenuation of the lowpass filter. $I_0(x)$ is the modified Bessel function and practical designs use about 20 terms of

$$I_0(x) = 1 + \sum_{k=1}^{\infty} \left[\frac{(0.5x)^2 k}{k!} \right]^2.$$

Minimax Criteria (Equiripple Filter)

The filter with the smallest maximum error in passband and stopband is an *equiripple filter*. The equal heights of the ripples (and the number of ripples) assure that the error cannot be reduced—some ripple sizes will go up if others go down. A polynomial of degree N cannot have alternating signs at all ripples. The Remez algorithm to equalize the ripples was adapted to filter design by Parks and McClellan.

Figure 2.5 shows the magnitude response plot of an equiripple lowpass filter. Given a frequency specification in terms of cutoff frequencies (ω_p, ω_s), filter length ($N + 1$) and relative errors (δ_p, δ_s), the equiripple filter has the smallest maximum error in the frequency interval $0 \leq \omega \leq \pi$. The design algorithm for equiripple filters is the Remez exchange (McClellan-Parks) algorithm.

The order (N) of an equiripple filter is estimated by

$$N \approx \frac{-20 \log_{10} \sqrt{\delta_1 \delta_2} - 13}{14.6 \Delta f} \quad (2.27)$$

where $\Delta f = (\omega_s - \omega_p)/2\pi$ is the transition band.

Equiripple designs are optimal in important respects—but not optimal for iteration. The reason is that they have at most one zero at $\omega = \pi$. The sampling operators ($\downarrow 2$) will mix up the frequency bands that an equiripple filter so carefully separates! *The Daubechies filters go to the other extreme—no ripples at all and maximum flatness at $\omega = \pi$.* Then iteration of

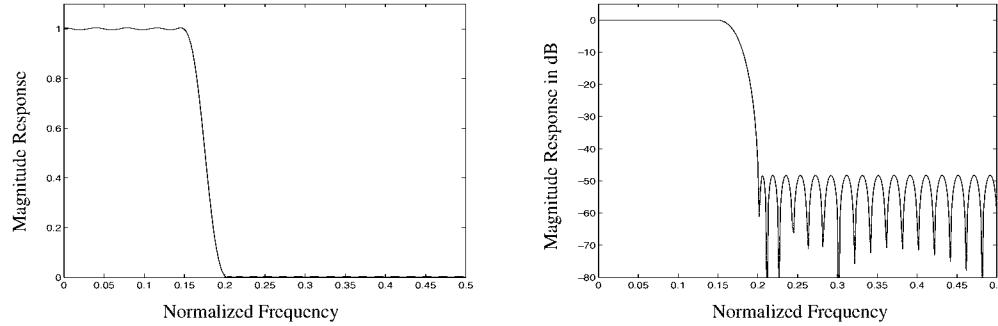


Figure 2.5: Magnitude response of an equiripple filter: normal and dB scales.

($\downarrow 2$) \mathbf{H} is very stable. But Section 5.5 will show that the transition band $\Delta\omega$ widens from N^{-1} for equiripple to $N^{-\frac{1}{2}}$ for the maxflat Daubechies filters.

A compromise is certainly possible, constraining the minimax design to have a fixed number p of zeros at $\omega = \pi$.

Weighted Least Squares (Eigenfilters)

The eigenfilter approach chooses the filter \mathbf{H} to minimize a (weighted) integral error

$$E = \int |D(\omega) - H(e^{j\omega})|^2 (\text{weight}) d\omega \quad (2.28)$$

The integral is over the passband and stopband, not the transition band. $D(\omega)$ is the desired frequency response, possibly one and zero in the two bands. The weighting function is optional. The goal is to express the error as a quadratic form $E = \mathbf{h}^T \mathbf{P} \mathbf{h}$. The unknown filter coefficients are in the vector \mathbf{h} .

The matrix \mathbf{P} is symmetric positive definite because $E > 0$. If the normalization has the form $\mathbf{h}^T \mathbf{h} = 1$ then the minimization is an eigenvalue problem $\mathbf{P} \mathbf{h} = \lambda_{\min} \mathbf{h}$ in linear algebra:

$$\text{The minimum of } E = \frac{\mathbf{h}^T \mathbf{P} \mathbf{h}}{\mathbf{h}^T \mathbf{h}} \text{ is } \lambda_{\min}(\mathbf{P}).$$

If the normalization is changed to $\mathbf{h}^T \mathbf{Q} \mathbf{h} = 1$, the eigenvalue problem $\mathbf{P} \mathbf{h} = \lambda \mathbf{Q} \mathbf{h}$ involves both matrices. When there are several quadratic constraints $\mathbf{h}^T \mathbf{Q}_k \mathbf{h} = 1$, we go beyond an eigenvalue problem — to the Quadratic Constrained Least Squares algorithm. Section 5.4 will apply this QCLS method to filter design. The applications of eigenfilters (one quadratic constraint) are very extensive, and we give two examples: lowpass filters and halfband filters.

Lowpass eigenfilter design: A symmetric filter $\mathbf{h}(n) = \mathbf{h}(2L - 1 - n)$ of length $2L$ has response

$$H(e^{j\omega}) = \sum_{n=0}^{2L-1} \mathbf{h}(n) e^{-j\omega n} = e^{-j\omega(L-\frac{1}{2})} \sum_0^{L-1} \mathbf{h}(n) c(\omega).$$

The last sum is $H_{\text{real}}(\omega) = \mathbf{h}^T \mathbf{c}(\omega)$ with $\mathbf{h} = [\mathbf{h}(0) \dots \mathbf{h}(L-1)]$. The vector $\mathbf{c}(\omega)$ has components $2 \cos(L - \frac{1}{2})\omega, \dots, 2 \cos \frac{\omega}{2}$. In the stopband, from ω_s to π , where $D = 0$ is the desired response, the error is

$$E_{\text{stop}} = \int H_{\text{real}}^2(\omega) d\omega = \mathbf{h}^T \int_{\omega_s}^{\pi} \mathbf{c}(\omega) \mathbf{c}(\omega)^T d\omega \mathbf{h} = \mathbf{h}^T \mathbf{P}_{\text{stop}} \mathbf{h}.$$

The entries of \mathbf{P}_{stop} are known integrals of cosines. In the passband from 0 to ω_p , we can normalize the desired constant response to be $D = \mathbf{h}^T \mathbf{c}(0)$. Then the passband error involves the difference between that desired response and the attained response $\mathbf{h}^T \mathbf{c}(\omega)$:

$$E_{\text{pass}} = \mathbf{h}^T \int_0^{\omega_p} (\mathbf{c}(0) - \mathbf{c}(\omega)) (\mathbf{c}(0) - \mathbf{c}(\omega))^T d\omega \mathbf{h} = \mathbf{h}^T \mathbf{P}_{\text{pass}} \mathbf{h}.$$

The entries of \mathbf{P}_{pass} are known integrals of $4[1 - \cos(n + \frac{1}{2})\omega][1 - \cos(m + \frac{1}{2})\omega]$.

We can weight the errors by $E = \alpha E_{\text{stop}} + (1-\alpha) E_{\text{pass}}$. The matrix whose lowest eigenvector is the best \mathbf{h} will be $\mathbf{P} = \alpha \mathbf{P}_{\text{stop}} + (1-\alpha) \mathbf{P}_{\text{pass}}$. Figure 2.6 shows the magnitude response of a lowpass filter of length 55 and cutoff frequencies $\omega_p = 0.2\pi$ and $\omega_s = 0.3\pi$. Here, the weight α is 0.5.

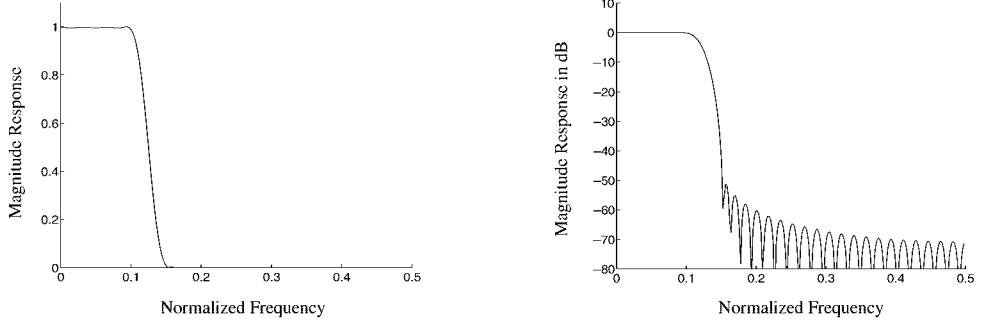


Figure 2.6: Magnitude response plot of a lowpass eigenfilter in normal and dB scale.

Halfband and Mth-band filter design: In many digital systems, a change in the sampling rate is essential for efficiency in real time. The design of suitable filters for the rate changing operations is important. When the subsampling is by a factor of $M = 2$, we are led to *halfband filters*. When we keep every M th sample, we need *Mth-band filters*. The center coefficient is $\mathbf{h}(0) = 1$ and all coefficients at multiples of M are zero:

$$\text{Mth-band: } \mathbf{h}(nM) = \delta(n) \text{ or in other words } (\downarrow M)\mathbf{h} = \delta. \quad (2.29)$$

The filter banks have two channels or M channels. The design begins with a halfband filter or an M th-band filter. This is the product filter of Section 4.1, which is factored into analysis times synthesis. We shift the filters to make them causal, for implementation. But for design we keep them centered, *and simply zero out the coefficients $\mathbf{h}(nM)$ for $n \neq 0$* . This zeros out the corresponding rows and columns of the error matrix \mathbf{P} in weighted least squares. The optimum \mathbf{h} is the lowest eigenvector of the reduced matrix \mathbf{P}_{red} , and this \mathbf{h} is *Mth-band*.

Design procedure: Given N , M , ω_p , ω_s for an M th-band filter, find \mathbf{P} using the eigenfilter formulation. Find \mathbf{P}_{red} by deleting the rows and columns of \mathbf{P} that correspond to zero coefficients in \mathbf{h} . Find the eigenvector \mathbf{h}_{red} corresponding to the minimum eigenvalue of \mathbf{P}_{red} . The optimal impulse response $\mathbf{h}(n)$ is obtained from $\mathbf{h}_{\text{red}}(n)$ by inserting the zero coefficients.

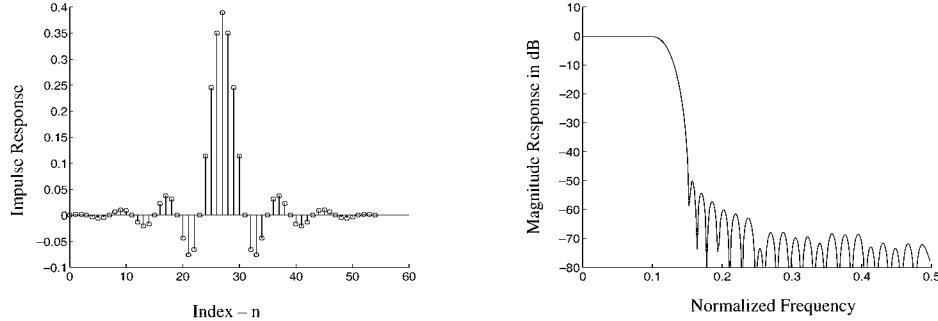


Figure 2.7: Impulse response and magnitude response of a 4th-band FIR filter.

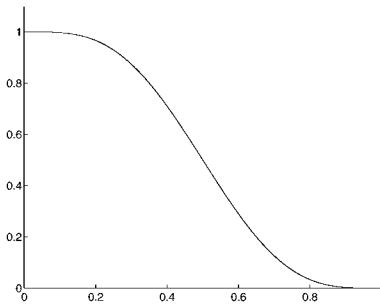
Figure 2.7 shows the magnitude response of an 4th-band filter with $\omega_p = 0.2\pi$ and $\omega_s = 0.3\pi$. Note that $\omega_p + \omega_s = 0.5\pi = \frac{2\pi}{M}$, as required for a symmetric 4th-band filter. The length is 55 (odd length is also required).

Maximally Flat Filter (Maxflat filter): The design of a maxflat filter begins with a maxflat polynomial $\tilde{H}(y)$. This polynomial of degree $2p - 1$ is determined by p conditions at $y = 0$ and at $y = 1$:

$$\tilde{H}^{(k)}(0) = \delta(k) \text{ and } \tilde{H}^{(k)}(1) = 0 \text{ for } 0 \leq k < p.$$

The p th order zero at $y = 1$ means that $\tilde{H}(y) = (1 - y)^p Q(y)$. Then the conditions at $y = 0$ determine $Q(y)$. The details are in Section 5.5, leading to the Daubechies filter by factoring this halfband filter. Here we highlight a remarkable result: $Q(y)$ consists of the first p terms of the series for $(1 - y)^{-p}$. With $p = 4$, for example, $\tilde{H}(y) = (1 - y)^4(1 + 4y + 10y^2 + 20y^3) \approx (1 - y)^4(1 - y)^{-4}$.

Those coefficients 1, 4, 10, 20 come from the binomial series for $(1 - y)^{-4}$. They also appear in $(1 + y + y^2 + y^3)^4$. They are binomial numbers. The product $\tilde{H}(y)$ then has three zero derivatives at $y = 0$. Its graph is in the figure below.



The relations among the variables y and ω and z are

$$\begin{aligned} y &= \frac{1 - \cos \omega}{2} = \frac{1}{4}(2 - z - z^{-1}) = -z\left(\frac{1 - z^{-1}}{2}\right)^2. \\ 1 - y &= \frac{1 + \cos \omega}{2} = \frac{1}{4}(2 + z + z^{-1}) = z\left(\frac{1 + z^{-1}}{2}\right)^2. \end{aligned} \quad (2.30)$$

The zeros at $y = 1$ become zeros at $\omega = \pi$ and at $z = -1$. The flatness at $y = 0$ becomes flatness at $\omega = 0$ and at $z = 1$. We give the frequency response for the centered halfband Daubechies filter:

$$H(e^{j\omega}) = \left(\frac{1 + \cos \omega}{2}\right)^p \sum_{k=0}^{p-1} \binom{p-1+k}{k} \left(\frac{1 - \cos \omega}{2}\right)^k. \quad (2.31)$$

It is a binomial exercise to show that $H = \frac{1}{2}$ at $\omega = \frac{\pi}{2}$. To find $H(z)$, substitute using equations (2.30). Then shift by z^{-p} to make the filter causal.

Note Another form of this polynomial (which is so important in wavelet theory) is the “Bernstein form”

$$H(e^{j\omega}) = \sum_{k=p+1}^{2p+1} \binom{2p+1}{k} \left(\frac{1 + \cos \omega}{2}\right)^k \left(\frac{1 - \cos \omega}{2}\right)^{2p+1-k}. \quad (2.32)$$

As p increases, $H(e^{j\omega})$ approaches the ideal lowpass response (one for $\cos \omega > 0$ and zero for $\cos \omega < 0$). The $p^{-1/2}$ width of the transition band is established in Section 5.5, together with the approximation of the zeros of $H(z)$.

The ideal is infinitely flat. Of course it needs infinitely many coefficients.

Problem Set 2.3

1. Truncate the ideal lowpass filter after three nonzero coefficients. What is this windowed filter $\mathbf{h}(n)$? Sketch the graph of $H(\omega)$.
2. Truncate the ideal lowpass filter after 4 terms and 8 terms. Draw the frequency responses $H(\omega)$ to see the Gibbs phenomenon.
3. Compare the graphs of the ideal brick wall filter truncated after 20 terms (rectangular window) and the Kaiser window $w(n)$. Choose a suitable Kaiser parameter β . What are the maximum errors in the passband?
4. Use MATLAB to design equiripple halfband filters of length 8 and 20. Compute the height of the ripples.
5. What is the frequency response for the maxflat Daubechies filter (2.31) with $p = 2$? Graph $H(\omega)$ by hand or by computer. What are its symmetries?
6. Graph the maxflat Daubechies filter response (2.31) for $p = 8$. What are the differences from the truncated ideal filter and the equiripple filter?
7. Construct a 4-tap lowpass filter that you approve of. What properties have you achieved?
8. Derive the Bernstein form (2.32) of the Daubechies polynomial $H(e^{j\omega})$ from her original form (2.31).
9. Formulate the eigenfilter design for a highpass filter with cutoff frequency ω_c .
10. Compute $\mathbf{h}(n)$ for the halfband Daubechies filter with $p = 5$. Verify that $H(e^{j\omega})$ has four zero derivatives at $\omega = 0$ and $\omega = \pi$.

2.4 Fourier Analysis

This section might be called “Notes on Fourier Analysis.” The subject is enormous—too large for our book! We pick out key points that are needed for signal processing and for wavelets.

The Fourier transform of a signal $\mathbf{x}(n)$ is a function $X(\omega)$. The signal is in the time domain, $X(\omega)$ is in the frequency domain. The time variable n is discrete, the frequency variable ω is continuous. $X(\omega)$ is 2π -periodic because each exponential $e^{in\omega}$ is 2π -periodic:

$$X(\omega) = \sum_{n=-\infty}^{\infty} \mathbf{x}(n)e^{-in\omega}. \quad (2.33)$$

Fourier analysis studies the connections between $\mathbf{x}(n)$ and $X(\omega)$ —how the properties of the signal are reflected in its transform. The **inverse Fourier transform** recovers $\mathbf{x}(n)$:

$$\mathbf{x}(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega)e^{in\omega} d\omega. \quad (2.34)$$

This formula synthesizes \mathbf{x} by combining the complex exponentials. To find $\mathbf{x}(N)$, multiply equation (2.33) by $e^{iN\omega}$ and integrate from $-\pi$ to π . The integral of $e^{-in\omega}$ times $e^{iN\omega}$ is zero except when $n = N$. That integral is 2π , leading to (2.34).

Fourier analysis usually starts with $f(t)$ and computes its coefficients. Signal processing starts with the coefficients $\mathbf{x}(n)$ and transforms to $X(\omega)$.

Note about orthogonality. Real vectors are orthogonal (perpendicular) when $\mathbf{x} \cdot \mathbf{y} = 0$. Real functions are orthogonal when $\int X(\omega)Y(\omega)d\omega = 0$. If the vectors or the functions are complex, there is a small but important change. We take complex conjugates of one vector (say \mathbf{x} , in the physics convention) and of one function $X(\omega)$. *Orthogonality in the complex case means*

$$\bar{\mathbf{x}} \cdot \mathbf{y} = \sum \overline{\mathbf{x}(n)}\mathbf{y}(n) = 0 \quad \text{and} \quad \langle X, Y \rangle = \int_{-\pi}^{\pi} \overline{X(\omega)}Y(\omega) d\omega = 0. \quad (2.35)$$

The integration $\int e^{-in\omega}e^{iN\omega} d\omega = 0$ does not say that $e^{-in\omega}$ is orthogonal to $e^{iN\omega}$. It says that $e^{in\omega}$ is orthogonal to $e^{iN\omega}$ (for $N \neq n$). It is this orthogonality that allows the inverse transform to have the clean and simple formula (2.34).

The discrete analogue of an orthonormal transform is a square matrix with orthonormal columns. This is an “orthogonal” matrix if real, a “unitary” matrix if complex. For such a matrix U , the inverse is again clean and simple. It equals the conjugate transpose \overline{U}^T . This applies to orthonormal filter banks, when the rows of $(\downarrow 2)\mathbf{H}_0$ and $(\downarrow 2)\mathbf{H}_1$ are orthonormal. Their transposes are the columns of $\mathbf{F}_0(\uparrow 2)$ and $\mathbf{F}_1(\uparrow 2)$ —also orthonormal.

Is there a connection between discrete and continuous orthogonality? If two signals are orthogonal (in discrete time), are their transforms orthogonal (in continuous frequency)? The question is important and the answer is *yes*.

This answer follows from the orthogonality of the exponentials $e^{in\omega}$ —on which the whole theory depends. For any \mathbf{x} and \mathbf{y} , the inner products in the time and frequency domains are equal up to a factor of 2π :

$$2\pi \sum_{n=-\infty}^{\infty} \overline{\mathbf{x}(n)}\mathbf{y}(n) = \int_{-\pi}^{\pi} \overline{X(\omega)}Y(\omega) d\omega. \quad (2.36)$$

For proof, substitute $\sum \overline{\mathbf{x}(N)} e^{iN\omega}$ for $\overline{X(\omega)}$ on the right. Also substitute $\sum \mathbf{y}(n) e^{-in\omega}$ for $Y(\omega)$. Integrate from $-\pi$ to π . By orthogonality, the only terms with nonzero integrals are those with $N = n$. The integral is 2π , multiplied by the number $\overline{\mathbf{x}(n)} \mathbf{y}(n)$. The left side of (2.36) is the sum of these nonzero terms.

Special case when $\mathbf{x} = \mathbf{y}$. Now the two vectors are the same. Their transforms are the same. We are integrating $\overline{X(\omega)} X(\omega)$, which is $|X(\omega)|^2$, because $a + ib$ times its conjugate $a - ib$ is $a^2 + b^2$. For the same reason $\overline{\mathbf{x}(n)} \mathbf{x}(n) = |\mathbf{x}(n)|^2$. With $\mathbf{x} = \mathbf{y}$, the energy in the signal (times 2π) equals the energy in the transform:

$$2\pi \sum_{-\infty}^{\infty} |\mathbf{x}(n)|^2 = \int_{-\pi}^{\pi} |X(\omega)|^2 d\omega. \quad (2.37)$$

Example 2.8. Suppose \mathbf{x} is $(1, \beta, \beta^2, \dots)$. Its energy is $1 + \beta^2 + \beta^4 + \dots = 1/(1 - \beta^2)$. Its transform is a one-sided sum, in this case a geometric series:

$$X(\omega) = \sum_0^{\infty} \beta^n e^{-in\omega} = 1 + \beta e^{-i\omega} + (\beta e^{-i\omega})^2 + \dots = \frac{1}{1 - \beta e^{-i\omega}}.$$

Consider the inverse transform from $X(\omega)$ back to $\mathbf{x}(n)$:

$$\mathbf{x}(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{in\omega} d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{e^{in\omega} d\omega}{1 - \beta e^{-i\omega}}.$$

How do we see that this integral gives the correct signal $(1, \beta, \beta^2, \dots)$? The direct way is to write the integrand as $e^{in\omega} (1 + \beta e^{-i\omega} + \beta^2 e^{-2i\omega} + \dots)$. Integration picks out the correct power β^n . (If $n < 0$ then the integral gives zero.) The indirect way is to substitute z for $e^{i\omega}$, and integrate $z^n / (z - \beta)$ around the unit circle:

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{e^{in\omega} d\omega}{1 - \beta e^{-i\omega}} = \frac{1}{2\pi i} \int_{|z|=1} \frac{z^n (dz/z)}{1 - (\beta/z)}.$$

There is a pole at $z = \beta$. The *residue* at this pole is β^n . This is the answer we want. Again the case $n < 0$ is separate (with two poles) and gives zero.

The actual calculation of such an inversion integral is generally difficult or impossible. We seldom need to do it. For a ratio of polynomials it can be done in an emergency by the residue method of complex integration.

Example 2.9. An allpass filter has $|H(\omega)| = 1$ so that $|Y(\omega)| = |X(\omega)|$. The filter conserves energy. The integral of $|Y(\omega)|^2$ equals the integral of $|X(\omega)|^2$, because these functions are the same:

$$\text{Allpass: } \int_{-\pi}^{\pi} |Y(\omega)|^2 d\omega = \int_{-\pi}^{\pi} |H(\omega)X(\omega)|^2 d\omega = \int_{-\pi}^{\pi} |X(\omega)|^2 d\omega.$$

Therefore *the output energy equals the input energy*:

$$\sum_{-\infty}^{\infty} |\mathbf{y}(n)|^2 = \sum_{-\infty}^{\infty} |\mathbf{x}(n)|^2 \quad \text{or} \quad \|\mathbf{y}\|^2 = \|\mathbf{x}\|^2. \quad (2.38)$$

Please do not think that each $|\mathbf{y}(n)| = |\mathbf{x}(n)|$. It is the *frequency* response that has $|H(\omega)| = 1$. The energy in each frequency band is conserved by an allpass filter, not the energy in each time band.

Convergence of the Fourier Series

In defining $X(\omega)$, we have been assuming that this series converges. Otherwise what meaning do we give to $X(\omega)$? We are touching here on a central problem of mathematical analysis (with a literature that goes back for centuries). Touching this problem is as much as we can do—by identifying three types of convergence, and the signals that produce each type.

As with all infinite series, a few terms have nothing to do with convergence. By changing a finite number of inputs $x(n)$ we certainly change the transform—but we do not alter convergence or divergence. It is the behavior of $x(n)$ for large n that is crucial. Here are three types of convergence:

1. Uniform convergence with $\sum |x(n)| < \infty$
2. Strong convergence (in L^2) with $\sum |x(n)|^2 < \infty$
3. Weak convergence allowing polynomial growth in $x(n)$.

1. Uniform convergence Suppose the magnitudes $|x(n)|$ have a finite sum. These are also the magnitudes of $x(n)e^{-in\omega}$, because $|e^{-in\omega}| = 1$. Those terms have different phases, which may produce cancellation when we add them. When $\sum |x(n)|$ converges, we don't need that help. The series of magnitudes converges “absolutely,” and the series $\sum x(n)e^{-in\omega}$ converges “uniformly.” Then the sum $X(\omega)$ is a continuous function—with no jumps.

In the example with $x = (1, \beta, \beta^2, \dots)$ we imposed $|\beta| < 1$. That produced uniform convergence. The transform $X(\omega) = 1/(1 - \beta e^{-i\omega})$ is a continuous function. But not all continuous functions have $\sum |x(n)| < \infty$.

In the brick wall filter, the odd coefficients have magnitude $|h(n)| = \frac{1}{\pi n}$. The sum of magnitudes does not converge. The terms $\frac{1}{\pi n}$ do not go to zero quickly enough. The sum $H(\omega) = \sum h(n)e^{-in\omega}$ does not converge uniformly. And, in fact, $H(\omega)$ is a step function (or square wave) with a jump.

2. Convergence in energy (L^2 convergence) Suppose the squared magnitudes $|x(n)|^2$ have a finite sum. By squaring, small terms become much smaller. Convergence is easier to achieve. If the sum of $|x(n)|$ is finite, the sum of squares is certainly finite. Then the Fourier series converges in L^2 . This “squared” test is passed by the Fourier series for a step function:

$$|h(n)| = \frac{1}{\pi n} \quad \text{has} \quad \sum |h(n)|^2 = \sum \frac{1}{\pi^2 n^2} = \text{convergent series.}$$

Comparing the sums of $1/n$ and $1/n^2$ is like comparing the integrals of $1/x$ and $1/x^2$. One integral is $\log x$, which becomes large as $x \rightarrow \infty$. The area under $1/x^2$ stays finite as $x \rightarrow \infty$.

When the squared magnitudes $|x(n)|^2$ have a finite sum, the squared magnitude $|X(\omega)|^2$ has a finite integral. They are equal apart from 2π . Then $X(\omega)$ is a function in the Hilbert space denoted by L^2 , just as $x(n)$ is a vector in the Hilbert space denoted by ℓ^2 . These spaces contain all functions and vectors with finite energy: *the square of the L^2 norm or the ℓ^2 norm is the energy.*

Functions in Hilbert space may have jumps. Those jumps have no energy (no contribution to the integral). The derivative of a jump is a Dirac delta function. Its coefficients are all $x(n) = \frac{1}{2\pi}$ and its energy is infinite, so this function is outside the space L^2 .

But the delta function is included in the third type of convergence.

3. Weak convergence (to a distribution) Distributions $F(\omega)$ are defined by their inner products with smooth functions $G(\omega)$. For $\int F(\omega)G(\omega) d\omega$, use integration by parts. We integrate $F(\omega)$, which needs it, and we differentiate $G(\omega)$, which can take it. The indefinite integral of the delta function $F(\omega) = \delta(\omega)$ is the unit step $H(\omega)$. The definite integral of $\delta(\omega)G(\omega)$ is the number $G(0)$:

$$\int_{-\pi}^{\pi} \delta(\omega)G(\omega) d\omega = [H(\omega)G(\omega)]_{-\pi}^{\pi} - \int_{-\pi}^{\pi} H(\omega)G'(\omega) d\omega = G(\pi) - \int_0^{\pi} G'(\omega) d\omega = G(0).$$

This defines $\delta(\omega)$. It is a distribution, a derivative of a true function. Still it has Fourier coefficients that are easy to find, with $G(\omega) = e^{in\omega}$:

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \delta(\omega)e^{in\omega} d\omega = \frac{1}{2\pi} \cdot 1.$$

All frequencies are present in the same amount. The Fourier series is

$$\delta(\omega) = \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} e^{in\omega}. \quad (2.39)$$

On the left is a distribution. On the right is a divergent series. The terms don't even approach zero. But in a weak sense those terms cancel each other to produce zero, away from the spike at $\omega = 0$ where they reinforce.

Weak convergence is based on the same idea of testing inner products with smooth $G(\omega)$. The series *converges weakly*, say at $\omega = 0$:

$$G(0) = \int_{-\pi}^{\pi} \delta(\omega)G(\omega) d\omega = \frac{1}{2\pi} \sum_n \left(\int_{-\pi}^{\pi} e^{in\omega} G(\omega) d\omega \right) e^{in0}. \quad (2.40)$$

You see the Fourier coefficients of $G(\omega)$ on the right side, adding to $G(0)$.

Numerically, this weak convergence is not so great. Figure 2.8 shows the sum of 41 terms. In a pointwise sense, and in area, those side lobes will not shrink to zero! In the L_2 sense, the energy is growing with every term. But in a weak sense, this sum is approaching $\delta(\omega)$. As the number of terms increases, the oscillations become faster (not smaller). Multiplied by a smooth $G(\omega)$, the main central lobe picks out $G(0)$ and the integral over the oscillations approaches zero.

Question: What is the weak limit of pure oscillations $e^{in\omega}$ as $n \rightarrow \infty$?

Answer: The limit is the zero function. The inner products $\int e^{in\omega} G(\omega) d\omega$ approach zero. Oscillations converge *weakly* to their average value.

Question: Does the series $\delta'(\omega) = \frac{i}{2\pi} \sum n e^{in\omega}$ for the derivative of a Dirac function also converge weakly? The Fourier coefficients are growing with n .

Answer: Yes. Integrate again by parts. The integral of $\delta'(\omega)G(\omega)$ is $-G'(0)$.

The **Gibbs phenomenon** is just the integral of $\sum e^{in\omega}$. The integral of $\delta(\omega)$ is a step function. The integral of $e^{in\omega}$ introduces $\frac{1}{n}$, so there is L^2 convergence to the step (but not uniform convergence). With integration, *the area under the side lobes become crucial*. The area shows as a height in the Gibbs figure (the integral of the delta function is a step). This undesirable oscillation of Fourier series at a jump (an *edge* in image processing) has brought forward localized bases like wavelets.

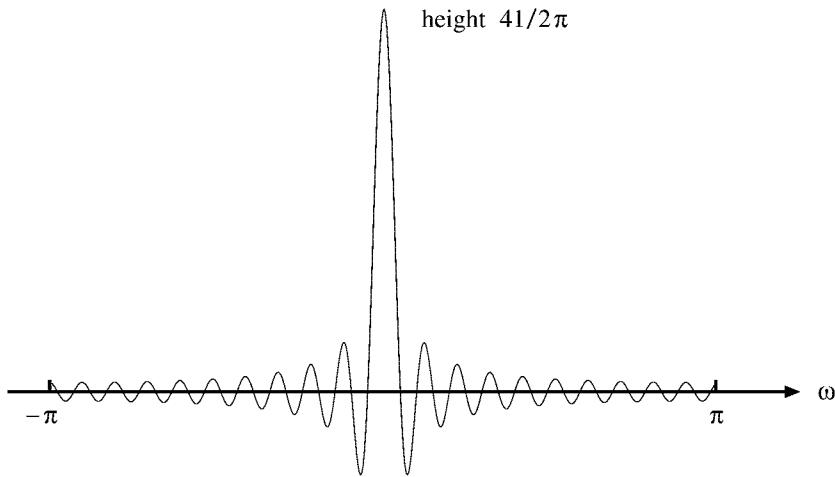


Figure 2.8: The Fourier series $\frac{1}{2\pi} \sum e^{-in\omega}$ converges weakly to $\delta(\omega)$. Its integral shows the Gibbs phenomenon (nonuniform convergence).

It is significant that the same Gibbs difficulty in shock calculations has been handled very differently. The finite difference schemes are made non-oscillatory by *nonlinear terms*. The nonlinearity is active where it is needed. It is inactive where the solution is smooth. Morel's book [Mo] gives a strong impetus to this nonlinear idea for image processing.

For iteration of the filter $\mathbf{h} = (\frac{1}{2}, 0, 0, \frac{1}{2})$, we will see weak convergence in Chapter 7. The cascade algorithm (= lowpass iteration) produces functions that oscillate between 1 and 0 on the interval $[0, 3]$. They have no limit in L^2 . The weak limit of the oscillations is a stretched box $\phi(t)$ with constant value $\frac{1}{3}$.

Good filters give L^2 convergence (and usually uniform convergence) when iterated. The necessary and sufficient Condition E is in Section 7.2. But a “good filter” in the classical sense, with small errors in the passband and stopband, may fail in iteration!

The requirement for success in iteration is flatness of the response $H(\omega)$. *The number of zeros at $\omega = \pi$ is absolutely critical.* This is the new property that has become important. It makes the iteration process strong and not weak, regular and not oscillatory. It must be built in or iterations will oscillate — as happened with highly regarded filters.

Poisson's Summation Formula

The Fourier coefficients of the Dirac delta function on $[-\pi, \pi]$ are all $\frac{1}{2\pi}$. By periodicity, the Dirac delta becomes a *Dirac comb*. There is an impulse at every multiple of 2π . The Fourier series for this periodic train of delta functions is

$$\sum_{k=-\infty}^{\infty} \delta(\omega - 2\pi k) = \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} e^{-in\omega}. \quad (2.41)$$

The left side is a sum of impulses. The right side is its Fourier series, *converging weakly* as above. The formula is often seen with t instead of ω , because Fourier analysis usually starts with continuous time. Then the frequencies are discrete. In signal processing it is the other way around.

Equation (2.41) is a short statement of the Poisson formula, but it involves weak convergence. The terms don't approach zero. To see ordinary convergence we take inner products with any smooth function $G(\omega)$. With integrals over the whole line, we ask $G(\omega)$ to decay as $|\omega| \rightarrow \infty$. The inner product with delta functions gives point values $G(2\pi k)$. The inner product on the right gives point values $\widehat{G}(n)$ of the Fourier transform. The formula says that the two sums are equal:

$$\textbf{\textit{Poisson's Summation Formula}} \quad \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} G(2\pi k) = \sum_{n=-\infty}^{\infty} \widehat{G}(n). \quad (2.42)$$

This is a remarkable formula. It relates the samples of G to the samples of its transform \widehat{G} . We can change the spacing of one set of samples to any $T > 0$, provided the other set of samples is spaced at $\frac{2\pi}{T}$.

Smoothness of $X(\omega)$ and Decay of $x(n)$

The regularity of a function is partly revealed in its Fourier coefficients. When $X(\omega)$ is continuous, its Fourier coefficients $x(n)$ approach zero. But the coefficients can approach zero when $X(\omega)$ is *not* continuous; the step function is an example. There is a gap between $x(n) \rightarrow 0$ and $\sum |x(n)| < \infty$, which is at the heart of Fourier analysis:

$$\sum |x(n)| < \infty \quad \Rightarrow \quad \text{continuous } X(\omega) \quad \Rightarrow \quad x(n) \rightarrow 0.$$

One virtue of wavelets is that such gaps can be closed. Instead of necessary conditions for smoothness, and then sufficient conditions for smoothness, we can find *necessary and sufficient conditions*. These are conditions on the wavelet coefficients $x(n)$, for the function $X(\omega)$ to have specified regularity. $X(\omega)$ lies in a specified function space when $x(n)$ lies in a corresponding vector space [DeLu]. The function norm and vector norm are equivalent—as for $X(\omega)$ in L^2 and $x(n)$ in ℓ^2 by Parseval's formula.

Fourier coefficients give partial information from the magnitudes $|x(n)|$. Each extra order of smoothness in $X(\omega)$ is reflected in one extra order of decay in $|x(n)|$.

Theorem 2.2 *Suppose $X(\omega)$ has s continuous derivatives. Then $n^s |x(n)| \rightarrow 0$ as $|n| \rightarrow \infty$.*

When $s = 0$ this is the Riemann-Lebesgue Lemma. The Fourier coefficients approach zero (we really only need $\int |X(\omega)| d\omega$ to be finite). For integers $s = 1, 2, 3, \dots$ we look at the Fourier coefficients of the derivative $X^{(s)}(\omega)$. Those coefficients are $(in)^s x(n)$. They approach zero (again by Riemann-Lebesgue) when $X^{(s)}(\omega)$ is continuous. This is the theorem.

In compression of a signal, this decay of coefficients is crucial. Small coefficients are removed (partly or completely). For a Fourier basis, the smoothness of an image determines the decay of coefficients. For a wavelet basis, it is the *piecewise smoothness* that matters. Wavelets are well adapted to piecewise smooth functions (with edges). Wavelets are local where Fourier waves are global.

The theorem was stated for integers $s = 0, 1, 2, \dots$ but fractions can be allowed. This is important for a satisfactory theory, because functions like $X(\omega) = |\omega|^{1/2}$ are more than continuous ($s = 0$) and less than differentiable ($s = 1$). The in-between smoothness is measured by the *Hölder exponent*. The function $X(\omega)$ belongs to the space C^s , $0 < s < 1$, if it satisfies the Hölder condition

$$|X(\omega_2) - X(\omega_1)| \leq \text{constant } |\omega_2 - \omega_1|^s.$$

The square root function $|\omega|^{1/2}$ belongs to $C^{1/2}$. Similarly $|\omega|^s$ belongs to C^s . For $s > 1$ we would take $[s]$ derivatives first. The Hölder exponent of that derivative is the fractional part $s - [s]$. Then the Fourier coefficients decay to order s at least: $n^s |\mathbf{x}(n)| \rightarrow 0$.

The Fourier basis automatically takes advantage of high regularity. *The wavelet basis takes full advantage of a high s , only if it is designed to do so.* The lowpass filter must have more than s zeros at π . Haar wavelet expansion does not converge faster as the function gets smoother. Therefore Haar compression is poor.

For biorthogonal wavelets, the analyzing filter \mathbf{H}_0 governs the decay of coefficients. The synthesizing filter \mathbf{F}_0 governs the smoothness of the output. *We would like both to have many zeros at π !* When forced to choose, whether to put zeros into \mathbf{H}_0 or \mathbf{F}_0 , the preference often goes to \mathbf{F}_0 — then the synthesis basis functions will be smooth.

We emphasize one more point. *In the L^2 norm*, where “mean square” replaces “maximum,” *the Fourier coefficients exactly reflect the smoothness*. The coefficients are in ℓ^2 precisely when the function is in L^2 . The energy is the same for both, by Parseval’s identity. This equality extends to the s th derivative $X^{(s)}(\omega)$, whether s is an integer or not:

$$\int_{-\pi}^{\pi} |X^{(s)}(\omega)|^2 d\omega = 2\pi \sum_{-\infty}^{\infty} |n^s \mathbf{x}(n)|^2. \quad (2.43)$$

When s is not an integer, we *define* this derivative $X^{(s)}(\omega)$ by its coefficients $(in)^s \mathbf{x}(n)$. The equation above is just Parseval itself, for the derivative. Wavelet theory is greatly simplified by working in the Hilbert spaces of functions with s derivatives in L^2 , rather than the Hölder spaces C^s of functions with s continuous derivatives.

The number p of zeros at π is crucial in both cases. For L^2 spaces we will find in Chapter 7 the exact smoothness of $\phi(t)$ and $w(t)$.

Heisenberg’s Uncertainty Principle

The underlying property of wavelets is that they are pretty well localized in both time and frequency. The functions $e^{i\omega t}$ are perfectly localized at ω but they extend over all time. Wavelets are *not* at a single frequency, or even a finite range, but they are limited to finite time. As we rescale, the frequency goes up by 2^j and the time interval goes down by 2^j . This suggests that the *product* of frequency interval and time interval is a stable quantity. The Heisenberg Uncertainty Principle makes those definitions precise, and gives a *lower bound* for the product.

We emphasize that the wavelet can only be “pretty well localized.” It cannot have finite support in both t and ω . (A famous theorem.) Instead of the support length we use the variances

$$\sigma^2 = \int_{-\infty}^{\infty} t^2 |f(t)|^2 dt \quad \text{and} \quad \widehat{\sigma}^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} \omega^2 |\widehat{f}(\omega)|^2 d\omega. \quad (2.44)$$

We are no longer working with periodic functions, so t and ω extend from $-\infty$ to ∞ .

Theorem 2.3 (Heisenberg) *If $\|f\| = 1$ then the product $\sigma \widehat{\sigma}$ is at least $\frac{1}{2}$.*

The lower bound $\frac{1}{2}$ is attained by the Gaussian function $f(t) = e^{-t^2}$. Its transform $\widehat{f}(\omega)$ is also a Gaussian. These have infinite support! But they are as local as possible, measured by σ and $\widehat{\sigma}$. We can rescale time by c and frequency by $\frac{1}{c}$, we can shift to $t - t_0$, and we can modulate by $e^{i\omega_0 t}$. The variances σ^2 and $\widehat{\sigma}^2$ would be computed around t_0 and ω_0 , and the bound $\frac{1}{2}$ is still reached.

This led Gabor to use Gaussians in constructing “time-frequency atoms.” But numerically, finite support in time is better.

Proof of the Heisenberg Principle. The key is that $\sigma = \|tf(t)\|$ and $\widehat{\sigma} = \|f'(t)\|$. The principle applies to pairs of operators, like position \mathbf{P} and momentum \mathbf{Q} , that have the property $\mathbf{QP} - \mathbf{PQ} = \mathbf{I}$. When this property holds, the proof comes directly from the Schwarz inequality:

$$1 = \|f\|^2 = \langle f, (\mathbf{QP} - \mathbf{PQ})f \rangle \leq 2 \|\mathbf{Q}f\| \|\mathbf{P}f\| = 2\widehat{\sigma}\sigma. \quad (2.45)$$

In our case \mathbf{P} is multiplication by t and \mathbf{Q} is differentiation:

$$(\mathbf{QP} - \mathbf{PQ})f(t) = \frac{d}{dt}(tf(t)) - t\frac{df}{dt} = \mathbf{If}(t) \quad \text{as required.}$$

Problem 4 develops this proof directly from the definitions of σ and $\widehat{\sigma}$.

Problem Set 2.4

1. Add the terms $\frac{1}{2\pi} \sum_{-N}^N e^{in\omega}$. This is a partial sum of the Fourier series for $\delta(\omega)$. At what ω does it come down to zero, at the end of the main lobe in Figure 2.8? Where is the end of the first side lobe?
2. Describe a continuous function $X(\omega)$ whose coefficients have infinite sum $\sum |x(n)| = \infty$.
3. What is Poisson’s summation formula for the Gaussian $G(\omega) = e^{-\omega^2}$?
4. Heisenberg’s Uncertainty Principle is $\sigma\widehat{\sigma} \geq \frac{1}{2}$. The Schwarz inequality gives

$$\left| \int tf(t) f'(t) dt \right|^2 \leq \int |tf(t)|^2 dt \int |f'(t)|^2 dt.$$

Identify the right side as $\sigma^2\widehat{\sigma}^2$. Integrate by parts on the left side to get

$$\int_{-\infty}^{\infty} t [f(t)f'(t)] dt = t \frac{f(t)^2}{2} \Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} \frac{f(t)^2}{2} dt.$$

If the integrated part is zero at $\pm\infty$, and if $\|f\| = 1$, deduce that $\sigma\widehat{\sigma} \geq \frac{1}{2}$. Equality holds when $tf(t)$ is proportional to $f'(t)$, which leads to Gaussians $f(t) = e^{-ct^2}$.

5. What power of $\frac{1}{n}$ gives the decay rate for the coefficients $x(n)$ of
 1. $X(\omega) = \omega$ = discontinuous function at $\omega = \pm\pi$.
 2. $X(\omega) = |\omega|$ = continuous function with period 2π .
 3. $X(\omega) = \omega^2$ = continuous function with period 2π .
 4. $X(\omega) = \text{spline with jump in the third derivative } X'''(\omega)$.
6. Compute the Fourier coefficients and the energy for
 - (a) $X(\omega) = \frac{1}{1 - \frac{1}{2}e^{-i\omega}}$
 - (b) $X'(\omega)$
 - (c) Periodic box function $H(\omega) = \begin{cases} 1 & 0 \leq \omega < \pi \\ 0 & -\pi \leq \omega < 0 \end{cases}$
7. Show that $X(\omega) = (1 - \beta e^{-i\omega})^{-1}$ has the same energy as $x = (1, \beta, \beta^2, \dots)$. Use a substitution in the integral of $|X(\omega)|^2$.
8. Show that the Heisenberg product $\sigma\widehat{\sigma}$ is not changed by *dilation, modulation, and translation*: $f(t)$ is transformed to $2^{j/2}f(2^j t)$ and $e^{i\omega t}f(t)$ and $f(t-s)$.
9. Determine the Fourier transform of the signal $x(n) = \alpha^{|n|}$, $|\alpha| < 1$.

2.5 Bases and Frames

Above all, this book is about **the choice of a good basis**. In reality, that choice governs everything. Every transform is a change to a different basis. The contribution of filter banks and wavelets (and Fourier transforms and local cosines and wavelet packets) is *to offer new bases*. The whole subject has been reopened, by the demands of its applications.

What is a basis and what makes it good? A basis is a sequence of vectors v_1, v_2, \dots or functions $v_1(t), v_2(t), \dots$ with the property of *unique representation*:

Every vector v or function $v(t)$ in the space can be represented
in one and only one way as $v = \sum b_i v_i$ or $v(t) = \sum b_i v_i(t)$.

There is exactly one representation of every vector and function. The zero vector and zero function can only be represented with all $b_i = 0$. The basis functions are *linearly independent*.

It is usual to require convergence in norm, $\|v - \sum_1^N b_i v_i\| \rightarrow 0$ as $N \rightarrow \infty$. For L^2 spaces this norm is the square root of the energy. Section 1.5 mentions four reasons for that choice.

There are two separate properties to be established for any proposed basis: (1) *linear independence* and (2) *completeness*. Adding extra vectors will destroy independence. Removing vectors from the basis will kill completeness. *Linear independence is automatic for orthonormal vectors*. When all angles are 90° and all lengths are unity, there is no chance of degeneracy. In infinite dimensions we meet the possibility that angles can approach zero without reaching zero. In that case the basis is unstable. The coefficients b_i are out of control. In the good case, the coefficients satisfy

$$A \|v\|^2 \leq \sum |b_i|^2 \leq B \|v\|^2 \quad \text{with } A > 0. \quad (2.46)$$

This is the defining property of a *Riesz basis*, also known as a *stable basis* or an *unconditional basis*. A key assumption in wavelet theory is that the translates $\phi(t - n)$ of the scaling function are a Riesz basis (for the space V_0 inside L^2). We will find the test to be applied to $\widehat{\phi}(\omega)$, and the equivalent Condition E on the filter coefficients, to produce a Riesz basis of scaling functions.

Dual Bases and Dual Frames

In a few lines, we can give the main points about bases and frames. These ideas will be developed below — bases will come first. Here are the key points in a hurry:

Dual bases come from columns v_n of \mathbf{T}^{-1} and rows r_n of \mathbf{T} : $\mathbf{T}^{-1}\mathbf{T} = \mathbf{I}$

Dual frames come from columns v_n^+ of \mathbf{T}^+ and rows r_n of \mathbf{T} : $\mathbf{T}^+\mathbf{T} = \mathbf{I}$.

The difference is that *the frame vectors v_n^+ need not be independent*. They can be redundant (Figure 2.9). We still require the coefficients $b_i = \langle r_i, v \rangle$ to satisfy (2.46), but other combinations of the v_n^+ can reconstruct the same v .

\mathbf{T}^+ is only a left-inverse for a frame. It becomes a two-sided inverse for a basis.

In N -dimensional space, frames contain $M > N$ vectors. The matrix \mathbf{T} is $M \times N$, and its left-inverse \mathbf{T}^+ is $N \times M$. **The equation $\mathbf{T}\mathbf{T}^+ = \mathbf{I}$ is not true.** For finite dimensions the theory and examples are particularly clear — our eventual applications are to infinite dimensions.

Note 1 Change of Basis Suppose \mathbf{T} is a bounded linear operator with a bounded inverse. If the sequence $\{v_n\}$ is a Riesz basis, so is the sequence $\{\mathbf{T}^{-1}v_n\}$. When we expand $\mathbf{T}v = \sum c_n v_n$ in the



Figure 2.9: Good basis, bad basis, good frame, bad frame, all in \mathbf{R}^2 .

original basis and multiply by \mathbf{T}^{-1} , this gives the expansion $\mathbf{v} = \sum c_n(\mathbf{T}^{-1}\mathbf{v}_n)$ in the new basis: $A = B = 1$. The new Riesz constants A, B come from the original A, B and the norms of \mathbf{T} and \mathbf{T}^{-1} .

The classical transforms of mathematics have a stronger property than invertibility. *\mathbf{T} is often a unitary operator.* The inverse of \mathbf{T} becomes the conjugate transpose \mathbf{T}^* . The norms are $\|\mathbf{T}\| = \|\mathbf{T}^{-1}\| = 1$. If the original basis is orthonormal so is the new basis: $A = B = 1$. The Fourier transform and wavelet transform are unitary operators — when we use orthonormal wavelets! Biorthogonal wavelets lead to non-unitary operators \mathbf{T} and to Riesz bases — but not orthonormal. Fortunately $\|\mathbf{T}\|$ and $\|\mathbf{T}^{-1}\|$ are in practice surprisingly close to 1.

In finite dimensions, every basis is a Riesz basis. *The basis vectors for \mathbf{R}^N are the columns of an invertible $N \times N$ matrix.* That matrix is \mathbf{T}^{-1} . It is essential to distinguish the change of coordinates (which uses \mathbf{T}) from the new basis (the columns of \mathbf{T}^{-1}). The fundamental fact is $\mathbf{T}^{-1}\mathbf{T} = \mathbf{I}$. We use it now, multiplying *columns times rows*:

$$\mathbf{v} = \mathbf{T}^{-1}\mathbf{T}\mathbf{v} = \sum_{n=1}^N (\text{column } n \text{ of } \mathbf{T}^{-1})(\text{row } n \text{ of } \mathbf{T})\mathbf{v}$$

The n th basis vector is $\mathbf{v}_n = \text{column } n \text{ of } \mathbf{T}^{-1}$

The n th coordinate is $b_n = (\text{row } n \text{ of } \mathbf{T})\mathbf{v} = \langle \mathbf{r}_n, \mathbf{v} \rangle$.

Those three lines give $\mathbf{v} = \sum b_n \mathbf{v}_n$.

Every $N \times N$ matrix is a bounded operator. When the inverse exists, it is also bounded. But infinite matrices can represent unbounded operators. The averaging filter with coefficients $\frac{1}{2}, \frac{1}{2}$ is bounded. The inverse filter with coefficients $2, -2, 2, -2, 2, -2, \dots$ is unbounded. The Riesz requirement, that \mathbf{T} and \mathbf{T}^{-1} are both bounded, is needed for a stable change of basis. An orthonormal basis ($A = B = 1$) has condition number 1. Then the product $\|\mathbf{T}\| \|\mathbf{T}^{-1}\|$ is the **condition number** of the new basis.

Bases from Filter Banks

The important examples for us are the bases from filter banks (discrete time) and the bases from wavelets (continuous time). Chapter 1 gave an example of both. The discrete time basis vectors had only two nonzero components:

$$\dots, \begin{bmatrix} \cdot \\ 1 \\ 1 \\ 0 \\ 0 \\ \cdot \end{bmatrix}, \begin{bmatrix} \cdot \\ 1 \\ -1 \\ 0 \\ 0 \\ \cdot \end{bmatrix}, \begin{bmatrix} \cdot \\ 0 \\ 0 \\ 1 \\ 1 \\ \cdot \end{bmatrix}, \begin{bmatrix} \cdot \\ 0 \\ 0 \\ 1 \\ -1 \\ \cdot \end{bmatrix}, \dots$$

This is an orthogonal basis because the vectors are mutually perpendicular. It is an orthonormal basis when divided by $\sqrt{2}$.

The heart of the book is the construction of other (and better) filter banks. The filters will be longer and they will overlap. Orthogonality will not be automatic and it may not be true. The bases are *the impulse responses of the filters* — with a double shift as above. Here is an example with four nonzero components:

$$\dots, \begin{bmatrix} \cdot \\ 1 \\ 3 \\ 3 \\ 1 \\ 0 \\ 0 \\ \cdot \end{bmatrix}, \begin{bmatrix} \cdot \\ 1 \\ 3 \\ -3 \\ -1 \\ 0 \\ 0 \\ \cdot \end{bmatrix}, \begin{bmatrix} \cdot \\ 0 \\ 0 \\ 1 \\ 3 \\ 3 \\ 1 \\ \cdot \end{bmatrix}, \begin{bmatrix} \cdot \\ 0 \\ 0 \\ 1 \\ 3 \\ -3 \\ -1 \\ \cdot \end{bmatrix}, \dots$$

Those basis vectors are *not* orthogonal. We are alternating lowpass filters with highpass filters, but the first and third (lowpass and shifted lowpass) are not orthogonal. Remember that these vectors are the columns of \mathbf{T}^{-1} . In our other language *they are the impulse responses from the synthesis filters*. The synthesis filters combine to reconstruct the signal, which is exactly what the basis vectors do.

In this nonorthogonal case, the inverse is not the transpose. The basis is not self-orthogonal. It is *biorthogonal* to a different basis — which we now discuss.

Biorthogonal Bases (Dual Bases)

The basis $\{\mathbf{r}_n\}$ is biorthogonal to the basis $\{\mathbf{v}_n\}$ if the inner products are

$$\langle \mathbf{r}_i, \mathbf{v}_j \rangle = \delta(i - j). \quad (2.47)$$

This is the same property that governs the rows of a matrix \mathbf{T} and the columns of \mathbf{T}^{-1} . It comes directly from $\mathbf{T}\mathbf{T}^{-1} = \mathbf{I}$. So when the columns of \mathbf{T}^{-1} are a basis (as above), the rows of \mathbf{T} are the biorthogonal basis — the *dual basis*.

We transpose \mathbf{T} to turn its rows into columns. We transpose $\mathbf{T}^{-1}\mathbf{T} = \mathbf{I}$ into $\mathbf{T}^*\mathbf{T}^{-*} = \mathbf{I}$. Then the same idea that gave the basis $\{\mathbf{v}_n\}$ from \mathbf{T}^{-1} now gives the biorthogonal basis $\{\mathbf{r}_n\}$ from the columns of the transpose matrix \mathbf{T}^* :

$$\mathbf{v} = \mathbf{T}^*\mathbf{T}^{-*}\mathbf{v} = \sum_{n=1}^N (\text{column } n \text{ of } \mathbf{T}^*)(\text{row } n \text{ of } \mathbf{T}^{-*})\mathbf{v}$$

The n th dual basis vector is $\mathbf{r}_n = \text{column } n \text{ of } \mathbf{T}^*$

The n th dual coordinate is $d_n = (\text{row } n \text{ of } \mathbf{T}^{-*})\mathbf{v} = \langle \mathbf{v}_n, \mathbf{v} \rangle$.

When $\mathbf{T}^* = \mathbf{T}^{-1}$, the basis is self-dual and $\mathbf{v}_n = \mathbf{r}_n$. We have an orthonormal basis. In general the rows of \mathbf{T} and columns of \mathbf{T}^{-1} produce biorthogonal bases — provided always that \mathbf{T} and \mathbf{T}^{-1} are bounded. In this case we have one expansion of \mathbf{v} coming from $\mathbf{T}^{-1}\mathbf{T} = \mathbf{I}$, and another expansion from $(\mathbf{T}^{-1}\mathbf{T})^* = \mathbf{I}$:

Theorem 2.4 *If \mathbf{r}_n and \mathbf{v}_n are biorthogonal bases then any \mathbf{v} has two expansions:*

$$\mathbf{v} = \sum c_n \mathbf{v}_n = \sum \langle \mathbf{r}_n, \mathbf{v} \rangle \mathbf{v}_n \quad \text{and} \quad \mathbf{v} = \sum d_n \mathbf{r}_n = \sum \langle \mathbf{v}_n, \mathbf{v} \rangle \mathbf{r}_n. \quad (2.48)$$

In other words $\sum \mathbf{v}_n \mathbf{r}_n^T = \mathbf{I}$ and also $\sum \mathbf{r}_n \mathbf{v}_n^T = \mathbf{I}$.

Example 2.10. The four-tap filters in the example above are biorthogonal to these four-tap filters (after we divide by 16). Check the inner products:

$$\dots, \begin{bmatrix} \cdot \\ -1 \\ 3 \\ 3 \\ \dots \\ -1 \\ 0 \\ 0 \\ \cdot \end{bmatrix}, \begin{bmatrix} \cdot \\ -1 \\ 3 \\ -3 \\ 1 \\ 0 \\ 0 \\ \cdot \end{bmatrix}, \begin{bmatrix} \cdot \\ 0 \\ 0 \\ -1 \\ 3 \\ 3 \\ -1 \\ \cdot \end{bmatrix}, \begin{bmatrix} \cdot \\ 0 \\ 0 \\ -1 \\ 3 \\ -3 \\ 1 \\ \cdot \end{bmatrix}, \dots$$

The construction seems magical. We want more like this. Chapters 4 and 5 show how to get more—and the filters need not have equal length. These are the synthesis filters (in \mathbf{T}^{-1}) and the analysis filters (in the rows of \mathbf{T}) of a *perfect reconstruction filter bank*. The filter bank is biorthogonal.

The norms of \mathbf{T} and \mathbf{T}^{-1} , and therefore the condition number $\|\mathbf{T}\| \|\mathbf{T}^{-1}\|$, are easily computed for these bases. The double shift in \mathbf{T} means that the transform involves a 2×2 matrix function of ω . Maximizing its norm over ω yields $\|\mathbf{T}\|$, and maximizing the norm of the 2×2 inverse yields $\|\mathbf{T}^{-1}\|$.

A small note of caution. The scaling functions and wavelets will come from iterating the lowpass filter, with rescaling. Not every biorthogonal filter bank leads to biorthogonal bases. Sometimes the iteration diverges. There is a serious step from $L^2(\mathbf{Z})$ in discrete time to $L^2(\mathbf{R})$ in continuous time. The example above fails! Those particular filters with ± 1 and ± 3 are only good when they are not iterated too often.

Wavelet Packets and the Best Basis

In Chapter 1 only the lowpass filter was iterated. It was assumed that lower frequencies contained more important information than higher frequencies. For many signals this is not true. A *wavelet packet basis allows any dyadic tree structure* (Figure 2.10). At each point in the tree we have

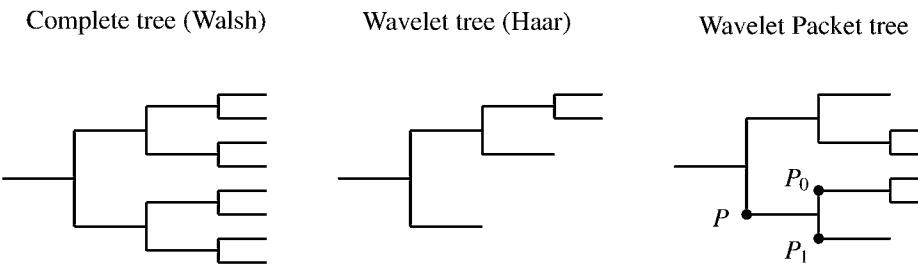


Figure 2.10: Each wavelet packet tree yields a basis, including Walsh and Haar.

the option to send the signal through the lowpass-highpass filter bank, *or not*.

One possibility is the *logarithmic tree*, with lowpass iteration only. Another possibility is the *complete tree*, analogous to the Short Time Fourier Transform. Wavelet packets make up the entire family of bases. Each one is associated with a particular *quadtree*, because it comes from splitting into two (or not splitting) at each step. The decision to split or to merge should

be aimed at achieving minimum distortion D —subject to cost and capacity constraints on the rate R .

The main point is that a library of wavelet packet bases is a practical possibility [W]. For a given signal and a given point P in the tree, we have the coefficients of the basis functions $w_P(t)$ for that point. If we choose to split, that set of functions is replaced by two half-size sets of functions—created by lowpass and highpass filtering:

$$\sum_{k=0}^N \mathbf{h}_0(k) w_P(2t - k) \quad \text{and} \quad \sum_{k=0}^N \mathbf{h}_1(k) w_P(2t - k).$$

Together with their time shifts, these are the new basis functions for the points P_0 and P_1 in the tree. At each of those new points we again have the option to split.

A point is at level j in the tree if it is reached after j splittings. The basis functions at the root (level $j = 0$) are the shifts $\phi(t - k)$ of the scaling function.

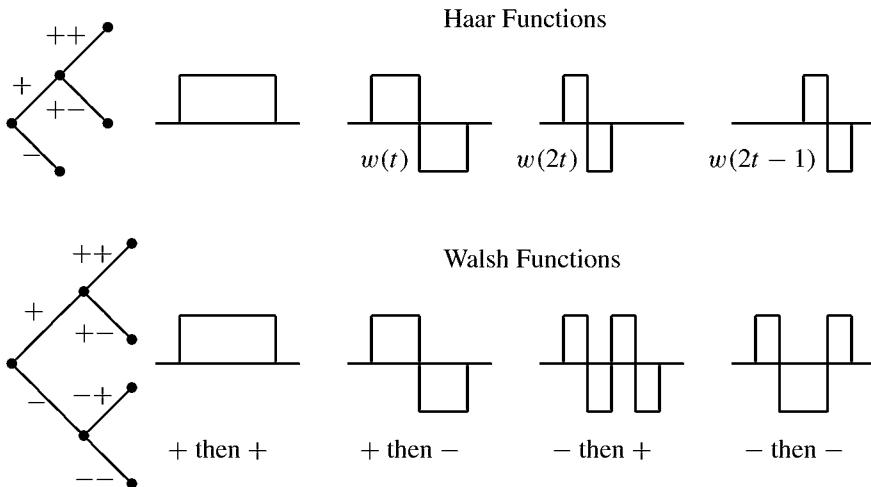


Figure 2.11: Haar iterates only the lowpass filter. Walsh iterates both filters.

Example 2.11. When $\phi(t)$ is the box function, the wavelet $w(t)$ is Haar's up-down square wave. The filter coefficients are $\mathbf{h}_0(k) = 1, 1$ for lowpass and $\mathbf{h}_1(k) = 1, -1$ for highpass, all divided by $\sqrt{2}$. We describe three special bases for the functions that are piecewise constant on intervals of length 2^{-j} . Within $[0, 1]$ this is a space of dimension 2^J :

1. **Box basis:** $\phi(2^J t - k)$ for $0 \leq k < 2^J$.
2. **Haar wavelet basis:** $\phi(t - k)$ and $w(2^J t - k)$ for $j < J$ and $0 \leq k < 2^j$.
3. **Walsh basis:** two functions $w_{J-1}(2t) \pm w_{J-1}(2t - 1)$ from the basis for $J - 1$.

A *Walsh basis function takes values 1 or -1 over the whole unit interval*. It comes from the complete tree with all branches. A wavelet basis function from the logarithmic tree is zero over most of $[0, 1]$. It is nonzero on an interval of length 2^{-j} , varying with j . The box basis functions are nonzero over intervals of fixed length 2^{-j} .

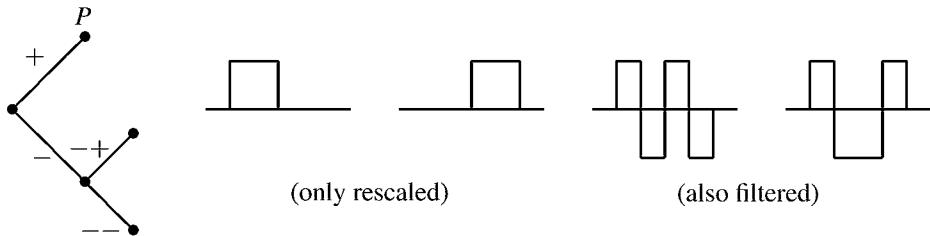
Figure 2.11 shows the four Haar and four Walsh functions (with $+$ or $-$ choices indicated). The *Walsh basis* chooses to split every time. The basis functions are like discrete cosines. The

wavelet basis splits only along one branch. This gives dilation and translation. The *box basis* (scaling function basis) never splits. This gives translation only—a row of small boxes.

The frequency of the ordinary cosine is replaced by the number of *zero-crossings* (sign changes) of the piecewise constant Walsh function. A typical wavelet packet falls between Haar and Walsh. It makes the Walsh decisions, + or −, up to certain points P on the tree. At those points it stops splitting and just rescales. The wavelet packet below could be the best basis for a signal with no low frequencies.

The wavelet packet bases are given by simple recursions (not by simple formulas). They inherit the properties of the filter bank—orthogonality or biorthogonality. The Riesz constants A, B and the condition number B/A are not necessarily in control as J increases. The condition number stays bounded for the logarithmic tree wavelet basis [CoDa]. The condition number is unbounded for the general wavelet packet tree.

We emphasize the recursive form of every wavelet packet decomposition. The coefficient sequence at a point P is treated in exactly the same way as the original sequence $x(n)$ at the original root of the tree. The packet splits or not. If it splits then the two new branches end in two new decision points.



There is a corresponding Fourier packet. Its decision is whether to break the interval in two. Instead of a fixed length 8 for all DFT blocks, and 8×8 for an image, the splitting decision is made locally—depending on the signal. The “butterfly” in the FFT is like the lowpass-highpass bank for wavelets. But the FFT admits all frequencies $0, 1, 2, 3, \dots$ where the wavelets are dyadic and octave-based.

Now we turn from bases to frames, and give up linear independence.

Frames and Frame Bounds

A frame $\{\mathbf{v}_n^+\}$ has one property of a Riesz basis $\{\mathbf{v}_n\}$ —every vector or function can be represented as $\sum c_n \mathbf{v}_n^+$ with control of $\sum |c_n|^2$. But the requirement of linear independence is dropped. A frame is associated with “oversampling” or “redundancy.” There are too many vectors for a basis. We could even repeat the same basis vectors several times—this produces a frame but not an interesting one. More interesting is a set of functions like $\{e^{icnt}\}$, which are a basis for $L^2[0, \pi]$ when $c = 1$ and a tight frame for $c < 1$ (higher than Nyquist rate). For $c = \frac{1}{2}$ this frame is a union of two bases $\{e^{int}\}$ and $\{e^{int} e^{it/2}\}$.

The place to start is in finite dimensions. The M rows of a rectangular matrix give a frame for \mathbf{R}^N —provided the columns are independent:

T is any $M \times N$ matrix with N independent columns.

In general $M > N$ and T^{-1} does not exist.

The left-inverse $T^+ = (T^* T)^{-1} T^*$ does exist and $T^+ T = I$.

Linear algebra [S] says that $\mathbf{T}^*\mathbf{T}$ always has the same rank as \mathbf{T} . By assumption of independent columns, this rank is N . The $N \times N$ matrix $\mathbf{T}^*\mathbf{T}$ with this rank is invertible. The identity $(\mathbf{T}^*\mathbf{T})^{-1}\mathbf{T}^*\mathbf{T} = \mathbf{I}$ shows that $\mathbf{T}^+\mathbf{T} = \mathbf{I}$.

This matrix $\mathbf{T}^+ = (\mathbf{T}^*\mathbf{T})^{-1}\mathbf{T}^*$ is a left-inverse and a “pseudo-inverse” of \mathbf{T} . The columns of \mathbf{T}^+ and the rows of \mathbf{T} are two frames, *dual to each other*. The key is $\mathbf{T}^+\mathbf{T} = \mathbf{I}$.

Example 2.12. The three rows $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ of \mathbf{T} constitute a frame for \mathbf{R}^2 :

$$\mathbf{T} = \begin{bmatrix} 2 & 0 \\ -1 & \sqrt{3} \\ -1 & -\sqrt{3} \end{bmatrix} \quad \text{has independent columns and} \quad \mathbf{T}^*\mathbf{T} = \begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix}.$$

This example is actually a *tight frame*, because $\mathbf{T}^*\mathbf{T}$ is a multiple of \mathbf{I} . The left-inverse is $(\mathbf{T}^*\mathbf{T})^{-1}\mathbf{T}^* = \frac{1}{6}\mathbf{T}^*$. This matrix \mathbf{T}^+ is 2×3 . It cannot possibly be a right-inverse of \mathbf{T} .

The frame vectors are not independent but they span the space. *We can recover \mathbf{v} from the columns $\mathbf{v}_1^+, \mathbf{v}_2^+, \mathbf{v}_3^+$ by using inner products $\langle \mathbf{r}_1, \mathbf{v} \rangle$ and $\langle \mathbf{r}_2, \mathbf{v} \rangle$ and $\langle \mathbf{r}_3, \mathbf{v} \rangle$.* The recovery is based on $\mathbf{T}^+\mathbf{T} = \mathbf{I}$:

$$\mathbf{v} = \mathbf{T}^+\mathbf{T}\mathbf{v} = \sum_{n=1}^M (\text{column } n \text{ of } \mathbf{T}^+) (\text{row } n \text{ of } \mathbf{T})\mathbf{v}$$

The n th analysis frame vector is $\mathbf{r}_n = \text{row } n \text{ of } \mathbf{T}$

The n th coordinate is $\langle \mathbf{r}_n, \mathbf{v} \rangle = (\text{row } n \text{ of } \mathbf{T})\mathbf{v}$

The n th synthesis frame vector is $\mathbf{v}_n^+ = \text{column } n \text{ of } \mathbf{T}^+$.

In this example the column vector $\mathbf{v} = [1 \ 1]'$ has coordinates $\langle \mathbf{r}_1, \mathbf{v} \rangle = 2$ and $-1 + \sqrt{3}$ and $-1 - \sqrt{3}$. To synthesize \mathbf{v} from the three columns of \mathbf{T}^+ , multiply the columns by those coordinates:

$$\mathbf{T}^+(\mathbf{T}\mathbf{v}) = \frac{1}{6} \begin{bmatrix} 2 & -1 & -1 \\ 0 & \sqrt{3} & -\sqrt{3} \end{bmatrix} \begin{bmatrix} 2 \\ -1 + \sqrt{3} \\ -1 - \sqrt{3} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The point of the pseudo-inverse \mathbf{T}^+ (which is one of many left-inverses) is that the sum of squares of the coordinates is as small as possible. This is the key to frames: *control the sum of squares of coordinates*. Now we define a frame in infinite dimensions.

Definition. An analysis frame is a set of vectors \mathbf{r}_n such that

$$A\|\mathbf{v}\|^2 \leq \sum |\langle \mathbf{r}_n, \mathbf{v} \rangle|^2 \leq B\|\mathbf{v}\|^2 \quad \text{for all } \mathbf{v}. \quad (2.49)$$

$A > 0$ and $B > 0$ are the “frame bounds.” A tight frame has $A = B$.

The frame vectors \mathbf{r}_n are not required to be independent. But they span the space! The only vector \mathbf{v} orthogonal to every \mathbf{r}_n is the zero vector, by (2.49).

The frame operator \mathbf{T} transforms \mathbf{v} into the sequence of numbers $\langle \mathbf{r}_n, \mathbf{v} \rangle$. In N dimensions these are M numbers, with $M > N$. In infinite dimensions we have infinitely many numbers, and the key is to recover \mathbf{v} from these numbers. The purpose of the frame bounds A and B is to make $\mathbf{T}^*\mathbf{T}$ and $(\mathbf{T}^*\mathbf{T})^{-1}$ bounded operators:

$$\text{Note carefully that } \langle \mathbf{v}, \mathbf{T}^*\mathbf{T}\mathbf{v} \rangle = \langle \mathbf{T}\mathbf{v}, \mathbf{T}\mathbf{v} \rangle = \sum |\langle \mathbf{r}_n, \mathbf{v} \rangle|^2. \quad (2.50)$$

Inserting into (2.49) gives $A\|\mathbf{v}\|^2 \leq \langle \mathbf{v}, \mathbf{T}^* \mathbf{T} \mathbf{v} \rangle \leq B\|\mathbf{v}\|^2$. The operator $\mathbf{T}^* \mathbf{T}$ is bounded by B . Its inverse is bounded by $1/A$. When we choose the tightest A and B , which we might as well do, the norms of $\mathbf{T}^* \mathbf{T}$ and its inverse are exactly B and $1/A$. *The frame ratio B/A is the condition number of $\mathbf{T}^* \mathbf{T}$.*

The frame bounds ensure that \mathbf{v} can be stably recovered from the components $\langle \mathbf{r}_n, \mathbf{v} \rangle$ of $\mathbf{T} \mathbf{v}$. The recovery operator — the synthesis operator — is the left-inverse $\mathbf{T}^+ = (\mathbf{T}^* \mathbf{T})^{-1} \mathbf{T}^*$:

$$\mathbf{v} = \mathbf{T}^+ \mathbf{T} \mathbf{v} = \sum (\text{column } n \text{ of } \mathbf{T}^+) \langle \mathbf{r}_n, \mathbf{v} \rangle = \sum c_n \mathbf{v}_n^+. \quad (2.51)$$

The coordinates are $c_n = \langle \mathbf{r}_n, \mathbf{v} \rangle$. The synthesis frame vectors \mathbf{v}_n^+ are the columns of \mathbf{T}^+ . When $\{\mathbf{r}_n\}$ is a basis, \mathbf{T}^+ is \mathbf{T}^{-1} and the dual frames are dual bases.

Example 2.13. The three rows of \mathbf{T} are an analysis frame (not tight) for \mathbf{R}^2 :

$$\mathbf{T} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{T}^* \mathbf{T} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}.$$

Any two rows of \mathbf{T} are a basis. The three rows are a frame. The frame bounds are $A = 2$ and $B = 3$. Those are the extreme eigenvalues of $\mathbf{T}^* \mathbf{T}$ — easy to find in this example. There are infinitely many left-inverses of \mathbf{T} , and here are three:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -1 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{T}^+ = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}.$$

The last one is the best one! It is the pseudoinverse $(\mathbf{T}^* \mathbf{T})^{-1} \mathbf{T}^*$. You see $\frac{1}{3}$ and $\frac{1}{2}$ from inverting $\mathbf{T}^* \mathbf{T}$. The columns of \mathbf{T}^+ are the dual frame vectors \mathbf{v}_1^+ and \mathbf{v}_2^+ and \mathbf{v}_3^+ .

Why is \mathbf{T}^+ best? Because it is the “smallest” left-inverse. We are reconstructing any vector \mathbf{v} in \mathbf{R}^2 from the three components c_1, c_2, c_3 of $\mathbf{T} \mathbf{v}$. In exact arithmetic, the only solution to $\mathbf{T} \mathbf{v} = \mathbf{c}$ is \mathbf{v} — and every left-inverse will find it. In actual arithmetic, and in actual measurements, there are errors in \mathbf{c} . We choose the least-squares solution to $\mathbf{T} \mathbf{v} = \mathbf{c}$. That comes from the normal equations $\mathbf{T}^* \mathbf{T} \mathbf{v} = \mathbf{T}^* \mathbf{c}$, whose solution is exactly $\mathbf{v} = \mathbf{T}^+ \mathbf{c}$. The synthesis frame reconstructs the truest \mathbf{v} .

Frames are associated with *oversampling* and *redundancy*. We meet this in irregular sampling when it is difficult to sample at exactly the right rate. Better to sample too often than too seldom. Oversampling is usually better than undersampling. The interpolating functions $\text{sinc}(t - t_n)$ are a basis for the band-limited space when the sampling rate is exactly right, and they are a frame when we oversample. They cannot reproduce all functions if we undersample.

For regular sampling at the times $t_n = nT$, the perfect rate is the Nyquist rate in Section 2.2. It requires two samples per period — the exact band of frequencies is $|\omega| \leq \pi/T$. In reality we often take more measurements than necessary, and reconstruct the signal by least squares.

The least-squares matrix is exactly $\mathbf{T}^ \mathbf{T}$!* This is the matrix in the normal equations, when \mathbf{T} has too many rows ($M > N$). The coefficient matrix \mathbf{T} is $M \times N$ and not invertible. But $\mathbf{T}^* \mathbf{T}$ is invertible exactly when the rows of \mathbf{T} are a frame. Instead of \mathbf{T}^{-1} the least-squares solution uses $\mathbf{T}^+ = (\mathbf{T}^* \mathbf{T})^{-1} \mathbf{T}^*$.

Note that a pseudoinverse \mathbf{T}^+ is defined for all matrices, of arbitrary rank [S]. \mathbf{T}^+ is a left-inverse when the rank is N and the columns of \mathbf{T} are independent and the rows of \mathbf{T} are a frame. Those are equivalent statements about \mathbf{T} in finite dimensions.

Mallat [Mt] identifies two approaches to the reconstruction of the signal v . If T^+ will be used often, we may compute it explicitly. Its columns are the synthesis frame vectors v_n^+ . For limited use we do not want the inverse of T^*T , only the solutions to specific equations with this coefficient matrix. Just as in linear equations $Ax = b$, we seldom want A^{-1} and we more often want $x = A^{-1}b$.

The iterative solution of the reconstruction problem for irregular sampling has been studied by Grochenig. This is an important application. His algorithms are much better than the simplest iterations.

We mention Kadec's $\frac{1}{4}$ Theorem for $L^2[0, 1]$: the exponentials $\{e^{2\pi i c_n t}\}$ are a basis if $|c_n - n|$ stays below a number $L < \frac{1}{4}$. This is nearly a regular sampling.

In our applications, the “rows” of T are usually functions $r_n(t)$. Then T maps functions $f(t)$ in L^2 to sequences $c_n = \langle r_n, f \rangle$ in ℓ^2 . The matrix TT^* contains the inner products of those rows with themselves. It is the “Gram matrix” whose (i, j) entry is the inner product $\langle r_i, r_j \rangle$. *The Riesz bounds A and B come from this matrix.* Its norm is B and the norm of its inverse is $\frac{1}{A}$.

Those numbers measure the linear independence of the functions $r_n(t)$. A case of special interest is when the rows $r_n(t)$ are translates $\phi(t - n)$ of a single scaling function. Theorem 6.6 will give a formula for A and B in this case, involving $\widehat{\phi}(\omega)$.

Summary: The bounds on TT^* are the *Riesz constants*—measuring independence of the rows of T . The bounds on T^*T are the *frame bounds*. When T is invertible, these ideas are the same—we have a Riesz basis. When T is not invertible the Riesz lower bound is $A = 0$.

Question: Could a frame include the zero vector $r = 0$?

Answer: Yes. The inner product $\langle 0, f \rangle = 0$ would not harm (or help) the frame bounds. A zero row of T has no effect at all on T^*T . And it destroys TT^* .

Question: Do a lowpass and highpass filter jointly produce a frame?

Answer: Yes, if their frequency responses have $|C(\omega)| + |D(\omega)| \geq A > 0$.

Question: When would the two filters jointly produce a tight frame?

Answer: When $|C(\omega)|^2 + |D(\omega)|^2 = \text{constant}$. This happens in an orthonormal filter bank. The frame constant is $2!$ Subsampling removes the redundancy and reduces that constant to 1. The tight frame becomes an orthonormal basis.

Construction of Frames

In finite dimensions, any set of vectors that spans \mathbf{R}^N is a frame. In infinite dimensions, the frame condition is not so simple. *The good constructions start with only one function.* We never want to compute with an infinite set of totally unrelated functions. The natural idea is to create an infinite family from that one function, in a systematic way.

Two systems are of special importance and they lead to *windows* and *wavelets*:

1. *Windows* come from one window $g(t)$ by *modulation* and *translation*:

$$g_{mn}(t) = e^{im\Omega t} g(t - nT). \quad (2.52)$$

2. *Wavelets* come from one wavelet $w(t)$ by *dilation* and *translation*:

$$w_{jk}(t) = a^{j/2} w(a^j t - kT). \quad (2.53)$$

The indices m , n , j , and k extend over the set \mathbf{Z} of all integers. We have a family $g_{mn}(t)$ of windows and a family $w_{jk}(t)$ of wavelets. The numbers Ω and T and a are fixed. *Those numbers decide whether the family is a frame.* Dividing T by 2 gives twice as many functions — increasing the likelihood of a frame. Dividing Ω by 2 works the same way; so does replacing a by \sqrt{a} . The crucial parameter is ΩT for the window family and $T \log a$ for the wavelet family. When these parameters are small, we have more functions and more likely a frame.

As T and Ω and $\log a$ approach zero, we begin to lose in efficiency. The ratio B/A eventually increases — the frame becomes excessively redundant. (Possibly B/A is a convex function of ΩT .) Most of this book is about $a = 2$ and $T = 1$ and special wavelets $w(t)$, leading not only to a frame but to a basis. Section 8.4 is about $\Omega = 1$ and $T = 1$ and special windows, again leading to a basis. Here we are only aiming for frames — much easier.

A point about normalization. The factor $a^{j/2}$ is included in the wavelets to keep the same norm. Without this factor the norms would go to zero as $j \rightarrow \infty$. The information to reconstruct $f(t)$ is still available in the inner products with the frame vectors, but the scaling is poor. This emphasizes that in infinite dimensions, the frame must span the space *stably*.

Window Frames: Examples and Theorems

Example 2.1. Suppose $g(t)$ is the unit box function on $[0, 1]$. For $T > 1$ the $g_{mn}(t)$ are not a frame. None of the boxes $g(t - nT)$ overlap the interval $[1, T]$. Any function $f(t)$ supported on this interval has zero inner product with all the windows. These spaced-out windows do not span L^2 .

For $T = 1$ the translated boxes $g(t - n)$ fit tightly. Within each box we have exponentials $e^{im\Omega t}$. For $\Omega = 2\pi$ this is an orthonormal basis on the interval $[0, 1]$. For $\Omega < 2\pi$ it is a frame.

We see how a basis appears on the “edge” of a family of frames. As soon as Ω passes 2π , there are not enough exponentials $e^{im\Omega t}$ and the frame is lost. (For $\Omega = 4\pi$ we will completely miss $e^{i2\pi t}$.) In the next example, and often, no basis appears — we go directly from a frame for small ΩT to failure for larger ΩT .

Example 2.2. Suppose $g(t)$ is the Gaussian function $e^{-t^2/2}$ as in [Gabor]. It produces a frame if $\Omega T < 2\pi$. It does *not* produce a frame if $\Omega T = 2\pi$, although this was Gabor’s favorite! At the edge of the frames, these functions $g_{mn}(t) = e^{2\pi imt} e^{-(t-n)^2/2}$ do span the space — but not stably. The inner products $\langle f, g_{mn} \rangle$ uniquely determine $f(t)$, but the operator $\mathbf{T}^* \mathbf{T}$ is not bounded below. The reconstruction formula from those inner products is not numerically stable.

For $\Omega T < 2\pi$ the tables in [D, p. 87] show how B/A depends on Ω and T .

We now state four general results about window frames, without proof. Three are positive, one is negative. The negative one is famous and comes first:

A. *Window frames are impossible with $\Omega T > 2\pi$. Smooth and decaying window frames are impossible with $\Omega T = 2\pi$.*

“Smooth and decaying” in this Balian-Low Theorem [BeHaWa] means that $g'(t)$ and $t g(t)$ are in $L^2(\mathbf{R})$. The window bases in Section 8.4 escape this restriction by a simple change in their construction (*cosines replace $e^{2\pi imt}$*). The impossibility for $\Omega T > 2\pi$ first came from Rieffel.

B. *The dual to a window frame comes from a dual window $\tilde{g}(t)$.*

This makes it worthwhile to compute the dual window. One function $\tilde{g}(t)$ gives the whole dual frame. In many other cases it is too much work to construct $(\mathbf{T}^* \mathbf{T})^{-1} \mathbf{T}^*$, whose columns contain the dual frame. Instead we solve a linear system for the synthesis coefficients in $\sum c_n \mathbf{r}_n$.

C. *Suppose $\sum |g(t - nT)|^2$ stays between positive constants for all t . Then there is a positive threshold Ω_0 such that $\Omega < \Omega_0$ yields a frame [D, p. 82].*

This assumes that $g(t)$ has compact support or $|t|^\alpha |g(t)| \rightarrow \infty$ for some $\alpha > 1$.

D. The frame bounds satisfy $A \leq 2\pi \|g\|/\Omega T \leq B$.

Wavelet Frames: Examples and Theorems

Example 2.1. The *Mexican hat* $w(t) = (1 - t^2)e^{-t^2/2}$ is the second derivative of the Gaussian. This wavelet is often used in analyzing vision. It has very rapid decay of both w and \hat{w} . For $a = 2$ and small T , the tables in [D, p. 77] show that B/A is very near 1. As T increases we suddenly lose the frame.

Example 2.2. The *Morlet wavelet* is a modulated Gaussian (complex for real signals):

$$w(t) = e^{-i\omega_0 t} e^{-t^2/2} \quad \text{with} \quad \omega_0 = \pi \sqrt{\frac{2}{\ln 2}} = 5.336.$$

This shift in frequency almost gives $\int w(t) dt = 0$ as required for wavelets. The actual value is below 10^{-6} , and negligible. The *phase* plot of the wavelet coefficients $\langle f, w_{jk} \rangle$ becomes very useful [Mt] in locating singularities of $f(t)$.

Some applications of these two examples use N different wavelets, called *voices*. This usually produces a tighter frame; B/A comes near 1. A good way to spread the N voices over an octave is by *fractional dilation* of a single wavelet:

$$w_\ell(t) = w(2^{-\ell/N}t), \quad \ell = 0, \dots, N-1. \quad (2.54)$$

The negative statement A and positive B in the window rules are reversed for wavelets.

A'. *There are wavelet frames (even orthonormal bases) for large values of $T \log a$. The restriction $\Omega T < 2\pi$ on smooth window frames does not apply to wavelets.*

B'. *The dual to a wavelet frame does not always come from one dual wavelet.*

C'. *If $\sum |\hat{w}(a^j \omega)|^2$ stays between positive constants for all ω , and $w(t)$ is smooth, there is a positive threshold such that $T < T_0$ yields a frame [D, p. 69].*

D'. *The frame bounds A and B are related to the wavelet constant C by*

$$A \leq \frac{C}{2T \log a} \leq B \quad \text{with} \quad C = 2\pi \int_{-\infty}^{\infty} |\hat{w}(\omega)|^2 \frac{d\omega}{|\omega|}. \quad (2.55)$$

For a tight frame equality holds. For an orthonormal basis $A = B = 1$. This constant C will be crucial for the integral wavelet transform in the next section.

Problem Set 2.5

1. Find a formula for the Walsh function that comes from the choices $- + -$. With $J = 3$ this function $w_{-+ -}(t)$ is equal to 1 or -1 on eight subintervals of $[0, 1]$.
2. Find a general formula for the Walsh functions at level $J = 3$. The numbers p, q, r give the three decisions $+1$ or -1 .
3. Count how many wavelet packets do not go beyond the level $J = 3$.
4. In Example 2.13 with $\mathbf{T} = [1 \ 1 \ 1; \ 1 \ 0 \ -1]$ suppose the vector \mathbf{v} is $[3 \ 4]'$.
 1. What are its three inner products c_1, c_2, c_3 with the rows of \mathbf{T} ?
 2. Verify that $2\|\mathbf{v}\|^2 \leq c_1^2 + c_2^2 + c_3^2 \leq 3\|\mathbf{v}\|^2$. The frame bounds are $A = 2$ and $B = 3$.
 3. Verify that $c_1\mathbf{v}_1^+ + c_2\mathbf{v}_2^+ + c_3\mathbf{v}_3^+$ reconstructs this \mathbf{v} .
5. Suppose the frame vectors are $\mathbf{r}_1 = [0 \ 2]$ and $\mathbf{r}_2 = [1 \ 1]$ and $\mathbf{r}_3 = [2 \ 0]$. Compute $\mathbf{T}^*\mathbf{T}$ and its eigenvalues A and B (the frame bounds). Also compute \mathbf{T}^+ and the dual frame.
6. The bounds for the dual frame (in the columns of \mathbf{T}^+) are $\frac{1}{B}$ and $\frac{1}{A}$. Prove this by showing that $\mathbf{T}^+(\mathbf{T}^+)^*$ equals $(\mathbf{T}^*\mathbf{T})^{-1}$. The extreme eigenvalues are _____, and

$$\frac{1}{B}\|\mathbf{v}\|^2 \leq \sum |\langle \mathbf{v}_n^+, \mathbf{v} \rangle|^2 \leq \frac{1}{A}\|\mathbf{v}\|^2.$$

7. A tight frame has $\mathbf{T}^*\mathbf{T} = A\mathbf{I}$. Explain why this is equivalent to $\sum |\langle \mathbf{r}_n, \mathbf{v} \rangle|^2 = A\|\mathbf{v}\|^2$. The frame bounds A and B are equal. The analysis frame $\{\mathbf{r}_n\}$ and synthesis frame $\{\mathbf{v}_n^+\}$ are both tight.
 8. The M th roots of 1 are the complex numbers with coordinates $(\cos \frac{2\pi k}{M}, \sin \frac{2\pi k}{M})$. Show that these M vectors in \mathbf{R}^2 form a tight frame.
 9. Prove that $\{e^{icnt}\}$ is a tight frame for $c < 1$ with constant $A = \frac{1}{c}$.
- Hint: Change $\int_0^{2\pi} v(t)e^{-icnt} dt$ to $\int_0^{2\pi} g(s)e^{ins} ds$ where $g(s) = \frac{1}{c}v\left(\frac{s}{c}\right)$ up to $s = 2\pi c$ (then zero). The sum of squares of coefficients is $\|g\|^2$ because $\{e^{ins}\}$ is _____. Verify that $\|g\|^2 = \frac{1}{c}\|v\|^2$ so that $A = \frac{1}{c}$.

2.6 Time, Frequency, and Scale

This book emphasizes bases more than frames. The synthesis operators are inverses rather than pseudo-inverses. The analysis bank produces the “right” number of outputs—the data is critically sampled and not oversampled. We want bases that are convenient for computation (by fast transform) and well adapted to the signal (for high compression).

This section is different. First it looks at transforms in continuous time and continuous frequency and continuous scale. Reconstruction is by an integral instead of a sum. This allows a very wide choice of windows and wavelets, in the Short Time Fourier Transform and the Integral Wavelet Transform. The analysis and synthesis formulas are simple and general. But when we sample in time and frequency and scale—in order to select a discrete set as a basis or a frame—the conditions on the windows and wavelets become tighter.

At the end of the section we relax by using many more functions than necessary. Those are **time-frequency atoms**—wavelets or windows or whatever. The search for the best representation from a big dictionary of atoms is a very active problem.

May we first contrast windows with wavelets? The competition between them is far from over. Both have the goal of localizing the basis functions. The windowing functions $g(t)$ achieve

that by dropping quickly to zero — they can be Gabor windows with the Gaussian factor e^{-t^2} , or they can have finite length. The shifted window $g(t - s)$ is localized around time s . The expansion functions are oscillations inside the window:

$$\text{Windowed exponentials } g_{\omega,s}(t) = g(t - s)e^{i\omega t}.$$

These are “time-frequency atoms.” When sampled at discrete times $t = n$ and discrete frequencies $\omega = 2\pi k$, the localized exponentials are $g(t - n)e^{2\pi i k t}$. If $g(t)$ is a unit box, these functions give a basis — but not smooth. When $g(t)$ is a smooth window we don’t have a basis (Balian-Low Theorem). The samples are over-complete (a frame) or they are incomplete — depending on $g(t)$ and the sampling rate. But Section 8.4 will show how smooth local cosines can give a very satisfactory basis when the frequencies are shifted to $k + \frac{1}{2}$.

Here we keep *all times s and all frequencies ω* . The familiar integral Fourier transform (no window) is a function of ω :

$$\widehat{f}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt. \quad (2.56)$$

The windowed Fourier transform is a function of ω and also the position s :

$$\text{Windowed Transform } F(\omega, s) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) g(t - s) e^{-i\omega t} dt. \quad (2.57)$$

This is the Fourier transform of the windowed functions $f(t) g(t - s)$ for all s . Without the windows, the reconstruction of $f(t)$ from $\widehat{f}(\omega)$ is famous:

$$f(t) = \int_{-\infty}^{\infty} \widehat{f}(\omega) e^{i\omega t} d\omega. \quad (2.58)$$

With the windows, we recover $f(t) g(t - s)$ by this integral over ω . Equation (2.58) for each s is

$$f(t) g(t - s) = \int_{-\infty}^{\infty} F(\omega, s) e^{i\omega t} d\omega. \quad (2.59)$$

Now multiply both sides by $g(t - s)$ (or $\overline{g(t - s)}$ if complex) and integrate over s :

$$f(t) = \frac{1}{\|g\|^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\omega, s) g(t - s) e^{i\omega t} d\omega ds. \quad (2.60)$$

This is reconstruction from the time-frequency transform $F(\omega, s)$.

Compare this windowed STFT with wavelets, which begin with *one function $w(t)$* . The position variable s still comes from translation to $w(t - s)$. Now comes the difference. *Instead of the frequency variable we have a scale variable*. Instead of modulating the wavelets we rescale them. The “time-scale atoms” are translates and dilates of $w(t)$:

$$\text{Wavelet functions } w_{a,s}(t) = |a|^{-1/2} w\left(\frac{t-s}{a}\right).$$

The mother wavelet $w(t)$ is $w_{1,0}(t)$ at unit scale $a = 1$ and position $s = 0$. The factor $|a|^{-1/2}$ assures that the rescaled wavelets have equal energy $\|w_{a,s}\| = \|w\|$. We normalize so that all these functions have unit norm $\|w_{a,s}\| = 1$.

Notice how scaling the time by a or a^j automatically scales the translation steps by a^{-1} or a^{-j} :

$$w(a^j t - k) = w(a^j[t - ka^{-j}]).$$

The mesh length at level j is scaled down by a^{-j} . The “frequency” is scaled up by a^j . This hyperbolic scaling or dyadic scaling or octave scaling ($a = 2$) is a prime characteristic of wavelet analysis.

The integral wavelet transform is an inner product with wavelets, just as the windowed transform was an inner product with windowed exponentials:

$$\text{Integral Wavelet Transform} \quad F_w(a, s) = |a|^{-1/2} \int_{-\infty}^{\infty} f(t) w\left(\frac{t-s}{a}\right) dt. \quad (2.61)$$

The transform F_w is defined on the *time-scale plane*. Again there are two variables, but the scale a has replaced the frequency ω . The subscript in F_w indicates this change.

The wavelet transform $F_w(a, s)$ is redundant, just as the windowed transform $F(\omega, s)$ was redundant. If we select a good wavelet, it will be enough to know F_w at a discrete set of scales and positions. That is the construction to implement digitally. Here we reconstruct $f(t)$ from its over-complete transform $F_w(a, s)$.

Important: The key to scaling is not a but $\log a$. The natural scale is *logarithmic*. The differential is not da but da/a . The frequency window da is proportional to a (this is often called *constant-Q*). Convolution with $\frac{1}{a} w\left(\frac{t}{a}\right)$ gives a unitary operator, when we integrate with respect to logarithmic measure da/a :

$$\int_0^\infty C_a C_a^* \frac{da}{a} = \mathbf{I} \quad \text{for convolution } C_a \text{ with } \frac{1}{a} w\left(\frac{t}{a}\right).$$

This is *Calderón's identity*, rediscovered by Grossmann and Morlet. It is proved in Theorem 2.5 below, with a different normalization of the wavelet and wavelet transform (each by $|a|^{-1/2}$). It gives the reconstruction formula that inverts (2.61) and recovers $f(t)$:

Reconstruction from Wavelet Transform

$$f(t) = \frac{1}{C} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F_w(a, s) |a|^{-1/2} w\left(\frac{t-s}{a}\right) \frac{da ds}{a^2}. \quad (2.62)$$

The constant for windows was $C = \|g\|^2$. The constant for wavelets is not $\|w\|^2$, which would be 2π times the integral of $|\widehat{w}|^2$. Instead the constant is $C = 2\pi \int |\widehat{w}|^2 d\omega / |\omega|$. Effectively, C is finite when the transform of the wavelet is zero at $\omega = 0$. This means that *the integral of the wavelet is zero*. Any smooth decaying function $w(t)$ is a mother wavelet for the integral transform provided $\int w(t) dt = 0$.

The Haar wavelet was not the box function. It was the up-down square wave with integral zero, one box minus another box. Other wavelets will be combinations of scaling functions $\phi(2t - k)$, with coefficients that add to zero. Those coefficients come from the highpass filter! They will be specially chosen, and $\phi(t)$ will be specially chosen, to give a discrete basis. In the continuous time-scale plane we only require that $\int w(t) dt = 0$.

The reconstruction formula (2.62) comes from the following theorem.

Theorem 2.5 For any $f(t)$ and $g(t)$ in L^2 , and C as above,

$$C \int_{-\infty}^{\infty} f(t) \overline{g(t)} dt = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F_w(a, s) \overline{G_w(a, s)} \frac{da ds}{|a|^2}. \quad (2.63)$$

With $f = g$ this is a “Parseval formula” in the time-scale plane. The energy in f equals the energy in its wavelet transform, when the time-scale area is measured correctly.

Ironically, the simplest proof of (2.63) uses the Fourier transform. For each a , the transform in (2.61) is a convolution of $f(t)$ with the scaled wavelet $|a|^{-1/2}w\left(\frac{-t}{a}\right)$. The Fourier transform of this convolution F_w is a multiplication $|a|^{1/2}\widehat{f}(\omega)\widehat{w}(a\omega)$. Similarly, the transform of \overline{G}_w is a multiplication $|a|^{1/2}\widehat{\overline{g}}(\omega)\widehat{w}(a\omega)$. Then for each a , the integral over s on the right side of (2.63) becomes

$$\int_{-\infty}^{\infty} F_w(a, s) \overline{G_w(a, s)} ds = 2\pi \int_{-\infty}^{\infty} |a| \widehat{f}(\omega) \overline{\widehat{g}(\omega)} |\widehat{w}(a\omega)|^2 d\omega. \quad (2.64)$$

Integrate this with respect to $da/|a|^2$ and reverse the order of integration. The integral over a gives the predicted constant C , after changing variables to $a\omega$. The right side of (2.63) becomes

$$2\pi \int_{-\infty}^{\infty} \widehat{f}(\omega) \overline{\widehat{g}(\omega)} \int_{-\infty}^{\infty} |\widehat{w}(a\omega)|^2 da d\omega = 2\pi C \int_{-\infty}^{\infty} \widehat{f}(\omega) \overline{\widehat{g}(\omega)} d\omega. \quad (2.65)$$

This equals the left side of (2.63) and completes the proof.

Suppose $g(t) = \delta(t)$ is the Dirac delta function (not in L^2). Then equation (2.63) formally becomes the reconstruction formula (2.62) at $t = 0$. The wavelet transform $G_w(a, s)$ of $\delta(t)$ is the wavelet $|a|^{-1/2}w\left(-\frac{s}{a}\right)$. This appears in (2.62) when $t = 0$. By shifting the delta function we reach the reconstruction formula at all t .

This use of the delta function is more legal and familiar than it seems. The ordinary Fourier reconstruction in (2.58) is reached the same way. The analogue of (2.63) is Parseval’s equation

$$\int_{-\infty}^{\infty} f(t) \overline{g(t)} dt = 2\pi \int_{-\infty}^{\infty} \widehat{f}(\omega) \overline{\widehat{g}(\omega)} d\omega \text{ for all } f, g \text{ in } L^2. \quad (2.66)$$

With $g(t) = \delta(t)$ this is the reconstruction $f(0) = \int \widehat{f}(\omega) d\omega$ at $t = 0$. A shifted delta function gives the inversion formula at any other t . These integrals are correct *at each t* when f is smooth. They are correct *in the L^2 sense* for a general function f in L^2 . Technically, we smooth f by restricting its transform to $|\omega| \leq \Omega$ and then let $\Omega \rightarrow \infty$.

Note. For real wavelets $w(t)$, when $|\widehat{w}|^2$ is an even function, integrate only over $a > 0$ and compensate by taking only half of the constant C :

$$C \rightarrow \int_0^{\infty} |\widehat{w}(\omega)|^2 \frac{d\omega}{\omega} = \frac{1}{2}C.$$

The Wigner-Ville Transform

The time-frequency analysis of a signal goes much further than windows and wavelets. By restricting to the choice of one function $g(t)$ or $w(t)$, we create good algorithms — which is our principal purpose. By allowing more general transforms, we can hope to get precise information about the “instantaneous frequency” and “instantaneous spectrum” of a signal. The windows and wavelets do a little smearing — not too much, and the time-frequency trace of the signal can be followed. But Wigner and Ville and many others wanted a sharp trace.

We shall devote just a very short space to this basic topic — the correlation of a signal with itself. Everywhere else we are correlating the signal with chosen windows or wavelets. As those

move and rescale, we identify our signal from the correlations. The self-correlation gives an energy density that should automatically pick out the position and frequency and scale of the signal.

We define the *Wigner-Ville transform* of a finite energy function $f(t)$:

$$W(t, \omega) = \int_{-\infty}^{\infty} f\left(t + \frac{\tau}{2}\right) \overline{f\left(t - \frac{\tau}{2}\right)} e^{-i\omega\tau} d\tau. \quad (2.67)$$

Notice especially that W is quadratic in f . We expect $W(t, \omega)$ to be an “energy density”—like the *square* of the window and wavelet transforms, but better. This density has several desirable properties:

$$\frac{1}{2\pi} \int W(t, \omega) d\omega = |f(t)|^2 \quad \text{and} \quad \int W(t, \omega) dt = |\widehat{f}(\omega)|^2. \quad (2.68)$$

$W(t - T, \omega - \Omega)$ is the transform of $e^{i\Omega t} f(t - T)$.

$W\left(\frac{t}{a}, a\omega\right)$ is the transform of $\frac{1}{a} f\left(\frac{t}{a}\right)$.

$W(t, \omega)$ determines $f(t)$ up to a constant multiplier $|c| = 1$.

The transform goes from one variable to two variables. As with wavelets and windows, most functions of two variables are *not* transforms of any $f(t)$. The transform of \widehat{f} flips the variables to reach $W(\omega, -t)$. And the Gaussian plays a special role. We get $W = e^{-t^2 - \omega^2}$ from the unit Gaussian and we get modulations of $W = e^{-p^2 t^2 - q^2 \omega^2 \pm pqt\omega}$ from all quadratic “chirps.”

What good properties does this transform *not* have? First, it is not always positive. Second, the sum of two Gaussians (modulated and shifted) has a transform with *four terms*. Two terms are in the expected positions (T_1, Ω_1) and (T_2, Ω_2) , as desired from the Gaussians. Two other terms are in the wrong positions (T_1, Ω_2) and (T_2, Ω_1) . These *ghosts* are highly oscillatory.

“Analytic signals” have no negative frequencies: $\widehat{f}(\omega) = 0$ for $\omega < 0$. Restricted to these signals, the transform is a success. Then $W(t, \omega)$ earns the name “instantaneous spectrum” of $f(t)$. And the instantaneous frequency —*the derivative of the phase* $\phi(t)$ — equals the average $\frac{1}{2\pi} \int \omega W(t, \omega) d\omega$ for a large class of signals.

Summary: The integral transforms, by windows and wavelets, make minimal demands on the functions $g(t)$ and $w(t)$. We can analyze and synthesize (transform and inverse transform) with extremely general functions. Reality sets in when the transform becomes discrete—the shifts are multiples of T , the modulations are multiples of Ω , the scales are powers of a . The elegant books by Meyer give an overview of this whole picture.

The rest of this text deals with the discrete transform. We will have very strict conditions on the wavelets! The scales are powers of a fixed integer $a = M$, most often $a = M = 2$. Extra conditions are imposed on the wavelet. There are absolute requirements and optional properties. The requirements make it work, the extra conditions make it work well. The goal is to achieve these properties with a fast algorithm:

1. Two-scale or M -scale equation (with special coefficients)
2. Smoothness of the wavelet (optional but desirable)
3. Symmetry or antisymmetry (optional but very desirable)
4. Vanishing moments (one required, more desirable).

Atomic Decompositions

Whether they are wavelets or sinusoids or cosine packets or Gabor functions, we are approximating $f(t)$ by “atoms”. A collection of atoms is a “dictionary”. We need a reasonable algorithm to choose the nearly best M atoms for a given $f(t)$, out of a dictionary of P atoms. A single basis may be too inflexible, and *the algorithm must adapt to the signal*. Major effort is going into algorithms *not* based on one orthonormal basis.

We will only mention three ideas and their developers. The first is **matching pursuit** (S. Mallat and Z. Zhang). The M atoms are chosen one at a time. The choice at step $k+1$ is the atom that comes closest to the current difference $f_k(t) = f(t) - c_1\phi_1(t) - \dots - c_k\phi_k(t)$. In practice matching pursuit takes (normalized) inner products $\langle f_k(t), \phi(t) \rangle$ and chooses a large one—an atom $\phi_{k+1}(t)$ that is highly correlated with $f_k(t)$. This is a *greedy and sub-optimal* selection process—greedy because each choice is ignorant of the later choices, sub-optimal because we don’t insist on maximizing $\langle f_k(t), \phi(t) \rangle$. The complexity is $O(N \ln N)$ at each iteration, for a signal of length N . The pursuit ends at $k = M$. We can then compute the best combination of the M choices (this is *back-projection*). Experimentally the error approaches white noise as $M \rightarrow \infty$.

A second adaptive method is the **best basis** algorithm (R. Coifman and V. Wickerhauser). This begins with a dictionary of bases, often orthonormal. The algorithm chooses the best basis to represent $f(t)$. For wavelet packets from a family of binary trees, the method is particularly well adapted. Available software includes the Wavelet Packet Laboratory for Windows (AK Peters, Wellesley MA 02181-5910). A modification that is optimal in the rate-distortion sense, for compression, is described in [VK, p. 426].

With orthonormal bases at every step, rates and mean-square distortions of the two branches are additive. This allows a fast algorithm to choose the packet that is best adapted to the particular signal. The reader understands that an actual implementation leads to many options. The mean-square ℓ^2 norm may be replaced by other norms (losing additivity at each split but gaining in perceptual quality).

A third method is **basis pursuit** (D. Donoho). The dictionary is still overcomplete. The synthesis $f(t) = \sum c_i \phi_i(t)$ (modelled by $f = \Phi c$) is underdetermined. *Frame theory chooses c to have a small ℓ^2 norm* (sum of c_i^2 is minimized, leading to linear equations and generalized inverses). *Basis pursuit chooses c to have a small ℓ^1 norm* (sum of $|c_i|$ is minimized, leading to nonlinear equations and linear programming). This minimizer is generally much sparser—fewer c_i are nonzero.

The ℓ^1 minimizer is also more expensive to compute. In place of the simplex method, interior point and log barrier algorithms associated with Karmarkar solve a sequence of weighted ℓ^2 problems. This can give reasonable success for large dictionaries ($P = 10^4$ and $N = 10^3$). The method can distinguish two nearby bumps in $f(t)$, where matching pursuit will select one centered bump that has high correlation. Linear programming pursues the best basis—which depends on the signal $f(t)$. Donoho also studies “empirical atomic decomposition” for $P \gg N$. This algorithm strongly controls the number M of atoms $\phi_k(t)$ in the approximation, by minimizing $\|f(t) - \sum c_i \phi_i(t)\|^2 + \lambda M$. The key is in the selection of $\lambda = \sigma \sqrt{2 \log P}$. Software is on the web at <http://playfair.stanford.edu>.

The multiplier λ becomes $\sigma \sqrt{2 \log N}$ for de-noising with an orthonormal basis. The noise is assumed Gaussian with standard deviation σ . The recommended solution is soft thresholding

of the inner products $y_i = \langle \phi_i, f \rangle$ with threshold λ . Each scalar y is shifted toward zero but not beyond:

$$\textbf{Thresholding} \quad c_{soft} = (y - \lambda)_+ \quad \text{for } y > 0 \quad \text{and} \quad (y + \lambda)_- \quad \text{for } y < 0.$$

Donoho notes that c_{soft} minimizes $\frac{1}{2}(y - c)^2 + \lambda|c|$. That $|c|$ is the l^1 norm again.

Finally we emphasize that the construction of wavelets has not ended. While writing the book we have learned of *directional* and *translation-invariant* and *steerable* wavelets. And the world of atoms is by no means restricted to wavelets!

Problem Set 2.6

1. Find the Wigner-Ville transform $W(t, \omega)$ when $f(t)$ is the Dirac function $\delta(t)$.
2. Show how $W(t, \omega)$ can also be expressed by an integral of \widehat{f} :

$$W(s, \xi) = \frac{1}{2\pi} \int \widehat{f}\left(\xi - \frac{\omega}{2}\right) \widehat{f}\left(\xi + \frac{\omega}{2}\right) e^{is\omega} d\omega.$$

Chapter 3

Downsampling and Upsampling

3.1 The Matrices for Downsampling and Upsampling

The previous chapter was about individual filters. The next chapter combines two or more filters into a *filter bank*. The idea is to separate the incoming signal into frequency bands. Often we will use a bank of two filters to explain a point clearly. A lowpass filter C and a highpass filter D will split the signal into two parts. Those parts can be compressed and coded separately (and efficiently). Then they can be transmitted and the signal can be recovered.

When the synthesis bank recovers the input signal exactly (apart from a time delay), it is a *perfect reconstruction* filter bank. These PR banks interest us most.

One problem to overcome. We don't want two full length signals in place of one, if we can help it. It is not desirable to double the volume of data. The information has not doubled; the outputs Cx and Dx from the two filters must be redundant.

The solution to this problem is beautiful. We keep only half of the outputs Cx and Dx . More precisely, we *downsample* those vectors by removing every other component. This operation is called *decimation*. Downsampling is represented by the symbol $(\downarrow 2)$ (pronounced “down two”):

$$(\downarrow 2) \begin{bmatrix} \cdot \\ y(-1) \\ y(0) \\ y(1) \\ y(2) \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot \\ y(-2) \\ y(0) \\ y(2) \\ y(4) \\ \cdot \end{bmatrix}.$$

Make no mistake. This operation is not invertible. Most vectors y cannot be recovered from $(\downarrow 2)y$. The odd-numbered components are lost. Normally the even-numbered components of both $y = Cx$ and $y = Dx$ will be needed to recover all components of x .

We note that recovery of x from $(\downarrow 2)x$ is possible if the transform $X(\omega)$ is zero over a half-band of frequencies. Such a signal is “band-limited”. It may be limited to the upper half-band or the lower half-band:

$$X(\omega) = 0 \text{ for } 0 \leq |\omega| < \frac{\pi}{2} \quad \text{or} \quad X(\omega) = 0 \text{ for } \frac{\pi}{2} \leq |\omega| < \pi.$$

It is amazing that for band-limited signals we can recover the odd-numbered components from the even-numbered components. This is achieved by the *Shannon Sampling Theorem* in Section 2.3.

Example 3.1. Downsampling a vector of alternating signs produces a vector of constant sign:

$$(\downarrow 2)(\dots, 1, -1, 1, -1, \dots) = (\dots, 1, 1, \dots).$$

The -1 's in the odd-numbered positions have disappeared. You will quickly think of other inputs that give the same output. One in particular is the input vector $(\dots, 1, 1, 1, 1, \dots)$ of all ones. Downsampling produces the same vector of ones. It is an eigenvector of the linear operator $(\downarrow 2)$, with eigenvalue $\lambda = 1$.

Another eigenvector is $\delta = (\dots, 0, 0, 1, 0, 0, \dots)$. This also has $\lambda = 1$, and $(\downarrow 2)\delta = \delta$.

It is valuable to compare the alternating and constant inputs. Vector $(\dots, 1, -1, 1, -1, \dots)$ is at the frequency $\omega = \pi$. That is the highest possible frequency in discrete time. No vector can oscillate faster. Its components are $x(n) = e^{i\pi n}$, which is $(-1)^n$. By contrast, the input of all ones is at the lowest frequency $\omega = 0$. Its components are $x(n) = e^{i0n} = 1$.

This is an example of *aliasing*. Two different frequencies, after downsampling, cannot be distinguished. The “alias” of the high frequency $\omega = \pi$ is the low frequency $\omega = 0$.

Certainly we cannot recover the input (which input would it be?) from the common output $(\downarrow 2)x = (\dots, 1, 1, 1, \dots)$. If we know that all frequencies in the input satisfy $|\omega| < \frac{\pi}{2}$, then we can “invert” the downsampling operation. The only possible input would be the constant vector, the DC input with $\omega = 0$.

Example 3.2. When the impulse response $C\delta = (\dots, c(0), c(1), c(2), \dots)$ is downsampled, the odd-numbered responses are lost:

$$(\downarrow 2)C\delta = (\dots, c(0), c(2), c(4), \dots).$$

This is the impulse response of a “decimation filter”—*filtering followed by downsampling*. The impulse δ came at time zero, and downsampling leaves the even coefficients. The response is entirely different if the impulse is delayed—we compute that now. **Downsampling is not time-invariant**. The delayed impulse is $S\delta = (\dots, 0, 1, 0, \dots)$. The response from the time-invariant filter C is similarly delayed:

$$CS\delta = (\dots, 0, c(0), c(1), \dots). \quad (3.1)$$

Because of the delay, downsampling now destroys the even-numbered components $c(0), c(2)$. We pick up the odd components:

$$(\downarrow 2)CS\delta = (\dots, 0, c(1), c(3), \dots).$$

First conclusion: Complete information may still be there if we downsample *two* vectors. Here the vectors were $C\delta$ and its shift $SC\delta = CS\delta$. Those outputs are not as “separated” as we want—the two pieces are probably far from orthogonal. Downsampling $C\delta$ and $CS\delta$ may be a waste of time, but we do see that the whole vector $(c(0), c(1), c(2), c(3), \dots)$ is recoverable from the two samples. Upsampling will be the way to recover it.

A better filter bank will use two different filters C and D . Part of the design problem is to ensure that $(\downarrow 2)Cx$ and $(\downarrow 2)Dx$ have all the information to recover x . Even better if those vectors are orthogonal, which is not true for this easy choice $D = \text{shift of } C$.

Definition 3.1 *The n th component of $v = (\downarrow 2)x$ is the $(2n)$ th component of x :*

$$v(n) = x(2n). \quad (3.2)$$

We will now identify the “downsampling matrix”. *It is the identity matrix with odd-numbered rows removed.* You could say that the identity matrix has been downsampled! The matrix form of downsampling is

$$\begin{bmatrix} \cdot & 1 & & & \\ & 0 & 0 & 1 & \\ & & 0 & 0 & 1 & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot \\ \mathbf{x}(-2) \\ \mathbf{x}(-1) \\ \mathbf{x}(0) \\ \mathbf{x}(1) \\ \mathbf{x}(2) \\ \cdot \end{bmatrix} = \begin{bmatrix} \mathbf{x}(-2) \\ \mathbf{x}(0) \\ \mathbf{x}(2) \\ \cdot \end{bmatrix}. \quad (3.3)$$

We are defining the linear operator $(\downarrow 2)$. Notice that the rows of the downsampling matrix are orthonormal. They are unit vectors and their dot products are all zero. The matrix multiplication $(\downarrow 2)(\downarrow 2)^T$ contains all these dot products, so it must be the identity matrix. The matrix product $(\downarrow 2)^T(\downarrow 2)$ is different, as we see below.

The operator $(\downarrow 2)^T$ is important. *The transpose of downsampling is upsampling:*

$$(\downarrow 2)^T = (\uparrow 2). \quad (3.4)$$

Upsampling places zeros into the odd-numbered components:

Definition of $(\uparrow 2)$: $(\uparrow 2)$

$$\begin{bmatrix} \cdot \\ \mathbf{v}(-1) \\ \mathbf{v}(0) \\ \mathbf{v}(1) \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot \\ \mathbf{v}(-1) \\ 0 \\ \mathbf{v}(0) \\ 0 \\ \mathbf{v}(1) \\ 0 \\ \cdot \end{bmatrix}.$$

We need a two-part formula to describe those components of $\mathbf{u} = (\uparrow 2)\mathbf{v}$:

$$\mathbf{u}(n) = \begin{cases} \mathbf{v}(k) & \text{if } n = 2k \\ 0 & \text{if } n = 2k + 1. \end{cases} \quad (3.5)$$

To do this with a matrix, insert zero rows between the rows of \mathbf{I} . You could say that *the identity matrix has been upsampled*. The matrix form of upsampling is

$$\begin{bmatrix} \cdot & 1 & 0 & & \\ & 0 & & & \\ & 1 & 0 & & \\ & & 0 & & \\ & & 1 & 0 & \\ & & & & \cdot \end{bmatrix} \begin{bmatrix} \cdot \\ \mathbf{v}(-1) \\ \mathbf{v}(0) \\ \mathbf{v}(1) \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot \\ \mathbf{v}(-1) \\ 0 \\ \mathbf{v}(0) \\ 0 \\ \mathbf{v}(1) \\ 0 \\ \cdot \end{bmatrix} \quad (3.6)$$

The matrix in (3.6) is the transpose of the matrix in (3.3). If we multiply them we do get the

identity matrix, verifying $(\downarrow 2)(\uparrow 2) = \mathbf{I}$:

$$\begin{bmatrix} \cdot & 1 & & & \\ & 0 & 0 & 1 & \\ & & 0 & 0 & 1 & \\ & & & 0 & 1 & \cdot \end{bmatrix} \begin{bmatrix} \cdot & & & & \\ 1 & 0 & & & \\ 0 & 0 & 1 & 0 & \\ & 1 & 0 & 0 & \\ & & 1 & 0 & \\ & & & \cdot & \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & \cdot \end{bmatrix}. \quad (3.7)$$

If we upsample and then downsample, we put in zeros and take them out. The original \mathbf{x} is recovered. This is not the usual order of the steps!

Normally we downsample and then upsample. *This product is not the identity matrix.* The output $(\uparrow 2)(\downarrow 2)\mathbf{x}$ is different from \mathbf{x} . Watch the steps:

$$\text{Down: } (\downarrow 2)\mathbf{x} = \begin{bmatrix} \mathbf{x}(-2) \\ \mathbf{x}(0) \\ \mathbf{x}(2) \\ \cdot \end{bmatrix}. \quad \text{Then up: } (\uparrow 2)(\downarrow 2)\mathbf{x} = \begin{bmatrix} \mathbf{x}(-2) \\ 0 \\ \mathbf{x}(0) \\ 0 \\ \mathbf{x}(2) \\ 0 \\ \cdot \end{bmatrix}. \quad (3.8)$$

In this usual order, $(\downarrow 2)$ removes the odd-numbered components and $(\uparrow 2)$ *puts in zeros*. The product $(\uparrow 2)(\downarrow 2)$ replaces the lost components by zeros:

$$(\uparrow 2)(\downarrow 2) = \begin{bmatrix} 1 & & & & \\ & 0 & & & \\ & & 1 & & \\ & & & 0 & \\ & & & & 1 \\ & & & & & \cdot \end{bmatrix} \quad (3.9)$$

Thus $(\downarrow 2)$ and $(\uparrow 2)$ are one-sided inverses but not two-sided inverses. This cannot happen for square matrices of finite size. For n by n matrices, $\mathbf{AB} = \mathbf{I}$ means that $\mathbf{BA} = \mathbf{I}$. For rectangular matrices and infinite matrices, a *one-sided inverse* is possible. Downsampling is a very important example.

The matrices for $(\downarrow 2)$ and $(\uparrow 2)$ came from downsampling and upsampling the columns of \mathbf{I} . Here is another way to describe those same matrices:

- The $(\downarrow 2)$ matrix has the rows of \mathbf{I} with a **double shift**.
- The $(\uparrow 2)$ matrix has the columns of \mathbf{I} with a **double shift**.

Problem Set 3.1

1. Downsampling the alternating vector $\mathbf{x} = (\dots, 1, -1, 1, -1, 1, \dots)$ produces the vector $(\downarrow 2)\mathbf{x} = (\dots, 1, 1, 1, \dots)$. If you delay \mathbf{x} to $\mathbf{Sx}(n) = \mathbf{x}(n-1)$, what is $(\downarrow 2)\mathbf{Sx}$?
2. Describe the operator $(\uparrow 2)(\downarrow 2)(\uparrow 2)(\downarrow 2)$.
3. Describe the operators $(\downarrow 3)$ and $(\uparrow 3)$ and $(\uparrow 3)(\downarrow 3)$.
4. What are the components of $(\uparrow 3)(\downarrow 2)\mathbf{x}$ and $(\downarrow 2)(\uparrow 3)\mathbf{x}$?

3.2 Subsampling in the Frequency Domain

For a pure exponential signal $\mathbf{x}(k) = e^{ik\omega}$, the effect of downsampling is particularly clear. The k th component of $\mathbf{v} = (\downarrow 2)\mathbf{x}$ is $v(k) = e^{i2k\omega}$. This is a pure exponential with frequency 2ω . Frequencies are doubled by downsampling.

It looks as if $V(2\omega) = X(\omega)$, in other words $V(\omega) = X\left(\frac{\omega}{2}\right)$. That is wrong.

Suppose \mathbf{x}' is another pure exponential, but with frequency $\omega + \pi$. Then $\mathbf{x}'(k) = e^{ik(\omega+\pi)}$ is downsampled to $\mathbf{v}'(k) = e^{i2k(\omega+\pi)}$. The factor $e^{i2k\pi}$ is always 1. This output is the same $e^{i2k\omega}$ as before! Adding π to the frequency only reversed the signs of the odd-numbered components of \mathbf{x}' . Those components disappear anyway in downsampling, and $\mathbf{v}'(k) = \mathbf{v}(k)$. The frequency 2ω appears by doubling ω , and it also appears by doubling $\omega + \pi$.

In other words, ω enters the downsampled vector \mathbf{v} not only by doubling $\frac{\omega}{2}$ but also by doubling $\frac{\omega}{2} + \pi$. There are two sources, not just one, contributing to $V(\omega)$:

Theorem 3.1 *The transform of $\mathbf{v} = (\downarrow 2)\mathbf{x}$ is $V(\omega) = \frac{1}{2}[X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega}{2} + \pi\right)]$.*

Proof. Downsampling in the time domain is very direct:

$$\mathbf{v} = (\downarrow 2)\mathbf{x} \text{ means } v(k) = \mathbf{x}(2k). \quad (3.10)$$

The Fourier transform of \mathbf{v} is

$$V(\omega) = \sum v(k)e^{-ik\omega} = \sum \mathbf{x}(2k)e^{-ik\omega}. \quad (3.11)$$

It is natural to set $n = 2k$, but do it carefully. The reverse direction $k = \frac{n}{2}$ is not safe. You might think that our sum is $\sum \mathbf{x}(n)e^{-in\omega/2}$, and the frequency variable ω is just halved. *But the sum is only over even $n = 2k$.* So we start again.

Let \mathbf{u} be the vector \mathbf{x} with its odd-numbered components set to zero:

$$\mathbf{u}(n) = \begin{cases} \mathbf{x}(n), & n \text{ even} \\ 0, & n \text{ odd} \end{cases} \text{ so that } \mathbf{u} = (\dots, \mathbf{x}(0), 0, \mathbf{x}(2), 0, \dots). \quad (3.12)$$

Certainly $(\downarrow 2)\mathbf{u}$ equals $(\downarrow 2)\mathbf{x}$. The transform of \mathbf{u} has two terms. The second term includes $(-1)^n$ or $e^{-in\pi}$ so that addition knocks out odd n :

$$\mathbf{u}(n) = \sum_{n \text{ even}} \mathbf{x}(n)e^{-in\omega} = \frac{1}{2} \sum_{\text{all } n} \mathbf{x}(n)e^{-in\omega} + \frac{1}{2} \sum_{\text{all } n} \mathbf{x}(n)e^{-in(\omega+\pi)}. \quad (3.13)$$

In frequency, this says that

$$U(\omega) = \frac{1}{2} [X(\omega) + X(\omega + \pi)]. \quad (3.14)$$

Now $V(\omega)$ comes safely from $U(\omega)$ by halving the frequency, because $\mathbf{u}(n)$ only involves even n :

$$V(\omega) = U\left(\frac{\omega}{2}\right). \quad (3.15)$$

Equations (3.15)–(3.16) mean that the downsampled signal $\mathbf{v} = (\downarrow 2)\mathbf{x}$ has transform

$$V(\omega) = \frac{1}{2} [X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega}{2} + \pi\right)]. \quad (3.16)$$

This is downsampling in the frequency domain. It is not time-invariant, because $X(\omega)$ is not just multiplied by $C(\omega)$. The rate changes and a new term appears.

Figures 3.1 and 3.2 and 3.3 show extreme aliasing and no aliasing and typical aliasing. You can see the two terms in Figure 3.1, where $X(\omega)$ is a sawtooth function. The average in $U(\omega)$ is a constant. That second term is responsible for **aliasing**: $X\left(\frac{\omega}{2}\right)$ overlaps $X\left(\frac{\omega}{2} + \pi\right)$ in $V(\omega)$. Their sum happens to be constant, so this special input x gives $v = (\downarrow 2)x = \frac{1}{2}\delta$. All its even-numbered samples are zero, except $x(0) = \frac{1}{2}$.

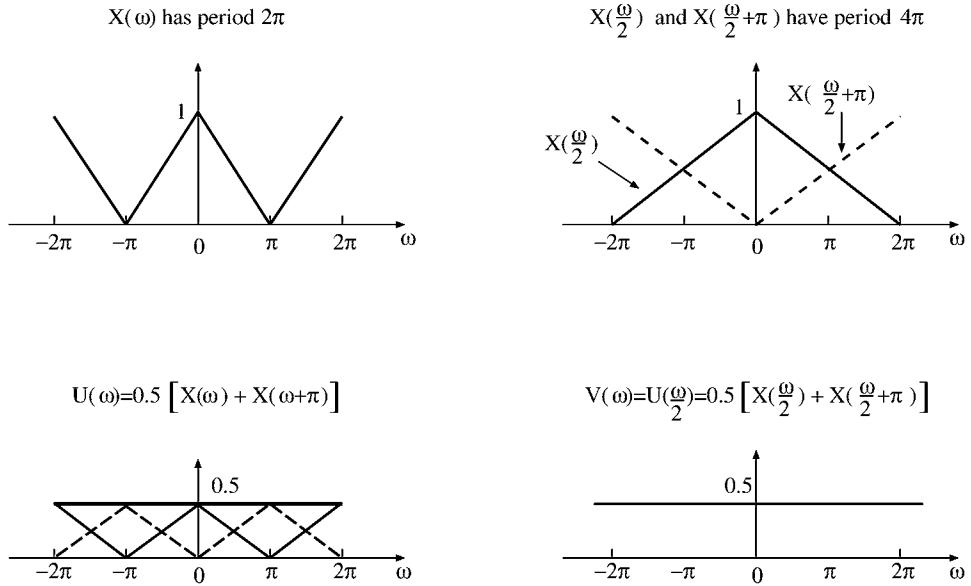


Figure 3.1: Downsampling stretches $X(\omega)$ to $X\left(\frac{\omega}{2}\right)$ and it also creates the alias $X\left(\frac{\omega}{2} + \pi\right)$. Because the alias overlaps, we cannot recover $X(\omega)$ from $U(\omega)$ or from the downsampled $V(\omega)$.

In Figure 3.2, downsampling does not lead to aliasing. The input $X(\omega)$ is properly band-limited. The stretched transform $X\left(\frac{\omega}{2}\right)$ does not overlap the aliasing term $X\left(\frac{\omega}{2} + \pi\right)$. The outputs $V(\omega)$ and $U(\omega)$ from downsampling and upsampling still contain this aliasing part! But a non-overlapping alias can be filtered away.

It is striking that the adjective “halfband” is used in both of these extreme cases, but in *different ways*:

x in Figure 3.1 is a *halfband impulse response* ($x(2n) = 0$ for $n \neq 0$).

x in Figure 3.2 is a *halfband signal* ($X(\omega) = 0$ for $\frac{\pi}{2} \leq |\omega| < \pi$).

Figure 3.3 is somewhere between Figure 3.1 and Figure 3.2. *Aliasing is expected.* The original x cannot be recovered from $(\downarrow 2)x$. But we don’t expect the unusual result in Figure 3.1, where downsampling produced a pure impulse.

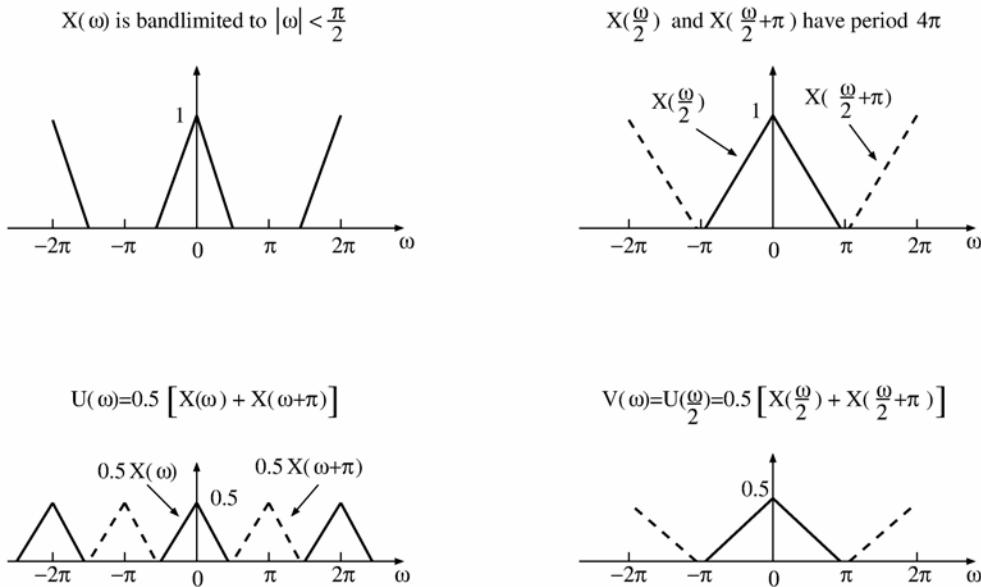


Figure 3.2: Downsampling this bandlimited signal stretches its transform to $[-\pi, \pi]$, no aliasing.

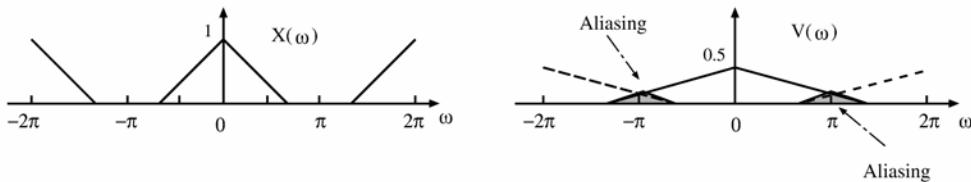


Figure 3.3: Typical aliasing from $(\downarrow 2)$. The stretched $X(\frac{\omega}{2})$ overlaps $X(\frac{\omega}{2} + \pi)$.

Upsampling in the Frequency Domain

In the time domain, upsampling needs separate formulas for even $n = 2k$ and odd $n = 2k + 1$:

$$\mathbf{u} = (\uparrow 2)\mathbf{v} \text{ means } \begin{cases} \mathbf{u}(2k) = \mathbf{v}(k) \\ \mathbf{u}(2k + 1) = 0. \end{cases} \quad (3.17)$$

In the frequency domain this becomes simple. Where downsampling $\mathbf{v}(k) = \mathbf{x}(2k)$ goes from one term in the time domain to two terms in frequency, upsampling goes from two terms to one. The zeros in $\mathbf{u}(2k + 1) = 0$ contribute nothing to $U(\omega)$. We now establish the upsampling formula in the ω domain:

Theorem 3.2 *The transform of $\mathbf{u} = (\uparrow 2)\mathbf{v}$ is*

$$U(\omega) = V(2\omega). \quad (3.18)$$

Proof. Only the even indices $n = 2k$ enter because $\mathbf{u}(2k+1) = 0$:

$$\begin{aligned} U(\omega) &= \sum \mathbf{u}(n)e^{-in\omega} = \sum \mathbf{u}(2k)e^{-i2k\omega} \\ &= \sum \mathbf{v}(k)e^{-i2k\omega} = V(2\omega). \end{aligned} \quad (3.19)$$

Where $V(\omega)$ has period 2π , this new function $V(2\omega)$ has period π . The original graph is compressed into $|\omega| \leq \frac{\pi}{2}$. An image of that compressed graph appears next to it. Upsampling creates an imaging effect! Figure 3.4 shows how two compressed copies of the graph of $V(\omega)$ appear in the spectrum of $U(\omega)$.

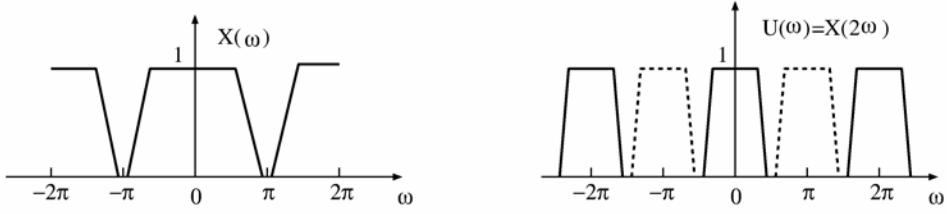


Figure 3.4: The upsampled spectrum $U(\omega) = V(2\omega)$ has compressed images of $V(\omega)$.

This imaging is the *opposite* of aliasing. In aliasing, two input frequencies ω and $\omega + \pi$ give the same output. In imaging, one input frequency ω is responsible for two outputs—at frequency $\frac{\omega}{2}$ and also at $\frac{\omega}{2} + \pi$. Upsampling produces imaging where downsampling produces aliasing.

Upsampling after Downsampling

The analysis bank includes downsampling. Then the synthesis bank includes upsampling. It is very common to see those two operations together:

$$\mathbf{v} = (\downarrow 2)\mathbf{x} \quad \text{and then} \quad \mathbf{u} = (\uparrow 2)\mathbf{v} \quad \text{give} \quad \mathbf{u} = (\uparrow 2)(\downarrow 2)\mathbf{x}.$$

Theorem 3.3 *The transform of $\mathbf{u} = (\uparrow 2)(\downarrow 2)\mathbf{x}$ is*

$$U(\omega) = \frac{1}{2} [X(\omega) + X(\omega + \pi)]. \quad (3.20)$$

This was equation (3.14). The aliasing $X(\omega + \pi)$ comes from $(-1)^n \mathbf{x}(n)$. When n is odd, this cancels $\mathbf{x}(n)$. The average of $\mathbf{x}(n)$ and $(-1)^n \mathbf{x}(n)$ is

$$\mathbf{u}(n) = \begin{cases} \mathbf{x}(n), & n \text{ even} \\ 0, & n \text{ odd.} \end{cases} \quad (3.21)$$

In the frequency domain the proof has two steps:

$$\begin{aligned} \text{down: } V(\omega) &= \frac{1}{2} \left[X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega}{2} + \pi\right) \right] \\ \text{then up: } U(\omega) &= V(2\omega) = \frac{1}{2} [X(\omega) + X(\omega + \pi)]. \end{aligned}$$

Conclusion: $(\uparrow 2)(\downarrow 2)$ produces *aliasing and also imaging*. Figure 3.5 shows them both; the image of the alias overlaps the original. We can't determine the original $X(\omega)$ from $U(\omega)$.

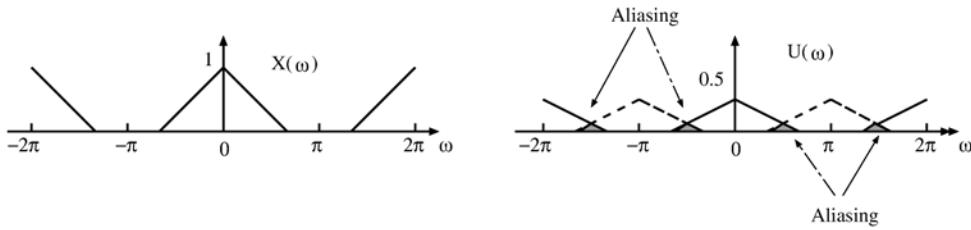
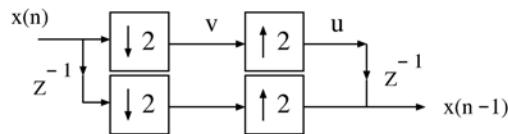


Figure 3.5: The spectrum is stretched by downsampling and compressed by upsampling. The alias from downsampling generally overlaps the image from upsampling.

Filter Bank without Filters

The idea of a filter bank without filters sounds absurd. But this is a very useful example. The filters come from the identity matrix and don't do anything (except possibly a delay). All the activity comes from downsampling and upsampling. It is good to see how the original signal \mathbf{x} is reconstructed.

Here is the block form of the simple filter bank, with no filters:



In words, the upper channel downsamples and upsamples. It produces the vector \mathbf{u} described above, with all odd components of \mathbf{x} replaced by zeros. Then this \mathbf{u} is delayed before output.

The lower channel delays \mathbf{x} at the start. Then downsampling followed by upsampling removes the *even-numbered* components $\mathbf{x}(2k)$. The combined output is just a delay of the input:

$$\begin{aligned} \text{delay of } (\uparrow 2)(\downarrow 2)\mathbf{x} + (\uparrow 2)(\downarrow 2)(\text{delay of } \mathbf{x}) &= \begin{bmatrix} \cdot \\ 0 \\ \mathbf{x}(0) \\ 0 \\ \mathbf{x}(2) \\ \cdot \end{bmatrix} + \begin{bmatrix} \cdot \\ \mathbf{x}(-1) \\ 0 \\ \mathbf{x}(1) \\ 0 \\ \cdot \end{bmatrix} = \text{delay of } \mathbf{x}. \end{aligned}$$

A delay multiplies the transform by $e^{-i\omega}$. The delay in a filter bank is indicated by z^{-1} . Instead of the Fourier transform we more often use the z -transform. A delay multiplies by z^{-1} , and a two-step delay multiplies by z^{-2} . An advance (non-causal!) multiplies by z .

We will next express $(\downarrow 2)$ and $(\uparrow 2)$ in the z -domain. And we take this opportunity to do the same for $(\downarrow M)$ and $(\uparrow M)$. Downsampling and upsampling involve every M th component of the signals in an M -channel filter bank.

Problem Set 3.2

1. Draw $V(\omega)$ and $U(\omega)$ when $X(\omega)$ is a constant function. What vector \mathbf{x} has this transform? What is the result of downsampling that \mathbf{x} ?
2. What are the usual periods of the functions $X(\omega)$, $X\left(\frac{\omega}{2}\right)$, and $V(\omega)$?

3. For downsampling by 3 instead of 2, write down the equations corresponding to (3.11) through (3.16). In equation (3.10), $\mathbf{v} = (\downarrow 3)\mathbf{x}$ means $\mathbf{v}(k) = \mathbf{x}(3k)$.
4. What is the original input \mathbf{x} in Figure 3.1, whose transform is the sawtooth function? To invert transforms use $\mathbf{x}(n) = \frac{1}{2\pi} \int X(\omega) e^{-in\omega} d\omega$.
5. By inverting $X(\omega)$ in Figure 3.2 find the original band-limited signal $\mathbf{x}(n)$. What is $\mathbf{v}(n) = (\downarrow 2)\mathbf{x}(n)$? Compare with the input signal in Figure 3.1.
6. (Review) Verify that the transform of $(\downarrow 2)(\uparrow 2)\mathbf{v}$ is still $V(\omega)$. Then $(\downarrow 2)(\uparrow 2) = \mathbf{I}$.
7. Draw the block form of an M -channel filter bank without filters. Imitating the $M = 2$ case above, the output equals the input after $M - 1$ delays.

3.3 Sampling Operations in the z -Domain

This section includes a conversion from the frequency variable ω to the complex variable z . The connection between them is always $e^{i\omega} = e^{j\omega} = z$. Real frequencies ω correspond to points on the unit circle $|z| = 1$. The transform is extended beyond this special circle of complex numbers, to include all z in the complex plane.

In the process we overcome one inconvenience. The frequencies ω and $\omega + 2\pi$ and $\omega + 4\pi$ are impossible to distinguish. In the z -domain we don't attempt to distinguish them. The complex numbers $z = e^{i\omega}$ and $z = e^{i(\omega+2\pi)}$ and $z = e^{i(\omega+4\pi)}$ are absolutely identical. This remains just as true when the "frequency" ω is not real. It depends only on the fact that $e^{2\pi i} = 1$.

We repeat the definition of the z -transform of \mathbf{x} :

$$\text{The signal } \mathbf{x}(n) \text{ is transformed to } X(z) = \sum_n \mathbf{x}(n) z^{-n}.$$

What we earlier called $X(\omega)$ is now $X(e^{j\omega})$. You will know immediately from the letter z that $X(z)$ refers to the z -transform rather than the Fourier transform.

At present, these infinite series are "formal power series". We are not worrying about their convergence. In reality they probably converge for some z and diverge for other z . When positive and negative powers are both included, the region of convergence is essentially an *annulus*. This is the region $r_{\text{inner}} < |z| < r_{\text{outer}}$ between two circles. Convergence may or may not occur on the circles $|z| = r_{\text{inner}}$ and $|z| = r_{\text{outer}}$. There is also the extreme but quite likely possibility that $r_{\text{inner}} = 0$ or $r_{\text{outer}} = \infty$ or $r_{\text{inner}} = r_{\text{outer}}$.

What is the effect on our down and up formulas? The vectors are $\mathbf{v} = (\downarrow 2)\mathbf{x}$ and $\mathbf{u} = (\uparrow 2)\mathbf{v}$. In frequency, their transforms are

$$V(\omega) = \frac{1}{2} \left[X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega}{2} + \pi\right) \right] \quad \text{and} \quad U(\omega) = V(2\omega).$$

We now derive and explain these rules for converting to the z -transform:

1. Halved frequency $\frac{\omega}{2}$ leads in the z -domain to $z^{1/2}$.
2. Doubled frequency 2ω leads in the z -domain to z^2 .
3. Shifted frequency by π leads in the z -domain to $-z$.

These rules follow directly from $z = e^{i\omega}$. Immediately $e^{i\omega/2}$ is $z^{1/2}$ and $e^{i2\omega}$ is z^2 and $e^{i(\omega+\pi)}$ is $-z$. The rules produce the correct down and up formulas in the z -domain:

Theorem 3.4 *The z -transforms of $\mathbf{v} = (\downarrow 2)\mathbf{x}$ and $\mathbf{u} = (\uparrow 2)\mathbf{v}$ are*

$$V(z) = \frac{1}{2} [X(z^{1/2}) + X(-z^{1/2})] \quad \text{and} \quad U(z) = V(z^2). \quad (3.22)$$

The transform of $\mathbf{u} = (\uparrow 2)(\downarrow 2)\mathbf{x}$ is

$$U(z) = \frac{1}{2} (X(z) + X(-z)). \quad (3.23)$$

The extension from two channels to M channels is straightforward and important. In the time domain, $\mathbf{v} = (\downarrow M)\mathbf{x}$ and $\mathbf{u} = (\uparrow M)\mathbf{v}$ have components

$$\mathbf{v}(n) = \mathbf{x}(Mn) \quad \text{and} \quad \mathbf{u}(k) = \begin{cases} \mathbf{v}\left(\frac{k}{M}\right), & M \text{ divides } k \\ 0, & \text{otherwise.} \end{cases}$$

Then $(\uparrow M)(\downarrow M)\mathbf{x}$ leaves every M th component unchanged (the components for which M divides k). The other components become zeros.

In the frequency domain there are $M - 1$ aliases in downsampling and $M - 1$ images from upsampling (it is very instructive to graph $V(\omega)$ and $U(\omega)$):

$$\begin{aligned} \text{down: } V(\omega) &= \frac{1}{M} \left[X\left(\frac{\omega}{M}\right) + X\left(\frac{\omega+2\pi}{M}\right) + \cdots + X\left(\frac{\omega+(M-1)2\pi}{M}\right) \right] \\ \text{up: } U(\omega) &= V(M\omega). \end{aligned} \quad (3.24)$$

Eliminating V leaves $U(\omega) = \frac{1}{M} [X(\omega) + X(\omega + 2\pi) + \cdots]$ for the upsampled $\mathbf{u} = (\uparrow M)(\downarrow M)\mathbf{x}$. Those aliases overlap the original (not if it is properly bandlimited). To transfer into the z -domain, extend the three rules from $M = 2$ to any M :

1. Dividing ω by M leads to $z^{1/M}$.
2. Multiplying ω by M leads to z^M .
3. Shifting ω by $2\pi/M$ leads to $ze^{2\pi i/M}$.

From this correspondence we can express $(\downarrow M)$ and $(\uparrow M)$ in the z -domain:

Theorem 3.5 *The z -transforms of $\mathbf{v} = (\downarrow M)\mathbf{x}$ and $\mathbf{u} = (\uparrow M)\mathbf{v}$ are*

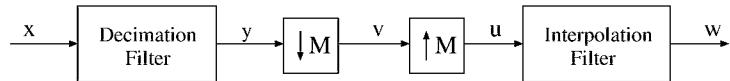
$$V(z) = \frac{1}{M} \sum_{k=0}^{M-1} X(z^{1/M} e^{2\pi i k/M}) \quad \text{and} \quad U(z) = V(z^M). \quad (3.25)$$

The transform of $\mathbf{u} = (\uparrow M)(\downarrow M)\mathbf{x}$, down and up, is by eliminating V :

$$U(z) = \frac{1}{M} [X(z) + X(ze^{2\pi i/M}) + \cdots + X(ze^{2\pi i(M-1)/M})]. \quad (3.26)$$

The *decimator* (downsampling operator) by itself creates aliases. The *expander* (upsampling operator) by itself creates images. To prevent the aliases and to remove the images, we filter the signal. A filter comes before the decimator, to band-limit the input. The aliasing terms become

zero. A filter comes *after* the expander, to wipe out the images. This filter removes (or nearly removes) the frequencies beyond $|\omega| = \frac{\pi}{M}$. The block forms are clear:



The decimation filter suppresses aliasing. The interpolation filter suppresses imaging.

It is easiest to see the decimation circuit in the frequency domain. The band-limiting filter cuts off $X(\omega)$ outside $[-\frac{\pi}{M}, \frac{\pi}{M}]$, to give $Y(\omega)$. Downsampling stretches that band to the full interval $[-\pi, \pi]$. But no aliases appear, as shown in Figure 3.6.

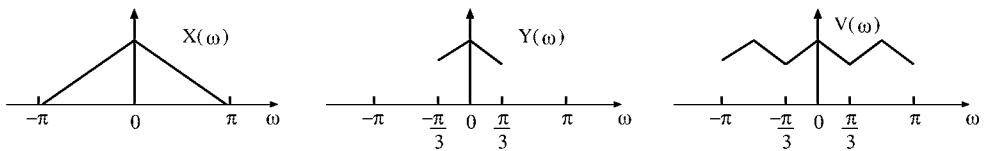
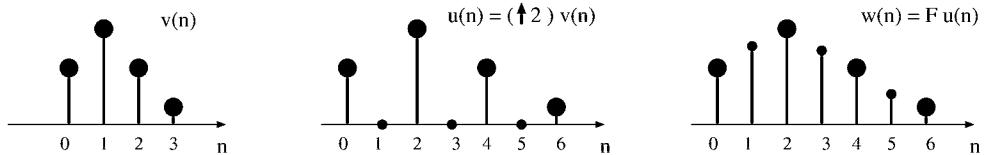


Figure 3.6: Spectra of $X(\omega)$, $Y(\omega)$, and $V(\omega)$ with $M = 3$.

From $V(\omega)$ we can recover $Y(\omega)$. Unless the original signal was band-limited, we cannot recover $X(\omega)$. If it was, we can. The interpolation circuit would do the recovery!

It is important to see the interpolation circuit in the time domain. The upsampling step inserts zeros into the signal. The interpolation filter F changes those zero values to smoother values $w(n)$. Remember that a lowpass filter reduces oscillations to give a smoother output. Here $M = 2$ and the interpolation circuit produces w from v :



Fractional Sampling Rates

To understand fractional decimation, study the product $(\uparrow 2)(\downarrow 3)$ and also $(\downarrow 3)(\uparrow 2)$. What are the outputs? In some way these should produce an output at a fractional rate, which is $\frac{2}{3}$ of the original rate. The transform $X(\omega)$ should be stretched by a factor $\frac{3}{2}$. Let us see.

The direct approach is in the time domain. When up follows down, we reach $x(0), 0, x(3), 0$:

$$\left[\begin{array}{c} x(0) \\ x(1) \\ x(2) \\ x(3) \\ \vdots \end{array} \right] \xrightarrow{\downarrow 3} \left[\begin{array}{c} x(0) \\ x(3) \\ \vdots \end{array} \right] \xrightarrow{\uparrow 2} \left[\begin{array}{c} x(0) \\ 0 \\ x(3) \\ 0 \\ \vdots \end{array} \right]$$

When down follows up, the remarkable thing is that the result is the same:

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ \vdots \\ x(3) \end{bmatrix} \xrightarrow{\uparrow 2} \begin{bmatrix} x(0) \\ 0 \\ x(1) \\ 0 \\ x(2) \\ 0 \\ x(3) \end{bmatrix} \xrightarrow{\downarrow 3} \begin{bmatrix} x(0) \\ 0 \\ x(3) \\ 0 \\ \vdots \\ x(3) \end{bmatrix}$$

Thus $(\uparrow 2)$ commutes with $(\downarrow 3)$. In either order, every third component appears in every second position. Remember that $(\uparrow 2)$ does *not* commute with $(\downarrow 2)$.

A natural question arises. **When does $(\uparrow L)$ commute with $(\downarrow M)$?** The output will come at $\frac{L}{M}$ times the original rate. It is best to reduce this fraction to lowest terms. The numbers L and M should be *relatively prime*, meaning that they have no common factors. The fraction $\frac{2L}{2M}$ is equal to $\frac{L}{M}$, but it is a mistake to use $(\uparrow 2L)$ and $(\downarrow 2M)$. Those operators do not commute and they are inefficient. The natural choice is the good choice.

Theorem 3.6 *The operators $(\downarrow L)$ and $(\uparrow M)$ commute if and only if L and M are relatively prime. Their greatest common divisor is $(L, M) = 1$. In that case $(\uparrow L)(\downarrow M)\mathbf{x} = (\downarrow M)(\uparrow L)\mathbf{x}$ has components*

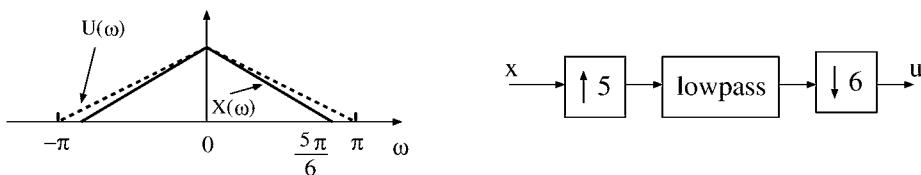
$$\mathbf{u}(n) = \begin{cases} \mathbf{x}(Mk) & \text{if } \frac{n}{L} = k \text{ is an integer} \\ 0 & \text{if } \frac{n}{L} \text{ is not an integer.} \end{cases} \quad (3.27)$$

This formula is always correct for the order $(\uparrow L)(\downarrow M)\mathbf{x}$. With $L = 2$ and $M = 3$ it yields $\mathbf{u}(1) = 0$ and $\mathbf{u}(2) = \mathbf{x}(3)$ as in the example above. With $L = 2$ and $M = 2$ it yields $\mathbf{u}(1) = 0$ and $\mathbf{u}(2) = \mathbf{x}(2)$. This is the familiar output from $(\uparrow 2)(\downarrow 2)\mathbf{x}$.

The formula (3.27) can be wrong if down follows up. In that order, the test is whether $\frac{Mn}{L}$ is an integer. For $M = L = 2$ this is true for every n . The output from $(\downarrow 2)(\uparrow 2)$ is the same as the input, and $(\downarrow 2)(\uparrow 2) = \mathbf{I}$. But for $M = 3$ and $L = 2$, the fraction $\frac{3n}{2}$ is only an integer when n is even. In this $\frac{2}{3}$ case, up-down agrees with down-up.

When M and L are relatively prime, the output in either order has $\mathbf{u}(n) = 0$ unless L divides n . If $\frac{n}{L}$ is not an integer, then $\frac{Mn}{L}$ is not an integer.

Why do we alter the rate by a fraction? The rate may need to change from 60 Hertz to 50 Hertz. The choices $L = 5$ and $M = 6$ will do it. (Not 50 and 60; they are not relatively prime.) We will get five samples instead of six samples, and 50 samples per second instead of 60 per second. There is no aliasing when the input signal is band-limited to $|\omega| < \frac{5\pi}{6}$. Then, stretching by $\frac{6}{5}$ will create no problems:



Problem Set 3.3

1. Find $U(\omega)$ from $X(\omega)$ if $\mathbf{u} = (\uparrow L)(\downarrow M)\mathbf{x}$. Use (3.25) for $(\downarrow M)$. Similarly find $V(\omega)$ if $\mathbf{v} = (\downarrow M)(\uparrow L)\mathbf{x}$. Show that $U(\omega) \equiv V(\omega)$ if (and only if) L and M are relatively prime.
2. Verify formula (3.27) directly. What happens to the odd-numbered components of \mathbf{x} when we compute $(\uparrow 2)(\downarrow 2)\mathbf{x}$? What happens to the odd part of $X(z)$ in equation (3.24)?
3. If a band-limited signal \mathbf{v} goes through an expander, how is $(\uparrow 2)\mathbf{v}$ related to \mathbf{v} ?
4. For a band-limited signal \mathbf{x} , draw in the frequency domain the steps of decimation and recovery by interpolation. Graph $X(\omega)$, $Y(\omega)$, $V(\omega)$, $U(\omega)$, and $W(\omega)$.
5. Draw $H(\omega)$ for an ideal decimator that prevents aliasing by $(\downarrow 6)$. Draw an ideal $F(\omega)$ that removes images created by $(\uparrow 5)$.
6. What lowpass filter placed between $(\uparrow 5)$ and $(\downarrow 6)$ avoids images and also aliasing?
7. In smoothing $\mathbf{u}(n)$ to get the final output $\mathbf{w} = \mathbf{F}\mathbf{u}$, which filters \mathbf{F} will *interpolate* and not change the even samples: $w(2k) = u(2k)$?

3.4 Filters Interchanged with Samplers

Our notation for the filter coefficients is $\mathbf{h}(n)$. When the input \mathbf{x} is the impulse δ , the responses are $\mathbf{h}(0), \dots, \mathbf{h}(N)$. The vector \mathbf{h} is the impulse response (in the time domain). The response in the z -domain (*we are now using this form systematically and often*) is $H(z) = \sum \mathbf{h}(n)z^{-n}$.

As an operator, the filter is a combination of delays (and possibly advances). *Note especially that a delay corresponds to z^{-1}* . The filter is linear and time-invariant. Expressed as an infinite matrix, \mathbf{H} has the number $\mathbf{h}(n)$ along its n th diagonal (counting down from the main diagonal). The i, j entry is $\mathbf{h}(i - j)$.

First Noble Identity

In the standard form, downsampling follows filtering. We apply \mathbf{H} and then $(\downarrow 2)$ or $(\downarrow M)$. Taken literally, this would be highly inefficient. We are apparently computing all components of \mathbf{Hx} , and then throwing away many of them. No sensible engineer would do that. The polyphase form of $H(z)$ will tell us how to reorganize the calculation and do the sampling first.

We cannot just reverse the order of $(\downarrow M)$ and \mathbf{H} (of course!). Here we note a special and important case, when the sampling can be done first. *The filter has $M - 1$ zero coefficients between each pair of nonzeros*. The nonzero coefficients $\mathbf{h}(0), \mathbf{h}(M), \mathbf{h}(2M), \dots$ will be renamed $\mathbf{g}(0), \mathbf{g}(1), \mathbf{g}(2), \dots$. Then $H(z)$ is the same as $G(z^M)$. For filters of this special type we can move the decimator outside the filter:

$$\mathbf{x} \xrightarrow{\downarrow M} \mathbf{G}(z) \xrightarrow{\quad} \mathbf{v} \quad = \quad \mathbf{x} \xrightarrow{\quad} \mathbf{G}(z^M) \xrightarrow{\downarrow M} \mathbf{v}$$

The new order starting with $(\downarrow M)$ cuts the sampling rate immediately. The old order applies \mathbf{H} to the full vector \mathbf{x} . Here is an example with $M = 2$ and $\mathbf{h} = (1, 0, 2)$:

$$\begin{array}{lll} (\downarrow 2)\mathbf{x} & \mathbf{G}(\downarrow 2)\mathbf{x} & \mathbf{Hx} \\ \left[\begin{array}{c} \mathbf{x}(0) \\ \mathbf{x}(2) \\ \vdots \end{array} \right] & \left[\begin{array}{c} \mathbf{x}(0) + 2\mathbf{x}(-2) \\ \mathbf{x}(2) + 2\mathbf{x}(0) \\ \vdots \end{array} \right] & \text{or} \quad \left[\begin{array}{c} \mathbf{x}(0) + 2\mathbf{x}(-2) \\ \mathbf{x}(1) + 2\mathbf{x}(-1) \\ \mathbf{x}(2) + 2\mathbf{x}(0) \\ \mathbf{x}(3) + 2\mathbf{x}(1) \\ \vdots \end{array} \right] \\ & & \left[\begin{array}{c} \mathbf{x}(0) + 2\mathbf{x}(-2) \\ \mathbf{x}(2) + 2\mathbf{x}(0) \\ \vdots \end{array} \right] \end{array} \quad (\downarrow 2)\mathbf{Hx} = \mathbf{G}(\downarrow 2)\mathbf{x}$$

In the z -domain, this example yields the *even powers* in $(1 + 2z^{-2})X(z)$. The odd powers in $X(z)$ don't matter! So the quick way is to pick out the even powers by downsampling first. This "Noble Identity" represents a simple but very useful fact of algebra:

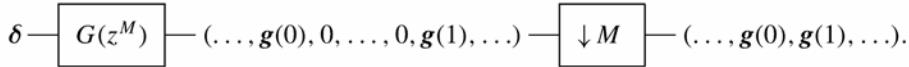
$$\text{First Noble Identity: } G(z)(\downarrow M) = (\downarrow M)G(z^M). \quad (3.28)$$

Proof Downsampling $x(n)$ leaves $x(Mn)$. Then the filter G produces

$$v(n) = \sum g(k)x(M(n - k)). \quad (3.29)$$

The other way (the slow way), $H(z) = G(z^M)$ uses every M th sample of the input. The filter gives $\sum g(k)x(n - Mk)$. Then the sampling operator replaces n by Mn . We reach the same $v(n)$.

You may find it useful to follow an impulse $x = \delta$ through both systems, to see that the impulse response is the same. On the left we get the vector $v = g$ of filter coefficients. On the right we keep all the zeros. Then downsampling ends with the same g :



We used G instead of the basic symbol H to emphasize that *the Noble Identities cannot be directly applied to a typical filter*. The identity applies only when $H(z)$ has the form $G(z^M)$. Only filters that have special impulse responses, with $M - 1$ zeros between the nonzeros, fit this form. The idea of the polyphase matrix is to decompose a typical H into a set of M filters with these special responses. Then the Noble Identities apply to each polyphase component of H , even if they don't apply to the whole filter.

Second Noble Identity

The second identity concerns *upsampling*. It is a transpose of the first identity. The standard form of a filter bank has $(\uparrow M)$ before a synthesis filter $F(z)$. For filters of the special form $F(z) = G(z^M)$, that order can be reversed. But again we must change $G(z^M)$ to $G(z)$:



This time we verify the equivalence in the z -domain. The output $(\uparrow M)Gx$ comes from upsampling $G(z)X(z)$:

$$U(z) = G(z^M)X(z^M).$$

In the other order, the upsampled signal $X(z^M)$ is filtered by the stretched-out $G(z^M)$. Again $U(z) = G(z^M)X(z^M)$, which proves the identity:

$$\text{Second Noble Identity: } (\uparrow M)G(z) = G(z^M)(\uparrow M). \quad (3.30)$$

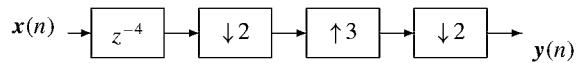
As before, this identity does not apply to most filters. We still need $M - 1$ zeros between the nonzeros. For a simple delay, $G(z^M) = z^{-1}$ is not possible.

The Noble Identities are derived for systems with one input and one output. They also hold for systems with multiple inputs and outputs, as long as the decimation (or interpolation) factors are the same for all channels.

We soon move to the polyphase decomposition of a filter (and a filter bank). The phases have $M - 1$ zeros and the special form involving z^M . In the polyphase form, *the downsampling can come first*. And upsampling comes last.

Problem Set 3.4

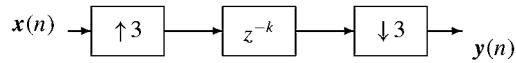
1. Verify the first Noble Identity $G(z)(\downarrow M) = (\downarrow M)G(z^M)$ in the z -domain.
2. Verify the second Noble Identity $(\uparrow M)G(z) = G(z^M)(\uparrow M)$ in the time domain.
3. Simplify the following system:



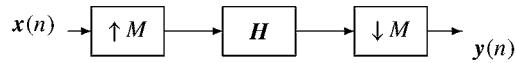
What is $Y(z)$ in terms of $X(z)$? Find $y(n)$ for the following inputs:

$$x(n) = -(n), \quad x(n) = (\dots, 1, 1, 1, 1, \dots), \quad x(n) = (\dots, 1, -1, 1, -1, \dots).$$

4. What is $Y(z)$ in terms of $X(z)$? What is $y(n)$ in terms of $x(n)$?



5. If $H(z) = G(z^M)$ has $M - 1$ zeros between $\mathbf{h}(0), \mathbf{h}(M), \dots$, find $y(n)$ in terms of $x(n)$ and $\mathbf{h}(n)$ for this system:



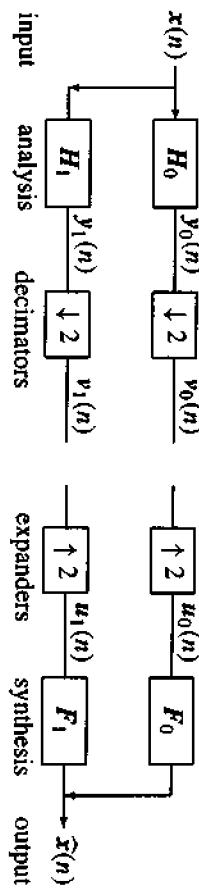
6. What identity would you use to compute $\mathbf{v} = (\downarrow 2)\mathbf{H}\mathbf{x}$ efficiently, if \mathbf{H} has only *odd-numbered* coefficients $\mathbf{h} = (0, \mathbf{h}(1), 0, \mathbf{h}(3))$?

Chapter 4

Filter Banks

4.1 Perfect Reconstruction

A filter bank is a set of filters, linked by sampling operators and sometimes by delays. The down-sampling operators are decimators, the upsampling operators are expanders. In a two-channel filter bank, the analysis filters are normally lowpass and highpass. Those are the filters H_0 and H_1 at the start of the following filter bank:



This structure was introduced in the 1980's. It gradually became clear how to choose H_0 , H_1 , F_0 , F_1 to get perfect reconstruction: $\hat{x}(n) = x(n - l)$. The gap in the figure indicates where the downsampled signals might be coded for storage or transmission. At that point we may compress the signal and destroy information. Perfect reconstruction assumes no compression, so the gap is closed.

To indicate that H_0 is lowpass and H_1 is highpass, we often sketch the frequency responses. Figure 4.1 shows that they are not ideal brick wall filters. The responses overlap. *There is aliasing in each channel.* There is also amplitude distortion and phase distortion (our drawing does not show the phase). The synthesis filters F_0 and F_1 must be specially adapted to the analysis filters H_0 and H_1 , in order to cancel the errors in this analysis bank.

The goal of this section is to discover the conditions for perfect reconstruction. This means that the filter bank is *biorthogonal*. The synthesis bank, from F_0 and F_1 and $\uparrow 2$, is the inverse of the analysis bank. Inverse matrices automatically involve biorthogonality. (The rows of T and the columns of T^{-1} are by definition biorthogonal.) This will extend in Chapter 6 to biorthogonal scaling functions and wavelets.

Perfect reconstruction is a crucial property. If the sampling operators ($\downarrow 2$) and ($\uparrow 2$) were not present, a reconstruction without delay would mean that $F_0 H_0 + F_1 H_1 = I$. A perfect reconstruction with an l -step delay would mean (in the z -domain) that

$$\text{without } (\downarrow 2) \text{ and } (\uparrow 2): \quad F_0(z)H_0(z) + F_1(z)H_1(z) = z^{-l}. \quad (4.1)$$

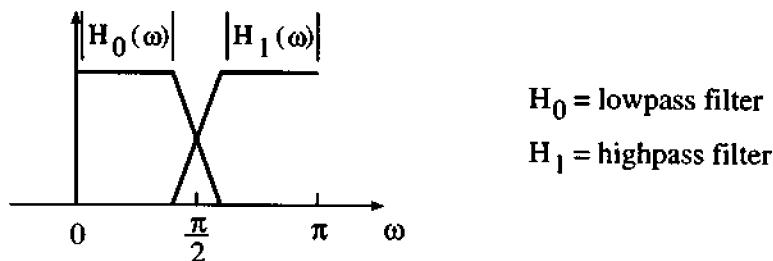


Figure 4.1: Rough sketch of frequency responses. Do not expect $|H_0(\omega)| + |H_1(\omega)| = 1$.

We do expect an overall delay z^{-l} , because each individual filter is causal.

Now take account of the sampling operators, which introduce *aliasing*. We recognize aliasing by the appearance of $-z$ as well as z (and $\omega + \pi$ as well as ω). The combination of $(\downarrow 2)$ followed by $(\uparrow 2)$ zeros out the odd-numbered components. In the z -domain it keeps only the even powers of $H_0(z)X(z)$:

The transform of $(\uparrow 2)(\downarrow 2)H_0x$ is $\frac{1}{2}(H_0(z)X(z) + H_0(-z)X(-z))$.

This is an even function, because the odd components are gone. The aliasing term $H_0(-z)X(-z)$ is multiplied by $F_0(z)$ at the synthesis step. This alias has to cancel the alias $F_1(z)H_1(-z)X(-z)$ from the other channel. So there is an alias cancellation condition in addition to a reconstruction condition:

$$\boxed{\text{Alias cancellation } F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0.} \quad (4.2)$$

Correction The sampling operators also produce a change in equation (4.1). *The right side has an extra factor 2*. You can see this by considering a simple set of filters: $H_0(z) = 1$ and $H_1(z) = z^{-1}$, $F_0(z) = z^{-1}$ and $F_1(z) = 1$. This satisfies (4.2) and cancels aliasing. The left side of equation (4.1) equals $2z^{-1}$ rather than z^{-1} . The overall delay is $l = 1$ for this filter bank, and its perfect reconstruction comes from

$$\boxed{\text{No distortion } F_0(z)H_0(z) + F_1(z)H_1(z) = 2z^{-1}.} \quad (4.3)$$

The next page establishes these two conditions (4.2–4.3) for perfect reconstruction.

One further point. In a genuine filter bank, the highpass filter has $H_1 = 0$ at $z = 1$ (or $\omega = 0$). Equation (4.3) becomes $F_0(1)H_0(1) = 2$. That equation is more natural if we include an extra factor $\sqrt{2}$ in the filter coefficients. For a similar reason the highpass filters can be normalized by an extra $\sqrt{2}$. We take this opportunity to assign single letters $C = \sqrt{2}H_0$ and $D = \sqrt{2}H_1$ to the analysis filters, with no need for subscripts. The sum of lowpass coefficients $c(n) = \sqrt{2}h(n)$ is $\sqrt{2}$.

No Aliasing and No Distortion Conditions (4.2) and (4.3) for perfect reconstruction come directly from following a signal through the filter bank. The original signal is $x(n)$. The lowpass analysis filter is H_0 . In the z -domain this produces $H_0(z)X(z)$. Now downsample and upsample:

First $(\downarrow 2)$ produces $\frac{1}{2}[H_0(z^{\frac{1}{2}})X(z^{\frac{1}{2}}) + H_0(-z^{\frac{1}{2}})X(-z^{\frac{1}{2}})]$

Then $(\uparrow 2)$ produces $\frac{1}{2}[H_0(z)X(z) + H_0(-z)X(-z)]$.

(↑ 2) (↓ 2) H_0x has zeros in its odd-numbered components. Those zeros are produced by averaging $H_0x(n)$ with its alternating alias $(-1)^n H_0x(n)$. In the z -domain this is the average of $H_0(z)X(z)$ with $H_0(-z)X(-z)$. The aliasing term has entered the filter bank.

The final filter multiplies by $F_0(z)$. This yields the output from the lowpass channel. Below it we write the corresponding output from the highpass channel (same formula with subscripts changed to 1):

$$\text{lowpass output} = \frac{1}{2}F_0(z)[H_0(z)X(z) + H_0(-z)X(-z)]$$

$$\text{highpass output} = \frac{1}{2}F_1(z)[H_1(z)X(z) + H_1(-z)X(-z)].$$

Now add. The filter bank combines the channels to get $\widehat{x}(n)$. In the z -domain this is $\widehat{X}(z)$. Half the terms involve $X(z)$ and half involve $X(-z)$:

$$\frac{1}{2}[F_0(z)H_0(z) + F_1(z)H_1(z)]X(z) + \frac{1}{2}[F_0(z)H_0(-z) + F_1(z)H_1(-z)]X(-z).$$

For perfect reconstruction with l time delays, this $\widehat{X}(z)$ must be $z^{-l}X(z)$. So the “distortion term” must be z^{-l} and the “alias term” must be zero:

Theorem 4.1 A 2-channel filter bank gives perfect reconstruction when

$$F_0(z)H_0(z) + F_1(z)H_1(z) = 2z^{-l} \quad (4.4)$$

$$F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0. \quad (4.5)$$

In vector-matrix form these two conditions involve the *modulation matrix* $H_m(z)$:

$$[F_0(z) \quad F_1(z)] \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} = [2z^{-l} \quad 0]. \quad (4.6)$$

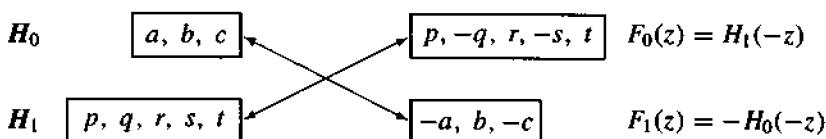
This matrix $H_m(z)$ will play a very important role. It involves the responses $H_k(z)$ and their alias terms $H_k(-z)$. For an M -channel bank the matrix will be $M \times M$. But the real problem is clearly identified by the separate conditions (4.4) and (4.5)—*how to design filters that meet those conditions?*

Alias Cancellation and the Product Filter $P_0 = F_0H_0$

At this point we have four filters H_0, H_1, F_0, F_1 to design. They must satisfy (4.4) and (4.5). It is almost irresistible to determine some of the filters from the others:

For alias cancellation choose $F_0(z) = H_1(-z)$ and $F_1(z) = -H_0(-z)$ (4.7)

Important: This choice automatically satisfies $F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0$. Aliasing is removed; it cancels itself! This relation of F_0 to H_1 and of F_1 to H_0 gives the *alternating signs* pattern of a 2-channel filter bank:



Now comes a definition that allows us to rewrite equation (4.4) for no distortion:

Define the product filter by $P_0(z) = F_0(z)H_0(z)$.

This is a lowpass filter. The highpass product filter is $P_1(z) = F_1(z)H_1(z)$. These products P_0 and P_1 are exactly the terms in (4.4). The crucial point is the relation between $P_0(z)$ and $P_1(z)$, when the synthesis filters are determined by $F_0(z) = H_1(-z)$ and $F_1(z) = -H_0(-z)$. We substitute directly to find that $P_1(z) = -P_0(-z)$:

$$P_1(z) = -H_0(-z)H_1(z) = -H_0(-z)F_0(-z) = -P_0(-z). \quad (4.8)$$

The reconstruction equation $F_0(z)H_0(z) + F_1(z)H_1(z) = 2z^{-l}$ simplifies to

$$F_0(z)H_0(z) - F_0(-z)H_0(-z) = P_0(z) - P_0(-z) = 2z^{-l}. \quad (4.9)$$

The design of a 2-channel PR filter bank is reduced to two steps:

Step 1. Design a lowpass filter P_0 satisfying (4.9).

Step 2. Factor P_0 into F_0H_0 . Then use (4.7) to find F_1 and H_1 .

The length of P_0 determines the sum of the lengths of F_0 and H_0 . There are many ways to design P_0 in Step 1. And there are many ways to factor it in Step 2. Experiments are going on as this book is written, and undoubtedly they are going on as the book is read, to find the best factors F_0 and H_0 of the best product filter P_0 .

Note that (4.9) is a condition on the *odd powers* in $P_0(z) = F_0(z)H_0(z)$. Those odd powers must have coefficient zero, except z^{-l} has coefficient one. The shift to $P(z)$ in equation (4.11) below will remove the even powers except z^0 .

A look forward To help the reader find the specific filters that are coming, we point to an outstanding choice for the product filter:

$$P_0(z) = (1 + z^{-1})^{2p} Q(z). \quad (4.10)$$

The polynomial $Q(z)$ of degree $2p - 2$ is chosen so that (4.9) is satisfied. There are $2p - 1$ odd powers in $P_0(z)$, and $2p - 1$ coefficients to choose in $Q(z)$. *Then $Q(z)$ is unique.* This is the Daubechies construction, with a history that we will outline in Section 5.5. Since the construction starts with the special factor $(1 + z^{-1})^{2p}$, these filters are called *binomial* or *maxflat*. The binomial factor gives a maximum number of zeros at $z = -1$, which means that the frequency response is maximally flat at $\omega = \pi$. The binomial by itself, without $Q(z)$, represents a “spline filter”. $Q(z)$ is needed to give perfect reconstruction.

Splitting P_0 into F_0H_0 can give linear phase filters (symmetry in F_0 and H_0 separately). It can give orthogonal filters (symmetry between F_0 and H_0). *It cannot give both*, except in the Haar case $p = 1$. Section 5.4 discusses the factorization, and Chapter 11 reports some comparisons for image processing.

Simplification The equation $P_0(z) - P_0(-z) = 2z^{-l}$ can be made a little more convenient. The left side is an odd function, so l is odd. Normalize $P_0(z)$ by z^l to center it:

The normalized product filter is $P(z) = z^l P_0(z)$.

Then $P(-z) = (-z)^l P_0(-z)$. Since l is odd, this is $-z^l P_0(-z)$. The reconstruction equation $P_0(z) - P_0(-z) = 2z^{-l}$ takes an extremely simple form when we multiply by z^l . The factor z^{-l} disappears and the minus sign becomes plus:

Perfect Reconstruction Condition. $P(z)$ must be a “halfband filter”

$$P(z) + P(-z) = 2. \quad (4.11)$$

This means that *all even powers in $P(z)$ are zero*, except the constant term (which is 1). The odd powers cancel when $P(z)$ combines with $P(-z)$ —so the coefficients of odd powers in $P(z)$ are design variables in 2-channel PR filter banks.

Example 4.1. The maxflat product filter $P_0(z) = (1 + z^{-1})^4 Q(z)$ happens to be

$$P_0(z) = \frac{1}{16}(-1 + 9z^{-2} + 16z^{-3} + 9z^{-4} - z^{-6}).$$

The center term is $z^{-3} = z^{-l}$. The requirement $P_0(z) - P_0(-z) = 2z^{-l}$ is quickly verified, because the even powers in P_0 cancel in the difference. The function $Q(z) = -1 + 4z^{-1} - z^{-2}$ was chosen so that the odd powers z^{-1} and z^{-5} are absent from $P_0(z)$.

A centering operation gives the normalized product filter $P(z)$. Multiply by $z^l = z^3$ to symmetrize the polynomial around the constant term z^0 :

$$P(z) = \frac{1}{16}(-z^3 + 9z + 16 + 9z^{-1} - z^{-3}).$$

This is halfband, because the only even power is z^0 and its coefficient is 1. The perfect reconstruction requirement $P(z) + P(-z) = 2$ is verified. The odd powers in P cancel in the sum.

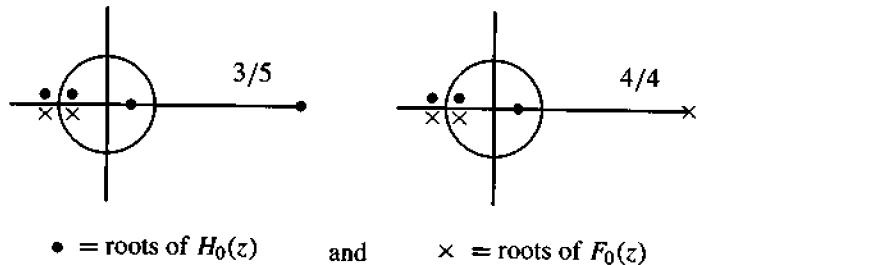
Notice the variety of factorizations into $P_0(z) = F_0(z)H_0(z)$. The polynomial $P_0(z)$ has six roots. The two roots from $Q(z)$ are at $c = 2 - \sqrt{3}$ and $\frac{1}{c} = 2 + \sqrt{3}$. The other four roots from $(1 + z^{-1})^4$ are at $z = -1$. Each factor normally has at least one root at $z = -1$. (But the factorization into $H_0 = 1$ and $F_0 = P_0$ is quite interesting.) Thus F_0 or H_0 (either order is possible!) could be

- | | | |
|-----|--|----------------|
| (a) | 1 | degree $N = 0$ |
| (b) | $(1 + z^{-1})$ | degree $N = 1$ |
| (c) | $(1 + z^{-1})^2$ or $(1 + z^{-1})(c - z^{-1})$ | degree $N = 2$ |
| (d) | $(1 + z^{-1})^3$ or $(1 + z^{-1})^2(c - z^{-1})$ | degree $N = 3$ |

The number $N + 1$ of filter coefficients is one greater than the degree. Thus (c) can produce a 5/3 filter, with $H_0(z) = \frac{1}{8}(-1 + 2z^{-1} + 6z^{-2} + 2z^{-3} - z^{-4})$ and $F_0(z) = \frac{1}{2}(1 + 2z^{-1} + z^{-2})$. The analysis length is given first. This is a possible choice for compression, with symmetric filters of very low complexity. The binomial $(1 + z^{-1})^2$ in the synthesis filter means that its continuous-time scaling function will be a hat function. Reversing F_0 and H_0 gives a 3/5 pair, not as successful in practice.

The choice (b) is also of interest. One factor is $1 + z^{-1}$ so one scaling function is the box function. Experiments indicate better performance when this short lowpass filter is in the analysis bank. Where 5/3 was preferred to 3/5 for odd-length filters, it seems that 2/6 is preferred to 6/2. *Five roots go into $F_0(z)$.*

The other outstanding choices are length 4/4 from the factorizations in (d). We get linear phase from $(1 + z^{-1})^3$ and $(-1 + 3z^{-1} + 3z^{-2} - z^{-3})$. The orthonormal Daubechies filter comes



from $(1+z^{-1})^2(c-z^{-1})$ and $(1+z^{-1})^2(\frac{1}{c}-z^{-1})$. These are not linear phase. (Problem 7 shows that linear phase requires $\frac{1}{c}$ to be a root when c is a root.) For the Daubechies orthogonal choice, the roots $-1, -1, c$ of one factor are the reciprocals of the roots $-1, -1, \frac{1}{c}$ of the other factor. Section 5.4 demonstrates that this balanced splitting (spectral factorization) of a halfband filter produces an orthogonal filter bank and orthogonal wavelets.

Example 4.2. (Haar filter bank) The average-difference analysis filter has

$$H_m(z) = \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1+z^{-1} & 1-z^{-1} \\ 1-z^{-1} & 1+z^{-1} \end{bmatrix}.$$

The synthesis filters are

$$\begin{aligned} F_0(z) &= H_1(-z) = (1+z^{-1})/\sqrt{2} \\ F_1(z) &= -H_0(-z) = -(1-z^{-1})/\sqrt{2}. \end{aligned}$$

The product filters are

$$\begin{aligned} P_0(z) &= F_0(z)H_0(z) = \frac{1}{2}(1+z^{-1})^2 \\ P_1(z) &= F_1(z)H_1(z) = -\frac{1}{2}(1-z^{-1})^2 = -P_0(-z). \end{aligned}$$

Both P_0 and P_1 contain $+z^{-1}$. Perfect reconstruction $P_0(z) - P_0(-z) = 2z^{-1}$ means that $l = 1$. The normalized product filter (symmetric halfband) is

$$P(z) = z^l P_0(z) = \frac{1}{2}z(1+z^{-1})^2 = \frac{1}{2}z + 1 + \frac{1}{2}z^{-1}.$$

Modulation Matrices

The conditions for perfect reconstruction are expressed in (4.6) by

$$[F_0(z) \quad F_1(z)] \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} = [2z^{-l} \quad 0]. \quad (4.12)$$

This displays the *analysis modulation matrix* $H_m(z)$ — which is central to filter bank theory. With no extra effort we can also produce the *synthesis modulation matrix* $F_m(z)$. The two matrices should play matching (and even reversible) roles. This balance between F_m and H_m is achieved by expanding (4.12) into a matrix equation:

$$\begin{bmatrix} F_0(z) & F_1(z) \\ F_0(-z) & F_1(-z) \end{bmatrix} \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} = \begin{bmatrix} 2z^{-l} & 0 \\ 0 & 2(-z)^{-l} \end{bmatrix}. \quad (4.13)$$

The second row of equations follows from the first, when $-z$ replaces z . Note that $F_1(z)$ is in the $(1, 2)$ position, but $H_1(z)$ is in the $(2, 1)$ position. This *transpose convention* between analysis and synthesis will appear again for polyphase matrices. The reader sees why it is necessary.

The reconstruction condition (4.13) applies to $\mathbf{F}_m(z)\mathbf{H}_m(z)$. If we “center” the filter coefficients around the zero position, the right side becomes the identity matrix! This is so desirable and memorable that we do it. It is the same normalization that centered $P_0(z)$ into $P(z)$, and it is especially clear when the filters are linear phase (and l is odd):

Theorem 4.2 *If all filters are symmetric (or antisymmetric) around zero, as in $H(z) = H(z^{-1})$ and $h(k) = h(-k)$, then the condition for perfect reconstruction becomes a statement about inverse matrices:*

$$\mathbf{F}_m(z)\mathbf{H}_m(z) = 2I. \quad (4.14)$$

The H ’s determine the F ’s. The analysis bank is inverted by the synthesis bank. When we express it that way, equation (4.14) becomes almost obvious.

A Brief History of H_1

The reader understands that the filters H_0 and H_1 are still to be chosen. These choices are connected. Historically, designers chose the lowpass filter coefficients $h(0), \dots, h(N)$ and then constructed H_1 from H_0 . Here are two possibilities that produce *equal length filters*. H_1 will be highpass whenever H_0 is lowpass:

Alternating signs : $H_1(z) = H_0(-z)$ comes from $(h(0), -h(1), h(2), -h(3), \dots)$

Alternating flip : $H_1(z) = -z^{-N}H_0(-z^{-1})$ comes from $(h(N), -h(N-1), \dots)$.

For convenience we are assuming real coefficients. The number N is odd in the alternating flip. The perfect reconstruction condition is *still to be imposed*. When that is satisfied, the overall system delay is $l = N$.

Early choice. Croisier-Estaban-Galand (1976) chose alternating signs $H_1(z) = H_0(-z)$. The resulting filter bank was called QMF (Quadrature Mirror Filter). The highpass response $|H_1(e^{j\omega})|$ is a mirror image of the lowpass magnitude $|H_0(e^{j\omega})|$ with respect to the middle frequency $\frac{\pi}{2}$ — the quadrature frequency. Note that IIR filters H_0 and H_1 are allowed (and needed for PR, except for Haar!). This name QMF has since been extended to a larger class of filter banks, allowing M channels.

Better choice. Smith and Barnwell (1984–6) and Mintzer (1985) chose the alternating flip $H_1(z) = -z^{-N}H_0(-z^{-1})$. This leads to orthogonal filter banks, when H_0 is correctly chosen. The Daubechies filters will fit this pattern.

General choice. The product $F_0(z)H_0(z)$ is a halfband filter. This gives biorthogonality, when aliasing is cancelled by the relation of F_0 to H_1 and F_1 to H_0 .

Actually the synthesis bank has little freedom. Alias cancellation requires $F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0$. Croisier-Estaban-Galand wrote each F_k directly in terms of H_k by

$$F_0(z) = H_0(z) \text{ and } F_1(z) = -H_1(z).$$

With alternating signs $H_1(z) = H_0(-z)$ inside the analysis bank, their synthesis construction agrees (as it must) with the anti-aliasing equations

$$F_0(z) = H_1(-z) \text{ and } F_1(z) = -H_0(-z). \quad (4.15)$$

Smith-Barnwell also made the anti-aliasing choice (4.15). With the alternating flip in $H_1(z)$, their synthesis filters (remembering that N is odd) are

$$F_0(z) = H_1(-z) = z^{-N} H_0(z^{-1}) \text{ comes from } (h(N), h(N-1), \dots, h(0)).$$

$$F_1(z) = -H_0(-z) = z^{-N} H_1(z^{-1}) \text{ comes from } (-h(0), h(1), -h(2), \dots, h(N)).$$

Notice! Each F_k in Figure 4.1 has become the ordinary flip of the corresponding H_k . In matrix language the synthesis matrices are the *transposes* of the analysis matrices. A shift by N delays makes them causal. When we flip to get F_0 and then alternate signs to get H_1 , we have the alternating flip from H_0 to H_1 .

The alternating flip automatically gives double-shift orthogonality between highpass and lowpass (to be explained). *Conclusion:* When the design of H_0 leads to perfect reconstruction in the alternating flip filter bank, it also leads to orthogonality.

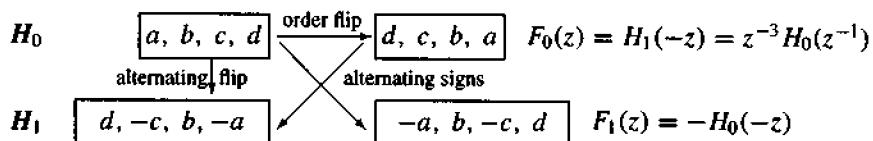


Figure 4.2: Relations between the filters allowing orthogonality when $N = 3$.

With aliasing cancelled, we now look at the PR condition

$$F_0(z)H_0(z) + F_1(z)H_1(z) = 2z^{-l}.$$

The early choice was alternating signs $H_1(z) = H_0(-z)$. With $F_0(z) = H_1(-z)$ and $F_1(z) = -H_0(-z)$, PR requires

$$H_0^2(z) - H_1^2(z) = H_0^2(z) - H_0^2(-z) = 2z^{-l}. \quad (4.16)$$

Therefore $H_0^2(z)$ has exactly one odd power z^{-l} . This is not easy for the square of a polynomial. An FIR filter is restricted to *two coefficients* (not good). Problem 4 asks for the reasoning, which forces the filters to be IIR.

The better choice is alternating flip. Perfect reconstruction is definitely possible. Product filters F_0H_0 and F_1H_1 become $P_0(z) = z^{-N} H_0(z^{-1})H_0(z)$ and $P_1(z) = -z^{-N} H_0(-z^{-1})H_0(-z)$. Multiply by $z^l = z^N$ to center these filters. The normalized product filter is $P(z)$ and the reconstruction condition is (4.11):

$$P(z) + P(-z) = 2 \text{ with } P(z) = H_0(z^{-1})H_0(z). \quad (4.17)$$

This is spectral factorization of a halfband filter! On the unit circle $z = e^{j\omega}$, the product $H_0(e^{-j\omega})H_0(e^{j\omega})$ is a magnitude squared:

$$P(e^{j\omega}) = \sum_{-N}^N p(n)e^{-jn\omega} = \left| \sum_0^N h(n)e^{-jn\omega} \right|^2. \quad (4.18)$$

The halfband coefficients are $p(n) = p(-n)$ for odd n and $p(n) = 0$ for even n (except $p(0) = 1$). We design $P(z)$ and factor to find $H_0(z)$. This symmetric factorization coincides with the Smith-Barnwell alternating flip. It yields orthogonal banks with perfect reconstruction. The flattest $P(z)$ will lead us to the Daubechies wavelets.

A note on biorthogonality (PR) with linear phase

Theorem 4.3 In a biorthogonal linear-phase filter bank with two channels, the filter lengths are all odd or all even. The analysis filters can be

- (a) both symmetric, of odd length
- (b) one symmetric and the other antisymmetric, of even length.

Proof: Odd and even lengths behave differently when we alternate signs:

$$\begin{array}{ll} \text{odd length: } & a \ b \ c \ b \ a \rightarrow a \ -b \ c \ -b \ a \quad (\text{remains symmetric}) \\ \text{even length: } & a \ b \ b \ a \rightarrow a \ -b \ b \ -a \quad (\text{becomes antisymmetric}). \end{array}$$

To cancel aliasing, there is sign alternation in $F_0(z) = H_1(-z)$. There is also alternation in $F_1(z) = -H_0(-z)$. The extra minus sign does not change the symmetry type. The two successful combinations are

$$\begin{array}{ll} H_0 = \text{symm} & F_0 = \text{symm} \\ H_1 = \text{symm} & F_1 = \text{symm} \end{array} \quad \begin{array}{ll} \times & \\ & \end{array} \quad \begin{array}{ll} H_0 = \text{symm} & F_0 = \text{symm} \\ H_1 = \text{anti} & F_1 = \text{anti} \end{array} \quad \begin{array}{ll} \times & \\ & \end{array} \quad \begin{array}{ll} & \\ \text{odd lengths} & \text{even lengths} \end{array}$$

The other possibilities are excluded by the PR condition: $F_0(z) H_0(z)$ has to be a *halfband filter*. It must have an odd number of coefficients, and the center coefficient must be 1. For $F_0(z) H_0(z)$ to have odd length (which means even degree), the factors $F_0(z)$ and $H_0(z)$ must be both odd length or both even length. If one is symmetric and the other antisymmetric, the product $F_0(z) H_0(z)$ will be antisymmetric with zero at the center — not allowed. We conclude that $F_0(z)$ and $H_0(z)$ must match: both odd length or both even length, both symmetric or both antisymmetric.

This leaves the two successful possibilities shown above, and two more: H_0 and F_0 both antisymmetric. But the sum of lowpass coefficients cannot be zero. So antisymmetry of H_0 is ruled out.

Perfect Reconstruction with M Channels In reality a filter bank can have M channels. Although $M = 2$ is standard in many applications, we often see $M > 2$. There are M analysis filters H_0, H_1, \dots, H_{M-1} . The sampling is done at the critical rate by ($\downarrow M$) and ($\uparrow M$). There

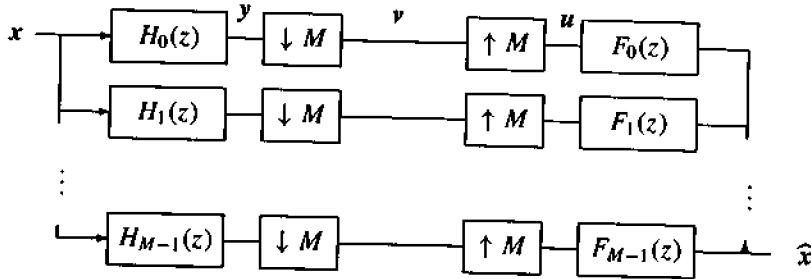


Figure 4.3: Standard form of an M -channel filter bank (maximally decimated).

are M filters F_0, F_1, \dots, F_{M-1} in the synthesis bank. The outputs from all channels are combined into a single output \hat{x} . Our standard picture of this implementation is Figure 4.3.

For this M -band case, the theory of perfect reconstruction was developed over several years. When we follow each channel from H_k through $(\downarrow M)$ and $(\uparrow M)$ and F_k , and add, we find M conditions for perfect reconstruction. These equations involve the $M \times M$ modulation matrix $H_m(z)$ —corresponding exactly to the 2-channel case. The modulations are by multiples of $2\pi/M$ in frequency, and by powers of $W = e^{-2\pi j/M}$ in the z -domain. We put that matrix on record here.

$$\text{Modulation matrix } H_m(z) = \begin{bmatrix} H_0(z) & H_0(zW) & \cdots & H_0(zW^{M-1}) \\ H_1(z) & H_1(zW) & \cdots & H_1(zW^{M-1}) \\ \vdots & \vdots & \ddots & \vdots \\ H_{M-1}(z) & H_{M-1}(zW) & \cdots & H_{M-1}(zW^{M-1}) \end{bmatrix}. \quad (4.19)$$

The last $M - 1$ columns represent the $M - 1$ aliases created by $(\downarrow M)$, just as $H(-z)$ represented the one alias ($W = -1$) created when $M = 2$. The transpose of $H_m(z)$ is the alias component matrix.

Another matrix will play an equally central role—in fact an interchangeable role, because it is very closely linked to $H_m(z)$. This new matrix is the *polyphase matrix* $H_p(z)$. It is developed and explained in the following section, as the natural way to follow the “phases” when a signal is subsampled. We put on record the 2-phase matrix—this is the polyphase matrix when $M = 2$:

$$H_p(z) = \begin{bmatrix} H_{0,\text{even}}(z) & H_{0,\text{odd}}(z) \\ H_{1,\text{even}}(z) & H_{1,\text{odd}}(z) \end{bmatrix}.$$

Still looking ahead, we mention especially the orthogonal case. The polyphase matrix is then *unitary* for $|z| = 1$ (this makes it *paraunitary*). $H_m(z)$ is also paraunitary, after dividing by $\sqrt{2}$ (Section 5.1). The analysis of paraunitary matrices was led by Vaidyanathan. He and others built onto the early theory (and nearly indecipherable exposition) of Belevitch. For $M = 2$, the paraunitary matrix leads back to the Smith-Barnwell construction of an orthogonal PR filter bank.

Several perfect reconstruction filter banks deserve special mention. Simplest is the average-difference pair from Chapter 1. This is a useful example but a poor filter. That is the first in a family of “maxflat filters”, corresponding to the Daubechies wavelets. The others in the family are orthogonal but not linear phase—since those two properties conflict.

Different factorizations of the product $P_0(z)$ lead to linear phase (not orthogonality). Those filters have become favorites for compression.

For $M > 2$, the design of separate filters H_0, H_1, \dots, H_{M-1} can become unwieldy. We look for constructions in which these all come from one prototype filter. A particular class is the *cosine-modulated filter banks* in Chapter 9. A phase change (= modulation) is the key to their construction. Those are efficient in every way.

Problem Set 4.1

- If $(H_m(z))^{-1}$ is also a polynomial, the synthesis bank as well as the analysis bank is FIR. Why must the determinant $H_0(z)H_1(-z) - H_1(z)H_0(-z)$ in the denominator of H_m^{-1} be a monomial cz^{-l} ? This determinant is an odd function, $\det H_m(z) = -\det H_m(-z)$. Then the exponent l must be odd.
- The solution of *both* equations (4.4–5) for F_0 and F_1 involves $(H_m(z))^{-1}$:

$$[F_0(z) \ F_1(z)] = [2z^{-l} \ 0](H_m(z))^{-1} = \frac{2z^{-l}}{\det H_m(z)} [H_1(-z) \ H_0(-z)].$$

If $\det H_m(z) = 2z^{-l}$, as normal, this yields $F_0(z) = H_1(-z)$ and $F_1(z) = -H_0(-z)$.

Extra credit: Verify that the key equation $P(z) + P(-z) = 2$ still holds in the IIR case with the extended definition of the product filter

$$P(z) = \frac{z^l F_0(z) H_0(z)}{\det H_m(z)}.$$

- Find all filters if $H_0(z) = (\frac{1+z^{-1}}{2})^3$ and $P_0(z) = \frac{1}{16}(-1 + 9z^{-2} + 16z^{-3} + 9z^{-4} - z^{-6})$.
- If an FIR filter $H_0(z)$ has three or more coefficients, explain why $H_0^2(z)$ has at least two odd powers. Then $H_0^2(z) - H_0^2(-z) = 2z^{-l}$ is impossible. The “alternating signs” construction is not PR. (This is extended in Theorem 5.3: Symmetry prevents orthogonality except with two coefficients.)
- If H_0 and P_0 are symmetric, why is F_0 symmetric? Why is H_1 linear phase, and when can it be antisymmetric?
- Prove Theorem 4.3 by observing that an order flip in linear-phase filters $H_0(e^{j\omega})$ and $F_0(e^{j\omega})$ gives $H_{0,R}$ and $F_{0,R}$:

$$\begin{cases} H_0(e^{j\omega}) = e^{-j(N_0/2)\omega} H_{0,R}(e^{-j\omega}), & F_0(e^{j\omega}) = e^{-j(N_1/2)\omega} F_{0,R}(e^{-j\omega}) \\ H_0(e^{j\omega}) F_0(e^{j\omega}) + H_0(e^{j(\omega+\pi)}) F_0(e^{j(\omega+\pi)}) = 2e^{-j\ell\omega}. \end{cases}$$

N_0 and N_1 are the degrees of $H_0(z)$ and $F_0(z)$, and ℓ is odd.

- A symmetric filter has $H(z^{-1}) = z^N$ times (—). If $H(c) = 0$ show that also $H(\frac{1}{c}) = 0$.
- In the example with six roots, show a 4/4 linear phase pair—the roots c and $\frac{1}{c}$ go together. Find the polynomials $H(z)$ and $F(z)$.
- The 10th degree halfband polynomial $P_0(z) = (1 + z^{-1})^6 Q(z)$ has four complex roots r , \bar{r} , r^{-1} , \bar{r}^{-1} in the right halfplane (roots of Q). Draw a figure to show the ten roots and how Daubechies 6/6 filters will divide them: r and \bar{r} are separated from r^{-1} and \bar{r}^{-1} .
- For the same 10th degree $P_0(z)$, show how the ten roots can give 6/6 filters with linear phase. One filter has 5 zeros at $z = -1$.
- (Good problem) Find the actual 4th degree $Q(z)$ that makes $P_0(z)$ halfband. If possible compute its roots.

12. Given a PR filter bank H_0, H_1, F_0, F_1 , interchange $H_k(z)$ and $F_k(z)$ (so that the synthesis bank has $H_k(z)$ and analysis bank has $F_k(z)$). Verify that the new system is PR. Define another system $\hat{H}_k(z) = H_k(-z)$ and $\hat{F}_k(z) = F_k(-z)$. Is this new system PR?
13. Let $H_0(z)$ be a symmetric lowpass filter with even length and $H_1(z) = H_0(-z)$. Verify that $H_1(z)$ is an antisymmetric highpass filter. Find the synthesis filters $F_k(z)$ that cancel aliasing. Can this system be PR? (Is $P(z)$ a halfband filter?)

4.2 The Polyphase Matrix

This section establishes a key idea and a valuable notation. The word “*polyphase*” has gained a certain mystique in the theory of multirate filters. Perhaps we can begin by explaining the meaning of the word, and also the purpose of the idea. Then the notation and applications will come naturally.

Meaning of polyphase: When a vector is downsampled by 2, its even-numbered components are kept. Its odd-numbered components are lost. Those are the two phases, *even* and *odd*. It is natural to follow the two phases of the input vector, x_{even} and x_{odd} , as they go through the filter bank. They are acted on by the two phases H_{even} and H_{odd} of the filter.

For downsampling by M there are M phases. The ideas still apply to this “several-phase” or “polyphase” decomposition. Instead of even and odd inputs we will have M vectors (phases of x). Instead of even and odd filters we will have M filters (phases of H). The vector of filter coefficients $h(n)$ is separated into phases, exactly as $x(n)$ is separated. Then we watch those phases during downsampling.

The word “phase” is applied because the even filter with coefficients $h(0), h(2)$ has a different delay (phase shift) from the odd phase with coefficients $h(1), h(3)$.

Purpose of polyphase: The operation $(\downarrow 2)Hx$, taken literally, is not efficient. We are computing all components of Hx and then destroying half of them. If we don’t compute them, the system is still working at a fast rate (high bandwidth). The output is at half rate, because of downsampling. Each output component needs N additions and $N + 1$ multiplications, to apply all the coefficients $h(0), \dots, h(N)$.

The polyphase implementation works on the different phases separately. The input vector is separated into x_{even} and x_{odd} . The operator $(\downarrow 2)$ comes *before the filter!* It changes one input at a high rate to two (or M) inputs at a lower rate. Then the separate phases of the filters act simultaneously (*in parallel*) on separate phases of the input.

The notation has to keep track of each phase. Often we find that “even multiplies even” and “odd multiplies odd”. The *Noble Identities* justify an interchange of filtering and sampling. For the whole filter this interchange is forbidden, but it is allowed for each phase.

Polyphase in the time domain (block Toeplitz matrix): We can display the infinite matrix for a 2-channel analysis bank. Recall that $c(n) = \sqrt{2} h_0(n)$ and $d(n) = \sqrt{2} h_1(n)$. The two filters $\sqrt{2}H_0 = C$ and $\sqrt{2}H_1 = D$ are downsampled by $(\downarrow 2)$. This removes the odd-numbered rows. Then we *interleave the rows* of $L = (\downarrow 2)C$ and $B = (\downarrow 2)D$ to see the analysis bank as a

block Toeplitz matrix:

$$\text{Block Toeplitz } H_b = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ c(3) & c(2) & c(1) & c(0) \\ d(3) & d(2) & d(1) & d(0) \\ c(3) & c(2) & c(1) & c(0) \\ d(3) & d(2) & d(1) & d(0) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

This takes the input in blocks (two samples at a time). It gives the output in blocks. It is time-invariant in blocks! By block z -transform, multiplication by the infinite matrix H_b (which is block convolution) becomes multiplication by the *polyphase matrix*:

$$\text{Polyphase matrix } H_p(z) = \begin{bmatrix} c(0) & c(1) \\ d(0) & d(1) \end{bmatrix} + z^{-1} \begin{bmatrix} c(2) & c(3) \\ d(2) & d(3) \end{bmatrix} \quad (4.20)$$

The polyphase matrix is nothing but the *z -transform of a block of filters*. There are 2^2 or M^2 filters, from M phases of M original filters. Here those filters have four coefficients and their phases have two coefficients.

Notice especially how the block matrix H_b relates to the two separate downsampled filters $(\downarrow 2)\mathbf{C}$ and $(\downarrow 2)\mathbf{D}$:

The efficient form downsamples the input *first* (to make blocks for H_b)

The inefficient form downsamples *last* (after the filters \mathbf{C} and \mathbf{D})

The Noble Identities prove the equivalence. It is just a removal of useless odd-numbered rows and an interleaving of the remaining rows. Next we discuss the algebra and the implementation.

Key identity in the z -domain: The even part of $X(z)$ is $\frac{1}{2}(X(z) + X(-z))$. The odd part is $\frac{1}{2}(X(z) - X(-z))$. The first has even powers $1, z^2, z^4$; the second has z, z^3, z^5 . The original X is the sum of even plus odd (obviously). The same splitting holds for $C(z)$, and furthermore for $C(z)X(z)$. The key is to find the even part of $C(z)X(z)$. It is the even coefficients of Cx that survive downsampling and appear in $(\downarrow 2)Cx$. In most of this section the lowpass filter is denoted by \mathbf{C} , to avoid the subscripts on H .

A simple and important identity shows how the even part of $C(z)X(z)$ comes from even times even plus odd times odd:

$$\begin{aligned} \frac{1}{2}[C(z)X(z) + C(-z)X(-z)] &= \frac{1}{4}[C(z) + C(-z)][X(z) + X(-z)] \\ &\quad + \frac{1}{4}[C(z) - C(-z)][X(z) - X(-z)]. \end{aligned} \quad (4.21)$$

In multiplying numbers, odd times odd is odd. But we are *adding exponents*, as in $(z^3)(z^5) = z^8$. So it is really odd *plus* odd, and even *plus* even, that yield the even part of the z -transform. This is the part that downsampling picks out, when Cx is decimated.

The importance of the key identity is this. The left side involves *all* coefficients of $C(z)$ and $X(z)$. Each product on the right involves only *half* the coefficients. The multiplication in the z -domain, which is $(\downarrow 2)Cx$ in the time domain, becomes computationally efficient. We don't

want all of $C(z)X(z)$, only the even half. The right side shows how to do half the work. Better still, it shows how even-even and odd-odd can be executed in parallel at half the rate.

Downsampling an even function effectively replaces z by $z^{1/2}$. It “closes the gaps” in $1, z^2, z^4$ by changing to $1, z, z^2$. For an odd function we will need a delay or an advance. We cannot change z and z^3 to $z^{1/2}$ and $z^{3/2}$. You will see how the coefficients of z^{-1}, z^{-3}, z^{-5} in $C(z)$ become coefficients of $1, z^{-1}, z^{-2}$ in the odd phase $C_{\text{odd}}(z)$. There is a delay for the odd phase and a “delay chain” when there are multiple phases.

This chapter works out the polyphase notation. We concentrate most on $M = 2$; the phases are even and odd. Then the polyphase forms of the analysis and synthesis banks lead quickly to a main goal of the theory. We find the perfect reconstruction condition on the polyphase matrices, when the filters are centered:

$$\mathbf{F}_p(z)\mathbf{H}_p(z) = \mathbf{I}.$$

This tells us, clearly and directly, what is required:

1. At a minimum, $\mathbf{H}_p(z)$ must be *invertible*. (biorthogonality)
2. Better than that, its inverse $\mathbf{F}_p(z)$ should be a *polynomial*. (FIR)
3. Better still, $\mathbf{F}_p(z)$ might be the *transpose* of $\mathbf{H}_p(z)$. (orthogonality)

In case 3, the polyphase matrices are “paraunitary”. The analysis and synthesis banks are orthogonal. In the more general case 1, the banks are “biorthogonal”. In case 2, the synthesis bank is biorthogonal and also FIR.

The rows of a matrix are always biorthogonal to the columns of its inverse. When the rows of one are identical to the columns of the other, the matrix is self-orthogonal. Then it is an *orthogonal* matrix if real, a *unitary* matrix if complex, and a *paraunitary* matrix if it is a function of a complex parameter z .

Polyphase for Vectors

Any input vector x and any filter vector c or h can be separated into even and odd:

$$x = (\dots, x(0), 0, x(2), 0, \dots) + (\dots, 0, x(1), 0, x(3), 0, \dots).$$

The z -transform is separated into even powers and odd powers, as in

$$X(z) = [x(0) + x(2)z^{-2} + \dots] + z^{-1}[x(1) + x(3)z^{-2} + \dots]. \quad (4.22)$$

The even part has powers of z^2 . So has the odd part, when we factor out z^{-1} . This is the polyphase decomposition of x in the z -domain:

$$X(z) = X_{\text{even}}(z^2) + z^{-1}X_{\text{odd}}(z^2) \quad (4.23)$$

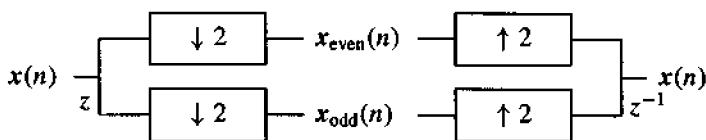
Each phase has its own z -transform:

$$\begin{aligned} x_{\text{even}} &= \begin{bmatrix} x(0) \\ x(2) \\ \vdots \end{bmatrix} \Leftrightarrow X_0(z) = \sum x(2k)z^{-k} \\ x_{\text{odd}} &= \begin{bmatrix} x(1) \\ x(3) \\ \vdots \end{bmatrix} \Leftrightarrow X_1(z) = \sum x(2k+1)z^{-k}. \end{aligned}$$

Because of the z^2 in the definition, the in-between zeros are gone from $X_0(z)$ and $X_1(z)$. Please verify that the phases of $X(z) = z^{-1} + z^{-2} + z^{-3}$ are $X_{\text{even}} = z^{-1}$ and $X_{\text{odd}} = 1 + z^{-1}$.

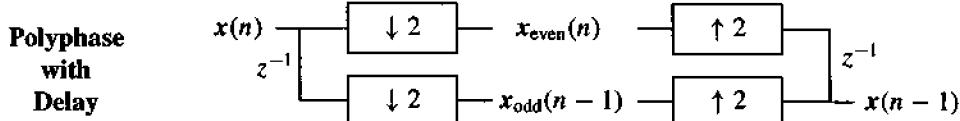
Now reverse the process, to recover x . Upsampling puts zeros back into x_{even} and x_{odd} . Those zeros change z to z^2 . The odd phase is delayed by z^{-1} , to move $x(1)$ from position 0 to position 1. Then addition reconstructs equation (4.25).

Here is the splitting and the reconstruction in block form. Notice that so far the filters are not included, and $(\downarrow 2)x$ is exactly x_{even} :



Important! The z at the start of the odd channel is because the odd phase has $x(1)$ in its zeroth position. We have to advance the signal to achieve that. (See Problem 1.) But advances look bad in our flow diagram. So the advance can be replaced by a delay, if we make up for it at the end by delaying the even part too.

Here is the “delay form” that we use in later sections. Please go through that form:

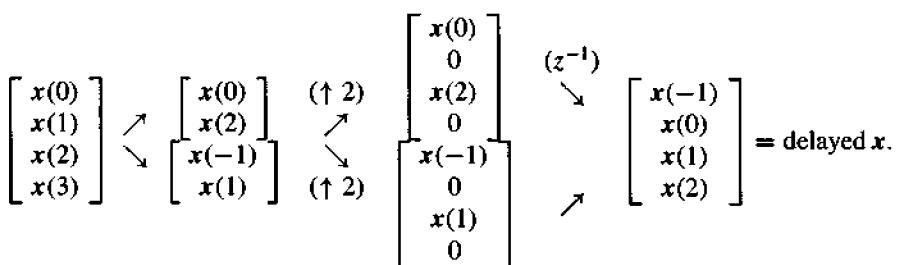


Only delays are involved! This is its advantage. Its disadvantage is that the output $\hat{x}(n)$ is $x(n - 1)$. The whole system equals a delay, where previously the system reproduced x . The delay form in the z -domain produces $z^{-1}X(z)$ by delaying the even term:

$$z^{-1}X(z) = z^{-1}X_0(z^2) + z^{-2}X_1(z^2). \quad (4.24)$$

This *polyphase with delay* is just the original definition multiplied by z^{-1} .

Your eye will pick out this delay form. The output \hat{x} comes later than the input. We still call this perfect reconstruction. Here is the same delay form in the time domain:



Polyphase Matrices for Filters

The polyphase form of a filter C comes directly from the polyphase form of c (the vector of filter coefficients). That vector separates into c_{even} and c_{odd} . Its z -transform $C(z)$ separates into phases exactly as $X(z)$ did:

$$C(z) = C_0(z^2) + z^{-1}C_1(z^2). \quad (4.25)$$

The filtering step is $C(z)X(z)$. This is ordinary filtering Cx , where even mixes with odd. But when downsampling picks out the even part of the product $C(z)X(z)$, it comes from even times even plus odd times odd. The transform of $(\downarrow 2)Cx$ is

$$(C(z)X(z))_{\text{even}} = C_0(z)X_0(z) + z^{-1}C_1(z)X_1(z). \quad (4.26)$$

The direct multiplication of $C(z)$ times $X(z)$ will have even parts from $C_0(z^2)X_0(z^2)$ and from $z^{-2}C_1(z^2)X_1(z^2)$. Those give the even part of $C(z)X(z)$. Downsampling picks out those terms. It changes z^2 to z in their transform. The result is the z -transform of $(\downarrow 2)Cx$.

Example 4.3. The moving average filter, downsampled.

Chapter 1 introduced the lowpass filter $\frac{1}{2}x(n) + \frac{1}{2}x(n-1)$. The coefficients are $h(0) = \frac{1}{2}$ and $h(1) = \frac{1}{2}$. Thus $\frac{1}{2}$ is the leading and only coefficient in H_{even} and H_{odd} . Both matrices have $\frac{1}{2}$ on one diagonal — the main diagonal. Here is the two-phase form of $(\downarrow 2)Hx$:

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \dots \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & & \\ & \frac{1}{2} & \\ & & \ddots \end{bmatrix} \begin{bmatrix} x(0) \\ x(2) \\ \vdots \end{bmatrix} + \begin{bmatrix} \frac{1}{2} & & \\ & \frac{1}{2} & \\ & & \ddots \end{bmatrix} \begin{bmatrix} x(-1) \\ x(1) \\ \vdots \end{bmatrix}$$

The polyphase components of $\frac{1}{2} + \frac{1}{2}z^{-1}$ are constants: $H_{\text{even}}(z) = H_{\text{odd}}(z) = \frac{1}{2}$.

Polyphase for one filter. The downsampling operator $(\downarrow 2)$ follows the filter C . The outputs are Cx and then $(\downarrow 2)Cx$. This is the normal order for filter banks, but it can be made more efficient. A close look at the product $(\downarrow 2)C$ shows that *the filter coefficients $c(n)$ are completely separated into even n and odd n* .

Thus C has two parts (or phases), which have their own z -transforms $C_0(z) = C_{\text{even}}(z)$ and $C_1(z) = C_{\text{odd}}(z)$. The 1 by 2 matrix $C_p(z)$ is the two-phase or polyphase form of $C(z)$:

$$C_p(z) = [C_0(z) \quad C_1(z)].$$

Warning. $C_0(z)$ is not $\frac{1}{2}(C(z) + C(-z))$. That involves $1, z^2, z^4, \dots$ with zeros between. This is $C_0(z^2)$. By definition, $C_0(z)$ closes the zero gaps and has coefficients $c(0), c(2), c(4)$. Similarly $C_1(z)$ has coefficients $c(1), c(3), c(5)$. An advance or delay is involved. We don't keep the zeros from the even powers in $\frac{1}{2}(C(z) - C(-z))$.

The same splitting occurs for the highpass filter D . Its even and odd phases are represented by $D_0(z)$ and $D_1(z)$. Those go into the 1 by 2 polyphase matrix $D_p(z)$. Then the whole analysis bank comes together when we combine the polyphase matrices for C and D into a single polyphase matrix $H_p(z)$:

$$\text{Polyphase Matrix } H_p(z) = \begin{bmatrix} C_p(z) \\ D_p(z) \end{bmatrix} = \begin{bmatrix} C_0(z) & C_1(z) \\ D_0(z) & D_1(z) \end{bmatrix} \quad (4.27)$$

This shows the matrix that we are aiming for. Now we go back for the close look at $(\downarrow 2)C$. This operator is fundamental in the theory of multirate filters and wavelets.

Polyphase in the time domain. When downsampling follows the filter C , we get the crucial matrix $L = (\downarrow 2)C$. This has to display the separation of even and odd, and I would like to show how this happens. Most of polyphase theory is developed in the z -domain, and we will do that too. But first, look at the filter matrix as it produces $y = Cx$:

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \\ \vdots \end{bmatrix} = \begin{bmatrix} \cdot & c(0) & & & \\ \cdot & c(1) & c(0) & & \\ \cdot & c(2) & c(1) & c(0) & \\ \cdot & c(3) & c(2) & c(1) & c(0) \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ \vdots \end{bmatrix}. \quad (4.28)$$

Downsampling leaves the even-numbered components $y(2n)$. To reach $v = (\downarrow 2)y$, we throw away the odd-numbered rows. This leaves the matrix $L = (\downarrow 2)C$:

$$\begin{bmatrix} \cdot \\ y(0) \\ y(2) \\ y(4) \\ \vdots \end{bmatrix} = \begin{bmatrix} \cdot & c(1) & c(0) & & \\ \cdot & c(3) & c(2) & c(1) & c(0) \\ \cdot & c(5) & c(4) & c(3) & c(2) & c(1) & c(0) \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} x(-1) \\ x(0) \\ x(1) \\ x(2) \\ \vdots \end{bmatrix}. \quad (4.29)$$

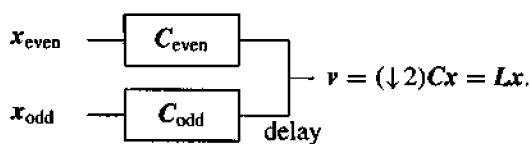
For polyphase here is the important point. *Only the even-numbered coefficients $c(2n)$ are multiplying the even-numbered coefficients $x(2n)$.* The even and odd c 's are in separate columns. The even-numbered $x(0)$ is multiplying the column that starts with $c(0)$. The odd-numbered component $x(1)$ is multiplying the column containing $c(1), c(3), \dots$. We can separate the matrix multiplication $(\downarrow 2)Cx$ into even times even and odd times odd:

$$\begin{bmatrix} \cdot \\ y(0) \\ y(2) \\ y(4) \\ \vdots \end{bmatrix} = \begin{bmatrix} \cdot & c(0) & & & \\ \cdot & c(2) & c(0) & & \\ \cdot & c(4) & c(2) & c(0) & \\ \cdot & \cdot & \cdot & \cdot & \end{bmatrix} \begin{bmatrix} \cdot \\ x(0) \\ x(2) \\ \vdots \end{bmatrix} + \begin{bmatrix} \cdot & c(1) & & & \\ \cdot & c(3) & c(1) & & \\ \cdot & c(5) & c(3) & c(1) & \\ \cdot & \cdot & \cdot & \cdot & \end{bmatrix} \begin{bmatrix} \cdot \\ x(-1) \\ x(1) \\ \vdots \end{bmatrix} \quad (4.30)$$

This is a matrix display of equation (4.21):

$$(\downarrow 2)Cx = C_{\text{even}} x_{\text{even}} + (\text{delay}) C_{\text{odd}} x_{\text{odd}}. \quad (4.31)$$

The two phases x_{even} and x_{odd} are filtered by the two polyphase components C_{even} and C_{odd} . We need a delay in the odd phase, because $c(1)x(1)$ contributes to $y(2)$ and not to $y(0)$. Then $(\downarrow 2)Cx$ is the sum from the two phases:



Notice something nice. The two matrices in equation (4.30) have constant diagonals. *The two operators C_{even} and C_{odd} are time-invariant filters.* They have frequency responses $C_0(z) =$

$C_{\text{even}}(z)$ and $C_1(z) = C_{\text{odd}}(z)$. The delay in the odd channel can go before or after C_1 , because it commutes with C_1 . (C_1 is time-invariant!) The two filters involve even coefficients $c(2n)$ and odd coefficients $c(2n + 1)$, without zeros in between. They can operate in parallel, more efficiently.

Summary: Polyphase form of $L = (\downarrow 2)C$

The matrix C_{even} multiplies x_{even} in equation (4.31). The matrix C_{odd} multiplies x_{odd} (with a delay). We repeat this time-domain multiplication so you can compare it with the z -domain:

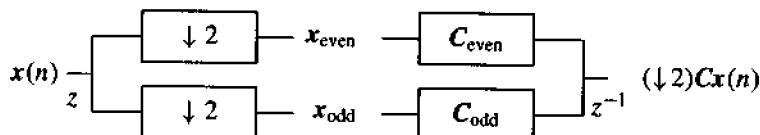
$$(\downarrow 2)Cx = [C_{\text{even}} \quad (\text{delay})C_{\text{odd}}] \begin{bmatrix} x_{\text{even}} \\ x_{\text{odd}} \end{bmatrix}. \quad (4.32)$$

This polyphase form has two ordinary filter matrices side by side. The z -domain polyphase form has two ordinary transfer functions side by side:

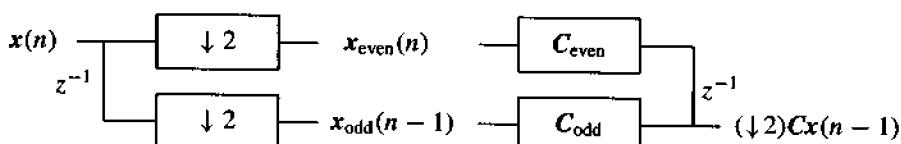
$$(C(z)X(z))_{\text{even}} = [C_{\text{even}}(z) \quad z^{-1}C_{\text{odd}}(z)] \begin{bmatrix} X_0(z) \\ X_1(z) \end{bmatrix}. \quad (4.33)$$

If C = identity then $C_{\text{even}}(z) = 1$ and the output is $X_0(z)$. If C = delay then $C_{\text{odd}}(z) = 1$ and the output is $X_1(z)$. All straightforward, but the block form shows something remarkable:
Downsampling comes before filtering!

The block form with an advance to compute x_{odd} is **Polyphase with Advance**:



An advance has slipped into this form to get $x(1)$ as the zeroth component of x_{odd} . If we prefer to have only delays (and we do), that is possible. Delay x in the odd channel so that $(\downarrow 2)$ produces $x_{\text{odd}}(n - 1)$. Then after filtering, add a delay in the even channel. The result is to delay the whole signal by one time step in **Polyphase with Delay**:



Key point: The polyphase form puts $(\downarrow 2)$ *before* the filters. This order is more efficient. It was possible to use the Noble Identity on each phase separately, because $C_{\text{even}}(z^2)$ and also $C_{\text{odd}}(z^2)$ appeared in the right place:

$$(\text{direct}) \quad (\downarrow 2)C_{\text{even}}(z) = C_{\text{even}}(z^2)(\downarrow 2) \quad (\text{polyphase}).$$

Note on our convention. It would be very satisfying to avoid the delays completely. We would prefer to have

$$V(z) = [C_0(z) \ C_1(z)] \begin{bmatrix} X_0(z) \\ X_1(z) \end{bmatrix}. \quad (4.34)$$

To achieve this we can alter the definition of $C_1(z)$. The polyphase decomposition of X would stay the same, but the decomposition of C would change to

$$C(z) = C_0(z^2) + zC_1(z^2). \quad (4.35)$$

Note z in the odd term where we had z^{-1} . This is the convention chosen by [VK]. It is just as good as ours. The zeroth component of C_{odd} becomes $c(-1)$. It multiplies $x(1)$. But I am afraid that in later chapters you would forget (I would too) this convention for c , different from x . So we keep the same even-odd decomposition of c and x , yielding parallel decompositions of $C(z)$ and $X(z)$ — and requiring the delay.

Problem Set 4.2

- Find $X_{\text{even}}(z)$ and $X_{\text{odd}}(z)$ when $X(z) = 1 + 2z^{-5} + z^{-10}$. Verify that $X_{\text{even}}(z^2) = \frac{1}{2}(X(z) + X(-z))$ and $X_{\text{odd}}(z^2) = \frac{1}{2}(X(z) - X(-z))$. The odd definition involves an advance!
- Express the z -transform of $\uparrow 2(\downarrow 2)x$ in terms of X_{odd} and/or X_{even} . What operations on x would produce the vector whose transform is $X_1(z^2)$?
- The phases C_0, C_1, D_0, D_1 are all time-invariant, so they commute with delays. Does it follow that

$$\begin{bmatrix} 1 & \\ & \text{delay} \end{bmatrix} \begin{bmatrix} C_0 & C_1 \\ D_0 & D_1 \end{bmatrix} = \begin{bmatrix} C_0 & C_1 \\ D_0 & D_1 \end{bmatrix} \begin{bmatrix} 1 & \\ & \text{delay} \end{bmatrix}?$$

4. Polyphase Representation of an IIR Transfer Function

Let $H(z) = \frac{1}{1-az^{-1}}$ where $0 < a < 1$. Its impulse response is $h(n) = a^n$ for $n \geq 0$ (and zero for negative n). The phases are $h_{\text{even}}(n) = (1, a^2, a^4, \dots)$ and $h_{\text{odd}}(n) = (a, a^3, a^5, \dots)$. The z -transforms are $H_{\text{even}}(z) = 1/(1-a^2z^{-1})$ and $H_{\text{odd}}(z) = a/(1-a^2z^{-1})$. This method is very cumbersome. One has to find the impulse response $h(n)$, then its even and odd parts $h_{\text{even}}(n)$ and $h_{\text{odd}}(n)$, then the z -transforms.

An alternate method is to write $H(z) = \frac{1}{1-az^{-1}}$ directly as $H(z) = H_{\text{even}}(z^2) + z^{-1}H_{\text{odd}}(z^2)$. The denominator must be a function of z^2 . So multiply above and below by $1+az^{-1}$:

$$H(z) = \frac{1}{1-az^{-1}} \frac{1+az^{-1}}{1+az^{-1}} = \frac{1+az^{-1}}{1-a^2z^{-2}} = \frac{1}{1-a^2z^{-2}} + z^{-1} \frac{a}{1-a^2z^{-2}}.$$

This displays $H_{\text{even}}(z)$ and $H_{\text{odd}}(z)$. An N th order filter can be factored as a cascade of first-order sections, and this method applies to each section.

- Let $H(z) = \frac{1}{1-\frac{2}{3}z^{-1}+\frac{1}{3}z^{-2}}$. Factor $H(z)$ into two first-order poles. Find the polyphase components of $H(z)$.
- Let $H(z) = \frac{1+2z^{-1}+5z^{-2}}{1-\frac{2}{3}z^{-1}+\frac{1}{3}z^{-2}}$. What are its polyphase components?

- For $M = 4$ channels, we want the four polyphase components of $H(z)$:

$$H(z) = H_0(z^4) + z^{-1}H_1(z^4) + z^{-2}H_2(z^4) + z^{-3}H_3(z^4).$$

- What polynomial multiplies $1-az^{-1}$ to produce $1-a^4z^{-4}$?

- Find the four components for $\frac{1}{1-az^{-1}}$ and $\frac{1+2z^{-1}+5z^{-2}}{1-\frac{2}{3}z^{-1}+\frac{1}{3}z^{-2}}$.

6. If H is a symmetric filter, how many of its phases are symmetric filters?
7. Let $H(z) = 1 + 2z^{-1} + 3z^{-2} + 4z^{-3} + 4z^{-4} + 3z^{-5} + 2z^{-6} + z^{-7}$. Find the polyphase components $H_{even}(z)$ and $H_{odd}(z)$. What is the relation between $H_{even}(z)$ and $H_{odd}(z)$ for antisymmetric filters of even length and symmetric filters of odd length?
8. What are the two polyphase components of a symmetric halfband filter? Generalize to an M -th band filter.

4.3 Efficient Filter Banks

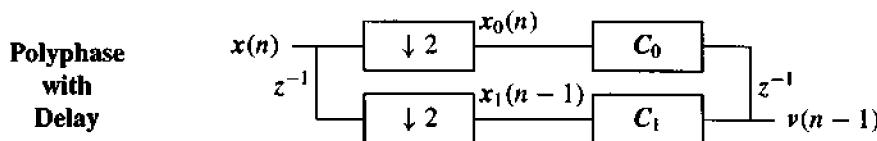
Let us repeat the key idea of the polyphase decomposition. The input is x , the filter is C , and we intend to downsample Cx . We want the *even powers of z* in the product $C(z)X(z)$:

$$\text{even powers come from } \begin{cases} \text{even powers in } C(z) \text{ times even powers in } X(z) \\ \text{odd powers in } C(z) \text{ times odd powers in } X(z). \end{cases}$$

The polyphase decomposition is exactly this separation. Even times even is $C_0(z^2)X_0(z^2)$. Odd times odd starts with $c(1)z^{-1}$ times $x(1)z^{-1}$. The product $c(1)x(1)$ enters $Cx(2)$ because indices add. Then $Cx(2)$ after downsampling is $v(1)$. Since $c(1)$ and $x(1)$ are the *zeroth components* of the odd phases, we need a delay to put their product into $v(1)$. The algebra with $X(z) = X_0(z^2) + z^{-1}X_1(z^2)$ and $C(z) = C_0(z^2) + z^{-1}C_1(z^2)$ is

$$\begin{aligned} (CX)_0(z^2) &= C_0(z^2)X_0(z^2) + z^{-2}C_1(z^2)X_1(z^2) \\ (CX)_0(z) &= C_0(z)X_0(z) + z^{-1}C_1(z)X_1(z). \end{aligned} \quad (4.36)$$

We reached this answer by filtering and then subsampling (the step when z^2 becomes z). But now we see a better way. *Sample first* to get $x_0 = x_{\text{even}}$ and $x_1 = x_{\text{odd}}$. Filter those separately (and in parallel) by C_0 and C_1 . Then combine the outputs with a suitable delay — the step down on the right:

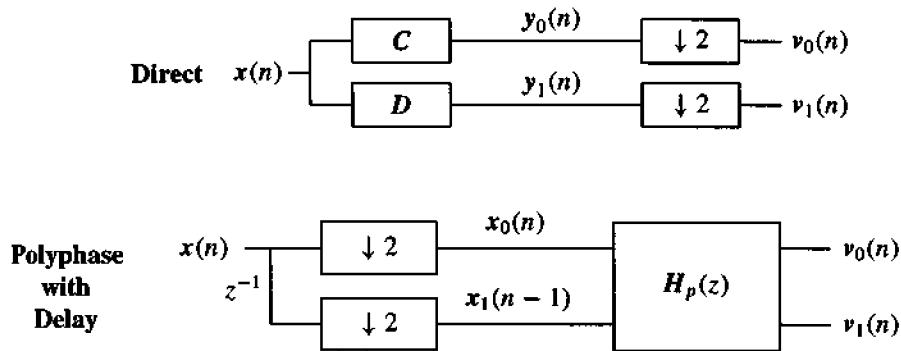


In this polyphase form, the filters C_0 and C_1 are half as long as C . The output can be computed twice as fast, if all operations are done at the same speed. We could also use two cheaper low bandwidth processors, both working full time. They do $\frac{N}{2}$ multiplications and additions per unit time instead of N . By downsampling first, we reduce the input rate to the filters. The bandwidth is halved.

Example 4.4. The averaging filter in its direct form computes $\frac{1}{2}x(0) + \frac{1}{2}x(-1)$. Then it sits idle for one clock step, before computing $\frac{1}{2}x(2) + \frac{1}{2}x(1)$. The polyphase form has H_{even} and H_{odd} multiplying separately by $\frac{1}{2}$. They do one multiplication each, in parallel. An easy-to-write code will execute the polyphase form.

Polyphase for Filter Banks

The polyphase idea extends from one filter to a bank of filters. The direct form of the analysis bank does the downsampling last. *The polyphase form does the downsampling first.* In the block diagram of the filter bank, the decimators move *outside* the filters. We can write C and D or H_0 and H_1 for the lowpass and highpass filters (0 and 1 do not mean even and odd!):



The *polyphase matrix* multiplies $X_0(z)$ and $z^{-1}X_1(z)$ to produce $V_0(z)$ and $V_1(z)$:

$$\begin{bmatrix} V_0(z) \\ V_1(z) \end{bmatrix} = \begin{bmatrix} C_0(z) & C_1(z) \\ D_0(z) & D_1(z) \end{bmatrix} \begin{bmatrix} X_0(z) \\ z^{-1}X_1(z) \end{bmatrix} = H_p(z) \begin{bmatrix} X_0(z) \\ z^{-1}X_1(z) \end{bmatrix} \quad (4.37)$$

This defines and displays $H_p(z)$. For FIR causal filters, the kind we expect to use, the polyphase components are polynomials in z^{-1} . When the input x is also causal, the outputs are causal.

Another point about notation. The indices in X_0 and X_1 refer to even and odd. The indices in V_0 and V_1 refer to the two channels. This is normal for matrix multiplication, when H_{ij} multiplies X_j and contributes to V_i . *Rows of $H_p(z)$ go with channels, and columns of $H_p(z)$ go with phases.*

In an M -channel bank, i is the channel index and j is the phase index in $H_{ij}(z)$. Then V_i is the output from channel i , and X_j is the j th phase of the input. We often reorganize a filter bank into its polyphase form.

Example 4.5. Average-difference filter bank in polyphase form.

The averaging filter $H_0x(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$ has polyphase components $H_{0,\text{even}} = H_{0,\text{odd}} = \frac{1}{2}$ (identity). The differencing filter $H_1x(n) = \frac{1}{2}x(n) - \frac{1}{2}x(n-1)$ has components $H_{1,\text{even}} = \frac{1}{2}I$ and $H_{1,\text{odd}} = -\frac{1}{2}I$. Note that $H_0(z)$ and $H_1(z)$ are linear but the polyphase matrix is constant — typical of a *block transform*:

$$H_p(z) = \begin{bmatrix} H_{00}(z) & H_{01}(z) \\ H_{10}(z) & H_{11}(z) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}.$$

Example 4.6. Four-tap filters yield two taps for each phase. The even phase C_{even} has two coefficients $c(0)$ and $c(2)$. The odd phase has $C_{\text{odd}} = c(1) + c(3)z^{-1}$. The same pattern holds for D . The polyphase matrix for the filter bank is

$$H_p(z) = \begin{bmatrix} c(0) + c(2)z^{-1} & c(1) + c(3)z^{-1} \\ d(0) + d(2)z^{-1} & d(1) + d(3)z^{-1} \end{bmatrix}. \quad (4.38)$$

Even is separated from odd. This reflects what happens in the filter bank, when Cx and Dx are downsampled:

$$\begin{bmatrix} v_0 \\ v_1 \end{bmatrix} = \begin{bmatrix} (\downarrow 2)Cx \\ (\downarrow 2)Dx \end{bmatrix} = \begin{bmatrix} C_{\text{even}} & C_{\text{odd}} \\ D_{\text{even}} & D_{\text{odd}} \end{bmatrix} \begin{bmatrix} 1 & \text{delay} \end{bmatrix} \begin{bmatrix} x_{\text{even}} \\ x_{\text{odd}} \end{bmatrix}$$

In case you like matrices, we are going to write the time-domain filter bank matrix in three ways. Downsampling is included in all three! First comes the matrix $H_d = H_{\text{direct}}$ that multiplies the input vector x in the direct form:

$$\text{Direct } H_d = \begin{bmatrix} \cdot & \cdot \\ c(3) & c(2) & c(1) & c(0) & c(3) & c(2) & c(1) & c(0) \\ \cdot & \cdot \\ d(3) & d(2) & d(1) & d(0) & d(3) & d(2) & d(1) & d(0) \\ \cdot & \cdot \end{bmatrix}. \quad (4.39)$$

Downsampling has removed every other row. That leaves this “square” infinite matrix. Each column is completely odd or completely even.

For the second form we rearrange the *rows* of H_d . The highpass outputs are interleaved with the lowpass outputs, both downsampled by 2. This produces the block-diagonal form (or *block Toeplitz form*) $H_b = H_{\text{block}}$:

$$\text{Block } H_b = \begin{bmatrix} \cdot & \cdot \\ c(3) & c(2) & c(1) & c(0) & c(3) & c(2) & c(1) & c(0) \\ d(3) & d(2) & d(1) & d(0) & d(3) & d(2) & d(1) & d(0) \\ \cdot & \cdot \\ c(3) & c(2) & c(1) & c(0) & c(3) & c(2) & c(1) & c(0) \\ d(3) & d(2) & d(1) & d(0) & d(3) & d(2) & d(1) & d(0) \\ \cdot & \cdot \end{bmatrix}. \quad (4.40)$$

Your eye will divide that matrix into 2 by 2 blocks. It is like an ordinary time-invariant constant-diagonal matrix, but the entries are blocks instead of scalars. The main diagonal block corresponds to the constants in the polyphase matrix. The subdiagonal block produces the z^{-1} terms. There are only two diagonals because the phases of C and D have *two* coefficients. The original C and D had four coefficients.

The third form is the polyphase form H_p . We are still in the time domain. For this third form we rearrange the *columns* of the direct form:

$$\text{Polyphase } H_p = \begin{bmatrix} \cdot & \cdot \\ c(2) & c(0) & c(3) & c(1) & c(2) & c(0) & c(3) & c(1) \\ \cdot & \cdot \\ d(2) & d(0) & d(3) & d(1) & d(2) & d(0) & d(3) & d(1) \\ \cdot & \cdot \end{bmatrix} = \begin{bmatrix} C_0 & C_1 \\ D_0 & D_1 \end{bmatrix}. \quad (4.41)$$

When the columns are rearranged, the vector \mathbf{x} must be rearranged. Here \mathbf{x}_{even} comes above \mathbf{x}_{odd} (delayed). The transform of the time-domain matrix \mathbf{H}_p is the z -domain polyphase matrix $\mathbf{H}_p(z)$. This 2 by 2 matrix of filters becomes a 2 by 2 matrix of functions.

The block form \mathbf{H}_b is an infinite matrix of 2 by 2 blocks. The polyphase form \mathbf{H}_p is a 2 by 2 matrix of infinite blocks. Each block is a time-invariant filter. Either form leads by z -transform to the 2 \times 2 polyphase matrix $\mathbf{h}_p(0) + z^{-1}\mathbf{h}_p(1)$:

$$\begin{array}{l} \text{Polyphase} \\ \text{matrix} \end{array} \quad \mathbf{H}_p(z) = \left[\begin{array}{cc} c(0) & c(1) \\ d(0) & d(1) \end{array} \right] + z^{-1} \left[\begin{array}{cc} c(2) & c(3) \\ d(2) & d(3) \end{array} \right] \quad (4.42)$$

Relation Between Modulation and Polyphase

To produce two vectors from \mathbf{x} , one way is by polyphase. The parts \mathbf{x}_{even} and \mathbf{x}_{odd} are half-length. The other way is by modulation. Both \mathbf{x} and \mathbf{x}_{mod} are full length (therefore redundant). The modulated vector \mathbf{x}_{mod} reverses the sign of odd-numbered components:

$$\mathbf{x}_{\text{mod}}(n) = (-1)^n \mathbf{x}(n) \quad \text{and} \quad X_{\text{mod}}(z) = X(-z).$$

In the frequency domain, $-1 = e^{i\pi}$ and ω is modulated by π --- which explains the name:

$$X_{\text{mod}}(\omega) = X(\omega + \pi). \quad (4.43)$$

The vector \mathbf{x}_{mod} appears naturally when upsampling follows downsampling. Remember that $\mathbf{u} = (\uparrow 2)(\downarrow 2)\mathbf{x}$ is the average of \mathbf{x} and \mathbf{x}_{mod} :

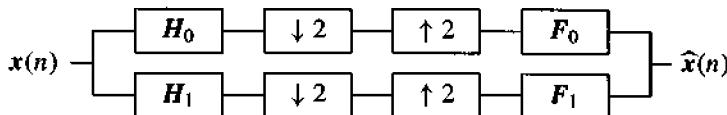
$$\mathbf{u} = \frac{1}{2}(\mathbf{x} + \mathbf{x}_{\text{mod}}) = (\dots, \mathbf{x}(0), 0, \mathbf{x}(2), 0, \dots). \quad (4.44)$$

In the z -domain, the same fact gives the formula we use constantly:

$$U(z) = \frac{1}{2}[X(z) + X(-z)].$$

For downsampling and upsampling by M , the frequency modulation is by multiples of $2\pi/M$. The z -domain equivalent is multiplication by $W = e^{-2\pi i/M}$. There are M terms $X(W^k z)$. We continue with $M = 2$ terms, $X(z)$ and $X(-z)$.

The extra term represents aliasing. This modulated signal can overlap the original signal (in the frequency domain). If it does, we cannot recover \mathbf{x} from \mathbf{u} . Now place this step $(\uparrow 2)(\downarrow 2)$ into a complete filter bank:



The transform of $H_0\mathbf{x}$ is $H_0(z)X(z)$. When this is downsampled and upsampled, its alias appears in $\frac{1}{2}[H_0(z)X(z) + H_0(-z)X(-z)]$. Multiplying by $F_0(z)$ gives the output from the top channel. The lower channel is the same with 0 replaced by 1. The final output is the sum of channels:

$$\widehat{X}(z) = \frac{1}{2}[F_0(z) \quad F_1(z)] \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} \begin{bmatrix} X(z) \\ X(-z) \end{bmatrix} \quad (4.45)$$

This is the 2×2 modulation matrix $H_m(z)$. It contains $H_0(z)$ and $H_1(z)$ along with their aliases. *The transpose of $H_m(z)$ is also called the alias component matrix.*

A perfect reconstruction filter bank has to avoid aliasing (and other distortions). There is no aliasing when the combination $F_0(z)H_0(-z) + F_1(z)H_1(-z)$ is zero. Section 4.1 presented the synthesis filters $F_0(z) = H_1(-z)$ and $F_1(z) = -H_0(-z)$ that cancel aliasing.

Now turn to polyphase, the separation into even and odd phases:

$$H(z) = \frac{1}{2}[H(z) + H(-z)] + \frac{1}{2}[H(z) - H(-z)]. \quad (4.46)$$

The first bracket is even. Replacing z by $-z$ leaves it unchanged. The second bracket is odd. Replacing z by $-z$ reverses its sign. Therefore (4.46) must be the polyphase decomposition $H_{\text{even}}(z^2) + z^{-1}H_{\text{odd}}(z^2)$:

$$\begin{aligned} [H_{\text{even}}(z^2) & z^{-1}H_{\text{odd}}(z^2)] = \frac{1}{2}[H(z) + H(-z) & H(z) - H(-z)] \\ &= \frac{1}{2}[H(z) & H(-z)] \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \end{aligned} \quad (4.47)$$

This connects polyphase to modulation. For one filter H_p and H_m have one row. For a bank of filters H_p and H_m are square matrices. A 2-channel analysis bank has equation (4.47) for H_0 in the top row, and the same equation for H_1 in the lower row:

$$\begin{bmatrix} H_{0,\text{even}}(z^2) & H_{0,\text{odd}}(z^2) \\ H_{1,\text{even}}(z^2) & H_{1,\text{odd}}(z^2) \end{bmatrix} \begin{bmatrix} 1 & \\ & z^{-1} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4.48)$$

Theorem 4.4 *The polyphase matrix H_p is connected to the modulation matrix H_m by a 2-point DFT and a diagonal delay matrix $D(z) = \text{diag}(1, z^{-1})$.*

This pattern extends in Section 9.2 to a bank of M filters. Each filter has M phases, and it has M modulations (by multiples of $2\pi/M$). The phases are associated with time shifts, and the modulations are frequency shifts. So the DFT connects them.

Problem Set 4.3

- From $H(z) = H_{\text{even}}(z^2) + z^{-1}H_{\text{odd}}(z^2)$, find the corresponding formula for $H(-z)$. Write the two equations as

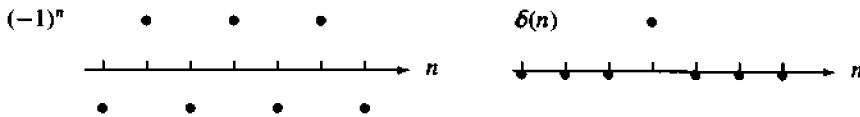
$$[H(z) & H(-z)] = [H_{\text{even}}(z^2) & H_{\text{odd}}(z^2)] \begin{bmatrix} 1 & \\ & z^{-1} \end{bmatrix} \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix}.$$

Identify that last matrix and verify that you have inverted equation (4.48). What is $H_m(z)$ in terms of $H_p(z)$?

- If $H_m^T(z^{-1})H_m(z) = 2I$ show from (4.48) that $H_p^T(z^{-1})H_p(z) = I$.
- Find a new example of matrices $H_m(z)$ and $H_p(z)$ for Problem 2.
- Write down the equations for $M = 4$ that are analogous to (4.46)–(4.48).

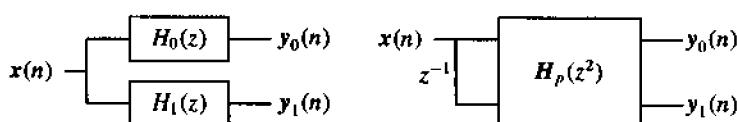
Problems 5–11 develop an important example of biorthogonal filters.

- The upper channel has responses $H_0(z) = 1$ and $F_0(z) = \frac{1}{2} + z^{-1} + \frac{1}{2}z^{-2} = \frac{1}{2}(1 + z^{-1})^2$. Follow the signals $x(n) = (-1)^n$ and $\delta(n)$ through this channel by plotting $(\uparrow 2)(\downarrow 2)H_0x(n)$.



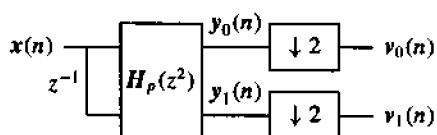
Plot the output $w_0(n) = F_0(\uparrow 2)(\downarrow 2)H_0x(n)$ on top of these inputs. Describe the output from this channel with any input $x(n)$: The odd $w_0(2k+1)$ are _____ and the even $w_0(2k)$ are _____.

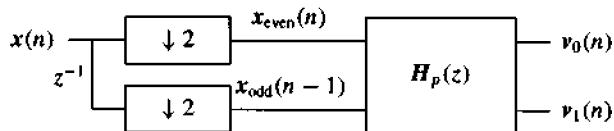
6. The lower channel has responses $H_1(z) = F_0(-z) = \frac{1}{2} - z^{-1} + \frac{1}{2}z^{-2} = \frac{1}{2}(1 - z^{-1})^2$ and $F_1(z) = -H_0(-z) = -1$. Follow the same two inputs $(-1)^n$ and $\delta(n)$ through this channel by plotting $H_1x(n)$ and $w_1(n) = F_1(\uparrow 2)(\downarrow 2)H_1x(n)$. Verify that the sum of outputs $w_0(n) + w_1(n)$ is $\hat{x}(n) = x(n-1)$. In words, the output $w_1(n)$ from the lower channel is _____ for $n = 2k+1$ and _____ for $n = 2k$ for any input $x(n)$.
 7. Verify the perfect reconstruction condition on $F_0(z)H_0(z) + F_1(z)H_1(z)$ for this filter bank. What is the delay ℓ ? What is the product filter $P_0(z)$? Find the centered product filter $P(z)$, which is halfband.
 8. Find the modulation matrix $H_m(z)$ for the analysis bank. Find the synthesis matrix $F_m(z)$, remembering the transpose. Compute the product $F_m(z)H_m(z)$.
 9. Reverse the filters in the upper channel by $H_0(z) = \frac{1}{2} + z^{-1} + \frac{1}{2}z^{-2}$ and $F_0(z) = 1$. Follow the same signals $x(n) = (-1)^n$ and $x(n) = \delta(n)$ through this new channel.
 10. Construct the corresponding $H_1(z) = F_0(-z)$ and $F_1(z) = -H_0(-z)$ and follow the two inputs through the lower channel. Verify that the outputs $w_0(n) + w_1(n)$ reconstruct the impulse $x(n)$. Certainly $F_0(z)H_0(z) + F_1(z)H_1(z)$ is the same as before (still PR). The halfband filter $P(z) = \frac{1}{2}z + 1 + \frac{1}{2}z^{-1}$ was factored into $P(z) * 1$ and then $1 * P(z)$. Which filter bank factors $P(z)$ into $(1 + z^{-1})/\sqrt{2}$ times $(1 + z)/\sqrt{2}$?
- Big question: Is the more regular filter $\frac{1}{2} + z^{-1} + \frac{1}{2}z^{-2}$ better in analysis (H_0) or in synthesis (F_0)? The output w_0 should be a “good but compressed” copy of x .
11. Show that the linear signal $x(n) = n$ goes entirely through the lowpass channel. The highpass channel has $H_1x(n) = \underline{\hspace{2cm}}$. The constant signal $x(n) = 1$ also goes through, so this filter bank (still from Problem 5) has accuracy $p = 2$.
 12. Explain the equivalence of these representations before downsampling:



In matrix notation this is $\begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix} = \begin{bmatrix} H_{00}(z^2) & H_{01}(z^2) \\ H_{10}(z^2) & H_{11}(z^2) \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix}$.

13. Explain the equivalence of these representations including downsampling:





Warning for downsampling by 3. The polyphase components when $M = 3$ are C_0, C_1, C_2 and X_0, X_1, X_2 . We want the component Y_0 of the product, because Y_1 and Y_2 are lost in downsampling. That component Y_0 does not involve C_1 times X_1 . Multiplying z^{-1} by z^{-1} does not give z^{-3} . The exponents 0, 3, 6, ... come from C_2 times X_1 and C_1 times X_2 (and also C_0 times X_0). To get z^{-3} in the product, we multiply z^{-1} by z^{-2} or z^0 by z^{-3} :

$$Y_0(z^3) = C_0(z^3)X_0(z^3) + z^{-3}C_1(z^3)X_2(z^3) + z^{-3}C_2(z^3)X_1(z^3).$$

Then replace z^3 by z to find the transform of $(\downarrow 3)\mathbf{y} = (\downarrow 3)\mathbf{Cx}$.

We save this pattern for a later discussion of $(\downarrow M)$. It uses “Type 1 polyphase” components C_0, C_1, C_2 . By defining Type 2 components R_0, R_1, R_2 we could achieve that R_i multiplies X_i . (The R ’s are a permutation of the C ’s.) Our immediate interest is restricted to $(\downarrow 2)$.

14. Write down the formula for $Y_0(z^3)$ that corresponds to equation (4.46). The polyphase components of C are C_0, C_1, C_2 .
15. Show that $X_{even}(z)$ is the z -transform of the downsampled vector $(\downarrow 2)\mathbf{x}$. What vector is transformed to $X_{odd}(z)$?
16. Write $H_p(z)$ in terms of $H_0(z)$ for these filters of length $N + 1$.
 - (a) $H_1(z) = H_0(-z)$
 - (b) $H_1(z) = z^{-N}H_0(z^{-1})$
 - (c) $H_1(z) = z^{-N}H_0(-z^{-1})$

17. Find the analysis filters $H_0(z)$ and $H_1(z)$ for the following polyphase matrices:

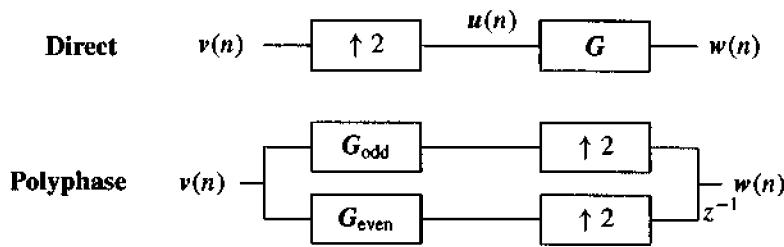
- a. $H_p(z) = \begin{bmatrix} 1 + 2z^{-1} - z^{-2} & 2 - z^{-1} \\ z^{-3} & 1 + 2z^{-1} + z^{-2} \end{bmatrix}$
- b. $H_p(z) = \begin{bmatrix} 1 + 2z^{-2} & 1 + z^{-1} \\ 1 - z^{-1} & 2 + z^{-1} \end{bmatrix} \begin{bmatrix} 2 & 1 + z^{-2} \\ 1 - z^{-1} & -z^{-3} \end{bmatrix}$
- c. $H_p(z) = \begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix}$
- d. $H_p(z) = \begin{bmatrix} \frac{1}{2+z^{-1}} & 1 \\ z^{-3} & \frac{1+z^{-1}}{3-z^{-1}} \end{bmatrix}$
- e. $H_p(z) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \frac{1+2z^{-1}}{2+z^{-1}} & 0 \\ 0 & \frac{1-2z^{-1}}{2-z^{-1}} \end{bmatrix}$

18. For each system (a-e) in Problem 17, find the modulation matrix $H_m(z)$.

4.4 Polyphase for Upsampling and Reconstruction

The reader can anticipate what is coming. The polyphase form above was for the analysis bank, with downsampling. The same ideas apply to the synthesis bank, with upsampling. The expanders ($\uparrow 2$) will again move outside the filters in the polyphase implementation. This time “outside” means *after* the synthesis filters.

We start with one filter called G , and move the upsampler:



The polyphase form is more efficient, because the half-length filters $G_0 = G_{\text{even}}$ and $G_1 = G_{\text{odd}}$ receive input at the *slow rate* (low bandwidth). The direct form unnecessarily doubles the rate by $(\uparrow 2)$ before the filter.

It is always important to verify formulas in the z -domain. In the direct form, upsampling produces $U(z) = V(z^2)$. The only powers to enter $U(z)$ are *even* powers, because upsampling puts zeros in the odd components. Then the filter produces $W(z) = G(z)V(z^2)$. In this multiplication, even powers in G yield even powers in W . The odd phase of G multiplies $V(z^2)$ to give the odd phase of W . That is why polyphase works in the synthesis filters. Even times even is separated from odd times even:

$$W(z) = [1 \ z^{-1}] \begin{bmatrix} W_{\text{even}}(z) \\ W_{\text{odd}}(z) \end{bmatrix} = [1 \ z^{-1}] \begin{bmatrix} G_{\text{even}}(z^2) \\ G_{\text{odd}}(z^2) \end{bmatrix} V(z^2).$$

Suppose the filter G is a simple delay. Then its Type 1 polyphase components are $G_{\text{even}} = 0$ and $G_{\text{odd}} = \text{identity}$. The output from this polyphase equation is correct—a delay of $\uparrow 2v$.

Example 4.7. Suppose G has four coefficients $g(0), g(1), g(2), g(3)$. In the time domain, G is a constant-diagonal matrix. When upsampling acts before the filter to produce $G(\uparrow 2)$, it removes columns of G . (Just as downsampling removed rows of C .) The time-domain matrix in direct and inefficient form is $G(\uparrow 2)$:

$$G(\uparrow 2) = \begin{bmatrix} g(0) & & & \\ g(1) & g(0) & & \\ g(2) & g(1) & g(0) & \\ g(3) & g(2) & g(1) & g(0) \\ & & \ddots & \end{bmatrix} \begin{bmatrix} 1 & & & \\ 0 & & & \\ & 1 & & \\ 0 & & \ddots & \end{bmatrix} = \begin{bmatrix} g(0) & & & \\ g(1) & & & \\ g(2) & g(0) & & \\ g(3) & g(1) & & \\ & & \ddots & \end{bmatrix} \quad (4.49)$$

Now comes the polyphase idea. Each row is either all even or all odd. The even rows (*not columns as for C*) combine into G_{even} . The odd rows combine into G_{odd} . Each of those is a constant-diagonal matrix (a time-invariant filter!). When they act separately and simultaneously, the process is more efficient:

$$(\uparrow 2) \begin{bmatrix} G_{\text{even}} \\ G_{\text{odd}} \end{bmatrix} = \begin{bmatrix} g(0) & & & \\ g(2) & g(0) & & \\ & & \ddots & \\ g(1) & & & \\ g(3) & g(1) & & \\ & & \ddots & \end{bmatrix}$$

In this polyphase form, upsampling comes *after* the two filters. The odd phase is delayed before the even and odd outputs are combined:

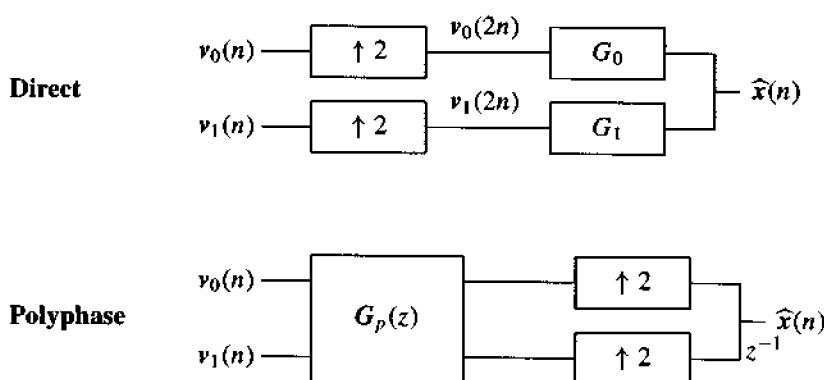
$$G(\uparrow 2) \text{ becomes } [1 \text{ delay}](\uparrow 2) \begin{bmatrix} G_{\text{even}} \\ G_{\text{odd}} \end{bmatrix}. \quad (4.50)$$

This is the polyphase form in the time domain. Its transform is the polyphase form in the z -domain. We are using the second Noble Identity *on each phase separately*:

$$\text{direct } (\uparrow 2)G_{\text{even}}(z^2) = G_{\text{even}}(z)(\uparrow 2) \quad \text{polyphase}$$

Synthesis Bank: Direct and Polyphase

Now put *two filters* into a synthesis filter bank. The filters will be G_0 and G_1 . Remember that synthesis has two input vectors (at half rate) and one output vector (at full rate). The direct form and the polyphase form are (with our present conventions) as follows:



This is a 2-channel synthesis bank. In the z -domain, the *polyphase matrix* $G_p(z)$ multiplies the inputs $V_0(z)$ and $V_1(z)$. The indices 0 and 1 are now channel numbers. The extension of equation (4.41) to two signals coming through two filters G_0 and G_1 will produce $G_p(z)$:

$$\hat{X}(z) = [1 \ z^{-1}] \begin{bmatrix} G_0, \text{even}(z^2) & G_1, \text{even}(z^2) \\ G_0, \text{odd}(z^2) & G_1, \text{odd}(z^2) \end{bmatrix} \begin{bmatrix} V_0(z^2) \\ V_1(z^2) \end{bmatrix} \quad (4.51)$$

Notice! In the analysis polyphase matrix $H_p(z)$, the lower left entry was H_{10} . Now G_{01} is at the lower left. Where we had the even part of H_1 , we have the odd part of G_0 . The channel index always comes before the phase index, and this forces a transpose in the polyphase synthesis matrix. Other authors agree with this convention — a necessary evil.

In the M by M case, H_{ij} is the j th polyphase component of the filter in the i th channel. So is G_{ij} . But the entry in row i , column j of the synthesis matrix is G_{ji} .

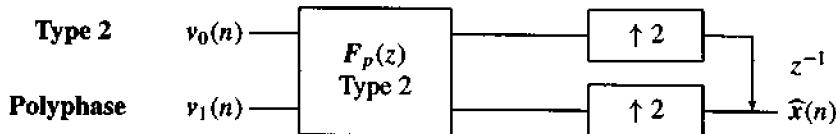
Useful convention for synthesis: Type 2 polyphase. Vaidyanathan delays channel 0 instead of channel 1, at the end of the synthesis bank. Therefore he reverses the two polyphase components. This produces the Type 2 polyphase decomposition, where 1 means even and 0 means odd. We write F rather than G to keep this type separate:

$$\text{Type 2 is } F(z) = F_1(z^2) + z^{-1}F_0(z^2). \quad (4.52)$$

In the Type 2 polyphase matrix for two filters, the delay z^{-1} goes with 0:

$$\widehat{X}(z) = [z^{-1} \ 1] \begin{bmatrix} F_{0,0}(z^2) & F_{1,0}(z^2) \\ F_{0,1}(z^2) & F_{1,1}(z^2) \end{bmatrix} \begin{bmatrix} V_0(z^2) \\ V_1(z^2) \end{bmatrix}. \quad (4.53)$$

We just reversed $[1 \ z^{-1}]$ into $[z^{-1} \ 1]$, and *reversed the rows of the matrix*. The product is exactly the same. The only difference in the block form is the new position of the delay, now in the upper channel:



We will follow this arrangement: Type 2 for synthesis and Type 1 for analysis. The main point is that upsampling follows filtering. The phases of the filters are shorter than the filters themselves. Those subfilters operate simultaneously to produce separate phases of the outputs. Then upsampling with delay assembles \hat{x} . We draw an M -channel synthesis bank in both Type 1 and Type 2 polyphase form:



The M polyphase components of a single filter H are

$$\text{Type 1: } H(z) = G_0(z^M) + z^{-1}G_1(z^M) + \cdots + z^{-(M-1)}G_{M-1}(z^M)$$

$$\text{Type 2: } H(z) = F_{M-1}(z^M) + z^{-1}F_{M-2}(z^M) + \cdots + z^{-(M-1)}F_0(z^M)$$

When the bank has M filters, each with M phases, we have an M by M matrix. This is the polyphase matrix containing M^2 time-invariant filters.

Perfect Reconstruction

One reason for introducing the polyphase form is efficiency. The analysis bank and synthesis bank are faster, when $(\downarrow 2)$ and $(\uparrow 2)$ are moved outside. The even and odd subfilters that appear inside are about 50% shorter. So they can be executed more quickly.

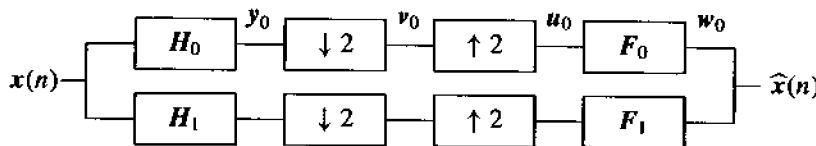
The other reason for polyphase is to simplify the theory. Underlying the whole book is the goal of perfect reconstruction. We have to *connect* the synthesis bank to the analysis bank. One must be the inverse of the other, if the signal \hat{x} is to agree with the input x . When pure filters connect to pure filters, the products are pure filters — and the z -transforms tell all.

Since the analysis bank is represented by a matrix $H_p(z)$, and the synthesis bank is represented by $F_p(z)$, we hope very much that *inverse filter banks* are associated with *inverse matrices*:

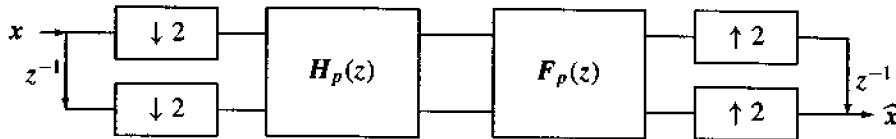
Perfect reconstruction should mean that $F_p(z) = H_p^{-1}(z)$.

We note that delays are allowed and expected. The output signal may be $\hat{x}(n) = x(n - l)$. Then the system delay is l . In the z -domain this is $F_p H_p = z^{-(l-1)/2} I$.

The direct connection of analysis bank to synthesis bank has the decimators and expanders inside. This produces the standard QMF bank:



But the standard order is inefficient. The polyphase order is much better, with $(\downarrow 2)$ and $(\uparrow 2)$ moved to the outside. We draw the polyphase form with Type 2 leading to $F_p(z)$. The final delay is moved from channel 2 in the Type 1 form (row 2 of G_p) to channel 1 in the Type 2 form (row 1 of F_p):



Polyphase is simpler and better because H_p and F_p are now side by side. Those are matrices coming from pure time-invariant filters — the even and odd phases of two analysis filters and two synthesis filters. In between would come compression or transmission. With nothing happening between, $F_p H_p$ disappears into I .

Example 4.8. The simplest analysis bank has $H_0(z) = 1$ and $H_1(z) = z^{-1}$. Their polyphase components are $H_{00} = 1$ and $H_{01} = 0$, then $H_{10} = 0$ and $H_{11} = 1$. The polyphase matrix is $H_p = I$.

The corresponding synthesis bank has $G_0(z) = z^{-1}$ and $G_1(z) = 1$ in the two channels. The Type 1 polyphase components of z^{-1} are $G_{00} = 0$ and $G_{01} = 1$, in the first column (not row!) of G_p . Then $G_{10} = 1$ and $G_{11} = 0$ go into the second column. These enter the synthesis matrix G_p , and in this example the transposing has no effect:

$$G_p = \begin{bmatrix} G_{00} & G_{10} \\ G_{01} & G_{11} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (4.54)$$

When we account for the delays in the lower channel, the final output is $\hat{x}(n) = x(n - 1)$. The QMF bank is a simple delay chain, with perfect reconstruction.

Now use Type 2 for synthesis. This reverses the rows of G_p to give F_p . Thus $F_p = I$. Starting from the two synthesis filters, z^{-1} has Type 2 components $F_{00} = 1$ and $F_{01} = 0$. The second channel yields $F_{10} = 0$ and $F_{11} = 1$ in the second column (not row!) of F_p :

$$F_p = \begin{bmatrix} F_{00} & F_{10} \\ F_{01} & F_{11} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (4.55)$$

You see the advantage of the second form, which is the Type 2 form $F_p(z)$. The perfect reconstruction from this delay chain is reflected in

$$F_p(z) H_p(z) = I. \quad (4.56)$$

That is the important equation in this section. The example itself is a terrible QMF bank. It has delays but no genuine filters. However the conclusion remains correct when $H_p(z)$ and $F_p(z)$ are polynomials in z^{-1} , from FIR filters. That is the whole point—that useful matrices can be inverses of each other and *both can be polynomials*.

The direct and polyphase forms of a QMF bank are externally equivalent. The observer of x and \hat{x} does not notice a difference. But the efficiency is improved and the theory is simplified. The theory of perfect reconstruction is a perfect matrix equation:

Theorem 4.5 *QMF banks give perfect reconstruction when F_p and H_p are inverses:*

$$F_p(z) H_p(z) = I \quad \text{or} \quad z^{-L} I.$$

$H_p(z)$ is Type 1, for analysis, and $F_p(z)$ is Type 2 transposed, for synthesis.

Example 4.9. In the average-difference filter bank all filters have *two taps*. Their even and odd phases have only one tap. Therefore the polyphase matrices are constant. But they are not diagonal, as they were in the simple delay chain. The analysis polyphase matrix is

$$H_p(z) = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The synthesis matrices are transposed as always. F_p has “even” in the lower row:

$$G_p(z) = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad F_p(z) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad F_p H_p = I.$$

Problem Set 4.4

- Express $X(z) = X_0(z^2) + z^{-1}X_1(z^2)$ as a sum $x = (\uparrow 2)x_0 + (__)$ in the time domain. What are the coefficients $x_0(n)$ and $x_1(n)$ in the polyphase decomposition (Type 1) in terms of the original $x(n)$?

The next three exercises are the synthesis bank equivalents of the time-domain analysis matrices H_d , H_p , H_b in Section 4.3.

- With filter coefficients $c(0), \dots, c(3)$ in G_0 and $d(0), \dots, d(3)$ in G_1 , write down the infinite time-domain matrix G_d for the synthesis bank. The analysis bank had the lowpass $(\downarrow 2)C$ above the highpass $(\downarrow 2)D$. The synthesis bank will have $G_0(\uparrow 2)$ and $G_1(\uparrow 2)$ side by side. Each of these parts looks like equation (4.49).
- Rearrange the rows of the direct matrix G_d in the previous exercise to give the polyphase matrix G_p in the time domain. This should be a 2 by 2 matrix with time-invariant filters C_{even} , C_{odd} , D_{even} , D_{odd} as the blocks ($C = G_0$ and $D = G_1$).
- Rearrange the columns of the direct matrix G_d to produce an infinite matrix G_b of 2 by 2 blocks.

5. Draw a delay chain with M channels. Then $H_0 = 1, H_1 = z^{-1}, \dots, H_{M-1} = z^{-(M-1)}$ leads to what matrix $\mathbf{H}_p(z)$? Choose the synthesis delays to reconstruct $\hat{x}(n) = x(n - (M - 1))$. Create the Type 1 matrix $\mathbf{G}_p(z)$ and the Type 2 matrix $\mathbf{F}_p(z)$. Check $\mathbf{F}_p \mathbf{H}_p = \mathbf{I}$.

6. Establish the relation between $\mathbf{G}_m(z)$ (modulation) and $\mathbf{G}_p(z)$ (polyphase Type 1).

7. If $\mathbf{G}_m(z) \mathbf{H}_m(z) = \begin{bmatrix} 2z^{-l} & 0 \\ 0 & -2z^{-l} \end{bmatrix}$ show that $\mathbf{G}_p(z) \mathbf{H}_p(z) = \begin{bmatrix} 0 & z^{-L} \\ z^{-L} & 0 \end{bmatrix}$, with $l = 2L + 1$. The matrices \mathbf{G}_m and \mathbf{G}_p include a transpose. Then \mathbf{F}_p includes a row exchange for Type 2, and $\mathbf{F}_p(z) \mathbf{H}_p(z) = z^{-L} \mathbf{I}$.

8. Find $F_0(z)$ and $F_1(z)$ for synthesis from the analysis polyphase matrix:

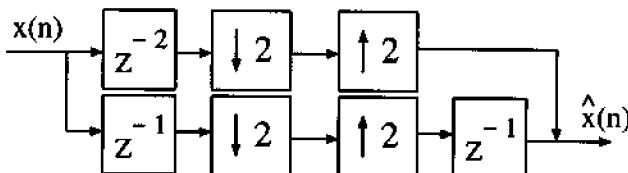
$$\text{a. } \mathbf{H}_p(z) = \begin{bmatrix} 2 - 4z^{-1} & 1 - z^{-1} \\ 3 + z^{-1} + 2z^{-2} & 1 \end{bmatrix}$$

$$\text{b. } \mathbf{H}_p(z) = \begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix} \begin{bmatrix} z^{-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}$$

$$\text{c. } \mathbf{H}_p(z) = \begin{bmatrix} \frac{1+z^{-1}}{5-z^{-1}} & 1 \\ \frac{1}{2-z^{-1}} & z^{-3} \end{bmatrix}$$

$$\text{d. } \mathbf{H}_p(z) = \begin{bmatrix} \frac{1+2z^{-1}}{2+z^{-1}} & 0 \\ 0 & \frac{1-3z^{-1}}{3-z^{-1}} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

9. Find the polyphase matrices $\mathbf{H}_p(z)$ and $\mathbf{F}_p(z)$. Is this system PR?



4.5 Lattice Structure

A filter bank is represented by its polyphase matrix. When we know the filters, we know the matrix $\mathbf{H}_p(z)$. In the opposite direction, we can choose a suitable $\mathbf{H}_p(z)$ —often as a product of simple matrices. Then the corresponding filter bank is a “lattice” of simple filters, easy to implement.

Here is a class of polyphase matrices to use as examples. They are linear in z^{-1} . One factor has the coefficients 1, 1 and 1, -1 from the Haar filter bank—averages and differences. Another factor has coefficients a, b, c, d that we are free to choose—as long as the matrix is invertible. Between those factors is a diagonal matrix with a delay in the second channel:

$$\mathbf{H}_p(z) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & z^{-1} \\ z^{-1} & 1 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad (4.57)$$

$$\mathbf{H}_p^{-1}(z) = \frac{1}{2} \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} \begin{bmatrix} 1 & z \\ z & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \quad (4.58)$$

First point: \mathbf{H}_p^{-1} contains an advance (denoted by z). No problem. Good causal polyphase matrices have anticausal inverses, as this one has. The main point is that \mathbf{H}_p^{-1} is a polynomial (FIR

not IIR). Add one delay to the whole synthesis bank (multiply by z^{-1}) and it becomes causal:

$$z^{-1} \mathbf{H}_p^{-1}(z) = \frac{1}{2} \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} \begin{bmatrix} z^{-1} & \\ & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \quad (4.59)$$

When you multiply out the factors in $\mathbf{H}_p(z)$, you see the possibilities:

$$\mathbf{H}_p(z) = \begin{bmatrix} a + cz^{-1} & b + dz^{-1} \\ a - cz^{-1} & b - dz^{-1} \end{bmatrix}. \quad (4.60)$$

The top row holds the even and odd phases of the lowpass filter C :

$$C(z) = a + bz^{-1} + cz^{-2} + dz^{-3} \text{ has phases } a + cz^{-1} \text{ and } b + dz^{-1}.$$

The second row is the polyphase form of the highpass filter D :

$$D(z) = a + bz^{-1} - cz^{-2} - dz^{-3} \text{ has phases } a - cz^{-1} \text{ and } b - dz^{-1}.$$

We design these 4-tap filters by choosing a, b, c, d . They can have *linear phase or orthogonality*. Our 2 by 2 matrix can be symmetric or it can be orthogonal:

Linear phase filters (symmetric-antisymmetric). Choose $a = d$ and $b = c$.

The lowpass filter C has coefficients a, b, b, a . The highpass filter D is antisymmetric, with coefficients $a, b, -b, -a$. For the polyphase matrix, just substitute $d = a$ and $c = b$:

$$\text{Linear phase} \quad \mathbf{H}_p(z) = \begin{bmatrix} a + bz^{-1} & b + az^{-1} \\ a - bz^{-1} & b - az^{-1} \end{bmatrix}. \quad (4.61)$$

How do you recognize the symmetry of C (or H_0) from its phases in the top row? One phase is the *flip* of the other phase. Then the whole filter a, b, b, a is the flip of itself, which makes it symmetric.

The synthesis filters are also linear phase. The underlying reason is that the inverse of a symmetric matrix is symmetric. Now change to orthogonal.

Orthogonal filter bank Choose $d = a$ and $c = -b$ and normalize.

The second row $[c \ d] = [-b \ a]$ becomes orthogonal to the first row $[a \ b]$. Both rows and both columns have length $\sqrt{a^2 + b^2}$. Each of the three factors in $\mathbf{H}_p(z)$ is a unitary matrix, after dividing by that length. The inverses come directly from the conjugate transposes:

$$\begin{aligned} \begin{bmatrix} a & b \\ -b & a \end{bmatrix}^{-1} &= \frac{1}{a^2 + b^2} \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \\ \begin{bmatrix} 1 & z^{-1} \end{bmatrix}^{-1} &= \begin{bmatrix} 1 & \bar{z}^{-1} \end{bmatrix}^T \quad \text{if } |z| = 1. \end{aligned}$$

We use the word *unitary* rather than *orthogonal* because z is complex. The inverse of a unitary matrix U is \bar{U}^T . The first matrix is real, so conjugating had no effect. The second is diagonal, so transposing had no effect. We always do both.

Notice that $|z| = 1$. The inverse of $z = e^{-i\omega}$ equals the conjugate: $z^{-1} = e^{i\omega} = \bar{z}$. Section 5.1 will introduce the word *paraunitary* for $\mathbf{H}_p(z)$, when it is a unitary matrix on the circle $|z| = 1$. Our present example is paraunitary.

Multiplying the three matrix factors gives $H_p(z)$, with $d = a$ and $c = -b$:

$$\text{Paraunitary polyphase matrix } H_p(z) = \frac{1}{2} \begin{bmatrix} a - bz^{-1} & b + az^{-1} \\ a + bz^{-1} & b - az^{-1} \end{bmatrix}. \quad (4.62)$$

The impulse response shows the difference between linear phase (earlier) and orthogonal (now). The lowpass $C(z)$ comes from the first row, where a and $-b$ give the even coefficients. The coefficients in $b + az^{-1}$ go with the odd powers z^{-1} and z^{-3} :

$$C(z) = a + bz^{-1} - bz^{-2} + az^{-3}.$$

The highpass response from the second row is

$$D(z) = a + bz^{-1} + bz^{-2} - az^{-3}.$$

We lost symmetry and gained orthogonality. One part of orthogonality is that $(a, b, -b, a)$ is perpendicular to $(a, b, b, -a)$. But this is not all. Another part is that $(a, b, -b, a, 0, 0)$ is perpendicular to $(0, 0, a, b, b, -a)$. We have to consider those *double shifts*, because they enter the time-domain matrix — two filters C and D with downsampling:

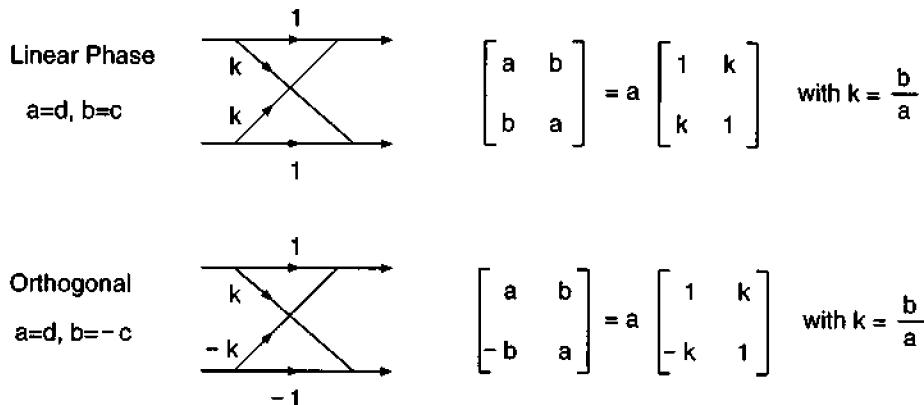
$$H = \begin{bmatrix} (\downarrow 2)C \\ (\downarrow 2)D \end{bmatrix} = \frac{1}{2} \begin{bmatrix} a & -b & b & a & & & \\ & a & -b & b & a & & \\ -a & b & b & a & & & \\ & -a & b & b & a & & \\ & & & & & \ddots & \\ & & & & & & \ddots \end{bmatrix}. \quad (4.63)$$

Linear phase and also orthogonal: Not possible for length greater than 2.

Later we prove this fact in general. Linear phase FIR pairs that are also orthogonal can have at most two nonzero coefficients in each filter. They are just variations on the average-difference pair. This is a one-step improvement over single filters, which can have only *one* nonzero coefficient. An allpass FIR filter is a delay, with one term $C(z) = cz^{-k}$.

Lattice Structures

The orthogonal bank will now be more general than this example. It will have more coefficients. But $H_p(z)$ can still be produced from simple factors. They will be *cascades* of constant matrices and diagonal matrices. The filter bank has a highly efficient *lattice structure*. We can see already the form of our a, b, c, d factor in the two important cases of linear phase and orthogonality:



Each lattice involves only *one multiplication* when implemented properly (Problem 7). There is also a single overall multiplying factor, collected from all factors in the cascade. The numbers k can be design parameters. For any k 's, a cascade of linear phase filters is linear phase. A cascade of orthogonal filters (including $-k$) is orthogonal. We now study a general lattice built from simple orthogonal factors.

Note. The lattice structure gives long orthogonal filters very easily. They can be good filters. But the k 's enter $H_p(z)$ in a complicated nonlinear way. For the *design* step, where filter characteristics are optimized, most engineers choose simpler parameters like the coefficients $h(n)$.

For the *implementation* step, the lattice has an important advantage. Often the coefficients $h(n)$ will be “quantized”—real numbers are replaced by binary numbers (finitely many digits). In general, this destroys orthogonality. But orthogonality is not lost when implemented as a lattice of simple filters. Each orthogonal factor is determined by a real number k (or by an angle θ). When k or θ is quantized, the factor is not exactly correct—but it is still orthogonal. Therefore the whole filter bank remains strictly orthogonal.

We now introduce a product of *rotation matrices* and *delay matrices*. The rotation matrices are constant, exactly as in our examples:

$$R = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} = \cos \theta \begin{bmatrix} 1 & k \\ -k & 1 \end{bmatrix}. \quad (4.64)$$

This matrix gives clockwise rotation through the angle θ . When $\cos \theta$ is factored out, it leaves the number $k = \tan \theta$. We will think of the rotation matrix (orthogonal matrix) in terms of θ , but we implement it with k to save multiplications. The delay matrices are used to delay the second channel:

$$\Lambda(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \quad \text{has determinant } z^{-1}.$$

It is convenient to have an extra factor of -1 in the lower channel. The matrix $\text{diag}(1, -1)$, which is also $\Lambda(-1)$, accomplishes this. Then the product of $\ell+1$ rotations separated by ℓ delays is the polyphase matrix for the whole lattice:

$$H_p(z) = \Lambda(-1) R_\ell \Lambda(z) R_{\ell-1} \Lambda(z) \cdots R_1 \Lambda(z) R_0. \quad (4.65)$$

The rotation R_ℓ is through an angle θ_ℓ . The rotation R_0 is through an angle θ_0 . The determinant is $-z^{-\ell}$, because of the delays and the sign change from $\text{diag}(1, -1)$. Without the delays in between, the product would be a single rotation through the total angle $\sum \theta_i$. With the ℓ delays, we have something much more important. It is essentially the most general two-channel orthonormal filter bank with ℓ delays.

$H_p(z)$ is the polyphase matrix of an orthonormal filter bank, because each factor is unitary. We show this analysis bank in its efficient form, with the downsampling operators before the filters. R_0 comes first in the structure because it is the right-hand factor in $H_p(z)$.

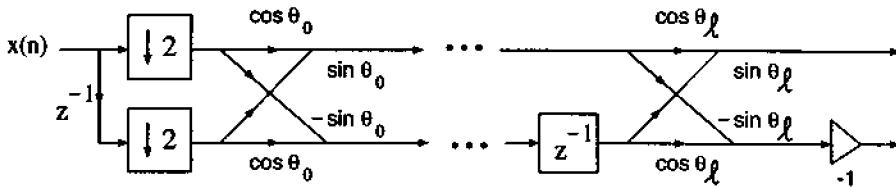


Figure 4.4: The lattice structure for $H_p(z) = \Lambda(-1) R_\ell \Lambda(z) \dots R_1 \Lambda(z) R_0$.

The factors $\cos \theta$ that are removed give a multiplication by $\cos \theta_\ell \dots \cos \theta_0$ at the end. Note that Haar's filter bank comes from *one* rotation with angle $\frac{\pi}{4}$:

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ \sin \theta & -\cos \theta \end{bmatrix} \quad \text{with } \theta = \frac{\pi}{4}.$$

Actually this is a picture of any polyphase matrix $H_p(z)$ at the particular value $z = 1$. The matrix $H_p(1)$ corresponds to zero frequency, because $e^0 = 1$. *The response to direct current is always the Haar matrix, when the lowpass filter has a zero at $\omega = \pi$.*

Theorem 4.6 If $H_0 = 0$ at $\omega = \pi$ (which is $z = -1$), then the polyphase matrix has

$$H_p(1) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

For an orthonormal filter bank the angles in the lattice structure add to $\frac{\pi}{4}$.

Proof: H_0 has a zero at $\omega = \pi$ when the sum of odd-numbered coefficients equals the sum of even-numbered coefficients: $h(0) - h(1) + h(2) - \dots = 0$. This *first sum rule* means that at $z = 1$, the even phase equals the odd phase:

$$H_{00}(1) = h(0) + h(2) + \dots = h(1) + h(3) + \dots = H_{01}(1). \quad (4.66)$$

Row 1 of the polyphase matrix has equal entries at $z = 1$. Row 2 comes from the highpass filter H_1 . This has the opposite property $H_1 = 0$ at $\omega = 0$. The odd sum is *minus* the even sum. With the right signs, $H_p(1)$ is the Haar matrix in the Theorem.

Since this is exactly what we get in the product of rotations, when $z = 1$ and the delay matrices become I , the total rotation angle must be $\sum \theta_i = \frac{\pi}{4}$.

The 4-tap Daubechies filter in Section 5.2 has angles $\frac{\pi}{3}$ and $-\frac{\pi}{12}$. Those angles add to $\frac{\pi}{4}$. Looking back at the start of this section, we realize that Haar followed by one rotation is actually

useless. That second rotation angle must be zero! The sum rule in the top row of the lowpass filter becomes $a + b = -b + a$, which forces $b = 0$. The good angles are $\frac{\pi}{3}$ and $-\frac{\pi}{12}$, not $\frac{\pi}{4}$ and 0.

The Synthesis Lattice

The synthesis bank inverts the analysis bank. To invert the lattice, reverse the order of rotations. The inverse $\Lambda(-1) \mathbf{R}_\ell \Lambda(z) \dots \mathbf{R}_1 \Lambda(z) \mathbf{R}_0$ is the product of inverses in reverse order:

$$(\mathbf{H}_p(z))^{-1} = \mathbf{R}_0^T \Lambda^{-1}(z) \mathbf{R}_1^T \dots \Lambda^{-1}(z) \mathbf{R}_\ell^T \Lambda(-1). \quad (4.67)$$

The inverse of each rotation is the transpose: $\mathbf{R}^{-1} = \mathbf{R}^T$ = rotation through $-\theta$. These constant matrices are orthogonal. The inverse of a delay matrix is an *advance*:

$$\Lambda^{-1}(z) = \begin{bmatrix} 1 & 0 \\ 0 & z \end{bmatrix} \quad \text{and} \quad z^{-1} \Lambda^{-1}(z) = \begin{bmatrix} z^{-1} & 0 \\ 0 & 1 \end{bmatrix}$$

Since $\mathbf{H}_p(z)$ has ℓ delays, its inverse has ℓ advances. Multiplying each advance by z^{-1} puts ℓ delays into the upper channel. The synthesis half has *upsamplers last* (this is the polyphase form in Figure 4.5). An extra advantage is that we can add or remove rotations without losing

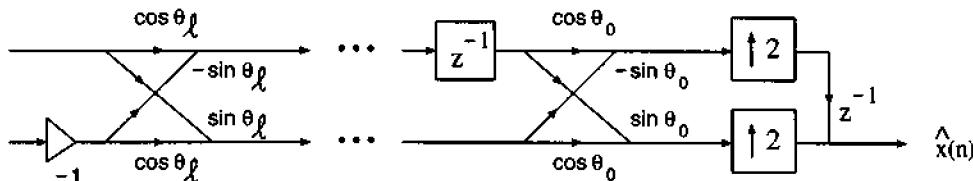


Figure 4.5: Orthogonal synthesis bank in lattice form.

orthogonality. In contrast, the alteration of one filter coefficient $h(n)$ is almost certain to destroy orthogonality and also perfect reconstruction.

What is the synthesis polyphase matrix?

Lattice Coefficients from Filter Coefficients

We now prove that any 2-channel orthonormal filter bank can be expressed in lattice form, with rotations and delays. Thus the lattice structure is “complete”. Starting with $\mathbf{H}_p(z)$ of degree ℓ , we find a rotation-delay that reduces the degree to $\ell - 1$:

$$\mathbf{H}_p(z) = \begin{bmatrix} \cos \theta_\ell & -\sin \theta_\ell \\ \sin \theta_\ell & \cos \theta_\ell \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \mathbf{H}_p^{(\ell-1)}(z). \quad (4.68)$$

The new matrix $\mathbf{H}_p^{(\ell-1)}(z)$ is still unitary for $|z| = 1$ and its determinant is $\pm z^{-(\ell-1)}$. So we reduce the degree again. After ℓ steps we reach $\mathbf{H}_p(0)$ whose determinant is ± 1 . It is unitary for $|z| = 1$ and the only matrices with this property are constants. After properly accounting for the matrix $\Lambda(-1) = \text{diag}(1, -1)$, we have the rotation angle θ_0 that completes the lattice.

Theorem 4.7 Every lowpass-highpass orthonormal filter bank has a polyphase matrix $\mathbf{H}_p(z)$ of the lattice form (4.65).

Proof. The step (4.68) requires an angle θ_ℓ such that

$$\begin{bmatrix} \cos \theta_\ell & \sin \theta_\ell \\ -\sin \theta_\ell & \cos \theta_\ell \end{bmatrix} \mathbf{H}_p(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \mathbf{H}_p^{(\ell-1)}(z). \quad (4.69)$$

On both sides, the second row must have **no constant terms**. Then we can factor z^{-1} from that row and the reduction succeeds — we get the next matrix $\mathbf{H}_p^{(\ell-1)}(z)$ and continue. If $\mathbf{H}_p(z) = \mathbf{h}_p(0) + \dots + \mathbf{h}_p(d)z^{-d}$, the constant term on the left side of the equation comes from $\mathbf{h}_p(0)$. The second row on that side must give

$$\begin{bmatrix} -\sin \theta_\ell & \cos \theta_\ell \end{bmatrix} \mathbf{h}_p(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}. \quad (4.70)$$

This row vector that produces zero must exist if $\mathbf{h}_p(0)$ is singular. The whole argument rests on the fact that $\mathbf{h}_p^T(d)\mathbf{h}_p(0)$ is the **zero matrix**. The key equation is $\bar{\mathbf{H}}_p^T \mathbf{H}_p = \mathbf{I}$:

$$[\mathbf{h}_p^T(0) + \dots + \mathbf{h}_p^T(d)z^d] [\mathbf{h}_p(0) + \dots + \mathbf{h}_p(d)z^{-d}] = \mathbf{I}. \quad (4.71)$$

The coefficient $\mathbf{h}_p^T(d)\mathbf{h}_p(0)$ of the highest order term must be zero. When two nonzero matrices multiply to give zero, $\mathbf{h}_p(0)$ and $\mathbf{h}_p(d)$ are both singular. There exists a vector $[-\sin \theta_\ell \ \cos \theta_\ell]$ that knocks out the constant term in the second row of (4.69). After ℓ steps we reach a final rotation \mathbf{R}_0 and the lattice is complete.

Lattices for Linear Phase Filters

For orthonormal filters, \mathbf{H}_0 and \mathbf{H}_1 have the same (even) length. One is symmetric and the other is antisymmetric. For linear phase filters, those statements are not necessarily true. The great variety of linear phase PR filter banks means a great variety of lattice structures. The equal-length case copies the orthonormal case, but with factors $\mathbf{S} = \begin{bmatrix} a & b \\ b & a \end{bmatrix}$ instead of $\mathbf{R} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$.

Theorem 4.8 *Every linear phase perfect reconstruction filter bank with equal (even) length filters has a lattice factorization*

$$\mathbf{H}_p(z) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{S}_L \Lambda(z) \mathbf{S}_{L-1} \Lambda(z) \cdots \mathbf{S}_1 \Lambda(z) \mathbf{S}_0. \quad (4.72)$$

The filter bank is a cascade of simple two-tap filters:

$$\Lambda(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \text{ and } \mathbf{S}_i = \begin{bmatrix} a_i & b_i \\ b_i & a_i \end{bmatrix} = a_i \begin{bmatrix} 1 & k_i \\ k_i & 1 \end{bmatrix}.$$

The proof is entirely parallel to the orthogonal case (Theorem 4.7). The implementation just has a sign change. The number of lattice sections is half the length of \mathbf{H}_0 (and \mathbf{H}_1). All factors a_i are collected into a single factor $a = \prod a_i$, for efficiency. And the two multiplications by k_i are further reduced to one multiplication (and three additions) in Problem 7.

What is not parallel is that linear phase allows further possibilities: unequal length and symmetric-antisymmetric pairs. Here are the rules for PR with linear phase:

Even length must differ by a multiple of 4: $H_0 = \text{symm}$ and $H_1 = \text{anti}$.

Odd length must *not* differ by a multiple of 4: $H_0 = \text{symm}$ and $H_1 = \text{symm}$.

An example of the second type is in the “Guide to the Book”, with lengths 5 and 3: $\frac{1}{8}(-1, 2, 6, 2, -1)$ in H_0 and $\frac{1}{4}(1, -2, 1)$. For symmetric filters whose lengths differ by 2, the (complete) lattice factorization is

$$H_p(z) = D \prod_{k=2}^L A_k(z) \text{ with } A_k(z) = \begin{bmatrix} 1 + z^{-1} & 1 \\ 1 + \beta_k z^{-1} + z^{-2} & 1 + z^{-1} \end{bmatrix} \begin{bmatrix} \alpha_k & 0 \\ 0 & 1 \end{bmatrix} \quad (4.73)$$

D is diagonal, and a practical system can have only $\beta_2 \neq 0$. This speeds up the implementation. The FBI 9/7 system is an example that requires only two lattice sections.

The general case has a similar (but longer) factorization. The proof is also longer. Our purpose has been to establish the main point: the efficiency and the availability of the lattice structure.

Perfect Reconstruction with FIR

We are interested above all in FIR systems. Then both H_p and F_p have a finite number of terms, from the finite number of filter coefficients. The crucial observation is that with well-chosen matrices, *the product of two polynomials can be a constant or a monomial*:

$$\begin{aligned} H_p^{-1}(z)H_p(z) &= I \\ F_p(z)H_p(z) &= z^{-L}I \end{aligned} \quad \text{are possible for matrix polynomials.}$$

In the scalar case, $1 + \beta z^{-1}$ is a polynomial but $1/(1 + \beta z^{-1})$ is not. The reciprocal of a polynomial scalar is not a polynomial. If an ordinary time-invariant filter is FIR, its inverse is IIR. The only scalar exceptions are trivial cases (delays). *The real exceptions are matrix polynomials — which are polyphase matrices of filter banks.* That is what this book is about.

The inverse of a matrix polynomial can be a matrix polynomial. We ask when.

Theorem 4.9 *An FIR analysis bank has an FIR synthesis bank that gives perfect reconstruction if and only if the determinant of $H_p(z)$ is a monomial cz^{-l} with $c \neq 0$.*

Proof. The entries of $(H_p(z))^{-1}$ are always cofactors divided by the determinant:

$$(H_p^{-1})_{ij} = \frac{(j, i) \text{ cofactor of } H_p(z)}{\text{determinant of } H_p(z)}.$$

The cofactor (the determinant without row j and column i) is certainly a polynomial. If the determinant is cz^{-l} (one term only!) then the division leaves a polynomial: the inverse is FIR. When this inverse $H_p^{-1}(z)$ is delayed by z^{-l} , it becomes causal. This is $F_p(z)$. It is the Type 2 polyphase matrix of the FIR synthesis bank.

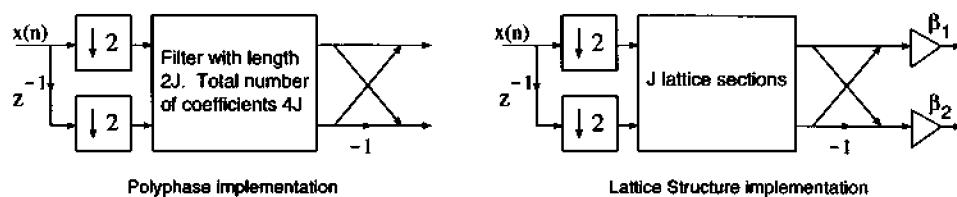
The design problem is to maintain a monomial determinant cz^{-l} while building an efficient filter bank. For an M by M matrix the cofactors of $H_p(z)$ may have high degrees. The analysis and synthesis filters generally have different lengths. The design problem is harder, but more is possible. Cosine-modulated filter banks are winners.

Efficiency of Lattice Structure

Consider a symmetric filter bank with length $N = 2J$. The number of multiplications per unit time is $4J$ using the direct form and $2J$ using polyphase. Additions are the same, with two extra at the 2-pt DFT matrix.

On the other hand, there are J lattice sections. Each section requires 2 multiplications and 2 additions. Counting β_1 and β_2 in the figure brings the total complexity to $2J$ multiplications and $2J + 2$ additions. The effective complexity (at the input rate) is J multiplications and $J + 1$ additions.

Problem 7 discusses an alternative that uses 1 multiplier and 3 adders per lattice section. In summary, the lattice complexity is approximately half of the polyphase complexity. The same is true for orthogonal filters.



Problem Set 4.5

1. Show that this matrix gives linear phase. Find a, b, c, d and $H_p^{-1}(z)$:

$$H_p(z) = \frac{1}{2} \begin{bmatrix} \cos \Theta + z^{-1} \sin \Theta & -\sin \Theta + z^{-1} \cos \Theta \\ \cos \Theta - z^{-1} \sin \Theta & -\sin \Theta - z^{-1} \cos \Theta \end{bmatrix}.$$

2. Suppose U and V are unitary matrices (constant but possibly complex). This means that $U^{-1} = U^T$ and $V^{-1} = V^T$. Show why their product UV is also unitary. Notice the key point: Inverses come in reverse order $V^{-1}U^{-1}$ and so do transposes.
3. For a 6-tap antisymmetric filter $a + bz^{-1} + cz^{-2} - cz^{-3} - bz^{-4} - az^{-5}$, show that the flip of one phase is minus the other phase. What if the number of taps is odd?
4. Redraw the lattice cascade with the downsampling operators moved to the right, after the butterfly filters. Be sure to change z^{-1} to z^{-2} (why?).
5. Find the product $H_p(z)$ when the rotation angles are $\theta_0 = 0$, $\theta_1 = \frac{\pi}{2}$, and $\theta_2 = 0$. Check the determinant of $H_p(z)$, remembering the $\ell = 2$ delays. This is an example in which $H_p(z)$ contains no terms in $z^{-\ell}$, although the determinant has degree ℓ . In the notation of Theorem 4.7, $\ell = 2$ but $d = 1$.

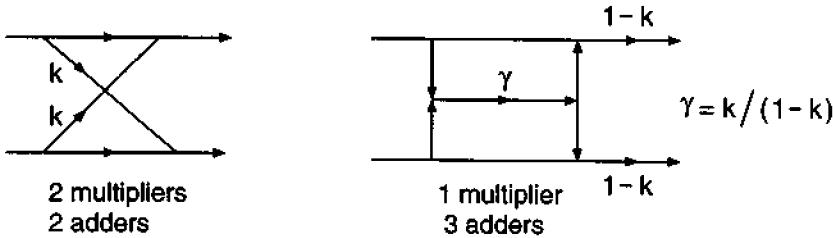
The correct definition of degree of $H_p(z)$ is the *Smith-McMillan degree = minimum number of delays to realize the system*. For orthonormal filter banks, this equals the degree ℓ of the determinant.

6. Find the polyphase matrix $H_p(z)$ for the pair $H_0(z) = 1$ and $H_1(z) = \frac{1}{2} - z^{-1} + \frac{1}{2}z^{-2}$ in Problem Set 4.3. Find the synthesis polyphase matrix $F_p(z)$ for $F_0(z) = \frac{1}{2} + z^{-1} + \frac{1}{2}z^{-2}$ and $F_1(z) = -1$. Remember to transpose and reverse rows for $F_p(z)$, and compute $F_p(z)H_p(z)$.

7. We can multiply by $S = \begin{bmatrix} a & b \\ b & a \end{bmatrix}$ with one multiplication by $c = \frac{a+b}{a-b}$ (and one by $\frac{a-b}{2}$ that can be collected with others at the end of the cascade):

$$S = \frac{a-b}{2} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} c & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

Write S with 3 adds using the figure. Find a “one multiplication per rotation R ”.



8. Find the polyphase matrix $H_p(z)$ for the analysis pair $H_0(z) = 1$ and $H_1(z) = \frac{1}{16}(-1 + 9z^{-2} - 16z^{-3} + 9z^{-4} - z^{-6})$. Find $F_p(z)$, transposed and row reversed, for the synthesis pair $F_0(z) = \frac{1}{16}(-1 + 9z^{-2} + 16z^{-3} + 9z^{-4} - z^{-6})$ and $F_1(z) = -1$. Verify that $F_p(z)H_p(z) = z^{-1}I$. Note $L = 1$ and $\ell = 3$.
9. What is the synthesis polyphase matrix F_p for the lattice structure in the last figure?
10. The rules for linear phase with PR were stated separately for even and odd lengths. Combine them into one rule for the degrees of $H_0(z)$ and $H_1(z)$: $N_0 + N_1 + 2$ must be a multiple of 4.
11. Change a given PR filter bank by choosing $\hat{H}_0(z) = H_0(z)$ and $\hat{H}_1(z) = z^{-2}H_1(z)$. What is the relation between $H_p(z)$ and $\hat{H}_p(z)$? Is this new filter bank PR?
12. Find the analysis filters and the relation between $H_0(z)$ and $H_1(z)$:
$$H_p(z) = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} 1 & -3 \\ -3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-2} \end{bmatrix} \begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix}$$
13. Let $H_p(z) = \begin{bmatrix} z^{-N} & 0 \\ \beta(z) & 1 \end{bmatrix}$, where $\beta(z)$ is a polynomial. Find $H_0(z)$ and $H_1(z)$. Find $F_p(z)$ for a PR system. What are the synthesis filters?
14. Let $H_p(z) = \prod_{k=1}^L \begin{bmatrix} z^{-1} & 0 \\ \beta_k(z) & 1 \end{bmatrix}$. Find $F_p(z)$ and all the filters for a PR system.
15. What is the lowpass orthogonal filter with $\theta_1 = \frac{\pi}{3}$, $\theta_2 = -\frac{\pi}{2}$, and $\theta_3 = \frac{\pi}{4}$?
16. What are the symmetric even-length analysis filters with $k_1 = 2$, $k_2 = 0$, $k_3 = -7$, and $k_4 = 5$? What are the PR synthesis filters?
17. If we impose orthogonality on the symmetric factors S_i in (4.72), show that the filters only have two nonzero coefficients. This yields another proof of Theorem 5.3 that symmetry prevents orthogonality.
18. Show that $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & k \\ k & 1 \end{bmatrix}$ is equivalent to $\begin{bmatrix} 1+k & 0 \\ 0 & 1-k \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. This form of the last two blocks in (4.72) reduces the lattice computation time.

Chapter 5

Orthogonal Filter Banks

5.1 Paraunitary Matrices

For an orthogonal matrix, the inverse is the transpose. When the matrix is 2 by 2, this imposes a tremendous constraint. Suppose we choose just one entry of the matrix, in the upper left corner. Our choice should not exceed 1 in absolute value, and we call it $\cos \theta$:

$$\mathbf{H} = \begin{bmatrix} \cos \theta & - \\ - & - \end{bmatrix}.$$

The other entries are almost completely determined by this choice. Below $\cos \theta$ we need $\sin \theta$ or $-\sin \theta$, to make the first column a unit vector. The second column must be a unit vector orthogonal to the first one, and we select

$$\mathbf{H} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}. \quad (5.1)$$

\mathbf{H} rotates every plane vector x by θ . The length is preserved: $\|\mathbf{H}x\| = \|x\|$.

This is not the only orthogonal matrix that starts with $\cos \theta$. We could multiply the second column or the second row by -1 . Then the rotation matrix in (5.1) becomes a *reflection* matrix; its determinant changes to -1 . If these are *complex* matrices, we could multiply any column and any row by numbers on the unit circle $|z| = 1$. This yields every unitary 2 by 2 matrix. Essentially, one entry determines the whole matrix. The inverse is the conjugate transpose: $\mathbf{H}^{-1} = \overline{\mathbf{H}}^T$.

An analysis bank is represented by an infinite matrix (in the time domain). But in the frequency domain or z -domain, the matrix is 2 by 2 (from $M = 2$ channels and $M = 2$ phases). This matrix depends on a parameter ω or z . Therefore we stretch the definition from unitary to *paraunitary*:

Definition 5.1 *The matrix $\mathbf{H}(z)$ is paraunitary if it is unitary for all $|z| = 1$:*

$$\mathbf{H}^T(e^{-j\omega})\mathbf{H}(e^{j\omega}) = \mathbf{I} \quad \text{for all } \omega. \quad (5.2)$$

This extends to all $z \neq 0$ by $\tilde{\mathbf{H}}(z) = \mathbf{H}^T(z^{-1})$. Then a paraunitary matrix has

$$\mathbf{H}^T(z^{-1})\mathbf{H}(z) = \tilde{\mathbf{H}}(z)\mathbf{H}(z) = \mathbf{I} \quad \text{for all } z. \quad (5.3)$$

When the coefficients $\mathbf{h}(k)$ are complex, they are conjugated in $\tilde{\mathbf{H}}(z)$.

The matrix \mathbf{H} need not be 2 by 2. If it is 1 by 1, then $|\mathbf{H}(e^{j\omega})| = 1$. The corresponding filter is *allpass*. The best allpass examples are ratios of polynomials coming from IIR filters—since only trivial polynomials z^{-l} can have $|\mathbf{H}(e^{j\omega})| = 1$.

If $\mathbf{H}(z)$ is $M \times M$, it could come from an M -channel filter bank. It might be the polyphase matrix $\mathbf{H}_p(z)$ or the modulation matrix $\mathbf{H}_m(z)$ (divided by $\sqrt{2}$). We will show that the filter bank is orthogonal if these matrices are paraunitary. That is the important connection for this book.

Equation (5.2) gives the inverse matrix by transposing and conjugating the original. The synthesis bank comes by “reversing” the analysis bank. Note that for a square matrix, $\mathbf{H}(z)$ is paraunitary when $\mathbf{H}^{-1}(z)$ and $\mathbf{H}^T(z)$ and $\tilde{\mathbf{H}}(z)$ are paraunitary. And notice especially what equation (5.3) says about the *determinants* of these matrices:

$$(\det \tilde{\mathbf{H}}(z)) (\det \mathbf{H}(z)) = 1. \quad (5.4)$$

The determinants are 1 by 1 allpass!

Theorem 5.1 *If a square paraunitary matrix $\mathbf{H}(z)$ is FIR (= polynomial), then its determinant must be a delay:*

$$\det \mathbf{H}(z) = \pm z^{-l}. \quad (5.5)$$

The determinant of $\mathbf{H}_p(z)$ is also a delay for any *bi*-orthogonal filter bank. Orthogonality requires more; the polyphase matrix $\mathbf{H}_p(z)$ must be paraunitary.

If $\mathbf{H}(z)$ is rectangular, say M by r , then $\tilde{\mathbf{H}}(z)\mathbf{H}(z) = \mathbf{I}$ is still possible. The identity matrix is now r by r (and necessarily $r \leq M$, since the rank cannot exceed M). The matrix is still called paraunitary. There is no inverse matrix $\mathbf{H}^{-1}(z)$ in the rectangular case, but $\tilde{\mathbf{H}}(z)$ is a left-inverse. We hope and expect that $\mathbf{H}(z)$ can be completed to a square paraunitary matrix, by adding $M - r$ more columns. In the applications, we are starting with r filters and creating $M - r$ additional filters—while preserving orthogonality.

When $\tilde{\mathbf{H}}(z)\mathbf{H}(z) = d\mathbf{I}$ with $d > 0$, we could still use the word paraunitary. The chief example is the modulation matrix, which has $d = 2$. Some authors keep this flexibility.

Our chief interest is in the case $M = 2$. A 2 by 2 paraunitary matrix is essentially determined by one entry. For a paraunitary polyphase matrix, the even phase $H_{00}(z)$ of the lowpass filter essentially determines the whole orthogonal filter bank. For filter banks with four taps, there are two free parameters—which can be $c(0)$ and $c(2)$. The design problem is greatly reduced when the filter bank is orthogonal and its polyphase matrix is paraunitary.

2 by 2 Paraunitary Examples

The first example is $\mathbf{H}(z) = \mathbf{I}$. It corresponds to a “lazy filter bank” without filters. The polyphase matrix is $\mathbf{H}_p = \mathbf{I}$ when the filters are $H_0(z) = 1$ and $H_1(z) = z^{-1}$. The bank just splits the input vector \mathbf{x} into odd and even phases, without filtering it.

The next paraunitary matrix is $\mathbf{H}(z) = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. This is still constant. It is the polyphase matrix of the Haar average-difference filter bank. We move on to more interesting examples.

Suppose $\mathbf{R}_0, \dots, \mathbf{R}_l$ are constant rotation matrices as in equation (5.1). The rotation angles are $\theta_0, \dots, \theta_l$. If we multiply those matrices, we get a single rotation through the total angle $\sum \theta_i$. But if we introduce a diagonal matrix $\Lambda(z) = \text{diag}(1, z^{-1})$ between those rotations, we get something much more general and important:

$$\mathbf{H}(z) = \mathbf{R}_l \Lambda(z) \mathbf{R}_{l-1} \Lambda(z) \cdots \mathbf{R}_1 \Lambda(z) \mathbf{R}_0. \quad (5.6)$$

This matrix is paraunitary, because it is the product of paraunitary factors. Its determinant is z^{-l} (the product of determinants). The filter bank can be realized as a lattice structure involving l delays. Section 4.5 proved that *this is the most general 2 by 2 paraunitary matrix of degree l*. The only subtle point is the meaning of the word “degree.” Here are two small examples to make that point, both with $R_0 = R_2 = I$. Then $H(z)$ is $\Lambda(z)R_1\Lambda(z)$. The only difference is in R_1 :

$$\begin{aligned} H(z) &= \begin{bmatrix} 1 & z^{-1} \\ -z^{-1} & 1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & z^{-1} \\ z^{-1} & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & z^{-1} \\ -z^{-1} & z^{-2} \end{bmatrix} \\ H(z) &= \begin{bmatrix} 1 & z^{-1} \\ -z^{-1} & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & z^{-1} \\ z^{-1} & 1 \end{bmatrix} = \begin{bmatrix} 0 & z^{-1} \\ -z^{-1} & 0 \end{bmatrix} \end{aligned}$$

Both have determinant z^{-2} . Both have degree $l = 2!$ But the power z^{-2} , which appears in the first, does not appear in the second. The correct definition of degree is the *Smith-McMillan degree* — the minimum number of delays required to realize the system. For paraunitary systems, but not for all systems, the degree is revealed by the determinant.

For lowpass-highpass filters with four taps, the even and odd phases have two terms each. If $H(z)$ is paraunitary, it fits the general form (5.6) with $l = 1$. There are two parameters θ_0 and θ_1 to be chosen. Again we have two design parameters, for 4-tap orthogonal filters, but this time they are angles.

This chapter constructs a family of *maxflat filters*. Then Chapter 6 shows that these filters give the *Daubechies wavelets*. We mention the connection now, so you will attach importance to maxflat filters. They don’t have a sharp transition band — they don’t have the quickest transition from passband to stopband — but they produce the most vanishing moments.

Problem Set 5.1

1. Suppose $H(z) = h(0) + h(1)z^{-1} + \dots + h(N)z^{-N}$ is paraunitary with $N > 0$. Show that $h(0)^T h(N) = \text{zero matrix}$. Deduce that $h(0)$ and $h(N)$ are both singular matrices.
2. Multiply out $H(z) = R_1\Lambda(z)R_0$ for angles θ_1 and θ_0 . If this is a polyphase matrix, what filter coefficients $d(k)$ come from the even and odd phases in the lower row of $H(z)$? If that is a highpass filter, with $D = 0$ at $\omega = 0$ (which is $z = 1$), show that $\theta_1 + \theta_0 = -\frac{\pi}{4} + n\pi$. (Haar has $\theta_0 = -\frac{\pi}{4}$ and $\theta_1 = 0$.)
3. Complete this polynomial matrix to have $\det H(z) \equiv 1$. Is it paraunitary?

$$H(z) = \begin{bmatrix} 1 + z^{-1} & z^{-1} \\ \underline{\quad} & \underline{\quad} \end{bmatrix}.$$

4. When can a row be the first row of a paraunitary matrix?
5. Show that $H = I - 2vv^T$ is a unitary matrix, where $v^T v = 1$. Compute H^{-1} and the determinant of H . Construct an example.
6. Show that $H = I - vv^T + z^{-1}vv^T$ is a paraunitary matrix for a unit-norm vector v . Compute its inverse and determinant. The factorization of $H_p(z)$ using these Householder matrices is in [V].

5.2 Orthonormal Filter Banks

This section brings together the requirements for an orthonormal filter bank. We will see those requirements in the *time domain* and the *polyphase domain* and the *modulation domain*. These requirements are conditions on the filter coefficients $c(k)$ and $d(k)$. Then equation (5.19) indicates a simple choice of the d 's coming from the c 's. If the lowpass filter meets the orthogonality requirements, it is easy to construct a highpass filter to go with it.

The discussion is in terms of a 2-channel FIR filter bank, $M = 2$. But the conditions extend immediately to any M . *The polyphase matrix and the modulation matrix must be paraunitary.* In the M -channel case, the lowpass filter does not immediately determine the $M - 1$ remaining filters (which are bandpass). There is some freedom in their construction, and we come back in Chapter 9 to $M > 2$.

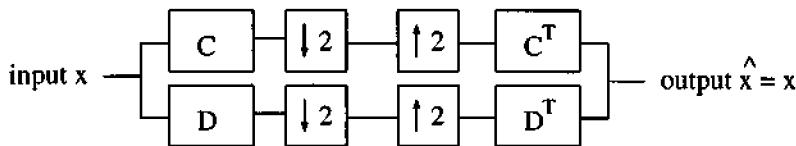


Figure 5.1: An orthogonal filter bank has synthesis bank = transpose of analysis bank.

Figure 5.1 shows the structure of an orthogonal filter bank. We intend to achieve $\hat{x} = x$, with synthesis filters C^T and D^T that are time-reversals of the analysis filters:

$$\tilde{C} = C^T \quad \text{and} \quad \tilde{c}(n) = c(-n) \quad (5.7)$$

$$\tilde{D} = D^T \quad \text{and} \quad \tilde{d}(n) = d(-n) \quad (5.8)$$

As it stands, \tilde{C} and \tilde{D} are anticausal. At the end we make them causal by N delays. The output $\hat{x}(n)$ is equally delayed; it is $x(n - N)$. But the algebra is easiest for C^T and D^T with no delays.

This special structure imposes special conditions on the c 's and d 's for perfect reconstruction. We will call the requirements **Condition O** (for orthogonality). This section finds four equivalent forms: Condition O on the infinite matrix, on the lowpass coefficients, and on the polyphase and modulation matrices H_p and H_m . We look first at the infinite matrices in the time domain.

Time Domain: Condition O and the Alternating Flip

The key matrix in the time domain is H_I . It represents the direct form of the analysis bank, with downsampling. The lowpass part $L = (\downarrow 2)C$ comes above the highpass part $B = (\downarrow 2)D$. We display this infinite matrix for filters of length four:

$$H_I = \begin{bmatrix} L \\ B \end{bmatrix} = \begin{bmatrix} c(3) & c(2) & c(1) & c(0) & & & \\ & c(3) & c(2) & c(1) & c(0) & & \\ d(3) & d(2) & d(1) & d(0) & & \ddots & \\ & d(3) & d(2) & d(1) & d(0) & & \\ & & & & & \ddots & \\ & & & & & & \ddots \end{bmatrix}. \quad (5.9)$$

The shifts by 2 were created by downsampling, which removed the odd-numbered rows.

With orthogonality, the synthesis filters are to be time-reversals of the analysis filters. The infinite synthesis matrix contains the transposes of $(\downarrow 2)\mathbf{C}$ and $(\downarrow 2)\mathbf{D}$. Those transposes are $\mathbf{C}^T(\uparrow 2)$ and $\mathbf{D}^T(\uparrow 2)$, since upsampling is the transpose of downsampling:

$$\mathbf{H}_t^T = [\mathbf{L}^T \quad \mathbf{B}^T] = \begin{bmatrix} \mathbf{c}(3) & \mathbf{d}(3) \\ \mathbf{c}(2) & \mathbf{d}(2) \\ \mathbf{c}(1) & \mathbf{c}(3) & \mathbf{d}(1) & \mathbf{d}(3) \\ \mathbf{c}(0) & \mathbf{c}(2) & \mathbf{d}(0) & \mathbf{d}(2) \\ & & \mathbf{c}(1) & \cdot & \mathbf{d}(1) & \cdot \\ & & \mathbf{c}(0) & \cdot & \mathbf{d}(0) & \cdot \end{bmatrix}. \quad (5.10)$$

The shifts by 2 in the columns are created by upsampling, which removes every other column. We require $\hat{\mathbf{x}} = \mathbf{x}$. This means that $\mathbf{H}_t^T \mathbf{H}_t = \mathbf{I}$. The matrix \mathbf{H}_t is required to be an *orthogonal matrix*. Its columns are orthonormal and so are its rows: $\mathbf{H}_t \mathbf{H}_t^T = \mathbf{I}$. We can express this Condition O in matrix form, and in block form, and in coefficient form.

Condition O An orthogonal filter bank comes from an orthogonal matrix:

$$\mathbf{H}_t^T \mathbf{H}_t = \mathbf{I} \text{ and } \mathbf{H}_t \mathbf{H}_t^T = \mathbf{I}. \quad (5.11)$$

In block form this means that

$$[\mathbf{L}^T \quad \mathbf{B}^T] \begin{bmatrix} \mathbf{L} \\ \mathbf{B} \end{bmatrix} = \mathbf{L}^T \mathbf{L} + \mathbf{B}^T \mathbf{B} = \mathbf{I} \quad (5.12)$$

and

$$\begin{bmatrix} \mathbf{L} \\ \mathbf{B} \end{bmatrix} [\mathbf{L}^T \quad \mathbf{B}^T] = \begin{bmatrix} \mathbf{L}\mathbf{L}^T & \mathbf{L}\mathbf{B}^T \\ \mathbf{B}\mathbf{L}^T & \mathbf{B}\mathbf{B}^T \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}. \quad (5.13)$$

For the coefficients $\mathbf{c}(k)$ and $\mathbf{d}(k)$, equation (5.13) becomes *orthogonality to double shifts*:

$$\mathbf{L}\mathbf{L}^T = \mathbf{I}: \quad \sum \mathbf{c}(n)\mathbf{c}(n-2k) = \delta(k) \quad (5.14)$$

$$\mathbf{L}\mathbf{B}^T = \mathbf{0}: \quad \sum \mathbf{c}(n)\mathbf{d}(n-2k) = 0 \quad (5.15)$$

$$\mathbf{B}\mathbf{B}^T = \mathbf{I}: \quad \sum \mathbf{d}(n)\mathbf{d}(n-2k) = \delta(k). \quad (5.16)$$

Because of (5.14)–(5.16), we refer to Condition O as *double-shift orthogonality*. Its equivalent in the frequency domain is presented below. This double-shift orthogonality immediately rules out odd length filters! If the length is $N+1=5$, a shift by 4 gives an inner product that cannot be zero:

$$(\mathbf{c}(0), \mathbf{c}(1), \mathbf{c}(2), \mathbf{c}(3), \mathbf{c}(4)) \cdot (0, 0, 0, 0, \mathbf{c}(0)) = \mathbf{c}(0)\mathbf{c}(4) \neq 0.$$

N cannot be even (the filter length cannot be odd) because $\frac{N}{2}$ double shifts would give a shift by N — and the inner product $\mathbf{c}(0)\mathbf{c}(N)$ is not zero. So N is odd. The degree $2N$ of the halfband product filter \mathbf{P} is 2, 6, 10, ... The clearest examples have $N=3$ and $2N=6$.

Example. Condition O in (5.14) imposes two constraints on four coefficients:

$$\mathbf{c}(0)^2 + \mathbf{c}(1)^2 + \mathbf{c}(2)^2 + \mathbf{c}(3)^2 = 1 \text{ and } \mathbf{c}(0)\mathbf{c}(2) + \mathbf{c}(1)\mathbf{c}(3) = 0. \quad (5.17)$$

Equation (5.16) is an identical condition on the \mathbf{d} 's, from $\mathbf{B}\mathbf{B}^T = \mathbf{I}$. Equation (5.15) is the orthogonality of the rows of \mathbf{L} to the rows of \mathbf{B} . Those are the fundamental design constraints on an orthogonal filter bank.

The conditions on the c 's and d 's are independent, but something good happens. If the c 's satisfy equation (5.14), it is easy to choose the d 's. We display a choice of d 's that automatically gives orthogonality:

$$\begin{bmatrix} L \\ B \end{bmatrix} = \begin{bmatrix} c(3) & c(2) & c(1) & c(0) & & & \\ & c(3) & c(2) & c(1) & c(0) & & \\ -c(0) & c(1) & -c(2) & c(3) & & \ddots & \\ & -c(0) & c(1) & -c(2) & c(3) & & \\ & & & & & \ddots & \\ & & & & & & \ddots \end{bmatrix}. \quad (5.18)$$

This is the *alternating flip*. The c 's are reversed in order and alternated in sign, to produce the d 's. We start with lowpass coefficients $c(0), \dots, c(N)$, where N is odd. The highpass coefficients are

$$d(k) = (-1)^k c(N-k). \quad (5.19)$$

The essential point can be checked by eye, in the infinite matrix (5.18). If the top rows are orthogonal to each other, then all rows are orthogonal. The zero dot products in LB^T are:

$$\begin{aligned} -c(3)c(0) + c(2)c(1) - c(1)c(2) + c(0)c(3) &= 0 \\ -c(1)c(0) + c(0)c(1) &= 0. \end{aligned} \quad (5.20)$$

Furthermore, the d 's are orthonormal within themselves ($BB^T = I$) because the c 's are. Equations (5.17) hold for the d 's, when they are constructed by flipping the c 's. Minus signs cancel in $d(3)d(1)$ which is $(-c(0))(-c(2))$.

Our four-tap example has $N = 3$. The alternating flip gives $LB^T = 0$ for every odd N . The top rows of H , in (5.18) are always orthogonal to the bottom rows. Also (5.16) for the d 's follows from (5.14) for the c 's. Thus the alternating flip reduces orthogonality to (5.14):

Condition O on the coefficients: $\sum c(n)c(n-2k) = \delta(k)$.

Polyphase Domain: Condition O and the Alternating Flip

The polyphase form separates $(\downarrow 2)C$ and $(\downarrow 2)D$ into even phase and odd phase. In the time domain, we are rearranging the columns of the matrix H_t . All even columns come before the odd columns. The matrix goes into the 2 by 2 block form of Section 4.4, with time-invariant filters as the blocks:

$$H_{\text{block}} = \begin{bmatrix} C_{\text{even}} & C_{\text{odd}} \\ D_{\text{even}} & D_{\text{odd}} \end{bmatrix}.$$

In the z -domain, we are rearranging the response functions $C(z)$ and $D(z)$:

$$\sum c(k)z^{-k} = C_{\text{even}}(z^{-2}) + z^{-1}C_{\text{odd}}(z^{-2}). \quad (5.21)$$

Those phase responses are written $H_{00}(z)$ and $H_{01}(z)$ when $C(z)$ is $H_0(z)$. The highpass response $D(z) = H_1(z)$ decomposes in the same way. The polyphase matrix is

$$H_p(z) = \begin{bmatrix} C_{\text{even}}(z) & C_{\text{odd}}(z) \\ D_{\text{even}}(z) & D_{\text{odd}}(z) \end{bmatrix} = \begin{bmatrix} H_{00}(z) & H_{01}(z) \\ H_{10}(z) & H_{11}(z) \end{bmatrix}. \quad (5.22)$$

Now Condition O translates directly into a requirement on $\mathbf{H}_p(z)$. A filter bank is **orthogonal** when its polyphase matrix is **paraunitary**:

$$\mathbf{H}_p^T(e^{-j\omega})\mathbf{H}_p(e^{j\omega}) = \mathbf{I} \text{ for all } \omega \text{ and } \tilde{\mathbf{H}}_p(z)\mathbf{H}_p(z) = \mathbf{I} \text{ for all } z. \quad (5.23)$$

The inverse of $\mathbf{H}_p(z)$ is the synthesis polyphase matrix. The matrix and its inverse can be multiplied in either order — here is analysis times synthesis:

$$\begin{bmatrix} C_{\text{even}}(z) & C_{\text{odd}}(z) \\ D_{\text{even}}(z) & D_{\text{odd}}(z) \end{bmatrix} \begin{bmatrix} C_{\text{even}}(z^{-1}) & D_{\text{even}}(z^{-1}) \\ C_{\text{odd}}(z^{-1}) & D_{\text{odd}}(z^{-1}) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (5.24)$$

On the unit circle, where z^{-1} is \bar{z} , row 1 times column 1 becomes

$$|C_{\text{even}}(z)|^2 + |C_{\text{odd}}(z)|^2 = 1 \text{ when } |z| = 1. \quad (5.25)$$

This is the essence of Condition O in the polyphase domain.

For the example with four coefficients, it is helpful to multiply out equation (5.25):

$$\begin{aligned} (c(0) + c(2)z^{-1})(c(0) + c(2)z) + (c(1) + c(3)z^{-1})(c(1) + c(3)z) = \\ c(0)^2 + c(2)^2 + c(1)^2 + c(3)^2 + [c(0)c(2) + c(1)c(3)](z^{-1} + z) = 1. \end{aligned}$$

Thus (5.25) is equivalent to the explicit statement (5.17). The sum of squares is 1 and the dot product $c(0)c(2) + c(1)c(3)$ is zero.

The other multiplications in (5.24) give answers 1 or 0 in the same way. All these requirements on the d 's are automatically satisfied by the flip construction! We write that choice $d(k) = (-1)^k c(N-k)$ in the z -domain:

$$\sum d(k)z^{-k} = \sum c(N-k)(-z)^{-k} = \sum c(n)(-z)^{n-N}.$$

This relation between highpass and lowpass is an *alternating flip*:

$$D(z) = (-z)^{-N} C(-z^{-1}). \quad (5.26)$$

The number N is odd. Because of $(-z)^{-N}$, the even and odd phases in C are reversed to odd and even phases in D . We take $N = 3$ as typical. An alternating flip of $c(0), c(1), c(2), c(3)$ yields

$$\begin{aligned} D(z) &= c(3) - c(2)z^{-1} + c(1)z^{-2} - c(0)z^{-3} \\ &= (c(3) + c(1)z^{-2}) - z^{-1}(c(2) + c(0)z^{-2}). \end{aligned} \quad (5.27)$$

With this flip in row 2, the multiplication $\mathbf{H}_p(z)\mathbf{H}_p^T(z^{-1})$ becomes

$$\begin{bmatrix} c(0) + c(2)z^{-1} & c(1) + c(3)z^{-1} \\ c(3) + c(1)z^{-1} & -c(2) - c(0)z^{-1} \end{bmatrix} \begin{bmatrix} c(0) + c(2)z & c(3) + c(1)z \\ c(1) + c(3)z & -c(2) - c(0)z \end{bmatrix} = \mathbf{I}.$$

The off-diagonal entries of the product are *automatically* zero. The *alternating flip achieves $\mathbf{LB}^T = \mathbf{0}$* with or without orthogonality. The 2, 2 entry of the product is the same as the 1, 1 entry, when $|z| = 1$. The orthogonality requirement (5.25) makes the 1, 1 entry equal to 1.

Modulation Domain: Condition O and the Alternating Flip

The function that arises from modulating $C(z)$ is $C(-z)$. The frequency in $z = e^{j\omega}$ changes by π to produce $-z = e^{j(\omega+\pi)}$. This modulation takes $C(\omega)$ to $C(\omega + \pi)$. The frequency response graph is shifted by π .

Our goal is to relate $C(z)$ to $C(-z)$ and $D(z)$ to $D(-z)$ for an orthogonal filter bank. We already know Condition O on the coefficients:

$$c(0)^2 + c(1)^2 + c(2)^2 + c(3)^2 = 1 \quad \text{and} \quad c(0)c(2) + c(1)c(3) = 0.$$

Watch how 1 and 0 appear in $|C(z)|^2$. Stay on the unit circle where $\bar{z}^{-1} = z$:

$$\begin{aligned} & (c(0) + c(1)z^{-1} + c(2)z^{-2} + c(3)z^{-3})(c(0) + c(1)z + c(2)z^2 + c(3)z^3) = \\ & 1 + [c(0)c(1) + c(1)c(2) + c(2)c(3)](z^{-1} + z) + c(0)c(3)(z^{-3} + z^3). \end{aligned}$$

Now change z to $-z$. The odd powers z and z^3 change sign. When we add, those odd powers cancel:

$$|C(z)|^2 + |C(-z)|^2 = |C(\omega)|^2 + |C(\omega + \pi)|^2 = 2 \quad \text{for all } z = e^{j\omega}. \quad (5.28)$$

This is the *halfband condition*, also called the *Nyquist condition*: $|C(z)|^2$ is a halfband filter. Those filters we can design! In other words, the lowpass analysis filter $C(z)$ is a spectral factor of a halfband filter.

Condition O on $H_m(z)$. *The modulation matrix of an orthogonal filter bank is a paraunitary matrix times $\sqrt{2}$:*

$$H_m(z)\tilde{H}_m(z) = 2I \quad \text{for all } z. \quad (5.29)$$

On the circle $z = e^{j\omega}$, the modulation matrix is a unitary matrix times $\sqrt{2}$:

$$\begin{bmatrix} C(\omega) & C(\omega + \pi) \\ D(\omega) & D(\omega + \pi) \end{bmatrix} \begin{bmatrix} \overline{C(\omega)} & \overline{D(\omega)} \\ \overline{C(\omega + \pi)} & \overline{D(\omega + \pi)} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}. \quad (5.30)$$

The 1, 1 entry of this matrix product is $|C(\omega)|^2 + |C(\omega + \pi)|^2 = 2$ by (5.28). The other entries, when we multiply them out, follow immediately from (5.15) and (5.16). Thus Condition O on the coefficients is equivalent to Condition O on the modulation matrix H_m . It is also equivalent to Condition O on H_p . It is the statement that the analysis bank followed by its transpose gives perfect reconstruction. We summarize:

Theorem 5.2 *For an orthogonal filter bank the lowpass coefficients must satisfy Condition O (we give four equivalent forms).*

Matrix form	$LL^T = (\downarrow 2)CC^T(\uparrow 2) = I$
Coefficient form	$\sum c(n)c(n - 2k) = \delta(k)$
Polyphase form	$ C_{\text{even}}(e^{j\omega}) ^2 + C_{\text{odd}}(e^{j\omega}) ^2 = 1$
Modulation form	$ C(\omega) ^2 + C(\omega + \pi) ^2 = 2$

By an alternating flip, $LB^T = 0$ and $BB^T = I$ follow immediately from the lowpass part $LL^T = I$. The real problem is the design of the lowpass filter.

Symmetry Prevents Orthogonality

It is natural to want two good properties at once. Symmetry is good for the eye, and orthogonality is good for the algorithm. But the only filters with both properties are averaging filters (Haar filters) with two coefficients. We are forced to use IIR filters, or M channels, or multifilters with matrix coefficients (Section 7.5). Extra computation is unavoidable, because the next theorem rules out a perfect filter.

Theorem 5.3 *A symmetric orthogonal FIR filter can only have two nonzero coefficients.*

Proof. N is odd for orthogonality. The filter length must be even. With $N = 5$ a symmetric filter of length 6 has the form $(c(0), c(1), c(2), c(2), c(1), c(0))$. This vector must be orthogonal to all its double shifts. The inner product with its shift by *four* must be $2c(0)c(1) = 0$. Therefore $c(1) = 0$. Then the inner product with its shift by *two* gives $2c(0)c(2) = 0$. The only nonzero coefficient is $c(0)$ at both ends of the filter. This completes the proof.

By convention $c(0)$ is the first nonzero coefficient. Shift the filter if necessary to achieve this. The only symmetric orthogonal possibilities are $c = (1, 1)/\sqrt{2}$ and $(1, 0, 0, 1)/\sqrt{2}$ and $(1, 0, \dots, 0, 1)/\sqrt{2}$. Only the Haar coefficients $(1, 1)/\sqrt{2}$ will lead to orthogonal wavelets. Symmetry really conflicts with orthogonality.

A second proof observes that the odd phase is the flip of the even phase:

$$(c(0), c(4), c(2), c(2), c(4), c(0)) \text{ has } |C_{\text{even}}(z)|^2 = |C_{\text{odd}}(z)|^2.$$

Condition O is $|C_{\text{even}}(z)|^2 + |C_{\text{odd}}(z)|^2 = 1$. With symmetry this separates into $|C_{\text{even}}(z)|^2 = 1$ and $|C_{\text{odd}}(z)|^2 = 0$. The even phase is an allpass filter! So is the odd phase. But FIR allpass filters can only have one nonzero coefficient, which completes the second proof.

A third proof is based on the zeros of $C(z)$. This is in Section 5.4 below.

Problem Set 5.2

1. (a) Show that the alternating flip with odd N gives $\overline{D(\omega)} = -e^{iN\omega}C(\omega + \pi)$.
 (b) Then $\overline{D(\omega + \pi)} = e^{iN\omega}C(\omega)$. Verify that $C(\omega)\overline{D(\omega)} + C(\omega + \pi)\overline{D(\omega + \pi)} = 0$.
 (c) Also $|C(\omega)|^2 + |C(\omega + \pi)|^2 = 2$ implies that $|D(\omega)|^2 + |D(\omega + \pi)|^2 = 2$
2. For any four coefficients $h(0), \dots, h(3)$, verify that

$$|H_{\text{even}}(z)|^2 + |H_{\text{odd}}(z)|^2 = \frac{1}{2}(|H(z)|^2 + |H(-z)|^2).$$

Then Condition O for polyphase equals Condition O for modulation.

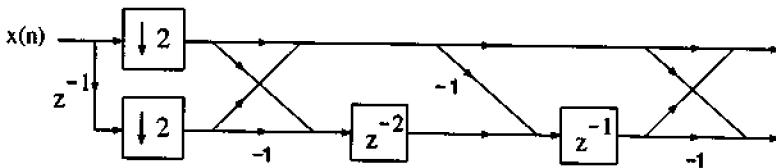
3. Find the flaw in this construction of the modulation matrix $H_m(z)$. Start with an arbitrary upper left entry $C(z)$. Complete the 2 by 2 matrix to be paraunitary (times $\sqrt{2}$). Then the filter bank is orthogonal.
4. Find d by an alternating flip of $c = (c(0), \dots, c(5))$. Verify equation (5.15) directly to show that c is double-shift orthogonal to d .
5. Verify that $c = \frac{1}{4\sqrt{2}}(1 + \sqrt{3}, 3 + \sqrt{3}, 3 - \sqrt{3}, 1 - \sqrt{3})$ satisfies Condition O (Daubechies).
6. If two lowpass filters C and H satisfy Condition O, does their product satisfy Condition O?
7. If two polyphase matrices $H_p(z)$ and $K_p(z)$ satisfy Condition O (they are paraunitary), does their product satisfy Condition O?

8. If two modulation matrices $H_m(z)$ and $K_m(z)$ satisfy Condition O, show that $H_m(z)K_m(z)/\sqrt{2}$ is also paraunitary. What is the lowpass filter in the product?
9. Why does orthogonality *require* an alternating flip between the lowpass filter C and the high-pass filter D ? Explain why the paraunitary matrix

$$H_p(z) = \begin{bmatrix} C_{\text{even}}(z) & C_{\text{odd}}(z) \\ D_{\text{even}}(z) & D_{\text{odd}}(z) \end{bmatrix} \text{ must have } \begin{aligned} |D_{\text{even}}(z)| &= |C_{\text{odd}}(z)| \\ |D_{\text{odd}}(z)| &= |C_{\text{even}}(z)|. \end{aligned}$$

Go further to show that $D_{\text{even}} = \pm z^{-2L}$ (flip of C_{odd}) and $D_{\text{odd}} = \pm z^{-2L}$ (flip of $-C_{\text{even}}$). This gives the alternating flip with any even delay z^{-2L} .

10. Find $H_0(z)$ and $H_1(z)$ and $H_p(z)$. Is this paraunitary? Find the PR synthesis filters. What is $F_p(z)$?



11. For $H_p(z) = I - vv^T + z^{-1}vv^T$ with $v = \frac{1}{3} [2 -1 \quad 1 \quad \sqrt{3}]^T$, find the four PR filters.

12. Let $H_p(z) = R_1 \Lambda(z) R_0$ where $\theta_1 = \frac{\pi}{4}$ and $\theta_0 = -\frac{\pi}{2}$. What are the analysis and synthesis filters? Plot the frequency responses of $H_k(z)$.

5.3 Halfband Filters

Out of all the equations in the previous section, we would like to emphasize one. It came at the end. It applied first of all to the lowpass filter $C(z)$, and then by the alternating flip also to $D(z)$. It was equation (5.28), that the frequency response $C(\omega) = \sum c(k)e^{-j\omega k}$ satisfies

$$|C(\omega)|^2 + |C(\omega + \pi)|^2 = 2. \quad (5.31)$$

The key question is, *what does equation (5.31) say about $|C(\omega)|^2$ itself?*

We assign the symbol $P(\omega)$ to this important quantity $|C(\omega)|^2$. It is the *power spectral response*. Because $C(\omega)$ multiplies $\overline{C(\omega)}$, the filter with this response $P(\omega)$ is *symmetric*. It is the “autocorrelation filter”:

$$P(\omega) = \sum_{-N}^N p(n)e^{-j\omega n} = \left(\sum_0^N c(k)e^{-j\omega k} \right) \left(\sum_0^N c(k)e^{j\omega k} \right). \quad (5.32)$$

The function $P(\omega) = |C(\omega)|^2$ is real and nonnegative. It equals its complex conjugate. This verifies the symmetry $p(n) = p(-n)$ that was expected (with real coefficients).

To repeat: When $\sum c(k)e^{-j\omega k}$ multiplies its conjugate $\sum c(l)e^{j\omega l}$, we watch for $n = k - l$ which is $l = k - n$. The coefficient $p(n)$ is the sum of $c(k)$ times $c(k - n)$:

$$p(n) = \sum c(k)c(k - n) = \text{autocorrelation of the sequence } c(k). \quad (5.33)$$

Autocorrelation is $p = c * c^T$. This is the convolution of $c = (c(0), c(1), c(2), \dots)$ with its time reversal $c^T = (\dots, c(2), c(1), c(0))$. Replacing $-n$ by n in (5.33) brings no change in p .

The reason for the surprising notation c^T is that multiplying $C(\omega)$ by $\overline{C(\omega)}$ corresponds exactly to multiplying the infinite filter matrix C by C^T :

$$P = CC^T = \begin{bmatrix} & & & & \\ & \ddots & & & \\ & & c(0) & & \\ & & c(1) & & \\ & & c(2) & & \\ & & & \ddots & \\ & & & & \ddots \end{bmatrix} \begin{bmatrix} & & & & \\ & & & & \\ & & c(2) & & \\ & & c(1) & & \\ & & c(0) & & \\ & & & \ddots & \\ & & & & \ddots \end{bmatrix}. \quad (5.34)$$

P is a symmetric positive definite (or semidefinite) Toeplitz matrix. The transpose of CC^T is CC^T . The diagonal $p(0) = c(0)^2 + c(1)^2 + c(2)^2 + \dots$ is certainly positive. Equation (5.31) says that $p(0) = 1$, when the matrix C comes from an orthonormal filter bank.

What does equation (5.31) say about the other coefficients $p(n)$? In a word, it says *nothing* about the odd coefficients and it assigns *zero* to the even coefficients. C is the start of an orthogonal filter bank if and only if the autocorrelation filter P is a *halfband filter*:

$$P(z) + P(-z) = 2. \quad (5.35)$$

The *even coefficients* with $n = 2m$ must be $\delta(m)$:

$$\text{Halfband filter } p(2m) = \sum c(k)c(k - 2m) = \begin{cases} 1 & \text{if } m = 0 \\ 0 & \text{if } m \neq 0. \end{cases} \quad (5.36)$$

The odd coefficients are not necessarily zero! They cancel automatically in equation (5.35). To require $CC^T = I$, with odd coefficients zero, would make C an allpass filter. It could only be a delay. The requirement $P(z) + P(-z) \equiv 2$ is much weaker than $P(z) \equiv 1$. Perfect reconstruction is possible for an FIR filter bank, but essentially impossible for one FIR filter.

The highpass response $D(\omega)$ leads similarly to the autocorrelation $P_1(\omega) = |D(\omega)|^2$. Then DD^T must also be a normalized halfband filter. That result is automatic with the alternating flip, which gives $P_1(\omega) = P(\omega + \pi)$. Then $p_1(2m) = p(2m) = \delta(m)$ for the even coefficients. The odd coefficients change sign, $P_1(z) = P(-z)$, but the halfband condition $P_1(z) + P_1(-z) = 2$ remains true.

The sum $|C(\omega)|^2 + |C(\omega + \pi)|^2$ is $P(\omega) + P(\omega + \pi)$. For a halfband filter, this sum is a constant. The graph of $P(\omega)$ shows a special symmetry with respect to the halfband frequency $\omega = \frac{\pi}{2}$ — hence the name. Notice what happens in downsampling — the even coefficients yield the identity filter:

$$(\downarrow 2)P = I \text{ when } P \text{ is normalized halfband.} \quad (5.37)$$

Example 5.1. The symmetric filter with $p(0) = 1$ and $p(1) = p(-1) = \frac{1}{2}$ is a halfband filter. Its response is

$$P(z) = 1 + \frac{z^{-1} + z}{2} \text{ and equivalently } P(\omega) = 1 + \cos \omega.$$

In the z -domain, $P(z) + P(-z) = 2$. The odd powers cancel and there is no z^2 term. In the ω -domain $1 + \cos \omega + 1 - \cos \omega = 2$. The odd frequency cancels. Notice that $P(\omega) = 1 + \cos \omega$

is never negative! It does reach zero at the highest frequency $\omega = \pi$, corresponding to $z = -1$. When $P(\omega)$ touches zero, its spectral factor $C(\omega)$ must also touch zero — since $P = |C|^2$. This leads us to $C(\omega) = (1 + e^{-i\omega})/\sqrt{2}$:

$$P(\omega) = |C(\omega)|^2 = (1 + e^{-i\omega})(1 + e^{i\omega})/2 = 1 + \cos \omega. \quad (5.38)$$

These coefficients $c(0) = c(1) = 1/\sqrt{2}$ come from the familiar averaging filter. The division by $\sqrt{2}$ gives an orthonormal filter. Figure 5.2 shows the lowpass halfband filter P with response $1 + \cos \omega$. Added to $P(\omega + \pi)$ it gives the constant 2.

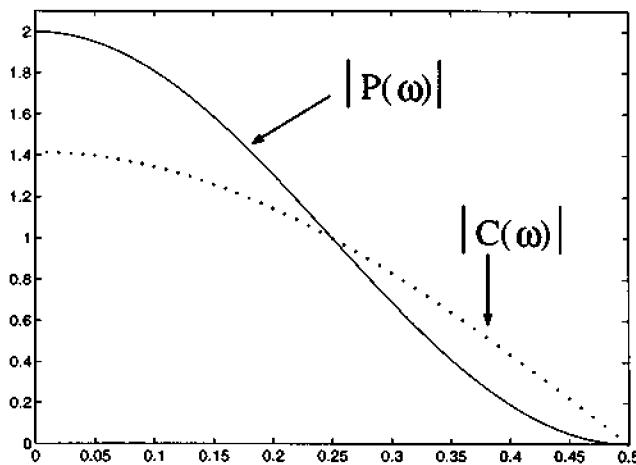


Figure 5.2: The orthonormal filter $C(\omega)$ has $P(\omega) = |C(\omega)|^2$. The normalized frequency 0.5 is $\omega = \pi$.

The requirement $P(\omega) \geq 0$ is crucial. Otherwise we could not factor $P(\omega)$ into $|C(\omega)|^2$. The halfband filter with coefficients $p(-1) = p(0) = p(1) = 1$ could never be $|C(\omega)|^2$. Its response $P(\omega) = e^{i\omega} + 1 + e^{-i\omega}$ is negative at $\omega = \pi$.

Example 5.2. The Daubechies 4-tap filter picks out $C(\omega)$ from $P = |C|^2$ when

$$P(\omega) = (1 + \cos \omega)^2 (1 - \frac{1}{2} \cos \omega). \quad (5.39)$$

Note the double zero at $\omega = \pi$, coming from $(1 + \cos \omega)^2$. If we keep only that factor, this $P(\omega)$ would be the square of the previous example. Its factor would be the square of the previous $C(\omega)$, namely $\frac{1}{2} + e^{-i\omega} + \frac{1}{2}e^{-2i\omega}$. Those coefficients $\frac{1}{2}, 1, \frac{1}{2}$ are important — they will lead to the *hat function*, when we study wavelets. But $P(\omega) = (1 + \cos \omega)^2$ does not by itself yield a halfband filter, so the lowpass C with coefficients $\frac{1}{2}, 1, \frac{1}{2}$ cannot go into an orthogonal filter bank. The hat function is not orthogonal to its translates.

To repeat: $(1 + \cos \omega)^2$ includes the term $\cos^2 \omega$. This produces an even frequency $\cos 2\omega$. In the z -domain we are squaring $1 + \frac{1}{2}(z^{-1} + z)$, which produces the even power z^{-2} . This is not halfband! Daubechies' extra factor $1 - \frac{1}{2} \cos \omega$ must be included, to cancel the 2ω term in $P(\omega)$ and the z^2 term in $P(z)$. We will have orthogonality, thanks to that factor.

To see that $P(\omega)$ is halfband, multiply it out. The $\cos 2\omega$ term is missing:

$$P(\omega) = (1 + 2 \cos \omega + \cos^2 \omega)(1 - \frac{1}{2} \cos \omega) = 1 + \frac{3}{2} \cos \omega - \frac{1}{2} \cos^3 \omega.$$

In the z -domain, the z^2 term is missing. Its coefficient $p(2)$ is zero:

$$P(z) = -\frac{1}{16}z^3 + \frac{9}{16}z + 1 + \frac{9}{16}z^{-1} - \frac{1}{16}z^{-3}. \quad (5.40)$$

We know that $P(\omega) \geq 0$ because $1 - \frac{1}{2}\cos\omega > 0$. Therefore it can be factored into $|C(\omega)|^2$ (*spectral factorization*). This is not a trivial calculation. It is made easier by the fact that we already know the factor for $1 + \cos\omega$, from the first example. This leaves only the linear piece $Q(\omega) = 1 - \frac{1}{2}\cos\omega$ or $Q(z) = 1 - \frac{1}{4}(z^{-1} + z)$. We factor Q in three steps:

$$1 - \frac{1}{4}(z^{-1} + z) = (b(0) + b(1)z^{-1})(b(0) + b(1)z) \quad (5.41)$$

$$1 = b(0)^2 + b(1)^2 \quad \text{and} \quad -\frac{1}{4} = b(0)b(1) \quad (5.42)$$

Solving the quadratic equations gives $b(0) = (1 + \sqrt{3})/\sqrt{8}$ and $b(1) = (1 - \sqrt{3})/\sqrt{8}$. The solutions are real because $1 - \frac{1}{2}\cos\omega$ is safely positive.

Another approach, basically the same, is to multiply (5.41) by z to get an ordinary quadratic. The quadratic formula gives its roots as $2 \pm \sqrt{3}$. Since $Q = 0$ at these roots, we have

$$b(0) + b(1)(2 \pm \sqrt{3})^{-1} = 0. \quad (5.43)$$

The previous solution is correct because $1 + \sqrt{3} + (1 - \sqrt{3})(2 - \sqrt{3})^{-1} = 0$. This is the *minimum phase solution*, from the root $2 - \sqrt{3}$ inside the unit circle. There is another solution from $2 + \sqrt{3}$, in which $b(0)$ and $b(1)$ are exchanged. This is *maximum phase*. Two more solutions come from reversing signs to $-b(0)$ and $-b(1)$. We are seeing the limited number of possible spectral factors in $|C(\omega)|^2$. The general rule for higher degree polynomials and longer filters is the same:

Minimum phase: Choose roots of $z^N P(z)$ that are on or *inside* $|z| = 1$.

Maximum phase: Choose roots of $z^N P(z)$ that are on or *outside* $|z| = 1$.

Mixed phase is also possible, choosing some roots inside and some outside. That can bring us to linear phase. We will show how linear phase factors of the Daubechies polynomials lead to *biorthogonal filter banks* which are among the current favorites.

Completion of Example. We factored $1 - \frac{1}{2}\cos\omega$ and $1 + \cos\omega$. Multiply to obtain

$$\begin{aligned} C(z) &= \frac{1}{4\sqrt{2}} (1 + z^{-1})^2 ((1 + \sqrt{3}) + (1 - \sqrt{3})z^{-1}) \\ &= \frac{1}{4\sqrt{2}} [(1 + \sqrt{3}) + (3 + \sqrt{3})z^{-1} + (3 - \sqrt{3})z^{-2} + (1 - \sqrt{3})z^{-3}]. \end{aligned} \quad (5.44)$$

Those are the four coefficients $c(0), \dots, c(3)$ of the famous Daubechies filter D_4 . In our present normalization, they are divided by $\sqrt{32} = 4\sqrt{2}$. In other normalizations they are divided by 4. Remember their two key properties:

1. The halfband filter has $P(z) + P(-z) = 2$. The factor $C(z)$ goes into an orthonormal filter bank. $D(z)$ comes from $C(z)$ by an alternating flip.
2. The response $C(z)$ has a double zero at $z = -1$. In frequency, $C(\omega)$ has a double zero at $\omega = \pi$. The response is *flat* at π because of $(1 + \cos\omega)^2$.

The double zero at $\omega = \pi$ will produce *two vanishing moments* for the Daubechies wavelets in Chapter 6.

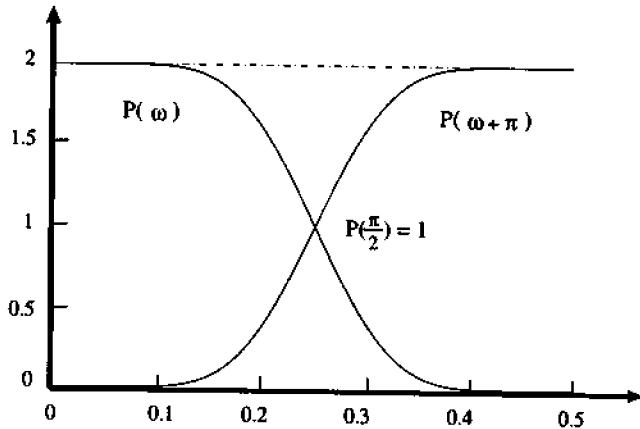


Figure 5.3: A halfband filter $P(\omega)$ and its mirror image $P(\omega + \pi)$. Their sum is constant.

Figure 5.3 shows a graph of $|C(\omega)|^2$, so you can see the halfband property that produces orthogonality. The highpass response in that figure is $|C(\omega + \pi)|^2$, which equals $|D(\omega)|^2$. The sum of the two is constant, so there is no amplitude distortion. The flatness gives great accuracy near $\omega = 0$ and $\omega = \pi$ (not so great in the middle). The filter bank gives perfect reconstruction.

Problem Set 5.3

1. Solve equations (5.42) for $x = b(0)^2$. Confirm the factorization.
2. What is the 2 by 2 polyphase matrix $H_p(z)$ from the Daubechies $C(z)$ and the alternating flip?
3. What is the 2 by 2 modulation matrix $H_m(z)$ for that four-tap Daubechies example? Verify that $\tilde{H}_m H_m = 2I$.
4. If a linear phase halfband filter satisfies $G(z) + G(-z) = z^{-l}$, what is the relation between l and N ? Can $G(z)$ be an antisymmetric?

5.4 Spectral Factorization

In an orthonormal filter bank, $C(z) = \sqrt{2}H(z)$ is a *spectral factor* of a symmetric halfband filter $P(z)$. The factorization is $P(z) = C(z^{-1})C(z)$ and the halfband property is $P(z) + P(-z) = 2$. In frequency, $P(\omega) = |C(\omega)|^2$ achieves the orthogonality condition $|C(\omega)|^2 + |C(\omega + \pi)|^2 = 2$. In the reverse direction, $P(z)$ is the *autocorrelation* of $C(z)$. This intimate relation of spectral factor $C(z)$ and its autocorrelation $P(z)$ is fundamental throughout signal processing.

Two questions arise immediately:

1. (Theory) Can every polynomial with $P(\omega) \geq 0$ be factored into $|C(\omega)|^2$?
2. (Practice) How is this spectral factorization actually done?

The answer to Question 1 is yes. This is the Féjer-Riesz Theorem. The answer to Question 2 is not so quick. There are many competing algorithms for spectral factorization. Short filters offer

no serious difficulty, but with 100 or even 50 coefficients the weaker algorithms become slow and/or unreliable. When $C(\omega)$ is only approximate, the reconstruction is not perfect.

The trigonometric polynomials $P(\omega)$ and $C(\omega)$ are both of degree N :

$$\sum_{-N}^N p(n)e^{-in\omega} = |C(e^{i\omega})|^2 = \left| \sum_0^N c(n)e^{-in\omega} \right|^2.$$

$P(z)$ has symmetric coefficients $p(n) = p(-n)$. There are $N + 1$ independent coefficients in P and the same number in C . They are linked by quadratic equations, when we solve $P(\omega) = |C(\omega)|^2$. Those equations are solvable if and only if $P(\omega) \geq 0$ for all ω .

As an aside, note that *matrix* spectral factorization is also possible where $P(\omega)$ is symmetric positive definite. Both 1 and 2, theory and practice, are nontrivial. The Riccati equation is involved.

We indicate four factorization methods. Three are actually used; Method C is for conversation only. The first method begins by finding the zeros of a polynomial (by a good algorithm!). This proves that spectral factorization is possible, by doing it.

Method A (zeros of a polynomial). With real symmetric coefficients $p(n)$, we have $P(z) = P(1/z)$. If z_i is a root, so is $1/z_i$. When z_i is inside the unit circle, $1/z_i$ is outside. The roots z_j on the unit circle must have even multiplicity, by the crucial assumption that $P(\omega) \geq 0$. Therefore the polynomial $z^N P(z)$ of degree $2N$, with leading coefficient $p(N) \neq 0$, must have these $2N$ factors:

$$z^N P(z) = p(N) \prod_{i=1}^M (z - z_i) \left(z - \frac{1}{z_i} \right) \prod_{j=1}^{N-M} (z - z_j)^2. \quad (5.45)$$

This contains the key point, but we know more. Real coefficients ensure that the complex conjugate \bar{z} is a root when z is a root. The complex roots off the unit circle actually come *four at a time*: z_i and \bar{z}_i inside, $1/z_i$ and $1/\bar{z}_i$ outside. The complex roots on the circle also come four at a time: z_j twice and \bar{z}_j twice. Real roots on the circle come two at a time (even multiplicity).

Now construct $C(z)$ by taking *all* the roots z_i (including \bar{z}_i) inside the circle, and also take one out of every double root z_j on the circle:

$$z^N C(z) = |p(N)|^{1/2} \prod_{i=1}^M (z - z_i) \prod_{j=1}^{N-M} (z - z_j). \quad (5.46)$$

This is the “minimum phase spectral factor.” It has no roots outside the circle. The coefficients of $C(z)$ are still real, because the complex roots are automatically in conjugate pairs: \bar{z}_i and \bar{z}_j came with z_i and z_j .

Example 5.3. The 4-tap Daubechies filter in the previous section led to zeros at $z_i = 2 - \sqrt{3}$ and $z_i^{-1} = 2 + \sqrt{3}$. The other four roots of $z^3 P(z)$ are at $z_j = -1$ (on the unit circle and again real). Two of those roots go into the spectral factor (5.46). Thus $z^3 C(z)$ is a cubic polynomial with roots $2 - \sqrt{3}$, -1 , and -1 . It is minimum phase.

Every factorization of $P(z)$ into $F(z) H(z)$ must put some roots into $H(z)$ and the remaining roots into $F(z)$. The rules for this separation of roots of $P(z)$ are:

- For F and H to be *real* filters, z and \bar{z} must stay together.
- For F and H to be *symmetric* filters, z and z^{-1} must stay together.

- For F to be the transpose of H , z and z^{-1} must go separately. This is the spectral factorization $C(z) C(z^{-1})$ that gives an orthogonal filter bank when P is halfband.

Figure 5.4a shows a partition of the zeros into circles and squares that makes both factors symmetric. The splitting in Figure 5.4b makes one factor the transpose (coefficients reversed) of the other factor. *To achieve both properties at the same time, all zeros of $P(z)$ – not just the zeros on the unit circle – must be of even multiplicity.* We now show that this is impossible for a halfband filter. This gives another proof of Theorem 5.3, that orthogonality conflicts with symmetry.

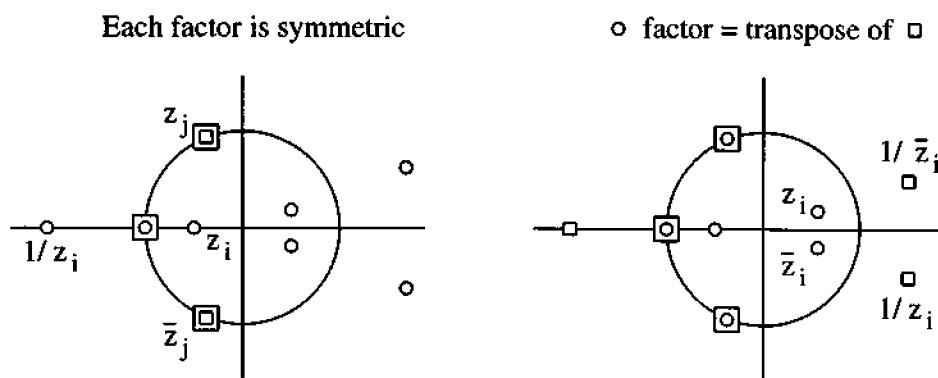


Figure 5.4: Twelve zeros of a halfband filter separate into analysis (○) and synthesis (□).

Theorem 5.4 A symmetric orthogonal FIR $C(z)$ can only have two nonzero coefficients.

Proof. An FIR filter is symmetric when $z^N C(z) = C(z^{-1})$. If z_i is a zero of $C(z)$, so is z_i^{-1} . Then $P(z) = C(z^{-1})C(z)$ has a double root at z_i . More precisely, all roots of the polynomial $z^N P(z)$ have even multiplicity. This polynomial is a perfect square $[R(z)]^2$.

If the filter is also orthogonal, $P(z)$ must be halfband:

$$z^N P(z) = [r(0) + \dots + r(N)z^N]^2 \text{ has only one odd power } z^N.$$

The first odd power in $R(z)$ produces (when it multiplies $r(0)$) an odd power in $[R(z)]^2$. The last even power in $R(z)$ also produces (when it multiplies $r(N)z^N$) an odd power in $[R(z)]^2$. But our halfband filter has only one odd power. We cannot allow any even powers or any odd powers in the terms \dots indicated by the three dots. The polynomial $R(z)$ has only two terms and $P(z)$ has three terms. Then $C(z)$ only has two nonzero coefficients.

All symmetric orthogonal FIR filters have $C(z) = (1+z^{-N})/\sqrt{2}$ with odd N . The halfband product filter is $P(z) = \frac{1}{2}z^N + 1 + \frac{1}{2}z^{-N}$.

Example 5.4. For symmetric filters, the roots $z_i = 2-\sqrt{3}$ and $z_i^{-1} = 2+\sqrt{3}$ must stay together when we factor $P(z)$. The four roots at $z = -1$ can be split between $H(z)$ and $F(z)$. One symmetric splitting is $H(z) = (1+z^{-1})(-1+4z^{-1}-z^{-2})/2\sqrt{2}$ and $F(z) = (1+z^{-1})^3/4\sqrt{2}$. There are several symmetric factorizations, but none of them can be orthogonal.

Now we return to **computation of zeros of polynomials**. For long filters, a good algorithm is needed to find the zeros z_i and z_j of $P(z)$. We quote from the 1994 abstract by Lang and Frenzel [La,Fr]:

Finding polynomial roots rapidly and accurately is an important problem in many areas of signal processing. We use Müller's method for computing a root of the deflated polynomial. This estimate is improved by applying Newton's method to the original polynomial. Furthermore we give a simple approach to improve the accuracy for spectral factorization when there are double roots on the unit circle.

Müller's method uses three previous estimates of the root of $z^N P(z)$ to find the next estimate. The parabola that interpolates at the three old points has a root at the new point. Since parabolas can have complex roots, Müller's algorithm can find complex roots from a real start — while Newton can only move chaotically on the real line.

Newton's method uses the most recent estimate z_k . For real roots, the tangent line at z_k to the graph of $z^N P(z)$ crosses zero at the new point z_{k+1} . This is the outstanding method for solving nonlinear equations, provided z_0 is close enough — which is the task of Müller's method. The polynomial $z^{10000} - 1$ was one of the tests (not the only one!). The code is on ftp from cml.rice.edu under directory *pub/software*.

MATLAB uses an eigenvalue method. Its subroutine `roots` is effective up to quite large degree. The roots of a polynomial $z^N + \dots$ are the eigenvalues of its $N \times N$ *companion matrix*, which has 1's down a diagonal and minus the polynomial coefficients along a row. For example, $z^3 - 2z^2 - 5z - 9$ is specified by the vector $v = [1 \ -2 \ -5 \ -9]$. The command $M = \text{compan}(v)$

produces the matrix $M = \begin{bmatrix} 2 & 5 & 9 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ with $\det(zI - M) = z^3 - 2z^2 - 5z - 9$. MATLAB finds the eigenvalues (by the QR method) which are the roots of the polynomial.

The next section mentions how rescaling γ to 4γ allowed us to compute Daubechies

The non-integer moments now resulting from γ allowed us to compute Daubechies filters of twice the length achievable without this scaling. The coefficients in $P(z)$ were better controlled. Linear phase filters with extremely good stopband attenuation have many zeros on or near the unit circle. These are the hardest zeros to compute.

Method B (solve quadratic equations). We are looking for $N + 1$ numbers $c(0), \dots, c(N)$. The $N + 1$ equations are of second degree, involving c 's times c 's. The equations come from matching powers of $e^{i\omega}$ in $\overline{C(\omega)}C(\omega) = P(\omega)$:

$$\left(\sum_0^N c(k) e^{ik\omega} \right) \left(\sum_0^N c(k) e^{-ik\omega} \right) = \sum_{-N}^N p(n) e^{-in\omega}. \quad (5.47)$$

One way to make those equations explicit is in matrix form:

The first equation is $c(0)^2 + \cdots + c(N)^2 = p(0)$. This gives the constant term in (5.47). The second equation gives the $e^{-i\omega}$ term. The last equation is $c(0)c(N)e^{-iN\omega} = p(N)e^{-iN\omega}$.

Equation (5.48) is not a linear system! It is quadratic, in fact homogeneous of degree 2. It has a real solution if and only if $P(\omega) \geq 0$ for all ω . This is not an easy condition to verify on the coefficients $p(n)$. If $P(\omega) \geq 0$ is not true—in which case our solution methods must fail—we can add enough to the DC term $p(0)$ to make it true.

For orthogonality, the $p(n)$ come from a halfband filter. The even coefficients $p(2), p(4), \dots$ are all zero. But our discussion is not in any way limited to this halfband case. Spectral factorization applies to all filters with $P(\omega) \geq 0$. It even applies to IIR filters, but those lead to infinitely many equations.

Example 5.5. The previous section factored $P(\omega) = 1 - \frac{1}{4}(e^{-i\omega} + e^{i\omega})$. Comparing coefficients of 1 and $e^{-i\omega}$ led us to

$$\begin{aligned} c(0)^2 + c(1)^2 &= 1 \\ c(0)c(1) &= -\frac{1}{4} \end{aligned} \quad \text{or} \quad \begin{bmatrix} c(0) & c(1) \\ 0 & c(0) \end{bmatrix} \begin{bmatrix} c(0) \\ c(1) \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{1}{4} \end{bmatrix}. \quad (5.49)$$

This is our system (5.48) for that particular example with $N = 1$. Eliminating $c(1)$ gave a single quadratic equation. The unknown was $x = c(0)^2$ and the equation was $x + \frac{1}{16x} = 1$ or $16x^2 - 16x + 1 = 0$. For $N > 1$ we *cannot* reduce the $N + 1$ quadratic equations to a single equation for $c(0)$. An approximate solution by method A, B, C, or D (or another method E) is the best we can expect.

To use a nonlinear equation solver, write the k th quadratic equation as $\frac{1}{2}\mathbf{c}^T Q(k)\mathbf{c} = p(k)$. The symmetric matrix $Q(k)$ has 1's along its k th subdiagonal and superdiagonal (and $Q(0) = 2I$). Here is $Q(2)$ with $N = 3$:

$$\frac{1}{2} \begin{bmatrix} c(0) & c(1) & c(2) & c(3) \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} c(0) \\ c(1) \\ c(2) \\ c(3) \end{bmatrix} = p(2). \quad (5.50)$$

This is $c(0)c(2) + c(1)c(3) = p(2)$ from matching the $e^{-2i\omega}$ terms in $|C(\omega)|^2 = P(\omega)$. The partial derivatives of that left side $L(2)$ are in the gradient vector $Q(2)\mathbf{c}$:

$$\begin{aligned} \partial L(2)/\partial c(0) &= c(2) \\ \partial L(2)/\partial c(1) &= c(3) \\ \partial L(2)/\partial c(2) &= c(0) \quad \text{agrees with} \\ \partial L(2)/\partial c(3) &= c(1) \end{aligned} \quad \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} c(0) \\ c(1) \\ c(2) \\ c(3) \end{bmatrix}.$$

The k th quadratic function is $L(k) = \frac{1}{2}\mathbf{c}^T Q(k)\mathbf{c}$, and its gradient is $Q(k)\mathbf{c}$.

The second derivatives are also desired by nonlinear subroutines, and also readily available. They are in the constant matrix $Q(k)$. One successful program using gradients and second derivatives has been the Quadratic Constrained Least Squares (QCLS) optimization code by anonymous ftp from <eceserv0.ece.wisc.edu> under the directory <pub/nguyen/software/QCLS>. We use it in Chapter 9 to design M -band filter banks.

Method C (matrix factorization). Equation (5.48) is an attractive form for the quadratic equations. But there is a more symmetric form. If you like infinite matrices, you will enjoy this. Instead of a finite matrix times a vector, it has an infinite constant-diagonal matrix C times its

transpose:

$$\mathbf{C}^T \mathbf{C} = \begin{bmatrix} c(0) & c(1) & \cdots & c(N) \\ c(0) & c(1) & \cdots & c(N) \\ c(0) & c(1) & \cdots & c(N) \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix} \begin{bmatrix} c(0) & c(0) & & \\ c(1) & c(1) & c(0) & \\ \vdots & \vdots & c(1) & \cdots \\ c(N) & c(N) & c(1) & \cdots \\ c(N) & c(N) & \vdots & \vdots \\ c(N) & \vdots & & \vdots \end{bmatrix} = \mathbf{P}.$$

The columns of \mathbf{P} have entries $p(-N), \dots, p(0), \dots, p(N)$. In this matrix form of $|C(\omega)|^2 = P(\omega)$, the symmetric matrix \mathbf{P} factors into *upper triangular* \mathbf{C}^T times *lower triangular* \mathbf{C} . This is only possible if \mathbf{P} is *positive semi-definite* — which is exactly our condition $P(\omega) \geq 0$.

Matrices are decomposed into triangular factors every day. This is the matrix statement of ordinary Gaussian elimination. The factorization is usually written $\mathbf{A} = \mathbf{LU}$. It gives *lower triangular times upper triangular*. Fortunately, infinite constant-diagonal matrices commute; our equation is also $\mathbf{CC}^T = \mathbf{P}$. The harder problem is to factor infinite matrices in finite time.

Approximate method: take a finite section of \mathbf{P} . Keep R rows and columns, where R is larger (perhaps much larger) than N . This finite piece \mathbf{P}_R is still symmetric and positive definite. Therefore \mathbf{P}_R can be factored into $\mathbf{C}_R \mathbf{C}_R^T$, where \mathbf{C}_R is lower triangular. That is the *Cholesky factorization* of \mathbf{P}_R . It is a symmetrized form of $\mathbf{A} = \mathbf{LU}$, available because \mathbf{P}_R is symmetric positive definite.

The finite matrix \mathbf{C}_R does not contain the exact $c(k)$. It is not even true that \mathbf{C}_R has constant diagonals (although \mathbf{P}_R has). The reduction to a finite matrix has chopped the tail ends of the row-column multiplications, either in $\mathbf{C}^T \mathbf{C}$ or in \mathbf{CC}^T . But the rows of the computed factor \mathbf{C}_R do approach the rows of \mathbf{C} . The correct (minimum-phase) coefficients $c(k)$ appear in the limit as $R \rightarrow \infty$.

We demonstrate with $P(\omega) = 1 - \frac{1}{2} \cos \omega$. This Daubechies example was solved exactly for $c(0) = (1 + \sqrt{3})/\sqrt{8} = 0.9659$ and $c(1) = (1 - \sqrt{3})/\sqrt{8} = -0.2588$. Take $R = 4$ and use `chol` in MATLAB to factor \mathbf{P}_4 into \mathbf{CC}^T :

$$\mathbf{P}_4 = \begin{bmatrix} 1 & -0.25 & & \\ -0.25 & 1 & -0.25 & \\ & -0.25 & 1 & -0.25 \\ & & -0.25 & 1 \end{bmatrix} \quad \text{and} \\ \mathbf{C} = \begin{bmatrix} 1 & & & \\ -0.25 & 0.9682 & & \\ -0.2582 & 0.9661 & & \\ -0.2588 & 0.9659 & & \end{bmatrix}.$$

This matrix has $c(0)$ and $c(1)$ correct to four places in the last row. But with long filters this method is very slow.

Method D (Cepstral method: Take logarithms). The idea is to convert the multiplication $P(z) = C(z^{-1})C(z)$ into addition. Formally, $\log(\sum p(n)z^{-n})$ is easily separated into positive and negative powers of z . The symmetry $p(n) = p(-n)$ and the positivity $P(\omega) \geq 0$ yield a logarithm $L(z)$ with coefficients $l(n) = l(-n)$:

$$\log P(z) = \sum_{-\infty}^{\infty} l(n)z^{-n} = \left(\frac{l(0)}{2} + \sum_1^{\infty} l(n)z^n\right) + \left(\frac{l(0)}{2} + \sum_1^{\infty} l(n)z^{-n}\right).$$

These are infinite series. The logarithm of a polynomial is not a polynomial. This means that our finite computations can only be approximate. The easy separation into $\log C(z^{-1}) + \log C(z)$ is the key advantage of the method.

The sequence $I(n)$ is the *complex cepstrum* of $p(n)$. The series for $L(z)$ converges in an annulus $|z_i| < |z| < 1/|z_i|$ of the complex plane. Here z_i is the largest root of $P(z)$ inside the unit circle, and $1/z_i$ is the smallest root outside. (The method is in trouble with roots z_j on the circle. Best to remove those first. Otherwise $L(z) = \log P(z)$ will be infinite at z_j and the series cannot converge.) The computation of the c 's requires an inverse Fourier transform of $\log P(z)$. Then the c 's are computed from the I 's.

A detailed treatment of the cepstrum $I(n)$ is given by Oppenheim and Schafer [OS]. The constant terms give $c(0) = \exp \frac{1}{2}I(0)$. The next term $c(1)$ is interesting because the z^{-1} terms only involve $I(0)$ and $I(1)$. The recursion for $n = 1, 2, \dots$ turns out to be

$$c(n) = I(n)c(0) + \frac{n-1}{n}I(n-1)c(1) + \cdots + \frac{1}{n}I(1)c(n-1).$$

We need only $N + 1$ coefficients $I(0), \dots, I(N)$ in the logarithm to find all $N + 1$ coefficients $c(0), \dots, c(N)$ in the spectral factor. To find those $I(n)$ from the given $p(n)$, we use a large-size FFT in the z -domain. A typical size is $8N$, for acceptable accuracy.

This cepstral method does *not* compute zeros of polynomials. So it doesn't find symmetric filters. It is a good way to find orthogonal filters. A code is available by anonymous ftp at `eceserv0.ece.wisc.edu` under directory `pub/nguyen/software/CEPSTRAL`.

Very optional comment. Spectral factorization also solves *singly infinite* constant-diagonal systems. (This is the genuine Toeplitz problem. Doubly infinite matrices could be named after Laurent — but mostly we still say Toeplitz.) The coefficient matrix P_+ has entries $p(i - j)$ only for $i \geq 0$ and $j \geq 0$:

$$P_+ x_+ = b_+ \text{ is } \begin{bmatrix} p(0) & p(1) & p(2) & \cdot \\ p(1) & p(0) & p(1) & \cdot \\ p(2) & p(1) & p(0) & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \cdot \end{bmatrix} = \begin{bmatrix} b(0) \\ b(1) \\ b(2) \\ \cdot \end{bmatrix}. \quad (5.51)$$

This corresponds in continuous time $t \geq 0$ to a *Wiener-Hopf integral equation*:

$$\int_0^\infty p(s-t)x(t) dt = b(s) \text{ for } s \geq 0. \quad (5.52)$$

P_+ does not have constant-diagonal factors in $P_+ = LU$. Lower triangular times upper destroys the time-invariant pattern. (Starting at time zero is responsible.) The beautiful Wiener-Hopf idea is that *upper times lower succeeds perfectly*:

$$P_+ = C_+^T C_+ = \begin{bmatrix} c(0) & c(1) & c(2) & \cdot \\ c(0) & c(0) & \cdot & \cdot \\ c(0) & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} c(0) & & & \\ c(1) & c(0) & & \\ c(2) & c(1) & c(0) & \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}. \quad (5.53)$$

Wiener and Hopf computed this spectral factorization of P_+ by Method D. One of Norbert Wiener's great theorems is that $\sum |I(n)| < \infty$ when $\sum |p(n)| < \infty$. With no zeros of $P(z)$ on the unit circle, he could take the logarithm even for IIR filters. The solution is

$$x_+ = (P_+)^{-1} b_+ = (C_+)^{-1} (C_+^T)^{-1} b_+. \quad (5.54)$$

The inverses of C_+ and C_+^T are constant-diagonal. So Wiener-Hopf can compute the spectral factorization $P(\omega) = |C(\omega)|^2$ and transform back to the time domain.

Finite constant-diagonal matrices don't have constant-diagonal factors and spectral factorization no longer succeeds. Nevertheless $P_N x_N = b_N$ can be solved quickly by the Levinson algorithm or a "superfast" algorithm or by preconditioned conjugate gradients [ChSt].

We added these comments because transform methods are so central to signal processing. This is the whole underpinning of filter theory.

Problem Set 5.4

1. Suppose $P(z)$ has six zeros at $z = -1$ and four other real zeros at $z = a, a^{-1}, b, b^{-1}$. Draw the complex plane and indicate which zeros go into the minimum phase spectral factor $C(z)$.
2. For the same ten zeros, indicate a set of zeros that produces a symmetric (linear phase) $C(z)$. Also indicate a second possibility.
3. Why must all roots of $P(z)$ on the unit circle have even multiplicity, to allow $P(z) = C(z) \times C(z^{-1})$ and $P(\omega) = |C(\omega)|^2$?
4. The coefficients of the Daubechies polynomials $C(z)$ up to order 12 are tabulated at the end of this chapter. Find the zeros using *roots* in MATLAB or another algorithm. What were the roots of the halfband polynomial $P(z)$?
5. Show that $P(z) = z^{-N} C(z) C(z^{-1})$ must be symmetric. If C is lowpass with passband and stopband cutoff frequencies ω_p and ω_s and errors δ_p and δ_s , what are the cutoff frequencies and the errors of P ?

5.5 Maxflat (Daubechies) Filters

This section is about an important family of filters, which will lead to an outstanding family of wavelets. The same construction yields both. Wavelets come from filters with special properties. Historically, their close relation was not immediately seen — now it is the subject of Chapter 6. The importance of this special construction is in its combination of two key properties:

1. These particular filters (and wavelets) are *orthogonal*.
2. The frequency responses have *maximum flatness* at $\omega = 0$ and $\omega = \pi$.

The lowpass filters will have $p = 1, 2, 3, 4, \dots$ zeros at π . They have $2p = 2, 4, 6, 8, \dots$ coefficients, so that $N = 2p - 1$. We use *boldface p* for the coefficients of $P(\omega) = |C(\omega)|^2$ and *lightface p* to count the zeros of $C(\omega)$ at $\omega = \pi$. The highpass coefficients $d(k)$ come from an alternating flip. The first member of this family was the subject of Chapter 1: $c(0) = c(1) = 1/\sqrt{2}$. Note the normalization $c(0)^2 + c(1)^2 = 1$. These numbers go into a *unitary matrix*. For each $p = 1, 2, 3, 4, \dots$ the filter bank is orthonormal. The product filters have degree $2N = 4p - 2$:

$$P_0(z) = \left(\frac{1+z^{-1}}{2}\right)^{2p} Q_{2p-2}(z) \text{ will be halfband by special choice of } Q.$$

In the literature on filters, this family is described as *maxflat*. The coefficients were given by [Herrmann]. They were already in formulas for interpolation, described below. In the history

of wavelets, we are reproducing the great 1988 discovery by Ingrid Daubechies. The filters are FIR with $2p$ coefficients. The wavelets are supported on the interval $[0, N] = [0, 2p - 1]$. As p increases, the filters are increasingly “regular” and the wavelets are increasingly “smooth.”

This section concentrates on filter properties and coefficients. We give a simple derivation of $P(z)$, and new facts about its zeros. The next chapters will concentrate on wavelets and the step into continuous time.

Condition O and Condition A_p

Before starting, it is helpful to count the requirements we must impose. There are $2p$ numbers to be chosen. These can be the coefficients $c(0), \dots, c(2p-1)$ in the lowpass filter, with frequency response $C(\omega)$. They could equally well be the coefficients $p(0), \dots, p(2p-1)$ of the centered (even) polynomial $P(\omega) = |C(\omega)|^2$. The c 's come from the p 's by spectral factorization. The nonnegative polynomial $P(\omega)$ is factored by the methods of the previous section:

$$P(\omega) = \sum_{n=0}^{2p-1} p(n)e^{-in\omega} \text{ equals } |C(\omega)|^2 = \left| \sum_{n=0}^{2p-1} c(n)e^{-in\omega} \right|^2. \quad (5.55)$$

Our formulas yield the numbers $p(n) = p(-n)$. Except for the first few filters in the family, there are no simple formulas for $c(n)$.

These $2p$ numbers are determined by p conditions for orthogonality from Condition O, and p conditions for a flat response from Condition A . More precisely, the requirement is “Condition A_p ”—the subscript indicates the order of flatness at $\omega = \pi$ (and $\omega = 0$). Here are the $p + p$ conditions:

Condition O $P = |C|^2$ is a normalized halfband filter:

$$\begin{aligned} & \text{Condition O: } \int_{-\pi}^{\pi} P(\omega) d\omega = 1, \text{ or } \int_{-\pi}^{\pi} |C(\omega)|^2 d\omega = 1, \\ & \text{or } p(0) = 1 \text{ and } p(2) = p(4) = \dots = p(2p-2) = 0. \end{aligned} \quad (5.56)$$

Condition A_p $C(\omega)$ has a zero of order p at $\omega = \pi$:

$$\begin{aligned} & \text{Condition } A_p: \text{ Condition O plus } C(\pi) = C'(\pi) = \dots = C^{(p-1)}(\pi) = 0. \end{aligned} \quad (5.57)$$

The equation $C(\pi) = 0$ says that $\sum c(n)(-1)^n = 0$. The odd-numbered coefficients have the same sum as the even-numbered coefficients:

$$\text{Condition } A_1 \text{ on } c(n): \quad \sum_{\text{odd } n} c(n) = \sum_{\text{even } n} c(n). \quad (5.58)$$

This is the first of the “sum rules.” Altogether we can impose the p th order zero in (5.57) as p sum rules on the coefficients:

$$\text{Condition } A_p \text{ on } c(n): \quad \sum_{n=0}^{2p-1} (-1)^n n^k c(n) = 0 \quad \text{for } k = 0, 1, \dots, p-1. \quad (5.59)$$

The factor n^k comes from the k th derivative of $\sum c(n)e^{-in\omega}$. Then $(-1)^n$ comes from substituting $\omega = \pi$. The convention for n^0 is 1.

Note on $C(0) = \sqrt{2}$: The sum rule (5.58) also applies to the coefficients $p(n)$, because $P(\omega) = |C(\omega)|^2$ also vanishes at $\omega = \pi$. The odd sum must be 1, since the only nonzero even-numbered coefficient is $p(0) = 1$:

$$\sum_{\text{odd } n} p(n) = \sum_{\text{even } n} p(n) = p(0) = 1. \quad (5.60)$$

The sum over all n is $P(0) = 2$. Then $P(\omega) = |C(\omega)|^2$ yields $C(0) = \pm\sqrt{2}$. We always choose the plus sign for a lowpass filter, so the DC term at $\omega = 0$ is not reversed in sign:

$$\sum_{\text{all } n} c(n) = C(0) = \sqrt{P(0)} = \sqrt{2}. \quad (5.61)$$

The p zeros at π mean that $C(\omega)$ has a factor $(1 + e^{-i\omega})^p$:

$$\text{Condition } A_p \text{ on } C(\omega): \quad C(\omega) = \left(\frac{1 + e^{-i\omega}}{2}\right)^p R(\omega). \quad (5.62)$$

$R(\omega)$ has degree $p - 1$, to bring the total degree of $C(\omega)$ to $2p - 1$. You could say that the p th order flatness is accounted for by $(1 + e^{-i\omega})^p$. Then the p coefficients in $R(\omega)$ are chosen to satisfy the p equations of Condition O.

To repeat: p equations for orthogonality and p equations for flatness. Condition O is applied to $P(\omega)$; it must be halfband. Condition A_p is applied to $C(\omega)$; it must have the factor $(1 + e^{-i\omega})^p$. This is easily converted to a condition on $P(\omega) = |C(\omega)|^2$, when we use $|1 + e^{-i\omega}|^2 / 2 = (1 + \cos \omega)$:

$$\text{Condition } A_p \text{ on } P(\omega): \quad P(\omega) \text{ has a factor } \left(\frac{1 + \cos \omega}{2}\right)^p. \quad (5.63)$$

Formulas for $P(\omega)$

We intend to give two formulas for $P(\omega) = |C(\omega)|^2$. The one associated with Ingrid Daubechies has $(1 + \cos \omega)^p$ times a sum of p terms. The formula associated with Yves Meyer gives the derivative of $P(\omega)$ as $-c(\sin \omega)^{2p-1}$. Then integration determines c and $P(\omega)$.

The best starting point is the ordinary polynomial $B_p(y)$. This has degree $p - 1$, with p coefficients. It is the binomial series for $(1 - y)^{-p}$, truncated after p terms:

$$B_p(y) = 1 + py + \frac{p(p+1)}{2}y^2 + \cdots + \binom{2p-2}{p-1}y^{p-1} = (1 - y)^{-p} + O(y^p). \quad (5.64)$$

The coefficient of y^k is $\binom{p+k-1}{k}$. The remainder has order y^p because this is the first term to be dropped. The complex zeros of this polynomial $B_p(y)$ will be all-important for the Daubechies filters.

We combine $B_p(y)$ with the factor $(1 - y)^p$ that has p zeros at $y = 1$. The variable y on $[0, 1]$ will correspond to the frequency ω on $[0, \pi]$. The product $\tilde{P}(y) = 2(1 - y)^p B_p(y)$ has exactly the flatness we want at $y = 0$:

$$2(1 - y)^p B_p(y) = 2(1 - y)^p [(1 - y)^{-p} + O(y^p)] = 2 + O(y^p). \quad (5.65)$$

This is a polynomial of degree $2p - 1$. It is the unique polynomial with $2p$ coefficients that satisfies p conditions at each endpoint:

$\tilde{P}(y)$ and its first $p - 1$ derivatives are zero at $y = 0$ and $y = 1$, except $\tilde{P}(0) = 2$.

Two more properties follow quickly. First, the derivative has $p - 1$ zeros at both end points. It is a polynomial of degree $2p - 2$ and with those zeros it must be

$$\tilde{P}'(y) = -Cy^{p-1}(1-y)^{p-1} \text{ for some } C. \quad (5.66)$$

The second property comes when we add $\tilde{P}(y)$ to $\tilde{P}(1-y)$. The sum equals 2 at both ends and is still flat. Its $2p$ coefficients are uniquely determined — it must be the constant polynomial 2:

$$\tilde{P}(y) + \tilde{P}(1-y) \equiv 2. \quad (5.67)$$

At $y = \frac{1}{2}$ this gives $\tilde{P}(\frac{1}{2}) = 1$. Figure 5.5 shows how $\tilde{P}(y)$ is odd around its middle value. This “Hermite interpolating polynomial” drops from 2 to 0 with flatness at the ends. Here are the polynomials for $p = 2$ and $p = 3$:

$$\begin{aligned} B_2(y) &= 1 + 2y & \text{and} & \tilde{P}(y) = 2(1-y)^2(1+2y) = 2 - 6y^2 + 4y^3 \\ B_3(y) &= 1 + 3y + 6y^2 & \text{and} & \tilde{P}(y) = 2(1-y)^3B_3(y) = 2 - 20y^3 + 30y^4 - 12y^5. \end{aligned}$$

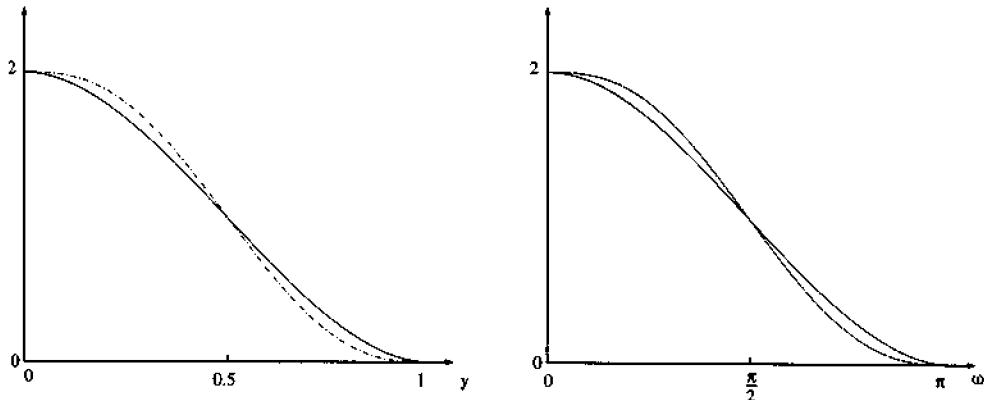


Figure 5.5: $\tilde{P}(y)$ on the left and $P(\omega)$ on the right, for $p = 2$ and $p = 3$.

Now we go from ordinary polynomials in y to trigonometric polynomials in ω . The degree stays at $2p - 1$. The change that takes $0 \leq y \leq 1$ into $0 \leq \omega \leq \pi$ is

$$y = \frac{1 - \cos \omega}{2} \text{ and } 1 - y = \frac{1 + \cos \omega}{2}. \quad (5.68)$$

The polynomial $\tilde{P}(y)$ becomes our desired $P(\omega)$. We summarize its properties.

Theorem 5.5 The polynomial $2(1-y)^p B_p(y)$ becomes the halfband response

$$P(\omega) = 2 \left(\frac{1 + \cos \omega}{2} \right)^p \sum_{k=0}^{p-1} \binom{p+k-1}{k} \left(\frac{1 - \cos \omega}{2} \right)^k. \quad (5.69)$$

This satisfies Conditions O and A_p . Its Meyer form, by integrating $P'(\omega)$ and choosing c to give $P(\pi) = 0$, is

$$P(\omega) = 2 - c \int_0^\omega (\sin \omega)^{2p-1} d\omega. \quad (5.70)$$

For $p = 1, 2, 3$ the Daubechies and Meyer forms are

$$\begin{aligned} P(\omega) &= 1 + \cos \omega = 2 - \int_0^\omega \sin \omega d\omega \\ P(\omega) &= (1 + \cos \omega)^2 (1 - \frac{1}{2} \cos \omega) = 2 - \frac{3}{2} \int_0^\omega \sin^3 \omega d\omega \\ P(\omega) &= (1 + \cos \omega)^3 (1 - \frac{9}{8} \cos \omega + \frac{3}{8} \cos^2 \omega) = 2 - \frac{15}{4} \int_0^\omega \sin^5 \omega d\omega \end{aligned}$$

Most authors emphasize the Daubechies form, with its highly visible factor $(1 + \cos \omega)^p$. That immediately ensures a p th order zero for the factors at $\omega = \pi$. Spectral factorization is speeded up, because only a lower-degree polynomial remains. It may not be so clear that (5.69) is a halfband filter. The even powers like $\cos^2 \omega$ and $\cos^4 \omega$ must disappear and they do. In the explicit formula for $p = 2$, multiplication produces $P(\omega) = 1 + \frac{3}{2} \cos \omega - \frac{1}{2} \cos^3 \omega$.

The halfband property is $P(\omega) + P(\omega + \pi) \equiv 2$. This addition cancels the odd powers of $\cos \omega$, and the even powers are not present (except the constant term 1). This identity follows immediately from (5.67) because $1 - y = \frac{1+\cos\omega}{2} = \frac{1-\cos(\omega+\pi)}{2}$:

$$\tilde{P}(y) + \tilde{P}(1 - y) \equiv 2 \text{ becomes } P(\omega) + P(\omega + \pi) \equiv 2. \quad (5.71)$$

The reader recognizes this “Condition O” as $|C(\omega)|^2 + |C(\omega + \pi)|^2 = 2$.

The halfband property is immediate in the Meyer form, with absolutely no calculations. Replace y by $(1 - \cos \omega)/2$ in (5.66) to find $P'(\omega) d\omega$:

$$-Cy^{p-1}(1-y)^{p-1}dy = -C\left(\frac{1-\cos\omega}{2}\right)^{p-1}\left(\frac{1+\cos\omega}{2}\right)^{p-1}\frac{\sin\omega}{2}d\omega. \quad (5.72)$$

This is $-c(1 - \cos^2 \omega)^{p-1} \sin \omega d\omega$, which is also $-c(\sin \omega)^{2p-1} d\omega$. Its integral is

$$-c \int (1 - \cos^2 \omega)^{p-1} \sin \omega d\omega = \text{odd powers of } \cos \omega.$$

The only even frequency is a constant of integration. The filter is halfband.

The flatness condition requires first of all that $P(\pi) = 0$. The constant c makes this true. The derivative $P'(\omega) = -c(\sin \omega)^{2p-1}$ has a zero of order $2p - 1$ at $\omega = \pi$. Then P itself has a zero of order $2p$. Its factor C has a zero of order p . Condition A_p is satisfied and Meyer's formula is confirmed.

Note that $P(\omega)$ decreases monotonically from $P(0) = 2$ to $P(\pi) = 0$. Its derivative $-c(\sin \omega)^{2p-1}$ is everywhere negative between 0 and π . There are no ripples in Figure 5.5. Therefore $P(\omega) \geq 0$ for all ω , and a factorization into $|C(\omega)|^2$ is assured.

The transition from passband (low frequencies) to stopband (high frequencies) becomes steeper and sharper as p increases. The slope at the midpoint $\omega = \frac{\pi}{2}$ is $-c(\sin \frac{\pi}{2})^{2p-1}$, which is $-c$. We will show that c increases asymptotically like \sqrt{p} as $p \rightarrow \infty$. Thus the transition band has width of order $1/\sqrt{p}$.

The Halfband Filter $P(z)$

Now we change from y and ω to the complex variable z . This will produce the filter coefficients in $P(z)$. That polynomial will be halfband and centered. The shifted polynomial $P_0(z) = z^{-N}P(z) = z^{1-2p}P(z)$ will be halfband and causal. The change of variables comes from $z = e^{i\omega}$:

$$\frac{z + z^{-1}}{2} = \cos \omega = 1 - 2y. \quad (5.73)$$

Thus $y = 0$ and $\omega = 0$ give $z = 1$. Similarly $y = 1$ and $\omega = \pi$ give $z = -1$.

Notice that the midpoints $y = \frac{1}{2}$ and $\omega = \frac{\pi}{2}$ give $z = \pm i$. There are two z 's for each y , from $z + z^{-1} = 2 - 4y$. (This is a quadratic equation for z .) One z is inside the unit circle, the other is $1/z$ outside. This "Joukowski transformation" is also central in fluid flow. The endpoints $z = 1$ and $z = -1$ are really double roots of $z + z^{-1} = 2$ and $z + z^{-1} = -2$.

The change of variable gives $1 - y$ and y in factored form:

$$1 - y = \frac{1 + \cos \omega}{2} = \left(\frac{1+z}{2}\right)\left(\frac{1+z^{-1}}{2}\right) \text{ and } y = \frac{1 - \cos \omega}{2} = \left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right). \quad (5.74)$$

Substituting in $\tilde{P}(y)$, the maxflat filter in the z -domain becomes $P(z)$:

$$P(z) = 2\left(\frac{1+z}{2}\right)^p\left(\frac{1+z^{-1}}{2}\right)^p \sum_{k=0}^{p-1} \binom{p+k-1}{k} \left(\frac{1-z}{2}\right)^k \left(\frac{1-z^{-1}}{2}\right)^k. \quad (5.75)$$

This factors into $P(z) = C(z)C(z^{-1})$ when $P(\omega)$ factors into $|C(\omega)|^2$. The p zeros at $y = 1$ and $\omega = \pi$ are now $2p$ zeros at $z = -1$. Half of them go into $C(z)$. The $p - 1$ complex zeros of the other factor $B_p(y)$ become $2p - 2$ zeros of $P(z)$. Half of those (the $p - 1$ zeros inside the circle $|z| = 1$, if we want minimum phase) also go into $C(z)$. So the spectral factor $C(z)$ can be computed in two steps:

1. Find the $p - 1$ zeros of $B_p(y)$ and the $p - 1$ corresponding z 's with $|z| < 1$.
2. Include p zeros at $z = -1$. Then $C(z)$ has these $2p - 1$ zeros.

Example. $p = 2$ leading to Daubechies D_4 from $B_2(y) = 1 + 2y$, which is $\frac{1}{2}(-z + 4 - z^{-1})$.

The zero is at $y = -\frac{1}{2}$. Therefore $z + z^{-1} = 4$. This quadratic equation has roots $z = 2 \pm \sqrt{3}$. Then the $2p - 1$ roots of $C(z)$ are $-1, -1, 2 - \sqrt{3}$. The coefficients of D_4 are approximately 0.4830, 0.8365, 0.2241, and -0.1294:

$$\begin{aligned} C(z) &= \alpha(1 + z^{-1})^2(1 - (2 - \sqrt{3})z^{-1}) \\ &= [(1 + \sqrt{3}) + (3 + \sqrt{3})z^{-1} + (3 - \sqrt{3})z^{-2} + (1 - \sqrt{3})z^{-3}] / 4\sqrt{2}. \end{aligned}$$

Example. $p = 70$ leading to Daubechies D_{140} from $B_{70}(y)$.

From $p = 2$ to $p = 70$ is quite a jump! Figure 5.6 displays the 69 zeros of $B_{70}(y)$. They are close to a limiting curve in the complex y -plane. The equation $|4y(1 - y)| = 1$ of that curve is discussed below. In the z -plane the limiting curve is moon-shaped, with the beautiful formula $|z - z^{-1}| = 2$. It consists of two circles! The roots of the minimum phase factor $C(z)$ are close

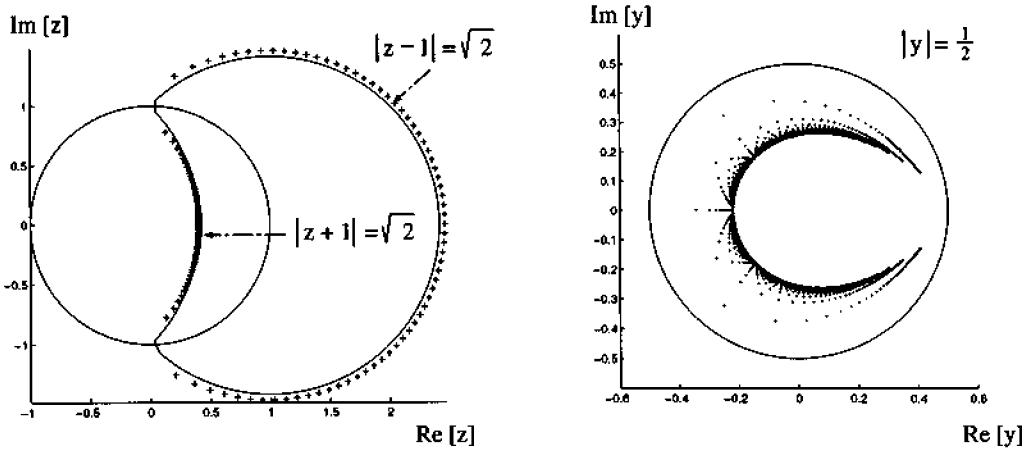


Figure 5.6: The 138 zeros of $P(z)$ and the zeros of $B_p(y)$ up to $p = 60$.

to the inner circle. To those $p - 1 = 69$ inner zeros we add $p = 70$ zeros at $z = -1$. Then the spectral factor of $P(z)$ has 139 roots, and the constant β makes $C(1) = \sqrt{2}$:

$$C(z) = \beta \left(\frac{1+z^{-1}}{2} \right)^{70} \prod_1^{69} (1 - z^{-1} Z_j). \quad (5.76)$$

Same Coefficients in Interpolation

The coefficients in $-\frac{1}{16} + \frac{9}{16}z^{-2} + z^{-3} + \frac{9}{16}z^{-4} - \frac{1}{16}z^{-6}$ appear in many places. This is a typical example, with $p = 2$, of a maxflat halfband filter. Its factor $(1 + z^{-1})^4$ yields $2p = 4$ zeros at $z = -1$. The halfband property means that P is an *interpolating filter*: Px keeps the even-numbered coefficients of $x = (\dots, x(0), 0, x(2), 0, \dots)$. Four zeros at π mean that the four polynomials $1, t, t^2, t^3$ are correctly interpolated in the odd components:

$$\begin{aligned} x = (\dots, 1, 0, 1, 0, 1, \dots) &\quad \text{gives} \quad Px = (\dots, 1, 1, 1, 1, 1, \dots) \\ x = (\dots, 0, 0, 2, 0, 4, \dots) &\quad \text{gives} \quad Px = (\dots, 0, 1, 2, 3, 4, \dots) \\ x = (\dots, 0, 0, 4, 0, 16, \dots) &\quad \text{gives} \quad Px = (\dots, 0, 1, 4, 9, 16, \dots) \\ x = (\dots, 0, 0, 8, 0, 64, \dots) &\quad \text{gives} \quad Px = (\dots, 0, 1, 8, 27, 64, \dots) \end{aligned}$$

Expressed differently, $\frac{9}{16}(x(1) + x(-1)) - \frac{1}{16}(x(3) + x(-3))$ is fourth-order accurate at the midpoint $t = 0$. This links wavelet theory to *recursive subdivision* (interpolation to create smooth curves). Starting with equally-spaced values $x(2n)$, P puts new values at the halfway points $t = n$. Then P produces new values at $t = \frac{n}{2}$. The established values do not change. The limit is a smooth curve through the original values. The monograph [CDM] develops this important application, with references. The stability of recursive interpolation is controlled by Condition E applied to P .

Asymptotics of the Daubechies Filters

Figure 5.6 practically requires us to study the zeros Y and Z as $p \rightarrow \infty$. The first steps were taken by [LeKa] and [ShSt]. The truncated binomial series $B_p(y)$ has degree $p - 1$. Its $p - 1$ zeros yield $2p - 2$ zeros in the z -plane, from $Z + Z^{-1} = 2 - 4Y$. The main facts proved so far are:

1. All the zeros have $|Y| \leq \frac{1}{2}$ and $\operatorname{Re} Z > 0$.
2. In the y -plane, the zeros are all outside the limiting curve $|w| = |4y(1 - y)| = 1$.
3. In the z -plane, the zeros are all outside the limiting curve $|z - z^{-1}| = 2$.
4. The zeros are near a uniform distribution along the circle of radius $1 + \log(4\pi p)/2p$ in the w -plane.
5. The far left zero is $Z = i - W/\sqrt{p} - iW^2/2p + O(p^{-3/2})$ where $\operatorname{erf}(W) = 1$.

Note that if z lies on that moon-shaped limiting curve $|z - z^{-1}| = 2$, so do \bar{z} and z^{-1} and \bar{z}^{-1} . The complex roots in the z -plane come *four at a time*, for finite p and in the limit $p = \infty$. The moon consists of two circles of radius $\sqrt{2}$ (Problem 1). The outer circle is $|z - 1| = \sqrt{2}$ with center at 1. The inner circle is $|z + 1| = \sqrt{2}$ with center at -1. The circles meet at $z = \pm i$, and the far left zeros go slowly toward these two points.

Figure 5.6 shows the zeros in the y -plane up to $p = 60$. Their approach to the limiting curve is fascinating. The application of long filters, with large p , is still to be developed. It can use the lattice structure of Section 4.5

Computing the Spectral Factor $C(z)$

For large p , computing the zeros of $B_p(y)$ and the spectral factorization $P(z) = C(z)C(z^{-1})$ are challenging tasks. They are related but not identical. After we find the zeros, the filter coefficients in $C(z)$ are fully determined — but not necessarily in a well-conditioned way. The direct multiplication of 69 or 139 linear factors is not safe. The cepstral method (Section 5.4) is competitive because it goes directly to $C(z)$, without the zeros. It is based on splitting $\log P(z)$ into $\log C(z) + \log C(z^{-1})$. Our experience with the zeros, using MATLAB and also Lang's code from Rice, showed the importance of a simple weighting:

The zeros of $B_p(y)$ were correct up to $p = 34$. The codes fail for $p = 35$.

With weighted variable $4y$, the zeros became correct up to $p = 80$.

The reason for the breakup at $p = 35$ is the wide dynamic range of coefficients in $B_p(y)$. The constant term is 1. The highest term y^{p-1} has coefficient

$$\frac{(2p-2)!}{(p-1)!(p-1)!} \approx \frac{\sqrt{2\pi(2p-2)}}{2\pi(p-1)} \frac{(2p-2)^{2p-2}}{(p-1)^{2p-2}} = \frac{4^{p-1}}{\sqrt{\pi(p-1)}} \quad (5.77)$$

by Stirling's formula. The leading term 4^{p-1} multiplying y^{p-1} suggests that $4y$ is a better variable than y . This was strongly confirmed by experiment. The recursion

$$b(p) = 1; \text{ for } p-1 : 1 : 1 \quad b(i) = b(i+1) * (2p-i-1)/4 * (p-i)$$

produces the coefficients in MATLAB order for the command $Y = \text{roots}(b)/4$.

The zeros are needed in a linear phase factorization $P_0(z) = H_0(z)F_0(z)$, which does not come from the cepstral method. In this case all four zeros $z, \bar{z}, z^{-1}, \bar{z}^{-1}$ go into the same factor (two from inside the unit circle, two from outside). One possibility is to put those quartets alternately in analysis and synthesis, H_0 and F_0 , to give filters of nearly equal length. Both get p zeros at $\omega = \pi$. Another possibility is for $H_0(z)$ to be very short, like Haar. Then $F_0(z)$ comes from an exact division $P_0(z)/H_0(z)$. No zeros are needed! Experiment will show which linear-phase factors give the best compression of signals and images.

Transition Band for Maxflat Filters

The equiripple filters from the Remez-Parks-McClellan algorithm have a sharp transition from lowpass to highpass. Their transition band has width of order $\frac{1}{N}$. Their slope at the midpoint $\omega = \frac{\pi}{2}$ is of order N . They minimize the maximum error, giving the best pointwise approximation to 1 in the passband and 0 in the stopband. But they only have one zero (at most) at $\omega = \pi$.

The Daubechies filters, with many zeros at π , have no ripples. $P(\omega)$ and $|C(\omega)|$ decrease monotonically between 0 and π . We now show that the slope at $\omega = \frac{\pi}{2}$ is much smaller, of order \sqrt{N} instead of $N = 2p - 1$.

Theorem 5.6 *The maxflat filter has center slope proportional to \sqrt{N} . The transition from $P(\omega) = 0.98$ to $P(\omega) = 0.02$ is over an interval of length $4/\sqrt{N}$.*

Proof. The constant c in Meyer's form is fixed by $c \int_0^\pi (\sin \omega)^N d\omega = 2$. This definite integral is known to be a ratio of Gamma functions (which are factorials $\Gamma(n+1) = n!$). We use Stirling's formula to estimate the integral:

$$\sqrt{\pi} \frac{\Gamma(\frac{N+1}{2})}{\Gamma(\frac{N+2}{2})} \simeq \sqrt{\pi} \left(\frac{N-1}{2e} \right)^{\frac{N-1}{2}} \left(\frac{2e}{N} \right)^{\frac{N}{2}} = \sqrt{\frac{2\pi e}{N-1}} \left(1 - \frac{1}{N} \right)^{\frac{N}{2}} \simeq \sqrt{\frac{2\pi}{N}}.$$

The slope of $P(\omega)$ at $\omega = \frac{\pi}{2}$ is $-c \simeq -\sqrt{2N/\pi}$ in Meyer's form. The transition bandwidth is therefore $O(1/\sqrt{N})$ and we make this more precise. Between $\frac{\pi}{2} - \frac{\sigma}{\sqrt{N}}$ and $\frac{\pi}{2} + \frac{\sigma}{\sqrt{N}}$ the drop in $P(\omega)$ is the integral of $c(\sin \omega)^N$. Shift by $\frac{\pi}{2}$ to center the integral, replacing $\sin(\frac{\pi}{2} - \omega)$ by $\cos \omega$:

$$\text{drop} = c \int_{-\sigma/\sqrt{N}}^{\sigma/\sqrt{N}} (\cos \omega)^N d\omega \simeq \frac{c}{\sqrt{N}} \int_{-\sigma}^{\sigma} \left(1 - \frac{\theta^2}{2N} \right)^N d\theta \simeq \sqrt{\frac{2}{\pi}} \int_{-\sigma}^{\sigma} e^{-\theta^2/2} d\theta. \quad (5.78)$$

Here $\theta = \omega\sqrt{N}$. Thus 95% of the drop in $P(\omega)$ comes with $\sigma = 2$ (within two standard deviations of the mean, for the normal distribution in statistics). This transition interval has width $\Delta\omega = 4/\sqrt{N}$, as the theorem predicts. That rule was found experimentally by Kaiser and Reed in 1977, at the beginning of the triumph of digital filters.

Problem Set 5.5

1. The halfband filter $P(\omega) = 1 + \sum p(n)e^{-in\omega}$ (odd n only) satisfies $P(\pi) = 0$. Deduce directly that $P(0) = 2$.
2. Find the zeros of $B_3(y)$ and the Daubechies 6-tap filter D_6 .

3. The definite integral $\int_0^\pi (\sin \omega)^N d\omega$ equals $\frac{1 \cdot 3 \cdot 5 \cdots N}{2 \cdot 4 \cdots (N-1)}$ for odd N . Express this in factorials and use Stirling's formula to rederive the estimate $\sqrt{2\pi/N}$. Then the slope at the center frequency $\frac{\pi}{2}$ is $O(\sqrt{N})$.
4. The points $z = 1 + \sqrt{2}e^{i\theta}$ are on the circle $|z - 1| = \sqrt{2}$. Substitute for z to show that this circle is one part of our limiting curve:

$$\left| \frac{z - z^{-1}}{2} \right| = \left| \frac{1 + \sqrt{2}e^{-i\theta}}{1 + \sqrt{2}e^{i\theta}} \right| = 1.$$

Chapter 6

Multiresolution

6.1 The Idea of Multiresolution

Our main approach to wavelets is through 2-channel filter banks. Everything develops from the filter coefficients. All constructions are concrete and highly explicit. Choose good coefficients and you get good wavelets. The heart of the theory is to see how conditions on the numbers $h(k)$ and $c(k)$ and $d(k)$ determine properties of $\phi(t)$ and $w(t)$ —the scaling function and the basic wavelet. Then the problem is to design filters that achieve those properties.

By iterating the filter bank, Section 6.2 reaches the *dilation equation* for $\phi(t)$ and the *wavelet equation* for $w(t)$. Sections 6.3 and 6.4 study those equations in the time domain and frequency domain. Conditions O and A lead to orthogonality and approximation accuracy. The Daubechies wavelets are “optimal” with respect to those two properties. But these orthogonal wavelets are not and cannot be symmetric (except for Haar). Also the transition from passband to stopband is not sharp. So the design problem is still open. Better wavelets remain to be constructed.

This opening section aims for an overview that brings out the key ideas. Before the construction using discrete time, we describe what is wanted in continuous time. The goal is a decomposition of the whole function space into subspaces. That implies a decomposition of each function—*there is a piece of $f(t)$ in each subspace*. Those pieces (or projections) give finer and finer details of $f(t)$. The signal is “resolved” at scales $\Delta t = 1, 1/2, \dots, (1/2)^j$.

For audio signals, these scales are essentially *octaves*. They represent higher and higher frequencies. For images and indeed for all signals, the simultaneous appearance of multiple scales is known as *multiresolution*.

Multiresolution will be described first for subspaces V_j and W_j . The scaling spaces V_j are *increasing*. The wavelet space W_j is the *difference* between V_j and V_{j+1} . *The sum of V_j and W_j is V_{j+1}* . Then these extra conditions involving dilation to $2t$ and translation to $t - k$ define a genuine multiresolution:

If $f(t)$ is in V_j then $f(t)$ and $f(2t)$ and all $f(t - k)$ and $f(2t - k)$ are in V_{j+1} .

In the end, one wavelet generates a whole basis. The functions $w(2^j t - k)$ come by dilation and translation (all j and all k). There are six steps toward this goal, and we take them one at a time:

1. An increasing sequence of subspaces V_j (complete in L^2)

2. The wavelet subspace W_j that gives $V_j + W_j = V_{j+1}$
3. The dilation requirement from $f(t)$ in V_j to $f(2t)$ in V_{j+1}
4. The basis $\phi(t - k)$ for V_0 and $w(t - k)$ for W_0
5. The basis $\phi(2^j t - k)$ for V_j and $w(2^j t - k)$ for W_j
6. The basis of all wavelets $w(2^j t - k)$ for the whole space L^2 .

A shortcut to multiresolution. Before those six steps, may I mention one shortcut step that starts with the filter coefficients $h(k)$. That step is to solve the dilation equation for the scaling function $\phi(t)$:

$$\phi(t) = \sum h(k) \phi(2t - k).$$

The first requirements on the coefficients are $\sum h(k) = 1$ and $\sum (-1)^k h(k) = 0$. The full requirement is Condition E in Section 7.2. When this is satisfied, $\phi(t)$ can be computed. Then $\{\phi(2^j t - k)\}$ is a basis for V_j . These spaces are automatically increasing and complete and shift-invariant and connected by dilation. Thus multiresolution is achieved.

A Scale of Subspaces

Each V_j is contained in the next subspace V_{j+1} . A function in one subspace is in all the higher (finer) subspaces:

$$V_0 \subset V_1 \subset \cdots \subset V_j \subset V_{j+1} \subset \cdots$$

A function $f(t)$ in the whole space has a piece in each subspace. Those pieces contain more and more of the full information in $f(t)$. The piece in V_j is $f_j(t)$. One requirement on the sequence of subspaces is *completeness*:

$$f_j(t) \rightarrow f(t) \quad \text{as } j \rightarrow \infty.$$

The first example will not have the dilation feature required for multiresolution:

Example 6.1. V_j contains all trigonometric polynomials of degree $\leq j$.

Certainly V_j is contained in V_{j+1} . The spaces are growing. (Since Daubechies uses $-j$ where we use j , her subspaces are *decreasing*. Most authors now use an increasing sequence, for simpler numbering.) The piece of $f(t)$ in V_j is the partial sum $f_j(t)$ of its Fourier series:

$$f_j(t) = \sum_{|k| \leq j} c_k e^{ikt} \text{ is the piece in } V_j.$$

This is the *projection* of $f(t)$ onto V_j . The exponentials e^{ikt} are orthogonal, so the energy in $f_j(t)$ is the sum of $|c_k|^2$ over low frequencies $|k| \leq j$. The energy in $f(t) - f_j(t)$ is the sum over high frequencies $|k| > j$. This approaches zero as $j \rightarrow \infty$. Therefore the sequence V_j is complete in the whole 2π -periodic space L^2 .

Now we identify the second family of subspaces. W_j contains the new information $\Delta f_j(t) = f_{j+1}(t) - f_j(t)$. This is the “detail” at level j . From the viewpoint of individual functions,

$$f_j(t) + \Delta f_j(t) = f_{j+1}(t). \quad (6.1)$$

With orthogonality of each piece $f_j(t)$ to the next detail $\Delta f_j(t)$, these subspaces are orthogonal. But we emphasize now that *orthogonality is not essential*.

A nonorthogonal example comes directly from any nonorthogonal basis $b_0(t), b_1(t), \dots$. The piece $f_j(t)$ includes all the terms through $b_j(t)$:

$$\begin{aligned} \text{Sum up to } j : \quad f_j(t) &= \sum_0^j c_k b_k(t) \quad \text{is in } V_j \\ \text{Next term :} \quad \Delta f_j(t) &= c_{j+1} b_{j+1}(t) \quad \text{is in } W_j. \end{aligned}$$

The pattern is not lost, just the orthogonality. The new space V_{j+1} is still the “*direct sum*” of V_j and W_j , which intersect only at the zero vector. The angle between subspaces can be less than 90°, as long as every f_{j+1} in V_{j+1} has exactly one splitting into $f_j + \Delta f_j$:

$$V_j \cap W_j = \{0\} \quad \text{and} \quad V_j + W_j = V_{j+1}. \quad (6.6)$$

This nonorthogonal situation applies to *biorthogonal filters* and wavelets (Section 6.5). There W_j is orthogonal to a different subspace \tilde{V}_j . The extra freedom can be put to good use.

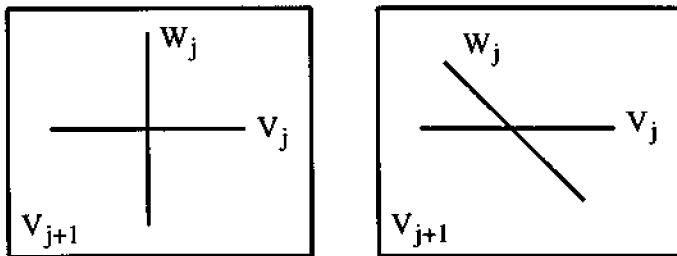


Figure 6.1: An orthogonal sum and a direct sum. Both written $V_j \oplus W_j = V_{j+1}$ and both allowed.

The Dilation Requirement

So far we have an increasing and complete scale of spaces. Each V_j is contained in the next V_{j+1} . For multiresolution, the crucial word *scale* carries an additional meaning. V_{j+1} consists of all rescaled functions in V_j :

$$\text{Dilation: } f(t) \text{ is in } V_j \iff f(2t) \text{ is in } V_{j+1}.$$

The graph of $f(2t)$ changes twice as fast as the graph of $f(t)$. On a map, the scale is doubled. At 3,000,000:1 the state of Utah fills a page. At 6,000,000:1 its height is a half page. The length that represents a mile is cut in half. This length is Δt or Δx or h .

The example using $f(t) = c_{-j} e^{-i jt} + \dots + c_j e^{ijt}$ does not meet this rescaling requirement. The highest frequency only increases by one, between V_j and V_{j+1} . But when t is changed to $2t$, the highest frequency becomes $2j$. *The frequencies must double*. The new space V_{j+1} is required to contain all those new frequencies. To satisfy the scaling requirement, the partial sums go an octave at a time. *The sum for f_j should stop at frequency 2^j instead of j* . Then Δf_j contains all frequencies between 2^j and 2^{j+1} :

$$\begin{aligned} \text{Multiresolution example :} \quad f_j(t) &= \sum c_k e^{ikt} \quad \text{for } |k| \leq 2^j \\ \text{Next detail :} \quad \Delta f_j(t) &= \sum c_k e^{ikt} \quad \text{for } 2^j < |k| \leq 2^{j+1}. \end{aligned}$$

This is a genuine multiresolution, in which V_j and W_j have roughly the same dimension. It is the Littlewood-Paley decomposition of a Fourier series, into octaves instead of single terms.

This is a chief part of the mathematical background. To fit the requirements precisely, when $f(t)$ is defined on the whole line $-\infty < t < \infty$, we should use all frequencies ω and not just integers:

$$f_j(t) = \frac{1}{2\pi} \int_{|\omega| \leq 2^j} \widehat{f}(\omega) e^{i\omega t} d\omega.$$

Now the spaces V_j go down the scale toward $j = -\infty$, as well as up the scale. The continuous frequency ω can be halved as well as doubled. The basis functions become *sinc functions*, by the sampling theorem. *Continuous frequency but discrete basis*, as is normal for L^2 . And the nested spaces include $j < 0$:

$$\cdots \subset V_{-1} \subset V_0 \subset \cdots \subset V_j \subset V_{j+1} \subset \cdots \quad (6.7)$$

In addition to completeness as $j \rightarrow \infty$, we require emptiness as $j \rightarrow -\infty$:

$$\bigcap V_j = \{0\} \quad \text{and} \quad \overline{\bigcup V_j} = \text{whole space.} \quad (6.8)$$

Emptiness means that $\|f_j(t)\| \rightarrow 0$ as $j \rightarrow -\infty$. Completeness still means that $f_j(t) \rightarrow f(t)$ as $j \rightarrow \infty$. The detail $\Delta f_j = f_{j+1} - f_j$ belongs to W_j and we still have

$$V_j \bigoplus W_j = V_{j+1}. \quad (6.9)$$

This can be an orthogonal sum, with Δf_j orthogonal to f_j . It must be a direct sum, with $V_j \cap W_j = \{0\}$. The reconstruction of $f(t)$ from its details Δf_j can start at $j = 0$ as before, or it can start at $j = -\infty$:

$$f(t) = f_0(t) + \sum_0^{\infty} \Delta f_j(t) \quad \text{or} \quad f(t) = \sum_{-\infty}^{\infty} \Delta f_j(t).$$

The sum of subspaces can start at $j = 0$ or $j = -\infty$. When the sum stops at $J \geq 0$, we have the subspace V_{J+1} :

$$V_{J+1} = V_0 + \sum_{j=0}^J W_j \quad \text{or} \quad V_{J+1} = \sum_{j=-\infty}^J W_j.$$

The left sum includes the scaling functions in V_0 . The sum on the right involves only the wavelets. That form includes all the very large time scales $\Delta t = 2^{-j}$ as $j \rightarrow -\infty$.

In practice we use the first sum. Our calculations begin at some unit scale. The scaling functions at $j = 0$ and the wavelets with $j \geq 0$ are the basis. I suppose the scaling functions at level $j = J$ and the wavelets with $j \geq J$ are another basis.

The Translation Requirement and the Basis

Instead of rescaling $f(t)$, we now shift its graph. This is *translation*, and it leads to the fundamental requirement of time-invariance in signal processing. The subspaces are *shift-invariant*:

If $f_j(t)$ is in V_j then so are all its translates $f_j(t - k)$.

Suppose $f(t)$ is in V_0 . Then $f(2t)$ is in V_1 and so is $f(2t - k)$. By induction, $f(2^j t)$ is in V_j and so is $f(2^j t - k)$. Dilation and translation are now built in.

With translation we are committed to working on the whole line $-\infty < t < \infty$, or to periodicity. A particular $f(t)$ may have compact support, but the whole space V_0 (all functions together) is shift-invariant. For finite intervals, the requirements have to be (and can be) adjusted. Dilation and translation operate freely on the whole line, and can be studied by Fourier transform.

The final requirement for multiresolution concerns a *basis* for each space V_j . If we choose one function $\phi(t)$ in V_0 , its translates $\phi(t - k)$ may be independent. These translates may span the whole space V_0 . They may even be orthonormal. The starting assumption, to be weakened later, is that V_0 contains such a function:

There exists $\phi(t)$ so that $\{\phi(t - k)\}$ is an orthonormal basis for V_0 .

When the functions $\phi(t - k)$ are an orthonormal basis for V_0 , the rescaled functions $\sqrt{2}\phi(2t - k)$ will be an orthonormal basis for V_1 . At scaling level j , the basis functions $\phi(2^j t - k)$ are normalized by $2^{j/2}$. We collect all the requirements in one place:

Multiresolution Analysis

The subspaces V_j satisfy requirements 1 to 4:

1. $V_j \subset V_{j+1}$ and $\bigcap V_j = \{0\}$ and $\overline{\bigcup V_j} = L^2$ (completeness).
2. *Scale invariance:* $f(t) \in V_j \iff f(2t) \in V_{j+1}$.
3. *Shift invariance:* $f(t) \in V_0 \iff f(t - k) \in V_0$.
4. *Shift-invariant basis:* V_0 has an orthonormal basis $\{\phi(t - k)\}$.
- 4'. *Shift-invariant basis:* V_0 has a stable basis (Riesz basis) $\{\phi(t - k)\}$.

4 and 4' are interchangeable. A stable basis can be orthogonalized in a shift-invariant way. This is in Section 6.4, together with the definition: stable = Riesz = uniformly independent. In practice we choose a convenient basis, orthogonal or not. Then V_j has the basis $\phi_{jk}(t) = 2^{j/2}\phi(2^j t - k)$:

$$f_j(t) = \sum_{k=-\infty}^{\infty} a_{jk} \phi_{jk}(t) \text{ is the piece in } V_j.$$

In the orthogonal case, the energy in this piece is

$$\|f_j\|^2 = \sum_{k=-\infty}^{\infty} |a_{jk}|^2. \quad (6.10)$$

Shift-invariance and scale-invariance are built in through the basis $\{2^{j/2}\phi(2^j t - k)\}$. This basis combines requirements 2, 3, and 4!

We have at least three ways to construct or describe a multiresolution:

1. By the spaces V_j

2. By the scaling function $\phi(t)$
3. By the coefficients $2h(k)$ in the dilation equation.

Our next examples use the spaces V_j and their bases. Then we move to description 3 and the dilation equation. Section 6.4 will orthogonalize the basis. The result will be the orthonormal $\phi(t - k)$ that multiresolution originally asks for. What we really need is a good shift-invariant basis.

It is also possible to allow *several* scaling functions ϕ_1, \dots, ϕ_r , when one function (with its translates) cannot produce the whole space V_0 . This occurs in Example 3 below. It corresponds to “*multiwavelets*”.

The framework for multiresolution is set by the dilation-translation requirement. Examples come first. Then we study the dilation equation, and construct wavelets.

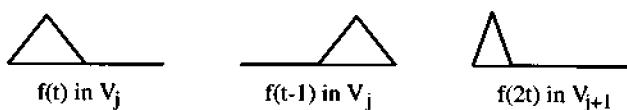


Figure 6.2: Translation stays in V_j . Dilation moves into V_{j+1} . Why is $f(2t) - f(t)$ not in V_j ?

Examples of Multiresolution

1. Piecewise constant functions. V_0 contains all functions in L^2 that are constant on unit intervals $n \leq t < n + 1$. These functions are determined by their values $f(n)$ at all integer times $t = n$:

$$f(t) = f(\text{integer part of } t).$$

The function $f(2t)$ in V_1 is then constant on half-intervals. The functions in V_j are constant on intervals of length 2^{-j} . The spaces are increasing, $V_j \subset V_{j+1}$, because any function that is constant on intervals of length 2^{-j} is automatically constant on intervals of half that length. These are *dyadic intervals*, starting at a dyadic number $t = n/2^j$ and ending at $t = (n + 1)/2^j$.

These spaces are shift-invariant—the translate of a piecewise constant function is still piecewise constant. The step from j to $j + 1$ rescales time by 2 and produces V_{j+1} . What about a basis? The simplest choice is the *box function*:

$$\phi(t) = \begin{cases} 1 & \text{for } 0 \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{is orthogonal to its translates } \phi(t - k).$$

Every function in V_0 is a combination of boxes $f(t) = \sum f(n)\phi(t - n)$. So requirement 4 is satisfied by the box function $\phi(t)$.

2. Continuous piecewise linear functions. The functions $f(t)$ are now linear between each pair of values $f(n)$ and $f(n + 1)$. Notice again the shift-invariance and the scale-invariance:

Shift: If $f(t)$ is piecewise linear, so is $f(t - k)$.

Scale: If $f(t)$ is linear on unit intervals, then $f(2t)$ is linear on half-intervals.

The spaces are right for multiresolution. Is there a shift-invariant basis?

The basis function that comes to mind is the *hat function* $H(t)$, equal to one at $t = 1$, and linear between its values $H(n) = \delta(n - 1)$. The translates $H(t - k)$ generate all piecewise linear functions on unit intervals. Any function $f(t)$ in V_0 can be expressed as $\sum f(n - 1)H(t - n)$. However $H(t)$ is *not orthogonal* to the neighboring hat $H(t - 1)$. The product $H(t)H(t - 1)$ is positive on the one interval $1 < t < 2$ where the hats overlap. Its integral (the inner product of the hats) is not zero.

We must work harder to find an orthogonal basis, and the eventual $\phi(t)$ will not have compact support. Or else we keep this non-orthogonal basis.

3. Discontinuous piecewise linear functions. Now $f(t)$ in V_0 may have a jump at each meshpoint $t = n$. There is a value $f(n_-)$ from the left and a value $f(n_+)$ from the right. The hat function is still in the space, but so is the box function! The spaces V_j are clearly shift-invariant and scale-invariant. If $f(t)$ is linear between integers (where it jumps), then $f(2t)$ is linear between half-integers (where it jumps).

There are two degrees of freedom at each meshpoint, the values $f(n_-)$ and $f(n_+)$. Therefore *two scaling functions* $\phi_1(t)$ and $\phi_2(t)$ are required for a shift-invariant basis. They can both be supported on the unit interval, and they can be orthogonal:

$$\phi_1(t) = \text{box function} \quad \text{and} \quad \phi_2(t) = \text{sloping line} = 1 - 2t.$$

The union of $\{\phi_1(t - k)\}$ and $\{\phi_2(t - k)\}$ is an orthonormal basis—which illustrates the idea behind “*multiwavelets*”. The usual dilation equation for $\phi(t)$ becomes a vector equation for $\phi_1(t)$ and $\phi_2(t)$. The coefficients $c(k)$ in that equation are 2×2 matrices. The associated filter bank in Section 7.5 contains “*multifilters*”.

4. Cubic splines. V_0 consists of piecewise cubic polynomials on unit intervals, with $f(t)$ and $f'(t)$ and $f''(t)$ continuous. The third derivative $f'''(t)$ may jump at the integers $t = n$, so the cubics are different in neighboring intervals. We have shift-invariance and scale-invariance, when V_1 contains the cubic splines on half-intervals. *This is the main point:* Approximating subspaces on regular meshes automatically fit the requirements for multiresolution.

The shortest cubic spline is a *B-spline*. It consists of different third-degree polynomials on the four unit intervals within $0 \leq t \leq 4$. The letter *B* stands for basis, but not for orthogonal basis. In complete analogy with the hat function, which is a linear spline, the scaling function $\phi(t)$ for the cubic splines cannot have compact support if we insist on orthogonality. An orthogonal basis $\{\phi(t - k)\}$ does exist, but it requires Fourier analysis to find it.

The cubic *B-spline* satisfies a dilation equation with very simple coefficients, proportional to $1, 4, 6, 4, 1$. But those coefficients do not lead to orthogonal filters. We can stay with these coefficients and go to biorthogonal filters (the best plan). Or we can orthogonalize, losing compact support and reaching a filter with infinitely many coefficients. Section 7.4 develops the theory of splines.

5. Daubechies functions. The search for orthogonal filter banks leads to the four coefficients of a “maxflat” lowpass filter. The response $C(\omega)$ has a double zero at the highest frequency $\omega = \pi$. This is maximal flatness, with four coefficients and orthogonality:

$$c(0), c(1), c(2), c(3) = 1 + \sqrt{3}, 3 + \sqrt{3}, 3 - \sqrt{3}, 1 - \sqrt{3} \text{ times } 1/4\sqrt{2}.$$

Their sum is $\sqrt{2}$. Their sum of squares is unity. They are orthogonal to their double shifts, because $c(0)c(2) + c(1)c(3) = 0$. From these coefficients Daubechies constructed $\phi(t)$ by solving the dilation equation

$$\phi(t) = \sqrt{2} \sum_{k=0}^3 c(k) \phi(2t - k).$$

The solution comes in the next section. The zeroth space V_0 contains every $\phi(t - k)$. Those functions are an orthonormal basis. The rescaled functions $\phi(2^j t - k)$ span V_j .

This is our best description of the Daubechies spaces V_j , to give the dilation equation for $\phi(t)$. In Examples 1–4, we started with the spaces. In Example 5, Daubechies started with the coefficients and found $\phi(t)$ — which produces the spaces. Either way, we have the scale-invariance and shift-invariance of multiresolution analysis.

The Dilation Equation

The space V_0 is contained in V_1 . Therefore $\phi(t)$ is also in V_1 . It must be a combination of the basis functions $2^{1/2}\phi(2t - k)$ for that subspace. The coefficients in the combination will be called $c(k)$. Bring the factor $2^{1/2} = \sqrt{2}$ outside:

$$V_0 \subset V_1 \text{ means } \phi(t) = \sqrt{2} \sum_k c(k) \phi(2t - k). \quad (6.11)$$

This is the dilation equation. It is a two-scale equation, involving t and $2t$. It is also called a refinement equation, because it displays $\phi(t)$ in the refined space V_1 . That space has the finer scale $\Delta t = 1/2$, and it contains $\phi(t)$ which has scale $\Delta t = 1$.

To emphasize: The dilation equation is a direct consequence of $V_0 \subset V_1$. It is not an extra requirement! There will be a finite set of coefficients $c(0), \dots, c(N)$ when $\phi(t)$ is supported on $[0, N]$. In general, $\phi(t)$ has infinite support and we need infinitely many $c(n)$.

To find $c(n)$, multiply the dilation equation (6.11) by $\sqrt{2}\phi(2t - n)$. Integrate and use orthogonality:

$$\sqrt{2} \int_{-\infty}^{\infty} \phi(t)\phi(2t - n) dt = c(n). \quad (6.12)$$

If $\phi(t)$ is the unit box and $\phi(2t)$ is the half-box, this gives $c(0) = \sqrt{2}/2$ and $c(1) = \sqrt{2}/2$. The dilation equation for the box function then has coefficients 1 and 1:

$$\phi(t) = \phi(2t) + \phi(2t - 1). \quad (6.13)$$

From orthogonality of the basis $\{\phi(t - k)\}$ we have double-shift orthogonality of the dilation coefficients $c(k)$. And unit energy in $\phi(t)$ gives a unit vector of c 's:

$$\text{Double-shift: } \sum c(k)c(k - 2m) = \delta(m). \quad \text{Unit vector: } \sum |c(k)|^2 = 1 \quad (6.14)$$

For proof, multiply the dilation equations for $\phi(t)$ and $\phi(t - m)$ and integrate. Orthonormality of the ϕ 's yields double-shift orthogonality of the c 's:

$$\int_{-\infty}^{\infty} \phi(t)\phi(t - m) dt = \sum_k c(k)c(k - 2m) = \delta(m). \quad (6.15)$$

The coefficients $c(k)$ go into an orthonormal filter bank! Starting with the spaces V_j in a multiresolution, the dilation equation has brought us back to filters—where the key matrix is $L = (\downarrow 2)C$. Double-shift orthogonality becomes $LL^T = I$. The rows of L contain the double shifts $L_{ij} = c(2i - j)$.

Box example:

$$L = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & 1 & 1 \\ & & & & \dots \end{bmatrix}$$

Daubechies example:

$$L = \begin{bmatrix} c(3) & c(2) & c(1) & c(0) & & & \\ & c(3) & c(2) & c(1) & c(0) & & \\ & & c(3) & c(2) & c(1) & c(0) & \\ & & & c(3) & c(2) & \dots & \\ & & & & & \dots & \end{bmatrix}.$$

To end this section, we have to identify the wavelet spaces W_j .

The Wavelet Equation

The scaling functions $\phi(2^j t - k)$ are orthogonal at each scale separately. But $\phi(t)$ is not orthogonal to $\phi(2t)$. They are *not* orthogonal across scales; the level j must be fixed. The function $\phi(t)$ in V_0 is also in V_1 (the dilation equation). Orthogonality *across scales* comes from the wavelet subspaces W_j and their basis functions $w_{jk}(t)$. We study those now, from three starting-points:

1. The spaces W_j .
2. The wavelets $w(t)$.
3. The coefficients $d(k)$.

Use Method 1 if you have the V_j . Their differences yield the spaces W_j . Use Method 2 if you can identify the wavelets. Just shift and rescale. Use Method 3 if you have the numbers $c(k)$. The *alternating flip* yields $d(k) = (-1)^k c(N - k)$. Then $w(t)$ comes from the wavelet equation below, and W_j contains the combinations of $w(2^j t - k)$.

The box function gives an example in which all three approaches will work. We construct W_0 and $w(t)$ and the d 's:

1. *From the subspaces:* V_0 contains constant functions on unit intervals, and V_1 contains constant functions on half-intervals. The space W_0 is in V_1 (therefore constant on half-intervals). It is orthogonal to V_0 , so the integral over each full interval is zero. This fact produces the complementary subspace W_0 , orthogonal to V_0 inside V_1 :

$$W_0 = \{ \text{constants on half-intervals with } f(n) + f(n + 1/2) = 0 \}.$$

$V_0 \oplus W_0$ does give V_1 . Combining *equal* values at n and $n + 1/2$ from V_0 with *opposite* values from W_0 gives *any* values $f(n)$ and $f(n + 1/2)$ for V_1 .

2. *From the wavelets:* The important function in W_0 is the up-down square wave:

$$\text{Haar wavelet } w(t) = \begin{cases} 1 & \text{for } 0 \leq t < \frac{1}{2} \\ -1 & \text{for } \frac{1}{2} \leq t < 1 \\ 0 & \text{otherwise.} \end{cases}$$

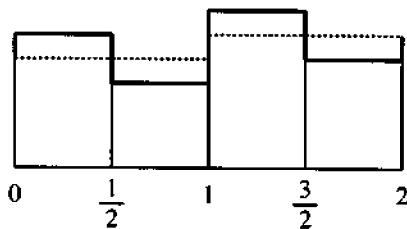


Figure 6.3: Two bases for V_1 : Halfsize boxes $\phi(2t - k)$ or full boxes $\phi(t - k)$ plus up-down Haar wavelets $w(t - k)$.

This is orthogonal to the box function $\phi(t)$. It is orthogonal to translates of ϕ and also to its own translates (there is no overlap of $w(t)$ with $w(t - 1)$). More than that, multiresolution says that the wavelet $w(t)$ is orthogonal to rescalings of itself and to translates of rescalings:

$$\int_{-\infty}^{\infty} w(t)w(2^j t - k) dt = 0 \text{ unless } j = k = 0.$$

The translates of $w(t)$ span W_0 . The translates of $w(2^j t)$ span W_j . Those wavelet spaces are orthogonal because $W_0 \subset V_j$ and $V_j \perp W_j$. (Exchange j and 0 if j is negative.) From orthogonal spaces we have orthogonal basis functions. Then completeness makes the whole orthonormal system $\{2^{j/2}w(2^j t - k)\}$ a basis for L^2 .

3. From the coefficients $c(0) = c(1) = 1/\sqrt{2}$: The flip construction gives $d(0) = 1/\sqrt{2}$ and $d(1) = -1/\sqrt{2}$. Those coefficients go into the wavelet equation:

$$\text{Wavelet equation } w(t) = \sqrt{2} \sum d(k) \phi(2t - k). \quad (6.16)$$

This equation produces the wavelet directly from the scaling functions—no equation to solve! The wavelet is $w(t) = \phi(2t) - \phi(2t - 1)$. This is a half-box minus a shifted half-box. It is the up-down square wave, which is Haar's wavelet.

Our final example, from Daubechies, starts with the four c 's. Then the flip construction gives the four d 's (to normalize, divide again by $4\sqrt{2}$):

$$d(0), d(1), d(2), d(3) = 1 - \sqrt{3}, -(3 - \sqrt{3}), 3 + \sqrt{3}, -(1 + \sqrt{3}).$$

Their sum is zero. Their sum of squares (normalized) is 1. They are orthogonal to their double shifts, because the c 's are. The wavelet equation gives the Daubechies wavelet $w(t)$, which has no simple formula. The orthogonality to $w(t - k)$ and $\phi(t - k)$ is only known indirectly—from the double-shift orthogonality of the d 's. The structure of multiresolution gives crucial information that we cannot find in a table of integrals.

The actual construction of $\phi(t)$ and $w(t)$, and the drawing of their graphs, is immediately ahead.

Example 6.2. (Strange but beautiful.) Suppose $\phi(t)$ is the delta function $\delta(t)$. This is not in L^2 but continue anyway. The space V_0 contains combinations $\sum a(n)\delta(t - n)$ of delta functions at the integers. What orthogonal wavelet $w(t)$ goes with this scaling function $\delta(t)$?

By scale invariance, V_1 contains $\delta(2t - n)$. The spikes for V_1 are at $t = 0, \pm\frac{1}{2}, \pm 1, \dots$. Since V_0 holds the delta functions at integers, W_0 contains delta functions at the midpoints $t = n + \frac{1}{2}$.

The integers and the midpoints combine to give $V_0 \oplus W_0 = V_1$. The wavelet is the delta function at $t = \frac{1}{2}$.

Similarly, V_j contains delta functions at $t = n/2^j$. W_j contains delta functions at the midpoints $(n + \frac{1}{2})/2^j$. What is W_{-1} ? Its delta functions are at $t = (n + \frac{1}{2})/2^{-1} = 2n + 1$. These are odd integers $\pm 1, \pm 3, \pm 5, \dots$. The spacing between them is 2 as expected. Then W_{-2} has delta functions at $\pm 2, \pm 6, \pm 10, \dots$ with spacing 4. The union of all W_j has delta functions at all binary points.

The dilation equation for the delta function is $\delta(t) = 2\delta(2t)$. The only nonzero coefficient is $h(0) = 1$. The filter is the identity. The wavelet equation with only one term is $w(t) = 2\delta(2t - 1) = \delta(t - \frac{1}{2})$. This confirms what we found, that W_0 contains delta functions at all midpoints between integers. Notice! An odd number of coefficients (one) means that $N = 0$ (even). The alternating flip must shift by an odd integer, for double-shift orthogonality. So the nonzero highpass coefficient was $d(1)$ not $d(0)$.

To linger one last second on this trivial great example, the double-shift matrices from the low and high channels are $L = (\downarrow 2)$ and $B = (\downarrow 2)(\text{delay})$.

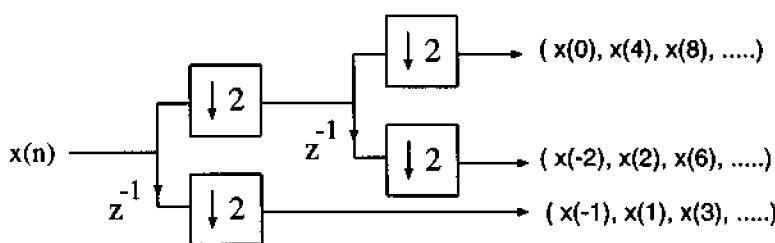


Figure 6.4: The lazy filter $H = I$ leads to delta functions.

The Scaling Function is Supported on $[0, N]$

A remarkable feature of $\phi(t)$ is that it is zero outside the interval $0 \leq t \leq N$. This could never happen to a *one-scale* difference or differential equation (homogeneous). The solutions would be combinations of λ^n and $e^{\lambda t}$, and only occasionally zero. The compact support of $\phi(t)$ comes from the two scales in the dilation equation

$$\phi(t) = \sum_{k=0}^N 2h(k) \phi(2t - k). \quad (6.17)$$

Theorem 6.1 The scaling function $\phi(t)$ is supported on the interval $[0, N]$.

Proof. Suppose we know that the support is a finite interval $[a, b]$. Then $\phi(2t)$ is supported on $[\frac{a}{2}, \frac{b}{2}]$. The shifted function $\phi(2t - k)$ is supported on $[\frac{a+k}{2}, \frac{b+k}{2}]$. The index k goes from zero to N , so the right side of the dilation equation is supported between $\frac{a}{2}$ and $\frac{b+N}{2}$. Comparing with the left side,

$$[a, b] = \left[\frac{a}{2}, \frac{b+N}{2} \right] \text{ leads to } a = 0 \text{ and } b = N.$$

How do we know that the support is a finite interval in the first place? From the cascade algorithm. The box function $\phi^{(0)}(t)$ is supported on $[0, 1]$. When this box is substituted into the right side of the dilation equation, the function $\phi^{(1)}(t)$ that comes out has support $[0, \frac{1+N}{2}]$. Then $\phi^{(1)}(t)$ is substituted into the right side and the result $\phi^{(2)}(t)$ is zero outside $[0, \frac{1+3N}{4}]$. The limiting function $\phi(t)$ is certain to be zero outside $[0, N]$. This cascade is studied in the next section.

It will be useful to reach the same conclusion based on the Fourier transform (Section 6.4). That argument can assume less about the filter coefficients. We mention that there are never gaps where $\phi(t)$ is zero on an interval inside $[0, N]$. And if the highpass coefficients $h_1(k)$ run from $k = 0$ to $k = \tilde{N}$, then the wavelet $w(t) = \sum 2h_1(k)\phi(2t - k)$ has support $[0, \frac{1}{2}(N + \tilde{N})]$. The last term $\phi(2t - \tilde{N})$ is zero after $2t - \tilde{N}$ reaches N .

Problem Set 6.1

1. Explain why the scaling requirement, that $f(t)$ is in V_j if and only if $f(2t)$ is in V_{j+1} , can be restated as $\hat{f}(\omega)$ is in \widehat{V}_j if and only if $\hat{f}(2\omega)$ is in \widehat{V}_{j-1} . Here \widehat{V}_j is the space of Fourier transforms of functions in V_j .
2. For the space V_0 of piecewise constant functions in Example 1, show that the only shift-invariant basis $\phi(t - k)$ contains box functions. What is the corresponding statement about allpass FIR filters?
3. For piecewise constants, show that $f(t)$ is in L^2 if and only if $f(n)$ is in l^2 .
4. Find 2 by 2 matrices $c(0)$ and $c(1)$ so that the box function $\phi_1(t)$ and sloping line $\phi_2(t) = 1 - 2t$ in Example 3 satisfy

$$\begin{bmatrix} \phi_1(t) \\ \phi_2(t) \end{bmatrix} = c(0) \begin{bmatrix} \phi_1(2t) \\ \phi_2(2t) \end{bmatrix} + c(1) \begin{bmatrix} \phi_1(2t - 1) \\ \phi_2(2t - 1) \end{bmatrix}.$$

5. If $f(t)$ is in V_0 and $g(t)$ is in V_1 , why is it generally false that $g(t) - f(t)$ is in W_1 ?
6. What multiresolution requirements are violated if W_j consists of all multiples of $\cos(2^j t)$?

6.2 Wavelets from Filters

The previous section reached the dilation equation and the wavelet equation:

$$\phi(t) = \sum \sqrt{2} c(n) \phi(2t - n) \quad \text{and} \quad w(t) = \sum \sqrt{2} d(n) \phi(2t - n). \quad (6.18)$$

Those equations are the crucial connections between wavelets and filters. Historically, their development was separate. Now you have to see them together. The lowpass filter $c(0), \dots, c(N)$ determines the scaling function $\phi(t)$. Then the highpass coefficients produce the wavelets.

Working with $\phi(t)$ and $w(t)$, we really have three basic jobs:

1. Compute the coefficients in $f_j(t) = \sum_k a_{jk} \phi_{jk}(t)$ and $f(t) = \sum_j \sum_k b_{jk} w_{jk}(t)$.
2. Construct $\phi(t)$ by actually solving the dilation equation.
3. Connect the properties of $\phi(t)$ and $w(t)$ to properties of the c 's and d 's.

This section will do part of each job, the *recursive part*. This shows how multiresolution (for functions) connects to subband filtering (for vectors). The three parts that we can do immediately are:

1. Compute a_{jk} and b_{jk} recursively from $a_{j+1,k}$ (and vice versa).
2. Set up a recursion (the cascade algorithm) to construct $\phi(t)$.
3. Prove orthogonality for $\phi_{jk}(t)$ and $w_{jk}(t)$ from orthogonality of c 's and d 's.

Those are the three subsections. Later we have to initialize the recursion in 1, execute and study the cascade algorithm in 2, and derive other properties in 3.

Wavelet Coefficients by Recursion

Suppose $f_1(t)$ is in V_1 . It is a combination of the basis functions $\sqrt{2}\phi(2t - k)$. These functions $\phi_{1k}(t)$ are at level 1. Multiresolution splits this level into $V_1 = V_0 \oplus W_0$, so $f_1(t)$ is also a combination of the basis functions for V_0 and W_0 . Those basis functions are $\phi_{0k}(t) = \phi(t - k)$ and $w_{0k}(t) = w(t - k)$:

$$\begin{aligned} \sum a_{1k} \phi_{1k}(t) &= \sum a_{0k} \phi_{0k}(t) + \sum b_{0k} w_{0k}(t) \\ &= \sum a_{0k} \phi(t - k) + \sum b_{0k} w(t - k). \end{aligned} \tag{6.19}$$

We are computing a change of basis. Given the coefficients $a_{1k}(t)$ in the V_1 basis, we want the coefficients a_{0k} and b_{0k} in the $V_0 \oplus W_0$ basis. The same step will apply at every level. It takes us from the coefficients $a_{j+1,k}$ in the basis for V_{j+1} , to the coefficients a_{jk} and b_{jk} in the bases for V_j and W_j . This is the recursion that makes the wavelet transform fast.

We will suppose that these bases are *orthonormal*. Later in this section we prove this property (assuming the cascade algorithm uses orthogonal filters and converges). Orthonormality makes the formulas easy and it makes the inverse easy. Section 6.5 will derive the biorthogonal recursion, when orthogonality is not assumed.

To find the recursion, shift equation (6.18) by k and set $n = \ell - 2k$:

$$\begin{aligned} \text{Dilation equation: } \phi(t - k) &= \sum \sqrt{2} c(n) \phi(2t - 2k - n) = \sum c(\ell - 2k) \phi_{1\ell}(t) \\ \text{Wavelet equation: } w(t - k) &= \sum \sqrt{2} d(n) \phi(2t - 2k - n) = \sum d(\ell - 2k) \phi_{1\ell}(t) \end{aligned} \tag{6.20}$$

Multiply by $f_1(t)$ and integrate with respect to t . Since the basis functions are orthonormal, the integral gives the coefficients of $f_1(t)$ in each basis:

$$a_{0k} = \sum c(\ell - 2k) a_{1\ell} \quad \text{and} \quad b_{0k} = \sum d(\ell - 2k) a_{1\ell}. \tag{6.21}$$

This is the key recursion. It is the action of a filter bank, which inputs $a_{1\ell}$ and outputs a_{0k} and b_{0k} . But we have to watch indices, because an ordinary convolution would be $\sum c(k - \ell) a_{1\ell}$ and downsampling would give $\sum c(2k - \ell) a_{1\ell}$. There is a time reversal between this filter C and the transpose filter C^T that appears in the recursion (6.21):

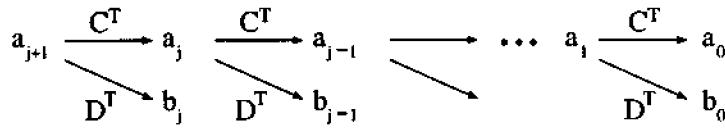
$$c^T(n) = c(-n) \quad \text{and} \quad d^T(n) = d(-n). \tag{6.22}$$

Going between levels of a multiresolution is subband filtering with C^T and D^T :

Theorem 6.2 A function $\sum a_{j+1,\ell} \phi_{j+1,\ell}(t)$ in the space $V_{j+1} = V_j \oplus W_j$ has coefficients a_{jk} and b_{jk} in the new orthonormal basis $\{\phi_{jk}(t), w_{jk}(t)\}$:

$$a_{jk} = \sum_\ell c(\ell - 2k) a_{j+1,\ell} \quad \text{and} \quad b_{jk} = \sum_\ell d(\ell - 2k) a_{j+1,\ell}. \quad (6.23)$$

In vector notation this is $\mathbf{a}_j = (\downarrow 2) C^T \mathbf{a}_{j+1}$ and $\mathbf{b}_j = (\downarrow 2) D^T \mathbf{a}_{j+1}$. The pyramid is



Proof. For $j = 0$, formula (6.23) is (6.21). The extension to every j comes from the dilation equation. Again $n = \ell - 2k$:

$$2^{j/2} \phi(2^j t - k) = 2^{j/2} \sum_n \sqrt{2} c(n) \phi(2^{j+1} t - 2k - n) = \sum_\ell c(\ell - 2k) \phi_{j+1,\ell}(t). \quad (6.24)$$

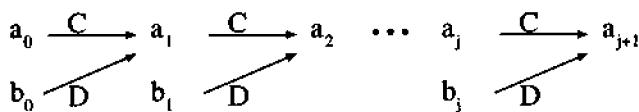
The wavelet equation has d in place of c . The inner products of these equations with $f(t)$ give the recursions (6.23) for the coefficients a_{jk} and b_{jk} .

Now go in the opposite direction. Change from the basis $\{\phi_{jk}(t), w_{jk}(t)\}$ back to the basis $\{\phi_{j+1,\ell}(t)\}$. Since the bases are orthonormal, the inverse operation is given by the transpose.

Theorem 6.3 $a_{j+1,\ell}$ comes from a_{jk} and b_{jk} by a synthesis filter bank:

$$a_{j+1,\ell} = \sum_n c(2k - \ell) a_{jk} + d(2k - \ell) b_{jk}. \quad (6.25)$$

The inverse pyramid is the fast inverse wavelet transform:



Lowpass Iteration and the Cascade Algorithm

We begin the solution of the dilation equation. Our goal is to construct the scaling function $\phi(t)$. The only inputs are the filter coefficients $c(0), \dots, c(N)$. The first solution method we propose is the **cascade algorithm**.

Start the cascade with $\phi^{(0)}(t) = \text{box function on } [0, 1]$. Iterate the lowpass filter:

$$\phi^{(i+1)}(t) = \sum_n \sqrt{2} c(n) \phi^{(i)}(2t - n) = \sum_n 2 h(n) \phi^{(i)}(2t - n). \quad (6.26)$$

The algorithm works with functions in continuous time. Those functions are piecewise constant and the pieces become shorter (their length is 2^{-i}). If $\phi^{(i)}(t)$ converges suitably to a limit $\phi(t)$, then this limit function solves the dilation equation.

Notice the two time scales, t and $2t$, which come from the continuous form of downsampling. In place of $(\downarrow 2)\phi(n) = \phi(2n)$, we have $(\downarrow 2)\phi(t) = \phi(2t)$. The cascade algorithm is really iteration with the filter matrix $M = (\downarrow 2)2H$ —as we will see in detail. It is an infinite iteration, and our final formula for $\phi(t)$ will involve an infinite product.

It is easy to associate a continuous-time function $x(t)$ with a discrete-time vector $x(n)$. The function takes the value $x(n)$ over the n^{th} time interval. That is the interval $n \leq t < n+1$. Thus the constant vector $x = (\dots, 1, 1, 1, \dots)$ produces the constant function $x(t) \equiv 1$. The impulse $x = (\dots, 0, 1, 0, \dots)$ produces the standard *box function*. In general $x(t)$ is *piecewise constant*: $x(t) = x(n)$ on the interval $n \leq t < n+1$.

The iterations start from the box function $\phi^{(0)}(t)$. There are two steps in each iteration—*filtering* and *rescaling*. Suppose the filter coefficients are $h(0) = 2/3$ and $h(1) = 1/3$. Filtering the input gives $\frac{2}{3}\phi^{(0)}(t) + \frac{1}{3}\phi^{(0)}(t-1)$. Then rescaling t to $2t$ compresses the graph. To maintain a constant area we multiply the height by 2:

$$\phi^{(1)}(t) = \frac{4}{3}\phi^{(0)}(2t) + \frac{2}{3}\phi^{(0)}(2t-1).$$

Filtering and rescaling one box produces two half-width boxes of height $\frac{4}{3}$ and $\frac{2}{3}$. That iteration step preserves the area (=1). Now filter and rescale $\phi^{(1)}(t)$. The two half-boxes become four quarter-boxes, from $\phi^{(2)}(t) = \frac{4}{3}\phi^{(1)}(2t) + \frac{2}{3}\phi^{(1)}(2t-1)$. The first quarter-box has height $\frac{16}{9}$. That height is multiplied by $\frac{4}{3}$ at every iteration!

We wish we could say that the iterations $\phi^{(i)}(t)$ are converging. Their limit $\phi(t)$ would satisfy the dilation equation $\phi(t) = \frac{4}{3}\phi(2t) + \frac{2}{3}\phi(2t-1)$. In some weak sense, this may be true. In a pointwise sense at $t = 0$, the functions $\phi^{(i)}(0)$ diverge because of $(4/3)^i$. The coefficients 2/3 and 1/3 illustrate the iteration process, but not its convergence.

We want to see that process also by algebra. It is clearest if we ignore the rescaling and just execute the filtering with coefficients $h(k)$. The heights of the boxes would be $\frac{2}{3}, \frac{1}{3}$, and then $\frac{4}{9}, \frac{2}{9}, \frac{1}{9}$. In the z -domain, this corresponds to

$$H(z) = \frac{2}{3} + \frac{1}{3}z^{-1} \quad \text{and} \quad H(z^2)H(z) = \frac{4}{9} + \frac{2}{9}z^{-1} + \frac{2}{9}z^{-2} + \frac{1}{9}z^{-3}. \quad (6.27)$$

The actual time intervals go from length 1 to $\frac{1}{2}$ to $\frac{1}{4}$. The actual graph heights are doubled at each step, to preserve area. But the essential point is the product $H(z^2)H(z)$. After three steps, the iteration will produce $H^{(3)}(z) = H(z^4)H(z^2)H(z)$. After i steps we have

$$H^{(i)}(z) = \prod_{k=0}^{i-1} H(z^{2^k}). \quad (6.28)$$

This product is the z -domain equivalent of iterating the lowpass filter $H(z)$. The values of $\phi^{(i)}(t)$ —the heights of the graph after i iterations— are the coefficients of $2^i H^{(i)}(z)$. That factor 2^i accounts for the height-doublings that preserve area, when the time intervals for $\phi^{(i)}(t)$ become 2^{-i} .

You may ask, why not choose the usual averaging filter as a first example? Let me show you why. The averaging coefficients are $h(0) = h(1) = \frac{1}{2}$. The first step of the iteration, with coefficients $2h(0) = 2h(1) = 1$, is

$$\phi^{(1)}(t) = \phi^{(0)}(2t) + \phi^{(0)}(2t-1).$$

From the box function $\phi^{(0)}(t)$ this produces the same box: $\phi^{(1)}(t) = \phi^{(0)}(t)$.

The output equals the input. The iteration process converges immediately. We have found the scaling function! In general $\phi(t)$ is the limit of the sequence $\phi^{(i)}(t)$, when that limit exists as $i \rightarrow \infty$. Here $\phi^{(0)} = \phi^{(1)}$ and the box function is a “fixed point” of the iteration. When we filter and rescale $\phi(t)$ we get back $\phi(t)$, because the sum of two half-length boxes is the original box:

$$\text{Box: } \phi(t) = \phi(2t) + \phi(2t - 1). \quad (6.29)$$

The z -domain equivalent is a product built from

$$H(z) = \frac{1}{2} + \frac{1}{2}z^{-1}.$$

Please notice that we *do not square* this function. $H^{(2)}(z)$ is $H(z^2)H(z)$:

$$H^{(2)}(z) = (\frac{1}{2} + \frac{1}{2}z^{-2})(\frac{1}{2} + \frac{1}{2}z^{-1}) = \frac{1}{4} + \frac{1}{4}z^{-1} + \frac{1}{4}z^{-2} + \frac{1}{4}z^{-3}. \quad (6.30)$$

After i iterations, $H^{(i)}(z)$ will have 2^i coefficients all equal to 2^{-i} . After rescaling, this still corresponds to the box function.

Now use three filter coefficients $h = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$. The box $\phi^{(0)}(t)$ produces *three* half-boxes in

$$\phi^{(1)}(t) = \frac{1}{2}\phi^{(0)}(2t) + \phi^{(0)}(2t - 1) + \frac{1}{2}\phi^{(0)}(2t - 2).$$

Then there are *seven* quarter-boxes in $\phi^{(2)}(t)$. Rescaling prevents the support interval from becoming long. The limiting interval is $0 \leq t < 2$.

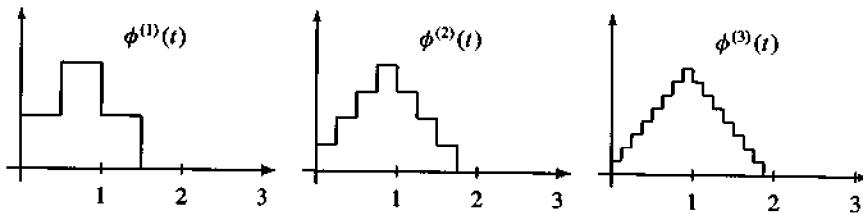


Figure 6.5: The cascade algorithm for $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ converges to the hat function.

A reasonable guess for the limiting function $\phi(t)$ is the *hat function*. This is piecewise linear, going up to $\phi(1) = 1$ and down to $\phi(2) = 0$. We verify that the hat function is a fixed point of the iteration. Filtering and rescaling leaves this scaling function $\phi(t)$ unchanged:

$$\phi(t) = \frac{1}{2}\phi(2t) + \phi(2t - 1) + \frac{1}{2}\phi(2t - 2). \quad (6.31)$$

Notice how the coefficients $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ are doubled. The hat function is a combination of three narrower hats. For future reference, we note the different properties of these examples:

1. $H(z) = \frac{2}{3} + \frac{1}{3}z^{-1}$ is not zero at $z = -1$, corresponding to $\omega = \pi$. The iterations fail to converge.
2. $H(z) = \frac{1}{2} + \frac{1}{2}z^{-1}$ is zero at $z = -1$. The iterations converge. The filter $H(z)H(z^{-1})$ is halfband: no even powers except the constant term. The box function is orthogonal to its translates.

3. $H(z) = \frac{1}{4} + \frac{1}{2}z^{-1} + \frac{1}{4}z^{-2}$ is zero (twice) at $z = -1$. The iterations converge. The filter $H(z)H(z^{-1})$ is *not* halfband. It contains the even powers z^2 and z^{-2} . The hat function $\phi(t)$ is *not* orthogonal to $\phi(t-1)$.

We must quickly emphasize that a zero at $z = -1$ (which is $\omega = \pi$ in the frequency domain) does not guarantee the convergence of $\phi^{(i)}(t)$. But without that zero in the filter response, strong convergence has no chance.

Similarly, a halfband filter does not guarantee that $\phi(t)$ is orthogonal to its translates. But without that halfband property of $H(z)H(z^{-1})$, orthogonality has no chance. Section 7.2 will further indicate those connections; they are not quite two-way implications:

$$\begin{aligned} \text{Convergence of } \phi^{(i)}(t) \text{ to } \phi(t) \text{ needs } H = 0 \text{ at } z = -1 \\ \text{Orthogonality of } \phi(t-k) \text{ needs } H(z)H(z^{-1}) \text{ to be halfband} \end{aligned}$$

Orthogonal Functions from Orthogonal Filters

When the filter bank is orthonormal in discrete time, we hope for orthogonal basis functions in continuous time. All wavelets $w(2^j t)$ should be orthogonal to the scaling functions $\phi(t-k)$. Furthermore, the wavelets $w(2^j t - k)$ should be mutually orthogonal and the scaling functions $\phi(t-k)$ should be mutually orthogonal. Note that $\phi(t)$ is *not* orthogonal to $\phi(2t)$.

Theorem 6.4 *Assume that the cascade algorithm converges: $\phi^{(i)}(t) \rightarrow \phi(t)$ uniformly in t . If the coefficients $c(k)$ and $d(k)$ come from an orthonormal filter bank, so they have double-shift orthogonality; then*

1. *The scaling functions $\phi(t-n)$ are orthonormal to each other:*

$$\int_{-\infty}^{\infty} \phi(t-n)\phi(t-m) dt = \delta(m-n).$$

2. *The scaling functions are orthogonal to the wavelets:*

$$\int_{-\infty}^{\infty} \phi(t-m)w(t-n) dt = 0.$$

3. *The wavelets $w_{jk}(t) = 2^{j/2}w(2^j t - k)$ at all scales are orthonormal:*

$$\int_{-\infty}^{\infty} w_{jk}(t)w_{lkm}(t) dt = \delta(j-l)\delta(k-m).$$

Proof of 1: The box functions $\phi^{(0)}(t-k)$ are certainly orthonormal (because nonoverlapping). We will show that when $\phi^{(i)}(t-k)$ are orthogonal, the next iterates $\phi^{(i+1)}(t-k)$ are also orthonormal. Then the limits $\phi(t-k)$ are orthonormal.

The induction step from i to $i+1$ assumes that the $\phi^{(i)}(t-k)$ are orthonormal, and sets $l = m - n$:

$$\begin{aligned}
& \int \phi^{(i+1)}(t-m)\phi^{(i+1)}(t-n) dt \\
= & 2 \int (\sum c(k)\phi^{(i)}(2t-2m-k)) (\sum c(k)\phi^{(i)}(2t-2n-k)) dt \quad (6.32) \\
= & \int (\sum c(k)\phi^{(i)}(2t-2m-k)) (\sum c(k-2l)\phi^{(i)}(2t-2m-k)) 2dt \\
= & \sum c(k)c(k-2l) = \delta(l) = \delta(m-n).
\end{aligned}$$

The crucial step came in the last line, when we used the orthogonality of the row $[c(0) \dots c(N)]$ to its double shifts. These are rows of $L = (\downarrow 2)C$. The orthogonality is in the statement $LL^T = I$. Equivalently, it is in the statement that $|\sum c(k)e^{-ik\omega}|^2$ is a normalized halfband filter: no even powers except the constant term 1.

Note the important point! Orthogonality of wavelets came from orthogonality of filters. When the infinite iterations converge, the limits retain orthogonality. This holds at each scale level j . In $\int \phi(2^j t - m)\phi(2^j t - n) dt$, we replace $2^j t$ by T . Orthogonality *does not hold* between scaling functions at different levels. Certainly, $\phi(t)$ is not orthogonal to all $\phi(2t - n)$, or the dilation equation would require $\phi \equiv 0$.

Proof of 2: Repeat the integration steps above for ϕ times w :

$$\begin{aligned}
& \int \phi(t-m)w(t-n) dt \\
= & \int (\sum c(k)\sqrt{2}\phi(2t-2m-k)) (\sum d(k)\sqrt{2}\phi(2t-2n-k)) dt \\
= & \dots = \sum c(k)d(k-2l) = 0.
\end{aligned}$$

Always, $l = m - n$. The last step uses the orthogonality of the rows of $L = (\downarrow 2)C$ to the rows of $B = (\downarrow 2)D$. Again the double shift is essential. It is false that *all* rows of C and D are orthogonal.

The matrix form of this double-shift orthogonality is $LB^T = 0$. It comes from the alternating flip. That choice always produces double-shift orthogonality of d 's to c 's, but it does not by itself make $w(t)$ orthogonal to $\phi(t)$. To reach the end of part 2, we needed part 1—orthogonality between the ϕ 's.

Proof of 3: The orthogonality of wavelets $w_{jk}(t)$ at the same scale level (the same j) is proved as in parts 1 and 2:

$$\int_{-\infty}^{\infty} w(t-m)w(t-n) dt = \dots = \sum d(k)d(k-2l) = \delta(l) = \delta(m-n).$$

Again continuous time orthogonality follows from discrete time orthogonality. This is not $DD^T = I$. It is $BB^T = I$, with double shifts in the rows of $B = (\downarrow 2)D$.

The orthogonality of wavelets at different scale levels (different j) is immediate from the rules of multiresolution. Suppose $j < J$. Then W_J is orthogonal to V_J by part 2. But W_j is contained in V_{j+1} and therefore in V_J . So W_j is orthogonal to W_J . This proves the orthogonality theorem.

Final note: It was convenient to start from the box function $\phi^{(0)}(t)$, which is orthogonal to its translates. Then an orthogonal filter bank maintains this orthogonality to translates. Other starting functions will lead to the same fixed point $\phi(t)$, or at least to a multiple $c\phi(t)$ —if strong convergence holds.

In general, convergence can be “weak” or “strong”. For weak convergence, the functions $\phi^{(i)}(t)$ can oscillate faster and faster. You would not call this convergence. But the *integral* of $\phi^{(i)}(t)$ converges to the integral of $\phi(t)$, on every fixed interval $[0, T]$. (Integration controls the oscillations.) In the convergence that we assumed, $\phi^{(i)}(t)$ approaches $\phi(t)$ at every point.

There is a better starting function $\phi^{(0)}(t)$ than the box. The constant value $\phi^{(0)}(n)$ on each interval $n \leq t < n + 1$ can be the *correct* $\phi(n)$. The values of ϕ are filled in at half-integers and quarter-integers by the iterations $\phi^{(1)}(t)$ and $\phi^{(2)}(t)$. The graph of $\phi(t)$ appears, 2^i points at a time. We stop when we have enough points for the printer to connect into a continuous graph.

The next section explains how to start with the correct values of $\phi(n)$ at the integers.

Problem Set 6.2

1. For the filter with $h(0) = h(1) = \frac{1}{2}$ and any $\phi^{(0)}(t)$, describe and draw $\phi^{(i)}(t)$.
2. If $H(z)$ is a polynomial of degree N , what is the degree of $H(z^2)H(z)$? What is the degree of $H^{(i)}(z) = \prod_{k=0}^{i-1} H(z^{2^k})$?
Rescaling will replace z by $z^{1/2}$. After i steps, the degree is divided by 2^i . Show that the degree of $H^{(i)}(z^{1/2^i})$ approaches N as $i \rightarrow \infty$.
3. With coefficients $h(0), \dots, h(N)$, the support interval of $\phi^{(i)}(t)$ grows to $[0, N]$. What happens if $\phi^{(0)}(t)$ is a box on $[0, 2N]$?
4. The unit area of the box is preserved if and only if $h(0) + \dots + h(N) = 1$. Are negative coefficients allowed?
5. Suppose the filter coefficients $h(k)$ are $\frac{1}{2}, 0, 0, \frac{1}{2}$. Starting from the box function, take one step of the cascade algorithm and draw $\phi^{(1)}(t)$. Then take a second step and draw $\phi^{(2)}(t)$. Describe $\phi^{(i)}(t)$ — on what fraction of the interval $[0, 3]$ does $\phi^{(i)}(t) = 1$?
6. Suppose the only filter coefficient is $h(0) = 1$. Starting from the box function $\phi^{(0)}(t)$, draw the graphs of $\phi^{(1)}(t)$ and $\phi^{(2)}(t)$. In what sense does $\phi^{(i)}(t)$ converge to the delta function $\delta(t)$? To verify the dilation equation $\delta(t) = 2\delta(2t)$, multiply by any smooth $f(t)$ and compare the integrals of both sides.
7. Suppose $\phi^{(0)}(t)$ is a stretched box of unit area: $\phi^{(0)}(t) = 1/2$ for $0 \leq t < 2$. Draw the graphs of $\phi^{(1)}(t)$ and $\phi^{(2)}(t)$ when $h(0) = h(1) = 1/2$. On what interval is $\phi^{(i)}(t)$ nonzero? What is the limit $\phi(t)$?
8. Suppose $\phi^{(0)}(t)$ is the Haar wavelet with zero area:

$$\phi^{(0)}(t) = 1 \text{ for } 0 \leq t < 1/2 \text{ and } \phi^{(0)}(t) = -1 \text{ for } 1/2 \leq t < 1.$$

With $h(0) = h(1) = 1/2$, draw the graphs of $\phi^{(1)}(t)$ and $\phi^{(2)}(t)$. The sequence $\phi^{(i)}(t)$ converges “weakly” to what multiple $c\phi(t)$?

6.3 Computing the Scaling Function by Recursion

The main point of this section can be stated in three sentences. Then you can follow through on the details, or look ahead for the matrices $m(0)$ and $m(1)$:

The dilation equation gives $\phi(0), \phi(1), \dots$ as the eigenvector of a matrix $m(0)$.

Then $\phi(t)$ at $t = \text{half-integers}$ comes from multiplying by a matrix $m(1)$.

Then $\phi(t)$ at every dyadic t comes by recursion. Each step uses $m(0)$ or $m(1)$.

The scaling function is created recursively. This section gives the rule.

The dilation equation is easiest with only *two coefficients* ($N = 1$). Then $\mathbf{m}(0) = 2\mathbf{h}(0)$ and $\mathbf{m}(1) = 2\mathbf{h}(1)$ are scalars not matrices. The two-coefficient dilation equation is

$$\phi(t) = \mathbf{m}(0)\phi(2t) + \mathbf{m}(1)\phi(2t - 1). \quad (6.33)$$

The solution will be zero outside the interval $0 \leq t < 1$. Inside that interval, set $t = 0$ to find $\phi(0)$ and $t = \frac{1}{2}$ to find $\phi(\frac{1}{2})$:

$$\phi(0) = \mathbf{m}(0)\phi(0) \quad \text{and} \quad \phi\left(\frac{1}{2}\right) = \mathbf{m}(1)\phi(0).$$

Now set $t = \frac{1}{4}$ and $\frac{3}{4}$, then $\frac{5}{8}$ and $\frac{7}{8}$, and onward through all the dyadic points $t = n/2^i$. Directly from the equation you find

$$\begin{aligned} \phi\left(\frac{1}{4}\right) &= \mathbf{m}(0)\phi\left(\frac{1}{2}\right) \quad \text{and} \quad \phi\left(\frac{3}{4}\right) = \mathbf{m}(1)\phi\left(\frac{1}{2}\right) \\ \phi\left(\frac{1}{8}\right) &= \mathbf{m}(0)\phi\left(\frac{1}{4}\right) \quad \text{and} \quad \phi\left(\frac{5}{8}\right) = \mathbf{m}(1)\phi\left(\frac{1}{4}\right) \\ \phi\left(\frac{3}{8}\right) &= \mathbf{m}(0)\phi\left(\frac{3}{4}\right) \quad \text{and} \quad \phi\left(\frac{7}{8}\right) = \mathbf{m}(1)\phi\left(\frac{3}{4}\right). \end{aligned}$$

Each new value comes from multiplying a previous value by $\mathbf{m}(0)$ or $\mathbf{m}(1)$. At each time t , the right side of equation (6.33) has only *one* nonzero term. Thus $\phi(\frac{3}{4})$ equals $\mathbf{m}(1)\phi(\frac{1}{2})$ which is $\mathbf{m}(1)\mathbf{m}(1)\phi(0)$.

At the next step $\phi(\frac{3}{8})$ equals $\mathbf{m}(0)\mathbf{m}(1)\mathbf{m}(1)\phi(0)$. The key is in the order of $\mathbf{m}(0)$ and $\mathbf{m}(1)$. It is the same order as in the binary expansions $\frac{3}{4} = 0.11$ and $\frac{3}{8} = 0.011$. At any point $t = n/2^i$, the solution $\phi(t)$ has i factors:

$$\text{If } t = 0.01101 \text{ in base 2, then } \phi(t) = \mathbf{m}(0)\mathbf{m}(1)\mathbf{m}(1)\mathbf{m}(0)\mathbf{m}(1)\phi(0).$$

We have now solved the two-coefficient dilation equation at all dyadic points.

Admittedly, the restriction to two coefficients looks severe. The pattern is correct and important, but two numbers $\mathbf{m}(0)$ and $\mathbf{m}(1)$ are not enough. The only normal case is $\mathbf{m}(0) = \mathbf{m}(1) = 1$, when we get the box function. For $\mathbf{m}(0) = \frac{4}{3}$ and $\mathbf{m}(1) = \frac{2}{3}$, the first equation becomes $\phi(0) = \frac{4}{3}\phi(0)$. This produces a singularity of $\phi(t)$ at all dyadic points.

We will not pursue that example here, because there is a more valuable application — which reduces $N + 1$ coefficients to two. This is the familiar step of reducing a high-order equation to a low-order system. For differential equations that produces a matrix, as in the system $u' = Au$. For dilation equations the reduction will produce *two matrices* $\mathbf{m}(0)$ and $\mathbf{m}(1)$. The dilation equation will become a *two-coefficient matrix equation*. The recursion will not change, except it has vectors and matrices.

Vector Form of the Dilation Equation

The $N + 1$ coefficients in the dilation equation are $\sqrt{2}c(k) = 2\mathbf{h}(k)$:

$$\phi(t) = 2 \sum_{k=0}^N \mathbf{h}(k) \phi(2t - k). \quad (6.34)$$

Outside the interval $0 \leq t < N$, we want and expect $\phi(t) \equiv 0$. Inside that interval, substitute $t = 0, t = 1, \dots, t = N - 1$ to determine $\phi(t)$ at the integers. You will see again the crucial

fact that even k goes with even $2t - k$, and odd k goes with odd $2t - k$. (Reason! *The sum of k and $2t - k$ is even.*) The right side of (6.34) leads to an N by N matrix $\mathbf{m}(0)$ which is displayed for $N = 5$:

$$\begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \\ \phi(4) \end{bmatrix} = 2 \begin{bmatrix} h(0) & & & & \\ h(2) & h(1) & h(0) & & \\ h(4) & h(3) & h(2) & h(1) & h(0) \\ & h(5) & h(4) & h(3) & h(2) \\ & & h(5) & h(4) & \end{bmatrix} \begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \\ \phi(4) \end{bmatrix} = \mathbf{m}(0)\Phi(0). \quad (6.35)$$

This is the dilation equation restricted to the integers. It is an eigenvalue problem for $\mathbf{m}(0)$. That matrix has even and odd in separate columns. For a nontrivial solution, this matrix (including the factor 2) must have $\lambda = 1$ as an eigenvalue. Assume this is true. Then the values $\phi(n)$ are in the eigenvector (*which we call $\Phi(0)$*). That eigenvalue problem for $\mathbf{m}(0)$ sets the integer values $\phi(n)$, and the recursion starts.

Now look at the vector of half-integer values. Substitute $t = \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}, \frac{9}{2}$ into the dilation equation. This leads to a closely related matrix $\mathbf{m}(1)$. The first row comes from $\phi(\frac{1}{2}) = 2h(1)\phi(0) + 2h(0)\phi(1)$. Notice that $2t$ is an *odd* integer, so *the sum of k and $2t - k$ is now odd*. The matrix is $\mathbf{m}(1)$:

$$\begin{bmatrix} \phi(1/2) \\ \phi(3/2) \\ \phi(5/2) \\ \phi(7/2) \\ \phi(9/2) \end{bmatrix} = 2 \begin{bmatrix} h(1) & h(0) & & & \\ h(3) & h(2) & h(1) & h(0) & \\ h(5) & h(4) & h(3) & h(2) & h(1) \\ & h(5) & h(4) & h(3) & \\ & & h(5) & & \end{bmatrix} \begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \\ \phi(4) \end{bmatrix} = \mathbf{m}(1)\Phi(0). \quad (6.36)$$

As expected, the values at half-integers come from the values at integers. A vector $\Phi(\frac{1}{2})$ comes from a vector $\Phi(0)$. In matrix notation (6.35) was an eigenproblem for $\Phi(0)$ and (6.36) is the step to $\Phi(\frac{1}{2})$:

$$\Phi(0) = \mathbf{m}(0)\Phi(0) \quad \text{and} \quad \Phi(\frac{1}{2}) = \mathbf{m}(1)\Phi(0).$$

This is exactly like the two-coefficient case! Now $\mathbf{m}(0)$ and $\mathbf{m}(1)$ are $N \times N$ matrices. The beautiful fact is that the same pattern continues to quarter-integers and beyond.

When t is a quarter-integer, the times $2t - k$ are half-integers. The values $\phi(\frac{1}{4}), \phi(\frac{3}{4}), \dots$ come from $\phi(\frac{1}{2}), \phi(\frac{3}{2}), \dots$. The dilation equation connects those vectors by the matrix $\mathbf{m}(0)$. Similarly the values $\phi(\frac{3}{4}), \phi(\frac{7}{4}), \dots$ come from multiplying those half-integer values in the vector $\Phi(\frac{1}{2})$ by the matrix $\mathbf{m}(1)$:

$$\Phi(\frac{1}{4}) = \mathbf{m}(0)\Phi(\frac{1}{2}) \quad \text{and} \quad \Phi(\frac{3}{4}) = \mathbf{m}(1)\Phi(\frac{1}{2}).$$

Exactly as before, the binary expansion of $t = n/2^l$ reveals the order of the factors $\mathbf{m}(0)$ and $\mathbf{m}(1)$ —as they multiply the initial eigenvector $\Phi(0)$ of values at the integers. We describe the recursion and then prove it is correct.

Theorem 6.5 *The vector form of the dilation equation is*

$$\Phi(t) = \mathbf{m}(0)\Phi(2t) + \mathbf{m}(1)\Phi(2t - 1). \quad (6.37)$$

The vector $\Phi(t)$ is zero outside the interval $0 \leq t \leq 1$. Its components are the N slices $\phi(t)$, $\phi(t+1)$, $\phi(t+2)$, ... of the scaling function. Substituting $t = 0$, the values $\phi(n)$ at the integer times $t = 0, 1, \dots, N-1$ are in the eigenvector of $m(0)$ with $\lambda = 1$:

$$\text{(Fixed point)} \quad \Phi(0) = m(0)\Phi(0). \quad (6.38)$$

The vector $\Phi(t)$ of values at the dyadic points $t, t+1, \dots, t+N-1$ comes from i multiplications by $m(0)$ and $m(1)$. Here $t = n/2^i < 1$ with n odd:

$$\text{If } t = \frac{3}{8} = 0.011 \text{ then } \Phi(t) = \begin{bmatrix} \phi(t) \\ \phi(t+1) \\ \vdots \\ \phi(t+N-1) \end{bmatrix} = m(0)m(1)m(1)\Phi(0). \quad (6.39)$$

The scalar equation of high order is reduced to a vector equation of low order. It is just a recursion, in which the 0-1 digit t_1 tells whether to use $m(0)$ or $m(1)$:

$$\text{Vector recursion: } \Phi(. t_1 t_2 t_3 \dots) = m(t_1) \Phi(. t_2 t_3 t_4 \dots). \quad (6.40)$$

The vector $\Phi(2t)$ is nonzero on the half-interval $0 \leq t < \frac{1}{2}$. The other vector $\Phi(2t-1)$ is nonzero for $\frac{1}{2} \leq t < 1$. These two vectors of compressed slices are multiplied by $m(0)$ and $m(1)$. To identify those particular matrices as correct, one way is to substitute t into the dilation equation and watch the numbers $2t - k$. As t crosses $\frac{1}{2}$, those numbers cross an integer — they go from one slice to the next. At that moment $m(0)$ is exchanged for $m(1)$.

The matrix $m(0)$ has the same “double-shift” between rows that we saw earlier in filter banks. The earlier matrix was $L = (\downarrow 2)C$, with entries $L_{ij} = c(2i-j)$. Now this matrix L appears in the dilation equation! It is multiplied by the extra factor $\sqrt{2}$ to become M . Its entries are $2h(2i-j)$:

$$M = \sqrt{2}L = 2 \begin{bmatrix} \cdots & h(0) & & & & \\ \cdots & h(2) & h(1) & h(0) & & \\ \cdots & h(4) & h(3) & h(2) & h(1) & h(0) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}. \quad (6.41)$$

We now show that the dilation equation has an even more compact form $\Phi_\infty(t) = M\Phi_\infty(2t)$. These are infinite vectors and matrices. The nonzero part will be exactly the two-term form of the dilation equation.

Dilation Equation in Infinite Vector Form $\Phi_\infty(t) = M\Phi_\infty(2t)$

$$\text{This form is } \begin{bmatrix} \vdots \\ \phi(t-1) \\ \phi(t) \\ \phi(t+1) \\ \vdots \end{bmatrix} = M \begin{bmatrix} \vdots \\ \phi(2t-1) \\ \phi(2t) \\ \phi(2t+1) \\ \vdots \end{bmatrix} \text{ for } -\infty < t < \infty \quad (6.42)$$

Restricted to the interval $0 \leq t < 1$, only rows $0, 1, \dots, N-1$ of $\Phi_\infty(t)$ are nonzero. Those N slices of $\phi(t)$ form the vector $\Phi(t)$ with N components. The dilation equation (6.42) reduces to

the vector form (6.37):

$$\Phi(t) = \begin{cases} \mathbf{m}(0)\Phi(2t) & \text{for } 0 \leq t < \frac{1}{2} \\ \mathbf{m}(1)\Phi(2t-1) & \text{for } \frac{1}{2} \leq t < 1 \end{cases} = \mathbf{m}(0)\Phi(2t) + \mathbf{m}(1)\Phi(2t-1).$$

The matrices $\mathbf{m}(0)$ and $\mathbf{m}(1)$ are N by N sections of the infinite matrix \mathbf{M} . For $i, j = 0, 1, \dots, N-1$ the matrix entries are

$$\mathbf{m}(0)_{ij} = M_{ij} = 2h(2i-j) \quad \text{and} \quad \mathbf{m}(1)_{ij} = M_{i,j-1} = 2h(2i-j+1).$$

Proof. The verification is in three steps, first for \mathbf{M} and then $\mathbf{m}(0)$ and then $\mathbf{m}(1)$. I hope the intuition is already in place, to see the sum $\sum 2h(k)\phi(2t-k)$ as $\mathbf{M}\Phi(2t)$ or $\sqrt{2}\mathbf{L}\Phi(2t)$ or $2(\downarrow 2)\mathbf{H}\Phi(2t)$. We now follow each step:

(Verify \mathbf{M}) Row zero of $\Phi_\infty(t) = \mathbf{M}\Phi_\infty(2t)$ is $\phi(t) = 2 \sum h(k)\phi(2t-k)$.

Row one is $\phi(t+1) = 2 \sum h(k)\phi(2t+2-k)$. The double shift works.

(Verify $\mathbf{m}(0)$) Restrict to $0 \leq t < \frac{1}{2}$ and keep only rows 0, 1, ..., $N-1$.

Since $\phi(2t-1) = 0$ and $\phi(2t+N) = 0$ we only need columns 0 to $N-1$ of the infinite matrix \mathbf{M} . This N by N section is $\mathbf{m}(0)$.

(Verify $\mathbf{m}(1)$) Restrict to $\frac{1}{2} \leq t < 1$ and keep only rows 0, 1, ..., $N-1$.

Since $\phi(2t-2) = 0$ and $\phi(2t+N-1) = 0$ we only need columns $-1, 0, \dots, N-2$ of \mathbf{M} . This N by N section is $\mathbf{m}(1)$.

The change from $\mathbf{m}(0)$ to $\mathbf{m}(1)$ comes as t crosses $\frac{1}{2}$, because the nonzero entries in $\Phi_\infty(2t)$ appear one component earlier.

We now go back to the eigenvalue problem $\Phi(0) = \mathbf{m}(0)\Phi(0)$. Condition A_1 leads to the eigenvalue $\lambda = 1$, and guarantees a solution.

The Fixed Point Equation $\Phi(0) = \mathbf{m}(0)\Phi(0)$

The rows of $\mathbf{m}(0)$ have a double shift. The columns have entirely even indices or entirely odd indices. The 5 by 5 matrix ($N = 5$) shows this pattern:

$$\mathbf{m}(0) = 2 \begin{bmatrix} h(0) & & & & \\ h(2) & h(1) & h(0) & & \\ h(4) & h(3) & h(2) & h(1) & h(0) \\ & h(5) & h(4) & h(3) & h(2) \\ & & & h(5) & h(4) \end{bmatrix}.$$

The key requirement on the coefficients $h(n)$ is Condition A_1 : *The frequency response $H(\omega)$ has a zero at $\omega = \pi$:*

$$h(0) - h(1) + h(2) - \dots = 0.$$

Combined with $h(0) + h(1) + h(2) + \dots = 1$, this means that every column of $\mathbf{m}(0)$ and $\mathbf{m}(1)$ and \mathbf{M} adds to 1.

Theorem 6.6 *Condition A_1 guarantees that $\lambda = 1$ is an eigenvalue of \mathbf{M} and $\mathbf{m}(0)$ and $\mathbf{m}(1)$:*

$$\begin{aligned} h(0) - h(1) + h(2) - \dots &= 0 \\ h(0) + h(1) + h(2) + \dots &= 1 \end{aligned} \quad \text{yields } 2 \sum_{\text{even } n} h(n) = 2 \sum_{\text{odd } n} h(n) = 1.$$

Any matrix with unit column sums has $\lambda = 1$ as an eigenvalue. Therefore $\Phi(0) = \mathbf{m}(0)\Phi(0)$ can be solved to give the scaling function at integer times $\Phi(0) = (\phi(0), \phi(1), \dots, \phi(N-1))^T$.

Proof. Adding the two equations gives the even part. Subtracting gives the odd part. These are the column sums of the matrix $\mathbf{m}(0)$, all equal to 1. A matrix with unit column sums has a left eigenvector of ones, $\mathbf{e}\mathbf{m}(0) = \mathbf{e}$, because the multiplication just adds up each column:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \mathbf{m}(0) = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}.$$

This means that $\mathbf{m}(0) - \mathbf{I}$ is not invertible, and $\lambda = 1$ is an eigenvalue. The left eigenvector is \mathbf{e} . The right eigenvector is $(\phi(0), \phi(1), \dots, \phi(N-1))$:

$$(\mathbf{m}(0) - \mathbf{I})\Phi(0) = 0 \text{ which means } \Phi(0) = \mathbf{m}(0)\Phi(0).$$

The fundamental fact is that a square matrix and its transpose have the same determinant and same rank and same eigenvalues.

The columns of $\mathbf{m}(1)$ and \mathbf{M} also add to 1, producing the eigenvalue $\lambda = 1$. Small point! We can safely normalize the eigenvector $\Phi(0)$ by $\sum \phi(n) = 1$. This is the “unit area” requirement that we impose on the function $\phi^{(0)}(t)$ at the start of the iterations. Then the scaling function has $\int \phi(t)dt = 1$ at the end.

Corollary *The sum $\sum \phi(t+k)$ is identically 1.*

Proof. Multiply the vector dilation equation $\Phi(t) = \mathbf{m}(0)\Phi(2t) + \mathbf{m}(1)\Phi(2t-1)$ on the left by \mathbf{e} . Use the fact that $\mathbf{e}\mathbf{m}(0) = \mathbf{e}$ and $\mathbf{e}\mathbf{m}(1) = \mathbf{e}$:

$$\mathbf{e}\Phi(t) = \mathbf{e}\Phi(2t) + \mathbf{e}\Phi(2t-1).$$

This is a dilation equation for $\mathbf{e}\Phi(t)$ and its solution is the box function! Thus $\mathbf{e}\Phi(t) = 1$. The “periodized scaling function” $\sum \phi(t+k)$ is identically one.

Example 6.3. The coefficients $2\mathbf{h}(k) = \frac{1}{2}, 1, \frac{1}{2}$ lead to the hat function. The 2 by 2 eigenvalue problem for $\mathbf{m}(0)$ gives the correct values of $\phi(0)$ and $\phi(1)$:

$$\begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} \phi(0) \\ \phi(1) \end{bmatrix} = \begin{bmatrix} \phi(0) \\ \phi(1) \end{bmatrix} \text{ gives } \begin{aligned} \phi(0) &= 0 \\ \phi(1) &= 1. \end{aligned}$$

The sum of all hat functions $\phi(t+n)$ is identically one. Notice that the first row of the eigenvalue equation is always $2\mathbf{h}(0)\phi(0) = \phi(0)$. Then $\phi(0)$ is zero, apart from the exceptional case $\mathbf{h}(0) = \frac{1}{2}$ which occurs for the box function. This means that the scaling function $\phi(t)$ is zero up to and including $t = 0$. The box function starts with a jump at $t = 0$, because $\mathbf{h}(0) = \frac{1}{2}$.

Example 6.4. The Daubechies coefficients have $8\mathbf{h}(k) = 1 + \sqrt{3}, 3 + \sqrt{3}, 3 - \sqrt{3}$, and $1 - \sqrt{3}$. Dividing by 4 we have $2\mathbf{h}(k)$, the numbers that enter $\mathbf{m}(0)$:

$$\frac{1}{4} \begin{bmatrix} 1 + \sqrt{3} & 0 & 0 \\ 3 - \sqrt{3} & 3 + \sqrt{3} & 1 + \sqrt{3} \\ 0 & 1 - \sqrt{3} & 3 - \sqrt{3} \end{bmatrix} \begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \end{bmatrix} = \begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \end{bmatrix}.$$

The eigenvector gives $\phi(0)$, $\phi(1)$, and $\phi(2)$:

$$\phi(0) = 0 \quad \phi(1) = \frac{1}{2}(1 + \sqrt{3}) \quad \phi(2) = \frac{1}{2}(1 - \sqrt{3}).$$

We now know the Daubechies scaling function $\phi(t)$ at the integers. The only nonzeros in the fixed-point eigenvector Φ are $\phi(1)$ and $\phi(2)$. From these values at $t = 1$ and $t = 2$, the recursion produces $\phi(t)$ at any dyadic point.

Practical conclusion Every $\phi(n/2^i)$ comes via $m(0)$ and $m(1)$.

Theoretical conclusion Those dyadic values have a uniform bound if and only if all products of $m(0)$ and $m(1)$ in all orders have a *uniform upper bound*. When this holds, the dilation equation has a bounded solution $\phi(t)$ for all t .

We can propose sufficient conditions so that all products of $m(0)$ and $m(1)$ have a uniform bound. We can also propose necessary conditions. It is not known how to verify the necessary and sufficient condition (Section 7.3).

Derivatives of the Dilation Equation

While working in the time domain, we might as well take the derivative of $\phi(t)$. The result is highly interesting and not fully understood. Part of the problem is that the derivative $\phi'(t)$ may not exist.

The plan is to differentiate each term in the dilation equation for $\phi(t)$:

$$\phi'(t) = 4 \sum h(k)\phi'(2t - k).$$

This is another dilation equation, with every coefficient doubled. The equation $\Phi(t) = M\Phi(2t)$ has led to $\Phi'(t) = 2M\Phi'(2t)$. At $t = 0$ this yields the fixed-point equation $\Phi'(0) = 2M\Phi'(0)$. The eigenvector $\Phi'(0)$ contains the derivatives $\phi'(n)$ at the integers $t = n$.

To solve $\Phi'(0) = M\Phi'(0)$, we have a new requirement. *The number $\lambda = \frac{1}{2}$ must also be an eigenvalue of M .* Again this applies to the $N \times N$ matrices $m(0)$ and $m(1)$. This new requirement on the entries is stated as Condition A_2 in the following theorem.

Theorem 6.7 *The matrices M and $m(0)$ and $m(1)$ have eigenvalues I and $\frac{1}{2}$ if and only if the filter coefficients satisfy Condition A_2 which includes A_1 :*

$$\text{Condition } A_2 : \sum_0^N (-1)^k h(k) = 0 \quad \text{and} \quad \sum_0^N (-1)^k kh(k) = 0.$$

The eigenvector for $\lambda = 1$ is $\Phi(0)$, containing the values $\phi(0), \dots, \phi(N-1)$. The eigenvector for $\lambda = \frac{1}{2}$ is $\Phi'(0)$, containing the derivatives $\phi'(0), \dots, \phi'(N-1)$.

In this case $H(\omega)$ has a *double zero* at $\omega = \pi$. This beautiful pattern extends onward to Condition A_p . The matrices have eigenvalues $\lambda = 1, \frac{1}{2}, \dots, (\frac{1}{2})^{p-1}$ if and only if the filter coefficients satisfy p sum rules:

$$\text{Condition } A_p : \sum (-1)^k k^m h(k) = 0 \quad \text{for } m = 0, \dots, p-1.$$

The eigenvector for $\lambda = \left(\frac{1}{2}\right)^m$ contains values of the m^{th} derivative of $\phi(t)$ at the integers. Formally, $\Phi^{(m)}(0) = 2^m M \Phi^{(m)}(0)$ comes from differentiating the dilation equation m times at the integers. We mention the frequency domain equivalent:

Condition A_p : The frequency response $H(\omega)$ has a zero of order p at $\omega = \pi$.

We will see this again! And we also begin to uncover the crucial role of the *left eigenvectors* (which are row vectors). Those tell how to produce polynomials from combinations of the translates $\phi(t - k)$. Under Condition A_p , these low order polynomials $1, t, \dots, t^{p-1}$ are in the low-pass space V_0 . They are the keys to approximation of a function $f(t)$ by functions in V_0 .

The letters A_p indicate “approximation of order p .” The theorem above, with its extension from 2 to p , is absolutely basic to the algebra of downsampled filters. These eigenvalues and eigenvectors control everything in Chapter 7.

You will see that the derivative $\phi'(t)$ is often *one-sided*. Derivatives of $\phi(t)$ may not exist in the usual sense. This subject still contains some mysteries.

Example 6.5. The hat function coefficients $2h(k) = \frac{1}{2}, 1, \frac{1}{2}$ satisfy Condition A_2 :

$$\begin{aligned} \text{First sum rule: } & \frac{1}{2} - 1 + \frac{1}{2} = 0 \\ \text{Second sum rule: } & 0\left(\frac{1}{2}\right) - 1(1) + 2\left(\frac{1}{2}\right) = 0. \end{aligned}$$

Therefore $m(0)$ will have eigenvalues 1 and $\frac{1}{2}$:

$$m(0) = \begin{bmatrix} 2h(0) & 0 \\ 2h(2) & 2h(1) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 1 \end{bmatrix}.$$

The eigenvector for $\lambda = 1$ has components 0 and 1. They agree with $\phi(0)$ and $\phi(1)$, the hat function at the integers. The eigenvector for $\lambda = \frac{1}{2}$ has components 1 and -1 . They are $\phi'_+(0)$ and $\phi'_+(1)$, the slopes $\phi'(t)$ of the hat function in the two intervals. These are derivatives *from the right* at the points $t = 0$ and $t = 1$. The slopes on the left side of those points are different because the hat function has corners.

The matrix $m(1)$ must also have eigenvalues 1 and $\frac{1}{2}$:

$$m(1) = \begin{bmatrix} 2h(1) & 2h(0) \\ & 2h(2) \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & \frac{1}{2} \end{bmatrix}.$$

The eigenvector for $\lambda = 1$ has components 1 and 0. Those agree with the hat function at the shifted points $t = 1$ and $t = 2$. The eigenvector for $\lambda = \frac{1}{2}$ has components 1 and -1 . Those agree with the slopes of the hat function *from the left* at $t = 1$ and $t = 2$. Remember that $m(0)$ is involved at the start of an interval and $m(1)$ is involved at the end of an interval.

Condition A_3 is not satisfied for the hat function. There is no eigenvalue $\lambda = \frac{1}{4}$. The hat function has no second derivatives at the integers.

Problem Set 6.3

- If the filter $H(z)$ is halfband, show that the eigenvector in $m(0)\Phi = \Phi$ is an impulse $\delta(n)$. What are the values of $\phi(t)$ at the integers?

2. If $\phi_1(t)$ and $\phi_2(t)$ satisfy dilation equations, does their product $P(t) = \phi_1(t)\phi_2(t)$ satisfy a dilation equation?
3. Show that the convolution $\phi_1(t) * \phi_2(t)$ *does* satisfy a dilation equation with coefficients from $h_1 * h_2$.
4. Find a specific function $f(t)$ that does not satisfy any dilation equation.

6.4 Infinite Product Formula

The scaling function $\phi(t)$ comes from the dilation equation

$$\phi(t) = 2 \sum_{k=0}^N h(k)\phi(2t-k).$$

Thus $\phi(t)$ is the fixed point, or fixed function, when we iterate with H and rescale. In the time domain, the matrix that filters and rescales is $M = (\downarrow 2)2H$. Now we intend to find the Fourier transform $\widehat{\phi}(\omega)$ in the frequency domain. Just as the time-domain solution involved products of $m(0)$ and $m(1)$, the frequency-domain solution will involve an infinite product of $H(\omega)$'s.

It is quite remarkable that two-scale equations received so little attention for so long. Historically, t and $2t$ were not often seen in the same equation. They began to appear prominently for fractals, which are self-similar. Now, multiple scales seem to be everywhere. We meet them in this book through multirate filters — with two scales. Then the iteration leads to all scales.

If the 2's were removed, the dilation equation would be an ordinary difference equation. The coefficients are constant, so we look for pure exponential solutions $e^{i\omega t}$. When you make that substitution, you are effectively taking the Fourier transform of the equation. The transform turns difference equations and differential equations (*and dilation equations*) into algebraic equations. We do that now for the two-scale equation, and we watch how $2t$ leads to $\omega/2$.

The dilation equation becomes $\widehat{\phi}(\omega) = H(\frac{\omega}{2})\widehat{\phi}(\frac{\omega}{2})$. This leads recursively to an infinite product for $\widehat{\phi}(\omega)$. This transform must be a *sinc function* when $h(0) = h(1) = 1/2$ — because the time-domain solution $\phi(t)$ is a box function. That sinc function must be orthogonal to its modulations by $e^{-i\omega k}$, because the box function is orthogonal to its translates $\phi(t - k)$. We have to study orthogonality and also approximation, which is controlled by “zeros at π .” These properties are now studied in the frequency domain.

1. Condition O for orthogonality:

$$\begin{aligned} |H(\omega)|^2 + |H(\omega + \pi)|^2 &\equiv 1 \text{ in the frequency domain} \\ 2 \sum h(k)h(k - 2\ell) &= \delta(\ell) \text{ in the time domain} \end{aligned}$$

This is double-shift orthogonality of the lowpass filter coefficients. It connects to orthogonality of the scaling functions $\phi(t - k)$. We use the word “connects” rather than “implies,” because a further condition is eventually needed to insure orthogonality in the limit of the iteration. The step from discrete time to continuous time seldom goes wrong, but it can.

This orthogonality will appear in Theorem 6.10 as a neat statement about the Fourier transforms of ϕ and w :

$$\sum_{-\infty}^{\infty} |\widehat{\phi}(\omega + 2\pi n)|^2 = 1 \quad \text{and} \quad \sum_{-\infty}^{\infty} \widehat{\phi}(\omega + 2\pi n) \overline{\widehat{w}(\omega + 2\pi n)} = 0.$$

The other side of the theory is about approximation. This imposes a very different condition on the $h(k)$ and the polynomial $H(\omega) = \sum h(k)e^{-ik\omega}$.

$$2. \text{ Condition } A_p: H(\omega) = \left(\frac{1 + e^{-i\omega}}{2} \right)^p Q(\omega).$$

This factor $(1 + e^{-i\omega})^p$ means that $H(\omega)$ has a *zero of order p* at $\omega = \pi$. We will prove that this puts the polynomials $1, t, \dots, t^{p-1}$ in the scaling subspace V_0 . They are combinations of $\phi(t - n)$ and they are orthogonal to $w(t - n)$. In the frequency domain, there is again a neat statement about the Fourier transforms of ϕ and w :

$$\begin{aligned}\widehat{\phi} &\text{ has a zero of order } p \text{ at every } \omega = 2\pi n, n \neq 0 \\ \widehat{w} &\text{ has a zero of order } p \text{ at zero frequency.}\end{aligned}$$

The wavelet coefficients of a smooth function $f(t) = \sum b_{jk} w_{jk}(t)$ decrease faster when p is larger. The estimate is $|b_{jk}| = O(2^{-jp})$. This is valuable for compression. This section does the frequency-domain algebra, to solve the dilation equation and to explain Condition O and Condition A_p .

Transform and Solution of the Dilation Equation

To transform the dilation equation, multiply by $e^{-i\omega t}$. Integrate with respect to t :

$$\int_{-\infty}^{\infty} \phi(t)e^{-i\omega t} dt = 2 \sum_{k=0}^N h(k) \int_{-\infty}^{\infty} \phi(2t - k)e^{-i\omega t} dt.$$

The left side is $\widehat{\phi}(\omega)$. In the integral on the right, set $u = 2t - k$ and $t = (u + k)/2$:

$$\int_{-\infty}^{\infty} \phi(2t - k)e^{-i\omega t} 2 dt = \int_{-\infty}^{\infty} \phi(u)e^{-i\omega(u+k)/2} du = e^{-i\omega k/2} \widehat{\phi}\left(\frac{\omega}{2}\right). \quad (6.43)$$

Instead of t and $2t$, the transform involves ω and $\omega/2$. The dilation equation becomes

$$\widehat{\phi}(\omega) = \left(\sum h(k)e^{-i\omega k/2} \right) \widehat{\phi}\left(\frac{\omega}{2}\right) = H\left(\frac{\omega}{2}\right) \widehat{\phi}\left(\frac{\omega}{2}\right). \quad (6.44)$$

This is the result of filtering and rescaling. Filtering multiplies $\widehat{\phi}(\omega)$ by $H(\omega)$. Rescaling changes ω to $\omega/2$. The scaling function (which is unique up to a constant multiple C — this is still to be proved) comes out unchanged:

$$\boxed{\widehat{\phi}(\omega) = H\left(\frac{\omega}{2}\right) \widehat{\phi}\left(\frac{\omega}{2}\right)}.$$

Now iterate this equation. It connects ω to $\omega/2$ and therefore it connects $\omega/2$ to $\omega/4$:

$$\widehat{\phi}(\omega) = H\left(\frac{\omega}{2}\right) [H\left(\frac{\omega}{4}\right) \widehat{\phi}\left(\frac{\omega}{4}\right)].$$

After N iterations, this becomes

$$\widehat{\phi}(\omega) = H\left(\frac{\omega}{2}\right) H\left(\frac{\omega}{4}\right) \cdots H\left(\frac{\omega}{2^N}\right) \widehat{\phi}\left(\frac{\omega}{2^N}\right).$$

In the limit as $N \rightarrow \infty$, we have a formula for the solution $\widehat{\phi}(\omega)$. Note that $\omega/2^N$ is approaching zero, and $\widehat{\phi}(0) = \int \phi(t) dt$ is the area under the graph of $\phi(t)$. This equals one. We impose

the normalization $\widehat{\phi}(0) = 1$ in the frequency domain, just as we required unit area in the time domain. Then the formal limit of the iteration leads to the famous infinite product for $\widehat{\phi}$:

$$\widehat{\phi}(\omega) = \prod_{j=1}^{\infty} H\left(\frac{\omega}{2^j}\right). \quad (6.45)$$

We note the minimum requirement for convergence of this or any infinite product: the factors $H(\omega/2^j)$ must approach 1 as $j \rightarrow \infty$. Thus we need $H(0) = 1$. By periodicity $H(2\pi) = H(4\pi) = 1$. Then the equation $\widehat{\phi}(\omega) = H\left(\frac{\omega}{2}\right) \widehat{\phi}\left(\frac{\omega}{2}\right)$ has a remarkable consequence. *The values $\widehat{\phi}(2\pi)$, $\widehat{\phi}(4\pi)$, $\widehat{\phi}(8\pi)$, ... are all equal. If $H(\pi) = 0$, those values equal zero because $\widehat{\phi}(2\pi) = H(\pi) \widehat{\phi}(\pi)$.*

This “zero at π ” is a natural requirement on $H(\omega)$ in order that $\widehat{\phi}(\omega)$ may decay and $\phi(t)$ may be a reasonable function.

The infinite product converges for every ω and every $H(\omega)$. We have an explicit formula for $\widehat{\phi}(\omega)$. Whether any function $\phi(t)$ has this Fourier transform is another matter! Convergence follows from a rough bound on $H(\omega)$ in terms of $C = \max |H'(\omega)|$:

$$|H(\omega)| = |1 + H(\omega) - H(0)| \leq 1 + C|\omega| \leq e^{C|\omega|}.$$

Then the product $\widehat{\phi}(\omega)$ has the same upper bound:

$$|\widehat{\phi}(\omega)| = |H\left(\frac{\omega}{2}\right)| |H\left(\frac{\omega}{4}\right)| \cdots \leq e^{C|\omega|/2} e^{C|\omega|/4} \cdots = e^{C|\omega|}.$$

This is a wild overestimate of $\widehat{\phi}(\omega)$, as almost any example will show.

Box example. The coefficients are $h(0) = h(1) = \frac{1}{2}$. Then $H(\omega) = \frac{1}{2}(1 + e^{-i\omega})$. The product of the first N factors contains 2^N terms. Looked at correctly, those terms are the first 2^N powers of $e^{-i\omega/2^N}$:

$$\begin{aligned} H^{(N)}(\omega) &= \frac{1}{2^N} (1 + e^{-i\omega/2})(1 + e^{-i\omega/4}) \cdots (1 + e^{-i\omega/2^N}) \\ &= \frac{1}{2^N} \sum_{k=0}^{2^N-1} e^{-i\omega k/2^N} \quad (\text{geometric series}) \\ &= \frac{1 - e^{-i\omega}}{2^N(1 - e^{-i\omega/2^N})} \quad (\text{sum of } 2^N \text{ terms}). \end{aligned} \quad (6.46)$$

Now let $N \rightarrow \infty$. The denominator has $1 - e^{-i\theta} = 1 - (1 - i\theta + \dots) = i\theta + \dots$ with $\theta = \omega/2^N$. The limit of $2^N i\theta$ is $i\omega$. Therefore the limit of the partial product is the infinite product

$$\widehat{\phi}(\omega) = \prod \left(\frac{1}{2} + \frac{1}{2} e^{-i\omega/2^j} \right) = (1 - e^{-i\omega})/i\omega. \quad (6.47)$$

This sinc function is the transform of the box function. The integral of $e^{-i\omega t}$ from 0 to 1 agrees with $\widehat{\phi}(\omega)$. Instead of increasing like $e^{C|\omega|}$, as allowed by the general estimate, the transform $\widehat{\phi}(\omega)$ actually decreases to zero as ω becomes large.

Compare the construction of $\phi(t)$ with $\widehat{\phi}(\omega)$. In Section 6.2, we assumed that $\phi^{(i)}(t)$ converged uniformly to $\phi(t)$. Then we studied its properties. In this section, the convergence of the infinite product is cheap (for each separate ω). What we need is sufficient decay of $\widehat{\phi}(\omega)$ as

$|\omega| \rightarrow \infty$. Our precise assumption will be continuity of the function $A(\omega)$ in Theorem 6.10 below. Then we can safely study $\widehat{\phi}(\omega)$ in the frequency domain. Note first that for real frequencies, the growth of $\widehat{\phi}(\omega)$ is at most polynomial:

Theorem 6.8 $|\widehat{\phi}(\omega)| \leq e^{C|\omega|}$ for complex ω and $|\widehat{\phi}(\omega)| \leq c(1 + |\omega|^M)$ for real ω .

Brief reason: $H(\omega)$ is periodic. It has a maximum value 2^M . The equation $\widehat{\phi}(2\omega) = H(\omega)\widehat{\phi}(\omega)$ says that $\widehat{\phi}$ grows by at most 2^M when ω is doubled. The bound $|\omega|^M$ has this growth rate. A constant is included to make $c(1 + |\omega|^M)$ correct for small $|\omega|$.

The example $H(\omega) = \frac{1}{2} + \frac{1}{2}e^{-i\omega}$ is bounded by 1 for real ω and by $e^{|\omega|}$ for complex ω . Then $M = 0$. The transform $\widehat{\phi}(\omega)$ of the box function has those same bounds:

$$|\widehat{\phi}(\omega)| = \left|H\left(\frac{\omega}{2}\right)\right| \left|H\left(\frac{\omega}{4}\right)\right| \cdots \leq \begin{cases} 1 & \text{for real } \omega \\ e^{|\omega|/2} e^{|\omega|/4} \dots = e^{|\omega|} & \text{for complex } \omega. \end{cases}$$

Section 6.1 showed that the support interval for $\phi(t)$ is $[0, N]$. This can be proved in the frequency domain too. Our bounds on $\widehat{\phi}(\omega)$ show two fundamental facts about $\phi(t)$:

Theorem 6.9 Any dilation equation with $h(0) + \dots + h(N) = 1$ has a unique and compactly supported solution $\phi(t)$. This solution may be a distribution.

Compact support comes from $|\widehat{\phi}(\omega)| \leq e^{C|\omega|}$. The Paley-Wiener Theorem implies that $\phi(t)$ is supported on the interval $[-C, C]$. With more care [D, p.176] we could find again the exact support interval $[0, N]$.

Uniqueness comes from our formula for the solution! The infinite product converges to $\widehat{\phi}(\omega)$, which is continuous because $\phi(t)$ has compact support:

$$\widehat{\phi}^{(i)}(\omega) = \left(\prod_{j=1}^i H(\omega/2^j) \right) \widehat{\phi}(\omega/2^i) \text{ approaches } \widehat{\phi}(\omega) = \left(\prod_{j=1}^{\infty} H(\omega/2^j) \right) \widehat{\phi}(0).$$

In the IIR case, suitable hypotheses will again give uniqueness (of course not compact support). At the other extreme, note how the lazy filter with $h(0) = 1$ leads to $\phi(t) = \delta(t)$. The dilation equation $\phi(t) = 2\phi(2t)$ is solved by $\phi(t) = \delta(t)$:

In frequency: $H(\omega) \equiv 1$ so $\widehat{\phi}(\omega) = 1$.

Cascade algorithm: $\phi^{(i)}(t) = \text{box function on } [0, 2^{-i}] \text{ with height } 2^i$.

Verify directly: $\delta(t) = 2\delta(2t)$ from $\int f(t)\delta(t)dt = f(0) = \int f(t)\delta(2t)2dt$.

All these methods show that $h(0) = 1$ produces the best-known distribution $\phi(t) = \delta(t)$.

Orthogonality in the Frequency Domain

The product formula for $\widehat{\phi}(\omega)$ applies with or without Condition O. When that condition holds, we expect orthogonality of the translates $\phi(t - k)$. To establish this orthogonality in the frequency domain, we need to know that the equivalent statement is $A(\omega) \equiv 1$. The function $A(\omega)$ enters naturally into this discussion. It is the transform $\sum a(k)e^{i\omega k}$ of the vector of inner products of $\phi(t)$ with $\phi(t - k)$:

Theorem 6.10 *The inner products $a(k)$ are the Fourier coefficients of the 2π -periodic function $A(\omega)$:*

$$a(k) = \int_{-\infty}^{\infty} \phi(t) \overline{\phi(t-k)} dt \quad \text{transforms to} \quad A(\omega) = \sum_{-\infty}^{\infty} |\hat{\phi}(\omega + 2\pi n)|^2.$$

The translates $\phi(t - k)$ are orthonormal if and only if $A(\omega) \equiv 1$.

Proof. An inner product in the time domain equals an inner product in the frequency domain, by Parseval's identity. The inner product in the time domain is between $\phi(t)$ and $\phi(t - k)$. The transforms of these functions are $\hat{\phi}(\omega)$ and $e^{-i\omega k} \hat{\phi}(\omega)$. Each inner product integrates one function times the complex conjugate of the other:

$$\begin{aligned} a(k) &= \int_{-\infty}^{\infty} \phi(t) \overline{\phi(t-k)} dt &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{\phi}(\omega) \overline{\hat{\phi}(\omega)} e^{i\omega k} d\omega \\ &= \frac{1}{2\pi} \int_0^{2\pi} \sum_{-\infty}^{\infty} |\hat{\phi}(\omega + 2\pi n)|^2 e^{i\omega k} d\omega. \end{aligned} \quad (6.48)$$

The last integral split $(-\infty, \infty)$ into an infinite number of 2π -pieces, using the periodicity of $e^{i\omega k}$. This integral defines the k^{th} Fourier coefficient of $A(\omega)$. Thus $A(\omega) = \sum a(k) e^{i\omega k}$.

For an FIR filter, $\phi(t) = 0$ outside the interval $[0, N]$. The inner products are $a(k) = 0$ for $|k| > N$, because $\phi(t)$ and $\phi(t - k)$ have no overlap. The function $A(\omega) = \sum a(k) e^{i\omega k}$ is a trigonometric polynomial of degree N , which is not obvious from $\sum |\hat{\phi}(\omega + 2\pi n)|^2$. In Section 7.3 we will compute $a(k)$ directly from the coefficients $h(n)$. This is always a main point of the theory, to return every calculation to those numbers $h(n)$.

When the translates are orthonormal, all inner products $a(k)$ are zero except for $a(0) = 1$. The function with those coefficients is the constant function $A(\omega) \equiv 1$:

$$\boxed{\phi(t - k) \text{ are orthonormal} \iff \sum_{-\infty}^{\infty} |\hat{\phi}(\omega + 2\pi n)|^2 \equiv 1.}$$

We now apply Condition O in the frequency domain to deduce this orthogonality of $\phi(t - k)$. We are repeating in the frequency domain the result of Section 6.2 in the time domain. I believe this is worthwhile! The arguments in the two domains look quite different. Recall the condition on the frequency response $H(\omega)$ to produce an orthonormal filter bank:

$$\text{Condition O: } |H(\omega)|^2 + |H(\omega + \pi)|^2 \equiv 1.$$

This function $H(\omega)$ leads to $\hat{\phi}(\omega)$ which leads to $A(\omega)$. Somehow, Condition O must imply that $A(\omega) \equiv 1$. The steps are typical of computations in the frequency domain.

Theorem 6.11 *If $A(\omega)$ is continuous, $A(\omega) \equiv 1$ is equivalent to Condition O.*

Proof. We use a very important two-scale identity, proved below:

$$A(2\omega) = |H(\omega)|^2 A(\omega) + |H(\omega + \pi)|^2 A(\omega + \pi). \quad (6.49)$$

If $A(\omega) \equiv 1$, this immediately gives that $|H(\omega)|^2 + |H(\omega + \pi)|^2 \equiv 1$.

For the converse, suppose that Condition O holds. The identity says that $A(2\omega)$ is a weighted average of $A(\omega)$ and $A(\omega + \pi)$. At the point ω_0 where $A(2\omega)$ reaches its maximum, $A(\omega_0)$ and $A(\omega_0 + \pi)$ must also reach that maximum. Now repeat the argument at $\omega_0/2$, to show that $A(\omega_0/2)$ shares this same maximum with $A(\omega_0)$. Continuing, the maximum of $A(\omega)$ is achieved at $\omega_0/4$ and $\omega_0/8$ and eventually (by continuity) at $\omega = 0$.

By a similar argument, which is due to Tchamitchian, the minimum of $A(\omega)$ is also attained at $\omega = 0$. Therefore $A(\omega)$ is constant. We verify below that the constant is one: $A(\omega) \equiv 1$. It only remains to prove (6.49).

This valuable identity for $A(2\omega)$ has a nice proof. It uses the dilation equation $\widehat{\phi}(2\omega) = H(\omega)\widehat{\phi}(\omega)$. At the points $2\omega + 2\pi n$, this splits into separate cases for even n and odd n :

$$\begin{aligned}\widehat{\phi}(2\omega + 2\pi n) &= H(\omega + \pi n)\widehat{\phi}(\omega + \pi n) \\ &= \begin{cases} H(\omega)\widehat{\phi}(\omega + 2k\pi) & \text{if } n = 2k \\ H(\omega + \pi)\widehat{\phi}(\omega + (2k+1)\pi) & \text{if } n = 2k+1. \end{cases}\end{aligned}$$

Now square both sides. Sum from $-\infty$ to ∞ on n and therefore on k . The sum of squares is our function $A(2\omega)$ in the desired identity:

$$\begin{aligned}A(2\omega) &= |H(\omega)|^2 \sum_{-\infty}^{\infty} |\widehat{\phi}(\omega + 2\pi k)|^2 + |H(\omega + \pi)|^2 \sum_{-\infty}^{\infty} |\widehat{\phi}(\omega + \pi + 2\pi k)|^2 \\ &= |H(\omega)|^2 A(\omega) + |H(\omega + \pi)|^2 A(\omega + \pi).\end{aligned}\tag{6.50}$$

The final step is to confirm that $A(0) = 1$. This comes from our other condition on the lowpass filter, not yet used in the frequency domain. Condition A_1 is $H(\pi) = 0$. In the time domain, this first sum rule guaranteed an eigenvalue $\lambda = 1$ for the matrices M and $m(0)$ and $m(1)$. The fixed-point equation $\phi^{(1)}(n) = \phi^{(0)}(n)$ at the integers could be solved. Condition A_1 is equally essential in the frequency domain. Here we use it to pin down the value $A(0) = 1$.

Theorem 6.12 If $H(\pi) = 0$ then $\widehat{\phi}(2\pi n) = 0$ for all $n \neq 0$. Therefore

$$A(0) = \sum_{-\infty}^{\infty} |\widehat{\phi}(2\pi n)|^2 = |\widehat{\phi}(0)|^2 = 1.\tag{6.51}$$

Proof. The infinite product for $\widehat{\phi}(2\pi) = H(\pi)H(\pi/2)\dots$ starts with the factor $H(\pi)$. Immediately this product is zero. For any higher value $n > 1$, write $n = 2^j m$ with odd m . Then the $(j+1)^{\text{st}}$ factor in the infinite product is zero when $\omega = 2\pi n$:

$$\widehat{\phi}(2\pi n) = H(\pi n)H\left(\frac{\pi n}{2}\right)\dots H\left(\frac{\pi n}{2^j}\right)\dots = 0$$

because $H(\pi n/2^j) = H(\pi m)$. By periodicity this is $H(\pi) = 0$. The only nonzero term is $|\widehat{\phi}(0)|^2$. But $\widehat{\phi}(0) = H(0)H(0)H(0)\dots$ which is 1.

Orthogonalization of the Basis

The condition for an orthonormal basis is $A(\omega) \equiv 1$. When this is not satisfied, there is an easy way to make it satisfied. In other words: when the translates $\phi(t - n)$ are not orthonormal, there

is an easy way to make them orthonormal. Divide $\widehat{\phi}(\omega)$ by the given $A(\omega)$ (or rather, its square root) to get the new orthogonalized function $\widehat{\phi}_{\text{orth}}(\omega)$:

$$\widehat{\phi}_{\text{orth}}(\omega) = \frac{\widehat{\phi}(\omega)}{\sqrt{A(\omega)}}.$$

This immediately gives orthogonality of the new basis $\{\phi_{\text{orth}}(t - n)\}$:

$$A_{\text{orth}}(\omega) = \sum \left| \frac{\widehat{\phi}(\omega + 2\pi n)}{\sqrt{A(\omega)}} \right|^2 = \frac{A(\omega)}{A(\omega)} = 1.$$

That succeeds if $A(\omega)$ is never zero. This is the condition for a *Riesz basis*.

Theorem 6.13 *The upper and lower bounds on $A(\omega)$ are the Riesz constants B and A for the basis $\{\phi(t - k)\}$ of V_0 . Thus $A(\omega) \geq A > 0$ gives a stable basis, and dividing $\widehat{\phi}(\omega)$ by $\sqrt{A(\omega)}$ gives an orthonormal basis.*

When $\phi(t)$ comes from a dilation equation — this is our normal situation — Condition E in Chapter 7 gives an equivalent test for a Riesz basis (in terms of eigenvalues). If this test is passed, the wavelets $w_{jk}(t)$ are a Riesz basis for $L^2(\mathbf{R})$.

Proof. To test the linear independence of the functions $\phi(t - k)$, form the matrix A from their inner products. The entries are $A_{ij} = \langle \phi(t - i), \phi(t - j) \rangle$. That number is $a(j - i)$, and A is a Toeplitz matrix! It is the matrix TT^* in Section 2.5. In the frequency domain it becomes multiplication by $A(\omega)$. The upper and lower bounds on $A(\omega)$ determine whether $\{\phi(t - k)\}$ is a Riesz basis.

Orthogonalization is always a basic step in linear algebra. There it is done by the Gram-Schmidt algorithm. We start with independent vectors and produce orthonormal vectors (or functions). This algorithm is not successful here, because it is not time-invariant. The orthogonalized functions will certainly not be translates — when the Gram-Schmidt algorithm works on functions in a definite order like $\phi(t), \phi(t - 1), \phi(t + 1), \dots$. To keep a shift-invariant basis, we needed to orthogonalize all these translates at once. The division by $\sqrt{A(\omega)}$ did it.

In matrix language, $M_{\text{orth}} M_{\text{orth}}^T = I$. In the improved factorization by Fourier methods, all rows of M_{orth} come from the zeroth row by double shifts. In other words, M_{orth} comes from a filter.

One problem with dividing by $\sqrt{A(\omega)}$. This destroys the finite response of the original filter H . The new filter H_{orth} is IIR, not FIR. The new scaling function $\phi_{\text{orth}}(t)$ that corresponds to $\widehat{\phi}(\omega)/\sqrt{A(\omega)}$ does not have compact support. Vetterli and Herley noticed that this is not as bad as it seems. Since $\phi(t)$ is zero outside the interval $[0, N]$, the inner products $a(k) = \int \phi(t)\phi(t - k) dt$ are zero for $|k| > N$. The function $A(\omega)$ with these Fourier coefficients is a real non-negative trigonometric polynomial of degree N . Its square root $G(\omega) = \sum g(k)e^{-ik\omega}$, by spectral factorization, is also a polynomial of degree N . The frequency response of the new orthogonalized filter is a ratio of polynomials

$$H_{\text{orth}}(\omega) = \frac{H(\omega)}{G(\omega)}.$$

The input-output equation $y(k) = \sum h_{\text{orth}}(k)x(n - k)$ is an implicit difference equation, from an autoregressive moving average filter:

$$\sum_0^N g(k)y(n - k) = \sum_0^N h(k)x(n - k).$$

The new filter is IIR but it only involves $2N + 2$ parameters $g(k)$ and $h(k)$. Therefore it can be physically realized, and now the basis has been orthogonalized.

Problem Set 6.4

1. Use the identity $\sin 2\theta = 2 \sin \theta \cos \theta$ to show that

$$\left(\cos \frac{\omega}{2}\right) \left(\cos \frac{\omega}{4}\right) \cdots \left(\cos \frac{\omega}{2^N}\right) = \frac{1}{2^N} \frac{\sin \omega}{\sin \frac{\omega}{2}} \frac{\sin \frac{\omega}{2}}{\sin \frac{\omega}{4}} \cdots \frac{\sin \frac{\omega}{2^{N-1}}}{\sin \frac{\omega}{2^N}}.$$

Cancel sines and let $N \rightarrow \infty$ to find a great infinite product:

$$\prod_1^\infty \cos \left(\frac{\omega}{2^j} \right) = \frac{1}{\omega} \sin \omega.$$

2. The Haar filter has $H(\omega) = \frac{1}{2}(1 + e^{-i\omega}) = e^{-i\omega/2} \cos \frac{\omega}{2}$. Use $\frac{\omega}{2}$ in Problem 1 to give a new proof for the infinite product (6.47) of $H(\omega/2^j)$:

$$\left(e^{-i\omega/4} \cos \frac{\omega}{4}\right) \left(e^{-i\omega/8} \cos \frac{\omega}{8}\right) \cdots = \left(e^{-i\omega/2} \frac{2}{\omega} \sin \frac{\omega}{2}\right) = \frac{1}{i\omega} (1 - e^{-i\omega}).$$

3. Suppose $H(\omega) = \frac{1}{4}(1 + e^{-i\omega})^2$. Find $\widehat{\phi}(\omega)$ and $\phi(t)$.
4. If $H(\omega)$ has p zeros at $\omega = \pi$, show that $\widehat{\phi}(\omega)$ has p zeros at $\omega = 2\pi n$ for each $n \neq 0$.

6.5 Biorthogonal Wavelets

This chapter has concentrated on orthogonal wavelets, coming from an orthogonal filter bank. The synthesis filters are transposes of the analysis filters. One multiresolution is all we need. The synthesis wavelets are the same, in this self-orthogonal case, as the analysis wavelets. But from *biorthogonal filters* we must expect *biorthogonal wavelets*.

We now meet a new scaling function $\tilde{\phi}(t)$. Its translates $\tilde{\phi}(t - k)$ will span a new lowpass space \tilde{V}_0 —different from V_0 . There is also a wavelet $\tilde{w}(t)$. The translates $\tilde{w}(t - k)$ span a complementary highpass space \tilde{W}_0 . The sum of those spaces will be $\tilde{V}_1 = \tilde{V}_0 + \tilde{W}_0$, the next space in the second multiresolution. To a large extent, the theory is achieved by inserting a tilde where appropriate. We want to indicate why this second scale of spaces \tilde{V}_j is needed.

Biorthogonality comes automatically with inverse matrices. The rows of a 2×2 matrix and the columns of its inverse are biorthogonal:

$$\begin{bmatrix} \text{row} & 1 \\ \text{row} & 2 \end{bmatrix} \begin{bmatrix} \text{column} & \text{column} \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Notice something pleasant. The product in the other order is still I . The right-inverse is also the left-inverse. This order involves *columns times rows*, which are full matrices:

$$\begin{bmatrix} \text{column} \\ 1 \end{bmatrix} \begin{bmatrix} \text{row} & 1 \end{bmatrix} + \begin{bmatrix} \text{column} \\ 2 \end{bmatrix} \begin{bmatrix} \text{row} & 2 \end{bmatrix} = I.$$

Those two column-row products are projections, and they add to I . These simple facts about 2×2 matrices have important parallels for biorthogonal filters and biorthogonal wavelets.

Filter banks display those parallels immediately. The analysis bank has filters $(\downarrow 2)H_0$ and $(\downarrow 2)H_1$. Those are rows 1 and 2. The synthesis bank has expanders before filters, $F_0(\uparrow 2)$ and $F_1(\uparrow 2)$. Those are columns 1 and 2. In the orthogonal case F_0 is H_0^T and F_1 is H_1^T . In the biorthogonal case we don't have transposes but we still have inverses. To understand the pattern for wavelets, we absolutely must return to multiresolution. One scale of spaces $V_0 \subset V_1 \subset \dots \subset V_j$ is too limited. We need two hierarchies of spaces, V_j in synthesis and \tilde{V}_j in analysis.

Tilde Notation *Does the tilde go on the analysis functions or the synthesis functions?* Both conventions are equally possible. We hope to agree with other authors! More and more, the tilde is going on the *analysis* functions. Then $f(t)$ is expanded in synthesis functions, which have no tilde. But the coefficients come from the analysis functions and have tildes:

$$\begin{aligned} f_0(t) &= \sum \tilde{a}_{0k} \phi(t - k) \text{ is in } V_0, \text{ with } \tilde{a}_{0k} = \int f(t) \tilde{\phi}(t - k) dt \\ f(t) &= \sum \sum \tilde{b}_{jk} w_{jk} \text{ is in } L^2, \text{ with } \tilde{b}_{jk} = \int f(t) \tilde{w}_{jk}(t) dt \end{aligned} \quad (6.52)$$

What does this mean for the filter banks that process the coefficients? Those filters use the letter H in analysis and F in synthesis. We will stay with H and F (rather than C and D) when discussing biorthogonal filters. An important result in this section is the Fast Wavelet Transform in equation (6.70), and its inverse (the biorthogonal IFWT) in Theorem 6.16.

Biorthogonal Multiresolution

This chapter began with orthogonal bases $\{\phi(t - k)\}$ for V_0 and $\{w(t - k)\}$ for W_0 . The equation $V_0 \oplus W_0 = V_1$ started a multiresolution. W_j was the orthogonal (!) complement of V_j inside V_{j+1} . All is well if $\phi(t)$ and $w(t)$ come from an orthogonal bank of FIR filters. Their translates are all orthogonal. They span perpendicular spaces and we have an orthogonal multiresolution.

All is *not* so well if $\phi(t)$ and $w(t)$ fail to have compact support. The filters fail to be FIR. Often this means that we have asked for too much! Instead of orthogonal bases, we should be content with stable bases. An outstanding example is the space of piecewise linear functions. The stable basis consists of the hat function $\hat{\phi}(t)$ and its translates. That basis is *not orthogonal*.

When the basis is not orthogonal, there is no reason to insist that W_0 must be orthogonal to V_0 . If we do, the multiresolution is called *semi-orthogonal* in Section 7.4, and we have “pre-wavelets”. But the important property is a stable basis $\{w(t - k)\}$. The highpass coefficients will construct $w(t)$.

Remember the pattern for perfect reconstruction. Coefficients are chosen so that $F_0(z)H_0(z)$ is halfband. When $\phi(t)$ is the hat function from $F_0(z) = \left(\frac{1+z^{-1}}{2}\right)^2$, the other factor $H_0(z)$ needs five coefficients. This means $N = 2$ but $\tilde{N} = 4$. The wavelet has 3-interval support. Then $\phi(t - k)$ and $w(t - k)$ span V_0 and W_0 , without orthogonality.

The new *analysis multiresolution* is the point of this section. The coefficients from $H_0(z)$ go into a different dilation equation, whose solution is the *analyzing function* $\tilde{\phi}(t)$:

$$\text{Analysis Dilation Equation: } \tilde{\phi}(t) = \sum_0^{\tilde{N}} 2h_0(k) \tilde{\phi}(2t - k). \quad (6.54)$$

The coefficients $h_0(k)$ add to 1 as before. The new multiresolution obeys the same conditions as before; just add a tilde. \tilde{V}_0 is spanned by $\{\tilde{\phi}(t - k)\}$. The space \tilde{V}_j is spanned by $\{\tilde{\phi}(2^j t - k)\}$.

They are clearly shift-invariant. The dilation equation (6.54) says that $\tilde{V}_0 \subset \tilde{V}_1$. Then also $\tilde{V}_j \subset \tilde{V}_{j+1}$. The highpass coefficients produce the wavelet:

$$\text{Analysis Wavelet Equation: } \tilde{w}(t) = \sum_0^N 2h_1(k) \tilde{\phi}(2t - k). \quad (6.55)$$

This wavelet is supported on $[0, \ell]$, where $2\ell = N + \tilde{N}$. That sum $N + \tilde{N}$ is the degree of the product filters $F_0(z)H_0(z)$ and $F_1(z)H_1(z)$. Those are symmetric halfband filters, and in the hat function example the degree is $N + \tilde{N} = 2 + 4$. Then $\ell = 3$ is odd. The four functions $\phi(t)$, $w(t)$, $\tilde{\phi}(t)$, $\tilde{w}(t)$ are graphed in Figure 6.6.

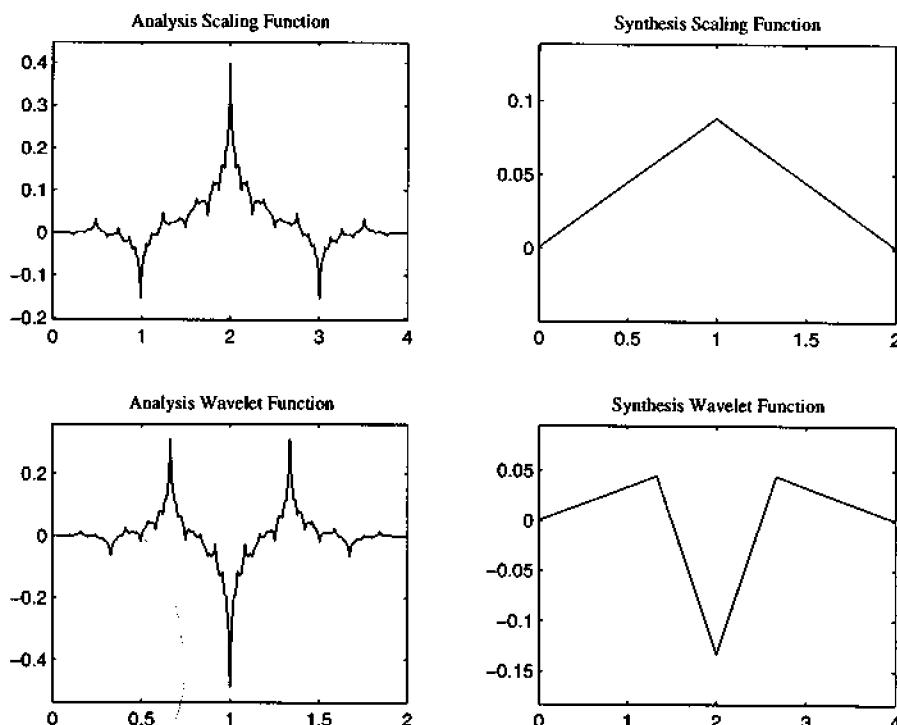


Figure 6.6: Biorthogonal scaling functions and wavelets from a 5/3 PR filter bank.

Biorthogonality in Continuous Time

Our construction of $\phi(t)$, $w(t)$, $\tilde{\phi}(t)$, $\tilde{w}(t)$ starts with biorthogonal filters. The lowpass analysis coefficients $h_0(k)$ are *not* double-shift orthogonal to themselves. They are double-shift **biorthogonal** to the lowpass synthesis coefficients $f_0(k)$:

$$2 \sum h_0(k)f_0(k + 2n) = \delta(n). \quad (6.56)$$

This means that $F_0(z)H_0(z)$ is halfband. Similarly the highpass filters give $F_1(z)H_1(z) =$ half-band:

$$2 \sum h_1(k)f_1(k + 2n) = \delta(n). \quad (6.57)$$

The other key relation is biorthogonality of highpass to lowpass:

$$\sum h_0(k)f_1(k+2n) = 0 \quad \text{and} \quad \sum h_1(k)f_0(k+2n) = 0. \quad (6.58)$$

The reader knows that all these equations restate perfect reconstruction:

$$F_0(z) = H_1(-z), \quad F_1(z) = -H_0(-z), \quad F_0(z)H_0(z) \text{ is a halfband filter.}$$

Our question is, *how does biorthogonality appear in continuous time?* The functions $\phi(t)$ and $\tilde{\phi}(t)$ come from iterating the lowpass filters F_0 and H_0 (with rescaling!). Start the cascade of iterations from $\phi^{(0)}(t) = \tilde{\phi}^{(0)}(t) = \text{box function on } [0, 1]$. Their translates have biorthogonality. The box $\phi^{(0)}(t-k)$ has no overlap with $\tilde{\phi}^{(0)}(t-\ell)$ when $k \neq \ell$. This biorthogonality is preserved at every iteration step, when we use equation (6.56). This is exactly parallel to the earlier proof (6.25) that orthogonality is preserved at each iteration. When $\phi^{(i)}(t)$ and $\tilde{\phi}^{(i)}(t)$ converge in L^2 to the scaling functions $\phi(t)$ and $\tilde{\phi}(t)$, those limit functions inherit the same biorthogonality:

$$\int_{-\infty}^{\infty} \phi(t-k)\tilde{\phi}(t-\ell) dt = \delta(k-\ell). \quad (6.59)$$

With $i \rightarrow \infty$ in the cascade algorithm, these limits $\phi(t)$ and $\tilde{\phi}(t)$ solve the synthesis and analysis dilation equations. Now bring in the wavelet equations:

$$\int_{-\infty}^{\infty} \phi(t)\tilde{w}(t) dt = \int_{-\infty}^{\infty} \left(\sum 2h_0(k)\phi(2t-k) \right) \left(\sum 2f_1(\ell)\tilde{\phi}(2t-\ell) \right) dt. \quad (6.60)$$

That right side is zero because of (6.58) and (6.59). And biorthogonality extends to the translates for the same reason:

$$\int_{-\infty}^{\infty} \phi(t-k)\tilde{w}(t-\ell) dt = 0 \quad \text{for all } k \text{ and } \ell. \quad (6.61)$$

Finally the wavelets w and \tilde{w} are biorthogonal from the wavelet equations and (6.57):

$$\int_{-\infty}^{\infty} w(t-k)\tilde{w}(t-\ell) dt = \delta(k-\ell). \quad (6.62)$$

All this is routine, *provided the cascade algorithms for $\phi(t)$ and $\tilde{\phi}(t)$ both converge in L^2 .* Wavelet theory gives the requirements in Section 7.2, as tests on eigenvalues of two matrices T and \tilde{T} . Suppose those tests are passed (not at all automatic!). The basis functions are biorthogonal. What does this say about the subspaces they span?

Theorem 6.14 *Suppose the filter coefficients satisfy (6.56)–(6.58) and also Condition E (for L^2 convergence of the cascade algorithm). Then the synthesis functions $\phi(t-k)$, $w(t-k)$ are biorthogonal to the analysis functions $\tilde{\phi}(t-\ell)$, $\tilde{w}(t-\ell)$ as in (6.59)–(6.62). Each scaling space is orthogonal to the dual wavelet space:*

$$V_j \perp \tilde{W}_j \quad \text{and} \quad W_j \perp \tilde{V}_j. \quad (6.63)$$

V_0 and \tilde{W}_0 are perpendicular because their bases $\phi(t-k)$ and $\tilde{w}(t-k)$ are perpendicular. When t is replaced by $2^j t$, the zero inner products are still zero. (Change variables back to $T = 2^j t$.) So at each scale j we have perpendicular subspaces.

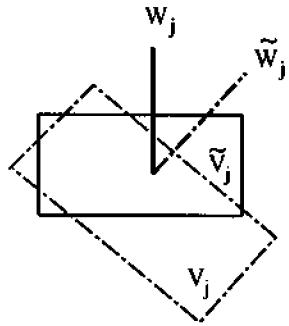


Figure 6.7: V_j is perpendicular to \tilde{W}_j , while \tilde{V}_j is perpendicular to W_j .

The two multiresolutions are intertwining. As always we have

$$V_j + W_j = V_{j+1} \quad \text{and} \quad \tilde{V}_j + \tilde{W}_j = \tilde{V}_{j+1}. \quad (6.64)$$

These are direct sums but generally not orthogonal sums. The subspaces V_j and W_j have zero intersection, but they are not perpendicular. Instead V_j is perpendicular to \tilde{W}_j . All the subspaces W_{j-1}, W_{j-2}, \dots are then perpendicular to \tilde{W}_j . Similarly all the subspaces $\tilde{W}_{j-1}, \tilde{W}_{j-2}, \dots$ are perpendicular to W_j . Therefore we have *biorthogonal bases (dual bases)*:

Corollary *The wavelets $w_{jk}(t) = 2^{j/2}w(2^j t - k)$ and $\tilde{w}_{jk}(t) = 2^{j/2}\tilde{w}(2^j t - k)$ are biorthogonal bases for L^2 :*

$$\int_{-\infty}^{\infty} w_{jk}(t)\tilde{w}_{JK}(t) dt = \delta(j - J)\delta(k - K). \quad (6.65)$$

Representing $f(t)$ in a Wavelet Series

If we have a wavelet basis, we have a wavelet series. Any square-integrable (finite energy) function $f(t)$ can be expanded in wavelets:

$$f(t) = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} \tilde{b}_{jk} w_{jk}(t). \quad (6.66)$$

The synthesis wavelets are used to synthesize the function (of course). But the coefficients \tilde{b}_{jk} come from inner products with the analysis wavelets. *This is why \tilde{b}_{jk} has a tilde.* Multiply (6.66) by the analysis wavelet $\tilde{w}_{JK}(t)$ and integrate over t . Biorthogonality yields

$$\tilde{b}_{JK} = \int_{-\infty}^{\infty} f(t) \tilde{w}_{JK}(t) dt. \quad (6.67)$$

Equations (6.66) and (6.67) are the biorthogonal wavelet transform and its inverse. The transform takes function to coefficients, the inverse transform synthesizes the function. We show below how the coefficients can be computed recursively (or pyramidal). This is the *fast* wavelet transform.

Note that Parseval's equality between $\int |f(t)|^2 dt$ and $\sum \sum |b_{jk}|^2$ is *not true*. That required orthogonality — and not biorthogonality. When we square and integrate the series for $f(t)$, non-zero inner products come from the products $w_{jk}(t)w_{JK}(t)$. We do have an inequality

$$A \int_{-\infty}^{\infty} |f(t)|^2 dt \leq \sum \sum |\tilde{b}_{jk}|^2 \leq B \int_{-\infty}^{\infty} |f(t)|^2 dt. \quad (6.68)$$

With $B \geq A > 0$ this says we have a stable basis or *Riesz basis*. This is true for wavelets, subject to the same Condition E [Cohen-Daubechies].

Fast Biorthogonal Wavelet Transform

The reader knows that in practice the wavelet expansion cannot go all the way back to $j = -\infty$. We do not use arbitrarily low frequencies (longer and longer wavelets). A more practical expansion starts with V_0 and \tilde{V}_0 , at a scale normalized to $\Delta t = 1$, and goes to V_J and \tilde{V}_J , where the finer scale is 2^{-J} . Enough high-frequency details are included to reproduce an accurate signal.

Thus we work primarily with the subspaces $V_j = V_0 + W_0 + \dots + W_{j-1}$. There are two important bases for V_j . One is $\phi_{jk}(t) = 2^{j/2}\phi(2^j t - k)$ for $-\infty < k < \infty$. These scaling functions are *at level J*. The other basis consists of $\phi_{0k}(t)$ from V_0 and $w_{jk}(t)$ for $0 \leq j < J$. Since life is recursive, we are interested first of all in $J = 1$. Then the two ways to expand a signal in V_1 are the scaling basis (fine scale) or scaling functions plus wavelets (coarse scale):

$$\sum \tilde{a}_{1k} \sqrt{2} \phi(2t - k) = \sum \tilde{a}_{0k} \phi(t - k) + \sum \tilde{b}_{0k} w(t - k). \quad (6.69)$$

The key is the pyramid algorithm. This connects \tilde{a}_{1k} to \tilde{a}_{0k} and \tilde{b}_{0k} . We are using ϕ and w because this is synthesis of the signal. But the coefficients come from *analysis* of the signal, which uses $\tilde{\phi}$ and \tilde{w} :

$$\tilde{a}_{0k} = \int f(t) \tilde{\phi}(t - k) dt = \int f(t) \sum_{\ell} h_0(\ell - 2k) \tilde{\phi}_{1\ell}(t) dt.$$

For \tilde{b}_{0k} we use the wavelet equation with $h_1(\ell - 2k)$ instead of the dilation equation with $h_0(\ell - 2k)$. The pyramid has filtering and downsampling:

$$\text{Fast Wavelet Transform } \tilde{a}_{0k} = \sum h_0(\ell - 2k) \tilde{a}_{1\ell} \text{ and } \tilde{b}_{0k} = \sum h_1(\ell - 2k) \tilde{a}_{1\ell}. \quad (6.70)$$

This includes a time-reversal! The same filters H_0^T and H_1^T operate at level j .

That change of basis was not orthogonal. Going backwards, the inverse will not be the transpose. The synthesis filters F_0 and F_1 must do their part.

Theorem 6.15 Inverse Fast Wavelet Transform: *The coefficients $\tilde{a}_{1\ell}$ in the basis $\phi_{1\ell}(t)$ can be computed from \tilde{a}_{0k} and \tilde{b}_{0k} by time-reversed synthesis filters:*

$$\tilde{a}_{1\ell} = \sum_k f_0(\ell - 2k) \tilde{a}_{0k} + \sum_k f_1(\ell - 2k) \tilde{b}_{0k}. \quad (6.71)$$

Proof. Perfect reconstruction operates when (6.70) is substituted into (6.71):

$$\tilde{a}_{1\ell} = \sum_k f_0(\ell - 2k) \sum_m h_0(\ell - 2m) \tilde{a}_{1m} + \sum_k f_1(\ell - 2k) \sum_m h_1(\ell - 2m) \tilde{a}_{1m} \quad (6.72)$$

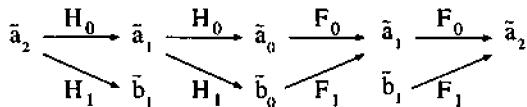


Figure 6.8: The pyramid algorithm from \tilde{a}_2 back to \tilde{a}_2 .

The double-shift biorthogonality in (6.56) and (6.57) makes this correct. The beauty of this Mallat algorithm (Figure 6.8 goes up and back down) is the way it connects continuous-time multiresolution to discrete-time filters.

Notice that the whole pyramid operates equally well if F_0 and F_1 are exchanged with H_0 and H_1 . The dual expansion of $f(t)$ is:

$$f(t) = \sum \sum b_{jk} \tilde{w}_{jk}(t) \quad \text{and} \quad b_{jk} = \int f(t) w_{jk}(t) dt. \quad (6.73)$$

All products have a tilde multiplying a non-tilde! This starts with the inverse relation of synthesis to analysis. Tildes can be exchanged with non-tildes throughout (if we want to do it). We will select ϕ and w to be effective in synthesis.

We emphasize one final point, often ignored. The *coefficients* in the expansion of $f(t)$ are really different from *samples* of $f(t)$. They are inner products, not point values. This distinction is made in Section 7.1.

Filter Construction by Lifting

Herley and Sweldens have proposed (independently) a systematic way to construct biorthogonal filter banks. Only one lowpass filter changes at each step. Starting from short filters, he quickly builds longer ones. In all cases the highpass filters remove aliasing in the standard way: $H_1(z) = F_0(-z)$ and $F_1(z) = -H_0(-z)$. Then equation (4.9) on H_0 and F_0 is the remaining condition for perfect reconstruction. *We drop the subscript zero on these lowpass filters*, and recall the condition (4.9) that removes distortion:

$$F(z)H(z) - F(-z)H(-z) = 2z^{-l} \quad (\text{odd } l). \quad (6.74)$$

Suppose this is satisfied by F and H ; the filter bank is PR. *Keeping F fixed, what are the other possible choices for H?* The answer is simple and important:

Theorem 6.16 (Lifting) *For fixed $F(z)$, the solutions $H^*(z)$ to (6.74) are*

$$H^*(z) = H(z) + F(-z)S(z^2) \quad \text{for any } S(z). \quad (6.75)$$

Proof. Substitute $H^*(z)$ to show that equation (6.74) is still satisfied. The new terms are $F(z)F(-z)S(z^2) - F(-z)F(z)S(z^2) = 0$. This is in [HerVet] and [Swel].

Note that with F fixed, the equation is linear in H . We are starting from a particular solution (right side = $2z^{-l}$). To this particular H we are adding solutions $F(-z)S(z^2)$ to the homogeneous equation (right side = zero). Thus the even $S(z^2)$ displays the degrees of freedom that remain when the PR condition is satisfied (Problem 5). That freedom is used in the Daubechies construction to achieve zeros at $z = -1$.

We still want and need zeros at -1 , which is $\omega = \pi$. We also need stable bases. (Section 7.2 will state the stability requirement as Condition E. There is not yet a simple way to decide which $S(z^2)$ are permitted by this condition. In practice, we construct a potentially useful $H^*(z)$ and test it for Condition E.) This section will build in other important properties — *linear phase, interpolating scaling functions, binary (dyadic) filter coefficients*. Those come at the expense of higher-order zeros at π .

Dual lifting is also useful. In this case we fix the analysis filter $H(z)$. The PR condition (6.74) becomes linear in $F(z)$. The lifted solutions are

$$F^*(z) = F(z) + H(-z)T(z^2) \quad \text{for any } T(z). \quad (6.76)$$

Starting from the Haar filter or even from the “Lazy filter”, we alternate lifting and dual lifting to construct high-order biorthogonal filter banks with good properties. All these filters would be attainable directly from the (second-degree) PR equations. Lifting is a way to solve them as a sequence of linear equations, with F or H fixed at each step. Then we control more closely the final result.

Sweldens also emphasizes that lifting yields a *faster implementation* of the wavelet transform and its inverse. The Mallat filter tree, which is subband filtering, is climbed in smaller steps. This is related to a lattice factorization.

Example 6.6. The Lazy filter has $H(z) = 1$ and $F(z) = z^{-1}$. There is no true filtering, only subsampling from $(\downarrow 2)$. Section 4.3 displayed block diagrams of this filter bank. Its polyphase matrix is $H_p = I$.

Suppose we keep $H(z) = 1$ and apply dual lifting to the synthesis filter:

$$F^*(z) = z^{-1} + T(z^2). \quad (6.77)$$

$F^*(z)$ can be any halfband filter, with one odd power $z^{-l} = z^{-1}$. We are allowing S and T to contain powers of z as well as z^{-1} . This is needed in order to create symmetric filters.

Earlier we centered the product $H(z)F(z)$, multiplying by z^l . In this case centering gives $zF^*(z) = 1 + zT(z^2)$. Then D_4 comes from $T(z) = (-z + 9 + 9z^{-1} - z^2)/16$. Every maxflat halfband filter D_{2p} can be lifted and centered from the Lazy filter. This section will create symmetric biorthogonal filters, by lifting $H = 1$ while $F = D_{2p}$.

Note the scaling functions for this important example. In analysis we have the delta function $\tilde{\phi}(t) = \delta(t)$ from $H(z) = 1$. This solves the dilation equation $\delta(t) = 2\delta(2t)$. In synthesis F^* yields an *interpolating scaling function*, with $\phi(n) = \delta(n)$. This is certainly biorthogonal to the analysis functions!

$$\int \phi(t)\tilde{\phi}(t-n) dt = \int \phi(t)\delta(t-n) dt = \phi(n) = \delta(n). \quad (6.78)$$

You can see in another way that $\phi(n) = \delta(n)$, because the values of ϕ at the integers come from the $\lambda = 1$ eigenvector of $(\downarrow 2)2F^*$. When the filter is halfband, the center column of the matrix $2F^*$ is the vector δ . This is an eigenvector with $\lambda = 1$. Assuming a stable basis, there are no other eigenvectors for $\lambda = 1$ by Condition E. So $\phi(n)$ agrees with $\delta(n)$.

This interpolating property is highly useful in several applications. But the analysis filter with $H(z) = 1$ and $\tilde{\phi}(t) = \delta(t)$ is generally not acceptable. Therefore we now lift H . That produces a new pair (H^*, F^*) which seems extremely promising.

Biorthogonal Filters with Binary Coefficients

A *binary coefficient* or *dyadic coefficient* is an integer divided by a power of 2. The maxflat halfband filters $D_{2p}(z)$ all have binary coefficients. This is clear from the Daubechies formula (5.75), where the binomial coefficients are integers. Multiplication by a binary number can be executed entirely by *shifts* and *adds*. Roundoff error is eliminated. And on some architectures, the filter needs less time and less space.

We are therefore highly interested in binary filters. Most factorizations of D_{2p} —this has been our route to orthogonal and biorthogonal filters— are *not* binary. There are zeros at irrational points like $z = 2 - \sqrt{3}$. But we can certainly move zeros at $z = -1$ between analysis and synthesis. This operation we call *balancing*.

Moving $(\frac{1+z^{-1}}{2})$ from $F(z)$ to $H(z)$ maintains binary coefficients and symmetry:

$$h_{new}(n) = \frac{1}{2}[h_{old}(n) + h_{old}(n-1)] \quad \text{and} \quad f_{new}(n) = \frac{1}{2}[f_{old}(n) - f_{new}(n-1)]. \quad (6.79)$$

Note f_{new} at the end. We are dividing $F(z)$ by $(\frac{1+z^{-1}}{2})$. The product $F_{new}H_{new}$ equals $F_{old}H_{old}$, so biorthogonality is preserved. The scaling function $\phi_{new}(t)$ is $\tilde{\phi}_{old}(t)$ convolved with the box function. Therefore it has exactly one more derivative than $\phi_{old}(t)$ (Section 7.3). Similarly $\phi_{new}(t)$ from F_{new} has one less derivative than $\phi_{old}(t)$ from F_{old} . In a filter bank we avoid the destructive factors $\sqrt{2}$, by putting both of them into synthesis. Our convention below is $H(1) = 1$ and $F(1) = 2$.

Our example $h1 = [1]$ and $f7 = [-1 \ 0 \ 9 \ 16 \ 9 \ 0 \ -1]/16$ is binary. This 1/7 filter bank has denominator 16. *Balancing will produce 2/6 and 3/5, still binary and symmetric:*

$$h2 = [1 \ 1]/2 \quad \text{and} \quad f6 = [-1 \ 1 \ 8 \ 8 \ 1 \ -1]/8 \quad \text{with } 1/3 \text{ zeros at } \pi$$

$$h3 = [1 \ 2 \ 1]/4 \quad \text{and} \quad f5 = [-1 \ 2 \ 6 \ 2 \ -1]/4 \quad \text{with } 2/2 \text{ zeros at } \pi.$$

These are very effective short filters. They are probably the best —after reversing the second pair to 5/3. In the experiments of Chapters 10 and 11 they are comparable and quite effective. As factors of the maxflat D_6 filter, we have seen them before. An interesting feature of $f5$ is that its scaling function $\phi(t)$ is infinite at all binary points! Section 7.3 confirms that this $\phi(t)$ nevertheless has finite energy (and even 0.44 derivatives in the energy sense). By removing two zeros at $z = -1$ from $f7$, we have stolen its smoothness. Enough is left to make it good among short filters (but moved to the analysis side).

For serious compression we need more zeros — which means longer filters. Our previous route was to factor a long maxflat filter. When one factor is $F(z) = (\frac{1+z^{-1}}{2})^p$ the pair is still binary and symmetric. The scaling function $\phi(t)$ for this factor is a *spline*. It has maximum smoothness for its length (p intervals) coming from a maximum number of zeros at π (p zeros). These are outstanding filters, when we keep enough smoothness in the other factor. Taking out three zeros from D_6 would leave a filter $[-1 \ 3 \ 3 \ -1]/2$ which has one zero but negative smoothness — too risky to iterate. Taking three zeros from D_8 is allowed. Now, instead of factoring a long filter, we will lift a short filter.

Lifting will not maximize the number of zeros at π (although we like those zeros). Our first lifting will go from 1/7 to 9/7, and we choose $S(z^2)$ in (6.74) to give two zeros at π . Here is the result of lifting $H = 1$ (all filters are symmetric):

$$h9 = [1 \ 0 \ -8 \ 16 \ 46 \ \dots]/64 \quad \text{and} \quad f7 = [-1 \ 0 \ 9 \ 16 \ \dots]/16 \quad (2/4 \text{ zeros}).$$

These are binary filters. Balancing the zeros by (6.79) yields 10/6 from 9/7:

$$h_{10} = [1 \ 1 \ -8 \ 8 \ 62 \ \dots]/128 \text{ and } f_6 = [-1 \ 1 \ 8 \ 8 \ 1 \ -1]/8 \text{ (3/3 zeros).}$$

This 10/6 pair gives better compression as 6/10—reversing analysis and synthesis. So does 5/11 from 11/5, after another balancing (or unbalancing!) step:

$$h_{11} = [1 \ 2 \ -7 \ 0 \ 70 \ 124 \ \dots]/256 \text{ and } f_5 \text{ above (4/2 zeros).}$$

Figure 6.9 shows scaling functions $\tilde{\phi}(t)$ and $\phi(t)$ for analysis and synthesis. You can see how the zeros affect the smoothness. You cannot easily see which pair is best in compression—that depends on the image.

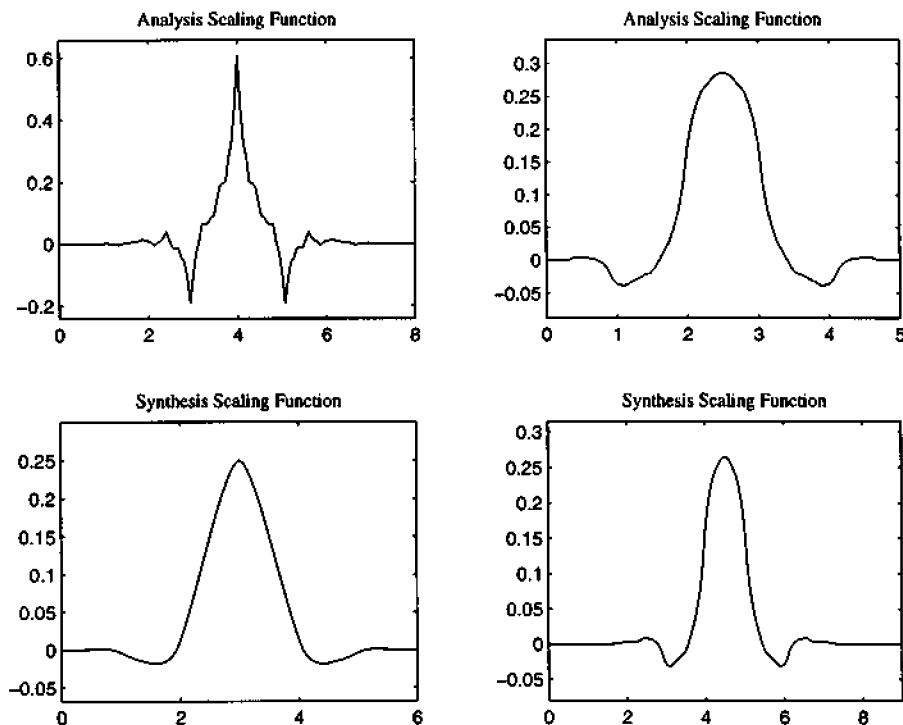


Figure 6.9: Scaling functions for h_9/f_7 and h_6/f_{10} .

Note 1 The reader might be interested in the construction of these new (1995) binary filters. The first author found the 9/7 pair in September, but not by lifting. With f_7 fixed, he solved the halfband equation $(\downarrow 2)(f_7 * h_9) = \delta$ for the symmetric filter h_9 with two zeros at π . (The first zero determines the middle coefficient from the others; the second zero is automatic by symmetry.) When reporting this result for the *Wavelet Digest*, he learned that Wim Sweldens had created a whole family of *binary symmetric filters* earlier in 1995 by lifting. We propose to call them “binlets”. Here are the next filters h_{13}/f_7 and $13h/f_{11}$. All signs indicate that h_{13}/f_7 is the right choice:

$$h_{13} = [-1 \ 0 \ 18 \ -16 \ -63 \ 144 \ 348 \ \dots]/512 \text{ with } f_7 \text{ (4/4 zeros)}$$

$$13h = [-3 \ 0 \ 22 \ 0 \ -125 \ 256 \ 724 \ 256 \ -125 \ 0 \ 22 \ 0 \ -3]/1024 \text{ with}$$

$$f11 = D_6 = [3 \ 0 \ -25 \ 0 \ 150 \ 256 \ 150 \ 0 \ -25 \ 0 \ 3]/256 \text{ (2/6 zeros).}$$

Extra length gives more zeros and higher compression, up to a point. *Then ringing destroys the image quality.* See Sections 10.1 and 11.2 for the artifacts that plague image compression. The boats in Figure 7.4 offer a visual comparison.

Note 2 [Majani2] emphasizes the importance of a *reversible integer implementation*. Integer inputs are reconstructed exactly. Orthogonal transforms seem not to be reversible (except Haar for $M = 2$ channels and Hadamard for certain $M > 2$). The 2/6 biorthogonal transform with $h2 = [1 \ 1]$ is reversible and very useful. Lowpass components y_{2i} come first:

$$y_{2i} = x_{2i} + x_{2i+1} \text{ and then } y_{2i+1} = x_{2i+1} - \lfloor y_{2i}/2 \rfloor + \lfloor (y_{2i-2} - y_{2i+2})/16 \rfloor.$$

The inverse also has “even = even + $f(\text{odd})$ ” and “odd = odd + $g(\text{even})$ ”:

$$x_{2i} = y_{2i} - x_{2i+1} \text{ and then } x_{2i+1} = y_{2i+1} + \lfloor y_{2i}/2 \rfloor - \lfloor (y_{2i-2} - y_{2i+2})/16 \rfloor.$$

Majani has shown that the new binary 9/7 and 13/7 transforms have reversible forms (lossless in integers). The normalized DCT is not reversible for integer data.

Note 3 The maxflat Daubechies filters with $4p - 1$ coefficients and $2p$ zeros are also known as Deslauriers-Dubuc filters [DesDub, CDM]. They interpolate because they are halfband. They leave the values $x(n)$ unchanged and produce midpoint values $x(n + \frac{1}{2})$. Section 5.5 confirmed that all polynomials of degree less than $2p$ are interpolated exactly. Recursive subdivision starting from $x(n) = \delta(n)$ converges to the scaling function $\phi(t)$ by the cascade algorithm!

Problem Set 6.5

1. Double-shift orthogonality of lowpass filters is $2 \sum h_0(k)f_0(k + 2n) = \delta(n)$. Show that in frequency this becomes

$$H_0(\omega)F_0(\omega) + H_0(\omega + \pi)F_0(\omega + \pi) = 1.$$

Write the same equation in the z -domain.

2. Problem 1 involves a row and column of the modulation matrices F_m and H_m :

$$F_m(z)H_m(z) = \begin{bmatrix} F_0(z) & F_1(z) \\ F_0(-z) & F_1(-z) \end{bmatrix} \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix}.$$

Which row-column multiplications correspond to which equations (6.56)–(6.58)?

3. Suppose $H_0(\omega)F_0(\omega) + H_0(\omega + \pi)F_0(\omega + \pi) = 1$ as required. By alternating flip

$$H_1(\omega) = e^{-i\omega}F_0(\omega + \pi) \quad \text{and} \quad F_1(\omega) = -e^{-i\omega}H_0(\omega + \pi).$$

Show that the other entries of $F_m(z)H_m(z) = I$ are then correct.

4. What wavelets come from the biorthogonal filters with $H_0 = 1$, $F_0 = \frac{1}{2}z + 1 + \frac{1}{2}z^{-1}$, $H_1 = \frac{1}{2}z - 1 + \frac{1}{2}z^{-1}$, $F_1 = -1$? Recognize the delta and hat:

$$\tilde{\phi}(t) = 2\tilde{\phi}(2t) \quad \text{and} \quad \phi(t) = \frac{1}{2}\phi(2t + 1) + \phi(2t) + \frac{1}{2}\phi(2t - 1).$$

Then construct wavelets from $\tilde{w}(t) = -\frac{1}{2}\tilde{\phi}(2t+1) + \tilde{\phi}(2t) - \frac{1}{2}\tilde{\phi}(2t-1)$ and $w(t) = 2\phi(2t-1)$. Check the biorthogonality conditions

$$\begin{aligned} \int \phi(t)\tilde{\phi}(t-k) dt &= \int w(t)\tilde{w}(t-k) dt = \delta(k) \quad \text{and} \\ \int \phi(t)\tilde{w}(t-k) dt &= \int \tilde{\phi}(t)w(t-k) dt = 0. \end{aligned}$$

5. Lifting from the Haar filters $H(z) = \frac{1}{2}(1+z^{-1})$ and $F(z) = 1+z^{-1}$, show that all PR solutions have the form (6.75). The difference $D = H^* - H$ must satisfy $D(z)F(z) = D(-z)F(-z)$ from (6.74). Substitute $D(z) = a + bz^{-1} + cz^{-2} + dz^{-3} + \dots$ to show that it has the form $F(-z)S(z^2)$.
6. Lifting (H, F) to (H^*, F) does not change the scaling function $\phi(t)$. Show that the new wavelet is $w^*(t) = w(t) - \sum s(k)\phi(t-k)$.
7. The *fast wavelet transform* is subband filtering of the inner products $a_{jk} = \langle f(t), \tilde{\phi}_{jk}(t) \rangle$. The highpass channel produces $b_{jk} = \langle f(t), \tilde{w}_{jk}(t) \rangle$ by (6.70):

$$a_{jk} = \sum h(l-2k)a_{j+1,l} \quad \text{and} \quad b_{jk} = \sum h_1(l-2k)a_{j+1,l}.$$

Suppose H is lifted to H^* . Show that the lifted a_{jk}^* are $a_{jk}^* = a_{jk} + \sum s(l-k)b_{jl}$. The fast inverse transform unlifts by $-s(l-k)$ in the same way. Then it inverts the (H, H_1) transform as usual by $a_{j+1,k} = \sum f(k-2l)a_{jl} + \sum f_1(k-2l)b_{jl}$.

8. From the input $x(n) = a_{1k}$ compute even samples $a_{0k} = a_{1,2k}$ and odd $b_{0k} = a_{1,2k+1} - (a_{0k} + a_{0,k+1})$. Then lift to $a_{0k}^* = a_{0k} + (b_{0,k-1} + b_{0k})/4$. Combine to recognize the 5/3 filter bank, computed more efficiently and in place — no auxiliary memory.
9. Which filter h gives linear interpolation at each step of recursive subdivision? What is $\phi(t)$?

Chapter 7

Wavelet Theory

This chapter follows through on important questions about scaling functions and wavelets. We study their smoothness, their inner products, their accuracy of approximation, and the number of vanishing moments. Even more basic are the existence and the construction of $\phi(t)$. *Does the dilation equation have a solution $\phi(t)$ with finite energy? Does the cascade algorithm converge to this solution?* We answer those questions in terms of the filter coefficients $h(k)$ (and the answers are not always yes).

Our overall aim is understanding, with a minimum of technical details. We point immediately to two fundamental operators in wavelet theory. They both have a double shift between rows coming from ($\downarrow 2$):

$$\mathbf{M} = (\downarrow 2) \mathbf{H} \quad \text{and} \quad \mathbf{T} = (\downarrow 2) \mathbf{H} \mathbf{H}^T. \quad (7.1)$$

Apart from its extra factor 2, \mathbf{M} is completely familiar: filter by \mathbf{H} and then downsample. This is the decimated lowpass analysis filter, with coefficients $h(k)$ adding to 1. In the time domain, those are on the diagonals of \mathbf{H} .

The symmetric product $\mathbf{H} \mathbf{H}^T$ is also a Toeplitz matrix. Its entries are the coefficients in $|H(\omega)|^2$. The rows are double-shifted by ($\downarrow 2$). In frequency, downsampling produces an aliasing term from modulation by π :

$$\begin{aligned} (\mathbf{M}f)(\omega) &= H\left(\frac{\omega}{2}\right)f\left(\frac{\omega}{2}\right) + H\left(\frac{\omega}{2} + \pi\right)f\left(\frac{\omega}{2} + \pi\right) \\ (\mathbf{T}f)(\omega) &= |H\left(\frac{\omega}{2}\right)|^2 f\left(\frac{\omega}{2}\right) + |H\left(\frac{\omega}{2} + \pi\right)|^2 f\left(\frac{\omega}{2} + \pi\right) \end{aligned} \quad (7.2)$$

The properties of \mathbf{M} and \mathbf{T} hold the answers to our questions. Iterating the lowpass filter, with subsampling, involves powers of matrices. The convergence depends on the eigenvalues. That will be the message in this chapter: *Watch the eigenvalues.*

The “transition operator” or “transfer operator” \mathbf{T} turns out to be simpler than \mathbf{M} — because $|H(\omega)|^2 \geq 0$ and $\mathbf{H} \mathbf{H}^T$ is symmetric positive definite. \mathbf{T} enters when we compute inner products and energies (L^2 norms). After i iterations of \mathbf{M} in the cascade algorithm, starting from $\phi^{(0)}(t)$ and reaching $\phi^{(i)}(t)$, the inner products are

$$a^{(i)}(k) = \int_{-\infty}^{\infty} \phi^{(i)}(t) \phi^{(i)}(t+k) dt. \quad (7.3)$$

The key point will be that $Ta^{(i)}(k)$ gives the inner products $a^{(i+1)}(k)$. The powers of \mathbf{T} (and therefore the eigenvalues of \mathbf{T} !) decide whether the cascade algorithm converges, when we use

the L^2 energy norm. The properties of M give *pointwise* answers, and the properties of T give *mean square* answers.

We can summarize in a few lines approximately what those answers are:

1. Combinations of $\phi(t - k)$ can exactly produce the polynomials $1, t, \dots, t^{p-1}$ if M has eigenvalues $1, \frac{1}{2}, \dots, (\frac{1}{2})^{p-1}$; T has eigenvalues $1, \frac{1}{2}, \dots, (\frac{1}{2})^{2p-1}$.
2. The wavelets are orthogonal to $1, \dots, t^{p-1}$ so they have p vanishing moments. The functions $\phi(t - k)$ give p th order approximation from the space V_0 .
3. The cascade algorithm converges to $\phi(t)$ in L^2 if the other eigenvalues of T satisfy $|\lambda| < 1$.
4. $\phi(t)$ and $w(t)$ have s derivatives in L^2 if the other eigenvalues of T satisfy $|\lambda| < 4^{-s}$.

In 1, the special eigenvalues (powers of $\frac{1}{2}$) give a new form of Condition A_p . It is equivalent to p zeros at π , from a factor $(1 + z^{-1})^p$ in $H(z)$.

In 3, the requirement $|\lambda| < 1$ on the other eigenvalues will be called **Condition E**. This is new to the book. It gives convergence of the cascade algorithm to a Riesz basis $\{\phi(t - k)\}$. Condition E is the key to iteration of filters, and thus to wavelets.

In 4, the smoothness s is very likely not an integer (but we work with integers for simplicity). The number s_{max} of derivatives of $\phi(t)$ is never greater than the number p of vanishing moments. The smoothness of $\phi(t)$ is important in image processing, and p is more important.

This chapter begins with the *accuracy of approximation*. That has the most direct answer. The approximation order is p when the frequency response $H(\omega)$ has a p th order zero at $\omega = \pi$. Zeros at π are the heart of wavelet theory! This multiple zero is produced by the factor $(1 + e^{-i\omega})^p$ that appears throughout the design of filters. T is associated with $|H(\omega)|^2$ for which the order of the zero becomes $2p$. This is reflected in the $2p$ special eigenvalues of T .

Sections 7.2–7.3 analyze the cascade algorithm $\phi^{(i+1)}(t) = \sum 2h(k) \phi^{(i)}(2t - k)$. Section 7.4 explains splines and spline wavelets. Section 7.5 introduces multiwavelets.

We note here the MATLAB commands to construct M and T from the vector h :

```
function M = down(h)
n = length(h); M = zeros(n,n); for i = 1:n, for j = 1:n,
if (0 < 2 * i - j) & (2 * i - j) <= n) M(i,j) = 2 * h(2 * i - j);
end
end, end
autocor = conv(h, fliplr(h)); T = down(autocor)
```

7.1 Accuracy of Approximation

In applications of wavelets, a function $f(t)$ is projected onto a scaling space V_j . This index j gives the time scale $\Delta t = 2^{-j}$ in the calculation. The scaling functions are $2^{j/2}\phi(2^j t)$ and its translates by $k\Delta t$. Those represent one basis for the space V_j . The projection $f_j(t)$ is the piece of $f(t)$ in that subspace, so it is a combination of those basis functions:

$$\text{For each } j, \quad f_j(t) = \sum_{k=-\infty}^{\infty} a_{jk} 2^{j/2} \phi(2^j t - k). \quad (7.4)$$

This is all at level j . In contrast, wavelets come from splitting functions into several scales. **Multiresolution** combines the details at levels zero through $j - 1$ and the coarse average at level zero. For subspaces this is $V_j = V_0 \oplus W_0 \oplus \cdots \oplus W_{j-1}$. Except for V_0 , the basis functions are now wavelets:

$$f_j(t) = \sum_k a_{0k} \phi(t - k) + \sum_k b_{0k} w(t - k) + \sum_k b_{1k} 2^{1/2} w(2t - k) + \cdots \quad (7.5)$$

In practice, the level j is determined by balancing *accuracy* with *cost*. In other words, we balance *distortion* with *rate*. The cost and the bit rate are approximately doubled between one level and the next — there are twice as many basis functions and twice as many coefficients. This section will estimate the improvement in accuracy (the drop in distortion).

The accuracy depends partly on the filter bank coefficients and partly on $f(t)$. A smooth function and a smooth signal will be easier to approximate and send. Part of our goal is to separate the two contributions to the error (the distortion), one part from the properties of $h(k)$ and the other from the properties of $f(t)$. We choose $h(k)$. The application presents us with $f(t)$. A typical form of the error estimate will involve the p th derivative of $f(t)$:

$$\|f(t) - f_j(t)\| \approx C(\Delta t)^p \|f^{(p)}(t)\|. \quad (7.6)$$

The constant C and the exponent p depend on our choice of $h(k)$. That determines $\phi(t)$ and the subspaces. The step from $\Delta t = 2^{-j}$ to $\Delta t = 2^{-(j+1)}$ divides the error by about 2^p . Thus the number p is critical, when we reach the level at which this “asymptotic” error estimate is accurate. Usually the constant C is less critical, but for wavelets it is an order of magnitude larger than for splines.

The smoothness or roughness of $f(t)$ may be outside our control. This contributes to the error through the norm $\|f^{(p)}(t)\|$ of the p th derivative. The global error estimate (7.6) can be made *local*, if there are regions where $f^{(p)}(t)$ is small and also regions of sudden change. The error will be small in one region and large in the other — unless we increase j in the region of sudden change. Then we have an *adaptive mesh*.

A major advantage of wavelets over Fourier methods is this possibility of local refinement. This is the multigrid idea for finite differences, and it is a key virtue of finite elements. Adaptivity holds also for wavelets — mesh refinement is relatively convenient. (The refinement is usually by factors of 2. Irregular meshes are anathema to Fourier.) We add the word “relatively” because adaptivity has an overhead that practitioners have come to respect.

The error estimate (7.6) is completely typical of numerical analysis. It appears in finite elements, where Δt becomes the element size [SF]. It appears in difference methods for initial-value problems ($p = 1$ for Euler’s method, $p = 2$ for centered leapfrog, $p = 4$ for Runge-Kutta, etc.). The form of (7.6) is already set by the basic problem of numerical integration:

$$\int_0^1 f(t) dt - \sum c_k f(t_k) \approx \begin{cases} (\Delta t)^1 : & \text{rectangle rule} \\ (\Delta t)^2 : & \text{midpoint or trapezoid rule} \\ (\Delta t)^4 : & \text{Simpson's rule...} \end{cases} \quad (7.7)$$

The exponent is p , the scale length is Δt , and the symbol \approx hides a constant C and a factor $\|f^{(p)}(t)\|$. All these examples, which extend to other numerical approximations too, have the same form because they are based on the same idea.

That key idea is: **Watch the polynomials**. The rectangular rule is exact for $f(t) = \text{constant}$. The midpoint rule is exact for $f(t) = \text{linear}$. Simpson’s rule is exact for $f(t) = \text{cubic polynomial}$.

The exponent p is the degree of the first polynomial that gives an error. Locally, every smooth function looks like a polynomial. This is the essential idea of the Taylor series (and of calculus!). The lowest degree term $C(\Delta t)^p f^{(p)}(t)$ in the error will dominate. We determine the order of accuracy p by computing with polynomials.

An important point is developed at the end of this section. In digital signal processing, the input is not a function $f(t)$. It is a discrete vector $x(n)$. This vector can come from sampling $f(t)$. But if you use the sampled values as the coefficients a_{jk} that enter the filter bank, you are doing violence to the projection. The theory requires a preprocessing step, to transform sampled values to wavelet coefficients. Then postprocessing converts coefficients back to function values. These two steps can be approximated; they should not be ignored.

Note. The constant C is much smaller for splines than for other well-known scaling functions.

Compare these asymptotic constants for $p = 2, 3, \dots, 9$:

Splines	0.07	0.03	0.02	0.02	0.02	0.02	0.02	0.03
Daubechies	0.22	0.30	0.56	1.3	3.8	13.0	49.0	216.0

The leading error term is $C(\Delta t)^p f^{(p)}(t)/p!$ with these constants [SwPi,Unser]. This error is $f(t)$ minus its projection onto V_j as $\Delta t = 2^{-j} \rightarrow 0$. The growth in the Daubechies constants means roughly that approximation at scale j is only as close as splines at the coarser scale $j - 1$ (*half the resolution and a fraction of the work*). There is a price after all for the irregularity of $\phi(t) = D_{2p}(t)$, even though it can reproduce polynomials and achieve the exponent in $(\Delta t)^p$.

Determination of the Accuracy p

Before developing the theory, we compute the number p . The approximation is by translates of $\phi(t)$ and $w(t)$, which can be difficult and intricate functions. *The computation of p always goes back to the lowpass coefficients $h(k)$.* We can determine p from the $h(k)$, or from $H(\omega)$. What we cannot do, and wish to avoid, is the exact projection in continuous time based on $\phi(t)$.

The test for accuracy p is Condition A_p . We recognize it in the time domain, the frequency domain, and now also in the eigenvalue domain.

Theorem 7.1 *The accuracy is p if the lowpass filter coefficients $h(k)$ satisfy these three equivalent forms of Condition A_p :*

1. *p sum rules on the coefficients:* $\sum_{n=0}^N (-1)^n n^j h(n) = 0$ for $j = 0, 1, \dots, p - 1$.
 2. *p zeros at π :* $H(\omega) = \left(\frac{1+e^{-i\omega}}{2}\right)^p Q(\omega)$ and $H(z) = \left(\frac{1+z^{-1}}{2}\right)^p Q(z)$.
 3. *p eigenvalues $1, \frac{1}{2}, \dots, \left(\frac{1}{2}\right)^{p-1}$ of the matrix $M = (\downarrow 2)2H = \{2h(2i - j)\}$:*
- $$M\Phi^{(j)} = \left(\frac{1}{2}\right)^j \Phi^{(j)} \quad \text{for } j = 0, 1, \dots, p - 1. \quad (7.8)$$

Proof. The equivalence of 1 and 2 is straightforward. Substituting $\omega = \pi$ in the frequency response yields the alternating sum of coefficients:

$$\sum h(n)e^{-in\pi} = h(0) - h(1) + h(2) - \dots \quad (7.9)$$

Therefore $H = 0$ at $\omega = \pi$ when the first sum rule holds. For the next rule, when $j = 1$, the derivative of $h(n)e^{-in\omega}$ brings down a factor $-in$:

$$\sum h(n)(-in)e^{-in\pi} = -i(0h(0) - 1h(1) + 2h(2) - \dots). \quad (7.10)$$

Then $H' = 0$ at $\omega = \pi$ is equivalent to the second sum rule. A similar reasoning applies for higher order zeros and higher sum rules. A p -fold zero at π is equivalent to p sum rules. The notation has assumed an FIR filter but it could be IIR. We turn now to statement 3 about eigenvalues, and we start with examples.

Suppose $H(\omega)$ is the p th power of $\frac{1}{2}(1 + e^{-i\omega})$. It has p zeros at π . The other factor in this $H(\omega)$ is simply $Q = 1$. In that particular case *all* the eigenvalues are powers of $\frac{1}{2}$. These examples for $p = 2, 3, 4$ come from double shifts of 1, 2, 1 and 1, 3, 3, 1 and 1, 4, 6, 4, 1:

$$m_2 = \frac{1}{2} \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \quad m_3 = \frac{1}{4} \begin{bmatrix} 1 & 0 & 0 \\ 3 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix} \quad m_4 = \frac{1}{8} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 6 & 4 & 1 & 0 \\ 1 & 4 & 6 & 4 \\ 0 & 0 & 1 & 4 \end{bmatrix}$$

$$\lambda = 1, \frac{1}{2}$$

$$\lambda = 1, \frac{1}{2}, \frac{1}{4}$$

$$\lambda = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$$

The N by N matrix m has entries $2h(2i-j)$ for $i, j = 0, \dots, N-1$. We use the letter m , because this is a submatrix of M . The eigenvalues of m are also eigenvalues of the infinite matrix M , with the eigenvectors extended in both directions by zeros. The submatrix m is called $m(0)$ in Section 6.3. The other N by N submatrix is $m(1) = 2h(2i-j)$ for $i, j = 1, \dots, N$. This matrix $m(1)$ also has the special eigenvalues $(\frac{1}{2})^k$.

To establish that m has these eigenvalues, we increase the number of “zeros at π ” one step at a time. At each step, we watch the eigenvectors and eigenvalues. It will be very convenient to work in the z -domain, where each additional zero at $z = -1$ (which is $\omega = \pi$) comes from another factor $(\frac{1+z^{-1}}{2})$. The new eigenvalues of m , when $H(z)$ has that extra factor, are *half* the old eigenvalues. There is also one new eigenvalue at $\lambda = 1$. Theorem 7.2 describes the new eigenvalues, and its proof completes Theorem 7.1.

Theorem 7.2 When $H(z)$ is multiplied by $(\frac{1+z^{-1}}{2})$, m_{new} is one size larger:

- (a) The eigenvalues λ_{new} are $\frac{1}{2}\lambda_{\text{old}}$. There is an extra eigenvalue $\lambda_{\text{new}} = 1$.
- (b) The eigenvectors x_{new} are the differences of the eigenvectors x_{old} :

$$x_{\text{new}}(k) = x_{\text{old}}(k) - x_{\text{old}}(k-1) \text{ and } X_{\text{new}}(z) = (1 - z^{-1})X_{\text{old}}(z). \quad (7.11)$$

The new $\lambda = 1$ has left eigenvector $e_0 = [1 \ 1 \ \dots \ 1]$. The right eigenvector for $\lambda = 1$ gives the values $\phi_{\text{new}}(n)$ of the scaling function at the integers.

Proof. We will write the eigenvalue equation $m_{\text{old}}x_{\text{old}} = \lambda_{\text{old}}x_{\text{old}}$ in the z -domain. Then multiplication by $(\frac{1+z^{-1}}{2})$ gives the corresponding equation for m_{new} . Notice the aliasing term from $(\downarrow 2)$:

$$mx = \lambda x \text{ is } H(z)X(z) + H(-z)X(-z) = \lambda X(z^2). \quad (7.12)$$

This is for X_{old} . Now give $H(z)$ the extra $(\frac{1+z^{-1}}{2})$ and $X(z)$ the factor $(1 - z^{-1})$:

$$(\frac{1+z^{-1}}{2})H(z)(1-z^{-1})X(z) + (\frac{1-z^{-1}}{2})H(-z)(1+z^{-1})X(-z) = \frac{1}{2}\lambda(1-z^{-2})X(z^2). \quad (7.13)$$

This is the eigenvalue equation $\mathbf{m}_{\text{new}} \mathbf{x}_{\text{new}} = \lambda_{\text{new}} \mathbf{x}_{\text{new}}$. The eigenvalue is multiplied by $\frac{1}{2}$ and the eigenvectors obey (7.11). The whole proof is in $(\frac{1+z^{-1}}{2})(1-z) = \frac{1}{2}(1-z^{-2})$. In the examples $\mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4$ above, the eigenvectors for $\lambda = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ are in the columns of these matrices:

$$\begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{2} & 1 & -2 \\ \frac{1}{2} & -1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 1 \\ \frac{1}{6} & \frac{1}{2} & 1 & -3 \\ \frac{4}{6} & 0 & -2 & 3 \\ \frac{1}{6} & -\frac{1}{2} & 1 & -1 \end{bmatrix}$$

Take differences of the eigenvectors $0, 1, 0, \dots$ and $1, -1, 0, \dots$ in the 2×2 matrix. Those differences $0, 1, -1, \dots$ and $1, -2, 1, \dots$ are in the second matrix. They are eigenvectors of \mathbf{m}_3 for $\lambda = \frac{1}{2}$ and $\frac{1}{4}$. The new eigenvector for $\lambda = 1$ gives the values $\phi(1) = \phi(2) = \frac{1}{2}$ of the scaling function for $H(z) = (\frac{1+z^{-1}}{2})^3$. Section 6.3 explained how the dilation equation at the integers is exactly $\mathbf{m}\Phi = \Phi$ (with $\lambda = 1$). In this example $\phi(t)$ is a *spline* — linear, quadratic, and cubic for $p = 2, 3, 4$.

The 4×4 matrix \mathbf{m}_4 comes from $H(z) = (\frac{1+z^{-1}}{2})^4$. Its eigenvectors in the last three columns are differences of the columns in the 3×3 matrix. The first column, for $\lambda = 1$, holds the values $\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$ of the cubic spline at the integers. Section 7.4 develops the special properties of splines.

Also of importance are the *left eigenvectors*. For the special eigenvalues $1, \frac{1}{2}, \frac{1}{4}, \dots$ those eigenvectors are “discrete polynomials”. This means that the left eigenvector for $\lambda = 2^{-k}$ is a combination of the row vectors $\mathbf{e}_0, \mathbf{e}_1, \dots, \mathbf{e}_k$:

$$\mathbf{e}_0 = [1 \ 1 \ \cdots \ 1], \mathbf{e}_1 = [0 \ 1 \ \cdots \ N-1], \mathbf{e}_k = [0^k \ 1^k \ \cdots \ (N-1)^k].$$

The all-ones vector satisfies $\mathbf{e}_0 = \mathbf{m}\mathbf{e}_0$, as we have seen before. It is the left eigenvector \mathbf{y}_0 of \mathbf{m} corresponding to $\lambda = 1$:

$$[1 \ 1 \ 1 \ \cdots \ 1] \begin{bmatrix} 2\mathbf{h}(0) & & & \\ 2\mathbf{h}(2) & 2\mathbf{h}(1) & 2\mathbf{h}(0) & \\ & 2\mathbf{h}(3) & 2\mathbf{h}(2) & \dots \\ \vdots & \vdots & \vdots & \dots \end{bmatrix} = [1 \ 1 \ 1 \ \cdots \ 1]. \quad (7.14)$$

This just says that the even sum $\sum 2\mathbf{h}(2k)$ and the odd sum $\sum 2\mathbf{h}(2k+1)$ are equal to 1. That comes from adding and subtracting the first sum rule $\mathbf{h}(0) - \mathbf{h}(1) + \cdots = 0$ and the lowpass rule $\mathbf{h}(0) + \mathbf{h}(1) + \cdots = 1$.

When we multiply $H(z)$ by $(\frac{1+z^{-1}}{2})$, the other left eigenvectors of \mathbf{m}_{new} come from the left eigenvectors of \mathbf{m}_{old} . Where the right eigenvectors take differences of \mathbf{x}_{old} , the left eigenvectors take *sums*. Summing increases the polynomial degree by 1. In the z -domain, this sum corresponds to multiplication by $\frac{1}{1-z} = \frac{z^{-1}}{z^{-1}-1}$:

Theorem 7.3 *The left eigenvector for $\lambda = 1$ is always \mathbf{e}_0 . The other left eigenvectors in $\mathbf{y}_{\text{new}} \mathbf{m}_{\text{new}} = \lambda_{\text{new}} \mathbf{y}_{\text{new}}$ come from \mathbf{y}_{old} by summing and adding $C\mathbf{e}_0$:*

$$Y_{\text{new}}(z) = \frac{1}{1-z} Y_{\text{old}}(z) + C E_0(z). \quad (7.15)$$

Proof. Left eigenvectors of \mathbf{m} are right eigenvectors of \mathbf{m}^T . We transpose the operations of $(\downarrow 2)$ and multiplication by $2H(z)$. The transposes are $(\uparrow 2)$ and multiplication by $2H(z^{-1})$. The eigenvalue equation $\mathbf{m}^T \mathbf{y}^T = \lambda \mathbf{y}^T$ has z^2 because of $(\uparrow 2)$:

$$2H(z^{-1})Y_{\text{old}}(z^2) = \lambda Y_{\text{old}}(z). \quad (7.16)$$

Now give $H(z^{-1})$ the extra factor $(\frac{1+z}{2})$, and divide $Y(z)$ by $1-z$:

$$\left(\frac{1+z}{2}\right)H(z^{-1})\frac{1}{1-z^{-2}}Y(z^2) = \frac{1}{2}\lambda\frac{1}{1-z}Y(z). \quad (7.17)$$

This is $\mathbf{m}_{\text{new}}^T \mathbf{y}_{\text{new}}^T = \lambda_{\text{new}} \mathbf{y}_{\text{new}}^T$, which completes the proof. The eigenvalue is multiplied by $\frac{1}{2}$ (of course, since \mathbf{m}^T has the same eigenvalues as \mathbf{m}). The summation in the left eigenvectors (*row vectors*) can be seen in the same three examples \mathbf{m}_2 , \mathbf{m}_3 , \mathbf{m}_4 :

$$\begin{array}{lll} \lambda = 1 & \lambda = \frac{1}{2} & \lambda = \frac{1}{4} \\ \lambda = \frac{1}{2} & \lambda = \frac{1}{4} & \lambda = \frac{1}{8} \\ \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 1 \\ \frac{3}{2} & \frac{1}{2} & -\frac{1}{2} \\ 1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & -1 \\ \frac{11}{6} & \frac{2}{6} & -\frac{1}{6} & \frac{2}{6} \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

These left eigenvector matrices are the inverses of the previous right eigenvector matrices! The left eigenvectors are always biorthogonal to the right eigenvectors. The diagonalization by $S^{-1}MS$ has the right eigenvectors in the columns of S and the left eigenvectors in the rows of S^{-1} .

The left eigenvector for $\lambda = 1$ is always e_0 , the row of ones. The other left eigenvectors of \mathbf{m}_3 come from *minus* the sum of the \mathbf{m}_2 eigenvectors, *plus constant*:

$$\begin{aligned} \begin{bmatrix} 1 & 1 \end{bmatrix} &\rightarrow \begin{bmatrix} 0 & -1 & -2 \end{bmatrix} + \begin{bmatrix} \frac{3}{2} & \frac{3}{2} & \frac{3}{2} \end{bmatrix} = \begin{bmatrix} \frac{3}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \\ \begin{bmatrix} 1 & 0 \end{bmatrix} &\rightarrow \begin{bmatrix} 0 & -1 & -1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}. \end{aligned}$$

The vector after the arrow has a minus sign and delay. This is because $\frac{1}{1-z}$ equals $-z^{-1}$ times the summing operator $1 + z^{-1} + z^{-2} + \dots$.

Extension to Infinite Matrices

For the infinite matrix \mathbf{M} , these left eigenvectors are *not* extended by zeros. The finite eigenvector e_0 becomes an infinite all-ones eigenvector. The linear vector $e_1 = [0 \ 1 \ \dots \ N-1]$ becomes infinite too: $e_1(n) = n$ for all n . All “polynomial vectors” e_k extend as polynomials. Then the combination of the e ’s that gives the left eigenvector of \mathbf{m} also gives the extended left eigenvector of the infinite \mathbf{M} .

The eigenvector $\begin{bmatrix} \frac{3}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$ of \mathbf{m}_3 extends to a linear left eigenvector of \mathbf{M}_3 :

$$\left[\dots \frac{5}{2} \frac{3}{2} \frac{1}{2} -\frac{1}{2} \dots \right] \frac{1}{4} \begin{bmatrix} \dots 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 \\ 0 & 0 & 1 & 3 \dots \end{bmatrix} = \frac{1}{2} \left[\dots \frac{5}{2} \frac{3}{2} \frac{1}{2} -\frac{1}{2} \dots \right].$$

Being linear, it has $\lambda = \frac{1}{2}$. The other eigenvector $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ of M_3 extends to a “quadratic” left eigenvector (for $\lambda = \frac{1}{4}$) of M_3 :

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} - \frac{3}{2} \begin{bmatrix} 0 & 1 & 2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 & 1 & 4 \end{bmatrix} = \mathbf{e}_0 - \frac{3}{2}\mathbf{e}_1 + \frac{1}{2}\mathbf{e}_2.$$

The n th component of the eigenvector will be $1 - \frac{3}{2}n + \frac{1}{2}n^2$. Now we explain why these left eigenvectors are important in wavelet theory.

Theorem 7.4 *The left eigenvector in $y_k M = (\frac{1}{2})^k y_k$ gives the combination of scaling functions $\phi(t+n)$ that equals t^k :*

$$\sum y_k(n) \phi(t+n) = t^k \quad \text{for } k = 0, 1, \dots, p-1. \quad (7.18)$$

Thus the space V_0 spanned by $\{\phi(t+n)\}$ contains all polynomials of degree less than p .

Proof. We are to show that the inner product $G(t) = y_k \Phi_\infty(t) = \sum y_k(n) \phi(t+n)$ equals a multiple of t^k . Here y_k is a left eigenvector of M and $\Phi_\infty(t) = M \Phi_\infty(2t)$ solves the dilation equation. Put those two facts together:

$$y_k \Phi_\infty(t) = y_k M \Phi_\infty(2t) = \left(\frac{1}{2}\right)^k y_k \Phi_\infty(2t). \quad (7.19)$$

The left side is $G(t)$ and the right side is $(\frac{1}{2})^k G(2t)$. Since those are equal, $G(t)$ is a multiple of t^k . That is what we really wanted to prove. More details are in [HeStSt].

The all-ones eigenvector \mathbf{e}_0 says that $\sum \Phi(t+n) = 1$. This constant function assures at least $p=1$. Since the wavelet is orthogonal to 1, we have $\int w(t) dt = 0$ — the first vanishing moment. Hopefully there are more, and $p > 1$.

For M_2 , it is no surprise that 1 and t can be produced from translates of the hat function. More important is that 1 and t can be produced from the Daubechies scaling function $\phi(t) = D_4(t)$. This is a typical case in which $H(z)$ has an extra factor $\frac{1}{2} \left[1 + \sqrt{3} + (1 - \sqrt{3})z^{-1} \right]$ for double-shift orthogonality. The matrix M is 3 by 3 but only two eigenvalues 1 and $\frac{1}{2}$ are special (with their constant and linear left eigenvectors y_0 and y_1):

$$\frac{1}{4} \begin{bmatrix} 1 + \sqrt{3} & & \\ 3 - \sqrt{3} & 3 + \sqrt{3} & 1 + \sqrt{3} \\ & 1 - \sqrt{3} & 3 - \sqrt{3} \end{bmatrix} \text{ has } \begin{array}{ll} \lambda = 1 & y_0 = [1 \ 1 \ 1] \\ \lambda = \frac{1}{2} & y_1 = [3 - \sqrt{3} \ 1 - \sqrt{3} \ -1 - \sqrt{3}] / 2 \\ \lambda = \frac{1+\sqrt{3}}{4} & y_2 = [1 \ 0 \ 0] \end{array}$$

The sum $\sum \phi(t-n)$ is identically 1. The combination $\sum y_1(n) \phi(t+n)$ equals a multiple of t . Thus the Daubechies space V_0 contains 1 and t . Those are orthogonal to the wavelets in W_0 . This orthogonality $\int w(t) dt = 0$ and $\int t w(t) dt = 0$ says that the Daubechies wavelet has two vanishing moments.

Corollary (7.4) *When $H(\omega)$ has p zeros at π , the wavelets orthogonal to $\phi(t-n)$ have p vanishing moments. Those are the synthesis wavelets $\tilde{w}(t)$:*

$$\int_{-\infty}^{\infty} \tilde{w}(t) dt = 0, \quad \int_{-\infty}^{\infty} t \tilde{w}(t) dt = 0, \quad \dots, \quad \int_{-\infty}^{\infty} t^{p-1} \tilde{w}(t) dt = 0. \quad (7.20)$$

Reason: $1, \dots, t^{p-1}$ are combinations of $\phi(t-n)$. Orthogonality to these polynomials means p vanishing moments for the wavelets.

Remember that V_0 is orthogonal to \tilde{W}_0 rather than W_0 . Thus it is $\tilde{w}(t)$, not $w(t)$, that has p vanishing moments! For an orthogonal example like a Daubechies filter, $\tilde{W}_0 = W_0$ and $\tilde{w}(t) = w(t)$. In the biorthogonal case, the analysis wavelet $w(t)$ has \tilde{p} vanishing moments when the lowpass synthesis filter has \tilde{p} zeros at π .

Note. The polynomials $1, \dots, t^{p-1}$ are not actually in the subspace V_0 . They are indeed combinations of the translates of $\phi(t)$. But polynomials have infinite energy, $\int_{-\infty}^{\infty} (t^j)^2 dt = \infty$. If V_0 is defined as a subspace of L^2 , it cannot contain polynomials.

This point is only formal, not essential. The eigenvectors y have infinitely many nonzero components, which multiply all translates of $\phi(t)$. This maintains the polynomial for all time. Figure 7.1 shows how a finite combination maintains the polynomials 1 and t for finite time. This figure uses the Daubechies scaling function $\phi = D_4$ which has $p = 2$.

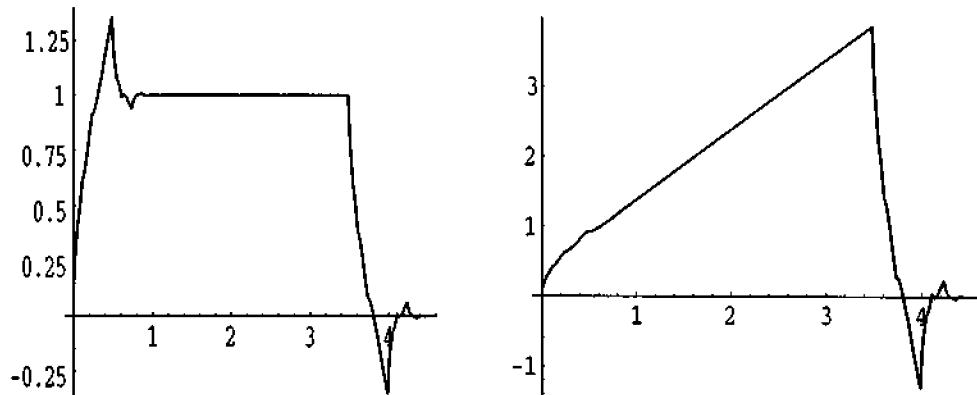


Figure 7.1: A combination of $D_4(t+n)$ can exactly reproduce 1 and t on any interval.

Approximation by Functions in V_j

In continuous time, vectors and matrices are replaced by functions of t . The lowpass filter coefficients $h(k)$ have done their part. By iteration they led to $\phi(t)$. Its translates can reproduce $1, \dots, t^{p-1}$ — and thus all polynomials of degree less than p . Its support interval is $[0, N]$ and we can estimate its smoothness. The theory in continuous time is now a job for “*harmonic analysis*”.

Harmonic analysis is the study of function spaces and transforms. It takes its name from the all-important Fourier example, which analyzes $f(t)$ into a sum of harmonics $e^{i\omega t}$. The key problem is to connect the properties of $f(t)$ with the properties of the transform (especially the size of the Fourier coefficients). Because $\sin \omega t$ and $\cos \omega t$ and $e^{i\omega t}$ have nonlocal support, cancellation is crucial. The size of the Fourier coefficients tells a lot, but not everything. Only in the energy norm do we have a perfect match:

The energy $\int |f(t)|^2 dt$ equals the energy $\frac{1}{2\pi} \int |\hat{f}(\omega)|^2 d\omega$.

In other L^p norms, and in other function spaces, the magnitude $|\widehat{f}(\omega)|$ does not completely decide whether $f(t)$ belongs to the space. We need the phase, which is more difficult. The theory using magnitudes is important. It can never be complete.

For a local wavelet basis, this situation is radically different. The magnitudes are enough! We can match the space of functions $f(t)$ to a space of wavelet coefficients b_{jk} . For $f(t)$ in L^p the coefficients are in the discrete space ℓ^p :

$$A \int |f(t)|^p dt \leq \sum_{j,k} |b_{jk}|^p \leq B \int |f(t)|^p dt. \quad (7.21)$$

In the language of harmonic analysis, wavelets are an *unconditional basis* when $p > 1$. The magnitudes $|b_{jk}|$ give sufficient information without phases. For L^2 , which is always the simplest and clearest, an unconditional basis is a *Riesz basis*:

$$A \int \left| \sum a_n \phi(t - n) \right|^2 dt \leq \sum |a_n|^2 \leq B \int \left| \sum a_n \phi(t - n) \right|^2 dt.$$

Then the translation invariance of the basis $\phi(t - n)$ yields the requirement (Section 6.5) on $A(\omega) = \sum a(k) e^{ik\omega}$ in the frequency domain:

$$0 < A \leq A(\omega) = \sum_{-\infty}^{\infty} |\widehat{\phi}(\omega + 2\pi k)|^2 \leq B \quad \text{for all } \omega. \quad (7.22)$$

Exact numbers A and B will come from the components $a(k)$ of the eigenvector $\mathbf{a} = T\mathbf{a}$. So will a similar inequality for the wavelet basis $w_{jk}(t)$ and the coefficients b_{jk} .

Approximation by Wavelets: Errors $f(t) - f_j(t)$

The number p of zeros at π tells how many basis functions are needed to approximate $f(t)$. The smoother the function, and the higher the order p , the faster the expansion coefficients go to zero and the fewer we need to keep.

We are touching here on the central problem of transform analysis — *to find a convenient basis that yields accurate approximation of the signal with few basis functions*. The best basis depends on the signal (of course). We have to choose a basis for a class of signals. For smooth signals, the Fourier basis is usually satisfactory. Perhaps the essential message of wavelet theory can be captured in a sentence:

For piecewise smooth functions, a wavelet basis is better.

These functions may have jumps. They may be smooth and suddenly rough. A wavelet basis, which is local, can separate those pieces. We keep more coefficients in the rough neighborhoods, by going to a smaller scale 2^{-j} . The mesh adapts to $f(t)$ in a way that Fourier finds difficult.

Here is the fundamental theorem on approximation by scaling functions and/or wavelets. The space of approximating functions is V_j , so the scale is $\Delta t = 2^{-j}$. This space is spanned by the scaling functions $\phi(2^j t - k)$, and it is also spanned by the wavelets $w_{jk}(t)$ at all scales below j . We may choose either basis, since multiresolution says that $V_j = V_0 \oplus W_0 \oplus \dots \oplus W_{j-1}$. The basis is not important at this point, because we are looking for the best function in the space.

Theorem 7.5 When $H(\omega)$ has p zeros at π , any $f(t)$ with p derivatives is approximated to order $(\Delta t)^p = 2^{-jp}$ by its projection $f_j(t)$ in V_j :

$$\|f(t) - f_j(t)\| \leq C(\Delta t)^p \|f^{(p)}(t)\|$$

$$\|f(t) - \sum_k a_k 2^{j/2} \phi(2^j t - k)\| \leq C 2^{-jp} \|f^{(p)}(t)\|. \quad (7.23)$$

This follows the expected pattern. For approximation by box functions or Haar wavelets, the error is of order Δt because $p = 1$. Try this piecewise constant approximation on the linear function $f(t) = t$. The closest constant on the interval $[0, \Delta t]$ is the halfway choice $a_1 = \frac{\Delta t}{2}$. The error $|f(t) - f_0(t)|$ on this interval is $|t - \frac{\Delta t}{2}|$. The maximum error is $\frac{\Delta t}{2}$ at $t = 0$. The L^2 error is

$$\left\|t - \frac{\Delta t}{2}\right\| = \left[\frac{1}{\Delta t} \int_0^{\Delta t} \left(t - \frac{\Delta t}{2}\right)^2 dt \right]^{1/2} = \frac{\Delta t}{2\sqrt{3}}. \quad (7.24)$$

The first derivative $f^{(1)}(t)$ on the right side of (7.23) is just 1. The constant is $C = \frac{1}{2}$ in the maximum norm. It is $C = 1/2\sqrt{3}$ if we use the L^2 norm. The important part is the power of Δt . But if a different $\phi(t)$ gives a much larger constant C , that is important too.

The main point is that the basis $\{\phi(t - k)\}$ can locally produce $1, \dots, t^{p-1}$. In each interval we can “essentially” match the start of the Taylor series. The error is the first Taylor series term we *cannot* match. This produces $(\Delta t)^p f^{(p)}(t)$ in the error bound. A detailed proof would lead far into approximation theory. The theorem was known earlier in the particular case of splines. Then it was extended to include *finite elements* [SF]. Those are local basis functions—normally piecewise polynomials. When the approximation theorem was first proved, nobody was thinking of wavelets! Who would use irregular functions to approximate smooth functions? It defies common sense, but wavelets come from iterations of simple filters. The computations are quick (if done recursively). The theorem applies because those irregular functions $\phi(t - k)$ can exactly reproduce polynomials.

The requirement on $H(\omega)$ is p zeros at π . It is interesting to note the corresponding requirements on $\widehat{\phi}(\omega)$. These are the so-called Strang-Fix conditions:

$$\widehat{\phi}(\omega) \text{ must have zeros of order } p \text{ at all frequencies } \omega = 2\pi n, n \neq 0. \quad (7.25)$$

The connection to zeros of $H(\omega)$ is through the infinite product $\widehat{\phi}(\omega) = \prod_1^\infty H(\omega/2^j)$. At frequency $\omega = 2\pi n$, we write $n = 2^j q$ with q odd. Then the $(j+1)$ st factor in the infinite product is $H(2\pi n/2^{j+1}) = H(q\pi)$. By periodicity this is $H(\pi)$. A p th order zero of H at $\omega = \pi$ yields a p th order zero of $\widehat{\phi}$ at $\omega = 2\pi n$. Thus the Strang-Fix condition on $\phi(t)$ becomes Condition A_p on $H(\omega)$.

The goal is always to find conditions on $H(\omega)$ that make $\widehat{\phi}(\omega)$ do what we want. For good approximation, we require zeros at π . This same condition improves the smoothness of $\phi(t)$ and stabilizes the lowpass filter under iteration. It is an open question *how many* zeros to ask for. Enough to stabilize the iterations, but not so many that we overconstrain the lowpass filter. Designers are often satisfied with two derivatives for $\phi(t)$, which occurs for $p \approx 4$. Other designers accept a smaller p .

Decay of the Wavelet Coefficients

The order p enters in another way, to improve compression. It allows the wavelet coefficients to do what Fourier coefficients do automatically: *they decrease rapidly for a smooth function.* For Fourier, $f(t)$ has to be smooth everywhere. One small jump, and the coefficients decrease no faster than $\frac{1}{k}$. For wavelets, the slow decrease will only apply around the jump. In smooth regions the coefficients drop off quickly. Multiresolution offers a wavelet basis in which the coefficients are directly linked to local properties of $f(t)$.

Theorem 7.6 *If $f(t)$ has p derivatives, its wavelet coefficients decay like 2^{-jp} :*

$$|b_{jk}| = \left| \int f(t) w_{jk}(t) dt \right| \leq C 2^{-jp} \|f^{(p)}(t)\|. \quad (7.26)$$

Proof. We plan to integrate by parts in $\int f(t) w_{jk}(t) dt$. That gives the derivative of f and the integral of w . (For biorthogonal wavelets replace w by \tilde{w} .) The first vanishing moment means that the integral of $w(t)$ from $-\infty$ to ∞ is zero. *The indefinite integral has compact support:*

$$I_1(t) = \int_{-\infty}^t w(u) du \text{ is nonzero only on } [0, N].$$

$I_1(t)$ is bounded, with finite energy. It will be a hat function, when $w(t)$ is Haar's up-down square wave. Integrate by parts in (7.26) to produce f' times a rescaled I_1 :

$$b_{jk} = 2^{-j} \int_{-\infty}^{\infty} f'(t) 2^{j/2} I_1(2^j t - k) dt = O(2^{-j}). \quad (7.27)$$

The factor 2^{-j} comes because we are integrating $w_{jk}(t)$ instead of $w(t)$:

$$\int_{-\infty}^t w_{jk}(t) dt = \int_{-\infty}^t 2^{j/2} w(2^j t - k) dt = 2^{-j} \int_{-\infty}^{2^j t - k} w(u) du.$$

Now repeat this step p times. Each integration by parts brings an extra derivative of $f(t)$ and an extra integral of $w_{jk}(t)$ — with its factor 2^{-j} . If successive integrals of $w(t)$ are $I_1(t)$ and $I_2(t)$ and finally $I_p(t)$, we end with

$$|b_{jk}| = \left| 2^{-jp} \int_{-\infty}^{\infty} f^{(p)}(t) 2^{j/2} I_p(2^j t - k) dt \right| \leq C 2^{-jp} \|f^{(p)}(t)\|.$$

Even when $f(t)$ has more derivatives, we cannot continue beyond p . The integral of $I_p(t)$ from $-\infty$ to ∞ is not zero! If it were, p integrations by parts would bring us back to $\int t^p w(t) dt = 0$ — but this p th moment of $w(t)$ does *not* vanish. $I_{p+1}(t)$ is a nonzero constant when t is large. Its energy is infinite and (7.26) breaks down for $p+1$. Look again at Haar wavelets with $p=1$. They have a square wave up and a square wave down, at scale 2^{-j} . The differences $f(t) - f(t - 2^{-j})$ are of order 2^{-j} . Integration gives a direct estimate $|b_{jk}| = O(2^{-j})$ in this particular case. The general method integrates by parts to get the derivative $f'(t)$ times the hat function I_1 times the key factor 2^{-j} .

Sample Values vs. Expansion Coefficients

Start with a function $x(t)$. Its samples $x(n)$ are often the inputs to the filter bank. Is this legal? **No.** It is a *wavelet crime*. Some can't imagine doing it, others can't imagine not doing it. Is this crime convenient? **Yes.** We may not know the whole function $x(t)$, it may not be a combination of $\phi(t - k)$, and computing the true coefficients in $\sum a(k)\phi(t - k)$ may take too long. But the crime cannot go unnoticed — we have to discuss it.

When the samples $x(n)$ are direct inputs to the filter bank (at unit scale $j = 0$), you effectively assume a particular continuous-time function. The pyramid algorithm (filter and downsample) acts on the numbers $x(n)$ as if they were expansion coefficients of its underlying function $x_s(t)$:

$$\text{Samples as coefficients: } x_s(t) = \sum x(n)\phi(t - n). \quad (7.28)$$

Does $x_s(t)$ have the correct sample values $x(n)$? This seems a minimum requirement. It holds when $\phi(k) = \delta(k)$. Then the only term in (7.28) at $t = n$ is the correct $x(n)$. A centered hat function has this property but most $\phi(t)$ do not. One possible solution is to adjust the coefficients in $\sum a(k)\phi(t - k)$ to produce the known samples $x(n)$:

$$\text{Determine } a_{int}(k) \text{ from } x(n) = \sum a_{int}(k)\phi(k - n). \quad (7.29)$$

This linear system has a constant-diagonal Toeplitz matrix. The n, k entry is $\phi(k - n)$. We are inverting an FIR filter that has the response $\sum \phi(k)e^{-ik\omega}$. Then the underlying function that interpolates the samples $x(n)$ is $x_{int}(t) = \sum a_{int}(k)\phi(t - k)$.

We believe that generally the samples $x(n)$ should be pre-filtered, before they enter the filter bank. Solving (7.29) puts the samples through an IIR filter. A different approach yields an FIR filter, by approximating the “correct” coefficients $a(k)$. Those are inner products of $x(t)$ with the analyzing function $\tilde{\phi}(t - k)$. The pre-filter replaces this integral by a sum:

$$\text{Replace } a(k) = \int x(t)\tilde{\phi}(t - k) dt \text{ by } a_q(k) = \sum x(n)\tilde{\phi}(n - k). \quad (7.30)$$

In the Daubechies examples, $\tilde{\phi} = \phi$ from orthogonality and the tilde disappears.

The pre-filter that gives $a_q(k)$ is normally FIR. Except for sinc wavelets and the duals to splines, our basis functions have compact support. The ideal function underlying this pre-filter is $x_q(t) = \sum a_q(k)\phi(t - k)$, a sensible choice.

Important: In continuous time, the synthesis $x(t) = \sum a(k)\phi(t - k)$ is consistent with the analysis $a(k) = \int x(t)\tilde{\phi}(t - k) dt$. In discrete time those are *not consistent*. When t changes to n and the integral changes to a sum, inverse operators do not become inverse matrices (unfortunately). But the discrete approximations are exactly correct for polynomials up to order p or \tilde{p} :

$$\sum_{-\infty}^{\infty} n^r \phi(n) = \int_{-\infty}^{\infty} t^r \phi(t) dt \text{ for } r < p. \quad (7.31)$$

The right side is the r th derivative of $\hat{\phi}(\omega) = \int \phi(t)e^{-i\omega t} dt$ at $\omega = 0$, times i^r . For the left side we use Poisson's summation formula (Chapter 2). This gives the same r th derivative at $\omega = 0$ and at points $\omega = 2\pi k$:

$$\sum_{-\infty}^{\infty} n^r \phi(n) = \sum_{-\infty}^{\infty} i^r \hat{\phi}^{(r)}(2\pi k).$$

But all terms on the right are zero except for $k = 0$, by the Strang-Fix condition (7.25) on the function $\phi(t)$. Thus the equality (7.31) holds for any $\phi(t)$ whose translates can reproduce polynomials to degree p . Similarly $a_q(k) = a(k)$, sum equals integral, when $x(t)$ is a low-degree polynomial.

Summary. We recommend that the samples $x(n)$ be converted to coefficients $a_q(k)$ by (7.30). Those enter the filter bank, not $x(n)$. The output $\hat{a}(k)$ can be post-filtered to recover sample values.

Other pre-filters are also reasonable. [Flandrin] proposed that the underlying $x(t)$ should be band-limited. The sampling theorem gives $x(t)$ as a sum of $\text{sinc}(t - n)x(n)$. Projecting this band-limited $x(t)$ onto V_0 gives $\sum a_{bl}(k)\phi(t - k)$, and those coefficients a_{bl} can enter the filter bank.

The samples $x(n)$ could be regarded as *averages* instead of point values. Then $x(t)$ is assumed piecewise constant, a combination of box functions $B(t - n)$. Projecting $x(t)$ onto V_0 gives $\sum a_{ave}(k)\phi(t - k)$. The filters can operate on $a_{ave}(k)$.

There is no unique answer to our question of how to process the sample values. We may choose $a_{int}(k)$ or $a_q(k)$ or $a_{bl}(k)$ or $a_{ave}(k)$. We should not send samples automatically through the filter bank.

Problem Set 7.1

1. Find the accuracy p from the sum rules for these filter coefficients:

- (a) $\mathbf{h} = \left(\begin{array}{ccc} \frac{1}{4}, & \frac{1}{2}, & \frac{1}{4} \end{array} \right)$
- (b) $\mathbf{h} = \frac{1}{16} \left(\begin{array}{ccccc} 1, & 4, & 6, & 4, & 1 \end{array} \right)$
- (c) $\mathbf{h} = \frac{1}{8} \left(\begin{array}{cccc} 1 - \sqrt{3}, & 3 - \sqrt{3}, & 3 + \sqrt{3}, & 1 + \sqrt{3} \end{array} \right)$ = Daubechies in reverse.
- (d) $\mathbf{h} = D_6$
- (e) $\mathbf{h} = \left(\begin{array}{ccc} \frac{1}{3}, & \frac{1}{3}, & \frac{1}{3} \end{array} \right)$

2. Factor the frequency response for filters (a) to (e) into $H(\omega) = (\frac{1+e^{-i\omega}}{2})^p R(\omega)$.

3. Find the eigenvalues of the N by N matrix \mathbf{m} for each of the filters (a) to (e). The number of taps is $N + 1$; cases (b) and (d) are less convenient by hand.

4. Which 5 by 5 matrix \mathbf{m}_5 comes from $H(z) = (\frac{1+z^{-1}}{2})^5$? Find the right eigenvectors for $\lambda = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$ from differences of the eigenvectors of \mathbf{m}_4 in the text. Find directly the eigenvector for $\lambda = 1$.

5. Verify that the left eigenvectors given in the text for \mathbf{m}_4 are sums of the left eigenvectors for \mathbf{m}_3 , plus a constant $[c \ c \ c \ c]$. Find the five left eigenvectors of \mathbf{m}_5 .

6. $\mathbf{M} = (\downarrow 2)2\mathbf{H}$ transforms $\widehat{f}(\omega)$ into $\widehat{\mathbf{M}f}(\omega) = H\left(\frac{\omega}{2}\right)\widehat{f}\left(\frac{\omega}{2}\right) + H\left(\frac{\omega}{2} + \pi\right)\widehat{f}\left(\frac{\omega}{2} + \pi\right)$. The first sum rule is $H(\pi) = 0$. If $\widehat{f}(0) = 0$, show that $\widehat{\mathbf{M}f}(0) = 0$, and explain what this means in the time domain.

7. The first two sum rules are $H(\pi) = 0$ and $H'(\pi) = 0$. Suppose that $\widehat{f}(0) = \widehat{f}'(0) = 0$. Show that $\widehat{\mathbf{M}f}(0) = (\widehat{\mathbf{M}f})'(0) = 0$.

8. If $w(t)$ has p vanishing moments, show that its Fourier transform has a p th order zero at $\omega = 0$. Then factoring $(i\omega)^p$ from \widehat{w} gives the transform \widehat{I}_p of the p -fold integral of $w(t)$.

9. Find the Fourier transform of the Haar wavelet and factor out $i\omega$ to obtain the transform of $I_1(t)$. Show that $I_1(t)$ is a hat function.

10. The left eigenvector $\begin{bmatrix} 1 & 0 \end{bmatrix}$ for M_2 extends to $\begin{bmatrix} \dots & 2 & 1 & 0 & -1 & \dots \end{bmatrix}$ for the infinite matrix M_2 . Write out $yM_2 = \frac{1}{2}y$ and circle the 2 by 2 subvector and submatrix in the middle. Verify that $\dots + 2\phi(t-1) + \phi(t) + 0 - \phi(t-1) - \dots$ equals t , when $\phi(t)$ is the hat function from this filter.
11. How would you compute $a_{ave}(k)$ so that $\sum a_{ave}(k)\phi(t-k)$ is the projection onto V_0 of the piecewise constant $x(t)$ with average $x(k)$ over the k th subinterval?

7.2 The Cascade Algorithm for the Dilation Equation

In the theory of wavelets, the two-scale dilation equation $\phi(t) = \sum 2h(k)\phi(2t-k)$ is central. Its solution is the scaling function, which leads to wavelets. The equation arises in the limit of the *cascade algorithm*

$$\phi^{(i+1)}(t) = \sum 2h(k)\phi^{(i)}(2t-k). \quad (7.32)$$

This is an iteration (with rescaling) of the lowpass filter. We find a simple proof of the necessary and sufficient condition on the $h(k)$ for $\phi^{(i)}(t)$ to converge in L^2 . The cascade normally begins from $\phi^{(0)}(t) = \text{box function}$. Problem 4 finds all other $\phi^{(0)}(t)$ that yield convergence to $\phi(t)$. Thus there are two conditions for convergence, one on the filter (this is the important one) and a condition on $\phi^{(0)}(t)$.

Our method is to compute the inner products $a^{(i)}(k) = \int \phi^{(i)}(t)\phi^{(i)}(t+k)dt$ at each step of the algorithm. The vectors $a^{(i+1)}$ and $a^{(i)}$ are connected by a *transition matrix* T formed from the $h(k)$. The cascade algorithm for $\phi(t)$ becomes the power method $a^{(i+1)} = Ta^{(i)}$ for the equation $a = Ta$. “Condition E” for convergence is that all eigenvalues of T satisfy $|\lambda| < 1$ except for a simple eigenvalue at $\lambda = 1$.

Recall that $\lambda = 1$ is an eigenvalue of $M = (\downarrow 2)2H$ by the *first sum rule*:

$$\sum_{\text{even } k} 2h(k) = \sum_{\text{odd } k} 2h(k) = 1.$$

These even k and odd k appear in separate columns of M . Each column adds to 1. Therefore $\lambda = 1$ is an eigenvalue, and the left eigenvector is $e = [1 \ 1 \ \dots \ 1]$. This is necessary for convergence, pointwise or L^2 , but far from sufficient. It means that $H(\omega) = \sum h(k)e^{-ik\omega}$ has a zero at $\omega = \pi$. Other things being equal, every zero at π gives a boost to convergence. This is a double zero in the nonnegative function $P(\omega) = |H(\omega)|^2$. The key to L^2 convergence is the matrix T associated with $|H(\omega)|^2$ in the same way that M is associated with $H(\omega)$:

$$T = (\downarrow 2)2HH^T.$$

When H and H^T are infinite Toeplitz matrices, M and T are also infinite. They are block Toeplitz matrices, with 1 by 2 blocks because of the operator $(\downarrow 2)$. T is illustrated in equation (7.38) below. The important action is in the finite matrix at the center. The central submatrix of T has order $2N - 1$:

$$T_{jk} = 2p(2j-k) \quad \text{if } |H(\omega)|^2 = \sum p(k)e^{-ik\omega}, \quad -N < j, k < N.$$

The columns of T are still even or odd, containing coefficients $p(2n)$ or else $p(2n+1)$. Those columns add to 1, since $|H(\omega)|^2$ has a zero at π . The all-ones vector e is still a left eigenvector for $\lambda = 1$:

$$eT = eMH^T = eH^T = e. \quad (7.33)$$

The crucial question is the significance of the *right eigenvector* in $T\mathbf{a} = \mathbf{a}$. It gives the inner products $\langle \phi(t), \phi(t+k) \rangle$. Convergence of the cascade algorithm in L^2 becomes convergence of the power method $\mathbf{a}^{(i+1)} = T\mathbf{a}^{(i)}$.

We include this convergence proof in the text because the argument is straightforward. Condition E on T also determines [Cohen-Daubechies] whether the translates $\phi(t+k)$ form a Riesz basis. [Eirola] and [Villemoes] use the same matrix T in a different way, to find the smoothness of $\phi(t)$ (next section).

We work with one-dimensional filters, but the analysis is the same in higher dimensions [Lawton-Lee-Shen]. In a sense our approach completes the convergence analysis of [CDM], by working with $|H(\omega)|^2$ — the autocorrelation of the filter (or mask). The key is to identify T and watch its eigenvalues.

Examples of Divergence and Weak Convergence

Example 7.1. The coefficients $h(k) = 1, \frac{1}{2}, -\frac{1}{2}$ have even sum = odd sum = $\frac{1}{2}$. The dilation equation is

$$\phi(t) = 2\phi(2t) + \phi(2t-1) - \phi(2t-2). \quad (7.34)$$

This looks innocent, but in a few steps the cascade algorithm is a disaster. The value at $t=0$ is doubled at every iteration. Each step gives $\phi^{(i+1)}(0) = 2\phi^{(i)}(0)$.

This blowup at a point does not by itself rule out finite energy. The function $f(t) = t^{-1/3}$ also blows up at $t=0$, but on the interval $[0, 1]$ its energy is $\int t^{-2/3} dt = 3$. Therefore this example is continued below, to prove that the energy in $\phi^{(i)}(t)$ does become infinite as $i \rightarrow \infty$. When the first coefficient in (7.34) is between $\frac{1}{2}(1-\sqrt{3})$ and $\frac{1}{2}(1+\sqrt{3})$, which allows the possibility of blowup at $t=0$, the cascade algorithm converges in L^2 .

Example 7.2. The coefficients $h(k) = \frac{1}{2}, 0, 0, \frac{1}{2}$ have even sum = odd sum = $\frac{1}{2}$ and also double-shift orthogonality. All the shifted vectors $0, 0, \frac{1}{2}, 0, 0, \frac{1}{2}, 0, \dots$ and $0, 0, 0, 0, \frac{1}{2}, 0, 0, \frac{1}{2}, \dots$ are orthogonal. The expected solution of the dilation equation is a *stretched box*:

$$\phi(t) = \phi(2t) + \phi(2t-3) \text{ leads to } \phi(t) = \begin{cases} \frac{1}{3} & 0 \leq t < 3 \\ 0 & \text{else.} \end{cases}$$

The box area is still one. Two half-boxes still fit into a whole box. But when the cascade algorithm starts with the unit box, it converges only *weakly* to $\phi(t)$.

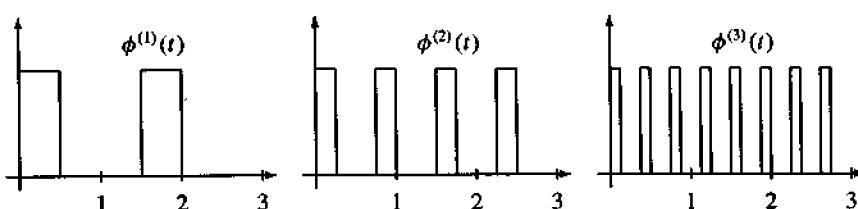


Figure 7.2: Three cascade steps $\phi^{(i+1)}(t) = \phi^{(i)}(2t) + \phi^{(i)}(2t-3)$.

The area between 0 and 3 is one-third filled, more and more densely. At no point in Figure 7.2 do the functions $\phi^{(i)}(t)$ actually equal $\frac{1}{3}$; they always equal 0 or 1. But the area above every interval Δt approaches $\frac{1}{3}\Delta t$. This is weak convergence to $\phi(t) = \frac{1}{3}$, stated for any smooth function $f(t)$:

$$\lim_{i \rightarrow \infty} \int_0^3 \phi^{(i)}(t) f(t) dt = \int_0^3 \frac{1}{3} f(t) dt. \quad (7.35)$$

The adjective “weak” means that integrals converge, even if functions themselves do not. Fast oscillations in $\phi^{(i)}(t)$ are averaged out by the integration.

Note that this shifted box $\phi(t)$ is *not orthogonal* to its translates. Furthermore, its energy $\int (\phi(t))^2 dt$ is $\frac{1}{3}$ instead of 1. Those facts seem surprising, because at every step $\phi^{(i)}(t)$ has unit energy and is orthogonal to all other $\phi^{(i)}(t+k)$. This illustrates the weakness of convergence—inner products of $\phi^{(i)}(t)$ with fixed $f(t)$ converge but inner products with $\phi^{(i)}(t+k)$ do not converge to $\int \phi(t)\phi(t+k) dt$.

Our inner product and energy formulas will involve T . This matrix always has $\lambda = 1$ as an eigenvalue, because of the zero at π . Explosive failure as in Example 1 is associated with an eigenvalue that has $|\lambda| > 1$. Weak convergence as in Example 2 is associated with a repeated $\lambda = 1$ and/or other eigenvalues with $|\lambda| = 1$ (with a full set of eigenvectors). Total success occurs when all other eigenvalues of T have $|\lambda| < 1$; this will be Condition E. The smaller those other eigenvalues, the smoother the function $\phi(t)$.

The Inner Product Formula

The energy in a real function $\phi(t)$ (its L^2 norm) is its inner product with itself. We also want its inner product with its translates $\phi(t+k)$. This inner product is $a(k)$:

$$a(k) = \int_{-\infty}^{\infty} \phi(t)\phi(t+k) dt, \quad -\infty < k < \infty. \quad (7.36)$$

Note that $a(k) = a(-k)$. The number $a(0)$ is the energy $\|\phi\|^2$. When $\phi(t)$ is zero outside the interval $0 \leq t < N$, the inner products $a(k)$ are all zero for $|k| \geq N$. The translated $\phi(t+N)$ does not overlap $\phi(t)$. Only the $2N - 1$ central components of a can be nonzero.

The cascade algorithm starts each step with $\phi^{(i)}(t)$. Suppose we know the inner products $a^{(i)}(k)$ between that function and its translates. Iteration produces a new function $\phi^{(i+1)}(t)$, and we want the new inner products $a^{(i+1)}(k)$. The new function is a combination $\sum 2h(k)\phi^{(i)}(2t-k)$, with t rescaled to $2t$. The new inner products depend on the old ones, and on the numbers $h(k)$. We now find the formula that the inner products obey.

Lemma. To find the vector $a^{(i+1)}$ of inner products, multiply $a^{(i)}$ by the Toeplitz matrix $2HH^T$ (which gives $4N - 1$ components) and downsample:

$$Ta^{(i)} = a^{(i+1)} = (\downarrow 2)2HH^T a^{(i)}. \quad (7.37)$$

Example 7.1 (continued) With $2h(k) = 2, 1, -1$, the numbers along the diagonals of the symmetric matrix $2HH^T$ are $\frac{1}{2}(2, 1, -1) * (-1, 1, 2) = \frac{1}{2}(-2, 1, 6, 1, -2)$. This comes from convolution and also from matrix multiplication: $2HH^T$ is

$$\frac{1}{2} \begin{bmatrix} \cdot & & & \\ -1 & 2 & & \\ & 1 & 2 & \\ & -1 & 1 & 2 \end{bmatrix} \begin{bmatrix} \cdot & 1 & -1 & & \\ & 2 & 1 & -1 & \\ & & 2 & 1 & \\ & & & 2 & \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \cdot, & \cdot, & \cdot, & \cdot, & \\ -2 & 1 & 6 & 1 & -2 \\ \cdot, & \cdot, & \cdot, & \cdot, & \cdot, \end{bmatrix}.$$

($\downarrow 2$) removes the odd-numbered rows to leave double shifts in T :

$$T = \frac{1}{2} \begin{bmatrix} \cdot & -2 & & & \\ \cdot & 6 & 1 & -2 & \\ & -2 & 1 & 6 & 1 & -2 \\ & & -2 & 1 & 6 & \cdot \\ & & & -2 & \cdot & \end{bmatrix}. \quad (7.38)$$

All columns of T add to 1! At the first cascade step, the box function $\phi^{(0)}(t)$ has inner products $a^{(0)} = (\dots, 0, 1, 0, \dots)$. The Lemma says that the inner products of $\phi^{(1)}(t)$ with its translates are

$$a^{(1)} = Ta^{(0)} = \frac{1}{2} \begin{bmatrix} \cdot \\ 0 \\ -2 \\ 6 \\ -2 \\ 0 \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot \\ 0 \\ -1 \\ 3 \\ -1 \\ 0 \\ \cdot \end{bmatrix}.$$

Since $N = 2$ for this filter H , all inner products are zero for $|k| \geq 2$. We only need the center submatrix of order $2N - 1 = 3$, and we iterate again:

$$a^{(2)} = \frac{1}{2} \begin{bmatrix} 1 & -2 & 0 \\ 1 & 6 & 1 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 3 \\ -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -7 \\ 16 \\ -7 \end{bmatrix}. \quad (7.39)$$

Clearly the numbers are growing. The cascade algorithm is diverging. T has an eigenvalue larger than 1. It is $\frac{5}{2}$, and this filter is a disaster in iteration.

Example 7.2 (continued) With coefficients $2h(k) = 1, 0, 0, 1$, the matrix $2HH^T$ has rows containing $\frac{1}{2}(1, 0, 0, 2, 0, 0, 1)$. Downsampling removes every other row to leave double shifts in T . Since $N = 3$ and $2N - 1 = 5$, we look only at T_5 :

$$T_5 = \frac{1}{2} \begin{bmatrix} 0 & 1 & & & \\ 2 & 0 & 0 & 1 & \\ 0 & 0 & 2 & 0 & 0 \\ & 1 & 0 & 0 & 2 \\ & & & 1 & 0 \end{bmatrix}.$$

Again all columns add to 1. Starting again with the box function $\phi^{(0)}(t)$, its inner products are $a^{(0)} = (0, 0, 1, 0, 0)$. Multiplying by T_5 produces this same vector $a^{(1)} = a^{(0)}$. Therefore $\phi^{(1)}(t)$ is also orthogonal to its translates.

That conclusion is no surprise. The $h(k)$ have double-shift orthogonality. The center column of T agrees with the identity matrix. At every step $a^{(i)} = \delta$.

Still there is weak trouble. The reason is that T_5 has a repeated eigenvalue at $\lambda = 1$. Its other eigenvalues are $-1, -\frac{1}{2}, \frac{1}{2}$. A second eigenvector for $\lambda = 1$ is $\frac{1}{9}(1, 2, 3, 2, 1)$. Those are the actual inner products of the stretched ϕ_{box} on the interval $0 \leq t < 3$. The inner products $a^{(i)} = \delta$ do not approach the inner products of ϕ_{box} . The cascade algorithm does not converge in L^2 to this stretched box.

Example 7.3. (Convergence to the hat function) With coefficients $2h(k) = \frac{1}{2}, 1, \frac{1}{2}$, the matrix $2HH^T$ has entries $\frac{1}{8}(1, 4, 6, 4, 1)$. Downsampling leaves $T_{2N-1} = T_3$:

$$T_3 = \frac{1}{8} \begin{bmatrix} 4 & 1 & 0 \\ 4 & 6 & 4 \\ 0 & 1 & 4 \end{bmatrix} \quad \text{has } \lambda = 1, \frac{1}{2}, \frac{1}{4}. \quad (7.40)$$

This example is successful but not orthogonal. Each multiplication of $a^{(0)} = (0, 1, 0)$ by T_3 gives the inner products at the next cascade step:

$$a^{(1)} = \frac{1}{8} \begin{bmatrix} 1 \\ 6 \\ 1 \end{bmatrix} \quad a^{(2)} = \frac{1}{64} \begin{bmatrix} 10 \\ 44 \\ 10 \end{bmatrix} \quad \dots \quad a^{(\infty)} = \frac{1}{6} \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix}. \quad (7.41)$$

We jumped to the limit $a^{(\infty)}$ because it is the eigenvector of T_3 for $\lambda = 1$.

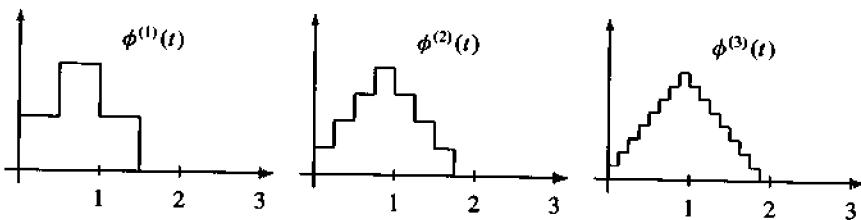


Figure 7.3: Three cascade steps $\phi^{(i+1)}(t) = \frac{1}{2}\phi^{(i)}(2t) + \phi^{(i)}(2t-1) + \frac{1}{2}\phi^{(i)}(2t-2)$.

The limit vector $a^{(\infty)} = a$ contains the inner products of the hat function with its translates. The $\phi^{(i)}(t)$ in Figure 7.3 are converging to $\phi^{(\infty)}(t) = \text{hat function}$, and the inner products $a^{(i)}(k)$ are converging to $a^{(\infty)}(k) = \frac{1}{6}, \frac{4}{6}, \frac{1}{6}$. We are seeing the **power method** in operation, for the functions and also for the vectors. The hat function is the steady-state fixed point of the operator in the cascade. The vector a is the steady-state eigenvector of T with $\lambda = 1$.

Proof of the Lemma To show that the inner products are $a^{(i+1)} = Ta^{(i)}$, it is very convenient to compute all $a^{(i+1)}(k)$ at once. Thus we work with vectors:

$$\Phi^{(i)}(t) = \begin{bmatrix} \cdot \\ \phi^{(i)}(t-1) \\ \phi^{(i)}(t) \\ \phi^{(i)}(t+1) \\ \cdot \end{bmatrix} \quad \text{and} \quad \Phi(t) = \begin{bmatrix} \cdot \\ \phi(t-1) \\ \phi(t) \\ \phi(t+1) \\ \cdot \end{bmatrix}. \quad (7.42)$$

The next vector in the cascade is $\Phi^{(i+1)}(t) = (\downarrow 2)2H\Phi^{(i)}(2t)$. The rescaling to $2t$ is accounted for by $(\downarrow 2)$. This is the way to a vector-based calculation:

$$\begin{aligned} a^{(i+1)} &= \int_{-\infty}^{\infty} \phi^{(i+1)}(t) \Phi^{(i+1)}(t) dt \\ &= \int_{-\infty}^{\infty} [2 \sum h(k)\phi^{(i)}(2t-k)] [(\downarrow 2)2H\Phi^{(i)}(2t)] dt. \end{aligned} \quad (7.43)$$

Bring the operator $(\downarrow 2)2H$ outside the integral. Change variables in the k th term to $u = 2t - k$. That k th term of the integral becomes (with $du = 2 dt$)

$$\int_{-\infty}^{\infty} h(k)\phi^{(i)}(u)\Phi^{(i)}(u+k)du = h(k)S^{-k}\alpha^{(i)}. \quad (7.44)$$

The k -step shift S^{-k} allowed us to write $\Phi^{(i)}(u+k)$ as $S^{-k}\Phi^{(i)}(u)$. Then the integration with respect to u produced $\alpha^{(i)}$. Now sum equation (7.44) on k to reach the matrix $\sum h(k)S^{-k}$, which is H^T as the Lemma requires:

$$\alpha^{(i+1)} = (\downarrow 2)2HH^T\alpha^{(i)} = T\alpha^{(i)}. \quad (7.45)$$

Corollary *The inner products $\alpha(k)$ of the scaling function $\phi(t)$ with $\phi(t+k)$ are in the eigenvector of T corresponding to $\lambda = 1$:*

$$\alpha = T\alpha. \quad (7.46)$$

This assumes that the scaling function exists in L^2 , which we will prove. To reach $\alpha = T\alpha$, repeat the calculation above without the superscripts.

The power method $\alpha^{(i)} = T^i\alpha^{(0)}$ converges when T has a non-repeated eigenvalue $\lambda = 1$ and all other eigenvalues have $|\lambda| < 1$. This “Condition E” gives $\alpha^{(i)} \rightarrow \alpha$. (Note! All vectors are normalized by $\sum \alpha(k) = 1$.) Convergence for the functions $\phi^{(i)}(t)$ is still to prove, but convergence for their inner products $\alpha^{(i)}$ is easier — just linear algebra.

Theorem 7.7 *The infinite matrix $T = (\downarrow 2)2HH^T$ and its submatrix T_{2N-1} always have $\lambda = 1$ as an eigenvalue. The power iteration $\alpha^{(i+1)} = T\alpha^{(i)}$ converges to the eigenvector $\alpha = T\alpha$ if and only if T_{2N-1} satisfies*

Condition E: T_{2N-1} has all $|\lambda| < 1$ except for a simple eigenvalue $\lambda = 1$.

Proof. Suppose the starting $\alpha^{(0)}$ is expanded as a combination $\alpha + c_2v_2 + c_3v_3 + \dots$ of eigenvectors of T_{2N-1} . Every time we multiply by that matrix, each v_j is multiplied by the corresponding λ_j . Since $|\lambda_j| < 1$ by Condition E, those components get smaller. In the limit as $i \rightarrow \infty$, the powers $T^i\alpha^{(0)}$ converge to the eigenvector α — whose coefficient stays at 1 (because it has $\lambda = 1$).

This proof only works if T_{2N-1} has a full set of eigenvectors, to expand $\alpha^{(0)}$. To cover all cases we use the Jordan form of T_{2N-1} . It has $\lambda = 1$ alone in a 1×1 block. All other blocks have $|\lambda| < 1$ and their powers approach zero.

Convergence of the Cascade Algorithm in L^2

The convergence proof will be easy if we know that the dilation equation $\phi(t) = \sum 2h(k)\phi(2t-k)$ has a finite energy solution. We now prove that $\phi^{(i)}(t)$ converges to this scaling function $\phi(t)$. Properly speaking, we must also show the existence of $\phi(t)$ itself. This existence step is logically first but it will come later for simplicity.

Theorem 7.8 *Assume that $\phi(t)$ is in L^2 . The cascade sequence $\phi^{(i)}(t)$ converges to $\phi(t)$ if and only if T satisfies Condition E. Then*

$$\|\phi^{(i)} - \phi\|^2 = \|\phi^{(i)}\|^2 - 2\langle \phi^{(i)}, \phi \rangle + \|\phi\|^2 = \alpha^{(i)}(0) - 2\alpha^{(i)}(0) + \alpha(0) \quad (7.47)$$

converges to $\alpha(0) - 2\alpha(0) + \alpha(0) = 0$.

Proof. The numbers $a^{(i)}(0)$ and $a(0)$ are the energies $\|\phi^{(i)}(t)\|^2$ and $\|\phi(t)\|^2$:

$$\int \phi^{(i)}(t)\phi^{(i)}(t+0) dt = a^{(i)}(0) \quad \text{and} \quad \int \phi(t)\phi(t+0) dt = a(0).$$

We know already that $a^{(i)}$ converges to a . This was the preceding theorem. Equation (7.47) also contains the inner product of $\phi^{(i)}(t)$ with $\phi(t)$. This is the zeroth component $b^{(i)}(0)$ of a new vector of inner products $b^{(i)}(k) = \int \phi^{(i)}(t)\phi(t+k) dt$.

Our main calculation found each vector $a^{(i+1)}$ from the previous $a^{(i)}$. The rule was to multiply by T . Condition E gave convergence to a . In the same way, we now show that $b^{(i+1)} = Tb^{(i)}$. Then the vectors $b^{(i)}$ converge (this is the power method again) to the same eigenvector a . Therefore $-2b^{(i)}(0)$ in (7.47) converges to $-2a(0)$, which completes the proof that $\|\phi^{(i)} - \phi\|^2$ converges to zero.

For the new calculation $b^{(i+1)} = Tb^{(i)}$, it is again convenient to work with vectors:

$$b^{(i+1)} = \int_{-\infty}^{\infty} \phi^{(i+1)}(t) \begin{bmatrix} \cdot \\ \phi(t-1) \\ \phi(t) \\ \phi(t+1) \\ \cdot \end{bmatrix} dt = \int_{-\infty}^{\infty} \phi^{(i+1)}(t)\Phi(t) dt.$$

Substitute the cascade formula for $\phi^{(i+1)}(t)$ and the dilation equation for $\Phi(t)$:

$$b^{(i+1)} = \int_{-\infty}^{\infty} [2 \sum_k h(k)\phi^{(i)}(2t-k)] [(\downarrow 2)2H\Phi(2t)] dt. \quad (7.48)$$

This matches equation (7.43) when $\Phi^{(i)}$ is replaced by Φ . Change variables in the k th term to $u = 2t - k$, and that term matches (7.44)—with a replaced by b . Then sum on k to match equation (7.45). These same steps give the new answer, with a changed to b :

$$b^{(i+1)} = (\downarrow 2)2HH^T b^{(i)} = Tb^{(i)}. \quad (7.49)$$

The normalization $\sum b^{(0)}(k) = 1$ is still true, because $\int \phi^{(0)}(t) \sum \phi(t+k) dt = \int \phi^{(0)} dt = 1$. So the power method starting from $b^{(0)}$ converges to the same vector a (not b !). By equation (7.47) the sequence $\phi^{(i)}(t)$ converges to $\phi(t)$.

The converse is also straightforward [Strang3]. Convergence of the functions $\phi^{(i)}$ to ϕ implies convergence of their inner products $a^{(i)}$ to a . Thus the power method always goes to the same limiting vector a . Condition E must hold.

Existence of the Scaling Function

This is not a book about proofs. But the dilation equation is so fundamental that we must be sure it has a solution. Formally the infinite product $\prod H(\omega/2^j)$ yields the Fourier transform $\widehat{\phi}(\omega)$. The real question is when this product converges strongly to a finite energy solution, and the answer to that question is Condition E.

We base existence of $\phi(t)$ on our calculation of inner products, which is the key to this section. The space L^2 is *complete*, so when we prove that the energies $\|\phi^{(m)} - \phi^{(n)}\|^2$ approach zero, there is guaranteed to exist a limit function $\phi(t)$ in L^2 .

Theorem 7.9 If Condition E holds then $\|\phi^{(m)} - \phi^{(n)}\|^2$ approaches zero as $m, n \rightarrow \infty$. Therefore the sequence $\phi^{(0)}(t), \phi^{(1)}(t), \dots$ converges to a limit $\phi(t)$ in L^2 .

Proof. The energy $\|\phi^{(m)} - \phi^{(n)}\|^2$ is $\|\phi^{(m)}\|^2 - 2\langle \phi^{(m)}, \phi^{(n)} \rangle + \|\phi^{(n)}\|^2$. The first and third terms are $\mathbf{a}^{(m)}(0)$ and $\mathbf{a}^{(n)}(0)$, both approaching the limit $\mathbf{a}(0)$. We must show that the inner product $\langle \phi^{(m)}, \phi^{(n)} \rangle$ also approaches $\mathbf{a}(0)$.

Suppose $m = i + n$ with fixed $i > 0$. The same inner product calculation, following the pattern (7.43)–(7.44)–(7.45) and multiplying by \mathbf{T} at each step, yields

$$\langle \phi^{(m)}, \phi^{(n)} \rangle = \mathbf{T}^n \langle \phi^{(i)}, \phi^{(0)} \rangle = \mathbf{T}^n \mathbf{c}^{(i)}(0). \quad (7.50)$$

The vector $\mathbf{c}^{(i)}$ contains the inner products $c^{(i)}(k) = \int \phi^{(i)}(t)\phi^{(0)}(t+k) dt$. For each fixed i , the power method $\mathbf{T}^n \mathbf{c}^{(i)}$ approaches \mathbf{a} . The limit of (7.50) is $\mathbf{a}(0)$ as desired. But the difficulty (since $i = m-n$ is arbitrary) is that this convergence must hold uniformly for all starting vectors $\mathbf{c}^{(i)}$. We know two facts about the components $c^{(i)}(k)$. They are uniformly bounded and they add to 1:

$$|c^{(i)}(k)| \leq \|\phi^{(i)}\| \|\phi^{(0)}\| \leq \|\mathbf{a}^{(i)}\| \leq C \quad (7.51)$$

$$\sum_k c^{(i)}(k) = \int_{-\infty}^{\infty} \phi^{(i)}(t) \sum_k \phi^{(0)}(t+k) dt = \int_{-\infty}^{\infty} \phi^{(i)}(t) dt = 1. \quad (7.52)$$

Condition E and the Jordan form J of \mathbf{T} give uniform convergence $\mathbf{T}^n \mathbf{c} \rightarrow \mathbf{a}$:

$$\mathbf{T}^n \mathbf{c} = \mathbf{S} \mathbf{J}^n \mathbf{S}^{-1} \mathbf{c} = \begin{bmatrix} \mathbf{a} & \cdots & \end{bmatrix} \begin{bmatrix} 1 & & \\ & \mathbf{B}^n & \\ & & \cdots \end{bmatrix} \begin{bmatrix} \mathbf{e} & & \end{bmatrix} \begin{bmatrix} \mathbf{c} & & \end{bmatrix}. \quad (7.53)$$

The left eigenvector $\mathbf{e} = [1 \ 1 \ \dots \ 1]$ is in row 1, and \mathbf{a} in column 1 is the right eigenvector. These eigenvector matrices \mathbf{S} and \mathbf{S}^{-1} are fixed, and the block \mathbf{B} has eigenvalues $|\lambda| < 1$. Therefore $\mathbf{B}^n \rightarrow \mathbf{0}$ and we have uniform convergence to $\mathbf{a}\mathbf{c} = \mathbf{a} \sum c(k) = \mathbf{a}$. Equation (7.50) approaches $\mathbf{a}(0)$ as m and n get large, completing the proof that $\|\phi^{(m)} - \phi^{(n)}\|^2 \rightarrow 0$.

Remark 1 The convergence of the cascade algorithm can be interpreted in the frequency domain, which is illuminating. The convergence of $\phi^{(i)}(t)$ to $\phi(t)$ becomes convergence of the infinite product to $\widehat{\phi}(\omega)$:

$$\widehat{\phi}^{(i)}(\omega) = \left[\prod_{j=1}^i H(\omega/2^j) \right] \widehat{\phi}^{(0)}(\omega/2^i) \text{ converges in } L^2 \text{ to } \widehat{\phi}(\omega) = \prod_{j=1}^{\infty} H(\omega/2^j).$$

Remark 2 For filters with double-shift orthogonality, there is no danger that \mathbf{T} has an eigenvalue with $|\lambda| > 1$. The norm of \mathbf{T} is $\sup(|H(\omega)|^2 + |H(\omega + \pi)|^2)$ and this is 1. Condition E reduces to the Cohen-Lawton condition that $\lambda = 1$ is a simple eigenvalue of \mathbf{T} . Then $\{\phi(t+k)\}$ is an orthonormal basis and $\mathbf{a} = \delta$.

Condition E holds in this orthogonal case if $H(\omega) \neq 0$ for $|\omega| \leq \frac{\pi}{3}$ [Mallat1, JiaWang].

Remark 3 The same method (7.43)–(7.44)–(7.45) that gives inner products of scaling functions also gives inner products with wavelets. The highpass operator \mathbf{H}_1 replaces the lowpass \mathbf{H} in the appropriate places:

$$\langle \phi(t), w(t+k) \rangle = (\downarrow 2) 2 \mathbf{H} \mathbf{H}_1^T \mathbf{a} \quad (7.54)$$

$$(w(t), w(t+k)) = (\downarrow 2) 2H_1 H_1^T a \quad (7.55)$$

These are derived in Section 11.6. We find inner product formulas for any functions that satisfy two-scale equations.

Remark 4 For “multifilters” the coefficients $\mathbf{h}(k)$ are $r \times r$ matrices. The dilation equation determines a vector of r scaling functions. The inner product $\int \phi(t)\phi^T(t+k) dt$ is an $r \times r$ matrix $\mathbf{a}(k)$. The equation $T\mathbf{a} = \mathbf{a}$ for a vector of matrices now involves a matrix convolution:

$$T\mathbf{a} = (\downarrow 2) 2\mathbf{h} * \mathbf{a} * \mathbf{h}^T. \quad (7.56)$$

In the frequency domain this *matrix transition operator* T becomes

$$T\mathbf{A}(\omega) = H\left(\frac{\omega}{2}\right) A\left(\frac{\omega}{2}\right) H^*(\frac{\omega}{2}) + H\left(\frac{\omega}{2} + \pi\right) A\left(\frac{\omega}{2} + \pi\right) H^*(\frac{\omega}{2} + \pi). \quad (7.57)$$

The theory develops on the same lines [Cohen-Daubechies-Plonka] to give the existence in L^2 , the smoothness, the approximation properties, and the stability of the basis $\{\phi_i(t+k)\}$. The eigenvalues of T are still in control.

Problem Set 7.2

1. If $\mathbf{h}(k)$ has double-shift orthogonality (Condition O), show that the central column of T is δ . This is an eigenvector of T for $\lambda = 1$.
2. Construct the finite matrices T for $\mathbf{h} = \frac{1}{3}(1, 1, 1)$ and $\mathbf{h} = (c, \frac{1}{2}, \frac{1}{2}, -c)$.
3. Draw the output from one Haar cascade step $\phi^{(1)}(t) = \phi^{(0)}(2t) + \phi^{(0)}(2t-1)$ with
 - (a) $\phi^{(0)}(t) = \text{unit box on the interval } [1, 2]$
 - (b) $\phi^{(0)}(t) = \text{hat function on the interval } [0, 2]$
 - (c) $\phi^{(0)}(t) = \text{hat function on the interval } [0, 1]$.

Two of those $\phi^{(0)}(t)$ lead to convergence. Which one doesn't?

4. Prove that $P^{(1)}(t) = \sum \phi^{(1)}(t-n)$ equals $P^{(0)}(2t) = \sum \phi^{(0)}(2t-n)$ for any starting function $\phi^{(0)}(t)$ and any coefficients $\mathbf{h}(k)$ with a zero at π : even sum = odd sum = $\frac{1}{2}$. It follows that $P^{(i)}(t) = P^{(0)}(2^i t)$ will oscillate faster and faster. There is no convergence unless $P^{(0)}(t) = \sum \phi^{(0)}(t-n)$ is identically one. This is the condition on $\phi^{(0)}(t)$ in the cascade algorithm.
5. Find the eigenvalues of T in Problem 2 (depending on c).
6. Show that the filter $\mathbf{h} = [-1 \ 3 \ 3 \ -1]/4$ does not satisfy Condition E, and T has $\lambda = 2.1712$. This bad \mathbf{h} is dual to the good spline filter $\mathbf{f} = [1 \ 3 \ 3 \ 1]/8$; their product $\mathbf{h} * \mathbf{f}$ is Daubechies maxflat halfband.

7.3 Smoothness of Scaling Functions and Wavelets

The previous section established Condition E for the convergence of the cascade algorithm. The eigenvalues of T are required to be less than 1 (except the simple eigenvalue $\lambda = 1$). The limit function $\phi(t)$ is then in L^2 — which assures some minimal smoothness, but not much. If the

eigenvalues of T are less than 4^{-s} , apart from the special eigenvalues that are powers of $\frac{1}{2}$, we now show that $\phi(t)$ and $w(t)$ have s derivatives.

This conclusion is true whether s is an integer or not. The proof for integers s is very direct, so we include it. The proof for noninteger s needs more space and effort, so we refer for example to [Villemoes]. These are derivatives in the L^2 sense because we work with the matrix T . In the frequency domain, each derivative is a multiplication by $i\omega$. Therefore $\phi(t)$ in L^2 has s derivatives in L^2 when

$$\|\phi^{(s)}(t)\|^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} |\omega|^{2s} |\widehat{\phi}(\omega)|^2 d\omega \text{ is finite.}$$

This definition allows s to be a fraction (or negative) with no difficulty. Since $w(t)$ is a combination of $\phi(2t - k)$, we only need to study the smoothness of $\phi(t)$.

The basic idea is simple. Each new factor $\left(\frac{1+z^{-1}}{2}\right)$ in $H(z)$ has four effects:

1. All eigenvalues of T are divided by 4.
2. The old $\phi(t)$ is convolved with the box function.
3. The old $\widehat{\phi}(\omega)$ is multiplied by $(1 - e^{-i\omega}) / i\omega$.
4. The new $\phi(t)$ has one more derivative than the old $\phi(t)$.

When we check these facts, our desired result is proved. You might think that the final $\phi(t)$ has the full p derivatives, because $H(z)$ has p factors of $\frac{1+z^{-1}}{2}$. But some of those factors are needed to get the eigenvalues of T below 1 (always excluding the special eigenvalues $1, \frac{1}{2}, \dots$). If s (integer) is less than the number s_{\max} below, there will be s factors still left after this Condition E is met. Then $\phi(t)$ has s derivatives. The non-special eigenvalues are below 4^{-s} .

The word “regularity” has been applied to s and also to p . Those are different numbers (we will prove $s < p$). So we avoid that word, and refer to smoothness s and accuracy p . We state the conclusions for any s ; our proof was for $s = \text{integer}$.

Theorem 7.10 Each new factor $\frac{1+z^{-1}}{2}$ has the effects 1, 2, 3, 4. Then $\phi(t)$ and $w(t)$ have s derivatives in L^2 when the non-special eigenvalues of T have $|\lambda| < 4^{-s}$. The supremum s_{\max} , when $\phi(t)$ comes from $H(z) = \left(\frac{1+z^{-1}}{2}\right)^p Q(z)$, is

$$s_{\max} = p - \log_4 |\lambda_{\max}(T_Q)| \quad \text{with} \quad T_Q = (\downarrow 2)2QQ^T. \quad (7.58)$$

A short MATLAB code will construct T or T_Q from h and find its eigenvalues. In practice, we exclude the special $\lambda = 1, \dots, (\frac{1}{2})^{2p-1}$ directly from the eigenvalues of T , and then $s_{\max} = -\log_4(|\lambda_{\max}(T)|)$.

Actually 1, 2, 3 are already proved. The effect on the eigenvalues of T was established in Section 7.2. Each eigenvalue is divided by 2 twice, because T comes from $H(z)H(z^{-1})$. So all eigenvalues of T_{old} are divided by 4. Then T_{new} also has the special eigenvalues 1 and $\frac{1}{2}$. The extra $\frac{1+z^{-1}}{2}$ in $H(z)$ changes the infinite product to $\widehat{\phi}_{\text{new}}(\omega)$:

$$\widehat{\phi}_{\text{new}}(\omega) = \prod_{j=1}^{\infty} \left(\frac{1}{2} + \frac{1}{2}e^{-i\omega/2^j} \right) \widehat{\phi}_{\text{old}}(\omega) = \left(\frac{1 - e^{-i\omega}}{i\omega} \right) \widehat{\phi}_{\text{old}}(\omega). \quad (7.59)$$

The extra “Haar factors” multiply to give the “box factor.” This is the infinite product in Section 6.4 that yields the box function $B(t)$. In the time domain $\phi_{\text{new}}(t)$ is the box convolved with $\phi_{\text{old}}(t)$. This adds one more derivative for ϕ_{new} .

Lemma 7.1 *The convolution $\phi_{\text{new}}(t) = B(t) * \phi_{\text{old}}(t)$ has $s + 1$ derivatives if and only if $\phi_{\text{old}}(t)$ has s derivatives.*

Proof. In frequency we are multiplying $\widehat{\phi}_{\text{old}}(\omega)$ by $(1 - e^{-i\omega})/i\omega$. This has magnitude at most $2/|\omega|$. Therefore $\widehat{\phi}_{\text{new}}$ decreases at least one order faster than $\widehat{\phi}_{\text{old}}$:

$$\int_{-\infty}^{\infty} |\omega|^{2(s+1)} |\widehat{\phi}_{\text{new}}(\omega)|^2 d\omega \leq 4 \int_{-\infty}^{\infty} |\omega|^{2s} |\widehat{\phi}_{\text{old}}(\omega)|^2 d\omega.$$

The factor 4 comes from $(2/|\omega|)^2$ times $|\omega|^2$. The last integral is finite when ϕ_{old} has s derivatives. So the first integral is finite and ϕ_{new} has $s + 1$ derivatives. In the time domain, the derivative of the convolution $\phi_{\text{new}}(t) = B(t) * \phi_{\text{old}}(t)$ is a *difference*:

$$\phi'_{\text{new}}(t) = \frac{d}{dt} \int_0^1 \phi_{\text{old}}(t-s) ds = \int_0^1 \phi'_{\text{old}}(t-s) ds = \phi_{\text{old}}(t) - \phi_{\text{old}}(t-1). \quad (7.60)$$

Again ϕ_{new} has one more derivative than ϕ_{old} . This is true in the pointwise sense as well as the L^2 sense. The smoothness increases by one from the extra zero at π in the filter.

For completeness we prove the converse, following [Villemoes]. If ϕ_{new} has $s + 1$ derivatives then certainly ϕ'_{new} has s derivatives. From (7.60) this means that $\phi_{\text{old}}(t) - \phi_{\text{old}}(t-1)$ has s derivatives. Use this fact N times:

$$\phi_{\text{old}}(t) - \phi_{\text{old}}(t-N) = [\phi_{\text{old}}(t) - \phi_{\text{old}}(t-1)] + [\phi_{\text{old}}(t-1) - \phi_{\text{old}}(t-2)] + \dots$$

Each difference on the right has s derivatives. So does the difference on the left. But $\phi(t)$ does not overlap $\phi(t-N)$, so $\phi_{\text{old}}(t)$ by itself must have s derivatives.

Example 7.4. For any $s < \frac{1}{2}$, the box function has s derivatives in L^2 .

Reason: $|(1 - e^{-i\omega})/i\omega|$ is below and often near $\frac{2}{|\omega|}$. The integral $\int |\omega|^{2s} \left| \frac{2}{\omega} \right|^2 d\omega$ is finite for $s < \frac{1}{2}$ but infinite for $s = \frac{1}{2}$. Therefore the value $s_{\max} = \frac{1}{2}$ is not actually achieved, which is typical. Normally $\phi(t)$ has s derivatives for all $s < s_{\max}$.

We can check formula (7.58). The box filter $H(z) = \frac{1+z^{-1}}{2}$ has $Q(z) \equiv 1$. Then $Q = I$ and $T_Q = (\downarrow 2)2I$ has $\lambda_{\max} = 2$. The logarithm of 2 to base 4 is $\frac{1}{2}$. Formula (7.58) correctly gives $s_{\max} = 1 - \frac{1}{2} = \frac{1}{2}$.

The splines of degree $p-1$, which come from $p-1$ additional convolutions of the box, have $s_{\max} = p - \frac{1}{2}$. This is the largest possible s_{\max} ! With p zeros at π , $\phi(t)$ cannot have more than $p - \frac{1}{2}$ derivatives in L^2 . We will stay with integers to prove that $s \leq p-1$, after noting how the smoothness of splines drops by $\frac{1}{2}$ when we change from L^2 to pointwise.

Pointwise, the box function has zero smoothness. The hat function has one derivative (only one-sided, because the slope jumps). The spline of degree $p-1$ has $p-1$ one-sided derivatives. The general theory says that pointwise smoothness for every function is between $s_{\max} - \frac{1}{2}$ and s_{\max} (this is a *Sobolev inequality*). For splines the pointwise smoothness is at the low end of that range, which is $p-1$.

Theorem 7.11 *If $\phi(t)$ has s derivatives in L^2 (integer s) then $s < p$. Allowing fractions, s_{\max} cannot exceed $p - \frac{1}{2}$. Pointwise, the smoothness cannot exceed $p - 1$.*

Proof. For $\phi(t)$ in L^2 we need at least one zero at π ($H(\pi) = 0$ by Chapter 6). For $\phi'(t)$ in L^2 we need at least two zeros at π (use the Lemma). Continuing, s derivatives in L^2 (integer s) require at least $s + 1$ zeros at π . Therefore $p \geq s + 1$ and $s < p$.

These conclusions are consistent with the s th derivative of the dilation equation:

$$\phi^{(s)}(t) = 2^s \sum 2h(k)\phi^{(s)}(2t - k). \quad (7.61)$$

The values $\phi^{(s)}(n)$ at the integers come from the eigenvalue problem $\Phi^{(s)} = 2^s M\Phi^{(s)}$. We know that M has eigenvalue 2^{-s} provided $s < p$. For higher derivatives we cannot even find values at the integers.

We caution that $s < p$ only means that equation (7.61) *might* produce an s th derivative in L^2 . It might not! Splines have the highest smoothness that p allows. Compare splines with other scaling functions when $p = 2$. The hat function has $s_{\max} = \frac{3}{2}$, the Daubechies function only has $s_{\max} = 1$. It has .55 derivatives in the pointwise sense [Daubechies-Lagarias]. Smoother wavelets have been constructed (orthogonal or biorthogonal). The neat fact is that the eigenvalues of T immediately give s_{\max} .

The Daubechies coefficients are $1 + \sqrt{3}, 3 + \sqrt{3}, 3 - \sqrt{3}, 1 - \sqrt{3}$ (divided by 8). The product $H(z)H(z^{-1})$ has coefficients $-1, 0, 9, 16, 9, 0, -1$ (divided by 16). This halfband property gives $(\dots, 0, 1, 0, \dots)$ in the center column of T :

$$T_5 = \frac{1}{16} \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 16 & 9 & 0 & -1 & 0 \\ 0 & 9 & 16 & 9 & 0 \\ 0 & -1 & 0 & 9 & 16 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix} \text{ has eigenvalues } 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}.$$

The eigenvector $a = (0, 0, 1, 0, 0)$ says that all $\phi(t - n)$ are orthogonal to $\phi(t)$. The approximation order is $p = 2$, since $H(\omega)$ has a double zero at π and $(\frac{1}{2})^4$ is not an eigenvalue of T . The smoothness index is $s_{\max} = 2 - 1 = 1$ because the largest "other eigenvalue" is the repeated $\lambda = 4^{-1}$. The function $\phi(t)$ has almost one derivative with finite energy. The integral of $|\omega|^2 |\widehat{\phi}(\omega)|^2$ is not finite, but any smaller power of $|\omega|$ will make it finite.

The smoothness of D_6 , with a triple zero at π , turns out to be $s_{\max} = 3 - \frac{\log 3}{\log 2}$. With accuracy p , the number of Daubechies coefficients is $2p$. Eirola computed the smoothness of all Daubechies functions up to D_{40} which has $p = 20$, and also found the asymptotic formula:

p	s_{\max}	p	s_{\max}	Asymptotically
1	0.5	6	2.388	$s_{\max} \approx 0.2075 p + \text{constant}$
2	1.0	7	2.658	
3	1.415	8	2.914	
4	1.775	9	3.161	
5	2.096	10	3.402	

Cascade Algorithm in the Frequency Domain

When t is rescaled to $2t$, the Fourier transform rescales to $\frac{\omega}{2}$. The shift by k that usually produces $e^{-ik\omega}$ now produces $e^{-ik\omega/2}$. The cascade equation $\phi^{(i+1)}(t) = \sum 2h(k)\phi^{(i)}(2t - k)$ transforms into

$$\widehat{\phi}^{(i+1)}(\omega) = \left(\sum h(k)e^{-ik\omega/2} \right) \widehat{\phi}^{(i)}\left(\frac{\omega}{2}\right) = H\left(\frac{\omega}{2}\right) \widehat{\phi}^{(i)}\left(\frac{\omega}{2}\right). \quad (7.62)$$

The first cascade step multiplies by $H\left(\frac{\omega}{2}\right)$ and the next step by $H\left(\frac{\omega}{4}\right)$. Thus $\widehat{\phi}^{(2)}(\omega)$ is $H\left(\frac{\omega}{4}\right)H\left(\frac{\omega}{2}\right)\widehat{\phi}^{(0)}\left(\frac{\omega}{4}\right)$. The output at step i involves $H^{(i)}$ with i factors:

$$\widehat{\phi}^{(i)}(\omega) = \left[\prod_{j=1}^i H\left(\frac{\omega}{2^j}\right) \right] \widehat{\phi}^{(0)}\left(\frac{\omega}{2^i}\right) = H^{(i)}(\omega) \widehat{\phi}^{(0)}\left(\frac{\omega}{2^i}\right). \quad (7.63)$$

We expect that this i -term product approaches the infinite product

$$\widehat{\phi}(\omega) = \text{limit of } \widehat{\phi}^{(i)}(\omega) = \prod_{j=1}^{\infty} H\left(\frac{\omega}{2^j}\right). \quad (7.64)$$

The question is: *When does this limit exist and how smooth is $\phi(t)$?* At each separate frequency ω , the limit exists. That requires only $H(0) = 1$ and a bound C on the derivative $|H'(\omega)|$. Then

$$\left| H\left(\frac{\omega}{2^j}\right) \right| \leq 1 + C \frac{|\omega|}{2^j} \leq e^{C|\omega|/2^j}.$$

If we take logarithms, to look at a sum instead of a product, that sum converges like $\sum C|\omega|/2^j$. The sum $\log |\widehat{\phi}(\omega)|$ is less than $C|\omega|$. Therefore the product $|\widehat{\phi}(\omega)|$ is less than $e^{C|\omega|}$.

Such a bound is useless for large ω ! For $\phi(t)$ to be a reasonable function, we need $\widehat{\phi}(\omega)$ to decay rather than grow as $|\omega| \rightarrow \infty$. Working with the energy $\int |\widehat{\phi}(\omega)|^2 d\omega$, there is no doubt that each iteration $\widehat{\phi}(\omega)$ retains finite energy. From step i to $i+1$ the energy grows by no more than

$$\|\widehat{\phi}^{(i+1)}\| \leq \|T\| \|\widehat{\phi}^{(i)}\| \quad \text{where} \quad \|T\|^2 = \max(|H(\omega)|^2 + |H(\omega + \pi)|^2). \quad (7.65)$$

The orthonormal case has $\|T\| = 1$, by Condition O. The energy is the same at every iteration of the cascade algorithm. For the biorthogonal case we expect $\|T\| > 1$ and the bound in (7.65) also becomes useless. To find the new energy $\|\widehat{\phi}^{(i+1)}\|^2$ from $\|\widehat{\phi}^{(i)}\|^2$, we look again at the operator T — which is absolutely fundamental to the theory of wavelets.

In the time domain, T is a double-shift Toeplitz matrix. That double shift corresponds to picking out *even frequencies* 0, 2ω , 4ω . In the z -domain it corresponds to picking out *even powers* of z :

Definition 7.1 The transition operator T in the three domains is:

$$\begin{aligned} \mathbf{T}\mathbf{a}(k) &= (\downarrow 2)2\mathbf{H}\mathbf{H}^T\mathbf{a}(k) \\ \mathbf{T}\mathbf{A}(2\omega) &= |H(\omega)|^2 A(\omega) + |H(\omega + \pi)|^2 A(\omega + \pi) \\ &= \text{even frequencies in } 2|H(\omega)|^2 A(\omega) \\ \mathbf{T}\mathbf{A}(z^2) &= H(z)A(z)H(z^{-1}) + H(-z)A(-z)H(-z^{-1}) \\ &= \text{even powers in } 2H(z)A(z)H(z^{-1}). \end{aligned}$$

$A(\omega)$ is the function $\sum a(k)e^{-ik\omega}$ and $A(z)$ is the function $\sum a(k)z^{-k}$. In our application, the $a(k)$ are inner products and $A(\omega)$ is taken from Section 6.4:

$$a(k) = \int \phi(t)\phi(t+k) dt \quad \text{and} \quad A(\omega) = \sum |\widehat{\phi}(\omega + 2\pi\ell)|^2.$$

We give an example immediately, and ask for more in the problem set.

Example 7.5. The coefficients $\mathbf{h}(k) = \frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ give

$$H(\omega) = \frac{1}{4}(1 + 2e^{-i\omega} + e^{-2i\omega}) \quad \text{and} \quad H(z) = (1 + 2z^{-1} + z^{-2}).$$

Suppose $\phi^{(0)}(t)$ is the box function, so its inner products are $\mathbf{a}^{(0)}(k) = (\dots, 0, 1, 0, \dots)$. The corresponding $A^{(0)}(\omega)$ is the constant function 1. The first step of the cascade algorithm produces $\phi^{(1)}(t)$ as three half-boxes with heights $\frac{1}{2}, 1, \frac{1}{2}$. The new energy $\|\phi^{(1)}\|^2$ is $\frac{1}{2}(\frac{1}{4} + 1 + \frac{1}{4}) = \frac{6}{8}$. This should agree with $\mathbf{a}^{(1)}(0)$ after the action of T :

$$\text{Time: } T \mathbf{a}^{(0)} = \frac{1}{8} \begin{bmatrix} 4 & 1 & 0 \\ 4 & 6 & 4 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 1 \\ 6 \\ 1 \end{bmatrix}$$

$$\begin{aligned} \text{Frequency: } A^{(1)}(2\omega) &= \text{even frequencies in } 2|H(\omega)|^2 A^{(0)}(\omega) \\ &= \text{even frequencies in } \frac{1}{8} (e^{2i\omega} + 4e^{i\omega} + 6 + 4e^{-i\omega} + e^{-2i\omega}) \\ A^{(1)}(\omega) &= \frac{1}{8} (e^{i\omega} + 6 + e^{-i\omega}) \end{aligned}$$

$$\begin{aligned} z\text{-domain: } A^{(1)}(z^2) &= \text{even powers in } 2H(z)A^{(0)}(z)H(z^{-1}) \\ &= \text{even powers in } \frac{1}{8}(z^2 + 4z + 6 + 4z^{-1} + z^{-2}) \\ A^{(1)}(z) &= \frac{1}{8}(z + 6 + z^{-1}). \end{aligned}$$

Summary: The main point is to connect L^2 convergence of the infinite product for $\widehat{\phi}(\omega)$ with the number p of vanishing moments of the wavelets, and the smoothness s of $\phi(t)$. Everything depends on the eigenvalues of the matrix T_{2N-1} :

Cascade convergence requires all $|\lambda| < 1$ except for a simple $\lambda = 1$.

Approximation of order p requires eigenvalues $\lambda = 1, \frac{1}{2}, \frac{1}{4}, \dots, (\frac{1}{2})^{2p-1}$.

Smoothness (s derivatives in L^2) requires all other $|\lambda| < 4^{-s}$.

Continuity of the Scaling Function

Functions in L^2 have finite energy. They may or may not be continuous. Continuity is a “pointwise” property, not revealed by inner products and not automatic under Condition E. (Haar satisfies this condition.) We describe now the test for continuity of $\phi(t)$. In borderline cases it is not always easy to apply, because it involves *two matrices*.

Those matrices are $\mathbf{m}(0)$ and $\mathbf{m}(1)$. Remember from Section 6.3 that these are $N \times N$ submatrices of the double-shift lowpass matrix $\mathbf{M} = (\downarrow 2)2\mathbf{H}$. The columns of $\mathbf{m}(0)$ and $\mathbf{m}(1)$ add to 1. If $\mathbf{e} = [1 \dots 1]$ is the all-ones row vector then $\mathbf{e}\mathbf{m}(0) = \mathbf{e}$ and $\mathbf{e}\mathbf{m}(1) = \mathbf{e}$. The dilation equation in vector form is on the interval $[0, 1)$:

$$\Phi(t) = \mathbf{m}(0) \Phi(2t) + \mathbf{m}(1) \Phi(2t - 1). \quad (7.66)$$

The vector $\Phi(t) = [\phi(t) \ \phi(t+1) \ \dots]^T$ stacks the N slices of $\phi(t)$. Because equation (7.66) has only two coefficients, it gives a simple recursion (Section 6.3). The first digit t_1 in $t = .t_1 t_2 t_3 \dots$ tells whether we use $\mathbf{m}(0)$ or $\mathbf{m}(1)$:

$$\Phi(t) = \mathbf{m}(t_1) \Phi(.t_2 t_3 \dots). \quad (7.67)$$

The next 0–1 digit t_2 tells whether the next step uses $m(0)$ or $m(1)$:

$$\Phi(t) = m(t_1)m(t_2)\Phi(.t_3t_4\cdots). \quad (7.68)$$

The matrices $m(0)$ and $m(1)$ can come in any order (determined by the digits in t). A nearby point T will begin with the same digits. At some later point the digits will differ. If $T = .t_1t_2T_3T_4\cdots$ then

$$\Phi(t) - \Phi(T) = m(t_1)m(t_2)[\Phi(.t_3t_4\cdots) - \Phi(.T_3T_4\cdots)]. \quad (7.69)$$

To prove continuity is to show that $\Phi(t)$ is close to $\Phi(T)$ when the neighbors t and T share many digits $t_1t_2\cdots t_K$. This will be true if the product of m 's in every order is small. Actually we work with matrices of order $N - 1$, after removing $\lambda = 1$.

Theorem 7.12 *The scaling function $\phi(t)$ is continuous if all products of $m_{N-1}(0)$ and $m_{N-1}(1)$ approach zero as the number of factors increases.*

The matrices $m(0)$ and $m(1)$ have the eigenvalue 1, with the all-ones left eigenvector e . All products of the m 's will have this eigenvalue and eigenvector. They won't go to zero! But in equation (7.69) these products multiply a vector that is orthogonal to e :

$$e \Phi(t) \equiv 1 \text{ implies that } e[\Phi(.t_3t_4\cdots) - \Phi(.T_3T_4\cdots)] = 1 - 1 = 0. \quad (7.70)$$

Restricted to vectors orthogonal to e , $m(0)$ becomes $m_{N-1}(0)$ and $m(1)$ becomes $m_{N-1}(1)$. If long products of these matrices are small, then (7.69) says that $\Phi(t)$ is close to $\Phi(T)$. This means that $\Phi(t)$ is continuous.

Continuity requires longer and longer products of $A = m_{N-1}(0)$ and $B = m_{N-1}(1)$ to approach zero. This may be easy to test, or hard. Any eigenvalue with $|\lambda| \geq 1$ guarantees failure. To have $\|A\| < 1$ and $\|B\| < 1$ guarantees success. But the right norm can be very difficult to find. The problem is the possibility of small eigenvalues and dangerous products:

$$A = \begin{bmatrix} \epsilon & 1 \\ 0 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} \epsilon & 0 \\ 1 & 0 \end{bmatrix} \text{ give } AB = \begin{bmatrix} 1 + \epsilon^2 & 0 \\ 0 & 0 \end{bmatrix}.$$

The powers A^n and B^n go to zero. But $ABABAB\cdots$ blows up because $1 + \epsilon^2 > 1$.

Problem 2 shows how to compute m_{N-1} from m_N . We cannot show how to test all products in all orders. [Heil-Colella] and many others discuss this problem but it has no complete solution.

Smoothness of Binary Filters

A striking example of the difference between pointwise smoothness and derivatives in L^2 is the filter $h = [-1 2 6 2 -1]/8$. Its scaling function is infinite at all dyadic points. Pointwise, $\phi(t)$ is a failure. The matrix $M = (\downarrow 2)H$ has a double eigenvalue at $\lambda = 1$, with only one eigenvector. The powers of M are therefore unbounded. But the eigenvalues of T —its MATLAB construction in Section 6.5—is followed by $\text{eig}(T)$ —are $\lambda = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, 0.5428, \dots$. The smoothness of $\phi(t)$ is $s_{\max} = -\log(.5428)/\log 4 = .44$. Thus $\phi(t)$ and $w(t)$ have 0.44 derivatives with finite energy, in spite of the fact that $\phi(t)$ blows up on a dense set!

The maxflat Daubechies filters of length $4p - 1$ have $2p$ zeros at π . Their scaling functions interpolate at the integers because D_{2p} is halfband: $\phi(n) = \delta(n)$ is the eigenvector of M for $\lambda = 1$. We expect high smoothness for $\phi(t)$ because of all the zeros:

$$\boxed{2p = 0, 2, 4, 6 \text{ leads to } s_{\max} = -0.5, 1.5, 2.44, 3.17.}$$

We were also interested in the smoothness of the new binary filters (dual to these symmetric Daubechies halfband filters). A new 9/7 pair was constructed by lifting in Section 6.5. There we also balanced its zeros to 10/6 by moving $1 + z^{-1}$ between analysis and synthesis. (This just changes s_{\max} by 1.) The new filters were

$$h9 = [1 \ 0 \ -8 \ 16 \ 46 \ 16 \ -8 \ 0 \ 1]/64 \text{ with 2 zeros and } s_{\max} = 0.59$$

$$h13 = [-1 \ 0 \ 18 \ -16 \ -63 \ 144 \ 348 \ \dots]/512 \text{ with 4 zeros and } s_{\max} = 1.18.$$

Compare with the standard symmetric biorthogonal FBI 9/7 pair, which has 16 nonzero coefficients. Two digits are inadequate but here they are:

$$hFBI = [0.03 \ -0.02 \ -0.08 \ 0.27 \ 0.60 \ \dots] \text{ and } fFBI = [-0.05 \ -0.03 \ 0.30 \ 0.56 \ \dots].$$

These have $s_{\max} = 1.4$ in analysis and 2.1 in synthesis. The FBI pair has higher coding gain and 4/4 zeros but no interpolating property. It gives higher PSNR and lower error on Lena and Barbara. But the perceptual quality of the new pair seems sharper (to our eyes). The analysis function $\tilde{\phi}_{\text{new}}(t)$ is more peaked and the synthesis $\phi_{\text{new}}(t)$ is smoother. Our latest test on the “boats” image at 0.32 bpp was a tie in objective measures (PSNR, MSE, Max error), but we see more in the new image (Figure 7.4). The cable at the upper right is lost by the standard 9/7 pair, and the ship name PICARDY becomes unreadable. You see that the reality of filter comparison is not totally precise!



Figure 7.4: Original of *boats* and two competing 9/7 reconstructions.

Problem Set 7.3

- Suppose $eA = e$. If x is a vector perpendicular to e , show that Ax is perpendicular to e . (If $ex = 0$ prove that $e(Ax) = 0$.)
- Suppose $eA = e$. Show that multiplying $S^{-1}AS$ by blocks gives

$$\begin{bmatrix} I_{N-1} & \mathbf{0}_{N-1} \\ e_{N-1} & 1 \end{bmatrix} \begin{bmatrix} a_{N-1} & b_{N-1} \\ c_{N-1} & d \end{bmatrix} \begin{bmatrix} I_{N-1} & \mathbf{0}_{N-1} \\ -e_{N-1} & 1 \end{bmatrix} = \begin{bmatrix} A_{N-1} & b_{N-1} \\ \mathbf{0}_{N-1} & 1 \end{bmatrix}.$$

The first $N - 1$ columns of S are perpendicular to e . The matrix $A_{N-1} = a_{N-1} - b_{N-1}e_{N-1}$ is the restriction of A to those vectors. Compute this restriction $m_{N-1}(0)$ for the matrix $m(0)$ from the hat coefficients $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$.

- For the transform $\widehat{\phi}(\omega) = (1 - e^{-i\omega})^2 / i^2\omega^2$ of the hat function show that $\int |\omega|^{2s} |\widehat{\phi}(\omega)|^2 d\omega < \infty$ if and only if $s < \frac{3}{2}$.
- Find by hand the matrix T and its eigenvalues and s_{\max} , starting from the filters $h = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$ and $h = (\frac{1}{6}, \frac{1}{2}, \frac{1}{3})$.
- Find by MATLAB the matrix T and its eigenvalues and s_{\max} for the “ D_6 filter” that has $h * h^T = \left(\frac{1+z^{-1}}{2}\right)^6 Q(z)$ = halfband of degree 10.
- Show from their lengths that h and f and $p = h * f$ cannot all be symmetric halfband.
- Dual to $f = [-1 \ 0 \ 9 \ 16 \ 9 \ 0 \ -1]/32$ is another halfband but unsymmetric filter $h^* = [1 \ 0 \ 23 \ 16 \ -9 \ 0 \ 1]/32$. Verify that $p^* = f * h^*$ is halfband to give PR. This is our first unsymmetric product filter. What is the system delay l ? What is $\lambda_{\max}(T^*)$ coming from h^* ? Why is $\tilde{\phi}^*(t)$ interpolating (equal to δ at the integers)? How smooth is it?
- Explain why the smoothness of the delta function is $s_{\max} = -\frac{1}{2}$.

7.4 Splines and Semiorthogonal Wavelets

Splines are piecewise polynomials, with a smooth fit between the pieces. They are older than wavelets. The “two-scale equation” or dilation equation was at first not particularly noticed. Now we will see that the numbers $h(k)$ are binomial coefficients, directly from Pascal’s triangle.

For a cubic spline the coefficients are 1, 4, 6, 4, 1 divided by 16. The transfer function is $H(z) = (1 + z^{-1})^4 / 16$. All four zeros are at $z = -1$. The filter is lowpass, the spline is as smooth as possible, and it has the highest accuracy $p = 4$ that is possible with $N = 4$. Almost every formula in this book comes out neatly and explicitly for splines.

One application of splines is to *interpolation*, when data points need to be connected by a smooth curve. To put one high-degree polynomial through all the points is very unwise. A small movement of a single point produces an extreme change in the polynomial (which oscillates violently between the interpolation points). It is much better to use short pieces of low-degree polynomials — often cubic splines.

A cubic spline has degree 3, for any number of interpolation points. It has *two continuous derivatives*, at the points where two different cubics meet. Thus $\phi_+(t)$ and $\phi'_+(t)$ and $\phi''_+(t)$ on one side of the meeting point agree with $\phi_-(t)$ and $\phi'_-(t)$ and $\phi''_-(t)$ on the other side. When $t = 0$ is the meeting point, only the coefficient of t^3 can change. The curve looks smooth and its coefficients are easy to find from the data points — but not trivial. There is a system of linear equations to solve, because all data points influence all coefficients. To say this in another way, the spline that matches the data values 0, 0, 1, 0, 0, ... is not zero in the intervals between those

values. It decays exponentially as $|t| \rightarrow \infty$ but this “cardinal spline” does not give the best basis for computations.

The good function is the “B-spline” with compact support. It is our scaling function $\phi(t)$. This function matches the data values $0, \frac{1}{6}, \frac{4}{6}, \frac{1}{6}, 0$, at the integers. It has unit area $\int \phi(t)dt = 1$ and a smooth fit (two derivatives). The spline is nonzero on *four intervals* (Figure 7.5). Outside this range $\phi(t)$ is identically zero.

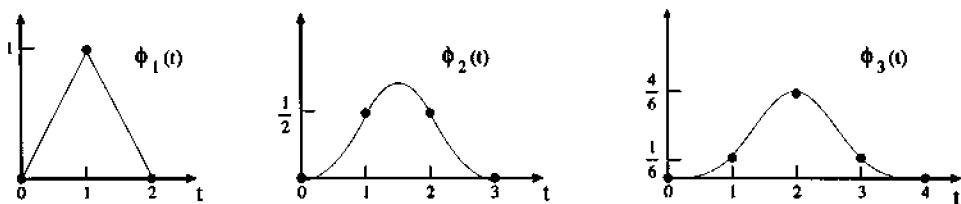


Figure 7.5: The spline $\phi_{N-1}(t)$ of degree $N - 1$ is the convolution of N box functions. It is supported on $[0, N]$. Its filter $H(z) = (\frac{1+z^{-1}}{2})^N$ has a zero of order $p = N$ at $z = -1$.

A note on interpolation at the integers $t = n$. To match a set of values $f(n)$, we look for a combination of B-splines. The equations are $\sum F(k)\phi(n - k) = f(n)$. This is a constant-diagonal (Toeplitz) system, with $\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$ on the diagonals coming from $\phi(n)$. To compute $F(k)$ we are inverting an FIR filter, and the inverse is IIR (a recursive filter). This explains why all $F(k)$ are nonzero when we interpolate an impulse $f(n) = \delta(n)$. It is like Shannon’s Sampling Theorem, with cardinal splines instead of sinc functions. *In fact the splines converge to $\frac{\sin \pi t}{\pi t}$ as $p = N \rightarrow \infty$.*

One major advantage: Splines do not require equally spaced data points. At the limit of unequal spacing, pairs of nodes come together. The spline becomes a *finite element* with *one* continuous derivative. The cubic finite element with values $0, 1, 0$, and slopes $0, 0, 0$, is nonzero only on *two intervals* around the center point. A second cubic element interpolates $0, 0, 0$, with slopes $0, 1, 0$. This short support (Figure 7.7) makes finite elements very popular in solving differential equations — much more popular than splines.

When there are two data values (function and slope) at the mesh points, each input $x(n)$ to the corresponding filter is a *pair of numbers*. We have a *multifilter* and it leads to *multiwavelets*. Everything is FIR. The values and slopes give the cubic in between. Multiwavelets are developed in the next section — we now return to the cubic spline $\phi(t)$.

Splines from Box Functions

Before two-scale equations, there was a direct approach to splines. This is still the fastest way. *The cubic B-spline is the convolution of four box functions:*

$$\phi_3(t) = (B * B * B * B)(t). \quad (7.71)$$

The results from each convolution step are in Figure 7.5. These are linear splines $\phi_1(t)$, quadratic splines $\phi_2(t)$, and cubic splines $\phi_3(t)$. They are in the continuity classes C^0, C^1 , and C^2 , since they have 0, 1, and 2 continuous derivatives. *The box function is $\phi_0(t)$.* The convolution of N box functions has degree $N - 1$ in each piece, with $N - 2$ continuous derivatives between pieces.

We want to show that the fourth derivative of a cubic spline is a sequence of delta functions. The coefficients of these delta functions are 1, -4, 6, -4, 1. These are the jumps in the third derivative, and this binomial pattern applies for every N . The first derivative of the hat function has jumps 1, -2, 1.

To take the fourth derivative of a cubic spline, or the N th derivative of a convolution of N box functions, we can work in time or frequency. We do both. First is the pleasant computation of $\widehat{\phi}_{N-1}(\omega)$ from convolving N box functions:

$$\widehat{\phi}_{N-1}(\omega) = (\widehat{\phi}_0(\omega))^N = \left(\frac{1}{i\omega}\right)^N (1 - e^{-i\omega})^N. \quad (7.72)$$

The box function has $\widehat{\phi}_0(\omega) = \int_0^1 e^{-i\omega t} dt = \frac{1}{i\omega}(1 - e^{-i\omega})$. Convolution in t is multiplication in ω , so the convolution of N box functions has transform $[\widehat{\phi}_0(\omega)]^N$.

Theorem 7.13 *The convolution of N boxes is a piecewise polynomial $\phi_{N-1}(t)$ of degree $N - 1$. The jumps in the $(N - 1)$ st derivative at $t = 0, 1, \dots, N$ are the alternating binomial coefficients $(-1)^t \binom{N}{t}$.*

Proof in the frequency domain: Each derivative multiplies the transform by $i\omega$. The N th derivative cancels the denominator in $\widehat{\phi}_{N-1}(\omega)$ and has transform $(1 - e^{-i\omega})^N$. This is the transform of a sequence of delta functions at the points $t = 0, 1, \dots, N$. Since the N th derivative is zero between those points, the spline $\widehat{\phi}_{N-1}(t)$ must be a piecewise polynomial of degree $N - 1$.

The fourth derivative of $\phi_3(t)$ has transform $(1 - e^{-i\omega})^4$. So the third derivative has jumps 1, -4, 6, -4, 1.

Proof in the time domain: The derivative of $f(t) * g(t)$ is $f'(t) * g(t)$ or equally it is $f(t) * g'(t)$. The fourth derivative of the cubic $\phi = B * B * B * B$ has four factors:

$$\frac{d^4\phi}{dt^4} = (B' * B' * B' * B')(t). \quad (7.73)$$

Each factor $B'(t)$ is $\delta(t) - \delta(t - 1)$. This is the derivative of the box function, which jumps up at $t = 0$ and down at $t = 1$. The convolution (7.73) becomes

$$\frac{d^4\phi}{dt^4} = \delta(t) - 4\delta(t - 1) + 6\delta(t - 2) - 4\delta(t - 3) + \delta(t - 4). \quad (7.74)$$

The third derivative has jumps 1, -4, 6, -4, 1 at $t = 0, 1, 2, 3, 4$. Otherwise $d^4\phi/dt^4 = 0$ and $\phi(t)$ is an ordinary cubic polynomial.

The Coefficients $h(n)$ for Splines

We know that the filter coefficients $\frac{1}{2}, \frac{1}{2}$ lead to the box function $\phi_0(t)$. The coefficients $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ lead to the hat function $\phi_1(t)$. These numbers appear in the lowpass filter H . We suspect that the coefficients $h(n) = \frac{1}{16}(1, 4, 6, 4, 1)$ lead to the cubic spline $\phi_3(t)$. The pattern for the convolution of N box functions could be described in two ways:

1. $h(n)$ are the coefficients in the special polynomial $H(z) = \left(\frac{1+z^{-1}}{2}\right)^N$.
2. $h(n)$ is the convolution $(\frac{1}{2}, \frac{1}{2}) * (\frac{1}{2}, \frac{1}{2}) * \dots * (\frac{1}{2}, \frac{1}{2})$.

Those two patterns are equivalent. Each multiplication by $(\frac{1+z^{-1}}{2})$ in the z -domain is convolution by $(\frac{1}{2}, \frac{1}{2})$ in the time domain. The question is, what is the effect on the scaling function? Apparently it is convolved with the box function.

This suggests a more general pattern. The transfer function $F(z)G(z)$ and the filter coefficients $f(n)*g(n)$ correspond to the scaling function $\phi_f(t)*\phi_g(t)$. *Multiplication of filters gives convolution of scaling functions!* This is not hard to prove. It applies immediately to the special case $F(z) = \frac{1+z^{-1}}{2}$. Convolution with the box function gives the next spline scaling function, one degree higher.

Lemma 7.2 The scaling function $\phi_h(t)$ corresponding to $H = FG$ and to $h(n) = f(n)*g(n)$ is the convolution of the scaling functions for F and G :

$$\phi_h(t) = \phi_f(t)*\phi_g(t) \text{ and } \widehat{\phi}_h(\omega) = (\widehat{\phi}_f(\omega))(\widehat{\phi}_g(\omega)). \quad (7.75)$$

Proof The dilation equations for ϕ_f and ϕ_g involve ω and $\omega/2$:

$$\widehat{\phi}_f(\omega) = F(\frac{\omega}{2})\widehat{\phi}_f(\frac{\omega}{2}) \text{ and } \widehat{\phi}_g(\omega) = G(\frac{\omega}{2})\widehat{\phi}_g(\frac{\omega}{2}).$$

Multiply to get $\widehat{\phi}_h(\omega) = H(\frac{\omega}{2})\widehat{\phi}_h(\frac{\omega}{2})$. This is the dilation equation for H . If you like infinite products, multiply $\prod F(\omega/2^i)$ and $\prod G(\omega/2^i)$ to get $\widehat{\phi}_h(\omega) = \prod H(\omega/2^i)$.

Example 7.6. If the coefficients in f and g are both $\frac{1}{2}, \frac{1}{2}$, the scaling functions $\phi_f(t)$ and $\phi_g(t)$ are the box function. Then $h = \frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ produces the hat function:

$$\begin{aligned} (\frac{1}{2}, \frac{1}{2}) * (\frac{1}{2}, \frac{1}{2}) &= (\frac{1}{4}, \frac{1}{2}, \frac{1}{4}) \text{ because } \left(\frac{1+z^{-1}}{2}\right)\left(\frac{1+z^{-1}}{2}\right) = \frac{1+2z^{-1}+z^{-2}}{4}. \\ B(t)*B(t) &= \text{Box function} * \text{Box function} = \text{Hat function.} \end{aligned}$$

Every extra factor $(\frac{1+z^{-1}}{2})$ means a convolution with the box function. The new $\phi(t)$ has one more derivative. For splines, each new $\phi_N(t) = \phi_{N-1}(t)*B(t) = \int_0^1 \phi_{N-1}(t-x)dx$ is one degree higher. This gives a nice formula for the time derivative:

$$\phi'_N(t) = \int_0^1 \phi'_{N-1}(t-x)dx = \phi'_{N-1}(t) - \phi'_{N-1}(t-1). \quad (7.76)$$

The spline of degree $N-1$ is the convolution of N box functions, and the corresponding filter has $H(z) = (\frac{1+z^{-1}}{2})^N$. We can verify in one step that this spline satisfies the correct dilation equation! Its transform $\widehat{\phi}_{N-1}(\omega)$ agrees with $H(\frac{\omega}{2})\widehat{\phi}_{N-1}(\frac{\omega}{2})$:

$$\left(\frac{1-e^{-i\omega}}{i\omega}\right)^N = \left(\frac{1+e^{-i\omega/2}}{2}\right)^N \left(\frac{1-e^{-i\omega/2}}{i\omega/2}\right)^N. \quad (7.77)$$

We started with $\phi(t)$ and determined $h(k)$. Normally we do the opposite — solve the dilation equation for $\phi(t)$. Here we have found the dilation equation for a spline:

$$\phi_{N-1}(t) = 2^{1-N} \sum_0^N \binom{N}{k} \phi_{N-1}(2t-k). \quad (7.78)$$

The lowpass space V_0 contains all smooth splines of degree $N-1$ on unit intervals. The B-splines are its basis (not orthogonal!). The space V_1 contains piecewise polynomials on half-intervals,

and therefore contains V_0 . The dilation equation (7.78) expresses $\phi_{N-1}(t)$ in V_0 as a combination of basis functions of V_1 . It remains to find the other space W_0 which contains the wavelets.

Summary: $H(z)$ has a zero of order $p = N$ at $z = -1$. There are N zeros at $\omega = \pi$. Correspondingly, the polynomials $1, t, t^2, \dots, t^{N-1}$ are combinations of the splines of degree $N-1$. Those polynomials are all in the lowpass space V_0 . The accuracy is $p = N$. The wavelets will have N vanishing moments. What are those spline wavelets?

Inner Products and Riesz Bounds

The inner products of splines with their translates give the values of higher splines. The inner products for the hat function are $a = \frac{1}{6}, \frac{4}{6}, \frac{1}{6}$:

$$a(0) = 2 \int_0^1 t^2 dt = \frac{4}{6} \quad \text{and} \quad a(1) = a(-1) = 2 \int_0^1 t(t-1) dt = \frac{1}{6}.$$

Those numbers agree with the cubic B-spline at $t = 1, 2, 3$. The formula is:

$$\text{Spline Inner Products} \quad a(k) = \int_{-\infty}^{\infty} \phi_{N-1}(t) \phi_{N-1}(t+k) dt = \phi_{2N-1}(N+k). \quad (7.79)$$

The integral is convolving N box functions with N more box functions — and shifting by k . The $2N$ boxes produce the higher spline ϕ_{2N-1} .

The vector a of inner products solves $a = Ta$. The operator T comes from the product $H(z)H(z^{-1})$, which for splines is $(\frac{1+z^{-1}}{2})^N (\frac{1+z}{2})^N$. This is z^N times the function $(\frac{1+z^{-1}}{2})^{2N}$ that gives the higher spline $\phi_{2N-1}(t)$. In other words, the matrix T for the lower spline is identical to the matrix M for the higher spline.

The eigenvector of M gives the values of the higher spline at the integers. The same eigenvector (of T !) gives the inner products of the lower splines in (7.79). All inner products have $a(k) \geq 0$, because all splines have $\phi(t) \geq 0$. The inner products sum to 1. The maximum value of $A(\omega) = \sum a(k)e^{-ik\omega}$ is $B = 1$, which occurs at $\omega = 0$. It is not hard to show that the minimum value of $A(\omega)$ occurs at $\omega = \pi$:

$$A_{\min} = \sum (-1)^k \phi_{2N-1}(N+k). \quad (7.80)$$

Thus the translates $\phi_{N-1}(t-k)$ form a Riesz basis for V_0 with bounds A_{\min} and 1. For the hat function ($N = 2$) the lower bound is $A_{\min} = -\frac{1}{6} + \frac{4}{6} - \frac{1}{6} = \frac{1}{3}$. In frequency this means that

$$\frac{1}{3} \leq A(\omega) = \sum_{-\infty}^{\infty} |\widehat{\phi}_1(\omega + 2\pi k)|^2 = \frac{4}{6} + \frac{2}{6} \cos \omega \leq 1. \quad (7.81)$$

The basis of hat functions is well conditioned. Of course it is not orthogonal. If we orthogonalize, following the shift-invariant method of Section 6.5, we get the Battle-Lemarié $\phi(t)$ and $w(t)$ — smooth, quickly decaying, infinite support.

Spline Wavelets

There are several important possibilities for the wavelets. One family is FIR and biorthogonal, the other is IIR and “semiorthogonal.” Semiorthogonal wavelets are perpendicular to the spline $\phi(t)$, but they are not orthogonal among themselves.

The FIR biorthogonal construction follows the usual rules. We need a halfband product filter $P(z)$. One factor is $(\frac{1+z^{-1}}{2})^N$, to produce the spline. This can be $H(z)$ in the analysis bank, or (better) it can be $F(z)$ in the synthesis bank. The other bank must contain an extra factor to make the product halfband. The natural choices for this second filter are $(\frac{1+z^{-1}}{2})^{2p-N} Q(z)$, which brings the product filter back to the standard Daubechies polynomial (where $Q(z)$ has degree $2p - N$). We need $2p > N$ to have a zero at $z = -1$. For the smallest p , the second filter may not satisfy Condition E for a stable basis in L^2 ; *more zeros may be needed*. The low degree spline filters and dual filters are the best known:

$$F(z) = \left(\frac{1+z^{-1}}{2}\right)^2 \text{ goes with } H(z) = \left(\frac{1+z^{-1}}{2}\right)^2(-1 + 4z^{-1} - z^{-2}) : \text{stable}$$

$$F(z) = \left(\frac{1+z^{-1}}{2}\right)^3 \text{ goes with } H(z) = \left(\frac{1+z^{-1}}{2}\right)(-1 + 4z^{-1} - z^{-2}) : \text{unstable}$$

$$F(z) = \left(\frac{1+z^{-1}}{2}\right)^4 \text{ goes with } H(z) = \left(\frac{1+z^{-1}}{2}\right)^4 Q_6(z) : \text{stable}.$$

The new book [CR] is a very good reference for biorthogonal FIR filters.

Example 7.7. Hat function from $F(z) = \left(\frac{1+z^{-1}}{2}\right)^2$ gives the 5/3 filter bank.

If the analysis filter has $H(z) \equiv 1$, the product with $F(z)$ is halfband. The analysis scaling function $\tilde{\phi}(t)$ is the delta function. Its translates $\delta(t - k)$ are biorthogonal to the hat functions $H(t - k)$. But the delta function is not acceptable.

If $H(z)$ is given two zeros at π , it needs the extra $Q(z)$:

$$H(z) = \left(\frac{1+z^{-1}}{2}\right)^2 \left(\frac{-1 + 4z^{-1} - z^{-2}}{2}\right) = \frac{-1 + 2z^{-1} + 6z^{-2} + 2z^{-3} - z^{-4}}{8}.$$

Those coefficients $-1, 2, 6, 2, -1$ appeared in the Guide to the Book. This is the analysis part of the biorthogonal 5/3 pair. The synthesis part is the linear spline (the hat). The product $P(z) = F(z)H(z)$ has four zeros at $z = -1$. Instead of the orthogonal D_4 factors of $P(z)$, the spline 5/3 factorization gives linear phase.

In these biorthogonal examples, one scaling function is a spline. Best if this is *synthesis*. The other scaling function is not a spline or a combination of splines or a piecewise polynomial. The space \tilde{V}_0 spanned by $\tilde{\phi}(t - k)$ is different from V_0 . This is normal, but spline people expect to live and work exclusively in spline spaces. They want $V_0 = \tilde{V}_0 = \text{all splines of degree } N - 1$. We now achieve this, but an IIR filter appears in the analysis half of the filter bank.

Semiorthogonal Wavelets

Start with a basis $\{\phi(t - k)\}$ for V_0 . Suppose this basis is not orthogonal. The hat function and the cubic B-spline are examples of $\phi(t)$. We want to find wavelets $w(t - k)$ that are *orthogonal* to $\phi(t)$. Thus we maintain what was true in the fully orthogonal case:

$$V_0 \perp W_0 \quad \text{and} \quad V_0 \oplus W_0 = V_1. \tag{7.82}$$

The spaces are orthogonal but the bases within those spaces are not orthogonal.

Multiresolution in the biorthogonal case always has

$$V_0 \perp \tilde{W}_0 \text{ and } W_0 \perp \tilde{V}_0 \text{ and } V_0 + W_0 = V_1 \text{ and } \tilde{V}_0 + \tilde{W}_0 = \tilde{V}_1. \quad (7.83)$$

Compare with (7.82) to see that $V_0 = \tilde{V}_0$ and $W_0 = \tilde{W}_0$. At every scale we will have $V_j \perp W_j$ and $V_j = \tilde{V}_j$ and $W_j = \tilde{W}_j$. There is only one multiresolution in the semiorthogonal case, one family $V_0 \subset V_1 \subset V_2$, as in the orthogonal case. The difference is that we have two bases for V_0 , the given basis $\phi(t - k)$ and the biorthogonal basis $\tilde{\phi}(t - k)$. This applies at every scale j . The tilde is needed for the dual basis, even if it is not needed for the space.

Semiorthogonality has an important property, directly from (7.83):

Semiorthogonal wavelets $w(2^j t - k)$ and $w(2^J t - l)$ are perpendicular if $j \neq J$.

At the same scale $j = J$, semiorthogonal wavelets are not generally perpendicular. But because W_j is orthogonal to V_j which contains all previous W_{j-1}, W_{j-2}, \dots , we are guaranteed that W_j is perpendicular to all wavelets at other scales.

To construct this new wavelet $w(t)$, we need the highpass coefficients $f_1(k)$. In the orthogonal case, they come from $f_0(k)$ by an alternating flip. $F_1(z)$ is $-z^{-N} F_0(-z^{-1})$, where z^{-1} gives the flip and $-z^{-1}$ makes it alternating. In that orthogonal case, the inner products $\langle \phi(t), \phi(t+k) \rangle$ are $a(k) = \delta(k)$. The polynomial $A(z) = \sum a(k)z^{-k}$ is identically 1. In the semiorthogonal case, when $\{\phi(t - k)\}$ is not orthonormal, this inner product polynomial enters the highpass coefficients.

The highpass function $F_1(z)$ becomes an alternating flip of $F_0(z)A(z)$. The analysis filters become IIR. Here is the general rule for semiorthogonality:

Theorem 7.14 Suppose the lowpass $F_0(z)$ leads to scaling functions $\phi(t - k)$ whose inner products are the coefficients in $A(z)$. Then the highpass $F_1(z)$ that yields semiorthogonal wavelets $w(t - k)$ is the alternating flip of $F_0(z)A(z)$:

$$F_1(z) = -z^{1-2N} F_0(-z^{-1}) A(-z^{-1}). \quad (7.84)$$

Proof. We want $\int \phi(t)w(t - n) dt = 0$ for all n . Use the dilation equation for $\phi(t)$ and the wavelet equation for $w(t)$:

$$\int_{-\infty}^{\infty} \left[\sum 2f_0(k)\phi(2t - k) \right] \left[\sum 2f_1(\ell)\phi(2t - 2n - \ell) \right] dt = 0. \quad (7.85)$$

Change the second sum to $\sum 2f_1(\ell - 2n)\phi(2t - \ell)$. The inner product of $\phi(2t - k)$ with $\phi(2t - \ell)$ is $a(\ell - k)$. The orthogonality requirement (7.89) becomes

$$\sum_{\ell} \sum_k f_0(k)a(\ell - k)f_1(\ell - 2n) = 0. \quad (7.86)$$

The highpass filter is double-shift orthogonal, but not to the lowpass filter. The double-shift orthogonality is to the sequence $\sum f_0(k)a(\ell - k)$ which corresponds to $F_0(z)A(z)$. Therefore $F_1(z)$ comes from $F_0(z)A(z)$ by an alternating flip. This completes the proof.

The sequence $f_0(0), \dots, f_0(N)$ gives a scaling function supported on $[0, N]$. The inner product $a(N) = \int \phi(t)\phi(t + N) dt = a(-N)$ is automatically zero. At most, the symmetric polynomial $A(z)$ has terms $a(k)z^{-k}$ and $a(k)z^k$ for $k < N$. Note that $A(z) = A(z^{-1})$. The degree of $F_0(z)A(z)$ is at most $2N - 1$.

Example 7.8. The hat function has $a(0) = \frac{4}{6}$ and $a(1) = a(-1) = \frac{1}{6}$. Find $F_1(z)$.

The product $F_0(z)A(z)$ is $\frac{1}{24}(1 + 2z^{-1} + z^{-2})(z + 4 + z^{-1})$. By alternating flip

$$\begin{aligned} F_1(z) &= -\frac{z^{-3}}{24}(1 - 2z + z^2)(-z + 4 - z^{-1}) \\ &= \frac{1}{24}(1 - 6z^{-1} + 10z^{-2} - 6z^{-3} + z^{-4}). \end{aligned}$$

The highpass coefficients $2f_1(k)$ yield $w(t)$ as drawn in Figure 7.6:

$$w(t) = \frac{1}{12}[\phi(2t) - 6\phi(2t-1) + 10\phi(2t-2) - 6\phi(2t-3) + \phi(2t-4)]. \quad (7.87)$$

This is orthogonal to all hat functions $\phi(t-k)$. It is *not* orthogonal to all $w(t-k)$. But it is orthogonal to every $w(2^j t - k)$ for $j \neq 0$. That is semiorthogonality.

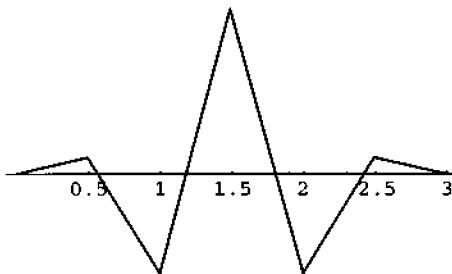


Figure 7.6: The linear wavelet $w(t)$ orthogonal to the hat functions $\phi(t-k)$.

The IIR Half of the Semiorthogonal Filter Bank

The analysis functions $H_0(z)$ and $H_1(z)$ must come from the perfect reconstruction condition $H_m(z)F_m(z) = 2z^{-\ell}\mathbf{I}$. We know the modulation matrix $F_m(z)$. Its inverse times $2z^{-\ell}$ gives $H_m(z)$:

$$\begin{aligned} \begin{bmatrix} F_0(z) & F_1(z) \\ F_0(-z) & F_1(-z) \end{bmatrix} &= \begin{bmatrix} F_0(z) & -z^{1-2N}F_0(-z^{-1})A(-z^{-1}) \\ F_0(-z) & z^{1-2N}F_0(z^{-1})A(z^{-1}) \end{bmatrix} \\ \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} &= \frac{2z^{-\ell}}{D(z)} \begin{bmatrix} z^{1-2N}F_0(-z^{-1})A(z^{-1}) & z^{1-2N}F_0(-z^{-1})A(-z^{-1}) \\ -F_0(-z) & F_0(z) \end{bmatrix} \end{aligned}$$

The key is always in the determinant $D(z)$. For an FIR filter bank, $D(z)$ is the delay $z^{-\ell}$. For these semiorthogonal filters, we don't get a delay but we do get a remarkable formula:

$$\begin{aligned} D(z) &= z^{1-2N}[F_0(z)F_0(z^{-1})A(z^{-1}) + F_0(-z)F_0(-z^{-1})A(-z^{-1})] \\ &= z^{1-2N}A(z^{-2}). \end{aligned} \quad (7.88)$$

This is the equation $T\mathbf{a} = \mathbf{a}$ for the eigenvector of the transition matrix! The matrix T in the time domain is $(\downarrow 2)2FF^T$. In the z -domain this is exactly what appears in (7.88); the aliasing

term with $-z$ comes from (↓ 2). The determinant $D(z)$ is identified. It is an odd function, and we take $\ell = 2N - 1$. The analysis filters can be read from the matrix $H_m(z)$:

$$H_0(z) = \frac{2z^{-\ell} F_0(z^{-1}) A(z^{-1})}{A(z^{-2})} \quad \text{and} \quad H_1(z) = \frac{2F_0(-z)}{A(z^{-2})}. \quad (7.89)$$

The division by $A(z^{-2})$ yields IIR filters except in the orthogonal case when $A \equiv 1$.

Example 7.9. (Hats continued) The modulation matrix from F_0 and F_1 is

$$\frac{1}{4} \cdot \frac{1}{24} \begin{bmatrix} 1 + 2z^{-1} + z^{-2} & 1 - 6z^{-1} + 10z^{-2} - 6z^{-3} + z^{-4} \\ 1 - 2z^{-1} + z^{-2} & 1 + 6z^{-1} + 10z^{-2} + 6z^{-3} + z^{-4} \end{bmatrix}.$$

The determinant is verified to be $\frac{1}{6}(z^{-1} + 4z^{-3} + z^{-5}) = z^{-3}A(z^{-2})$.

The synthesis coefficients require a division by $A(z^{-2})$. This has zeros at $z^2 = 2 \pm \sqrt{3}$. We can express $1/A(z^{-2})$ by partial fractions (Problem 7). The power series for these fractions will have $(2 - \sqrt{3})^n$ in the coefficients of z^{2n} and z^{-2n} . That gives the decay rate of the (IIR!) filter coefficients $h(k)$. The scaling functions $\tilde{\phi}(t - k)$ are biorthogonal to the hats.

Summary: Splines open the possibility of maintaining $V_j \perp W_j$ in the biorthogonal case. Then the wavelets are semiorthogonal. The highpass coefficients involve the inner products $a(k)$ of the scaling functions. The associated polynomial $A(z)$ is called the “*Euler-Frobenius polynomial*” in spline theory, but there is no restriction to splines. For every lowpass filter that yields a scaling function $\phi(t)$, the theory produces semiorthogonal wavelets.

For splines of degree $p - 1$, the inner products $a(k)$ are the values at integers of the B-splines of degree $2p - 1$. For all cases, the inner products are in the eigenvector $Ta = a$.

We emphasize that the special properties of splines are attracting a 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 high order p with low constant C in the error $C(\Delta t)^p$. The dual analysis filter has to be longer, but no construction will ever be perfect.

Problem Set 7.4

1. Prove that the spline $\phi_{N-1}(t)$ is symmetric about the center point $t = N/2$.
2. Explain the formula $\phi_{N-1}(t) = \frac{1}{(N-1)!} \sum (-1)^k \binom{N}{k} (t-k)_+^{N-1}$. Here $(t-k)_+ = \max(t-k, 0)$. One proof uses the jumps in the $(N-1)$ st derivative.
3. Take derivatives in Problem 2 to verify $\phi'_N(t) = \phi_{N-1}(t) - \phi_{N-1}(t-1)$.
4. (Challenge) Prove that $\phi_N(t) = \frac{1}{N}\phi_{N-1}(t) + \frac{N+1-t}{N}\phi_{N-1}(t-1)$. This recursion gives a quick stable computation of $\phi_N(t)$. Hint: take out a factor to get $\phi_N(t) = \frac{1}{N}\phi'_N(t) + \frac{N+1}{N}\phi_{N-1}(t-1)$. Use Problems 2–3.
5. Suppose $P(z) = F(z)H(z)$ is centered halfband. Why is $\phi_f(t) * \phi_h(t)$ equal to $\delta(n)$ at $t = n$?
6. Find the polyphase matrix for $h_0(z) = \frac{1}{4}(1, 2, 1)$ and $h_1(z) = \frac{1}{24}(1, -6, 10, -6, 1)$. Connect its determinant to $A(z)$.

7. Find A and B in the partial fraction expansion

$$\frac{6}{x^2 + 4x + 1} = \frac{A}{x + 2 - \sqrt{3}} + \frac{B}{x + 2 + \sqrt{3}}.$$

Expand the last two fractions in powers of $\frac{2-\sqrt{3}}{x}$ and $\frac{x}{2+\sqrt{3}}$. Combine into a power series for $6/(x + 4 + x^{-1})$. The coefficients of x^n and x^{-n} should be equal.

8. Show that the inner products $a(k) = \int \phi(t)\phi(t+k) dt$ for quadratic splines are $a = \frac{1}{120}(1, 26, 66, 26, 1)$. You could verify that this is the eigenvector in $Ta = a$, or evaluate the 5th-degree spline at the integers. What is the support interval of the semiorthogonal quadratic $w(t)$?
9. For splines show that $A(\omega) = \sum |\widehat{\phi}_{N-1}(\omega+2\pi k)|^2$ equals $(2 \sin \frac{\omega}{2})^{2N} \sum (\omega+2\pi k)^{-2N}$. Verify $A'(\pi) = 0$. The minimum is at $\omega = \pi$.

7.5 Multifilters and Multiwavelets

This brief section describes a recent development—to allow the filter coefficients $h(k)$ to be $r \times r$ matrices. Each input sample $x(n)$ is a vector with r components. So is each output $\widehat{x}(n)$. The bank of multifilters has the same structure as an ordinary filter bank, with the extra freedom that comes with matrix coefficients.

In continuous time, the dilation equation will have matrix coefficients $h(k)$. The solution gives r scaling functions $\phi_1(t), \dots, \phi_r(t)$. Then the wavelet equation with highpass matrices $h_1(k)$ yields r corresponding wavelets. *Properly chosen, all these functions can have symmetry as well as orthogonality!* They and all their translates are orthogonal when the polyphase matrix (of order $2r$) is paraunitary. Multiresolution produces an orthonormal basis of wavelets—the translates of r functions at all scales $-\infty < j < \infty$:

$$w_{ijk}(t) = 2^{-j/2} w_i(2^j t - k), \quad 1 \leq i \leq r.$$

In some situations $r > 1$ is quite reasonable. When sampling a function $x(t)$, we may also sample its slope $x'(t)$. Velocity may be involved as well as displacement. The pair $(x(n), x'(n))$ is a vector with $r = 2$. Those samples could go through two separate scalar filter banks, or through one bank of multifilters. This example already shows, because $x'(t)$ is dimensionally different from $x(t)$, that scaling is important for the r inputs.

The cubic scaling functions in this $(x(n), x'(n))$ example are “finite elements.” They are drawn in Figure 7.7, with support $[0, 2]$. Like splines, these cubics $\phi_1(t)$ and $\phi_2(t)$ have linear phase—but they are not orthogonal to their translates. They have *one* continuous derivative, not two. The space of C^1 cubics has a more local basis (but with $r = 2$ functions per interval), while the spline space of C^2 cubics has one basis function per interval (the B-spline).

Figure 7.7 also displays the wavelets $w_1(t)$ and $w_2(t)$ in the semiorthogonal case. They are supported on $[0, 3]$. They and their translates span W_0 and are orthogonal to $\phi_1(t)$ and $\phi_2(t)$. They are piecewise cubic on half-intervals, and they come from a wavelet equation $w(t) = \sum 2h_1(k)\phi(2t-k)$. Those coefficients $h_1(k)$ are 2×2 matrices!

The finite element spaces V_0 of degree 1, 3, 5, 7 are spanned by $r = 1, 2, 3, 4$ scaling functions. Degree 1 has $\phi(t) = \text{hat function}$ which has C^0 smoothness (no continuous derivatives). From the polynomials contained in V_0 we know the accuracy $p = 2, 4, 6, 8$. The smoothness is C^0, C^1, C^2, C^3 (and the pattern continues). The r semiorthogonal multiwavelets are always

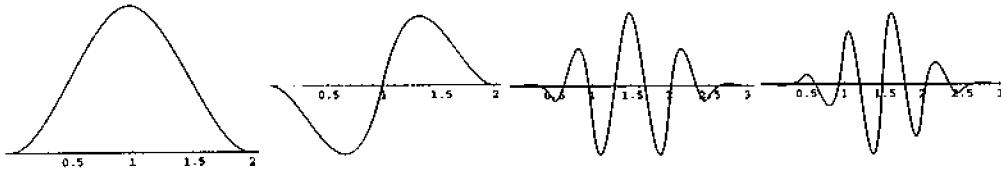


Figure 7.7: Cubic multiwavelets $w_1(t)$ and $w_2(t)$ orthogonal to cubic finite elements $\phi_1(t - k)$ and $\phi_2(t - k)$.

supported on $[0, 3]$. These finite element scaling functions are cousins to the splines — which have more smoothness but longer support. Here is a piecewise linear construction with even less smoothness and even shorter support.

Multiwavelet example: “Haar’s Hat” These functions are *linear* on each interval but *not continuous* at the ends. Each piece is a straight line between the end values. That piece is determined by $r = 2$ numbers, its average and its slope. Its average goes with the box function $\phi_1(t)$, its slope goes with $\phi_2(t) = 2t - 1$. Those scaling functions on $[0, 1]$ are combinations of the same functions on half-intervals:

$$\text{Box: } \phi_1(t) = \phi_1(2t) + \phi_1(2t - 1) \quad (\text{usual Haar equation})$$

$$\text{Slope: } \phi_2(t) = \frac{1}{2} [\phi_2(2t) + \phi_2(2t - 1) - \phi_1(2t) + \phi_1(2t - 1)] \quad (\text{zero outside } [0, 1])$$

This is a *matrix dilation equation*. The coefficients are 2×2 matrices $\mathbf{h}(0)$ and $\mathbf{h}(1)$:

$$\begin{bmatrix} \phi_1(t) \\ \phi_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \phi_1(2t) \\ \phi_2(2t) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \phi_1(2t - 1) \\ \phi_2(2t - 1) \end{bmatrix} \quad (7.90)$$

You see the eigenvalues 1 and $\frac{1}{2}$ that always accompany linear functions in V_0 . The accuracy is $p = 2$. The support is short! This is a block transform, not overlapping the next pair of samples — the average and slope on the next interval $[1, 2]$. We get good accuracy, but the price is complete lack of smoothness.

It is an exercise to find the wavelets for Haar’s Hat. They will be linear on half-intervals (and still discontinuous). Another example of two scaling functions: the real and imaginary parts of a complex scaling function. Symmetry with orthogonality is possible [Lena]. We turn to a more magical construction that combines orthogonality and symmetry and short support and continuity.

This special construction by Geronimo, Hardin, and Massopust gave a strong impetus to multiwavelet theory. The functions $\phi_1(t)$ and $\phi_2(t)$ in Figure 7.8 were found by a recursive interpolation process. The 2×2 coefficients $\mathbf{h}_0(k)$ in their dilation equation came later. So did the highpass coefficients and the wavelets [StSt1, GHM2]. All these functions have linear phase and short support and orthogonality. The only earlier example with these properties was Haar’s. Now the accuracy is $p = 2$, because the scaling functions can reproduce a hat:

$$\phi_1(t) + \phi_1(t - 1) + \phi_2(t) = \text{hat function.}$$

The space V_0 has an FIR orthogonal basis $\{\phi_1(t - k)\}$ joined with $\{\phi_2(t - k)\}$:

$$f_0(t) = \sum a_{10k} \phi_1(t - k) + a_{20k} \phi_2(t - k) \text{ is in } V_0. \quad (7.91)$$

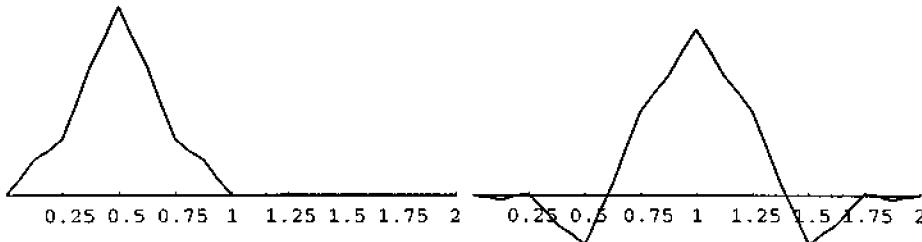


Figure 7.8: GHM scaling functions combine orthogonality of all translates and linear phase.

There is a serious problem for multifilters and multiwavelets. Everyone in signal processing asks about it immediately. We have twice as many filters (or r times as many filters) as usual. One input stream $x(n)$ produces $2r$ half-length outputs from the analysis bank. This means extra computation (but shorter filters). It also means that the stream $x(n)$ has to be *vectorized*—so r inputs go together. It is not at all satisfactory just to take the original stream in blocks of r samples. This does violence to the whole idea. The $\phi_i(t)$ are not time shifts of one function, they are r functions *at each time*.

The $(x(n), x'(n))$ example comes naturally in blocks of two at each time. So does Haar's Hat. For other multifilters, one possibility is to repeat $(x(n), cx(n))$ with a suitable scalar c . What we really want is a discrete form of (7.91). The recent papers [Heller2, XiaG] show how to convert a single input stream to multi-inputs which give samples of the data at half-intervals (thus $r = 2$ samples per interval). These Geronimo-Hardin-Massopust multiwavelets have given the first experimental results in compression. In competition against D_4 wavelets, with the same accuracy $p = 2$, the multiwavelets gave better compression and required more computations.

Perfect Reconstruction and Orthogonality and Accuracy

The theory of multiwavelets looks completely familiar up to one point, where everything changes. The multifilters still have transfer functions $H(z) = \sum h(k)z^{-k}$. These are now matrices. The perfect reconstruction conditions are the same as before:

$$\mathbf{F}_0(z)\mathbf{H}_0(z) + \mathbf{F}_1(z)\mathbf{H}_1(z) = 2z^{-\ell}\mathbf{I} \quad (7.92)$$

$$\mathbf{F}_0(z)\mathbf{H}_0(-z) + \mathbf{F}_1(z)\mathbf{H}_1(-z) = \mathbf{0}. \quad (7.93)$$

The big difference is that the anti-aliasing equation (7.93) is no longer satisfied by $\mathbf{F}_0(z) = \mathbf{H}_1(-z)$ and $\mathbf{F}_1(z) = -\mathbf{H}_0(-z)$. *The matrices \mathbf{H}_0 and \mathbf{H}_1 need not commute!* Probably they don't. A satisfactory construction method for PR (= biorthogonal) multifilters is not yet available.

For orthogonality the situation is similar. We want the synthesis filters to be the transposes $\mathbf{F}_0(z) = \mathbf{H}_0^T(z^{-1})$ and $\mathbf{F}_1(z) = \mathbf{H}_1^T(z^{-1})$ of the analysis filters (times a delay to make them causal). Omitting that delay, the PR conditions (7.92) require the modulation matrix to be paraunitary: $\mathbf{H}_m(z)\mathbf{H}_m^T(z^{-1}) = 2\mathbf{I}$. These matrices have order $2r$. The first block yields the famous Condition O:

$$\mathbf{H}_0(z)\mathbf{H}_0^T(z^{-1}) + \mathbf{H}_0(-z)\mathbf{H}_0^T(-z^{-1}) = 2\mathbf{I}. \quad (7.94)$$

Suppose this is achieved—it is the hard part. Then in the scalar case, H_1 is the alternating flip of H_0 . That no longer works. The lowpass and highpass rows

$$\begin{bmatrix} h(N) & h(N-1) & \cdots & h(1) & h(0) \\ -h(0) & h(1) & \cdots & -h(N-1) & h(N) \end{bmatrix}$$

are not orthogonal, if the $r \times r$ matrices $h(k)$ do not commute. We need to compute, from scratch, highpass coefficients $h_1(k)$ that will complete the paraunitary matrix $H_m/\sqrt{2}$.

This completion is possible [StSt1]. If we have r paraunitary rows of length $2r$, the factorization in equation (9.49) still exists. The constant matrix Q in that equation is $r \times 2r$. Complete it to a square constant orthogonal matrix. Then the factors multiply to give a square paraunitary matrix. Its last r rows contain the highpass coefficients we need.

Finally, we mention the accuracy p . As always, combinations of $\phi_i(t-k)$ must produce the polynomials $1, t, \dots, t^{p-1}$. In the scalar case, $H(z)$ will have a factor $(1+z^{-1})^p$. The matrix factorization is not so simple [CoDaPl]. Also in the scalar case, $M = (\downarrow 2)2H$ has eigenvalues $1, \frac{1}{2}, \dots, (\frac{1}{2})^{p-1}$. This is still the correct Condition A when the entries of M are $r \times r$ matrices:

$$M = 2 \begin{bmatrix} h(N) & h(N-1) & \cdots & \cdots \\ h(N) & h(N-1) & \cdots & \cdots \end{bmatrix}.$$

The column sums $2 \sum h(2k)$ and $2 \sum h(2k+1)$ were 1 in the scalar case. They need not be I in the matrix case. That would be far too restrictive. To achieve $p = 1$, the sums must have $\lambda = 1$ as an eigenvalue with the same left eigenvector. The condition for higher p is recursive [StSt2,HeStSt]. In some way the scalar case is understood more deeply, when the theory of multiwavelets forces us into the matrix case.

Problem Set 7.5

1. **(Haar's Hat)** Find wavelets $w_1(t)$ and $w_2(t)$ on $[0, 1]$ that are orthogonal to each other and to $\phi_1(t) = \text{box}$ and $\phi_2(t) = 2t - 1$. They will be linear on $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 1]$. Draw graphs of $w_1(t)$ and $w_2(t)$.
2. Express the wavelets in Problem 1 as combinations of $\phi_1(2t)$, $\phi_1(2t-1)$, $\phi_2(2t)$, $\phi_2(2t-1)$. What are the highpass matrix coefficients $h_1(0)$ and $h_1(1)$?
3. Display a 4×4 block of the analysis bank and verify orthogonality:

$$\text{block} = \begin{bmatrix} h(0) & h(1) \\ h_1(0) & h_1(1) \end{bmatrix} = \begin{bmatrix} \text{lowpass} \\ \text{highpass} \end{bmatrix}.$$

4. Find the matrix dilation equation for the C^1 cubics $\phi_1(t)$ and $\phi_2(t)$ drawn in Figure 7.8. They are combinations of C^1 cubics on half-intervals.
5. The Fourier transform gives what product formula to solve the matrix dilation equation $\phi(t) = \sum 2h(k)\phi(2t-k)$?

Chapter 8

Finite Length Signals

8.1 Circular Convolution and the DFT

Our signals have so far had infinite length. We expected $x(n)$ and $f(t)$ to be defined forever—for all integers n and all real numbers t . The data streams for audio signals are so long that this model is very reasonable. Other applications (like image processing) have *data streams of a definite length L* . Then the finiteness of the signal $x(0), \dots, x(L-1)$ must be taken into account.

The immediate problem is filtering a finite signal. The computation of $\sum h(k)x(n-k)$ may ask for $x(-1)$ which is not defined. The purpose of extension is to define it. The difficulty is not in the middle of the signal (unless the filter is IIR). The problem is near the ends. One possibility is to *change to a one-sided boundary filter*. The end values of x are processed without crossing the boundary. The other possibility is to *extend the signal beyond the boundary*.

We concentrate first on extending the signal, to define $x(n)$ for every n . Filtering this extension is equivalent to one particular set of boundary filters. Section 8.5 discusses boundary filters in general.

A finite-length filter bank uses L inputs to produce L outputs. An L by L matrix must be present. It is conceptually easiest to extend to an infinite signal, followed by ordinary time-invariant filtering. The question is how to extend, and there are many possibilities:

1. Extend by zeros (*zero-padding*) (*not studied*)
2. Extend by periodicity (*wraparound*) (*this section*)
3. Extend by reflection (*symmetric extension*) (*next section*).

These three methods are applied in Figure 8.1. You notice immediately a very important point:

Zero-padding and wraparound generally introduce a *jump in the function*.

Reflection generally introduces a *jump in the first derivative*.

Section 10.1 shows the three methods applied to a filtered image. The sky at the top is degraded by zero-padding. Wraparound is better but a thin error layer is visible. Symmetric extension is the best.

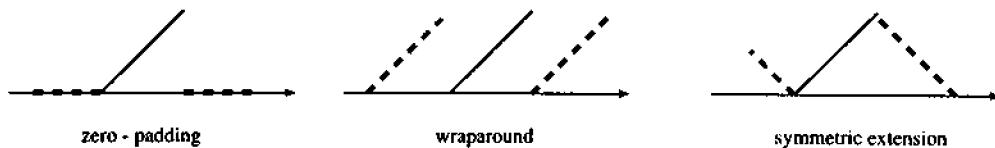


Figure 8.1: Three extension methods: zero-padding, wraparound, reflection.

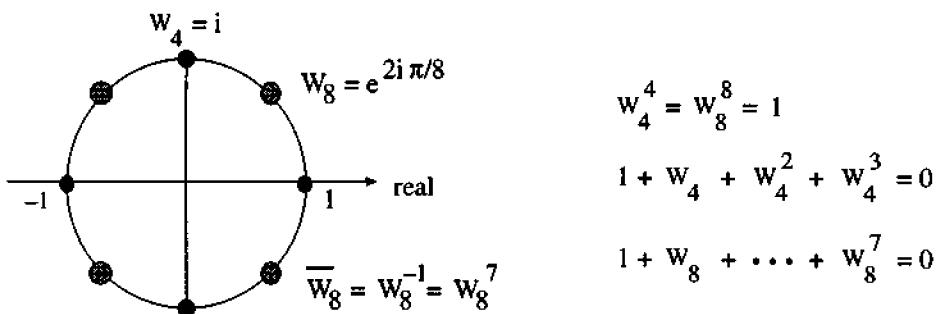


Figure 8.2: The fourth roots of 1 and the eighth roots of 1. They add to zero.

We do not pursue zero-padding in the text. It chops off the infinite Toeplitz matrix H , with no adjustment at the boundary. Wraparound is good if the original problem is genuinely periodic or close to it. The filter matrix becomes a *circulant*. All indices are interpreted “modulo L ”. The k th diagonal of the matrix contains $h(k)$, and it wraps around to continue as the $(k - L)$ th diagonal.

All circulant matrices are diagonalized by the DFT. Therefore multiplications and inversions are very fast. Convolution $h * x$ is still multiplication of transforms $H(z)X(z)$ (with $z^L = 1$). In frequency this is the component-by-component multiplication $\widehat{h}(k)\widehat{x}(k)$. The inverse matrix exists if all $h(k) \neq 0$, and it is also a circulant.

These circulant matrices are so simple and useful that we take time out to explain them (and also the DFT). Together they are a perfect match. Circulants have constant diagonals in the time domain. They are built out of (circular) shift matrices. The transform to frequency domain is executed at high speed by the Fast Fourier Transform, and we will put the FFT in a matrix form. The Fourier matrix is a product of very sparse matrices.

The next section returns to the critical question of symmetric extension. Do we want period $2(L - 1)$ or $2L$? The answer depends on the length of the signal and the length of the filter! We hope those details will be useful.

Note 1 Other extensions are possible and interesting. Zero-padding could change to constant-padding (almost as simple and more accurate). A more careful extension could have constant slope. Section 8.5 shows how to fit a polynomial and extrapolate. In all cases we only need to provide the filter with a few values (at most N values) beyond the endpoints 0 and $L - 1$. Symmetric extension finds those values entirely by data addressing, without extra computations.

When it helps the understanding, we are free to assume that the periodic signals exist at all

times $-\infty < n < \infty$. The essential problem is to implement a filter bank on a finite length signal, *without expanding that length*. Circular extension allows a probable jump at the endpoints.

Properties of the Circular Shift

The building blocks for time-invariant filters are the shift matrices S and S^{-1} . All components of a vector are delayed one step (by S) and advanced one step (by S^{-1}). The building blocks for circulants (*cyclic filters*) are cyclic shifts. Every component is delayed by S_L , except the last component which comes around to be the first component.

The 4 by 4 delay takes components 0, 1, 2, 3 *forward* into positions 1, 2, 3, 4 – except that 4 becomes 0. Algebraically, we work “modulo 4”. Intuitively, our problem has period 4, so that $x(4)$ is the same as $x(0)$. Here is S_4 multiplying the vector x :

$$S_4 x = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \begin{bmatrix} x(3) \\ x(0) \\ x(1) \\ x(2) \end{bmatrix}. \quad (8.1)$$

S_4 is a *cyclic permutation matrix*. It has the same rows as I , in a different order. Those rows are mutually orthogonal and they are unit vectors – so S_4 is an orthogonal matrix. The shift matrices have very special properties:

1. The inverse equals the transpose. The L th power of S_L is I .
2. The eigenvalues of S_L are the L roots of 1 in the complex plane (on the unit circle in Figure 8.2). They are the powers of the number $W = e^{2\pi i/L}$. They are also the powers of $\bar{W} = W^{-1} = e^{-2\pi i/L}$.

It is convenient to list the eigenvalues in the clockwise order $1, \bar{W}, \bar{W}^2, \dots$, because $\bar{W}^k = e^{-2\pi ik/L}$ fits into the Discrete Fourier Transform.

3. Corresponding to $\lambda = \bar{W}^k$ is the eigenvector $x = (1, W^k, W^{2k}, \dots)$. These eigenvectors are the columns of the L by L **Fourier matrix** = **DFT matrix** = F_L . The entries of F_L are $W^{nk} = e^{2\pi ink/L}$. This matrix diagonalizes the shift (and all circulants):

$$F_L^{-1} S_L F_L = D_L = \text{diag}(1, \bar{W}, \bar{W}^2, \dots). \quad (8.2)$$

We can quickly establish Properties 1–3, using S_4 as the example to work with:

$$S_4^T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{times} \quad S_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{is} \quad S_4^T S_4 = I.$$

An advance times a delay is the identity. This is true and useful on the whole line, with times $-\infty < n < \infty$. It is true here *on the circle*. The times are $0, \dots, L-1$, and they repeat. On an infinite half-line, $S_+^T S_+ = I$ is true but $S_+ S_+^T = I$ is not true! An advance S_+^T will wipe out the first component $x(0)$, and a delay S_+ cannot bring it back. Real boundaries are distinctly harder than periodic boundaries.

The fourth power of S_4 is the identity matrix. When we shift forward four times, the cycle completes – we come back to the start. Always $(S_L)^L = I$. Here are S_4 , S_4^2 , $S_4^3 = S_4^{-1}$, and $S_4^4 = I$:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Properties 2 and 3 are about eigenvalues and eigenvectors. To find the eigenvalues, we normally solve $\det(S - \lambda I) = 0$. For the L by L shift matrix, this determinant (see Problem 1) yields the equation $\lambda^L = 1$. The eigenvalues are roots of unity, in agreement with the fact that $S^L = I$.

The determinant of $\begin{bmatrix} -\lambda & 0 & 0 & 1 \\ 1 & -\lambda & 0 & 0 \\ 0 & 1 & -\lambda & 0 \\ 0 & 0 & 1 & -\lambda \end{bmatrix}$ *is* $\lambda^4 - 1$.

The four eigenvalues are $1, -i, -1, i$. Those fourth roots of 1 are equally spaced around the unit circle, at angles $\frac{8\pi}{4}, \frac{6\pi}{4}, \frac{4\pi}{4}, \frac{2\pi}{4}$. In general the L th roots are at multiples of the angle $\frac{2\pi}{L}$. Going clockwise around the circle, we have the L powers of $\bar{W} = e^{-2\pi i/L}$ in Figure 8.2.

For the circular shift, we can announce the eigenvectors at the same time. The eigenvector for $\lambda = 1$ is just $x = (1, 1, \dots, 1)$. The shift leaves it unchanged: $Sx = 1x$. The second eigenvalue \bar{W} has eigenvector $x = (1, W, W^2, \dots, W^{L-1})$. Shifting brings $W^{L-1} = W^{-1}$ to the top:

Circular shift of $\begin{bmatrix} 1 \\ W \\ W^2 \\ \vdots \\ W^{L-1} \end{bmatrix}$ *is* $\begin{bmatrix} W^{L-1} \\ 1 \\ W \\ \vdots \\ W^{L-2} \end{bmatrix} = W^{-1}x = \bar{W}x$.

Thus $Sx = \bar{W}x$ as required. Similarly the k th eigenvalue $\lambda = W^{-k} = \bar{W}^k$ has the eigenvector $x_k = (1, W^k, W^{2k}, \dots, W^{(L-1)k})$. Here is $Sx = \lambda x$:

Circular shift of $\begin{bmatrix} 1 \\ W^k \\ W^{2k} \\ \vdots \end{bmatrix}$ *is* $\begin{bmatrix} W^{(L-1)k} \\ 1 \\ W^k \\ \vdots \end{bmatrix} = W^{-k}x = \bar{W}^kx$.

We are making liberal use of the fact that $W^L = 1$. Now summarize:

Diagonalization of S by F . The eigenvectors of the shift S are the columns of the Fourier matrix F . The matrix $F^{-1}SF$ is diagonal:

$$F^{-1}SF = D \quad \text{and} \quad SF = FD \quad \text{and} \quad S = FDF^{-1}. \quad (8.3)$$

D is the diagonal matrix of eigenvalues $1, \bar{W}, \bar{W}^2, \dots$. For $L = 4$ these matrices are displayed

in the multiplication $S_4 F_4 = F_4 D_4$, whose columns are $Sx = \lambda x$:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} \begin{bmatrix} 1 \\ i \\ i^2 \\ i^3 \end{bmatrix} \quad (8.4)$$

The columns of F are *orthogonal eigenvectors* of S , with length \sqrt{L} .

Why are the eigenvectors orthogonal? The answer from linear algebra is: every real matrix with the property $SS^T = S^T S$ has orthogonal eigenvectors. These are the “normal matrices.” In our case $SS^T = S^T S = I$.

A second answer is to verify orthogonality directly. The eigenvectors of S (the columns of F) are complex even though S is real. Permutations like S generally have complex eigenvalues and eigenvectors. Take the inner product:

$$\bar{x}_k^T x_\ell = \begin{bmatrix} 1 & \bar{W}^k & \bar{W}^{2k} & \dots \end{bmatrix} \begin{bmatrix} 1 \\ W^\ell \\ W^{2\ell} \\ \vdots \end{bmatrix} = 1 + z + z^2 + \dots + z^{L-1} = 0.$$

Here $z = \bar{W}^k W^\ell$. The sum is $(1 - z^L)/(1 - z)$. This is zero because $z^L = 1$ (remember that $W^L = 1$). So the eigenvectors of S are orthogonal to each other: $\bar{F}F = LI$. We are ready to move forward to circulants, via the DFT.

Discrete Fourier Transform and Circulants

An ordinary filter H is a combination of powers of the (infinite) shift matrix S . A circular filter H_L is a combination of powers (with $N < L$) of the finite shift matrix S_L :

$$\text{Ordinary filter } H = \sum_{-\infty}^{\infty} h(k)S^k \quad \text{Circular filter } H_L = \sum_{n=0}^N h(n)(S_L)^n.$$

H is a Toeplitz matrix and H_L is a *circulant matrix*. Multiplication by H is convolution. Multiplication by H_L is circular convolution.

For $L = 4$, the four powers of S_4 were displayed earlier. They combine into a typical 4×4 circulant matrix $H_4 = h(0)I + h(1)S_4 + h(2)S_4^2 + h(3)S_4^3$:

$$H_4 = \begin{bmatrix} h(0) & h(3) & h(2) & h(1) \\ h(1) & h(0) & h(3) & h(2) \\ h(2) & h(1) & h(0) & h(3) \\ h(3) & h(2) & h(1) & h(0) \end{bmatrix}. \quad (8.5)$$

Notice how the diagonals wrap around. We still have $h(n)$ along diagonal n . In this circular world 3 is the same as -1 , so $h(3)$ is also on diagonal -1 . With periodicity we work modulo L . Two numbers are the same ($m \equiv n \pmod{L}$) when $m - n$ is a multiple of L .

Usually N is much smaller than L . We have a band of nonzeros below the diagonal, wrapping around to the upper right corner (as in S). Otherwise H_L is zero. We give the properties of

\mathbf{H}_L and then the applications. The powers of \mathbf{S} all have the same eigenvectors as \mathbf{S} , so the key to a circulant is its *eigenvectors* and *eigenvalues*.

Diagonalization of \mathbf{H}_L by \mathbf{F}_L . The eigenvectors of \mathbf{H}_L are the columns of the Fourier matrix \mathbf{F}_L . The eigenvalues appear in the discrete Fourier transform $(\widehat{\mathbf{h}}(0), \widehat{\mathbf{h}}(1), \widehat{\mathbf{h}}(2), \dots)$ of the coefficients $(\mathbf{h}(0), \mathbf{h}(1), \mathbf{h}(2), \dots)$:

$$\mathbf{F}_L^{-1} \mathbf{H}_L \mathbf{F}_L = \widehat{\mathbf{H}}_L = \text{diag}(\widehat{\mathbf{h}}(0), \widehat{\mathbf{h}}(1), \dots, \widehat{\mathbf{h}}(L-1)). \quad (8.6)$$

Proof. Every time we multiply $\mathbf{x} = (1, W^k, W^{2k}, \dots)$ by the shift matrix \mathbf{S} , we get one more power of $\lambda = \overline{W}^k$. A combination of powers \mathbf{S}^n produces the same combination of powers \overline{W}^{kn} :

$$\mathbf{H}_L \mathbf{x} = \left(\sum_0^N \mathbf{h}(n) \mathbf{S}^n \right) \mathbf{x} = \left(\sum_0^N \mathbf{h}(n) \overline{W}^{kn} \right) \mathbf{x} = \widehat{\mathbf{h}}(k) \mathbf{x}. \quad (8.7)$$

The eigenvalues of \mathbf{H}_L are exactly the components $\widehat{\mathbf{h}}(k)$ of the DFT:

$$\begin{aligned} \text{The eigenvalue for } \mathbf{x}_0 = (1, 1, \dots) &\text{ is } \widehat{\mathbf{h}}(0) + \widehat{\mathbf{h}}(1) + \dots = \widehat{\mathbf{h}}(0) \\ \text{The eigenvalue for } \mathbf{x}_k = (1, W^k, \dots) &\text{ is } \widehat{\mathbf{h}}(0) + \widehat{\mathbf{h}}(1) \overline{W}^k + \dots = \widehat{\mathbf{h}}(k). \end{aligned}$$

The matrix form $\mathbf{HF} = \mathbf{FH}$ just gives these L equations all at once.

Example 8.1. Choose $\mathbf{h}(0) = 4$, $\mathbf{h}(1) = 1$ and $\mathbf{h}(2) = 1$. Take $L = 3$:

$$\mathbf{H} = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 4 \end{bmatrix} \quad \text{has eigenvalues} \quad \begin{aligned} \widehat{\mathbf{h}}(0) &= 4 + 1 + 1 = 6 \\ \widehat{\mathbf{h}}(1) &= 4 + \overline{W} + \overline{W}^2 = 3 \\ \widehat{\mathbf{h}}(2) &= 4 + \overline{W}^2 + \overline{W}^4 = 3. \end{aligned}$$

The eigenvalues are real because \mathbf{H} is symmetric. The eigenvectors are orthogonal (of course). The first eigenvector is $(1, 1, 1)$. The other eigenvectors are $(1, e^{2\pi i/3}, e^{4\pi i/3})$ and its complex conjugate. We could change to real eigenvectors $(1, \cos \frac{2\pi}{3}, \cos \frac{4\pi}{3})$ and $(0, \sin \frac{2\pi}{3}, \sin \frac{4\pi}{3})$ if we wanted to (both with eigenvalue 3). This is the option in the real case of using sines and cosines. Everywhere we use $W^3 = 1$ and $1 + W + W^2 = 0$.

Example 8.2. Choose $\mathbf{h}(0) = 4$, $\mathbf{h}(1) = 1$, $\mathbf{h}(2) = 0$, $\mathbf{h}(3) = 1$ and $L = 4$:

$$\mathbf{H} = \begin{bmatrix} 4 & 1 & 0 & 1 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 1 & 0 & 1 & 4 \end{bmatrix} \quad \text{has eigenvalues} \quad \begin{aligned} \widehat{\mathbf{h}}(0) &= 4 + 1 + 0 + 1 = 6 \\ \widehat{\mathbf{h}}(1) &= 4 + i^3 + 0 + i = 4 \\ \widehat{\mathbf{h}}(2) &= 4 + i^2 + 0 + i^6 = 2 \\ \widehat{\mathbf{h}}(3) &= 4 + i + 0 + i^3 = 4. \end{aligned}$$

Again \mathbf{H} is symmetric and the eigenvalues are real. Note the eigenvector $(1, -1, 1, -1)$ corresponding to $\widehat{\mathbf{h}}(2) = 2$. The DFT matrix diagonalizes \mathbf{H}_L :

$$\begin{aligned} \text{For infinite signals } \mathbf{y} = \mathbf{Hx} &\text{ becomes } \mathbf{Y}(z) = \mathbf{H}(z)\mathbf{X}(z) \\ \text{For length } L \text{ signals } \mathbf{y} = \mathbf{Hx} &\text{ becomes } \widehat{\mathbf{y}} = \widehat{\mathbf{H}}\widehat{\mathbf{x}}. \end{aligned}$$

\mathbf{Hx} is a matrix-vector multiplication. $\widehat{\mathbf{H}}\widehat{\mathbf{x}}$ is a pointwise multiplication.

We now explain the finite Fourier transform $\widehat{\mathbf{x}}$ and circular convolution. The vector \mathbf{x} has length L and so does $\widehat{\mathbf{x}}$:

Definition 8.1 The Discrete Fourier Transform of x is $\hat{x} = \bar{F}_L x$. Its components are

$$\hat{x}(k) = \sum_{n=0}^{L-1} x(n) W^{nk} = \sum_{n=0}^{L-1} x(n) e^{-2\pi i nk/L}. \quad (8.8)$$

The index n corresponds to time and k corresponds to frequency:

$$\text{DFT}[x(n)] = X(k) = \hat{x}(k) \quad \text{IDFT}[\hat{x}(k)] = \text{IDFT}[X(k)] = x(n).$$

The inverse transform is $x = \bar{F}_L^{-1} \hat{x} = \frac{1}{L} F_L \hat{x}$. Its components are

$$x(n) = \frac{1}{L} \sum_{k=0}^{L-1} \hat{x}(k) e^{2\pi i nk/L}. \quad (8.9)$$

The DFT analyzes the signal into pure frequencies. The IDFT synthesizes it again. The exponent is $-2\pi i nk/L$ in (8.8) and $2\pi i nk/L$ in (8.9), just as the Fourier integral in continuous time has $e^{-i\omega t}$ and $e^{i\omega t}$. Do not forget why the complex conjugate gives the inverse! It is because the columns of $F_L = F_L^T$ are orthogonal. The inner products give a multiple of the identity matrix:

$$\bar{F}_L F_L = L I \quad \text{and} \quad F_L \bar{F}_L = L I \quad \text{and} \quad \bar{F}_L^{-1} = \frac{1}{L} F_L. \quad (8.10)$$

Dividing by L gives the inverse, like dividing by 2π in continuous time. The transform preserves energy (apart from this factor L). This is the discrete version of Parseval's theorem $\int |x(t)|^2 dt = \frac{1}{2\pi} \int |X(\omega)|^2 d\omega$:

$$\text{Discrete Parseval theorem: } \|x\|^2 = \frac{1}{L} \|\hat{x}\|^2. \quad (8.11)$$

This is $\sum |x(n)|^2 = \frac{1}{L} \sum |\hat{x}(k)|^2$. It is based on orthogonality of the columns:

$$(x, x) = (\frac{1}{L} F_L \hat{x}, \frac{1}{L} F_L \hat{x}) = \frac{1}{L} (\hat{x}, \hat{x}) \quad \text{because} \quad \bar{F}_L^T F_L = L I.$$

The Circular Convolution Rule

Diagonalization by the Fourier matrix is true for all circulants. Therefore any multiplication Hx can be done two ways. A direct matrix multiplication is a “circular convolution”. The indirect way is multiplication by F_L^{-1} , then the diagonal matrix D_L , then F_L . This is a transform to the frequency domain, where H is diagonalized. This **convolution rule** underlies the whole theory of filters: convolution in time \Leftrightarrow multiplication after DFT.

The circular filter $y = Hx$ becomes $\hat{y} = \hat{H}\hat{x}$ in frequency. This comes from the diagonalization $F^{-1}HF = \hat{H}$ and the fact that $\bar{F}\bar{F} = L I$:

$$\hat{y} = \hat{H}\hat{x} \quad \text{is} \quad \bar{F}y = (F^{-1}HF)\bar{F}x \quad \text{which is} \quad y = Hx.$$

In words, the diagonalization is a change of basis from time to frequency. In the frequency domain, the filter multiplies each $\hat{x}(k)$ by $h(k)$. We have agreement between a convolution Hx and a pointwise multiplication $\hat{H}\hat{x}$:

$$\begin{bmatrix} 4 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 4 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \end{bmatrix} \quad \text{transforms to} \quad \begin{bmatrix} 6 & & \\ & 3 & \\ & & 3 \end{bmatrix} \begin{bmatrix} \hat{x}(0) \\ \hat{x}(1) \\ \hat{x}(2) \end{bmatrix}.$$

For a specific example take

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \widehat{\mathbf{x}} = \begin{bmatrix} 1 & \frac{1}{W} & \frac{1}{W^2} \\ 1 & \frac{1}{W} & \frac{1}{W^2} \\ 1 & \frac{1}{W^2} & \frac{1}{W^4} \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix}.$$

Parseval's identity says that $2^2 + 1^2 + 1^2$ should be $\frac{1}{L}$ times $4^2 + 1^2 + 1^2$. (6 is $\frac{1}{3}$ of 18.) The matrix multiplication $H\mathbf{x}$ and the pointwise $\widehat{H}\widehat{\mathbf{x}}$ give

$$\mathbf{y} = H\mathbf{x} = \begin{bmatrix} 10 \\ 7 \\ 7 \end{bmatrix} \quad \text{and} \quad \widehat{\mathbf{y}} = \widehat{H}\widehat{\mathbf{x}} = \begin{bmatrix} 6 \cdot 4 \\ 3 \cdot 1 \\ 3 \cdot 1 \end{bmatrix} = \begin{bmatrix} 24 \\ 3 \\ 3 \end{bmatrix}.$$

Again Parseval's identity checks: $10^2 + 7^2 + 7^2 = 198$ is $\frac{1}{3}$ of $24^2 + 3^2 + 3^2$.

Finally we display $H\mathbf{x}$ as a *circular convolution*. One way is to extend \mathbf{x} periodically. Then ordinary convolution has periodic output 10, 7, 7 (repeated):

Periodic signal	2	1	1	2	1	1	2	1	1	2	1	1
Normal filter	1	1	4		1	1	4		1	1	4	
	<hr/>	10			<hr/>	7			<hr/>	7		

Another way is to think of convolution as multiplication $H(z)X(z)$. But in this circular case $z^3 = 1$ and $z^4 = z$. Therefore $H(z)$ times $X(z)$ is

$$(4+z+z^2)(2+z+z^2) = 8+6z+7z^2+2z^3+z^4 \quad (\text{ordinary}) \\ = 10+7z+7z^2 \quad (\text{wraparound}).$$

This is $Y(z)$. Therefore $\mathbf{y} = (10, 7, 7)$. We summarize the essential points:

The circular convolution of \mathbf{h} with $\mathbf{x} = (x(0), x(1), \dots)$ is $\mathbf{h} \circledast \mathbf{x} = H\mathbf{x}$.

All vectors have length L . The transform of $\mathbf{y} = \mathbf{h} \circledast \mathbf{x}$ is $\widehat{\mathbf{y}}(k) = \widehat{\mathbf{h}}(k)\widehat{\mathbf{x}}(k)$.

This convolution rule replaces the matrix multiplication $H\mathbf{x}$ by the pointwise $\widehat{H}\widehat{\mathbf{x}}$. We have to transform \mathbf{h} and \mathbf{x} to the Fourier domain! And then transform $\widehat{\mathbf{y}}$ back to \mathbf{y} . So three DFT's and the L multiplications $\widehat{\mathbf{h}}(k)\widehat{\mathbf{x}}(k)$ replace one circular convolution — which is multiplication by H :

$$\begin{aligned} \mathbf{h} &= \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix} \rightarrow \widehat{\mathbf{h}} = \begin{bmatrix} 6 \\ 3 \\ 3 \end{bmatrix} & \searrow \\ \mathbf{x} &= \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \rightarrow \widehat{\mathbf{x}} = \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix} & \nearrow \quad \widehat{\mathbf{y}} = \widehat{\mathbf{h}}\widehat{\mathbf{x}} = \begin{bmatrix} 24 \\ 3 \\ 3 \end{bmatrix} \rightarrow \mathbf{y} = \begin{bmatrix} 10 \\ 7 \\ 7 \end{bmatrix} \end{aligned}$$

In the time domain $H\mathbf{x}$ needs about NL multiplications. The N filter coefficients are used L times each. Compare with three DFT steps, which are executed by the *Fast Fourier Transform*. Each FFT requires only $\frac{1}{2}L \log_2 L$ individual steps. Transforming is worthwhile when $N > \frac{3}{2} \log_2 L$.

In matrix notation, F is the product of $\log_2 L$ very sparse matrices, involving only $L/2$ multiplications each. This FFT recursion connects the L -point transform to two copies of the $\frac{L}{2}$ -point

transform. It produces a matrix that is *half zero* and two very simple matrices around it. Here is the key to the FFT when $L = 1024$:

$$\text{FFT factors : } \mathbf{F}_{1024} = \begin{bmatrix} \mathbf{I}_{512} & -\mathbf{D}_{512} \\ \mathbf{I}_{512} & -\mathbf{D}_{512} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{512} & \\ & \mathbf{F}_{512} \end{bmatrix} \begin{bmatrix} \text{even} \\ \text{odd} \end{bmatrix}. \quad (8.12)$$

\mathbf{I}_{512} is the identity matrix. \mathbf{D}_{512} is the diagonal matrix with entries $(1, W, \dots, W^{511})$. The key is the two copies of \mathbf{F}_{512} (which use W^2 , the 512th root of 1). The permutation at the end separates the incoming vector into its even part $(\downarrow 2)\mathbf{x} = (x(0), x(2), \dots)$ and its odd part $(x(1), x(3), \dots)$.

This reduction from \mathbf{F}_L to two copies of $\mathbf{F}_{L/2}$ almost cuts the work in half. We only need an extra 512 multiplications for \mathbf{D}_{512} , plus additions and permutations. The full FFT algorithm keeps going, recursively. Each Fourier matrix \mathbf{F}_{512} is replaced by two copies of \mathbf{F}_{256} , with diagonal matrices \mathbf{D}_{256} and a permutation that starts with $(\downarrow 4)$. Since 1024 is 2^{10} , it takes only 10 steps to go from \mathbf{F}_{1024} to \mathbf{F}_1 ! Maybe we don't go that far, but we could:

\mathbf{F}_{1024} is the product of 10 matrices with \mathbf{F} 's and \mathbf{D} 's and a permutation.

The $10 = \log_2 L$ matrices each require $512 = \frac{1}{2}L$ multiplications for the \mathbf{D} 's. So the total count for the FFT is $\frac{1}{2}L \log_2 L$.

This simple idea changes $(1024)^2$ multiplications for the full \mathbf{F}_{1024} into 5(1024) multiplications for its ten sparse factors. That is a savings ratio of 200. Very rarely, perhaps never, has a single algorithm produced such a revolution as the FFT.

Problem Set 8.1

1. Find the determinant of $S_4 - \lambda I$ and the eigenvalues of S_4 :

$$\det(S_4 - \lambda I) = \det \begin{bmatrix} -\lambda & 0 & 0 & 1 \\ 1 & -\lambda & 0 & 0 \\ 0 & 1 & -\lambda & 0 \\ 0 & 0 & 1 & -\lambda \end{bmatrix}.$$

2. Each term in an L by L determinant has one factor from each row and column. Why are $(-\lambda)^L$ and 1^L the only possibilities in the determinant of $S_L - \lambda I$? Why not products of $-\lambda$ with 1?
3. If $\mathbf{h}(0), \dots, \mathbf{h}(3)$ is the first column of the circulant matrix \mathbf{H} and $\mathbf{g}(0), \dots, \mathbf{g}(3)$ is the first column of the circulant matrix \mathbf{G} , show that the product \mathbf{GH} is a circulant matrix with first column $\mathbf{g} \circledast \mathbf{h}$. One approach is to diagonalize $\mathbf{H} = \mathbf{F}\widehat{\mathbf{H}}\mathbf{F}^{-1}$ and $\mathbf{G} = \mathbf{F}\widehat{\mathbf{G}}\mathbf{F}^{-1}$.
4. Find the eigenvalues and eigenvectors of

$$\mathbf{H}_3 = \begin{bmatrix} 6 & -1 & 1 \\ 1 & 6 & -1 \\ -1 & 1 & 6 \end{bmatrix} \quad \text{and} \quad \mathbf{H}_4 = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{bmatrix}.$$

5. If $\mathbf{x} = (1, 1, 1)$ then $\mathbf{H}_3\mathbf{x} = (6, 6, 6)$. Find that answer from the convolution rule using $\widehat{\mathbf{h}}(0)\widehat{\mathbf{x}}(0)$ and $\widehat{\mathbf{h}}(1)\widehat{\mathbf{x}}(1)$ and $\widehat{\mathbf{h}}(2)\widehat{\mathbf{x}}(2)$.
6. Suppose the decimation operator $(\downarrow M)$ is applied to a periodized signal of length L . How many independent samples do we get if M divides L ? If this occurs in each of M channels, find the total number of subsamples. Why is M required to divide L ?

7. On the interval $0 \leq t \leq 1$, draw the four Haar wavelets and the four (periodized) Daubechies D_4 wavelets at scale level $\frac{1}{4}$. These are the basis functions for the 4-dimensional space W_2 .
8. When is a circulant matrix invertible? The entries in its first column are $h(0), \dots, h(L-1)$. The transform of that column vector is $(\widehat{h}(0), \dots, \widehat{h}(L-1))$. The answer is in the transform, since H becomes a multiplication.
9. Write out explicitly the factorization (8.12) for the Fourier matrix F_4 .

8.2 Symmetric Extension for Symmetric Filters

Now we come to the heart of the finite length problem. A signal of length L will be extended in a symmetric way. The extended signal is Ex and the filter produces HEx . If the filter is symmetric (or antisymmetric), then the output HEx will be symmetric (or antisymmetric). Downsampling yields $(\downarrow 2)HEx$. *The extension must be chosen so that this subsample is symmetric.* Then we restrict attention to a piece of that subband signal, of length approximately $L/2$. (The exact integer will be crucial.) This restriction R yields the output from that channel:

$$R(\downarrow 2)HEx = (\text{restriction})(\downarrow 2)(\text{filter})(\text{extension of } x).$$

To repeat: A linear phase signal convolved with a linear phase filter has a linear phase output. You can see this in the time domain by simple algebra. In the frequency domain, the linear exponents in the phase add and stay linear. The key question is whether the *subsample* has linear phase, and what phase it has. We need to know the *point of symmetry* at every step of $R(\downarrow 2)HEx$.

The advantage of symmetric extension is this. *We may introduce a corner but we don't introduce a jump.* Wraparound creates a jump because generally $x(L)$ does not equal $x(0)$. With the extension, $x(2L)$ does equal $x(0)$. So you should extend before wraparound.

The Discrete Cosine Transform (DCT) is an example of symmetric extension. But there is no filter. It extends a block at a time, period. The DFT of the double block has only cosine terms, by symmetry. The DCT of Type II is the restriction back to the original block.

The DCT of Type IV extends even further, to period $4L$ instead of $2L$. After computing the DFT of a full period, we always restrict back to L coefficients. (More would be redundant, since we started with L samples.) Section 8.3 will describe both types of DCT. The present section is about extension and restriction of ordinary signals, before and after ordinary filtering.

Symmetric Extension and Symmetric Subsamples

One decision becomes crucial for multirate filter banks. Do we repeat the first and last samples, or do we not repeat? The results after the sampling operator $(\downarrow 2)$ are very different. We face a question that does not arise for the DCT. *Is the subsampled signal still symmetric?* That requirement will govern our extension of the input signal. We will take up in order the key decisions and the reasons behind them:

1. When to repeat the first and last samples (depending on N).
2. Where to start the subsampling (depending on L).
3. How many subsamples to keep from each channel.

The input is a set of L numbers $x(0), \dots, x(L-1)$. The output is to be a total of L samples from the lowpass and highpass channels. We must choose a method that gives exactly L nonredundant outputs. Then we have a nonexpansive transform (a square matrix).

Repeating (H Extension) or Not Repeating (W Extension)

There are two symmetric ways to extend a signal that starts with $x(0)$. Everything depends on the choice of $x(-1)$. We may take $x(-1) = x(1)$, and then continue with $x(-2) = x(2)$. The “point of symmetry” is $t = 0$. When we do this at both ends, $t = 0$ and $t = L-1$, the period is $2L-2$. This is called ***whole-point symmetry***, and it is indicated by the letter **W**—not in italic.

This **W** extension corresponds to $x(-t) = x(t)$ in continuous time. In discrete time, it means that the number $x(0)$ is not repeated. At the other end, $x(L-1)$ also appears only once. All other samples $x(n)$ reappear as $x(-n)$.

The other possibility is to choose $x(-1) = x(0)$. The value at $t = 0$ is repeated at $t = -1$. The point of symmetry is halfway between, at $t = -\frac{1}{2}$. The extension continues with $x(-2) = x(1)$. Figure 8.3 shows how this ***half-point symmetry*** produces a signal that has period $2L$. It is sometimes referred to as a $(2, 2)$ extension, to indicate the repetition at both ends. The extension is symmetric about $t = -\frac{1}{2}$ and $t = L-\frac{1}{2}$ and $t = 2L-\frac{1}{2}$. This half-point symmetry is indicated by the non-italic letter **H**.

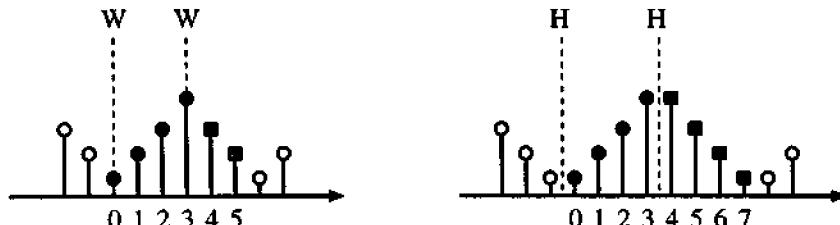


Figure 8.3: Whole-point symmetry (**W**) and half-point symmetry (**H**). The input has $L = 4$. The period is $2(L-1) = 6$ or $2L = 8$.

Which extension to choose, W or H? The answer depends on the filter length. *The right choice depends on whether the filter itself is W or H.* A symmetric filter with an odd number of coefficients, say $h(0), h(1), h(0)$, is a **W** filter. The point of symmetry is the center point 1. A symmetric filter with an even number of coefficients, say $h(0), h(1), h(1), h(0)$, is an **H** filter. It is symmetric about the half-point $\frac{3}{2}$. The **H** filter has repetition, $h(1) = h(2)$, and the **W** filter has whole-point symmetry with no repetition of the center coefficient.

The goal is to have a symmetric extension after downsampling. Then restriction step will succeed. The subsampled signal $(\downarrow 2)y = (\downarrow 2)\mathbf{H}\mathbf{E}x$ is always periodic, but it is symmetric only if we follow the rule:

Use a W (no repeat) extension for a W (odd length) filter.

Use an H (repeat) extension for an H (even length) filter.

Those rules both give a W extension for $\mathbf{H}\mathbf{E}x$. Downsampling preserves symmetry.

A mixture of **H** and **W** would give an **H** extension for $y = \mathbf{H}\mathbf{E}x$. Downsampling an **H** extension goes wrong! Try it for $y(2), y(1), y(0), y(0), y(1), y(2)$.

The clearest way to justify this rule is to try to break it. Figure 8.4 shows the filtered outputs $y = HEx$ when the extension and filter are of opposite types (one is W, the other is H). Downsampling y will not produce a symmetric signal. Then the restriction of $(\downarrow 2)y$ will fail. We show only failure here because there are many variations of success still to consider. The signal length L is going to enter soon (and $L = 4$ does not show all variations of failure either).

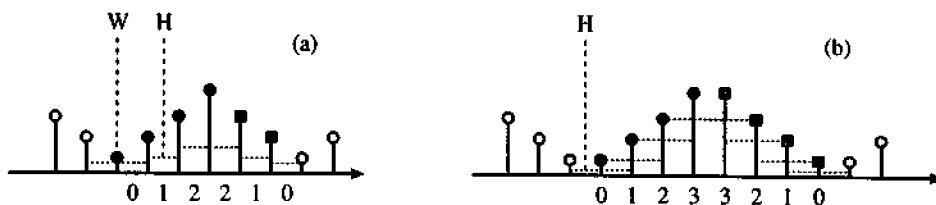


Figure 8.4: H and W do not mix. (a) W extension and H filter (two taps) give H extension. Downsampling yields 0, 2, 1 repeated. (b) H extension and W filter (three taps) give H extension. Downsampling yields 0, 2, 3, 1 repeated. *These subsamples are not symmetric and restriction fails.*

Length of Signal and Center of Filter

W extension will be chosen for W filters (odd length, symmetric or antisymmetric). H extension will be used for H filters. The next question is where to “center” the filter, and that depends on the signal length L . The goal is a total of L nonredundant samples from the two channels.

We do not study a (1, 2) or (2, 1) extension, with one endpoint repeated, because the period is then odd. This cannot be used with $(\downarrow 2)$. For an M -channel filter bank with odd M , new possibilities appear: W can mix with H. A complete table of admissible extensions is in the valuable paper by [Brislawn], which has been our guide. Earlier references are [SmEd, KiYaIw].

Start with the W extension (no repeat of end values). The filter length is odd, and both analysis filters (lowpass and highpass) are symmetric. The extended signal HEx has period $2(L - 1)$. Figure 8.5 shows $L = 4$ on the left with an extension of period 6. The filtered values $y = HEx$ also have period 6. Those y 's display a W extension (no repeats). Downsampling gives a symmetric extension. (It has to be (1, 2) because the half-period $L - 1$ is odd.) Then restriction produces $L/2 = 2$ independent samples from each channel.

On the right side of Figure 8.5 is an odd-length signal ($L = 5$). The filter has odd length 3. So a W extension of x leads to a W extension for y , after filtering. There are 5 independent y 's. By correct centering (in other words, correct phase) we get $\frac{1}{2}(L + 1)$ subsamples from the lowpass channel and $\frac{1}{2}(L - 1)$ subsamples from the highpass channel. The analysis bank is nonexpansive, producing L outputs.

In the time domain this analysis bank is expressed by an $L \times L$ matrix. Then the synthesis bank (from biorthogonal filters) is expressed by the inverse matrix.

Now we go to an H extension (with repeat) for an H filter (even length). The extension has period $2L$. The lowpass filter will be symmetric and the highpass filter will be antisymmetric. For simplicity we use the average – difference pair with coefficients $\frac{1}{2}, \frac{1}{2}$ and $\frac{1}{2}, -\frac{1}{2}$. We still aim for L nonredundant subband samples.

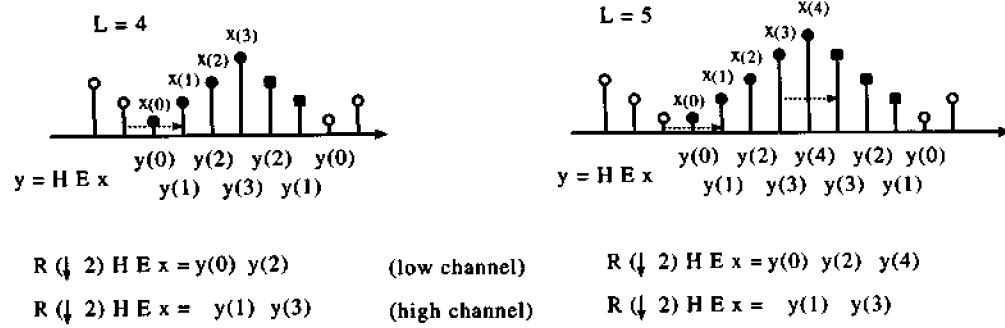


Figure 8.5: W extension and W filter yield $y = HEx$ with a W extension. The samples $(\downarrow 2)y$ are symmetric and the restriction R succeeds.

On the left of Figure 8.6 is the case of even length $L = 4$ (four solid bullets). The extended period is 8. The lowpass filter produces y 's and the antisymmetric highpass filter produces z 's. (This combination is required for biorthogonality with even length.) Now the centering makes a difference. We can get subsamples of the original size from downsampling:

$$R(\downarrow 2) HEx = y(0), y(2), y(4) \text{ and } z(2) \quad (\text{total } L = 4)$$

Or, as we prefer, a change of center gives equal size samples from the two channels:

$$R(\downarrow 2) HEx = y(1), y(3) \text{ and } z(1), z(3) \quad (\text{total } L = 4)$$

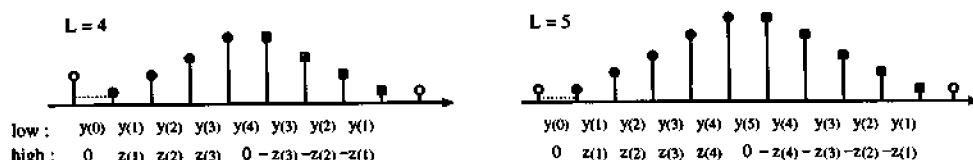


Figure 8.6: Even-length H filters with H extension. The filtered and subsampled signals have L independent samples: $L/2$ from each channel on the left, $(L+1)/2$ and $(L-1)/2$ on the right.

On the right of Figure 8.6 is a signal with odd length $L = 5$. The lowpass filter gives 6 independent y 's. The highpass filter gives 4 independent z 's. There will be 3 and 2 good subsamples. It is natural to keep the same centering for the two channels. We may select $y(0), y(2), y(4)$ and $z(2), z(4)$. Or select $y(1), y(3), y(5)$ and $z(1), z(3)$. In all cases L independent subsamples are returned by the analysis bank.

Problem Set 8.2

1. Show that the symmetric extension of $f(t)$ from $[0, \pi]$ to $[-\pi, 0]$ gives a function $Ef(t)$ that has no sine terms: $\int_{-\pi}^{\pi} Ef(t) \sin kt dt = 0$. Give a formula for $Ef(t)$.
2. Extend a function $x(t)$ to have period L (wraparound instead of symmetric extension). Draw the output from the centered filter $h(-1) = h(1) = \frac{1}{2}$. Rescale to $2t$ (period $L/2$) and draw the output $v(t)$.

3. Summarize the reason for choosing **H** extensions for **H** filters and **W** extensions for **W** filters:
*We want the filtered values $y = HEx$ to be **W**. If the filtered values are **H**, why does symmetry fail for $(\downarrow 2)y$?*
4. What is the difference in the z -domain between a **W**-extended signal and an **H**-extended signal? One of the extensions has $X(z) = X(z^{-1})$ and the other doesn't.
5. Draw the **W** and **H** extensions of a signal x of length $L = 6$. Apply a four-coefficient filter to each, and indicate the full period of the filtered signal y . Which yields a satisfactory subsample $(\downarrow 2)y$?
6. Draw the **W** and **H** extensions of a signal x of length $L = 7$. Apply a symmetric 5-coefficient filter and indicate a full period of the filtered signal y . Which subsamples would you choose?
7. Suppose the decimation operator is $(\downarrow 3)$ instead of $(\downarrow 2)$. How many independent samples if x has length L , with **W** extension and then with **H** extension? The main requirement is that the period $(2L$ or $2L - 2$ or even $2L - 1)$ should be divisible by M : *Why?*
8. Show that $Ex(t) = \sum_{-\infty}^{\infty} [f(t - 2n) + f(2n - t)]$ gives the symmetric extension with period 2 of a function $x(t)$ defined on $[0, 1]$.

8.3 Cosine Bases and the DCT

The Discrete Cosine Transform (DCT) improves on the DFT for the same reason that symmetric extension improves on periodic extension. *The symmetric extension is continuous.* The periodic extension generally has a jump. This section develops the DCT as an extension followed by the DFT of size $2L$ followed by a restriction back to length L . There are *four types* of DCT coming from four types of extension.

We start in continuous time, by extending $f(t)$ from $[0, \pi]$ to $[-\pi, \pi]$. The symmetric extension is $Ef(t) = f(-t)$ on $[-\pi, 0]$ and then 2π -periodic. This even function $Ef(t)$ is a combination of cosines:

$$Ef(t) = \sum_0^{\infty} a_k \cos kt.$$

The discrete analogue is the DCT of Type I or II. There is a choice between **W** extension and **H** extension! We may define $x(-1)$ to equal $x(1)$ or to equal $x(0)$. This choice doesn't arise in continuous time, but the previous section showed its importance in discrete time.

There is another extension in continuous time. It is surprising and very important. We call it E^{IV} because it leads to the DCT of Type IV. Start again with $f(t)$ on $[0, \pi]$. The extension is *even* across the left endpoint $t = 0$, as before, but it is *odd* across the right endpoint $t = \pi$. The extended (unfolded) function $E^{IV}f(t)$ is drawn in Figure 8.7. It has period 4π . There is a jump at $t = \pi$ where the extension is odd. From the left side the graph approaches $f(\pi)$, from the right side the graph approaches $-f(\pi)$. This even-odd extension E^{IV} is remarkably useful.

Notice that $E^{IV}f(t)$ involves cosines of period 4π (they are $\cos \frac{kt}{2}$). But it requires only *half of those cosines*, because the even frequencies $\ell = 2k$ have zero coefficients. These even frequencies yield $\cos kt$, which is an even function around $t = \pi$. The extension is an *odd* function around $t = \pi$, so the integral of $\cos kt$ times $E^{IV}f(t)$ over a period is zero. We are left with the odd frequencies $\ell = 2k + 1$ and the basis functions $\cos \frac{kt}{2} = \cos(k + \frac{1}{2})t$:

$$E^{IV}f(t) = \sum_0^{\infty} a_k \cos \left(k + \frac{1}{2} \right) t. \quad (8.13)$$

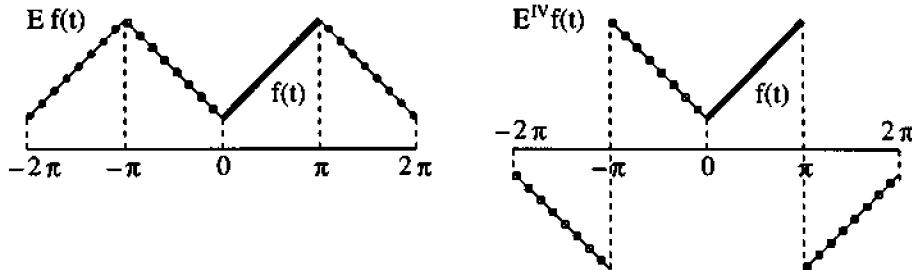


Figure 8.7: The symmetric extension $E f(t)$ is continuous with period 2π . The even-odd extension $E^{IV} f(t)$ is discontinuous with period 4π .

Discrete Extension and the DCT-II

The function $f(t)$ on $[0, \pi]$ is now replaced by a vector $(x(0), \dots, x(L-1))$. We will extend that vector and apply the DFT (the Fourier matrix F_{2L}) of doubled length $2L$. **The whole-sample W extension is not satisfactory.** By symmetry it creates $x(-n) = x(n)$, extending the definition to $|n| \leq L-1$. This is $2L-1$ samples, and we do not want an odd number. It is better to use the half-sample H extension, which repeats $x(0)$ as $x(-1)$. The point of symmetry is $-\frac{1}{2}$, but this inconvenience is small. The symmetric H extension has period $2L$, and we call it E^H :

$$E^H x(-n) = x(n-1) \quad \text{for } n = 1, \dots, L. \quad (8.14)$$

The Fourier expansion of $E^H x$ uses an ordinary DFT of length $2L$. That has $2L$ basis functions indexed by frequency k . The n th component of the k th basis vector is normally $e^{2\pi i k n / 2L}$. But we shift n to $n + \frac{1}{2}$ in order to move the point of symmetry. The shifted signal is even around zero, and can be expressed by L cosines. The shift brings out a factor $e^{i\pi k / 2L}$, and the expansion of the $2L$ -periodic signal is

$$E^H x(n) = \sum_{k=0}^{L-1} e^{i\pi k / 2L} \hat{x}^H(k) \cos \frac{\pi k (n + \frac{1}{2})}{L}. \quad (8.15)$$

Orthogonality of the exponentials leads to orthogonality of the cosines (not trivial to see). The modulation factors $e^{i\pi k / 2L}$ of absolute value 1 just multiply the separate basis functions (indexed by k).

The $L \times L$ DCT-II matrix in this cosine expansion has entries

$$C^H(k, n) = b(k) \sqrt{\frac{2}{L}} \cos \frac{\pi k (n + \frac{1}{2})}{L}.$$

The square root of $\frac{2}{L}$ is included to make the rows of C^H into unit vectors. So is the factor

$$b(k) = \begin{cases} 1/\sqrt{2} & \text{if } k = 0 \\ 1 & \text{if } k = 1, \dots, L-1. \end{cases}$$

This is the familiar factor for the zero frequency $k = 0$ when the cosine stays at 1.

The key points about C^H are its *orthogonality* and *fast execution*. Those are proved by the factorization (8.18) below. They follow from the connection to the DFT (and therefore the FFT).

Even–Odd Extension and the DCT-IV

The even–odd extension E^{IV} was applied to $f(t)$ in continuous time. Its discrete analogue leads to the DCT matrix of Type IV. This is the L by L matrix E^{IV} , with remarkable properties.

Again the original signal has L components $x(n)$. This is extended evenly across $n = -\frac{L}{2}$, with \mathbf{H} symmetry, to a signal with $2L$ components. *Then comes an antisymmetric \mathbf{H} extension to $4L$ components.* Figure 8.8 shows the basic period of $4L$, which repeats.

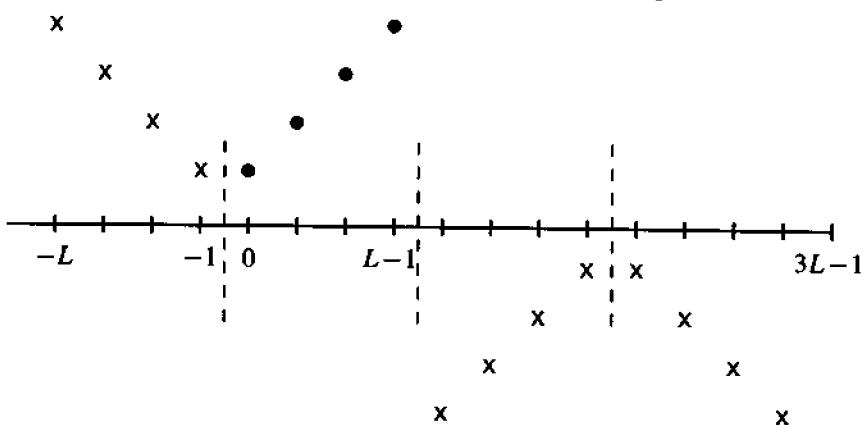


Figure 8.8: $x^{IV}(n)$ is even across $n = -\frac{L}{2}$ and odd across $n = L - \frac{1}{2}$. Here $L = 4$.

The basis functions for $E^{IV} f(t)$ in continuous time were $\cos(k + \frac{1}{2})t$. Those came from odd frequencies $\ell = 2k+1$ and from period 4π . The basis functions in discrete time have period $4L$ and the same odd frequencies $\ell = 2k+1$. There is also the shift from n to $n + \frac{1}{2}$, which moves the point of symmetry to zero and the point of antisymmetry to L . Then the basis functions appear in the $L \times L$ DCT-IV matrix

$$C^{IV}(k, n) = \sqrt{\frac{2}{L}} \cos \frac{\pi(k + \frac{1}{2})(n + \frac{1}{2})}{L}. \quad (8.16)$$

Notice the good properties of this matrix. It is symmetric. It needs no $b(k)$ to insert $\sqrt{2}$ into a zero-frequency term. Its columns are the even–odd basis vectors, and Figure 8.9 shows the first four vectors in the C^{IV} basis of length $L = 32$. The DCT matrix also has *orthogonality* and *fast execution*, because of its direct link to the DFT matrix of order $2L$ — which we now explain.

Matrix Factorizations and Orthogonality: DCT-II and DCT-IV

By slightly adapting Wickerhauser's very neat exposition [W, pp. 90–96], we can achieve three useful purposes at the same time:

1. Include the discrete *sine* transforms: DST-II and DST-IV
2. Prove the *orthogonality* of all four transforms: DCT and DST, II and IV
3. Show explicitly how these four transforms connect by extremely sparse matrices to the DFT matrix F_{2L} .

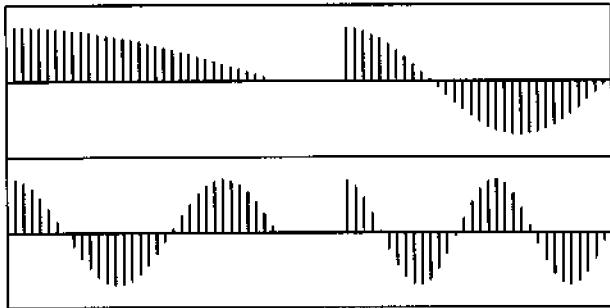


Figure 8.9: The first four basis functions of the DCT-IV, for $M = 32$.

These results come directly from two matrix factorizations, which we will display but not prove. A MATLAB exercise verifies their correctness for each L . You will notice that the DCT-IV and DST-IV formulas are cleaner, except for the multiplier $e^{-\pi i / 4L}$ from the shift by $\frac{1}{2}$. Type II must assign $1/\sqrt{2}$ to the zero frequency DC term. Here are the formulas that prove everything.

$$\text{Factorization IV: } e^{-\pi i / 4 L} \mathbf{R}^T \mathbf{F}_{2L} \mathbf{R} = \begin{bmatrix} \mathbf{C}^{IV} & \\ & -i \mathbf{S}^{IV} \end{bmatrix} \quad (8.17)$$

$$\text{Factorization II: } \bar{\mathbf{P}}^T \mathbf{F}_{2L} \mathbf{Q} = \begin{bmatrix} \mathbf{C}^{\text{II}} & \\ & \mathbf{S}^{\text{II}} \end{bmatrix} \quad (8.18)$$

All matrices on the left are square and orthogonal of size $2L$. Then each block on the right is square and orthogonal of size L . The sine and cosine transforms preserve length (discrete Parseval identity as for the DFT). Note that R is only transposed while P is also conjugated.

It remains to display the sparse matrices, with at most two entries in each column and row. J is the reverse identity, with 1's on the antidiagonal. The combination of I and J gives the Gray ordering in (8.21). The matrices R and P clearly have orthogonal columns, and the division by $\sqrt{2}$ yields unit vectors:

$$R = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & & & & \\ & \bar{W} & & & \\ & & \ddots & & \\ & & & \bar{W}^{L-1} & \\ & & & W^L & \\ & & & & \bar{W}^{L-1} \\ & & & & -W^L \\ & & & & \\ & W^2 & & & -W^2 \\ & & & -W & \\ & & & & W \end{bmatrix}, \quad W = e^{\pi i / 2L}$$

$$P = \frac{1}{\sqrt{2}} \begin{bmatrix} \sqrt{2} & 0 \\ W & \bar{W} \\ & \ddots & \ddots & \ddots \\ & & W^{L-1} & 0 \\ & & 0 & \bar{W}^{L-1} \\ & & \bar{W}^{L-1} & -\bar{W}^{L-1} \\ 0 & \bar{W} & -\bar{W} \end{bmatrix}, \quad Q = \frac{1}{\sqrt{2}} \begin{bmatrix} I & I \\ J & -J \end{bmatrix}.$$

Summary: We now have two discrete cosine transforms, II and IV. There are two further types, I and III, from W extensions. Those are definitely less useful. If we omit the $\sqrt{2/L}$ normalization for simplicity, the DCT-II and DCT-IV transform a length L input signal into vectors X^{II} and X^{IV} :

$$X^{II}(k) = b(k) \sum_{n=0}^{L-1} x(n) \cos \frac{(n + \frac{1}{2}) k \pi}{L} \quad (8.19)$$

$$X^{IV}(k) = \sum_{n=0}^{L-1} x(n) \cos \frac{(n + \frac{1}{2})(k + \frac{1}{2}) \pi}{L}. \quad (8.20)$$

Malvar describes the DCT-II as better for transform coding. The DCT-IV is good for spectrum estimation and adaptive filtering. For us the key point is the efficiency of the algorithms.

Several implementations are possible and we describe them briefly, for even L . First we connect each DCT to the DFT. Then we connect DCT-II to DCT-IV.

DCT-II via DFT. The trick is in this reordering of $x(n)$ (the Gray order):

$$y(n) = x(2n) \text{ and } y(L-n-1) = x(2n+1) \text{ for } n = 0, 1, \dots, \frac{L}{2}-1. \quad (8.21)$$

Now take the DFT of $y(n)$. The DCT-II coefficients are

$$X^{II}(k) = \cos \frac{\pi k}{2L} \operatorname{Re}[\hat{y}(k)] - \sin \frac{\pi k}{2L} \operatorname{Im}[\hat{y}(k)]. \quad (8.22)$$

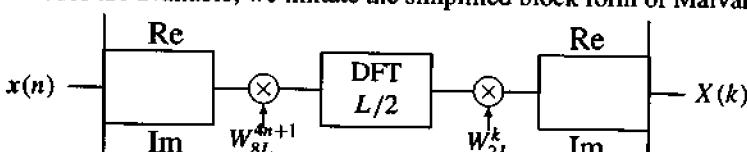
The numbers $X^{II}(k)$ and $X^{II}(L-k)$ come from a plane rotation by $\frac{\pi k}{2L}$. Each step (permutation, DFT, rotation) preserves vector length. Again, an *orthogonal matrix*.

DCT-IV via DFT. The same reordering and modulation will create $\frac{L}{2}$ complex numbers

$$c(n) = (x(2n) + ix(L-1-2n)) e^{-i(n+\frac{1}{4})\pi/L}. \quad (8.23)$$

Now follows the half-length DFT. The k th output is multiplied by $e^{-ik\pi/L}$. Then the real and imaginary parts of these $\frac{L}{2}$ complex numbers give the DCT-IV coefficients $X^{IV}(k)$.

Since codes are available, we imitate the simplified block form of Malvar:



The algebra behind this algorithm is expressed in two lines, which require superhuman patience. The first line separates even and odd components $x(n)$:

$$\begin{aligned} X^{IV}(k) = & \sum_{n=0}^{\frac{L}{2}-1} \left[x(2n) \cos \frac{(2n + \frac{1}{2})(k + \frac{1}{2})\pi}{L} \right. \\ & \left. + x(L-1-2n) \cos \frac{(L-1-2n + \frac{1}{2})(k + \frac{1}{2})\pi}{L} \right]. \end{aligned}$$

Replace cosines by exponentials and x 's by c 's. Then separate into the real and imaginary parts of an $\frac{L}{2}$ -point DFT times a modulation, for a fast DCT-IV:

$$\frac{X^{\text{IV}}(2k)}{X^{\text{IV}}(L-1-2k)} = \frac{\text{Re}}{-\text{Im}} \left[e^{-ik\pi/L} \sum_{n=0}^{\frac{L}{2}-1} c(n) e^{-4in\pi k n/L} \right] \quad (8.24)$$

DCT-II via DCT-IV. Finally we show that X^{II} of length L is immediately available from a DCT-II and a DCT-IV of length $\frac{L}{2}$. Recursively, this produces X^{II} from a sequence of DCT-IV's — whose fast computation was just given. The reduction of X^{II} comes by rewriting (8.19) in the form II plus IV:

$$\begin{aligned} X^{\text{II}}(k) &= b(k) \sum_{n=0}^{\frac{L}{2}-1} [x(n) + x(L-1-n)] \cos \frac{(n+\frac{1}{2})k\pi}{L/2} \\ &\quad + \sum_{n=0}^{\frac{L}{2}-1} [x(n) - x(L-1-n)] \cos \frac{(n+\frac{1}{2})(k+\frac{1}{2})\pi}{L/2}. \end{aligned} \quad (8.25)$$

This recursion to a shorter DCT-II and DCT-IV was known early. It requires only human patience to verify. When a fast DCT-IV was not available, the connection was useless. Now the DCT-IV algorithm gives a unified approach to the two most important versions of the discrete cosine transform.

In closing we emphasize two points. First, orthogonality of the DCT comes from extension followed by DFT followed by restriction. Second, there are still blocking effects from the limited smoothness. Symmetric extension is a big improvement on periodicity. The next section shows how to smooth the basis functions much more.

Problem Set 8.3

1. Find the DCT-II and DCT-IV transforms of the signals $x = (1, 1, 1, 1)$ and $x = (1, -1, 1, -1)$.
2. Find the DCT-II and DCT-IV transforms of the signals $x = (1, 0, 0, 0)$ and $x = (0, 1, 0, 0)$.
3. Show that the Gray reordering $y(n)$ in equation (8.21) transforms (8.19) into

$$x^{\text{II}}(k) = b(k) \sum_{n=0}^{L-1} y(n) \cos \frac{\pi(4n+1)k}{2L}.$$

4. Write out explicitly the 2×2 and 3×3 DCT-II matrices and show that they are orthogonal.
5. The DCT-IV matrix has entries $C_{kn} = \cos[\frac{\pi}{L}(k+\frac{1}{2})(n+\frac{1}{2})]$. Write down this symmetric $L \times L$ matrix for $L = 2$ and 3. Verify that columns 0 and 1 are orthogonal.
6. Execute this MATLAB verification of (8.17–8.18), connecting the DCT-DST-DFT:

```

L=16; i=eye(L); k=[0:L-1]; n=k';
dctiv=sqrt(2/L)*cos((k+1/2)*(n+1/2)*pi/L); % orthogonal dctiv matrix
dctii=sqrt(2/L)*cos(k*(n+1/2)*pi/L);
dctii(1,:)=sqrt(1/2)*dctii(1,:);
dft=sqrt(1/(2*L))*fft(eye(2*L)); % orthogonal dft matrix

Q=sqrt(1/2)*[i,i(L:-1:1,:)];
W=exp(sqrt(-1)*pi/(2*L)); Wb=conj(W); % W=2L-th root of one

```

```

p1=diag([sqrt(2)*W.^ (1:L-1)]); % P matrix
p2=diag([Wb.^ (1:L-1)]);
P=sqrt(1/2)*[p1;zeros(1,L);[zeros(L-1,1) p2(L-1:-1:1,:)];
c=P'*dft*Q; % dft to dctii mapping

r1=diag(Wb.^ (0:L-1)); r2=diag(W.^ (1:L));
R=sqrt(1/2)*[r1,r2(L:-1:1,:)]; % R matrix
z=exp(-sqrt(-1)*pi/(4*L)); % z=conjugate of 4L-th root of one
d=real(z*conj(R)/*dft*R); % dft to dactiv
% conj(x') gives x' without conjugate!
max(max(abs(c-dctii))) % peak error, should be VERY small
max(max(abs(d-dctiv))) % peak error, should be VERY small

```

8.4 Smooth Local Cosine Bases

To “localize” a cosine, just multiply it by a real window function $g(t)$. Windowing is the central idea of the Short Time Fourier Transform (STFT). When we use $e^{-i\omega t}$ rather than cosines, this is the representation pioneered by Gabor:

$$\text{The windowed transform of } f(t) \text{ is } F(\omega, a) = \int_{-\infty}^{\infty} f(t)g(t-a)e^{-i\omega t} dt. \quad (8.26)$$

Note the two variables ω = oscillation frequency and a = window position. We have a *time-frequency plane*, in the same way that k and j for the wavelet $w(2^j t - k)$ place us on a *time-scale plane*. With all frequencies ω and all positions a , we expect redundancy in $F(\omega, a)$ and cannot expect orthogonality. Nevertheless $f(t)$ can be recovered:

$$\text{The inverse windowed transform is } f(t) = C_g \iint F(\omega, a)g(t-a)e^{i\omega t} dt da.$$

The constant C_g depends on the window. This STFT is competition for the continuous wavelet transform. Both are described in Section 2.6.

With discrete frequencies and positions, we have an oscillation (a cosine) inside a modulating envelope (the window). When the window is a simple box, each block is independent. A compressed image looks as if it is built of tiles. This blocking effect is reduced by filtering and by smooth windows, but ringing often persists and the result is not perfect. The DCT was established as the JPEG standard in 1992, after six years of work by the Joint Photographic Experts Group. But the technology continues to move forward.

This whole construction has recently been made sharper, in order to obtain an *orthonormal basis*. Specific frequencies and specific windows are chosen, and this is what we want to explain. The frequencies involve the factor $k + \frac{1}{2}$ that is responsible for the even-odd symmetry of the extension E^{IV} and the discrete DCT-IV. Requirements are imposed on the windows to give orthogonality where one window overlaps its translate. This is not orthogonality of the windows alone or the cosines alone, but orthogonality of their products $g_{nk}(t)$ for $k = 0, 1, \dots$ and $-\infty < n < \infty$:

$$g_{nk}(t) = g(t-n) \cos \left[\left(k + \frac{1}{2} \right) \pi (t-n) \right]. \quad (8.27)$$

If $g(t)$ is the box function, then $g(t-n) = 1$ on the time interval $[n, n+1]$. We have the basis functions (8.13) of the even-odd extension. They are the continuous analog of the DCT-IV basis. Moving to smoother windows $g(t-n)$ will reduce the blocking effects at the box edges.

A more general form, very useful in applications, allows windows of *varying length*. The time line is divided into unequal intervals $[t_n, t_{n+1}]$. There is a window function $g_n(t)$ for each interval. It equals one on an inner interval, and it drops toward zero as it passes t_n and t_{n+1} . The window reaches zero well before t_{n-1} on the left and t_{n+2} on the right. This function $g_n(t)$ overlaps only its two neighbors $g_{n-1}(t)$ and $g_{n+1}(t)$. We mention immediately one key requirement for orthogonality of the basis:

$$\sum (g_n(t))^2 \equiv 1. \quad (8.28)$$

The cosines within the window still have the even-odd symmetry associated with DCT-IV. They are *even* when continued across the left end t_n of the window, and they are *odd* across the right end t_{n+1} . This symmetry leads to orthogonality, when the windows have even symmetry at both ends and satisfy $\sum (g_n(t))^2 = 1$. The more general form of the basis functions is

$$g_{nk}(t) = g_n(t) \cos \frac{(k + \frac{1}{2})\pi(t - t_n)}{t_{n+1} - t_n}. \quad (8.29)$$

This reduces to the previous form when $t_n = n$ and the window lengths are equal and the windows are translates of one window $g(t)$.

Varying window lengths is an extremely important advantage in speech processing. We attack and hold phonemes for very different lengths of time—and the waveforms in those intervals are very different. We indicate below how the windows lead to an orthogonal basis $g_{nk}(t)$ in this nonuniform case also. For simplicity of exposition, we concentrate first on the equal-interval construction of (8.27), which is important in discrete time too.

This section analyzes cosine windows in continuous time. The next chapter will study *cosine-modulated filter banks* in discrete time.

The Window Functions

Start with a single window $g(t)$ and its translates $g(t - n)$. This window function is supported on an interval $[-a, 1 + a]$ that stretches beyond $[0, 1]$. If the extension length is $a < \frac{1}{2}$ at both ends, $g(t)$ will overlap only its nearest neighbors $g(t - 1)$ and $g(t + 1)$.

The window rises from zero at $t = -a$ to one at $t = a$. Here is a continuous rise:

$$g(t) = \sin \left[\frac{\pi}{4} \left(1 + \frac{t}{a} \right) \right] \quad \text{for } -a \leq t \leq a.$$

At $t = -a$ this is $g(-a) = \sin 0 = 0$. At $t = a$ the window reaches $g(a) = 1$. The window continues with $g(t) \equiv 1$ on the inner interval $[a, 1 - a]$. At the right end $g(t)$ drops to zero in the same way that it rose. Then the window is symmetric around its center point $t = \frac{1}{2}$, and the reflection $t \rightarrow 1 - t$ leaves it unchanged:

$$g(t) = \sin \left[\frac{\pi}{4} \left(1 + \frac{1-t}{a} \right) \right] \quad \text{for } 1-a \leq t \leq 1+a.$$

We have drawn this $g(t)$ in Figure 8.10. The overlap with $g(t - 1)$ is crucial. The key property in that overlap region is

$$(g(t-1))^2 + (g(t))^2 = \sin^2 \left[\frac{\pi}{4} \left(1 - \frac{t}{a} \right) \right] + \sin^2 \left[\frac{\pi}{4} \left(1 + \frac{t}{a} \right) \right] = 1.$$

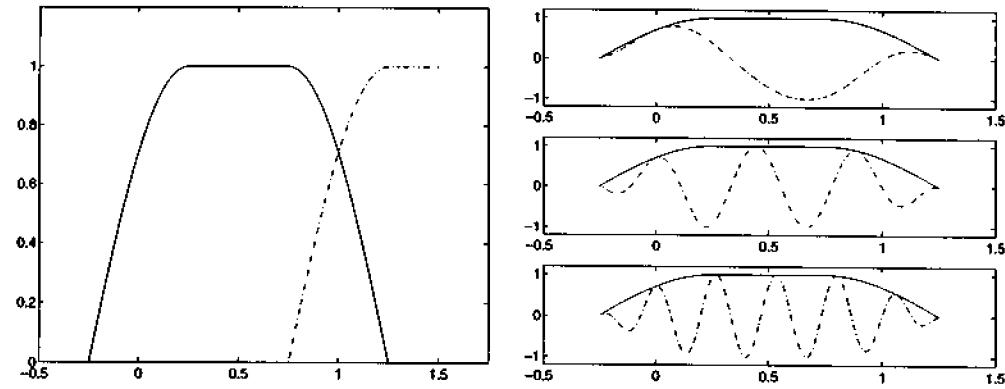


Figure 8.10: The window functions $g(t)$ and $g(t - 1)$ for $a = 0.25$ (left). The basis functions of the local cosine transform for $k = 1, 4$ and 7 .

This is really $(\cosine)^2 + (\sin)^2 = 1$. It leads directly to $\sum (g(t - n))^2 \equiv 1$ for all time. Away from the overlap intervals, a single window has unit height and all other windows are at zero.

This particular window can be made smoother. Replace t in its definition by $\sin \frac{\pi t}{2a}$. The derivative $g'(t)$ has a factor $\cos \frac{\pi t}{2a}$ from the chain rule, and that factor is zero at $t = \pm a$. Each repetition of this step introduces one more continuous derivative in $g(t)$. (We do not know of a careful search for an optimal a and an optimal window.) Allowing for unequal spacings and varying windows, the requirements at the overlap around $t = t_n$ are

$$(g_{n-1}(t))^2 + (g_n(t))^2 = 1 \quad \text{and} \quad g_{n-1}(t) = g_n(2t_n - t) \quad \text{for } |t - t_n| \leq a_n. \quad (8.30)$$

In the time-invariant case, with uniform spacing, a fixed window $g(t)$ is shifted to $g_n(t) = g(t - n)$. The conditions (8.30) around $n = 0$ and $t_0 = 0$ apply at every $t_n = n$. These are the conditions that we work with first:

$$(g(t + 1))^2 + (g(t))^2 = 1 \quad \text{and} \quad g(t + 1) = g(-t) \quad \text{for } |t| \leq a. \quad (8.31)$$

Local Orthogonal Bases

Suppose the window $g(t)$ is a box on $[0, 1]$. The basis functions $\cos((k + \frac{1}{2})\pi t)$ are orthogonal within that box. They are even functions around $t = 0$, where the cosine equals one. They are odd functions around $t = 1$, where the cosine of $(k + \frac{1}{2})\pi$ equals zero. The main point is that *we can smooth those windows and retain orthogonality*. The first author learned the proof from lectures at MIT by Stéphane Mallat, which will lead to a book on time-frequency resolutions including wavelets [Mt].

Theorem 8.1 *The functions $g_{nk}(t) = g(t - n) \cos[(k + \frac{1}{2})\pi(t - n)]$ are an orthogonal basis for $L^2(\mathbb{R})$.*

Proof. The function $g_{0k}(t)$ is centered on $[0, 1]$ and extends over $[-a, 1 + a]$. On the interval to the left, $g_{-1\ell}(t)$ is centered on $[-1, 0]$ and reaches as far right as $t = a$. The two functions overlap on the interval $[-a, a]$, where

$$g_{0k}(t) g_{-1\ell}(t) = g(t) g(t + 1) \cos[(k + \frac{1}{2})\pi t] \cos[(\ell + \frac{1}{2})\pi(t + 1)]. \quad (8.32)$$

We must show that the integral is zero and “the tails are orthogonal”.

The product $g(t)g(t+1) = g(t)g(-t)$ is an even function around $t = 0$. So is the cosine of $(k + \frac{1}{2})\pi t$. The second cosine is odd, because the addition formula for cosines gives a sine:

$$\cos[(\ell + \frac{1}{2})\pi(t+1)] = -\sin[(\ell + \frac{1}{2})\pi t] \sin[(\ell + \frac{1}{2})\pi].$$

The reader notices that the key is in $\cos(\ell + \frac{1}{2})\pi = 0$! Then the product (8.32) is odd and its integral over $[-a, a]$ is zero. This proves orthogonality of any two g_{nk} 's with different n 's.

Now consider $g_{0k}(t)g_{0\ell}(t)$. Both factors are centered on $[0, 1]$, with different frequencies. Define the product of cosines $C_{k\ell}(t)$ as $\cos[(k + \frac{1}{2})\pi t] \cos[(\ell + \frac{1}{2})\pi t]$. Then $C_{k\ell}(t)$ is even around $t = 0$ and also around $t = 1$. (Odd times odd is even.) The other factor is $(g(t))^2$. At the left end, the key is that $(g(t))^2 + (g(-t))^2 = 1$:

$$\begin{aligned} \int_{-a}^a (g(t))^2 C_{k\ell}(t) dt &= \int_0^a (g(-t))^2 C_{k\ell}(t) dt + \int_0^a (g(t))^2 C_{k\ell}(t) dt \\ &= \int_0^a C_{k\ell}(t) dt. \end{aligned} \quad (8.33)$$

Around the right endpoint we use $(g(t))^2 + (g(t-1))^2 = 1$:

$$\int_{1-a}^{1+a} (g(t))^2 C_{k\ell}(t) dt = \int_{1-a}^1 C_{k\ell}(t) dt. \quad (8.34)$$

The center interval $[a, 1-a]$ has $g(t) = 1$. Again we are integrating $C_{k\ell}(t)$, the product of cosines. Orthogonality of the cosines on $[0, 1]$ gives orthogonality of $g(t)$ times cosines on the larger interval. Combine (8.33) and (8.34) to see that $g(t)$ disappears:

$$\boxed{\int_{-a}^{1+a} (g(t))^2 C_{k\ell}(t) dt = \int_0^1 C_{k\ell}(t) dt = \frac{1}{2}\delta(k-\ell).} \quad (8.35)$$

This proves that the local cosines $g_{nk}(t)$ are **orthogonal**. The integral on the left can go from $-\infty$ to ∞ , since $g(t)$ is zero on the rest of the line. It remains to prove that the local cosines are a **basis** for $L^2(\mathbb{R})$. How do we express an arbitrary $f(t)$ as a combination of $g_{nk}(t)$?

The ordinary cosines $\cos[(k + \frac{1}{2})\pi t]$ are an orthogonal basis on $[0, 1]$. We “fold” the function $f(t)$ into this interval, creating $h(t)$:

$$\text{Folding } h(t) = \begin{cases} g(t)f(t) + g(-t)f(-t) & \text{on } [0, a] \\ f(t) & \text{on } [a, 1-a] \\ g(t)f(t) - g(2-t)f(2-t) & \text{on } [1-a, 1] \end{cases}$$

Within $[0, 1]$ expand $h(t)$ in the basis $\cos[(k + \frac{1}{2})\pi t]$. Those cosines extend $h(t)$ outside $[0, 1]$, keeping it even around $t = 0$ and odd around $t = 1$. These symmetries are built into the definition of $h(t)$, so there is no change in formula on the larger interval $[-a, 1+a]$. Multiplying by $g(t)$ turns $h(t) = \text{combination of cosines}$ into $g(t)h(t) = \text{combination of } g_{0k}(t)$. Here is $g(t)h(t)$:

$$g(t)h(t) = \begin{cases} (g(t))^2 f(t) + g(t)g(-t)f(-t) & \text{on } [-a, a] \\ f(t) & \text{on } [a, 1-a] \\ (g(t))^2 f(t) - g(t)g(2-t)f(2-t) & \text{on } [1-a, 1+a] \end{cases}$$

A similar step takes place on the interval $[-1, 0]$. Replace t by $t + 1$ in all formulas. Then $h_{-1}(t)$ is defined from $f(t)$ and expressed as a combination of cosines. Multiplication by the shifted window gives $g(t+1)h_{-1}(t)$ as a combination of windowed cosines $g_{-1\ell}$, while $g(t)h(t)$ is produced out of $g_{0k}(t)$. All we have to check is that on the overlap interval $[-a, a]$, the sum $g(t)h(t) + g(t+1)h_{-1}(t)$ adds to $f(t)$. Since $g(t+1) = g(-t)$ in the overlap, we have it:

$$(g(t))^2 f(t) + g(t)g(-t)f(-t) + (g(-t))^2 f(t) - g(-t)g(t)f(-t) = f(t). \quad (8.36)$$

Thus every $f(t)$ is a combination of local cosines. Those are an orthogonal basis. *Theorem proved.*

Note that $g(t)h(t)$ is the even part of $f(t)$ around $t = 0$, and $g(t+1)h_{-1}(t)$ is the odd part. Equation (8.36) is simply (even part) + (odd part) = $f(t)$. An early construction by Wilson alternated $\cos k\pi t$ in one interval with $\sin k\pi t$ in the next interval. The cosines are even at both ends, the sines are odd at both ends, and each overlap interval again has even and odd—which gives orthogonality and reproduces $f(t)$ as above.

The proof also yields a *fast algorithm*. To expand $f(t)$ in the local cosines, we expanded $h(t)$ and $h_{-1}(t)$ in ordinary cosines on $[0, 1]$ and $[-1, 0]$. So compute those coefficients b_{0k} and $b_{-1\ell}$ by a fast DCT-IV. The same computation on other unit intervals gives all coefficients in $f(t) = \sum b_{nk}g_{nk}(t)$. We record this fact for the basic interval $[0, 1]$ and the coefficients b_{0k} :

$$2b_{0k} = \int_{-\infty}^{\infty} f(t) g_{0k}(t) dt = \int_0^1 h(t) \cos \left[\left(k + \frac{1}{2} \right) \pi t \right] dt. \quad (8.37)$$

The local cosine coefficients of $f(t)$ are the ordinary cosine coefficients of the “folded” function $h(t)$. The normalization by 2 comes from (8.35).

Unequal Spacing and Different Windows

The analysis was simplified above, by assuming all windows to be translates $g(t-n)$ of a basic window. This is not necessary. The construction allows different windows $g_n(t)$ on intervals of different lengths. The time variable $t-n$ in the cosines can become $(t-t_n)/(t_{n+1}-t_n)$. The functions $g_{nk}(t)$ still provide an orthonormal basis.

The changes are easy to indicate, and their confirmation is a good exercise. The window $g_n(t)$ in Figure 8.11 extends from $t_n - a_n$ to $t_{n+1} + a_{n+1}$. The previous window $g_{n-1}(t)$ ends at $t_n + a_n$, and on the overlap interval the requirements are (8.30). The condition $a_n + a_{n+1} \leq t_{n+1} - t_n$ avoids any overlap of $g_{n-1}(t)$ with $g_{n+1}(t)$. This is the non-uniform equivalent of $2a \leq 1$. Then we have $\sum (g_n(t))^2 = 1$ as desired.

The local cosine coefficients b_{nk} of an arbitrary $f(t)$ are still the ordinary cosine coefficients of *folded functions* $h_n(t)$:

$$\int_{-\infty}^{\infty} f(t) g_n(t) \cos \left[\left(k + \frac{1}{2} \right) \pi \frac{t - t_n}{t_{n+1} - t_n} \right] dt = \int_{t_n}^{t_{n+1}} h_n(t) \cos \left[\left(k + \frac{1}{2} \right) \pi \frac{t - t_n}{t_{n+1} - t_n} \right] dt. \quad (8.38)$$

The latter integral moves to the interval $[0, 1]$ and is quickly computed by the DCT-IV. The folded function is

$$h_n(t) = \begin{cases} g_n(t)f(t) + g_{n-1}(t)f(2t_n - t) & \text{on } [t_n, t_n + a_n] \\ f(t) & \text{on } [t_n + a_n, t_{n+1} - a_{n+1}] \\ g_n(t)f(t) - g_{n+1}(t)f(2t_{n+1} - t) & \text{on } [t_{n+1} - a_{n+1}, t_{n+1}] \end{cases}$$

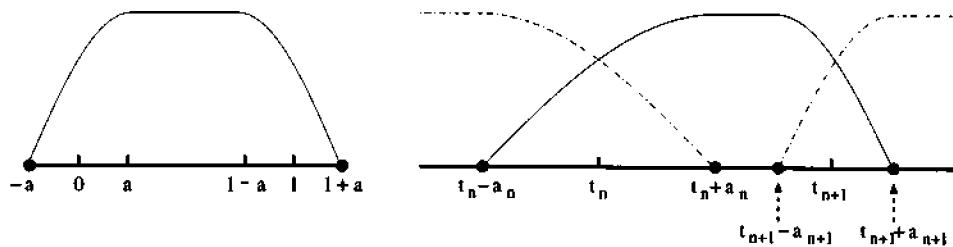


Figure 8.11: Window $g(t)$ and its non-uniform translates at t_n and t_{n+1} .

This even-odd folding matches the even-odd property of $\cos(k + \frac{1}{2})\pi t$. An even-even extension would fail! (Unless it is alternated with odd-odd.) The construction is neat and efficient, and must be regarded as strong competition for wavelets. A fully detailed description is given by Wickerhauser [W]. It has a close relation to wavelets, as we now indicate.

Sinc wavelets and Meyer wavelets The local cosines give a splitting of the time axis. The sinc wavelets are also local cosines, but on the frequency axis. Remember from Section 2.3 that the sinc wavelet and its Fourier transform are

$$\begin{aligned} w(t) &= \frac{\sin 2\pi t - \sin \pi t}{\pi t} = \phi(2t) - \phi(t) \\ \hat{w}(\omega) &= \begin{cases} 1 & \text{for } \pi \leq |\omega| < 2\pi \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

When the wavelet is dilated and shifted to $w(2^j t - k)$, its Fourier transform is moved to the interval $2^j \pi \leq |\omega| < 2^{j+1} \pi$. It is modulated by $\exp(-i\omega k/2^j)$. The transform becomes a local cosine, or rather a local exponential. It is not smooth in ω because the wavelet $w(t)$ does not decay quickly in time.

Note that the order of accuracy is $p = \infty$! The transform is *infinitely flat* at $\omega = 0$ and $\omega = \pi$. (The scaling function is a box function in ω , the ideal lowpass filter.) This corresponds to the *spectral method* in the numerical solution of partial differential equations, which also achieves $p = \infty$. Finite differences and finite elements have finite p , like FIR filters. An infinite value of p can only be achieved by functions $\phi(t)$ and $w(t)$ with infinite support. They are nonzero on the whole line, but the simplest sinc construction has slow decay.

Smoothing the local cosines (in ω) will produce faster decay (in t). This leads to the Meyer wavelets, which are *windowed in frequency*. The window can have infinite smoothness in ω , so the time decay can be faster than any power of t . The orthogonality of the sincs is maintained by the conditions on the windows. But windowing in time (to get smooth local cosines) is more practical.

Biorthogonal Local Cosines

Every wavelet construction becomes more general when the orthogonality requirement is lifted. The same is true of local cosines. For *biorthogonal* bases, the sum of squares $S(t) = \sum_{-\infty}^{\infty} (g(t-n))^2$ of window functions is not necessarily one. Then a *dual window* is defined by

$$\tilde{g}(t) = \frac{g(t)}{S(t)}. \quad (8.39)$$

The dual basis functions are the local cosines using the dual window:

$$\tilde{g}_{nk}(t) = \tilde{g}(t-n) \cos \left[(k + \frac{1}{2})\pi (t-n) \right]$$

[Matv] gives a nice discussion of biorthogonality of the dual bases. This follows directly from orthogonality with window functions $g_{\text{orth}}(t-n) = g(t-n)/\sqrt{S(t)}$ on the interval $[-\frac{1}{2}, \frac{1}{2}]$. The reader sees immediately that the sum of $g_{\text{orth}}^2(t-n)$ is $S(t)/S(t) = 1$.

The condition number of the dual bases governs stability. This number is one for an orthonormal basis (using the earlier windows $g(t-n)$) and greater than one for a biorthogonal basis (using $g(t)$ and $\tilde{g}(t)$). The condition number naturally depends on the distance of $S(t)$ from 1. Matviyenko shows how to construct windows with $S_{\max}(t) \leq 2S_{\min}(t)$. They give better compression than the orthogonal local cosines, and stability is well in control. He carries out an approximate optimization of $g(t)$ for the compression of sinusoids $\cos(\omega t + \alpha)$ of arbitrary frequency and phase, and tabulates the resulting windows. The graphs of $\tilde{g}(t)$ show a double peak which looks more alarming than it is.

The main point is *freedom versus constraints*, in the choice of window as in the choice of wavelets. Freedom gives the possibility of smooth windows. Constraints make those windows successful. A possible constraint is *accuracy*: to reproduce exactly a constant function (or all polynomials of degree less than p). We can further limit the search to functions $g(t) = \sum c_k \cos \left[(k + \frac{1}{2})\pi t \right]$ around $t = 0$, and ask for maximum smoothness at $t = \pm \frac{1}{2}$. This leads to “spline windows.” The real tests of coding gain and compression are still ahead.

Note. The *folding operator* is defined above by $h(t) = Ff(t)$. There is also an *unfolding operator*. When they use the same window, these operators F and U are transposes. The plus sign and minus sign on the first and third lines of $h(t)$ are reversed for the unfolded $u(t) = Uh(t)$. When F and U use dual windows $g(t)$ and $\tilde{g}(t)$, these operators are inverses. When they use the self-dual windows that have $S(t) = \sum g^2(t-n) \equiv 1$, the operators are both transposes and inverses. Therefore they are unitary operators. The families $\{F_n\}$ and $\{U_n\}$ which fold and unfold around $t = n$ (or more generally $t = t_n$) are the foundation for a full theory of smooth local cosines.

Jawerth, Liu, and Sweldens point out that in image compression, *folding can be seen as a preprocessing step for JPEG*. It is a generalized extension. The extension can be even or odd, symmetric or antisymmetric, and it changes the signal also *inside* the basic interval. The overlapping intervals allow perfect reconstruction — orthogonal or biorthogonal — of the signal.

Problem Set 8.4

1. Why does the linear rise $g(t) = \frac{1}{2} \left(1 + \frac{t}{a}\right)$ in the interval $[-a, a]$ not lead to orthogonality?
The problem is lack of smoothness.
2. A rise function can be written as $g(t) = \sin \left(\theta \left(\frac{t}{a} \right) \right)$. Here $\theta(-1) = 0$ and $\theta(1) = \frac{\pi}{2}$. Show that $\theta(t) + \theta(-t) = \frac{\pi}{2}$ assures $(g(t))^2 + (g(-t))^2 = 1$.
3. Compute a cubic spline $\theta(t)$, increasing from $\theta(-1) = 0$ to $\theta(1) = \frac{\pi}{2}$, that meets the requirement in Problem 2.
4. (Thesis project) Experiment with several uniform windows $g(t)$ and several overlap lengths a to optimize compression. Compare with the blocking effects from a window box.

5. Find the parity (even or odd?) of the function $\sin(k + \frac{1}{2})\pi t$ around the endpoints of $[0, 1]$.
 This is the sine-IV basis used in the DST-IV transform.
6. Show that the conditions on the uniform windows $g(t - n)$ can be written as

$$\begin{aligned} g^2(t) + g^2(-t) &= 1 & -\frac{1}{2} &\leq x &\leq \frac{1}{2} \\ g(t) = g(1-t) && \frac{1}{2} &\leq x &\leq \frac{3}{2} \\ g(t) = 0 && \text{elsewhere.} \end{aligned}$$

This yields $g(t) \equiv 1$ in $[a, 1-a]$ where there is no overlap.

7. (a) Draw a linear function $f(t)$ on $[0, 1]$ and its folding $h(t) = Ff(t)$.
 (b) Draw a linear function $h(t)$ on $[0, 1]$ and its unfolding $u(t) = Uh(t)$.
8. Explain (8.34) by following the proof of (8.33).

8.5 Boundary Filters and Wavelets

The striking fact about FIR filter banks is that *the banded analysis matrix has a banded inverse*. That inverse is the synthesis matrix.

Banded matrices correspond to FIR filters and compactly supported wavelets. The basis functions have finite length. The matrices may even be orthogonal. The question is how to maintain these properties at a boundary, by adding new rows and new functions:

Boundary filters are the “end rows” of an $L \times L$ filter matrix H_L .

Boundary scaling functions and wavelets are the “end functions” of a basis.

If we construct boundary filters, the dilation equation yields boundary functions. The reverse direction also succeeds. The functions have dilation coefficients that go into the filters. All constructions are in the time domain, because transform methods have major difficulty at boundaries. We begin with boundary filters (completing a matrix) and then create boundary functions (completing a basis). Those are really equivalent.

The matrices begin with the lowpass H_0 and the highpass H_1 , subsampled. These are infinite 1×2 block Toeplitz matrices. The rows have a double shift from $(\downarrow 2)$. We *interleave the filters* $(\downarrow 2)H_0$ and $(\downarrow 2)H_1$ to produce 2×2 blocks. Then there is an ordinary Toeplitz shift of *one block* between rows of H_b :

$$H_b = \left[\begin{array}{cccccc} \dots & \dots & & & & & \\ h_0(N) & h_0(N-1) & \dots & \dots & h_0(1) & h_0(0) & \\ h_1(N) & h_1(N-1) & \dots & \dots & h_1(1) & h_1(0) & \\ & h_0(N) & h_0(N-1) & \dots & \dots & h_0(1) & h_0(0) \\ & h_1(N) & h_1(N-1) & \dots & \dots & h_1(1) & h_1(0) \\ & & & & & \dots & \dots \end{array} \right]$$

Chapter 9 will have M filters and $(\downarrow M)$. Then H_b has $M \times M$ blocks.

The synthesis filters have $F_0(z) = H_1(-z)$ and $F_1(z) = -H_0(-z)$. In the orthogonal case, F_0 is the ordinary flip (the transpose) of H_0 . Therefore H_1 is the alternating flip. The key point is that both polyphase matrices, $H_p(z)$ in analysis and $H_p^{-1}(z)$ in synthesis, are *polynomial*. The determinant of $H_p(z)$ is z^{-N} , by the halfband condition on $H_0(z)H_1(-z)$ that makes everything work.

When signals have finite length L , the infinite matrix \mathbf{H}_b must change to $L \times L$. We assume that $L > N$ and probably $L \gg N$. Then the “middle” of the matrices is not affected, but the “ends” will be new. We have to choose those end rows — the *boundary filters* in the $L \times L$ matrix \mathbf{H} . The first question is whether FIR boundary filters can yield $\mathbf{H}^{-1}\mathbf{H} = \mathbf{I}$.

The answer is yes. The inverse matrix stays banded. Then the problem is to choose among boundary filters. We will indicate some reasonable choices, but more experiments and experience are needed. This is a research problem of great interest.

In the orthogonal case, the boundary filters are to preserve $\mathbf{H}^T\mathbf{H} = \mathbf{I}$. The middle filters (interior filters) have this orthogonal property — the middle is unchanged from $\mathbf{H}_b^T\mathbf{H}_b = \mathbf{I}$ in the infinite case. We have to be sure that the new end rows of \mathbf{H} and the new end columns of \mathbf{H}^{-1} have limited length $\leq N$ and not full length L .

Wavelets have similar difficulties. When $\phi(t - n)$ and $w(t - n)$ fall across the boundary, they must change. This happens at each scale, and the multiresolution $V_0 \subset V_1 \subset \dots$ should be maintained. The boundary functions are to be combinations of internal functions and boundary functions at scale $2t$. The shapes are the same at all scales. The coefficients in that boundary dilation equation and boundary wavelet equation give the boundary filters!

Filter Bank Completions

Our goal is an $L \times L$ analysis matrix \mathbf{H}_L . The synthesis matrix will be its inverse. In the orthogonal case, which we emphasize most, this inverse is \mathbf{H}_L^T .

\mathbf{H}_L begins with rows from the infinite filter matrix. For the interior part \mathbf{H}_{in} , we only save *complete rows*. Our example will be the Daubechies 4-tap filter with coefficients $(a, b, c, d) = (1 + \sqrt{3}, 3 + \sqrt{3}, 3 - \sqrt{3}, 1 - \sqrt{3})/4\sqrt{2}$. For convenience we save only *one* lowpass and highpass row:

$$\mathbf{H}_{in} = \begin{bmatrix} 0 & d & c & b & a & 0 \\ 0 & -a & b & -c & d & 0 \end{bmatrix}.$$

These rows are orthonormal. Therefore $(\mathbf{H}_{in})(\mathbf{H}_{in}^T) = \mathbf{I}$. The task is to add four new orthonormal rows — the boundary filters — to achieve $(\mathbf{H}_L)(\mathbf{H}_L^T) = \mathbf{I}$ with square matrices.

Those four filters will separate into left end and right end, and into lowpass and highpass. The square matrix will have the form

$$\mathbf{H}_L = \begin{bmatrix} \mathbf{H}_{left} \\ \mathbf{H}_{in} \\ \mathbf{H}_{right} \end{bmatrix} = \begin{bmatrix} r & s & t & 0 & 0 & 0 \\ u & v & w & 0 & 0 & 0 \\ 0 & d & c & b & a & 0 \\ 0 & -a & b & -c & d & 0 \\ 0 & 0 & 0 & e & f & g \\ 0 & 0 & 0 & x & y & z \end{bmatrix}. \quad (8.40)$$

Notice the freedom that has been removed and the freedom that remains. The boundary filters are allowed only three coefficients. At first this seems difficult, because the short rows (r, s, t) and (u, v, w) must be orthogonal to $(0, d, c)$ and also $(0, -a, b)$. Fortunately those vectors are parallel! The product $bd + ac$ is zero, from the key property of the Daubechies coefficients — that (a, b, c, d) is orthogonal to its double shift. The Gram-Schmidt process can easily produce (r, s, t) and (u, v, w) . Similarly we find the two short boundary filters at the right end, which go in the last two rows of \mathbf{H}_L .

What freedom remains? We can premultiply the first two rows by any 2×2 orthogonal matrix. We use that freedom to make $u + v + w = 0$; that boundary filter becomes highpass. Similarly $x + y + z = 0$ makes the last row orthogonal to DC inputs (constant inputs), which therefore go through the lowpass channel. The actual coefficients are

$$\begin{aligned} (r, s, t) &= (-0.93907, 0.29767, -0.17186) & (e, f, g) &= (0.40345, 0.69879, 0.59069) \\ (u, v, w) &= (-0.34372, 0.81326, -0.46954) & (x, y, z) &= (0.25535, 0.51155, -0.80690). \end{aligned}$$

In practice the interior matrix \mathbf{H}_{in} has many more rows ($L - 4$ rows). These inner rows are not truncated and remain safely orthogonal. They separate the left end from the right end.

We now show that a similar construction of boundary filters will succeed for other (longer) interior filters. It is almost simple enough to do by hand. Herley gives a MATLAB code. The starting point is $(\mathbf{H}_{\text{in}})(\mathbf{H}_{\text{in}}^T) = \mathbf{I}$. Remember that \mathbf{H}_{in} is rectangular. The product $(\mathbf{H}_{\text{in}}^T)(\mathbf{H}_{\text{in}})$ in the opposite order cannot equal \mathbf{I} . But the discrepancy is all near boundaries:

$$\mathbf{P} = \mathbf{I} - (\mathbf{H}_{\text{in}}^T)(\mathbf{H}_{\text{in}}) = \begin{bmatrix} \mathbf{P}_{\text{left}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \mathbf{P}_{\text{right}} \end{bmatrix}. \quad (8.41)$$

This matrix satisfies $\mathbf{H}_{\text{in}}\mathbf{P} = \mathbf{H}_{\text{in}} - (\mathbf{H}_{\text{in}})(\mathbf{H}_{\text{in}}^T)(\mathbf{H}_{\text{in}}) = \mathbf{0}$. *The columns of \mathbf{P}_{left} and $\mathbf{P}_{\text{right}}$ are orthogonal to the rows of \mathbf{H}_{in} .* These columns with short vectors (boundary filters) can complete a full set of orthogonal rows in \mathbf{H}_L . The rank of \mathbf{P} is necessarily the difference between the number of columns and rows in \mathbf{H}_{in} . This is the correct number to complete a square matrix.

For the 2×6 example \mathbf{H}_{in} we find a 6×6 matrix \mathbf{P} :

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & & & \\ 0 & d^2 + a^2 & cd - ab & & & \\ 0 & cd - ab & c^2 + b^2 & & & \\ & & & b^2 + c^2 & ab - cd & 0 \\ & & & ab - cd & a^2 + d^2 & 0 \\ & & & 0 & 0 & 1 \end{bmatrix}.$$

The double-shift orthogonality $ac + bd = 0$ gave the zero blocks. Here \mathbf{P}_{left} is 3×3 but its rank must be 2. The first two columns (or rows) of \mathbf{P} are orthogonal to the rows of \mathbf{H}_{in} . *They give two left boundary filters.* After normalization to unit length, and rotation to make the second one highpass, they become (r, s, t) and (u, v, w) .

Some patience is needed to find the right number of zero columns for \mathbf{H}_{in} . For D_4 , one zero column at each end gave two boundary filters at each end. [HerVet] propose a way to keep this number even for the Daubechies filters D_{2p} . We illustrate for D_6 a slightly different approach, which keeps four rows of \mathbf{H}_{in} with no zero columns:

$$\mathbf{H}_{\text{in}} = \begin{bmatrix} f & e & d & c & b & a \\ -a & b & -c & d & -e & f \\ f & e & d & c & b & a \\ -a & b & -c & d & -e & f \end{bmatrix}. \quad (8.42)$$

Thus \mathbf{H}_{in} is 4×8 (and $8 - 4$ is a multiple of 4). Using $(\mathbf{H}_{\text{in}})(\mathbf{H}_{\text{in}}^T) = \mathbf{I}$, we find 4×4 blocks in \mathbf{P} :

$$\mathbf{P} = \mathbf{I} - (\mathbf{H}_{\text{in}}^T)(\mathbf{H}_{\text{in}}) = \begin{bmatrix} \mathbf{P}_{\text{left}} & 0 \\ 0 & \mathbf{P}_{\text{right}} \end{bmatrix}. \quad (8.43)$$

P has rank 4 and each block has rank 2. We obtain two 4-tap boundary filters at each end. When H_L has more inner filters, for signal lengths $L > 8$, they start and end with at least four zeros. This makes all inner filters orthogonal to our boundary filters. The $L \times L$ orthogonal analysis bank is complete.

General case: Biorthogonal filters have $(H_{in})(F_{in}) = I$ but not $(F_{in})(H_{in}) = I$. Again these are rectangular matrices with no truncation of the interior filters. We need boundary filters to complete square matrices. Here is the key:

$$P = I - (F_{in})(H_{in}) \quad \text{has} \quad (H_{in})P = H_{in} - H_{in}F_{in}H_{in} = 0.$$

The columns of P contain boundary filters that can be included with the columns of F_{in} . Similarly $P(F_{in}) = 0$. The rows of P contain boundary filters that are included with the rows of H_{in} . Finally we biorthogonalize these boundary filters. A useful identity (to put it mildly) is $P^2 = P$.

Time-varying Filter Banks: A Remark

Changing from length 4 filters to length 6 will create an internal boundary, at the moment of change. One way to maintain orthogonality is to use boundary filters just before the change and just after. Such a transition has no overlap. A smoother approach is to create *transition filters* that cross the internal boundary. In this case H_{in} has filters purely to the left and purely to the right.

Again $(H_{in})(H_{in}^T) = I$. Now $P = I - H_{in}^T H_{in}$ contains the transition filters. See [HeKoRaVe] for details of this important construction.

Direct Extrapolation at the Boundary

We briefly mention another approach to boundary filters. It is closer to the usual treatment of boundary conditions for differential equations. *The governing principle is to fit a polynomial to the data and extend that polynomial.* The job of extrapolation is to define functions and signals beyond the boundary — so the filter can be applied.

Zero-padding and symmetric extension are basic extrapolation methods. Higher degree polynomials give higher degree extrapolation and higher accuracy.

To be consistent with multiresolution, the input $x(n)$ should give two half-length outputs. If the inputs are coefficients of $\phi(2t - n)$ in a function $f(t)$, the output are coefficients of $\phi(t - n)$ and $w(t - n)$ for $f(t)$ at the next scale. In the interior all is normal. *At the boundary we determine a polynomial $F_{p-1}(t)$ of degree $p - 1$ that models $f(t)$.* Using only the p numbers $x(0), \dots, x(p-1)$ to determine the p coefficients in the polynomial gives a square matrix — but this is dangerous. The preference in [WiAm] is to use more inputs $x(0), \dots, x(N)$ and determine the polynomial coefficients by least squares.

By extending the polynomial we extrapolate the signal. Then filter and downsample as usual. This corresponds to a splitting of V_1 into $V_0 + W_0$. It also corresponds to a completion of the filter matrix near the boundary. For the lowpass Daubechies D_4 , new coefficients appear in the first row of H_L . The second lowpass row contains the usual d, c, b, a :

$$\begin{bmatrix} 0.8924 & 0.5174 & 0.0129 & -0.0085 & 0 & 0 \\ -0.1294 & 0.2241 & 0.8365 & 0.4830 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}.$$

Orthogonality is lost; accuracy is preserved; highpass coefficients are in [WiAm]. Also important in differential equations is the fact that *boundary conditions can be built into the extrapolation*.

There is a third “quick and dirty” approach that deals entirely with the matrix entries. We follow [KwTa] by considering a biorthogonal example with attractive coefficients:

$$H = \frac{1}{4} \begin{bmatrix} -1 & 3 & 3 & -1 \\ -1 & 3 & -3 & 1 \\ -1 & 3 & 3 & -1 \\ -1 & 3 & -3 & 1 \\ \dots \end{bmatrix} \quad \text{and} \quad H^{-1} = \frac{1}{4} \begin{bmatrix} 1 & 1 \\ 3 & 3 \\ 3 & -3 & 1 & 1 \\ 1 & -1 & 3 & 3 \\ 3 & -3 & \dots \\ 1 & -1 & \dots \end{bmatrix}. \quad (8.44)$$

The column coefficients 1, 3, 3, 1 in H^{-1} come from $(1 + z^{-1})^3$. The synthesis scaling function is a quadratic spline (accuracy $p = 3$). The row coefficients $-1, 3, 3, -1$ come from $(1 + z^{-1})(-1 + 4z^{-1} - z^{-2})$. This single zero gives $\tilde{p} = 1$. The product filter is the familiar halfband $P(z)$ of degree 6. This is a linear phase choice, not to be iterated too often! We noted in Section 7.2 that Condition E is violated. The cascade algorithm will not converge for the filter H .

Nevertheless it is a good example. Suppose the second column shown for H corresponds to the sample $x(0)$. The first column would normally multiply $x(-1)$, which does not exist. Extrapolation could create $x(-1)$. Equivalently, this external column $c = -\frac{1}{4}, -\frac{1}{4}$ can be folded into the internal columns by one of these rules:

Zero-padding: Delete the external column c .

Constant extrapolation: Add c to the zeroth column (3/4, 3/4).

Linear extrapolation: Add 2c to column 0 and $-c$ to column 1.

Quadratic extrapolation: Add 3c, $-3c, c$ to columns 0, 1, 2.

The key point is that H and H^{-1} both maintain their stacked double-shift form (rows of H and columns of H^{-1}). The boundary filters based on constant extrapolation $x(-1) = x(0)$, which adds c to column zero, keep their lowpass and highpass character:

$$H_L = \frac{1}{4} \begin{bmatrix} 2 & 3 & -1 \\ 2 & 3 & -1 \\ -1 & 3 & 3 & -1 \\ -1 & 3 & -3 & 1 \\ \dots \end{bmatrix} \quad \text{and} \quad H_L^{-1} = \frac{1}{4} \begin{bmatrix} 4 & 4 \\ 3 & -3 & 1 & 1 \\ 1 & -1 & 3 & 3 \\ 3 & -3 & \dots \\ 1 & -1 & \dots \end{bmatrix}. \quad (8.45)$$

For longer filters the extrapolation rules are similar, but care is needed. There is still much room for experiment and comparison of boundary filters.

Lowpass Cascade and the Boundary Dilation Equation

Scaling functions come from lowpass filters. The lowpass coefficients go into the dilation equation. This equation is solved for $\phi(t)$ by iteration — which means cascade. A good example is the synthesis filter bank H_L^{-1} given above. Its interior lowpass columns contain $(\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{4})$ and those coefficients lead to the quadratic spline $\phi(t)$ on $[0, 3]$:

$$\phi(t) = \frac{1}{4}\phi(2t) + \frac{3}{4}\phi(2t-1) + \frac{3}{4}\phi(2t-2) + \frac{1}{4}\phi(2t-3). \quad (8.46)$$

This spline has an explicit formula, which we mention but do not need. $2\phi(t)$ equals t^2 and $-2t^2 + 6t - 3$ and $(t - 3)^2$ on the intervals $[0, 1]$ and $[1, 2]$ and $[2, 3]$.

The boundary filter $(1, \frac{3}{4}, \frac{1}{4})$ in the first column of H^{-1} leads to a boundary scaling function $\phi_b(t)$. The boundary dilation equation uses those coefficients:

$$\phi_b(t) = \phi_b(2t) + \frac{3}{4}\phi(2t) + \frac{1}{4}\phi(2t - 1) \quad \text{for } t \geq 0. \quad (8.47)$$

Notice how ϕ_b is included with the functions at scale $2t$. Since $\phi(2t)$ and $\phi(2t - 1)$ are already known, this is an “*inhomogeneous dilation equation*” for the boundary function $\phi_b(t)$. It can be solved in at least three ways:

1. By creating a general method for inhomogeneous two-scale equations.
2. By an inspired guess.
3. By iteration, cascading the lowpass filter.

We choose Method 2, because the insight is important. The boundary filter comes from constant extrapolation, which preserves minimum accuracy $p = 1$. The constant function will belong to V_0 . In the interior we do have $\sum \phi(t - n) \equiv 1$. But near $t = 0$, the pieces from $\phi(t + 1)$ and $\phi(t + 2)$ are missing — those functions cross the boundary. It is natural to suspect that the boundary function $\phi_b(t)$ takes their place — but only on the right side $t \geq 0$:

$$\text{Inspired guess: } \phi_b(t) = \phi(t + 1) + \phi(t + 2) \quad \text{for } t \geq 0. \quad (8.48)$$

The support is $[0, 2]$. We try this function in the boundary dilation equation (8.47). On the right side it contributes $\phi(2t + 1) + \phi(2t + 2)$. On the left side, replace $\phi(t + 1)$ and $\phi(t + 2)$ using the interior dilation equation (8.46). Compare only for $t \geq 0$, so terms involving $\phi(2t + 3)$ and $\phi(2t + 4)$ can be and must be ignored. Equation (8.47) is satisfied!

This function $\phi_b(t) = \sum_{n < 0} \phi(t - n)$ will appear again, as a direct construction in continuous time. Figure 8.12 shows how it builds $f(t) \equiv 1$ into V_0 .

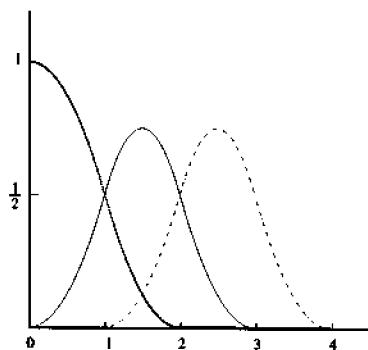


Figure 8.12: Quadratic splines and $\phi_b(t)$ add to 1.

Its dilation equation reveals the boundary filter. In that order, $\phi_b(t)$ produces the filter matrix rather than vice versa.

It is helpful also to see the lowpass cascade (Method 3). This means powers of the double-shift lowpass synthesis matrix:

$$L = \frac{1}{4} \begin{bmatrix} 4 & 3 & 1 & & & & \\ & 1 & 3 & 3 & 1 & & \\ & & 1 & 3 & 3 & 1 & \dots \end{bmatrix}.$$

Squaring leaves the unit column sums, and produces a *quadruple shift*:

$$L^2 = \frac{1}{16} \begin{bmatrix} 16 & 15 & 13 & 10 & 6 & 3 & 1 & \dots \\ & 1 & 3 & 6 & 10 & 12 & 12 & \dots \\ & & & 1 & 3 & \dots \end{bmatrix}.$$

The boundary filter affects only the first row. Elsewhere, convolving 1, 3, 3, 1 with 1, 0, 3, 0, 3, 0, 1 gives the interior sequence 1, 3, 6, 10, 12, 12, 10, 6, 3, 1. This corresponds to $H(z)H(z^2)$ — which is not $[H(z)]^2$. The cascade includes rescaling. L relates to integer spacing $\Delta t = 1$, while L^2 relates to $\Delta t = \frac{1}{2}$. The fact that all column sums are 1 means that, at convergence, the scaling functions $\phi_b(t), \phi(t), \phi(t-1), \dots$ add to 1. This brings back our formula (8.9) for $\phi_b(t)$.

It is unusual to solve a dilation equation so explicitly. But convergence of the cascade is not difficult to establish. Condition E holds for the interior filter. The boundary requires Condition E_b on a small square matrix in the corner of L . That 2×2 matrix has eigenvalues 1 and $\frac{1}{4}$ and the cascade converges.

For this example, the powers L^i are the slow way to solve the dilation equation. For most examples L^i is the only way. The cascade algorithm is also called the “graphical algorithm,” because a finite power of L yields an approximate graph of the scaling functions. Then one application of the other submatrix B (from the highpass filters) produces the boundary and interior wavelets.

Basis Completion by Boundary Functions

For a direct construction of boundary functions, suppose $\phi(t)$ is supported on $[0, 3]$. Our overall interval is finite, say $[0, 4]$. Then we can only shift once, to $\phi(t-1)$. All other translates of $\phi(t)$ cross the boundary and have to be changed. We are looking for four basis functions in V_0 and $4 \cdot 2^j$ basis functions in V_j . There are three important ways to construct them [CoDaVi]:

1. Periodic Extension: The shifted function $\phi(t-2)$ is supported on $[2, 5]$. The segment on $[4, 5]$ is wrapped back to $[0, 1]$. Similarly, the piece of $\phi(t-3)$ on $[4, 6]$ is wrapped back to $[0, 2]$. We are just periodizing the original functions.

This corresponds in the discrete case to *circulant matrices*. The lowpass analysis filter is a 4×4 circulant containing the four filter coefficients. It stretches to 4×8 with double shift from $(\downarrow 2)$. The highpass filter, similarly stretched, completes the 8×8 block circulant H_L .

2. Symmetric Extension: This would apply to the linear phase example based on the coefficients $(1, 3, 3, 1)$. An even-length symmetric filter (H filter) requires an H extension, centered at half-samples. The detailed discussion is in Section 8.2. Our point here is that the extension is implicitly creating boundary filters.

Symmetric extension is used for symmetric scaling functions. Those are not self-orthogonal; we are in the biorthogonal case. Our 1, 3, 3, 1 example has an even number of taps, equal in analysis and synthesis. The scaling function $\phi(t+1)$ on $[-1, 2]$ extends outside the basic interval $[0, 4]$, and is *folded* back inside by a symmetric extension to the whole line:

$$f^{\text{fold}}(t) = \sum_{-\infty}^{\infty} [f(t - 8n) + f(8n - t)]. \quad (8.49)$$

The terms with $n = 0$ are $f(t) + f(-t)$, even across $t = 0$. The other terms keep this symmetry and produce the double period 8. Thus any folded function is even with period 8:

$$f^{\text{fold}}(-t) = f^{\text{fold}}(t) \quad \text{and} \quad f^{\text{fold}}(t+8) = f^{\text{fold}}(t). \quad (8.50)$$

Figure 8.13 shows an idealized (!) picture of a symmetric $\phi(t)$ extended to the one-period in-

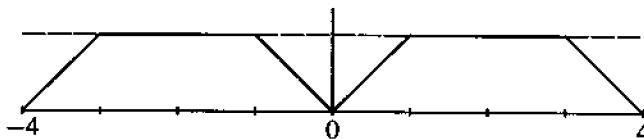


Figure 8.13: Symmetric folding of four symmetric scaling functions.

terval $[-4, 4]$. Their restrictions to $[0, 4]$ span the space V_0 with the correct dimension $4 \cdot 2^j = 4$. These are the four scaling functions and there are four corresponding wavelets. Similarly there will be four synthesis functions $\tilde{\phi}(t)$ and wavelets $\tilde{w}(t)$. *Furthermore biorthogonality is preserved* (Problems 1-2).

The original scaling functions $\phi(t+n)$ on the whole line add to the constant function 1, because the lowpass filter has the required zero at π . This good property is preserved by symmetric extension. *The folded functions still add to 1*. In our idealized figure the flat middle segments have height $\frac{1}{2}$. This piecewise linear $\phi(t)$ comes from a simple filter that cannot be part of an FIR filter bank (Problems 3-4). In general all the pieces of scaling functions that add to 1 are folded back into the interval so constants are in V_0 .

Even if $H(z)$ has extra zeros at π , we do not expect V_0 to give accuracy beyond $p = 1$. Symmetric extension (folding) generally produces jumps in the slope at the boundary. We turn to general methods that can achieve $p > 1$.

3. Boundary Functions: We need a completion method for the orthogonal case, when symmetric extension is unnatural. We also want to maintain the accuracy p of the internal functions. The following method will apply to all cases, orthogonal or not and linear phase or not. *We focus on completing the lowpass synthesis part* (the space V_0 and the filter F_0). A parallel construction applies to the analysis space \tilde{V}_0 and the filter H_0 . Then multiresolution makes W_0 and \tilde{W}_0 orthogonal to \tilde{V}_0 and V_0 .

The plan is to start with internal scaling functions supported inside the given interval, and add new functions at each end. To start, the new ones are linearly independent “boundary pieces.” We insist on short support and a dilation equation, so that $V_0 \subset V_1$. Additional properties lead to full accuracy. *Those short pieces are orthogonalized against the existing scaling function and wavelets*. In the orthogonal case those are the internal $\phi(t-n)$ and $w(t-n)$; in general they

are the synthesis functions. The algorithm is just Gram-Schmidt or its biorthogonal extension to “bi-Gram-Schmidt.”

The key point is that this orthogonalization produces functions supported *near the boundary*. Each boundary piece, however chosen, is already orthogonal to all scaling functions that are far inside the interval. There is no overlap. One source of boundary pieces is the tails of existing $\phi(t+k)$ and $w(t+k)$. Numerically this is a poor choice. The tails are very small for longer wavelets, which leads to severe ill-conditioning.

A better construction [CoDaVi] is to build polynomials up to degree $p-1$ into V_0 . On the whole line we have $\sum \phi(t-n) \equiv 1$. Then the first boundary piece ($t \geq 0$ only) is

$$\phi_b(t) = 1 - \sum_{n \geq 0} \phi(t-n) = \sum_{n < 0} \phi(t-n). \quad (8.51)$$

This is just a *sum of tails*. It is orthogonal to all internal $\phi(t-n)$, $n \geq 0$. Its support is $[0, N-1]$ because $\phi(t), \phi(t-1), \dots$ already add to 1 beyond $t = N-1$.

The boundary function $\phi_b(t)$ satisfies a dilation equation. Replacing t by $2t$ gives $\phi_b(2t) = 1 - \sum_{n \geq 0} \phi(2t-n)$. Use this to substitute for 1 in (8.51):

$$\phi_b(t) = \phi_b(2t) + \sum_{n \geq 0} [\phi(2t-n) - \phi(t-n)]. \quad (8.52)$$

Replace each $\phi(t-n)$ using its ordinary dilation equation. Thus $\phi_b(t)$ is a combination of $\phi_b(2t)$ and internal $\phi(2t-n)$. It belongs to V_1 and we are on our way.

Higher Accuracy. Suppose the original scaling function $\phi(t-n)$ can reproduce linear functions. The filter has at least two zeros at π . The wavelets have two vanishing moments. The combination $\sum n\phi(t-n)$ equals $\alpha t + \beta$; the constants $\alpha \neq 0$ and β are determined by the filter coefficients. Our task is to ensure that $\alpha t + \beta$ is in V_0 when we remove all $\phi(t-n)$ for $n < 0$. In their place goes $\phi_b(t)$ and a second boundary piece $\phi_{bb}(t)$:

$$\phi_{bb}(t) = \alpha t + \beta - \sum_{n \geq 0} n\phi(t-n) = \sum_{n < 0} n\phi(t-n). \quad (8.53)$$

The sum over $n < 0$ shows that the support is $[0, N-1]$. This piece need not be orthogonal to $\phi_b(t)$ or to the internal $\phi(t-n)$. We must orthogonalize it (or biorthogonalize it). Also we check that $\phi_{bb}(t)$ satisfies a dilation equation, so the multiresolution hierarchy $V_0 \subset V_1$ is still secure. Replacing t by $2t$ in (8.53) gives

$$\phi_{bb}(2t) = 2\alpha t + \beta - \sum_{n \geq 0} n\phi(2t-n). \quad (8.54)$$

Divide by 2 and substitute for αt in (8.54):

$$\phi_{bb}(t) = \frac{1}{2}\phi_{bb}(2t) + \frac{1}{2}\beta + \sum_{n \geq 0} n \left[\frac{1}{2}\phi(2t-n) - \phi(t-n) \right]. \quad (8.55)$$

Replace each $\phi(t-n)$ using its dilation equation, and replace $\frac{1}{2}\beta$ by $\left[\frac{1}{2}\beta\phi_b(2t) + \sum \phi(2t-n) \right]$. Then (8.55) guarantees that $\phi_{bb}(t)$ is in the next space V_1 .

It is important to count the functions! With p zeros at π , we are choosing p boundary pieces at each end to build the polynomials $1, \dots, t^{p-1}$ into V_0 . An interval of length L will have $L - 2p + 2$ internal Daubechies functions. For $L = 4$ and $p = 2$ we had $\phi(t)$ on $[0, 3]$ and $\phi(t-1)$ on $[1, 4]$. We only need $2p - 2$ additional functions, but we have $2p$.

To make space for $2p$ boundary pieces, we throw out the perfectly good $\phi(t)$ at the left end and the outermost scaling function at the right end. The term with $n = 0$ moves to the other side of (8.51). This only means that $\phi_b(t)$ has support $[0, N]$. A similar replacement at the other end assures that constant functions are in V_0 .

The simplest highpass construction starts with scaling functions $\phi(2t - n)$ that cross the boundary. Subtract their projections on V_0 to get *boundary wavelets* [CoDaVi, p. 73]. Together with the internal wavelets this produces $W_0 \perp V_0$ and $W_0 \subset V_1$. We indicate the D_4 case with $p = 2$ and normal Daubechies coefficients (a, b, c, d) :

$$\mathbf{F}_{\text{low}} = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times & \times & \times \\ & & d & c & b & a \\ & & & \times & \times & \times & \times & \times \\ & & & & \times & \times & \times \end{bmatrix}. \quad (8.56)$$

The 5×10 highpass matrix completes the 10×10 synthesis bank.

After orthogonalization, the filter coefficients are computed once and for all by [CoDaVi]. Those boundary filters are available from netlib@research.att.com with the message `send/stat/data/wavelets`.

Problem Set 8.5

1. Verify from (8.49) that folding plus restriction preserves inner products:

$$\int_0^4 f^{\text{fold}}(t)g^{\text{fold}}(t) dt = \int_{-\infty}^{\infty} f(t)g(t) dt.$$

2. If the functions $\phi(t-n)$ and $\tilde{\phi}(t-n)$ are biorthogonal on the whole line, verify that their folded versions are still biorthogonal on the finite interval. This interval is $[0, 4]$ when the folding in (8.49) has period 8.
3. What symmetric 4-tap filter would produce the piecewise linear $\phi(t)$ shown in Figure 8.13. How many zeros at π in $H(\omega)$?
4. Show that the filter with $\mathbf{h} = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ has $H(i) = H(-i) = 0$. It cannot be part of a PR filter bank! Why can no synthesis filter make $P(z) = F(z)H(z)$ into a halfband filter with $P(z) + P(-z) = z^4$?
5. Take two rows of (8.44) in \mathbf{H}_{in} and two columns of the inverse matrix in \mathbf{F}_{in} . Verify $\mathbf{H}_{\text{in}}\mathbf{F}_{\text{in}} = \mathbf{I}_{2 \times 2}$ and compute $\mathbf{P} = \mathbf{I} - \mathbf{F}_{\text{in}}\mathbf{H}_{\text{in}}$ (which is 6×6). Determine a set of boundary filters.
6. From the double-shift orthogonality of Daubechies D_6 , verify that the zero blocks of \mathbf{P} in equation (8.43) really are zero.
7. Suppose \mathbf{H}_{in} is $K \times L$ and $\mathbf{H}_{\text{in}}\mathbf{H}_{\text{in}}^T = \mathbf{I}_{K \times K}$. Verify that $\mathbf{Q} = \mathbf{H}_{\text{in}}^T\mathbf{H}_{\text{in}}$ and $\mathbf{P} = \mathbf{I} - \mathbf{Q}$ satisfy $\mathbf{Q}^2 = \mathbf{Q}$ and $\mathbf{P}^2 = \mathbf{P}$ and $\mathbf{Q}\mathbf{H}_{\text{in}}^T = \mathbf{H}_{\text{in}}^T$. \mathbf{Q} is the projection onto the column space of \mathbf{H}_{in}^T and \mathbf{P} is the projection onto the nullspace of \mathbf{H}_{in} .
8. (MATLAB) Zero-padding removes the first column \mathbf{c} of \mathbf{H} in the $-1, 3, 3, 1$ matrix in (8.44). Find the inverse of the resulting matrix.
9. (MATLAB) Linear extrapolation adds $2\mathbf{c}$ and $-\mathbf{c}$ to the next columns of \mathbf{H}_L , when the first column \mathbf{c} is removed. Find the upper left corner of the resulting \mathbf{H}_L^{-1} . What are the boundary dilation equations in analysis and synthesis?

Chapter 9

M-Channel Filter Banks

9.1 Freedom versus Structure

An M -channel filter bank has very considerable freedom for $M > 2$. That is true when we require perfect reconstruction. It is still true when we also require orthogonality. Those properties can be expressed directly as conditions on the $M \times M$ analysis polyphase matrix $\mathbf{H}_p(z)$:

1. Perfect reconstruction requires $\mathbf{H}_p(z)$ to be invertible for all z .
2. The synthesis filters are FIR when the determinant of $\mathbf{H}_p(z)$ is a delay z^{-l} .
3. The filter bank is orthogonal when $\mathbf{H}_p^T(z^{-1})\mathbf{H}_p(z) = I$.

In the orthogonal case, each synthesis filter F_k is the transpose (or flip, or time-reversal) of the analysis filter H_k . Then delays make F_k causal.

How to design a good M -band filter bank? For $M = 2$, orthogonality is already restrictive (and effectively prevents linear phase). The PR condition is less strict; linear phase can be achieved. In both cases the product filters $F_0(z)H_0(z)$ and $F_1(z)H_1(z)$ are halfband. Furthermore, one comes from the other by alternating signs. All four filters are created by factoring *one halfband filter*.

For $M > 2$, the number of freedoms grows faster than the number of restrictions. The choice of one M -th band filter (or one lowpass filter $H_0(z)$) does not determine the other choices. This is attractive at first — we can have linear phase with orthogonality. But decisions are still needed. We can make those decisions late or early! The range of designs can be left very wide (late decision). Or we can restrict the filter bank to have a structure that we know is desirable (early decision, simplifying the design). This chapter studies both possibilities. In all cases we absolutely want *fast implementation*.

In practice, fast algorithms come by cascading simple functions. At the lowest level they are based on 2×2 butterflies. At a higher level three structures are quick:

Rotations and delays or *DFT banks* or *DCT banks*.

All constant orthogonal matrices are products of $M(M - 1)/2$ plane rotations. All paraunitary matrices are products of rotations and delays. (Householder would use reflections and delays — this factorization is an important theorem.) Under quantization and roundoff, an orthogonal but-

terfly remains orthogonal. The difficulty will be the large number of rotation angles, approximately $NM(M - 1)/2$ for filters of lengths N . An optimal design may have excellent properties but it will not be easy to find.

This chapter will pursue all three structures. We develop the polyphase approach in Section 9.2 and the time domain approach (to LOT and GenLOT) in Section 9.3. Those sections extend to M channels the earlier ideas from two channels. Then Section 9.4 focuses on cosine modulation — not seen for two channels but now important.

It may assist the reader if we now briefly highlight DFT and DCT filter banks. These are *modulated* filter banks because all M filters come from frequency shifts of one prototype filter. It is usual to refer to DCT filter banks as *cosine-modulated filter banks*. In many applications, the DFT loses and the DCT wins.

Block Transforms

The simplest filter banks use only the M -point DFT or the M -point DCT. The signal is split into blocks of length M . These blocks are *separately* transformed. There is no overlapping or interaction between blocks, and no filter design is involved. The polyphase matrix can be the Fourier DFT matrix or the DCT matrix (this is the JPEG standard). In a block transform, H_p is just a constant matrix — and there is no smoothing between blocks.

The analysis half can be drawn with the modulators first or the decimators first. Figure 9.1 shows the direct form of the block DFT and the more efficient polyphase form ($W_M = e^{-j2\pi/M}$). Section 9.2 discusses the block generation step in detail, because a serial to parallel S/P converter is extremely important — as a way to start the filter bank.

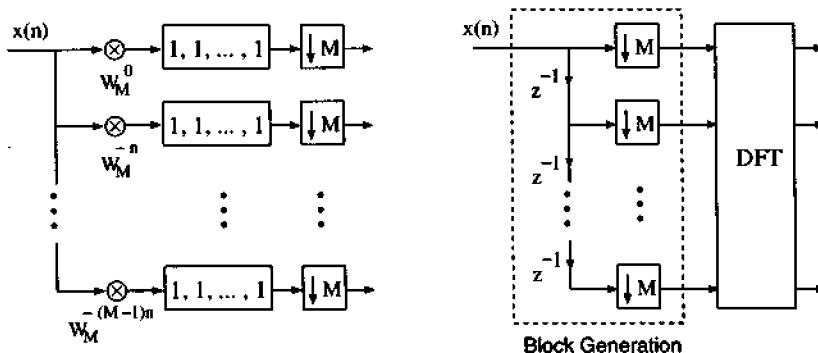


Figure 9.1: The DFT block transform: M samples at a time. Using DCT gives JPEG.

DFT Filter Banks

Now we include filters. The two-channel DFT filter bank appeared early in Chapter 4 (not with that name). The highpass frequency response shifted $H_0(e^{j\omega})$ by π . Consequently, $h_1(n)$ was the *alternating sign* version of $h_0(n)$. The reader will remember that $H_1(z) = H_0(-z)$ did not go well with perfect reconstruction. For two channels, the best arrangement is to alternate signs between H_0 and F_1 and between H_1 and F_0 . This is not DFT.

For an M -channel DFT bank, the frequency response from H_k is $H_0(e^{j\omega})$ shifted by $\frac{2\pi k}{M}$. The corresponding z-transform and time-domain relations are

$$H_k(z) = H_0(ze^{-j2\pi k/M}) \quad \text{and} \quad h_k(n) = h_0(n)e^{j2\pi k/M}.$$

The synthesis filters are modulated in the same way. Note that the coefficients become *complex* for $M > 2$. This is a disadvantage. The great advantage is the simplicity of design and the speed of implementation, when the whole analysis bank is based on one filter H_0 (and the DFT).

Figure 9.2 shows the direct form and polyphase form of the DFT analysis bank. It differs from Figure 9.1 only by including filters — which come before ($\downarrow M$) in the direct form and after ($\downarrow M$) in the polyphase form. We are free to use the DFT or the IDFT in analysis, and reverse this choice in synthesis. The transfer functions $E_k(z)$ are the polyphase components of $H_0(z)$.

Figure 9.2 reduces to Figure 9.1 when $h_0 = (1, 1, \dots, 1)$. Then the product of DFT and IDFT gives perfect reconstruction trivially, a block at a time. Section 9.2 derives the PR condition for a DFT bank that involves filters. The requirements on those filters are quite restrictive. DFT banks are generally superseded by filter banks based on the DCT, if reconstruction is desired.

We saw the same for continuous time in Chapter 8. *Cosine modulation is the best.*

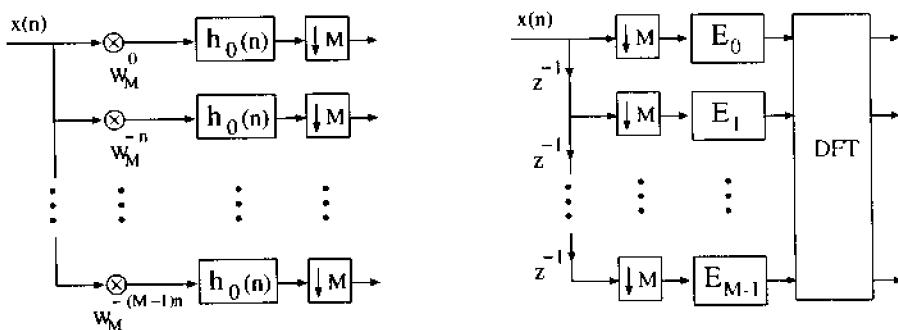


Figure 9.2: Direct form and polyphase form: DFT bank built from one filter.

Cosine-modulated Filter Banks

Cosine modulation replaces the complex DFT by the real DCT. Remember that there are four different cosine transforms. They use different extensions; Types II and IV are appropriate for filter banks. (Types I and III give undesirable bandwidths for the first and last filters. For the same reason, the filter length is constrained to be $2M$ or more generally $2KM$.) We use Type IV with the factors $k + \frac{1}{2}$ and $n + \frac{1}{2}$ in the frequencies.

The DCT matrix C^{IV} is symmetric and orthogonal. We shift the frequencies to achieve perfect reconstruction, starting from the lowpass prototype $p(n)$:

$$h_k(n) = f_k(n) = p(n) \sqrt{\frac{2}{M}} \cos \left[\left(k + \frac{1}{2} \right) \left(n + \frac{M+1}{2} \right) \frac{\pi}{M} \right]. \quad (9.3)$$

The shift of n by $\frac{M}{2}$ has the effect of centering p . That simplifies the conditions on this prototype filter — “the window” — to achieve perfect reconstruction. The PR conditions are beautiful for

filter length $2M$. We write them now without proof:

$$\text{Even symmetry} \quad p(n) = p(N - n) \quad (9.2)$$

$$\text{Orthogonality} \quad p^2(n) + p^2(n + M) = 1. \quad (9.3)$$

These are *precisely analogous* to the conditions on the continuous-time cosines in Chapter 8. There the window was subject to $g^2(t) + g^2(t + 1) = 1$. Section 9.4 will extend the discrete-time requirements to filter lengths $2K M$, and establish orthogonality and perfect reconstruction. There are analogous conditions in continuous time — when each window overlaps several other windows.

We can already see the major advantages of cosine modulation:

- Simplicity of design: one filter $p(n)$ only
- Symmetry and orthogonality
- Very fast implementation.

The simplicity is crucial when M is large. That is the central point — to accept and indeed to welcome restrictions that simplify the structure. The implementation is fast because the DCT is fast. Ultimately this is because the FFT is fast.

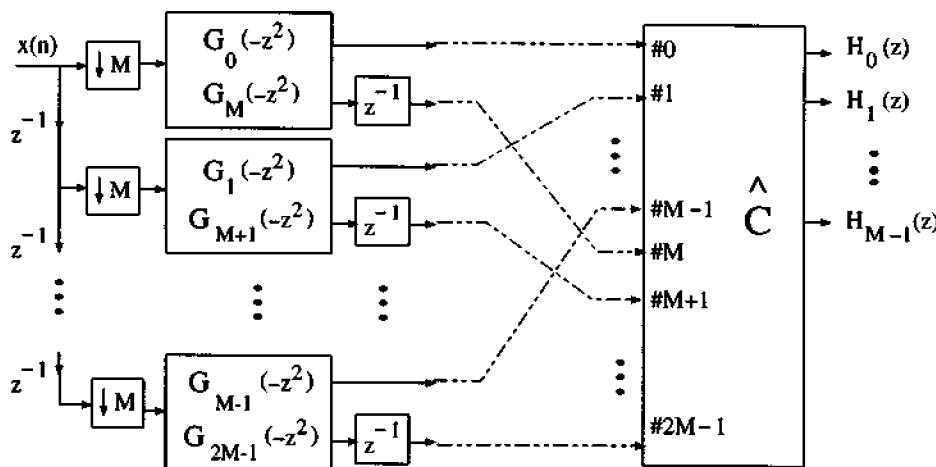


Figure 9.3: Cosine modulation by the DCT; k and $M + k$ are twins.

Remember that the M -point DCT is created from the $2M$ -point DFT. In our applications the DCT has real outputs, reducing the full (complex) computation by half. The analysis bank will involve

- $2M$ polyphase filters
- $2M$ complex modulators with real inputs
- one $2M$ -point DFT with real outputs.

Figure 9.3 is directly based on the DCT, and emphasizes how phases k and $M + k$ are twinned. Each pair of phases is like a 2-channel orthogonal filter bank! (On which the reader is by now an expert.) The good implementations have a delay chain of length $2M$ and decimators ($\downarrow M$)

coming first. They generate *two* blocks of length M in Figure 9.3, as the double-length DFT requires.

The cosine-modulated basis functions are *not* linear phase. They are even around $\omega = 0$ and odd around $\omega = \pi$, which produces Type-IV orthogonality. The first four basis functions when $M = 32$ are drawn in Section 8.3.

Figure 9.4a shows the idealized frequency response $P(e^{j\omega})$ of the lowpass prototype filter — the window p . Then Figure 9.4b shows the phase shifts from cosine modulation. Notice especially how each cosine (the sum of two exponentials) produces two copies of $P(e^{j\omega})$. One copy is shifted left and one copy is shifted right. A good prototype will have passband approximately $|\omega| \leq \frac{\pi}{2M}$. The design of that symmetric window subject to (9.2) gives a good M -channel filter bank — structured by cosine modulation.

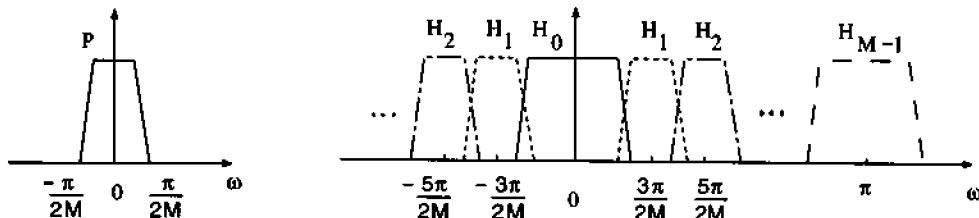


Figure 9.4: Idealized frequency response of the prototype and the cosine-modulated filters.

The original references for the discrete-time filter bank are in [Mr]. The key to the continuous-time construction was found by [Coifman-Meyer]. More recent references are in Section 9.4, where we also pursue the possibility of *biorthogonality*. In that case the analysis window $\tilde{p}(n)$ is dual to the synthesis window $p(n)$:

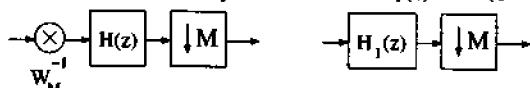
$$\tilde{p}(n) = \frac{p(n)}{p^2(n) + p^2(n+M)}.$$

This corresponds to $\tilde{g}(t) = g(t)/(g^2(t) + g^2(t+1))$ in continuous time.

It seems fair to say that the discrete-time construction is always a little more delicate. For orthogonality, we check sums instead of integrals. The sums often depend on equal spacing in time or frequency. Where time-varying windows and window lengths were no problem in Section 8.4, they are more difficult for discrete filters. But their potential value is so great that the effort is worth making.

Problem Set 9.1

1. Draw the DCT basis functions h_0, h_1, h_2, h_3 when $M = 8$. They are drawn in Section 8.3 for $M = 32$.
2. Suppose the prototype p in Figure 9.4 is an ideal brick wall. What are the coefficients $p(n)$ and what is $P(\omega)$?
3. Invent a prototype $p(n)$ that satisfies the PR conditions (9.2) and (9.3), with $M = 2$ and then general M . The filter has $N + 1$ coefficients.
4. Verify that the two systems below are equivalent where $H_1(z) = H(ze^{-j2\pi/M})$.



9.2 Polyphase Form: M Channels

Digital filter banks divide the signal into M subbands, then process the subbands and reconstruct. The analysis bank splits the input signal and the synthesis bank recombines it. The essential information is extracted from the subband signals in the *processing block*. Its form varies and depends on the applications. In an audio/video compression system, the spectral contents of the subband signals are coded depending on their energies. In a radar system, the subband signals might be used to null out a narrow-band interference adaptively. Other applications are image analysis and enhancement, robotics, computer vision, echo-cancellation, voice privacy and communications.

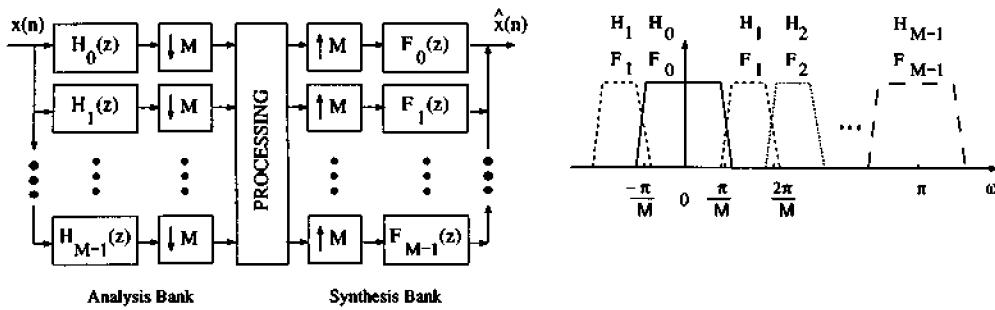


Figure 9.5: A maximally-decimated uniform filter bank with M channels.

Figure 9.5 illustrates a maximally decimated M -channel filter bank. The frequency responses $H_k(e^{j\omega})$ and $F_k(e^{j\omega})$ have passbands as shown. The analysis filters $H_k(z)$ channelize the input signal $x(n)$ into M subband signals, which are downsampled (decimated) by a factor M . At the receiving end, the M subband signals are decoded, interpolated and recombined by the synthesis filters $F_k(z)$. The decimator, which decreases the sampling rate, and the expander, which increases the rate, are denoted by $(\downarrow M)$ and $(\uparrow M)$.

Reconstruction Error

Since the filters $H_k(z)$ are not ideal, the filtered signals are not bandlimited to π/M . Therefore the aliases from downsampling will overlap. They depend on the stopband attenuation of $H_k(e^{j\omega})$ and their transition bands. The interpolated (upsampled) signals have M images of the compressed spectrum (assuming that no processing has been done after the analysis bank). These images are filtered by the synthesis filters $F_k(z)$.

There are two types of errors in the reconstructed output signal $\hat{x}(n)$: *distortions* (magnitude and phase) and *aliasing*. The nonideal filtering characteristics of $H_k(z)$ and $F_k(z)$ contribute to both distortion and aliasing. The changes in sampling rates (downsampling and upsampling) contribute to the aliasing error. A system with no aliasing error is “alias-free”. We compute the z-transform relation between input and output for $M = 3$:

- The analysis filters produce $H_0(z)X(z)$ and $H_1(z)X(z)$ and $H_2(z)X(z)$.
- $(\downarrow 3)$ and $(\uparrow 3)$ then produce two aliases in each channel $k = 0, 1, 2$:

$$\frac{1}{3} (H_k(z)X(z) + H_k(zW)X(zW) + H_k(zW^2)X(zW^2)) \quad \text{with } W = e^{-j2\pi/3}.$$

- The synthesis filters multiply those z -transforms by $F_k(z)$. Adding the nine terms, and collecting the terms for $X(z)$ and its aliases $X(zW)$ and $X(zW^2)$, the total output from the filter bank is

$$\widehat{X}(z) = T_0(z)X(z) + T_1(z)X(zW) + T_2(z)X(zW^2). \quad (9.4)$$

The perfect reconstruction conditions are $T_0(z) = z^{-\ell}$ (no amplitude distortion) and $T_1(z) = T_2(z) = 0$ (no aliasing). These three equations are written out explicitly in (9.10) below. For general M , the input-output relation is

$$\widehat{X}(z) = \sum_{k=0}^{M-1} T_k(z)X(zW^k) \quad \text{where} \quad T_k(z) = \frac{1}{M} \sum_{\ell=0}^{M-1} F_\ell(z)H_\ell(zW^k). \quad (9.5)$$

Since the transfer functions $T_1(z), T_2(z), \dots, T_{M-1}(z)$ multiply the shifted versions of the input spectrum, they are the *aliasing transfer functions*. The *distortion function* $T_0(z)$ multiplies the original spectrum. Then $T_0(z)X(z)$ is the output when all aliasing is cancelled. The objective is to find a set of PR filters $H_k(z)$ and $F_k(z)$:

$$\text{Perfect Reconstruction: } \begin{cases} T_0(z) = z^{-\ell} & (\text{no distortion for } k = 0) \\ T_k(z) = 0 & (\text{alias-free for } 1 \leq k < M). \end{cases} \quad (9.6)$$

Paraunitary and biorthogonal filter banks (with additional properties such as linear phase and cosine modulation) satisfy all conditions in (9.6). The conventional Pseudo-QMF bank cancels aliasing at adjacent bands ($T_1(z) = 0$). The “Near PR” banks have $T_0(z) = z^{-\ell}$ and $T_k(z)$ near zero. The DFT filter bank cancels all aliasing components, but suffers from distortions! It satisfies the last $M - 1$ conditions in (9.6) but not the first. In discussing a specific M -channel filter bank, one has to keep in mind its reconstruction properties.

Note: For a two-channel filter bank, perfect reconstruction requires

$$\begin{cases} T_0(z) = \frac{1}{2}[F_0(z)H_0(z) + F_1(z)H_1(z)] = z^{-\ell} \\ T_1(z) = \frac{1}{2}[F_0(z)H_0(-z) + F_1(z)H_1(-z)] = 0. \end{cases} \quad (9.7)$$

Solving for $F_k(z)$ yields the synthesis filters from the analysis filters:

$$F_0(z) = \frac{2z^{-\ell}}{\Delta(z)}H_1(-z) \quad \text{and} \quad F_1(z) = -\frac{2z^{-\ell}}{\Delta(z)}H_0(-z) \quad (9.8)$$

where $\Delta(z) = H_0(z)H_1(-z) - H_0(-z)H_1(z)$. An FIR system has ℓ delays:

$$\text{Determinant } \Delta(z) = 2z^{-\ell}, \quad F_0(z) = H_1(-z), \quad F_1(z) = -H_0(-z). \quad (9.9)$$

This simple relationship *does not extend* to filter banks with more channels. The perfect reconstruction equations for a three-channel bank are

$$\begin{cases} T_0(z) = F_0(z)H_0(z) + F_1(z)H_1(z) + F_2(z)H_2(z) = 3z^{-\ell} \\ T_1(z) = F_0(z)H_0(zW) + F_1(z)H_1(zW) + F_2(z)H_2(zW) = 0 \\ T_2(z) = F_0(z)H_0(zW^2) + F_1(z)H_1(zW^2) + F_2(z)H_2(zW^2) = 0 \end{cases} \quad (9.10)$$

where $W = e^{-j2\pi/3}$. Writing this system in matrix form, the solution for $F_k(z)$ is

$$\begin{bmatrix} F_0(z) \\ F_1(z) \\ F_2(z) \end{bmatrix} = \begin{bmatrix} H_0(z) & H_1(z) & H_2(z) \\ H_0(zW) & H_1(zW) & H_2(zW) \\ H_0(zW^2) & H_1(zW^2) & H_2(zW^2) \end{bmatrix}^{-1} \begin{bmatrix} 3z^{-\ell} \\ 0 \\ 0 \end{bmatrix}. \quad (9.11)$$

This inverse transpose of the modulation matrix $H_m(z)$ gives

$$\begin{bmatrix} F_0(z) \\ F_1(z) \\ F_2(z) \end{bmatrix} = \frac{3z^{-\ell}}{\Delta(z)} \begin{bmatrix} H_1(zW)H_2(zW^2) - H_1(zW^2)H_2(zW) \\ H_2(zW)H_0(zW^2) - H_2(zW^2)H_0(zW) \\ H_0(zW)H_1(zW^2) - H_0(zW^2)H_1(zW) \end{bmatrix} \quad (9.12)$$

where $\Delta(z)$ is the determinant of $H_m(z)$. The synthesis filter $F_k(z)$ depends on two filters $H_j(z)$ for $j \neq k$. This relationship complicates the design for M channels.

The number of parameters grows linearly with M . To simplify the design and implementation, explicit relations between the filters are often imposed:

Paraunitary Synthesis is time-reversed from analysis: $F_k(z) = z^{-N} H_k(z^{-1})$.

Linear Phase $H_k(z) = \pm z^{-N} H_k(z^{-1})$ (symmetric or antisymmetric).

DFT Filter Bank $H_k(z)$ comes from $H_0(zW^k)$.

Cosine Modulation $H_k(z)$ and $F_k(z)$ are DCT modulations of *one* prototype.

Pairwise Mirror Image $H_{M-1-k}(z) = H_k(-z)$ for odd M and $z^{-N} H_k(-z^{-1})$ for even M . The frequency responses are symmetric about $\frac{\pi}{2}$.

Each relation reduces the number of parameters by 2 (or by M , for DFT and cosine modulation). But the requirements might prevent our design methods from finding a good solution. The only two-channel linear-phase paraunitary filter bank is Haar's $H_0(z) = 1+z^{-1}$ and $H_1(z) = 1-z^{-1}$. By imposing both properties, we have limited our solution.

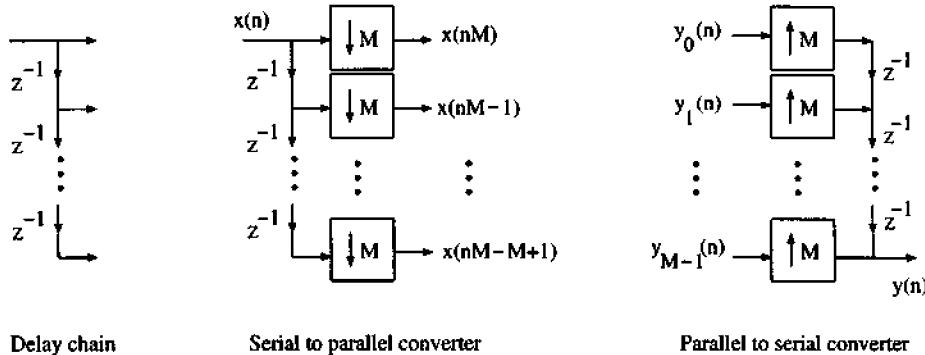
The choices depend on the applications. Cosine modulation is used in audio compression and telecommunications. It is efficient and has high stopband attenuation. Linear phase is important in image compression and signal detection.

Before developing filter banks for $M > 2$, we generalize the delay chain, the serial-parallel and parallel-serial converter, and the polyphase form.

Serial-Parallel and Parallel-Serial Converters

The figure below shows the block diagram of a delay chain. The transfer function from the input to the k th output is z^{-k} . By itself, the delay chain is not very interesting. However, cascading the delay chain with downsamplers or upsamplers will yield serial-parallel or parallel-serial converters.

Delay chain cascaded with downsamplers	=	Serial to Parallel (S/P) converter
Upsamplers cascaded with delay chain	=	Parallel to Serial (P/S) converter



The output at the k th branch of the S/P converter is $x(nM - k)$, which implies that the input sequence is selected in a *counter-clockwise* fashion. The branches select signals in the order $0, M - 1, M - 2, \dots, 2, 1, 0, M - 1, \dots$. When $x(n)$ is a causal signal, the output of the S/P converter ($M = 4$) is:

branch	0	$x(0)$	$x(4)$	$x(8)$	$x(12)$	\dots	
1	0	$x(3)$	$x(7)$	$x(11)$	\dots		output rate = (input rate)/ M
2	0	$x(2)$	$x(6)$	$x(10)$	\dots		
3	0	$x(1)$	$x(5)$	$x(9)$	\dots		

A P/S converter is a cascade of expanders and a *reverse-ordered* delay chain. The output $y(n)$ is an interleaved combination of the signals $y_k(n)$. Thus its rate is M times the rate of $y_k(n)$. The signals $y_k(n)$ are selected in a *clockwise* fashion. Assuming that $y_k(n)$ are causal, and $M = 3$, the serial output is

$$y_2(0) \ y_1(0) \ y_0(0) \ y_2(1) \ y_1(1) \ y_0(1) \ y_2(2) \ y_1(2) \ y_0(2) \ \dots$$

Polyphase Representation of a Filter Bank

An analysis filter $H_k(z)$ is the sum of M phases $H_{k,l}(z)$:

$$H_k(z) = \sum_{\ell=0}^{M-1} z^{-\ell} H_{k,\ell}(z^M), \quad h_{k,\ell}(n) = h_k(Mn + \ell). \quad (9.13)$$

The four phases ($M = 4$) of $1 + 3z^{-1} - 4z^{-2} + 7z^{-3} + 6z^{-4} - 3z^{-5} + z^{-6}$ are

$$\begin{cases} H_{k,0}(z) = 1 + 6z^{-1} \\ H_{k,1}(z) = 3 - 3z^{-1} \end{cases} \quad \begin{cases} H_{k,2}(z) = -4 + z^{-1} \\ H_{k,3}(z) = 7. \end{cases} \quad (9.14)$$

Consider a lowpass $H(z)$ of length 81. Figure 9.6(a) shows $|H(e^{j\omega})|$ with center frequency at $\frac{\pi}{4}$. The magnitudes (nearly constant) and phases (nearly linear) of the four components $H(z)$ are plotted in 9.6(b) and (c). The phase responses are such that $|H| \approx 1$ in the passband and $H \approx 0$ in the stopband. Since magnitudes are approximately equal, the phase angles accomplish this task. Figure 9.6(d) plots the offsets $\Delta_\ell(\omega) = -10\omega - \phi_\ell(\omega)$. We observe that $\Delta_\ell(\omega)$ is nearly $-\ell\omega/4$ (in general $-\ell\omega/M$). The polyphase components provide *fractional delays* so that $H(z)$ is a good lowpass filter.

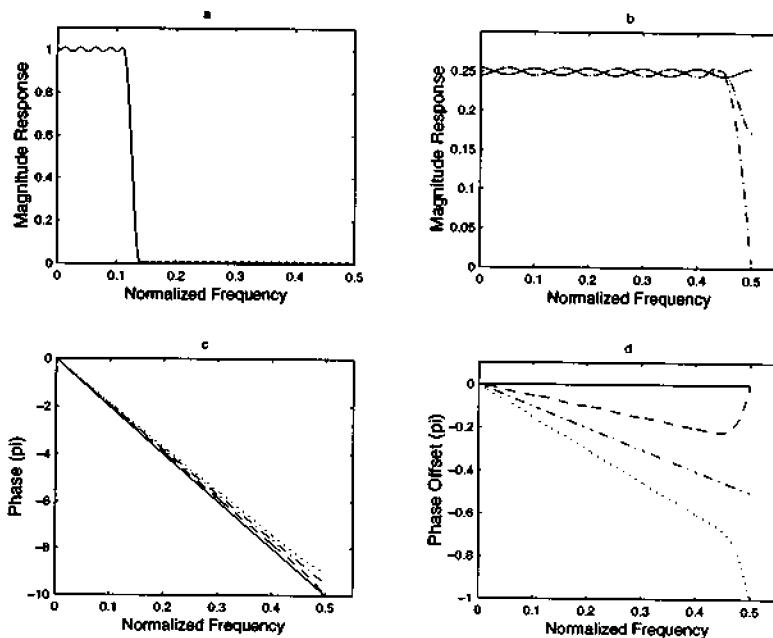


Figure 9.6: (a) Magnitude response of a lowpass filter, (b) Magnitude response of the polyphase filters, (c) Phase responses of the polyphase filters, (d) Phase offset.

There are M^2 polyphase components since each $H_k(z)$ has M components. Grouping these components in row k , the analysis transfer function is

$$\begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix} = \underbrace{\begin{bmatrix} H_{0,0}(z^M) & H_{0,1}(z^M) & \cdots & H_{0,M-1}(z^M) \\ H_{1,0}(z^M) & H_{1,1}(z^M) & \cdots & H_{1,M-1}(z^M) \\ \vdots & \vdots & \ddots & \vdots \\ H_{M-1,0}(z^M) & H_{M-1,1}(z^M) & \cdots & H_{M-1,M-1}(z^M) \end{bmatrix}}_{H_p(z^M)} \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-(M-1)} \end{bmatrix}. \quad (9.15)$$

$H_p(z)$ is the polyphase matrix of the analysis bank. The mapping between $h_k(n)$ and $H_p(z)$ is one to one. The impulse response of $H_{k,\ell}(z)$ is $h_{k,\ell}(n) = h_k(Mn + \ell)$.

Example 9.1. Suppose that the analysis filters of a 3-channel filter bank are

$$\begin{aligned} H_0(z) &= 1 + 2z^{-1} + 3z^{-2} + 4z^{-3} + 5z^{-4} + 6z^{-5} + 7z^{-6} \\ H_1(z) &= 1 - 2z^{-1} + 3z^{-2} - 4z^{-3} + 5z^{-4} - 6z^{-5} + 7z^{-6} \\ H_2(z) &= 1 + 2z^{-1} - 3z^{-2} + 4z^{-3} + 5z^{-4} - 6z^{-5} + 7z^{-6} \end{aligned}$$

The corresponding polyphase transfer matrix is

$$H_p(z) = \begin{bmatrix} 1 + 4z^{-1} + 7z^{-2} & 2 + 5z^{-1} & 3 + 6z^{-1} \\ 1 - 4z^{-1} + 7z^{-2} & -2 + 5z^{-1} & 3 - 6z^{-1} \\ 1 + 4z^{-1} + 7z^{-2} & 2 + 5z^{-1} & -3 - 6z^{-1} \end{bmatrix}.$$

The synthesis filter has a Type-II representation (phase ℓ before channel k):

$$F_k(z) = \sum_{\ell=0}^{M-1} z^{-(M-1-\ell)} F_{\ell,k}(z^M) \quad f_{\ell,k}(n) = f_k(Mn + M - 1 - \ell). \quad (9.16)$$

For $F_k(z) = 1 + 3z^{-1} - 4z^{-2} + 7z^{-3} + 6z^{-4} - 3z^{-5} + z^{-6}$, the four components that enter column k of the Type-II polyphase matrix $F_p(z)$ are

$$F_{0,k}(z) = 7 \quad F_{1,k}(z) = -4 + z^{-1} \quad F_{2,k}(z) = 3 - 3z^{-1} \quad F_{3,k}(z) = 1 + 6z^{-1}.$$

The corresponding synthesis bank can be rewritten as

$$\begin{bmatrix} F_0(z) \\ F_1(z) \\ \vdots \\ F_{M-1}(z) \end{bmatrix} = \underbrace{\begin{bmatrix} F_{0,0}(z^M) & F_{1,0}(z^M) & \cdots & F_{M-1,0}(z^M) \\ F_{0,1}(z^M) & F_{1,1}(z^M) & \cdots & F_{M-1,1}(z^M) \\ \vdots & \vdots & \ddots & \vdots \\ F_{0,M-1}(z^M) & F_{1,M-1}(z^M) & \cdots & F_{M-1,M-1}(z^M) \end{bmatrix}}_{F_p^T(z^M)} \begin{bmatrix} z^{-(M-1)} \\ z^{-(M-2)} \\ \vdots \\ 1 \end{bmatrix}. \quad (9.17)$$

Transposing, $F_p(z)$ is the polyphase transfer matrix of the synthesis bank. Using $H_p(z)$ (Type-I polyphase) and $F_p(z)$ (Type-II polyphase), one can redraw Figure 9.5(a) as Figure 9.7(a). Then decimators move to the left of $H_p(z^M)$ by the *Noble Identity*. Similarly the expanders move to the right of $F_p(z^M)$.

A few words on the implementation efficiency of Figure 9.7(b). The input is blocked into M vectors by a Serial/Parallel converter (implemented as cascade of delay chain and decimators). The blocks are filtered by $F_p(z)H_p(z)$ and then recombined using a Parallel/Serial converter. The total number of nonzero coefficients in $H_p(z)$ and $F_p(z)$ is the same as that in $H_k(z)$ and $F_k(z)$. The main difference is a more efficient rate of operation. The filtering in the polyphase form is done at the input rate divided by M .

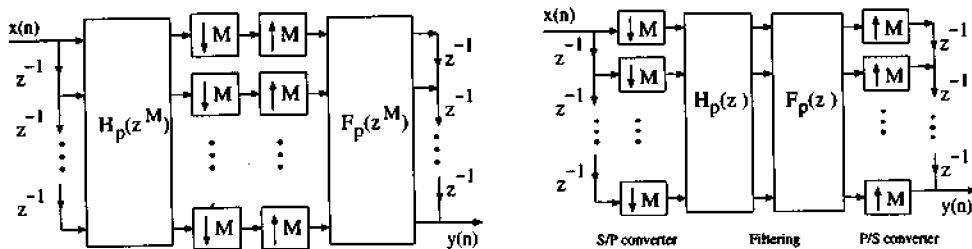


Figure 9.7: Polyphase representations of an M -channel uniform filter bank.

Modulation Matrix

Writing the PR conditions (9.6) in matrix-vector form displays the aliasing component matrix, which is the transposed modulation matrix $\mathbf{H}_m^T(z)$:

$$\begin{bmatrix} H_0(z) & H_1(z) & \cdots & H_{M-1}(z) \\ H_0(zW) & H_1(zW) & \cdots & H_{M-1}(zW) \\ \vdots & \vdots & \ddots & \vdots \\ H_0(zW^{M-1}) & H_1(zW^{M-1}) & \cdots & H_{M-1}(zW^{M-1}) \end{bmatrix} \begin{bmatrix} F_0(z) \\ F_1(z) \\ \vdots \\ F_{M-1}(z) \end{bmatrix} = M \begin{bmatrix} T_0(z) \\ T_1(z) \\ \vdots \\ T_{M-1}(z) \end{bmatrix}$$

The term *modulation* comes from the fact that row k of \mathbf{H}_m is obtained by modulating row zero (shift in frequency by $2\pi k/M$). The term *aliasing component* comes from the fact that row k of \mathbf{H}_m^T determines the aliasing transfer function $T_k(z)$ — which should be zero for $k > 0$. The theory of filter banks can be derived using either $\mathbf{H}_m(z)$ or $\mathbf{H}_p(z)$. Although they are equivalent, $\mathbf{H}_p(z)$ is preferred because it is used in the implementation.

To connect $\mathbf{H}_m(z)$ to $\mathbf{H}_p(z)$ we need two matrices. One is the diagonal delay matrix $\mathbf{D}(z) = \text{diag}(1, z^{-1}, \dots, z^{-(M-1)})$. The other is the M -point DFT matrix \mathbf{F}_M which gives the modulations:

$$\boxed{\text{Modulation and Polyphase } \mathbf{H}_m(z) = \mathbf{H}_p(z^M) \mathbf{D}(z) \mathbf{F}_M.} \quad (9.18)$$

Proof. The first row of $\mathbf{H}_m(z)$ contains the responses $H_0(zW^k)$ of the first filter. The first row of $\mathbf{H}_p(z^M)$ contains the phases H_{0j} of that same filter. To assemble phases of any function, we multiply by delays:

$$\begin{bmatrix} H_0(z) & H_0(zW) & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} = \begin{bmatrix} H_{00}(z^M) & H_{01}(z^M) & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots \\ z^{-1} & (zW)^{-1} & \cdots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

Factoring the delays $1, z^{-1}, \dots, z^{-(M-1)}$ from the rows of the last matrix leaves \mathbf{F}_M . The whole matrix multiplication becomes exactly $\mathbf{H}_m(z) = \mathbf{H}_p(z^M) \mathbf{D}(z) \mathbf{F}_M$.

The diagonal delay matrix is clearly paraunitary. The Fourier matrix \mathbf{F}_M/\sqrt{M} is unitary (thus paraunitary, but complex). Then (9.18) shows that $\mathbf{H}_p(z)$ is paraunitary when $\mathbf{H}_m(z)/\sqrt{M}$ is paraunitary.

Theorem 9.1 *The equivalent conditions in the z -domain for an orthogonal M -channel filter bank are*

1. *The polyphase matrix is paraunitary: $\mathbf{H}_p^T(z^{-1}) \mathbf{H}_p(z) = \mathbf{I}$.*
2. *The modulation matrix divided by \sqrt{M} is paraunitary: $\overline{\mathbf{H}}_m^T(z^{-1}) \mathbf{H}_m(z) = M\mathbf{I}$.*

It is satisfying that each condition has a direct proof, on its own. The direct proof for polyphase is to see the analysis bank ending with $\mathbf{H}_p(z)$, and the synthesis bank starting with $\mathbf{H}_p^T(z^{-1})$. That meeting produces \mathbf{I} at the center of the filter bank. In polyphase form, the whole bank generates blocks of M samples, filters with \mathbf{I} , and reconstructs the signal from the blocks. This is perfect reconstruction.

The direct proof of the modulation condition comes from following each channel instead of each phase. When we did this with synthesis filters F_k , the perfect reconstruction condition $T_k(z) = z^{-l} \delta(k)$ was

$$[F_0(z) \ F_1(z) \ \cdots \ F_{M-1}(z)] \mathbf{H}_m(z) = M[z^{-l} \ 0 \ \cdots \ 0]. \quad (9.19)$$

This is the top row of $\mathbf{F}_m(z)\mathbf{H}_m(z)$. The other rows come from modulating the F 's:

$$\begin{bmatrix} F_0(z) & F_1(z) & \cdots \\ F_0(zW) & F_1(zW) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} H_0(z) & H_0(zW) & \cdots \\ H_1(z) & H_1(zW) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} = M \begin{bmatrix} z^{-l} & & \\ & (zW)^{-l} & \\ & & \ddots \end{bmatrix}. \quad (9.20)$$

This is perfect reconstruction with causal filters. It applies to all cases—orthogonal or bi-orthogonal. In the orthogonal case the filters are $F_k(z) = z^{-l}H_k(z^{-1})$. So bring the diagonal matrix in (9.20) to the other side to find $\bar{\mathbf{H}}_m^T(z^{-1})\mathbf{H}_m(z) = M\mathbf{I}$. Note that \mathbf{H}_m has complex coefficients because of the powers of W . The polyphase matrix \mathbf{H}_p has real coefficients, because all filters are real.

Perfect Reconstruction: Polyphase Form

We have just given the perfect reconstruction condition in modulation form. It was written explicitly for $M = 3$ in equation (9.10), by following the signal through three channels. Equation (9.19) is the same statement for M channels. Equation (9.20) expresses that PR condition in terms of the matrices $\mathbf{H}_m(z)$ and $\mathbf{F}_m(z)$. Now we express perfect construction in terms of $\mathbf{H}_p(z)$ and $\mathbf{F}_p(z)$.

The reader will not be surprised by the result. The beauty of the polyphase form is the way it handles the algebra, for any number of filters.

Theorem 9.2 *An M -channel filter bank gives perfect reconstruction if*

$$\mathbf{F}_p(z)\mathbf{H}_p(z) = z^{-L}\mathbf{I}. \quad (9.21)$$

The overall delay of the system is $l = M - 1 + LM$, so that $T_0(z) = cz^{-l}$. The analysis and synthesis filters are biorthogonal.

A first proof begins with the modulation matrices in (9.20). Then the identity (9.18) converts to polyphase matrices. The result is the PR condition (9.21).

For a second proof, imagine the analysis-synthesis cascade with the polyphase matrices in the middle. The bank begins with an S/P converter and ends with a P/S converter. When the product of polyphase matrices is $z^{-L}\mathbf{I}$, the whole filter bank is a simple delay. This is perfect reconstruction.

Note: The most general case allows reordering of the channels. The product of the Type-I analysis matrix $\mathbf{H}_p(z)$ and the Type-II synthesis matrix $\mathbf{F}_p(z)$ is

$$\mathbf{F}_p(z)\mathbf{H}_p(z) = z^{-L} \begin{bmatrix} \mathbf{0} & I_{M-r} \\ z^{-1}I_r & \mathbf{0} \end{bmatrix}. \quad (9.22)$$

The overall delay is increased by r . We mention that z^{-1} is present below the diagonal whenever the system is alias-free; this gives the “pseudo-circulant” of Problem 4. For $r = 0$ we return to the fundamental case (9.21), when the product is $z^{-1}\mathbf{I}_M$.

Example 9.2. Suppose v is any unit column vector: $v^T v = 1$. Then choose

$$\mathbf{H}_p(z) = \mathbf{I} - vv^T + z^{-1}vv^T \quad \text{with} \quad \mathbf{H}_p^{-1}(z) = zvv^T + \mathbf{I} - vv^T. \quad (9.23)$$

You can verify immediately that the product is I . The matrix $H_p(z)$ gives an orthogonal analysis bank, and the causal matrix $F_p(z) = z^{-1}H_p^{-1}(z)$ gives the synthesis bank. These degree-one matrices are the building blocks for *all* paraunitary matrices. This is the great factorization theorem began by Belevitch and completed by Vaidyanathan [V, p. 273]:

$$\text{Every paraunitary } H_p(z) \text{ factors into } \left(\prod_{j=1}^L \left[I - v_j v_j^T + v_j v_j^T z^{-1} \right] \right) Q. \quad (9.24)$$

Example 9.3. Suppose $w^T v = 1$. Choose the analysis polyphase matrix

$$H_p(z) = I - vw^T + z^{-1}vw^T \quad \text{with} \quad H_p^{-1}(z) = zvw^T + I - vw^T. \quad (9.25)$$

Again the product is I . But H_p^{-1} is not H_p^T . The causal matrix $F_p(z) = z^{-1}H_p^{-1}(z)$ gives the *biorthogonal* synthesis bank (not orthogonal unless $v = w$). These degree-one matrices do *not* give building blocks for the most general biorthogonal filter banks. [PhVaid] have identified the correct subclass.

DFT Filter Banks

The analysis filters are all modulations $H_k(z) = H_0(zW^k)$ of the lowpass filter. The idealized responses are shown in Figure 9.8. Notice that the frequency allocations are very different from Figure 9.5. Similarly the synthesis filters are $F_k(z) = F_0(zW^k)$. All filters have linear phase if H_0 and F_0 have linear phase. But the filter coefficients are complex for $M > 2$, because $W = e^{-j2\pi/M}$ is complex.

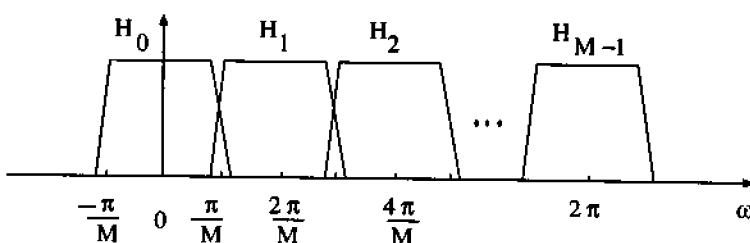


Figure 9.8: Idealized frequency response of the DFT filter bank.

This DFT filter bank is an excellent example for the use of polyphase matrices. Figure 9.2 showed the result of reordering the steps of subsampling and filtering. The lowpass filter coefficients $\mathbf{h}_0(n)$ on the left were separated into M phases on the right: $H_0(z) = E_0(z^M) + z^{-1}E_1(z^M) + \dots$. The polyphase matrix for the whole analysis bank is

$$H_p(z) = [\text{DFT}] \text{ diag}(E_0(z), E_1(z), \dots, E_{M-1}(z)). \quad (9.26)$$

The implementation of a DFT filter bank is shown in Figure 9.9. The input signal $x(n)$ is blocked into vectors of M components by the delay chain and downsamplers. The subband signals are then filtered by the polyphase components $E_\ell(z)$ of $H_0(z)$ and passed through an M -point DFT. The polyphase filters have real coefficients, so the inputs to the DFT are real. If the

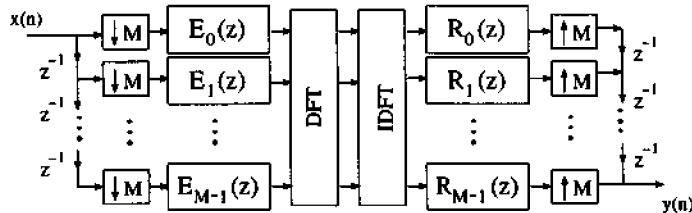


Figure 9.9: Polyphase representation of M -channel uniform DFT filter bank.

length of the lowpass filter is N , the computation load for the analysis bank is N multiplications and an M -point DFT for every M input samples.

The synthesis bank reverses these steps. We use the Type-II polyphase form of the lowpass filter $F_0(z) = R_{M-1}(z^M) + \dots + z^{-(M-1)}R_0(z^M)$. Then the polyphase matrix $F_p(z)$ has the IDFT multiplied by the diagonal matrix with entries $R_0(z), \dots, R_{M-1}(z)$. The product of analysis and synthesis is a *diagonal matrix* when the DFT and IDFT cancel:

$$F_p(z)H_p(z) = \text{diag}(R_0(z)E_0(z), R_1(z)E_1(z), \dots, R_{M-1}(z)E_{M-1}(z)). \quad (9.27)$$

This filter bank is alias-free if all diagonal elements are equal. It gives perfect reconstruction if those equal elements are pure delays:

$$\text{PR condition} \quad R_k(z)E_k(z) = z^{-L} \quad \text{for all } k. \quad (9.28)$$

But these filters $R_k(z) = z^{-L}E_k^{-1}(z)$ are IIR rather than FIR. For stability we would want each $E_k(z)$ to be minimum phase (zeros inside the unit circle). We also hope for linear phase. It is impossible to have both.

Theorem 9.3 [Nguyen-Vaidyanathan] *The polyphase components of a linear phase filter $H_0(z)$ cannot all have minimum phase.*

The anti-aliasing option is to choose FIR synthesis filters $R_k(z)$:

$$R_k(z) = E_0(z)E_1(z)\cdots E_{M-1}(z) \quad \text{omitting } E_k(z). \quad (9.29)$$

These filters are very long, about $M - 1$ times as long as the analysis filters. The products $R_k(z)E_k(z)$ are all equal and aliasing is cancelled. The distortion function $T_0(z)$ is the product with a delay:

$$T_0(z) = z^{-l}E_0(z)E_1(z)\cdots E_{M-1}(z). \quad (9.30)$$

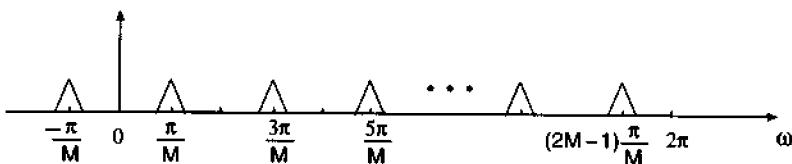
Perfect reconstruction is not possible. The natural question is whether one can design $H_0(z)$ such that either the magnitude or phase distortion is cancelled. Is there any choice of $H_0(z)$ such that T_0 is an allpass function (thus, no magnitude distortion) or a linear-phase function (thus, no phase distortion)? The answers are direct and beautiful:

If $H_0(z)$ has linear phase then $T_0(z)$ has linear phase.

If the components $E_k(z)$ are allpass then $T_0(z)$ is allpass.

In either case the output needs to be equalized. The equalizer for the allpass case should be an allpass filter that equalizes the phase distortion. The equalizer for the linear phase case should be a linear phase filter that equalizes the magnitude distortion.

For these alias-free filter banks, with synthesis filters from (9.29), we note that $H_0(z)$ and $F_0(z)$ cannot both be good lowpass filters. If they were good, there would be almost no overlap between $H_k(z)$ and $H_{k+2}(z)$, and between $F_k(z)$ and $F_{k+2}(z)$. Then the products $F_k(z)H_k(zW)$ do not overlap and their sum $T_1(z)$ cannot be zero! A typical $T_1(z)$ from good filters is drawn below:



Summary The polyphase method gives a direct approach to the analysis of DFT filter banks. Unfortunately, the results are almost all negative. We cannot even cancel aliasing. The DFT bank is fast, but not good in reconstruction. The DCT bank in Section 9.4 is also fast, and it reconstructs perfectly.

Properties of Paraunitary Filter Banks and Matrices

Power Complementary $\sum_{k=0}^{M-1} |H_k(e^{j\omega})|^2 = 1$ and $\sum_{k=0}^{M-1} H_k(z^{-1})H_k(z) = 1$.

This assures that there is no magnitude distortion: $T_0(z) = 1$. It does not determine the aliasing transfer functions. Thus paraunitary implies power complementary but power complementary does not imply paraunitary.

Time Reversal $F_k(z) = z^{-N}H_k(z^{-1})$

The magnitude responses satisfy $|F_k(z)| = |H_k(z)|$. $F_k(z)$ has maximum phase if $H_k(z)$ has minimum phase. If $H_k(z)$ is linear phase, then so is $F_k(z)$. The modulation matrix $H_m(z)$ and the AC matrix $H_p^T(z)$ are paraunitary. This follows from $H_m(z) = (\text{DFT})D(z)H_p^T(z^M)$.

Spectral Factor of M -th Band Filter The rows of $H_m(z)$ are paraunitary: $P_k = \tilde{H}_k H_k$ has

$$\sum_{l=0}^{M-1} \tilde{H}_k(zW^l)H_k(zW^l) = \sum_{l=0}^{M-1} P_k(zW^l) = 1.$$

Then P_k is an M -th band filter. Notice that $P_k(z)$ is linear phase by definition. Thus, $H_k(z)$ is a spectral factor of an M -th band linear phase filter.

Columnwise orthogonality The k th and ℓ th columns are orthogonal. The elements in column k are power complementary: $\sum_m H_{mk}(z^{-1})H_{mk}(z) = 1$.

Determinant is an allpass function Let $H(z)$ be a square paraunitary matrix and let $A(z)$ denote its determinant. Then $\tilde{H}(z)H(z) = I$ implies that $\tilde{A}(z)A(z) = 1$. Consequently, $A(z)$ is an allpass function. For an FIR paraunitary system, $A(z)$ is a delay.

Submatrices and cascades Any columns of a paraunitary matrix $H(z)$ give a paraunitary submatrix. Any cascade $H_1(z)H_2(z)$ is also paraunitary.

Problem Set 9.2

1. Show that $H = I - 2vv^T$ is a symmetric orthogonal matrix if $v^T v = 1$. This is a *Householder reflection* ($\det H = -1$) in the plane perpendicular to v . Show that $H = I - vv^T + z^{-1}vv^T$ is a paraunitary matrix.

2. What is the entry in row k , column l , of the modulation matrix $H_m(z)$?

3. Verify that $F_p(z)H_p(z) = z^{-1}I$ and find the coefficients in all four filters:

$$F_p(z) = \begin{bmatrix} 3+4z^{-1} & -3-2z^{-1} \\ -2-2z^{-1} & 2+z^{-1} \end{bmatrix} \quad H_p(z) = \begin{bmatrix} 2+z^{-1} & 3+2z^{-1} \\ 2+2z^{-1} & 3+4z^{-1} \end{bmatrix}.$$

Is this an orthogonal bank? Is $H_p(z)$ paraunitary? What are the determinants?

4. A two-channel bank is *alias-free* if $T_1(z) = F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0$. Verify that $F_m(z)H_m(z)$ is diagonal. Then substitute (9.18) to prove that $F_p(z)H_p(z)$ is a *pseudo-circulant matrix*:

$$F_m(z)H_m(z) = \begin{bmatrix} T_0(z) & \\ & T_0(-z) \end{bmatrix} \quad F_p(z)H_p(z) = \begin{bmatrix} T_{0,\text{even}} & T_{0,\text{odd}} \\ z^{-1}T_{0,\text{odd}} & T_{0,\text{even}} \end{bmatrix}.$$

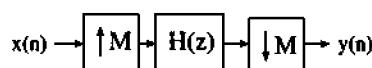
This pattern extends to all M -channel alias-free filter banks. When the aliasing functions T_1, \dots, T_{M-1} are zero, the product $F_m(z)H_m(z)$ is a diagonal matrix with entries $T_0(z), T_0(zW), \dots$. The product $F_p(z)H_p(z)$ is again a pseudocirculant. The prefix “pseudo” comes from the extra factor z^{-1} multiplying all entries below the main diagonal.

5. For the following analysis filters, find $H_p(z)$ and its determinant. Find synthesis filters $F_k(z)$ such that the overall system is PR.

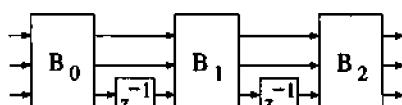
- (a) $H_0(z) = 1 + 3z^{-1} - 2z^{-2}, H_1(z) = z^{-1} + 2z^{-2}, H_2(z) = z^{-2}$
 (b) $H_0(z) = 1 + z^{-1} + z^{-2} + z^{-3} + 2z^{-4} - z^{-5}, H_1(z) = 1 + z^{-1} + z^{-2} + z^{-3} + 3z^{-4} + z^{-6} + 2z^{-7} - z^{-8}, H_2(z) = z^{-1} + z^{-2} - 2z^{-4} + z^{-5}$

6. Let $H_0(z) = 1 + z^{-1} + z^{-2} - 0.5z^{-3}, H_1(z) = z^{-1} + z^{-2} - 0.25z^{-4}$, and $H_2(z) = z^{-2}$. Find the PR synthesis filters in polyphase form $F_p(z)$ and modulation form $F_m(z)$ using (9.12).

7. Find the relation between $Y(z)$ and $X(z)$ below. Is it LTI? What is the system if $H(z)$ is an odd-length linear phase M -th band filter?

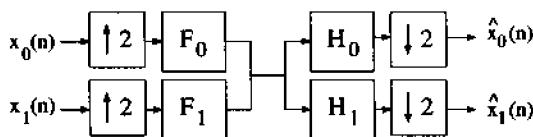


8. What is the polyphase matrix for a Serial/Parallel converter?
 9. Let $H_1(z)$ and $H_2(z)$ be paraunitary. Show that the cascade system $H_1(z)H_2(z)$ is paraunitary. How about the system $\frac{1}{2}(H_1(z) + H_2(z))$?
 10. Show that the 3×3 polyphase matrix is paraunitary, given that B_k are orthogonal matrices. Suppose the delays z^{-1} are replaced by the first-order allpass section $A(z) = \frac{z+z^{-1}}{1+az^{-1}}$. Find $H_p(z)$ and its determinant. Find the synthesis polyphase matrix $F_p(z)$ that cancels aliasing. Does PR synthesis exist?



11. A *transmultiplexer* is the reverse of a filter bank (synthesis first). Two or more signals $x_k(n)$ are multiplexed and sent over a high bandwidth channel. At the receiving end, the signal is demultiplexed. The outputs $\hat{x}_k(n)$ suffer from distortion and crosstalk. A PR transmultiplexer cancels crosstalk and reconstructs the signals $x_k(n)$ exactly.

- Express $\hat{X}_k(z)$ in terms of $X(z)$ and the filters. What are the conditions on $H_k(z)$ and $F_k(z)$ such that the transmultiplexer is PR?
- Suppose $(H'_k(z), F'_k(z))$ yield a PR filter bank, and a transmultiplexer is constructed by $H_k(z) = H'_k(z)$, $F_k(z) = z^{-1}F'_k(z)$. Show that this choice yields a PR transmultiplexer.



- Let $(H_k(z), F_k(z))$ be an M -channel PR filter bank. What are the exact conditions on L such that $(H_k(z^L), F_k(z^L))$ is also PR?
- Let $(H_k(z), F_k(z))$ be the filters in a paraunitary filter bank. Define $H'_k(z) = H_k(ze^{j\theta})$ and show that the filter bank is still paraunitary.

9.3 Perfect Reconstruction, Linear Phase, Orthogonality

There is much to recommend the time domain. The filter matrix H_b in block form — all M analysis channels together — is a *block Toeplitz matrix*. These matrices are easy to understand and analyze, if you remember that each entry is an $M \times M$ block. With filters of length $2M$, we have two blocks on each row as shown:

$$H_b = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ & \underline{h}(1) & \underline{h}(0) & \\ & \underline{h}(1) & \underline{h}(0) & \\ & \underline{h}(1) & \underline{h}(0) & \\ & & & \cdot \end{bmatrix} \quad (9.31)$$

This is the Lapped Orthogonal Transform (LOT) of Malvar. It is simply a filter bank with some overlapping but not much. We could have less overlapping or more:

- No overlapping: H_b is block diagonal = block transform (DFT and DCT).
- One overlap: H_b is block 2-diagonal = Lapped Orthogonal Transform (LOT).
- Filter lengths BM : H has B block diagonals = general case.

We are not introducing new filter banks. These are standard M -channel analysis banks, including the downsampling operation ($\downarrow M$). The only novelty is to interleave the channels, so that we watch all M channels at once. Previously $(\downarrow 2)H_0$ from the lowpass channel was written above $(\downarrow 2)H_1$ from the highpass channel:

$$\begin{bmatrix} (\downarrow 2)H_0 \\ (\downarrow 2)H_1 \end{bmatrix} = \begin{bmatrix} h_0(3) & h_0(2) & h_0(1) & h_0(0) \\ & h_0(3) & h_0(2) & h_0(1) & h_0(0) \\ & & \cdots & \cdots & \\ h_1(3) & h_1(2) & h_1(1) & h_1(0) & \\ & h_1(3) & h_1(2) & h_1(1) & h_1(0) \\ & & \cdots & \cdots & \end{bmatrix} \quad (9.32)$$

Interleaving rows gives our block Toeplitz matrix, with block size $M = 2$:

$$\underline{H}_b = \begin{bmatrix} \underline{h}_0(3) & \underline{h}_0(2) & \underline{h}_0(1) & \underline{h}_0(0) \\ \underline{h}_1(3) & \underline{h}_1(2) & \underline{h}_1(1) & \underline{h}_1(0) \\ & \underline{h}_0(3) & \underline{h}_0(2) & \underline{h}_0(1) & \underline{h}_0(0) \\ & \underline{h}_1(3) & \underline{h}_1(2) & \underline{h}_1(1) & \underline{h}_1(0) \\ & & \dots & \dots & \dots \\ & & \dots & \dots & \dots \end{bmatrix} \quad (9.33)$$

This is the time domain block form when $M = 2$ and $B = 2$. The filter lengths are $BM = 4$.

The z -transform is the polyphase matrix

$$\underline{H}_p(z) = \underline{h}(0) + z^{-1}\underline{h}(1) = \begin{bmatrix} \underline{h}_0(0) & \underline{h}_0(1) \\ \underline{h}_1(0) & \underline{h}_1(1) \end{bmatrix} + z^{-1} \begin{bmatrix} \underline{h}_0(2) & \underline{h}_0(3) \\ \underline{h}_1(2) & \underline{h}_1(3) \end{bmatrix}.$$

Note The block $\underline{h}(k)$ contains the k -th coefficient from each phase of each filter. The underbar in $\underline{h}(k)$ is used to emphasize that this $M \times M$ block is the coefficient of z^{-k} in $\underline{H}_p(z)$. When we form blocks, *the column order inside each block is to be reversed*. Thus $\underline{h}_0(0)$ is to the left of $\underline{h}_0(1)$ in the block, where it was to the right in (9.33). The orthogonality conditions on $\underline{h}(0)$ and $\underline{h}(1)$ are immediate in the time and z -domains:

Theorem 9.4 *The Lapped Orthogonal Transform (LOT) requires*

$$\begin{aligned} \underline{h}(0)^T \underline{h}(0) + \underline{h}(1)^T \underline{h}(1) &= I_{M \times M} \\ (\text{orthogonality of tails}) \quad \underline{h}(1)^T \underline{h}(0) &= \mathbf{0}_{M \times M} \end{aligned} \quad (9.34)$$

Also $\underline{H}_b \underline{H}_b^T = I$. This moves the transposes to the second factors in (9.34).

The general case with B blocks per row (and filter lengths BM) will produce B block equations from $\underline{H}_b^T \underline{H}_b = I$. You could say that these equations are “Condition O” in the time domain. The paraunitary requirements were “Condition O” in the polyphase domain and modulation domain. Notice the difference from the two-channel case. There Condition O was applied only to one filter \underline{H}_0 . The highpass filter \underline{H}_1 was determined by an alternating flip. Here, unless we impose a special structure on the bandpass filters $\underline{H}_1, \dots, \underline{H}_{M-1}$, we must include them all in Condition O.

The DFT filter bank does impose such a structure—but it makes orthogonality impossible (except in the block transform case $B = 1$ which is pure DFT with no filters). The DCT filter bank also imposes a structure. It produces all M filters from the knowledge of one filter. But this time, for the DCT bank, *orthogonality is possible*. In that case Condition O is no longer double-shift orthogonality, it is M -shift orthogonality.

For the most general LOT, multiply (9.34) by $\underline{h}(0)$ to find $\underline{h}(0)\underline{h}(0)^T \underline{h}(0) = \underline{h}(0)$. We freely use $\underline{h}(0)\underline{h}(1)^T = \mathbf{0}$ and $\underline{h}(0)^T \underline{h}(1) = \mathbf{0}$ to obtain

$$\begin{aligned} \underline{h}(0) &= \underline{h}(0)\underline{h}(0)^T(\underline{h}(0) + \underline{h}(1)) = \underline{P}\underline{Q} \\ \underline{h}(1) &= \underline{h}(1)\underline{h}(1)^T(\underline{h}(0) + \underline{h}(1)) = (\mathbf{I} - \underline{P})\underline{Q}. \end{aligned} \quad (9.35)$$

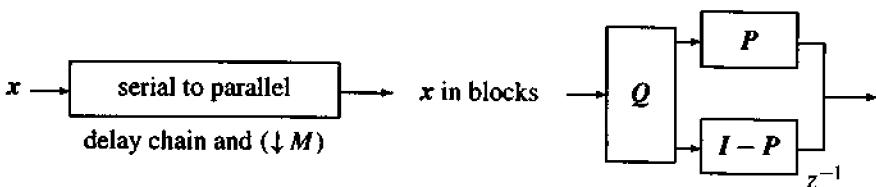
The matrix $\underline{Q} = \underline{h}(0) + \underline{h}(1)$ is orthogonal because (9.34) gives $\underline{Q}^T \underline{Q} = I$. The matrix $\underline{P} = \underline{h}(0)\underline{h}(0)^T$ is a projection matrix. It is symmetric and $\underline{P}^2 = \underline{h}(0)\underline{h}(0)^T \underline{h}(0)\underline{h}(0)^T =$

$\underline{h}(0)\underline{h}(0)^T = P$. Similarly, $\underline{h}(1)\underline{h}(1)^T$ is a projection, and the two projections add to I . The general solution (9.35) for the blocks in H_b was found by Heller and Tolimieri:

$$\underline{h}(0) = PQ \text{ and } \underline{h}(1) = (I - P)Q \text{ with } P = \text{projection}, Q = \text{orthogonal}.$$

The special case $P = I$ gives one block. Then H_b is block diagonal and it represents a simple orthogonal block transform. The general LOT can choose any projection matrix P and orthogonal matrix Q . Notice that $Q = H_p(1)$.

Note 1. The polyphase matrix has the convenient form $[P + (I - P)z^{-1}]Q$. This is fast if Q and P are fast. Below, we choose $Q = \text{DCT matrix}$:



Note 2. The *biorthogonal* lapped transform with $B = 2$ has a very similar pattern (Problem 2). The matrix $P = \underline{h}(0)\underline{f}(0)$ still equals P^2 . In this case, $P = P^T$ and $Q^TQ = I$ are not required. This gives the most general PR bank with filter lengths $2M$. The biorthogonal BOLT [PhVaid] extends this design to $B > 2$.

Note 3. For a paraunitary matrix $H_p(z)$, the degree of the determinant is the *Smith-McMillan degree*. When P has rank r , the degree is $L = M - r$. For the very special case of diagonal P , with r ones and $M - r$ zeros, the factors are clear:

$$H_p(z) = \begin{bmatrix} I_r & 0 \\ 0 & I_{M-r}z^{-1} \end{bmatrix} Q \text{ has determinant } z^{-(M-r)}.$$

Note 4. For fast implementation, Q often starts with the DCT matrix (the matrix C of cosines). The order M is even. Half the DCT rows and columns are symmetric, half are antisymmetric. It is natural to think of separating those parts and maintaining linear phase. Figure 9.10 does this separation by reordering the DCT rows to put C_{even} above C_{odd} . Then Malvar uses Haar butterflies and delays to separate $\underline{h}(1)$ from $\underline{h}(0)$. At the end we may apply different orthogonal matrices U and V . The blocks $\underline{h}(0)$ and $\underline{h}(1)$ are then

$$[\underline{h}(0) \quad \underline{h}(1)] = [U \quad V] \begin{bmatrix} C_e - C_o & (C_e - C_o)J \\ C_o - C_e & (C_e - C_o)J \end{bmatrix}. \quad (9.36)$$

The free parameters in this LOT are U and V . In practice, those must also allow fast multiplication. Malvar [Mr, p. 167] suggests $U = I$ and $V = \text{product of plane rotations or } V = \text{product of DST-IV and transposed DCT-II}$. The LOT is traditionally chosen to be linear phase and to start with the DCT-II.

Figure 9.11 shows the frequency responses of the LOTs. They are obtained by optimizing the coding gain (part a) and the stopband attenuation (part b). We show four basis functions (impulse responses) of the LOT that has high coding gain. Notice that they are symmetric and antisymmetric from the DCT. In summary, LOT is a linear phase paraunitary filter bank where the filter length is $2M$.

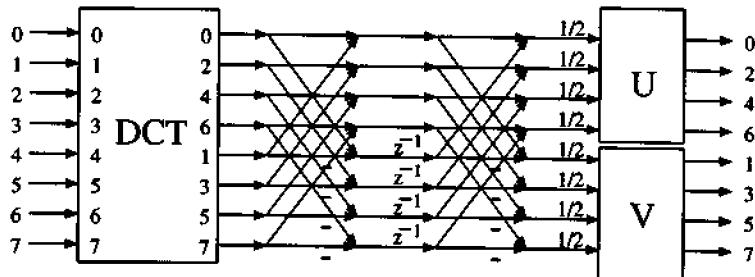


Figure 9.10: Polyphase transfer matrix of the Lapped Orthogonal Transform.

Longer Filters and GenLOT

You recognize that the two blocks ($B = 2$) of the LOT cannot give a sharp cutoff in frequency with great attenuation. The filters need to be longer. The time domain matrix H_b is always block Toeplitz when the input signal is blocked into M samples at a time. If the filter lengths are BM , there are B blocks in each row of H_b (and $B = 2$ for LOT):

$$H_b = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ & \underline{h}(B-1) & \cdots & \underline{h}(0) & \\ & \underline{h}(B-1) & \cdots & \underline{h}(0) & \\ & \cdots & \cdots & \cdots & \cdots \end{bmatrix}. \quad (9.37)$$

The polyphase matrix for this FIR analysis bank with decimators ($\downarrow M$) is

$$H_p(z) = \underline{h}(0) + \underline{h}(1)z^{-1} + \cdots + \underline{h}(B-1)z^{-(B-1)}. \quad (9.38)$$

Note again that all these blocks are $M \times M$. The conditions for orthogonality come directly from $H_b^T H_b = I$ and from $H_p^T(z^{-1}) H_p(z) = I$:

Theorem 9.5 *The analysis bank with filter length BM is orthogonal if*

$$\sum_{k=0}^{B-1-l} \underline{h}(k)^T \underline{h}(k+l) = \delta(l)I. \quad (9.39)$$

A block transform has $B = 1$ and only diagonal blocks in H_b . The constant polyphase matrix is $H_p(z) = \underline{h}(0)$. This gives bad results at block boundaries after compression. Overlapping blocks are much smoother but the filters have BM coefficients, which gives a huge design problem until we narrow it by structural decisions. One good decision is to start the filter bank with the DCT-II.

The simplest filters to follow the DCT are *block Haar* and *block diagonal and block delay*:

$$W = \frac{1}{\sqrt{2}} \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \text{ and } Q_i = \begin{bmatrix} U_i & 0 \\ 0 & V_i \end{bmatrix} \text{ and } D(z) = \begin{bmatrix} I & 0 \\ 0 & z^{-1}I \end{bmatrix}.$$

The blocks are of order $\frac{M}{2}$ and these matrices are orthogonal. Therefore, all products are paraunitary. The product also has linear phase, if Q at the beginning of the filter bank separates even and odd rows of the cosine matrix C^{II} (symmetric and antisymmetric cosine basis functions). The GenLOT is a cascade of these special filters:

$$\text{The GenLOT has } H_p(z) = (Q_{B-1} W D(z) W) \cdots (Q_1 W D(z) W) Q. \quad (9.40)$$

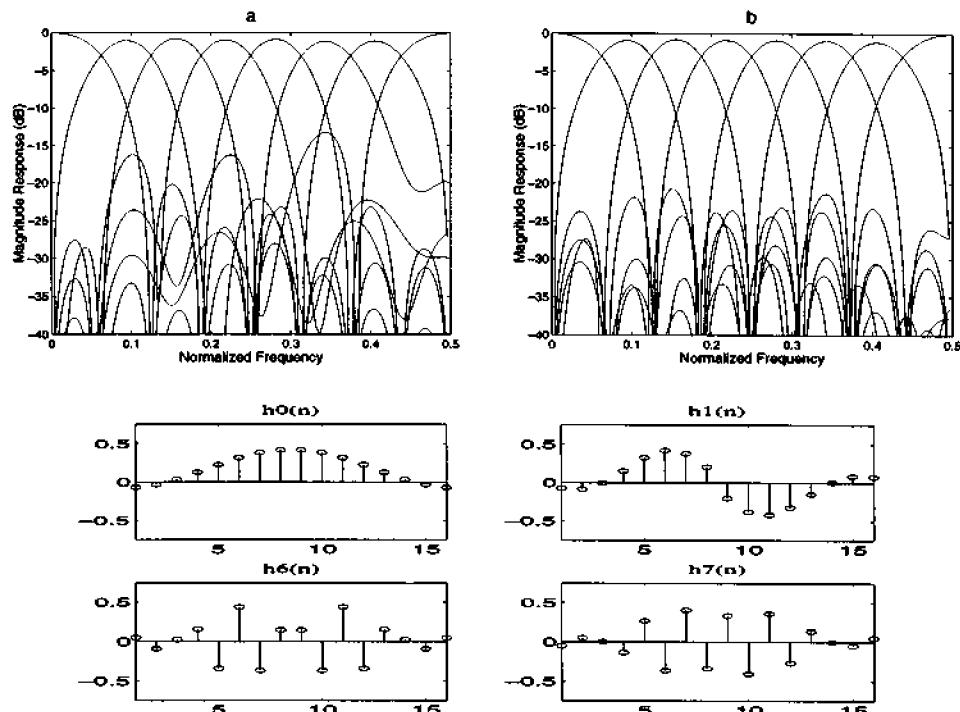


Figure 9.11: Frequency responses of the LOT. (a) High coding gain. (b) High stopband attenuation. Four of the eight symmetric-antisymmetric basis functions in (a).

The implementation flow graph for the analysis bank is in Figure 9.12 (see *GenLOT* in the Glossary for the details of K_1). The DCT-based LOT is the case with $B = 2$. The DCT itself is the case with $B = 1$ and $H_p = Q$. This framework covers all orthogonal filter banks with $\frac{M}{2}$ symmetric and antisymmetric channels — linear phase! The design problem is always to choose fast Q_i that also achieve sharp frequency discrimination.

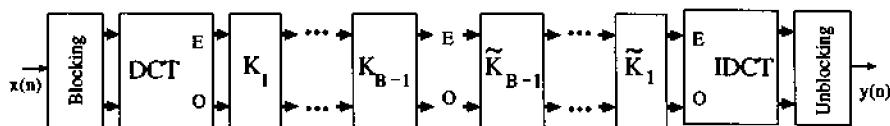


Figure 9.12: Polyphase transfer matrix of GenLOT.

Figure 9.13 shows the frequency responses of $H_k(z)$. The coding gain is 9.36 dB (left) and 23 dB attenuation (right). GenLOT design is included in

<http://saigon.ece.wisc.edu/~waveweb/QMF.html>

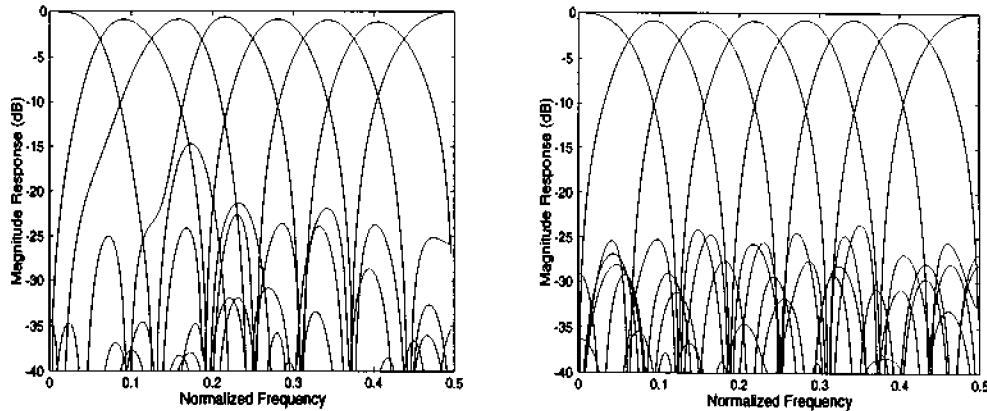


Figure 9.13: Frequency response plots for the 8-channel GenLOT (length 32) with high coding gain and high stopband attenuation.

Products of Rotations and Delays

Every $M \times M$ orthogonal matrix Q is a product of $\frac{1}{2}M(M - 1)$ plane rotations (*Givens rotations*). R_{ij} gives rotation by θ_{ij} in the plane of axes i and j :

$$R_{ij} = \begin{bmatrix} 1 & & & \\ & c & s & \\ & -s & c & \\ & & & 1 \end{bmatrix} \quad c = \cos(\theta_{ij}) \text{ and } s = \sin(\theta_{ij}) \text{ in rows and columns } i \text{ and } j.$$

The angles θ_{ij} can be design variables (but strongly nonlinear). The rotations in the GenLOT stay within even channels (to give U_i) or odd channels (to give V_i). The *Givens factorization* of any causal FIR paraunitary matrix is parallel to the *Householder factorization* in (9.24). Where Householder uses reflections, Givens uses a sequence of rotations:

$$\boxed{\mathbf{H}_p(z) = R_L D(z) \cdots R_2 D(z) R_1 D(z) Q.} \quad (9.41)$$

Here $D(z) = \text{diag}(1, 1, \dots, z^{-1})$ delays only the last channel. Its Smith-McMillan degree is 1, so the degree of $H_p(z)$ is L . Q is any unitary matrix. The matrices R_k can be chosen as *products of $M - 1$ rotations only*, in neighboring channels $(1, 2), (2, 3), \dots, (M - 1, M)$. The total number of parameters (plane rotations) is then $L(M - 1)$ plus $\frac{1}{2}M(M - 1)$ for Q . This is the same total as in the product (9.24) from Householder reflections.

Example 9.4. Consider a three-channel orthogonal filter bank whose polyphase transfer matrix has McMillan degree 2. Figure 9.14 shows the corresponding lattice structure. Each rotation is a simple butterfly connecting channels i and j . The total number of angles is $4 + 3 = 7$, and the filters have length 9.

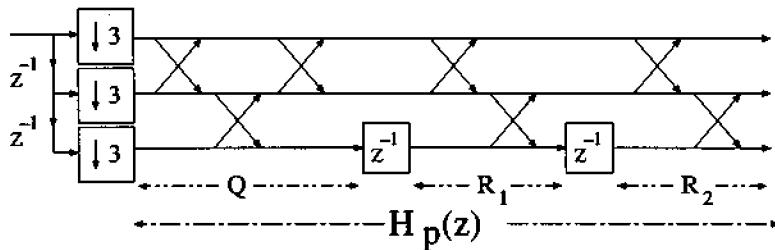


Figure 9.14: A three-channel paraunitary filter bank with McMillan degree 2.

Pairwise Mirror Image (PMI) Filter Banks

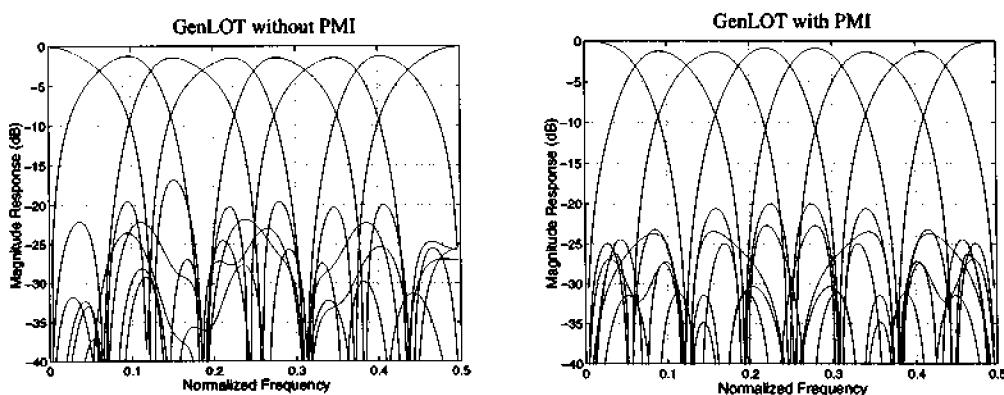
A further simplification of the GenLOT and other filter banks is to make each filter H_{M-1-k} a “mirror image” of H_k with respect to $\omega = \frac{\pi}{2}$:

$$|H_{M-1-k}(\omega)| = |H_k(\pi - \omega)|. \quad (9.42)$$

This reduces the number of design parameters by a factor of 2. When M is odd, say $M = 3$, the middle filter is its own mirror image and $H_1(z)$ is a function of z^2 . The other two channels have $H_2(z) = H_0(-z)$. There is a convenient lattice structure [NgVa3], and the PMI property is structurally imposed.

In the GenLOT this pairwise property connects the orthogonal matrices U_i and V_i . The polyphase matrix satisfies $JH_p(z) = H_p(z)\Gamma$, where $\Gamma = \text{diag}(1, -1, \dots, 1, -1)$. The matrices V_i are equal to $\Gamma U_i \Gamma$, except the last of the V 's is $JU_i \Gamma$.

We compare the frequency responses of two 8-channel GenLOT's with $B = 3$ and filter length 24. The second set of responses has the pairwise mirror image (PMI) property. It displays better stopband attenuation (measured in dB), which was the objective function in the design process.



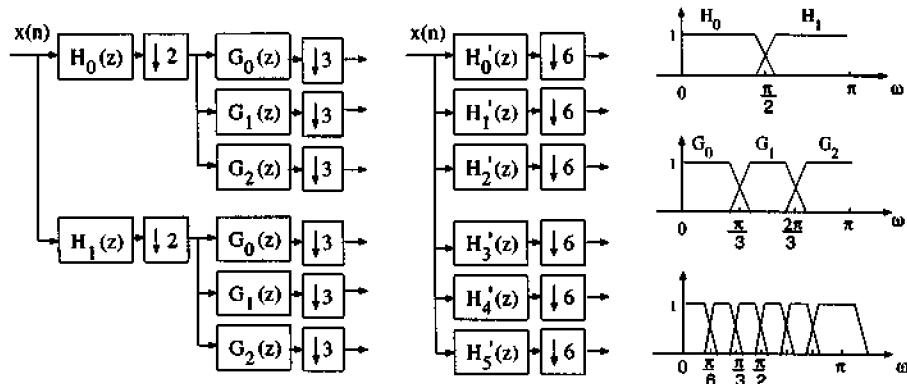
Permissible Lengths and Symmetry for Linear Phase

The filter lengths in GenLOT are the same and the number of symmetric and antisymmetric filters are equal. The conditions on the filter lengths $K, M + \beta$ and the symmetries for *biorthogonal* filter banks are summarized below [TranNg]:

Case	Symmetry/Antisymmetry	Lengths	Sum of Lengths
M even, β even	$\frac{M}{2}$ S & $\frac{M}{2}$ A	$\sum K_\ell$ is even	$2mM$
M even, β odd	$(\frac{M}{2} + 1)$ S & $(\frac{M}{2} - 1)$ A	$\sum K_\ell$ is odd	$2mM$
M odd, β even	$(\frac{M+1}{2})$ S & $(\frac{M-1}{2})$ A	$\sum K_\ell$ is odd	$(2m + 1)M$
M odd, β odd	$(\frac{M+1}{2})$ S & $(\frac{M-1}{2})$ A	$\sum K_\ell$ is even	$(2m + 1)M$

Tree-structured Filter Banks

A popular method to design M -channel filter banks is to cascade smaller systems. Wavelet packets use two-channel systems. The six-channel filter bank below is obtained from cascading a two-channel system with two three-channel systems:



Typical relations between $H'_k(z)$ and $H_k(z)$ and $G_k(z)$ are $H'_0(z) = H_0(z)G_0(z^2)$ and $H'_5(z) = H_1(z)G_2(z^2)$. The 6-channel filter bank is PR if and only if the 2-channel and 3-channel filter banks are PR.

One obtains *nonuniform* filter banks by cascading systems with different decimation factors [HoVaid]. A nonuniform bank with decimation factors (6, 6, 6, 4, 4) uses a 3-channel and a 2-channel system at the second tree level. This cascade is drawn in the Glossary under *Tree-Structured Filter Banks*. We emphasize the simplicity of these designs.

Summary The analysis bank is a block Toeplitz multiplication in the time domain. It is a polyphase matrix multiplication in the z -domain. That matrix contains the blocks from the Toeplitz form, just as $H(z)$ contains the coefficients from one filter. The polyphase matrix extends the familiar idea of $H(z)$ from one filter to M filters. This is the efficient order that gives the polyphase form:

- (direct time domain) Apply the analysis filters and then ($\downarrow M$).
- (more efficient order) Put the input in blocks and filter in parallel.
- (polyphase in z -domain) Multiply by $H_p(z) = \sum h_p(n)z^{-n}$.

For $M \times M$ blocks, the k th row contains coefficients from the k th filter. Column n has the M coefficients starting with $h_k(nM)$.

In the LOT case, with filter length $2M$ and two blocks per row, the first M coefficients $h_k(0), \dots, h_k(M-1)$ go into the main diagonal block $\underline{h}(0)$. The other coefficients $h_k(M), \dots, h_k(2M-1)$ go into the subdiagonal block $\underline{h}(1)$.

The synthesis bank is also a block Toeplitz matrix. The filter lengths and the number of blocks could be different, and the blocks in F_b are transposed. The coefficients from the k th synthesis filter are in the k th column of the blocks (not the k th row). Thus the two $M \times M$ blocks $\underline{f}(0)$ and $\underline{f}(1)$ in the length $2M$ case would be

$$\begin{bmatrix} f_0(0) & \cdots & f_{M-1}(0) \\ \vdots & & \vdots \\ f_0(M-1) & \cdots & f_{M-1}(M-1) \end{bmatrix} \text{ and } \begin{bmatrix} f_0(M) & \cdots & f_{M-1}(M) \\ \vdots & & \vdots \\ f_0(2M-1) & \cdots & f_{M-1}(2M-1) \end{bmatrix}.$$

The simplicity in the time domain resides in the fact that the whole filter bank becomes a matrix multiplication: $\hat{x} = F_b H_b x$. For $M > 2$ there is enough freedom to maintain orthogonality, rather than biorthogonality, while achieving other good properties. In this case we ask H_b to be orthogonal. We also ask for fast implementation, which leads us to GenLOT (with the DCT matrix). Cosine modulation is the alternative described next.

Problem Set 9.3

1. Show that $\underline{h}(0) = P Q$ and $\underline{h}(1) = (I - P)Q$ produce an orthogonal block Toeplitz H_b , for any P = symmetric projection matrix ($P^2 = P$) and Q = orthogonal matrix ($Q^T Q = I$).
2. When the synthesis bank is anti-causal, the PR condition is $F_b H_b = I$:

$$F_b = \begin{bmatrix} \underline{f}(0) & \underline{f}(1) & 0 \\ \underline{f}(0) & \underline{f}(1) & \cdot \\ \vdots & \vdots & \vdots \end{bmatrix} \text{ and } H_b = \begin{bmatrix} \underline{h}(0) & & \\ \underline{h}(1) & \underline{h}(0) & \\ 0 & \underline{h}(1) & \ddots \end{bmatrix}.$$

- (a) $F_b H_b = I$ and $H_b F_b = I$ give what six PR conditions on the blocks?
- (b) Deduce $f(0) = (f(0) + f(1))\underline{h}(0)f(0)$ and $\underline{h}(0) = \underline{h}(0)f(0)(\underline{h}(0) + \underline{h}(1))$. Write the corresponding equations for $f(1)$ and $\underline{h}(1)$.
- (c) Deduce also that $P = \underline{h}(0)f(0)$ equals P^2 and $\underline{h}(1)f(1)$ equals $I - P$.
- (d) Show finally that $\underline{f}(0) + \underline{f}(1)$ is the inverse of $Q = \underline{h}(0) + \underline{h}(1)$. Then PR for filter banks with $B = 2$ blocks has this form with $P^2 = P$:

$$\underline{f}(0) = Q^{-1}P, \underline{f}(1) = Q^{-1}(I - P), \underline{h}(0) = PQ, \underline{h}(1) = (I - P)Q.$$

3. Verify that $F_p(z)H_p(z) = I$ if $P^2 = P$ in the formulas

$$F_p(z) = Q^{-1}(P + (I - P)z) \text{ and } H_p(z) = (P + (I - P)z^{-1})Q.$$

4. If $P^2 = P$ has rank r , it can be diagonalized to give r ones and $M - r$ zeros. Show that $\det H_p(z) = z^{-(M-r)} \det Q$ which confirms that the filter bank is FIR.

5. By choosing 2×2 matrices P and Q in Problem 2 give examples of (a) an orthogonal bank and (b) a biorthogonal bank. What are the lowpass and highpass analysis coefficients in your examples?
6. Let $H_0(z) = \sum_{l=0}^{M-1} z^{-l} H_{0,l}(z^M)$ and $H_k(z) = H_0(z W^k)$. If $H_{0,l}(z)$ are allpass, the analysis filters $H_k(z)$ in this DFT bank are power complementary:

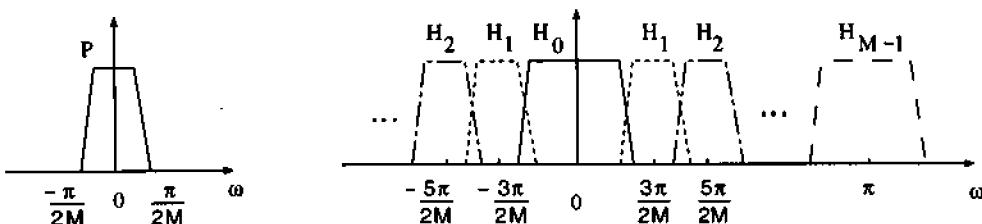
$$\begin{bmatrix} H_0(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix} = [\text{IDFT}] \begin{bmatrix} H_{0,0}(z^M) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & H_{0,M-1}(z^M) \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ z^{-(M-1)} \end{bmatrix}.$$

This is not PR! With $F_0(z) = H_0(z)$, the DFT bank has no distortion. It only has aliasing.

7. Let $\underline{h}_k(z)$ and $\underline{h}_\ell(z)$ be columns of a paraunitary matrix $H(z)$. They are mutually orthogonal. Show that the elements $H_{k,m}(z)$ in the k th column are power complementary filters:
 $\sum H_{k,m}(z^{-1}) H_{k,m}(z) = 1$.
8. We could design M linear phase M -th band filters $P_k(z)$ and find their spectral factors $H_k(z)$. What are the properties of this analysis bank? It is not necessarily PR. Does it have aliasing or amplitude or phase distortions?
9. Verify that the GenLOT in equation (9.40) has linear phase.
10. Find the relations between the analysis functions $H'_k(z)$ and their factors $H_{ij}(z)$ for the 5-channel system in the Glossary (Tree-structured Filter Bank). When the H_{ij} are ideal filters, sketch the ideal responses of $H'_k(z)$.
11. With filter lengths $3M$, the time domain matrix H_b will have blocks $\underline{h}(0), \underline{h}(1), \underline{h}(2)$ in each row. Write down the $B = 3$ equations for orthogonality, corresponding to the two equations in (9.34).

9.4 Cosine-modulated Filter Banks

The idea of modulation keeps developing and improving. It is a way to build the whole filter bank around one filter—the prototype filter $p(n)$. Instead of modulating by exponentials, we modulate by cosines. An exponential will shift the frequency in one direction. The cosine is a sum of two exponentials, so the frequency shift goes two ways. The frequency band is partitioned symmetrically by cosine modulation, as shown below.



The original construction [Rothweiler] was chosen to cancel aliasing between adjacent subbands. In the z -domain, only the first aliasing error $T_1(z)$ was exactly zero. See *Cosine-modulated Filter Bank* in the Glossary. The aliasing between other channels, such as k and $k+2$, will be small when the prototype response dies quickly in the stopband (good attenuation at $|\omega| = \frac{\pi}{2M}$). But now, with better designs, all aliasing is gone. The filter bank can give perfect reconstruction.

Another step forward is in the length of the filters. The maximum length was originally $2M$, in the Modulated Lapped Transform (MLT). Now the length can be $2KM$, in the Extended Lapped Transform (ELT). Those developments by Malvar [Mr] have parallels from other authors as the efficiency of cosine modulation is appreciated. A very active area is the construction of *time-varying filter banks*, in which the filter changes length as the signal passes through. Only with a simple basic design could we maintain orthogonality while the overlapping filters change with time.

Cosine modulation can be analyzed in two domains. In the time domain, certain constant matrices are *orthogonal*. In the z -domain, certain polynomial matrices are *paraunitary*. It seems right to do both, but perhaps to emphasize the time domain. We begin there with the orthogonality conditions on these filter coefficients:

$$h_k(n) = f_k(n) = p(n) \sqrt{\frac{2}{M}} \cos \left[(k + \frac{1}{2}) \left(n + \frac{M+1}{2} \right) \frac{\pi}{M} \right] \quad (9.43)$$

Does $p = (1, 1, \dots, 1)$ give the ordinary DCT block transform? No, because of the frequency shift. The M in the numerator of (9.43) changes the phase of the DCT-IV. It also affects the orthogonality. In fact the usual orthogonality is gone. We see this directly for $M = 2$, by comparing the matrix \mathbf{C}^{IV} with the new matrix \mathbf{E}_0 :

$$(\mathbf{C}^{\text{IV}})_{kn} = \cos \left[(k + \frac{1}{2})(n + \frac{1}{2}) \frac{\pi}{2} \right] = \begin{bmatrix} \cos(\frac{\pi}{8}) & \cos(\frac{3\pi}{8}) \\ \cos(\frac{3\pi}{8}) & \cos(\frac{5\pi}{8}) \end{bmatrix} = \begin{bmatrix} c & s \\ s & -c \end{bmatrix}$$

$$(\mathbf{E}_0)_{kn} = \cos \left[(k + \frac{1}{2})(n + \frac{3}{2}) \frac{\pi}{2} \right] = \begin{bmatrix} \cos(\frac{3\pi}{8}) & \cos(\frac{5\pi}{8}) \\ \cos(\frac{9\pi}{8}) & \cos(\frac{13\pi}{8}) \end{bmatrix} = \begin{bmatrix} s & -s \\ -c & c \end{bmatrix}$$

Here $c = \cos(\frac{\pi}{8})$ and $s = \sin(\frac{\pi}{8})$. Then immediately $(\mathbf{C}^{\text{IV}})^T (\mathbf{C}^{\text{IV}}) = \mathbf{I}$ and \mathbf{C}^{IV} is orthogonal. But \mathbf{E}_0 is not orthogonal:

$$\mathbf{E}_0^T \mathbf{E}_0 = \begin{bmatrix} s & -c \\ -s & c \end{bmatrix} \begin{bmatrix} s & -s \\ -c & c \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \mathbf{I} - \mathbf{J}. \quad (9.44)$$

Here and always \mathbf{J} is the reverse identity matrix. This convention is approaching the status of $\delta(n)$ and δ_{ij} , where definitions need not be repeated. Together with \mathbf{E}_0 in modulating a lapped transform comes the matrix \mathbf{E}_1 that saves orthogonality. With $M = 2$ this has the rest of the cosines:

$$(\mathbf{E}_1)_{kn} = \cos \left[(k + \frac{1}{2})(n + 2 + \frac{3}{2}) \frac{\pi}{2} \right] = \begin{bmatrix} \cos(\frac{7\pi}{8}) & \cos(\frac{9\pi}{8}) \\ \cos(\frac{11\pi}{8}) & \cos(\frac{13\pi}{8}) \end{bmatrix} \quad (9.45)$$

$$\mathbf{E}_1^T \mathbf{E}_1 = \begin{bmatrix} -c & -s \\ -c & -s \end{bmatrix} \begin{bmatrix} -c & -c \\ -s & -s \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \mathbf{I} + \mathbf{J} \quad (9.46)$$

$$\mathbf{E}_1^T \mathbf{E}_0 = \begin{bmatrix} -c & -s \\ -c & -s \end{bmatrix} \begin{bmatrix} s & -s \\ -c & c \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (9.47)$$

This pattern is extended in the Lemma below to all sizes M and to a longer sequence $\mathbf{E}_0, \dots, \mathbf{E}_{2K-1}$. Here, we stay with $M = 2$ and $K = 1$ to see orthogonality all the way. Our prototype filter will be $p = (1, 1, 1, 1)/\sqrt{2}$. Then the $2KM = 4$ coefficients in the $M = 2$ channels will be

the cosines in \mathbf{E}_0 and \mathbf{E}_1 divided by $\sqrt{2}$. The time-domain matrix \mathbf{H}_b , with the analysis channels interleaved to give the block diagonal form, has \mathbf{E}_0 along the main diagonal and \mathbf{E}_1 beside it:

$$\mathbf{H}_b = \frac{1}{\sqrt{2}} \begin{bmatrix} \cdot & & \\ \mathbf{E}_1 & \mathbf{E}_0 & \\ & \mathbf{E}_1 & \mathbf{E}_0 \\ & & \cdot & \cdot \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \cdot & \cdot & s & -s \\ -c & -c & s & -s \\ -s & -s & -c & c \\ & & -c & -c & s & -s \\ & & -s & -s & -c & c \\ & & & & & \cdot & \cdot \end{bmatrix} \quad (9.48)$$

Notice how double-shift orthogonality has become *block orthogonality* (and here $M = 2$). In the block form, this orthogonality comes from $\mathbf{E}_1^T \mathbf{E}_0 = 0$ off the diagonal and $\mathbf{E}_1^T \mathbf{E}_1 + \mathbf{E}_0^T \mathbf{E}_0 = 2I$ on the diagonal. The \mathbf{J} 's cancel in $I + \mathbf{J} + I - \mathbf{J}$. So we have an orthogonal filter bank.

The prototype filter p was symmetric, as we always assume. But cosine modulation shifted the phase. The individual filters $(-c, -c, s, -s)$ and $(-s, -s, -c, c)$ are not linear phase. And they do not have the accuracy of the Daubechies filters. *We do not get high accuracy from cosine modulation.* In this present form we do not even get $p = 1$! The lowpass filter applied to the alternating vector $x(n) = (-1)^n$ does not give zero output. Nevertheless, the coding gain is adequate for compression, and the implementation is extremely fast.

***M* Channels with Filter Lengths $2K\mathbf{M}$**

In general we have $2K$ matrices $\mathbf{E}_0, \dots, \mathbf{E}_{2K-1}$ and each matrix is $M \times M$:

$$(\mathbf{E}_\ell)_{kn} = \sqrt{\frac{2}{M}} \cos \left[(k + \frac{1}{2})(n + \ell M + \frac{M+1}{2}) \frac{\pi}{M} \right]. \quad (9.49)$$

These are the pieces of the extended cosine matrix $\mathbf{E} = [\mathbf{E}_0 \ \mathbf{E}_1 \ \cdots \ \mathbf{E}_{2K-1}]$, which would give the filter coefficients if the prototype filter had all $p(n) = 1$. We deal first with this case of a “box window”. Then we allow other windows and discover the orthogonality conditions on $p(n)$.

The rows of \mathbf{E} are orthogonal. Piece by piece we have the following identities.

Lemma: *The pieces of the extended cosine matrix \mathbf{E} satisfy*

$$\begin{aligned} \mathbf{E}_{2s} &= (-1)^s \mathbf{E}_0 & \text{and} & \mathbf{E}_\ell^T \mathbf{E}_{\ell+2s} &= (-1)^s [I + (-1)^{\ell+1} \mathbf{J}] \\ \mathbf{E}_{2s+1} &= (-1)^s \mathbf{E}_1 & & \mathbf{E}_\ell^T \mathbf{E}_{\ell+2s+1} &= 0. \end{aligned} \quad (9.50)$$

Proof: With $\ell = 2s$, we have added $(k + \frac{1}{2})(2sM) \frac{\pi}{M}$ to the arguments of the cosine in (9.49). This is the same as adding $s\pi$ which multiplies the cosine by $(-1)^s$. Similarly \mathbf{E}_{2s+1} is a shift by $s\pi$ from \mathbf{E}_1 . The two identities on the left are proved. The identities on the right then quickly reduce to the first cases

$$\mathbf{E}_0^T \mathbf{E}_0 = I - \mathbf{J} \quad \text{and} \quad \mathbf{E}_1^T \mathbf{E}_1 = I + \mathbf{J} \quad \text{and} \quad \mathbf{E}_1^T \mathbf{E}_0 = 0. \quad (9.51)$$

Those were checked above for $M = 2$. For larger M , the (n, \hat{n}) entry of $\mathbf{E}_0^T \mathbf{E}_0$ is

$$\begin{aligned} \frac{2}{M} \sum_{k=0}^{M-1} \cos \left[(k + \frac{1}{2})(n + \frac{M+1}{2}) \frac{\pi}{M} \right] \cos \left[(k + \frac{1}{2})(\hat{n} + \frac{M+1}{2}) \frac{\pi}{M} \right] &= \\ \sum_{k=0}^{M-1} \frac{1}{M} \cos \left[(k + \frac{1}{2})(n - \hat{n}) \frac{\pi}{M} \right] + \frac{1}{M} \cos \left[(k + \frac{1}{2})(n + \hat{n} + M + 1) \frac{\pi}{M} \right]. \end{aligned} \quad (9.52)$$

On line two, we wrote $(\cos a)(\cos b)$ using $a+b$ and $a-b$. Its first sum is $\delta(n-\dot{n})$, which gives the matrix I . Its second sum is zero, except when $n+\dot{n}=M-1$ and all terms are $\cos \pi = -1$. This gives the matrix $-J$ in $E_0^T E_0$. Note that the reverse identity has 1's when $n+\dot{n}=M-1$, because the numbering starts at zero. The identities $E_1^T E_1 = I + J$ and $E_1^T E_0 = 0$ have similar proofs.

Orthogonality Conditions on the Prototype Filter $p(n)$

Now multiply the cosines by $p(n)$ to get the true filter coefficients $h_k(n)$. The cosine matrix E is M by $2KM$, with columns indexed by n . It is multiplied by the diagonal matrix $P = \text{diag}(p(0), p(1), \dots, p(2KM-1))$. Their product EP splits into $2K$ square blocks $E_\ell \underline{P}_\ell$ of size M :

$$EP = [E_0 \cdots E_{2K-1}] \begin{bmatrix} \underline{P}(0) & & \\ & \ddots & \\ & & \underline{P}(2K-1) \end{bmatrix} = [E_0 \underline{P}_0 \cdots E_{2K-1} \underline{P}_{2K-1}] \quad (9.53)$$

The entries of EP are cosines multiplied by $p(n)$. In other words we have $h_k(n)$.

The square block \underline{P}_ℓ contains the prototype coefficients $p(\ell M), \dots, p(\ell M + M - 1)$. We are simply writing out a typical block row of the analysis bank matrix H_b . That row is EP as in (9.53), except the columns come in reverse order because $h_k(0)$ is at the right end of the row. This has no effect on orthogonality.

Under what condition is H_b an orthogonal matrix? The center block of $H_b^T H_b$ must be the identity and the off-diagonal blocks of $H_b^T H_b$ must be zero:

$$\sum_0^{2K-1} (\underline{E}_\ell \underline{P}_\ell)^T (\underline{E}_{\ell+i} \underline{P}_{\ell+i}) = \delta(i)I. \quad (9.54)$$

Inside those sums are the products $\underline{E}_\ell^T \underline{E}_{\ell+i}$ that are given by the Lemma. Substituting the identity (9.50) yields the orthogonality condition on the matrices \underline{P}_ℓ . That gives us the coefficients $p(n)$ of the (symmetric) prototype filter.

Theorem 9.6 *The cosine modulated filter bank is orthogonal if and only if the diagonal blocks \underline{P}_ℓ are double-shift orthogonal:*

$$\sum \underline{P}_\ell^T \underline{P}_{\ell+2s} = \delta(s)I. \quad (9.55)$$

For the coefficients this means that we shift by double blocks:

$$\sum_{\ell=0}^{2K-2s-1} p(n + \ell M) p(n + \ell M + 2sM) = \delta(s) \text{ for } n = 0, \dots, \frac{M}{2} - 1. \quad (9.56)$$

The special case $K = 1$ has only two blocks \underline{P}_0 and \underline{P}_1 . So there is only $s = 0$:

$$p^2(n) + p^2(n + M) = 1. \quad (9.57)$$

That corresponds exactly to the condition $g^2(t) + g^2(t+1) = 1$ on the window function in Section 8.4. That continuous-time theory only allowed neighboring windows to overlap. In discrete time this is $K = 1$.

The next discrete case $K = 2$ has blocks $\underline{P}_0, \underline{P}_1, \underline{P}_2, \underline{P}_3$. Now there are conditions from $s = 0$ and $s = 1$:

$$\begin{aligned} p^2(n) + p^2(n+M) + p^2(n+2M) + p^2(n+3M) &= 1 \\ p(n)p(n+2M) + p(n+M)p(n+3M) &= 0. \end{aligned} \quad (9.58)$$

We can design $4M$ lowpass coefficients to satisfy these conditions!

Proof: Theorem 9.6 is our main result. The analysis of cosine modulation led here. The conditions (9.54) for odd i are automatic because $\underline{E}_\ell^T \underline{E}_{\ell+i} = 0$ in (9.50). The conditions for even $i = 2s$ require the substitution of $(-1)^s [I + (-1)^{\ell+1} J]$ for $\underline{E}_\ell^T \underline{E}_{\ell+2s}$. The J 's cancel when we sum on ℓ because the prototype filter is symmetric. (You see why the filter length $2KM$ is an even multiple of M , to have an even number of J 's with alternating sign.) Then (9.55) and (9.56) reduce to $\sum \underline{P}_\ell^T \underline{P}_{\ell+2s}$. For orthogonality this must give $\delta(s)$ in (9.57) and (9.58).

Two symmetric solutions to $p^2(n) + p^2(n+M) = 1$ are the sine windows

$$p(n) = \sin \left[\frac{n\pi}{2(M-1)} \right] \text{ and } p(n) = -\sin \left[\left(n + \frac{1}{2} \right) \frac{\pi}{2M} \right]. \quad (9.59)$$

Malvar observed that the latter is the only prototype for which the resulting n th filter has frequency response $\delta(n)$ at $\omega = 0$. (The accuracy is $p = 1$.) The zeroth filter reproduces a DC input and the other filters null it, with no DC leakage. This normalization was expected in the rest of the book and is desirable in image coding. Malvar also gives a family of solutions when $K = 2$.

Polyphase and Lattice Structure

The polyphase coefficients are the blocks $\underline{E}_0 \underline{P}_0, \dots, \underline{E}_{2K-1} \underline{P}_{2K-1}$ along each row of the time domain matrix \underline{H}_b . The filter bank is orthogonal when this polyphase matrix $\sum \underline{E}_\ell \underline{P}_\ell z^{-\ell}$ is paraunitary:

$$\underline{H}_b^T(z^{-1}) \underline{H}_b(z) = \left(\sum \underline{P}_\ell^T \underline{E}_\ell^T z^\ell \right) \left(\sum \underline{E}_\ell \underline{P}_\ell z^{-\ell} \right) = I. \quad (9.60)$$

The constant term in the product is $\sum \underline{P}_\ell^T \underline{E}_\ell^T \underline{E}_\ell \underline{P}_\ell$. Equation (9.55) makes this the identity matrix I . The coefficient of z^{-i} in the product is $\sum \underline{P}_\ell^T \underline{E}_\ell^T \underline{E}_{\ell+i} \underline{P}_{\ell+i}$. Equation (9.55) makes this the zero matrix. Under these conditions $\underline{H}_b(z)$ is paraunitary.

For $K = 1$ the polyphase matrix simplifies to $\underline{E}_0 \underline{P}_0 + z^{-1} \underline{E}_1 \underline{P}_1$. It is paraunitary, after using $\underline{E}_1^T \underline{E}_0 = 0$ and $\underline{E}_1^T \underline{E}_1 = I + J$ and $\underline{E}_0^T \underline{E}_0 = I - J$, when

$$\underline{P}_1^T \underline{P}_1 + \underline{P}_0^T \underline{P}_0 = I. \quad (9.61)$$

This pairs off the prototype filter coefficients into equation (9.57).

Let $G_k(z), 0 \leq k \leq 2M-1$ be the polyphase components of the prototype filter $P(z)$. The pairs $(G_k(z), G_{M+k}(z))$ of a paraunitary cosine-modulated filter bank satisfy

$$G_k(z^{-1}) G_k(z) + G_{M+k}(z^{-1}) G_{M+k}(z) = \frac{1}{2M}. \quad (9.62)$$

This agrees with the orthogonality condition in a two-channel filter bank. The conditions (9.62) extend to filters of any length [NgKoil]. Thus, the cosine-modulated bank can be implemented by two-channel paraunitary filter banks in parallel, as depicted in the Glossary (see *Tree-structured Filter Bank*). This is efficient because of the lattice structures associated with the pair $[G_k(z), G_{k+M}(z)]$ and the matrix \widehat{C} .

Example 9.5. Figure 9.15 shows the frequency responses of an 8-channel PR cosine-modulated filter bank. The filter length is 128. The resulting filters have stopband attenuation about 80 dB. The design procedure is based on QCLS. Chapter 10 will elaborate more on this formulation.

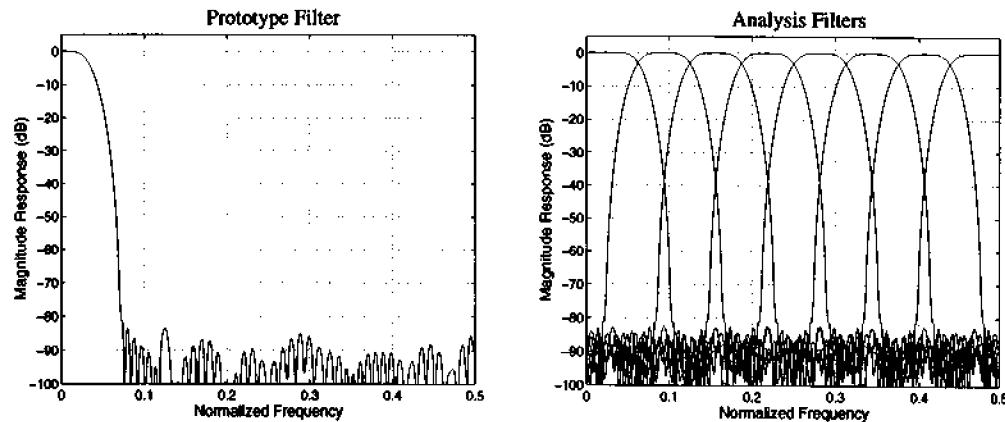


Figure 9.15: Frequency response plots for an 8-channel PR cosine-modulated filter bank.

Note 1 A pseudo-QMF bank is a cosine-modulated filter bank where the first aliasing transfer function $T_1(z)$ is cancelled. The other $T_2(z), \dots, T_{M-1}(z)$ are minimized by constraining the stopband cutoff frequency $\omega_c \leq \frac{\pi}{M}$ of $P(z)$ such that there is no overlap between $H_k(z)$ and $H_{k\pm 2}(z)$. Consequently, the aliasing error is comparable to the stopband attenuation of the prototype filter. The distortion function $T_0(z)$ is not a delay. The design procedure is to find a prototype filter that minimizes the following weighted objective function:

$$\alpha \int_{\omega_s}^{\pi} |P(e^{j\omega})|^2 d\omega + (1 - \alpha) \int_0^{\pi} |T_0(e^{j\omega}) - e^{-jn_0\omega}|^2 d\omega$$

where $0 \leq \alpha \leq 1$. One often has to trade off aliasing with distortion.

Note 2 The prototype filter $P(z)$ for an NPR Pseudo-QMF bank is a *linear phase spectral factor of a $2M$ -th band filter*. There is no distortion. The aliasing errors can be minimized by high attenuation.

Example 9.6. Figure 9.16 shows the frequency responses of a 16-channel NPR cosine-modulated filter bank. The filter length is 256. By constraining $P(z)$ to be the spectral factor of a 32-band filter, the resulting filter bank has no magnitude nor phase distortion. The only distortion here is the aliasing, which is less than or equal to -72 dB.

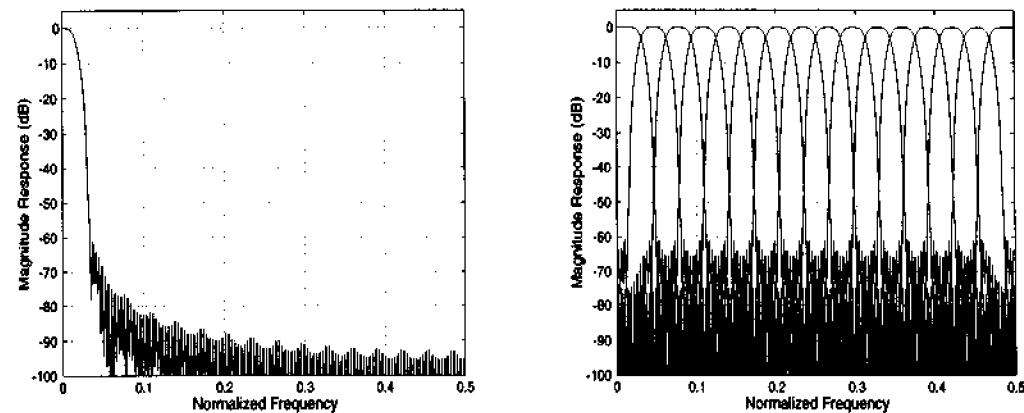


Figure 9.16: Frequency response plots for the 16-channel NPR cosine-modulated filter bank. Prototype filter $P(z)$ (left) and Analysis filters $H_k(z)$ (right).

Problem Set 9.4

1. Verify that Malvar's windows (9.59) satisfy $p^2(n) + p^2(n + M) = 1$ for orthogonality.
2. Construct explicitly (by hand or by Matlab) the 3×3 matrices C^T, E_0, E_1, E_2 for $M = 3$. Check by Matlab whether the identities (9.51) still hold for odd M .
3. For prototypes $p(n)$ in synthesis and $\tilde{p}(n)$ in analysis (so that $H_k \neq F_k$), what will be the conditions for a perfect reconstruction cosine-modulated filter bank (filter length $2M$ and eventually $2KM$)?
4. Let $p(n)$ have linear phase. Show that $h_k(n)$ in (9.43) cannot have linear phase.
5. Let $p(n)$ be a spectral factor of a $2M$ -th band filter. Show that the distortion function $T_0(z) = \sum_{k=0}^{M-1} F_k(z)H_k(z)$ is a delay.
6. Find the conditions on the prototype filters $p(n)$ for a *biorthogonal* cosine-modulated filter bank. Cheung's MIT thesis has shown that with symmetry, there are $M/2$ extra free parameters compared to the orthogonal case.

9.5 Multidimensional Filters and Wavelets

Images are two-dimensional. Processing those images is an extremely important application of subband filtering. We certainly need two-dimensional filters! Their construction is coming late in the book because it is either quite easy or quite hard --- depending on the type of filter:

1. *Separable*: Products of one-dimensional filters (easy).
2. *Nonseparable*: Genuinely two-dimensional filters (hard).

Each filter bank is associated with a subsampling matrix M . In d dimensions this matrix is $d \times d$. In one dimension it contains the usual number M . In two dimensions we consider the two leading possibilities:

$$(\text{separable}) \quad M_s = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad (\text{nonseparable quincunx}) \quad M_q = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}.$$

These matrices have eigenvalues $|\lambda| > 1$, as required. Their determinants are $M_s = 4$ and $M_q = 2$. The lightface symbol M still represents the number of channels. M is also the number of scaling functions plus wavelets.

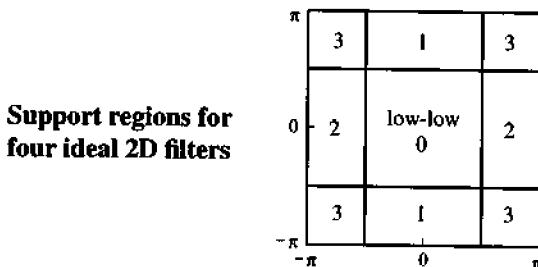
In two dimensions, a filter is a two-dimensional convolution $y = h * x$. In the $\omega = (\omega_1, \omega_2)$ and $z = (z_1, z_2)$ domains, we multiply by $H(\omega_1, \omega_2)$ and $H(z_1, z_2)$:

$$Hx(n_1, n_2) = \sum_{k_1} \sum_{k_2} h(k_1, k_2) x(n_1 - k_1, n_2 - k_2)$$

$$H(\omega) X(\omega) = \left(\sum \sum h(k_1, k_2) e^{-i(k_1\omega_1 + k_2\omega_2)} \right) \left(\sum \sum x(n_1, n_2) e^{-i(n_1\omega_1 + n_2\omega_2)} \right)$$

$$H(z) X(z) = \left(\sum \sum h(k_1, k_2) z_1^{-k_1} z_2^{-k_2} \right) \left(\sum \sum x(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \right).$$

The frequency response $H(\omega_1, \omega_2)$ has period 2π in both variables. A set of four brick wall filters will cover the period square with no overlap. These ideal filters are (0) low-low, (1) low-high, (2) high-low, and (3) high-high:



Those ideal filters are IIR. A set of four FIR separable filters is easily constructed from one-dimensional filters h_{low} and h_{high} :

$$h_0(n_1, n_2) = h_{low}(n_1) h_{low}(n_2) \quad h_2(n_1, n_2) = h_{high}(n_1) h_{low}(n_2)$$

$$h_1(n_1, n_2) = h_{low}(n_1) h_{high}(n_2) \quad h_3(n_1, n_2) = h_{high}(n_1) h_{high}(n_2).$$

This is the easy construction. After the filters, we sample by $(\downarrow 2)$ in each direction. This subsampling corresponds to the separable matrix $M_s = 2I$:

$$(\downarrow M_s) y(n_1, n_2) = y(Mn) = y(2n_1, 2n_2). \quad (9.63)$$

We are keeping one sample out of four. The four filters in four channels give critical sampling. The polyphase matrix will be 4×4 (generally $M \times M$).

Compare with the nonseparable quincunx filter bank. With $M_q = 2$ filters, the quincunx rule keeps the samples for which $n_1 + n_2$ is even:

$$(\downarrow M_q) y(n_1, n_2) = y(M_q n) = y(n_2 + n_1, n_2 - n_1). \quad (9.64)$$

Figure 9.17 shows the lattice of integers and the quincunx sublattice of samples:

Notice how $M_s = 4$ separable sublattices cover the whole lattice. Similarly $M_q = 2$ quincunx sublattices (staggered grids) cover all mesh points. The separable lattices are strongly oriented in the horizontal and vertical directions! The quincunx lattice has an extra symmetry at 45° and -45° . It is closer to isotropic, meaning independent of direction. A diagonal edge in an



Figure 9.17: The sampling lattices for M_s (separable) and M_q (quincunx).

image will be captured far better by the quincunx. But those filters are harder to design, if we want PR and especially if we want orthogonality.

The reader will see how other sampling matrices M give the lattice points Mn . Downsampling keeps all values $y(Mn)$ on this lattice. Then the upsampling step ($\uparrow M$) assigns a zero when (n_1, n_2) is not in the lattice. As always, $(\uparrow M)$ is the transpose of $(\downarrow M)$. If we apply both, we get the identity operator on the lattice and otherwise zero:

$$(\uparrow M)(\downarrow M)y(n_1, n_2) = \begin{cases} y(n_1, n_2) & \text{on the lattice} \\ 0 & \text{for other } (n_1, n_2). \end{cases} \quad (9.65)$$

A synthesis filter in each channel completes the filter bank: all normal.

Example 1 (Separable Haar): The Haar filter will be typical of separable filters. The four filters have coefficients $\pm\frac{1}{4}$ and the low-low response is $H_0(\omega_1, \omega_2) = \frac{1}{4}(1 + e^{-i\omega_1})(1 + e^{-i\omega_2})$. We “block” the input four samples at a time, with $x(0, 0), x(0, 1), x(1, 0), x(1, 1)$ in the zeroth block. Or in practice, the signal goes through ordinary Haar in the x direction and then in the y direction. The two-dimensional Haar bank is a block transform with 4×4 blocks:

$$\text{Haar block} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

This is just the two-dimensional DFT matrix. It is also the polyphase matrix. One constant term only, because Haar has no overlap. It is an orthogonal filter bank!

The Haar block shows a tensor product $H_{2 \times 2} \otimes H_{2 \times 2}$ of one-dimensional DFT's. The matrix $H_{2 \times 2} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ is multiplied by each of its four entries to give the four 2×2 subblocks. (A tensor product has order $N_1 N_2$ when the matrices have orders N_1 and N_2 . The matrix of order N_1 appears N_2 times on each block row of the tensor product, multiplied by entries from the matrix of order N_2 .) For separable filters, the 4×4 polyphase matrix is always the tensor product $H_p(z_1) \otimes H_p(z_2)$ of 2×2 one-dimensional polyphase matrices.

Every separable filter bank inherits the properties of its one-dimensional factors. The bank is orthogonal with accuracy (p, p) if the factors are orthogonal with accuracy p . The filters cannot also be linear phase (except for Haar). It is possible [KarVet] to achieve linear phase and orthogonality with nonseparable filters.

Example 2 (Nonseparable): Quincunx sampling has one alias term because $M - 1 = 1$. When $(\downarrow M_q)$ is followed by $(\uparrow M_q)$ as in (9.65), that alias is $X(-z_1, -z_2)$. The quincunx modulation

matrix is 2×2 :

$$\mathbf{H}_m(z_1, z_2) = \begin{bmatrix} H_0(z_1, z_2) & H_0(-z_1, -z_2) \\ H_1(z_1, z_2) & H_1(-z_1, -z_2) \end{bmatrix}.$$

Orthogonality requires that $\mathbf{H}_m(z_1, z_2) \mathbf{H}_m^T(z_1^{-1}, z_2^{-1}) = 2I$. The $(1, 1)$ entry is

$$H_0(z_1, z_2)H_0(z_1^{-1}, z_2^{-1}) + H_0(-z_1, -z_2)H_0(-z_1^{-1}, -z_2^{-1}) \equiv 2. \quad (9.66)$$

The product filter $P(z_1, z_2)$ must be halfband! The highpass H_1 comes from H_0 by an *alternating flip* in z_1 and z_2 :

$$H_1(z_1, z_2) = \text{odd 2D delay of } H_0(-z_1^{-1}, -z_2^{-1}). \quad (9.67)$$

All looks familiar. But in two dimensions there is one enormous difference. *We cannot factor most product filters.* Even if the symmetric polynomial $P(\omega_1, \omega_2)$ is nonnegative, it may be impossible to express it as the square $|H(\omega_1, \omega_2)|^2$ of a polynomial. The idea of computing zeros of $P(z)$ and separating them into two factors is strictly one-dimensional. We must design the filters directly, and not by factorization.

A cascade structure [VK, p. 182] can give orthogonality or linear phase:

$$\mathbf{H}_p(z_1, z_2) = \mathbf{R}_{2j} \begin{bmatrix} 1 & \\ & z_2^{-1} \end{bmatrix} \mathbf{R}_{2j-1} \begin{bmatrix} 1 & \\ & z_1^{-1} \end{bmatrix} \cdots \mathbf{R}_0.$$

The filter bank is orthogonal if the matrices \mathbf{R} are orthogonal. The shortest lowpass filter \mathbf{h} has parameters a, b, c in its eight nonzero coefficients:

$$\mathbf{h} = \begin{bmatrix} -b & -ab & & \\ -c & -ac & -a & 1 \\ abc & -abc & & \end{bmatrix}. \quad (9.68)$$

With constraints on a, b, c this is analogous to the Daubechies D_4 filter.

Example 3 (McClellan transformation): A convenient way to design linear phase 2D filter banks is to begin with a symmetric centered 1D filter:

$$H(\omega) = \sum_{-L}^L \mathbf{h}(n) \cos n\omega = \sum_{-L}^L \mathbf{h}(n) T_n(\cos \omega).$$

The Tchebycheff polynomial T_n produces $\cos n\omega$ from powers of $\cos \omega$. For example $\cos 2\omega = 2\cos^2 \omega - 1$, so that $T_2(x) = 2x^2 - 1$. The McClellan transformation replaces $\cos \omega$ by a symmetric 2D filter response $F(\omega_1, \omega_2)$. Then $H(\omega_1, \omega_2) = \sum \mathbf{h}(n) T_n(F(\omega_1, \omega_2))$ is still symmetric. In the quincunx case we choose $F = \frac{1}{2}(\cos \omega_1 + \cos \omega_2)$. More general polynomials than T_n have been used effectively in [TayKing].

By designing the 1D coefficients in $H(\omega)$ we get a good linear phase 2D filter $H(\omega_1, \omega_2)$. When the former gives perfect reconstruction with FIR inverse, so does the latter. (The 2D determinant is a monomial when the 1D determinant is. We cannot have orthogonality too.) The accuracy is also preserved, because a factor $(1 + \cos \omega)^p$ transforms to $(1 + \frac{1}{2}(\cos \omega_1 + \cos \omega_2))^p$. There are p zeros at the alias frequency (π, π) in the (ω_1, ω_2) plane.

Note that a separable filter with response $H(\omega_1)H(\omega_2)$ has p zeros at all three of the alias frequencies $(0, \pi)$ and $(\pi, 0)$ and (π, π) . The zeros are needed as in 1D for stability of the iterated lowpass filter and convergence to the scaling function.

Dilation Equation and Wavelets

The lowpass filter has coefficients $h_0(k_1, k_2)$. When we iterate with rescaling, the cascade algorithm hopes to converge to the scaling function. The limiting equation when $\phi^{(i+1)}$ and $\phi^{(i)}$ approach $\phi(t)$ is the dilation equation:

$$\phi(t_1, t_2) = M \sum h_0(n_1, n_2) \phi(Mt - n). \quad (9.69)$$

Here $t = (t_1, t_2)$ and $n = (n_1, n_2)$ are column vectors. The matrix M has determinant M . When we change variables from $Mt - n$ to τ , this determinant preserves the double integral:

$$M \iint \phi(Mt - n) dt_1 dt_2 = \iint \phi(\tau) d\tau_1 d\tau_2. \quad (9.70)$$

The lowpass normalization is still $\sum \sum h_0(n_1, n_2) = 1$. This filter leads to the scaling function. The other $M - 1$ filters lead to $M - 1$ wavelets by the usual wavelet equation:

$$w_k(t) = M \sum \sum h_k(n_1, n_2) \phi(Mt - n), \quad 1 \leq k < M. \quad (9.71)$$

Orthogonality will mean that the translates $\phi(t - n)$ at scale zero are an orthonormal basis for the space V_0 . The wavelet translates $w_1(t - n), \dots, w_{M-1}(t - n)$ are an orthonormal basis for W_0 . The wavelet translates and dilates $M^{j/2} w_k(M^j t - n)$ are an orthonormal basis for the whole finite energy space $L^2(\mathbf{R}^2)$.

Notice the dilation matrix M ! We nearly wrote 2^j instead of M^j . This would be correct only for the matrix $M = 2I$ that gives separable filters. The iteration then gives a separable scaling function and three separable wavelets:

$$\begin{array}{ll} \text{low-low: } \phi(t) = \phi(t_1)\phi(t_2) & \text{high-low: } w_2(t) = w(t_1)\phi(t_2) \\ \text{low-high: } w_1(t) = \phi(t_1)w(t_2) & \text{high-high: } w_3(t) = w(t_1)w(t_2) \end{array} \quad (9.72)$$

It is pleasant to verify (Problem 3) that separable filters give these separable solutions to the dilation equation and wavelet equation. With accuracy p in the one-dimensional filter and $1, t, \dots, t^{p-1}$ in its scaling space V_0 , the separable 2D filter (product filter) will inherit this accuracy. All p^2 of the polynomials $(t_1)^r (t_2)^s$ will be in V_0 , for $r < p$ and $s < p$. In the special case of splines, $\phi(t)$ is a product $\phi(t_1)\phi(t_2)$ of one-dimensional B-splines. The coefficients $h(n_1, n_2)$ are products of binomial coefficients. These are just the coefficients in $H(z_1, z_2) = (1+z_1^{-1})^p (1+z_2^{-1})^p$.

Note that a quincunx filter does not need this factor. It is $[1 + \frac{1}{2}(z_1^{-1} + z_2^{-1})]^p$ that gives accuracy p for quincunx. We would construct $\phi(t_1, t_2)$ by the cascade algorithm. Then $(t_1)^r (t_2)^s$ is locally in V_0 if $r + s < p$. For $p = 2$ we only need the linear polynomials $1, t_1, t_2$ and not the extra $t_1 t_2$ that comes with separable filters.

Remark We believe that *box splines* could lead to multidimensional algorithms. They are generated by filters for which $H(z_1, z_2)$ has simple factors. The factorization of H is a fundamental 2D difficulty, and box splines are direct constructions with known factors. The future will determine whether these (or other) multidimensional filters are successful.

Bounded Domains

For a bounded interval in one dimension, Section 8.5 proposed several constructions of boundary functions. Local support and the multiresolution property $V_j \subset V_{j+1}$ and even the polynomial accuracy p were preserved. The same properties are desirable in two dimensions but not so easy to achieve.

We note one immediate difficulty. The boundary scaling functions $\phi_b(t)$ were differences between monomials t^k and combinations $\sum y_k \phi(t - k)$ of interior functions. The combination reproduces t^k exactly except near the end of the interval. In one dimension, $\phi_b(t)$ will have local support. But in two dimensions the support will be a *thin ring* along the boundary — not local! A more careful construction is needed.

The new paper [CoDaDe] gives a simple local construction of boundary functions for a square domain, starting from separable wavelets. On a general domain their method is less simple. It seems better than earlier constructions, and this problem must be faced in solving partial differential equations by a wavelet method (Section 11.6).

Problem Set 9.5

1. Show that the quincunx upsampling $u(n) = x(M_q^{-1}n)$ yields

$$U(\omega_1, \omega_2) = \frac{1}{2} [X(\omega_1, \omega_2) + X(\omega_1 + \pi, \omega_2 + \pi)].$$

Express this also in the (z_1, z_2) domain. Which inputs give $u = 0$?

2. For ($\downarrow M_s$) the first row of the modulation matrix contains $H_0(z_1, z_2)$, $H_0(-z_1, z_2)$, $H_0(z_1, -z_2)$, $H_0(-z_1, -z_2)$. Using products $H_0(z_1)H_0(z_2)$ of one-dimensional filters, write out the 4×4 modulation matrix $H_m(z_1, z_2)$. How is it related to the 2×2 matrices $H_m(z_1)$ and $H_m(z_2)$?
3. With the separable filters in equation (9.72), show that the separable scaling function $\phi(t_1)\phi(t_2)$ solves the 2D dilation equation with $M = 2I$.
4. Find $M = \det M$ and draw the lattices of vectors Mn for

$$M_{\text{ul}} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \text{ and } M = \begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix} \text{ and } M_{\text{hex}} = \begin{bmatrix} 1 & 2 \\ -1 & 2 \end{bmatrix}$$

5. For M_s and M_q draw the *Voronoi cell* of points whose closest lattice point is the origin. What is the area of the Voronoi cell?
6. If the lowpass filter $h_0(n_1, n_2)$ satisfies the quincunx orthogonality condition (9.66), show that $H_1(z_1, z_2) = -z_1^{-1}H_0(-z_1^{-1}, -z_2^{-1})$ gives an orthogonal highpass filter. The modulation matrix should have $H_m H_m^T = 2I$.
7. A 2D analogue of the hat function $\phi(t)$ is the bilinear tent $\phi(x)\phi(y)$. Find its Fourier transform as a function of ω_x and ω_y .
8. Another 2D analogue of the hat function is the Courant finite element $C(x, y)$. It is linear in each triangle obtained from grid lines $x = k_1, y = k_2, y - x = k_3$. Draw the six triangles around the origin and show that the function that is zero outside, with $C(0, 0) = 1$, has transform
- $$\widehat{C}(\omega_1, \omega_2) = \left(\frac{\sin(\omega_1/2)}{\omega_1/2} \right) \left(\frac{\sin(\omega_2/2)}{\omega_2/2} \right) \left(\frac{\sin(\omega_1 + \omega_2)/2}{(\omega_1 + \omega_2)/2} \right).$$
9. Take the Fourier transform of the dilation equation (9.69). By recursion find the infinite product formula (notice the transpose) $\widehat{\phi}(\omega) = \prod H((M^{-j})^T \omega)$.
10. What is the z-transform of $(\uparrow M)(\downarrow M)y$?

Chapter 10

Design Methods

10.1 Distortions in Image Compression

One of the most successful applications of the wavelet transform is transform-based image compression (also called *image coding*). The coder in Figure 10.1 operates by transforming the data to remove redundancy, then quantizing the transform coefficients (a lossy step), and finally entropy coding the quantized output. Because of superior energy compaction and correspondence with the human visual system, wavelet compression has produced good results. Since the wavelet basis functions have short support for high frequencies and long support for low frequencies, large smooth areas of an image may be represented with very few bits. High frequency detail is added where it is needed.

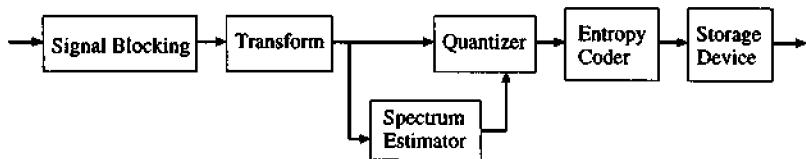


Figure 10.1: Transform-based image coder.

Figure 10.2 shows the original image and its decomposition (three levels) using the two-channel biorthogonal filter bank with (9, 7) coefficients. The upper left block is a smoothed approximation of the original image; it comes from three iterations of the lowpass filter with downsampling. The other subbands contain details at various scales. Since the filters have four vanishing moments, the signal energy in those subbands is small. Consequently, they have to be amplified in order to see the details.

The ten subimages are processed by a spectrum estimator for bit allocation. The statistical properties of the subimages guide this quantization step. The lowpass subband (upper left) is allocated the most bits. One strategy is based on the rate distortion curve. Given B bits, how does one allocate b_k bits per pixel to the k th subimage such that the reconstructed image has smallest distortion? Important distortion measures are mean-square-error (MSE), peak signal-to-noise (PSNR), and maximum error.

At high and medium bit rate (low and medium compression ratio), there is a strong correlation between the image quality and these objective measures. A good compression algorithm

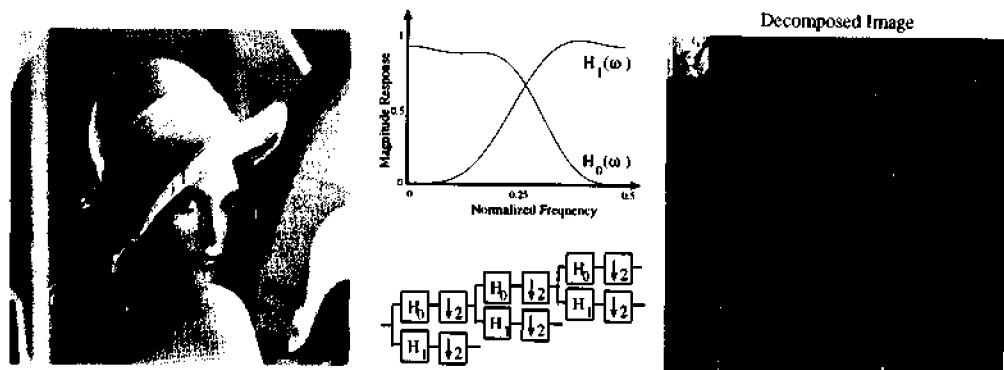


Figure 10.2: Original image "Lenna" and its multiresolution decomposition. The filter bank used is the (9, 7)-tap linear-phase filter bank.

would reconstruct the image with low MSE, high PSNR, and low Max error. At low bit rate (high compression), the basis function characteristics have profound effects on the quality of the reconstructed image. Typical distortions are ringing, blocking, and blurring artifacts.

Figure 10.3 shows the reconstructions for compression ratios of 32:1 (0.25 bits per pixel) and 64:1 (0.125 bpp) and 100:1 (0.08 bpp). The objective measures for 32:1 compression are MSE = 37.32, PSNR = 32.41 dB, Max error = 62.66. For 64:1 compression the error is larger: MSE = 61.36, PSNR = 30.25 dB, Max error = 68.67; and for 100:1 compression MSE = 104.3, PSNR = 27.95 dB, Max error = 121.3.

For medium bit rate (0.25 bpp), the objective measures are good indicators of the subjective quality of the image. At low bit rate, *this is not always the case!* There are many artifacts in the reconstructed image at 0.08 bpp. The choice of filters is important, as well as bit allocation.

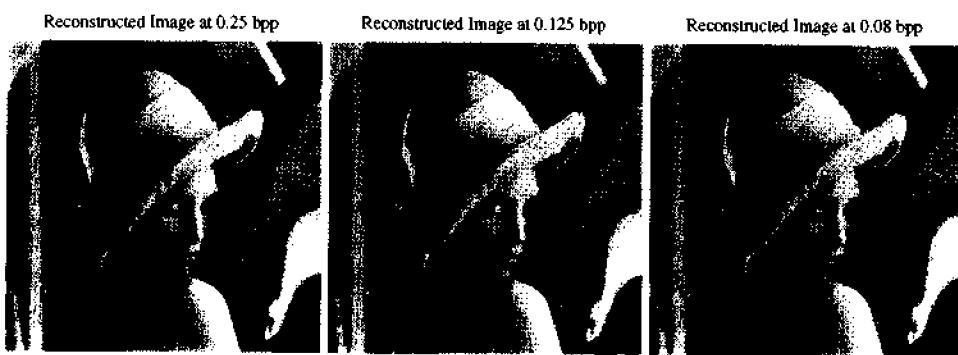
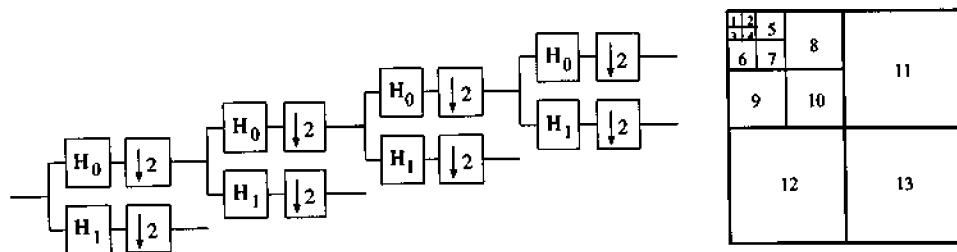


Figure 10.3: Reconstructed images for 32:1, 64:1, and 100:1 compression with (9, 7) filters.

With four levels and 13 subbands, there are several artifacts in the reconstructed images: blurring, border distortions, blocking, checkerboarding, and ringing. Each artifact will be elaborated below. They are the results of the filter choices, the bit allocation algorithms, and the convolutions.



Blurring Artifacts

Blurring occurs when the algorithm does not assign enough bits to the higher subbands 5 to 13. In the extreme case, we allocate all bits to subbands 1 to 4 and none to subbands 5 to 13. The compression ratio is 48 to 1 (0.167 bpp). Figure 10.4 shows that the reconstructed image is essentially an interpolated version of subband 1. It suffers from blurring.

The blurring will decrease if one allocates more bits to the higher subbands. Figure 10.4(b) shows the reconstructed image when the bit allocations are

9.29 5.63 6.52 4.96 3.38 4.20 2.81 0.97 1.73 0.23 0.00 0.00 0.00

The resulting compression is also 48 to 1. This image is much sharper.



Figure 10.4: Blurring in image compression: Too few bits for the detailed subbands.

Border Distortions

The image and the filters have finite length. One has to be careful in computing their convolution. Border distortion is the result of choosing the wrong extension. Chapter 8 discusses the filtering operations for finite-length signals by zero-padding of the input signals, circular convolution,

and symmetric extension. In image compression at 32 to 1 (0.25 bpp), with the (9, 7) symmetric filters, we can compare the effects of these three extensions.

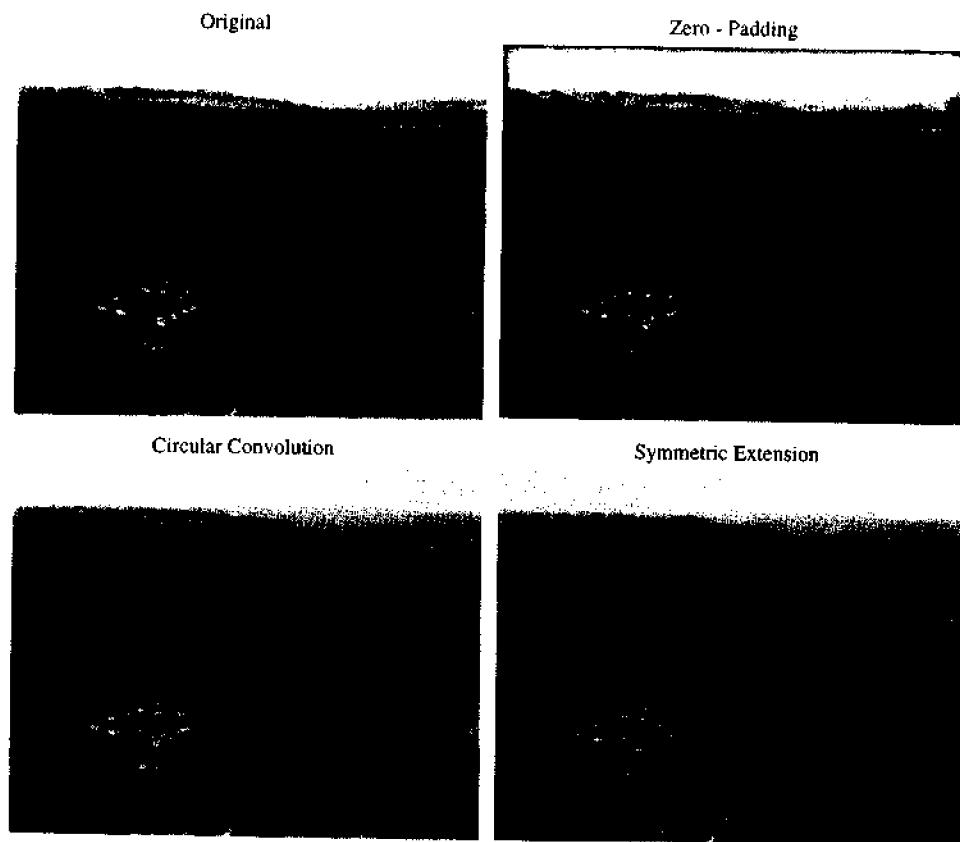


Figure 10.5	Zero-Padding	Circular Convolution	Symmetric Extension
MSE	278.40	96.10	88.67
PSNR (dB)	23.68	28.30	28.75
Max Error	168.40	54.43	55.26

The zero-padding method assumes that images are zero outside their borders. Therefore they are discontinuous signals. The pixel intensity at the reconstructed border is very low (dark). The large error of zero-padding occurs at the boundary.

Circular convolution assumes that the image is periodic. This assumption is not valid for our original image. If the subband images are not quantized, the reconstructed image has no distortion since the filter bank is a PR system. But quantization yields errors at the border when using circular convolution. The periodic assumption results in a discontinuous intensity level at the border.

Symmetric extension extends the images in a continuous way at the border. Although the objective performance of circular convolution is close, the subjective (perceptual) quality from

circular convolution is not as good as from symmetric extension.

Blocking Artifacts

One major disadvantage of the DCT in image coding is that it cannot be used for high compression because of its blocking artifacts. The basis functions typically have short length $M = 8$ for the DCT block transform with no overlap. Blocking will show up at the output. One way to improve is to use longer basis functions, which overlap neighboring blocks. Figure 10.5 also shows the reconstructed image using GenLOT (length 48). Most of the blocking artifacts are now unnoticeable.

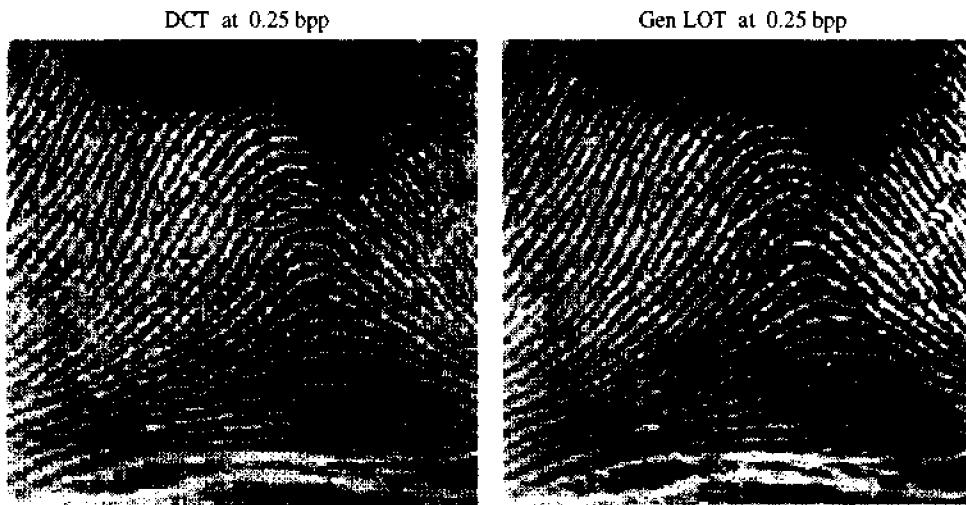


Figure 10.5: Blocking artifacts in image compression using DCT and lapped transforms.

High Frequency Distortion: Ringing

The reconstructed image is a linear combination of the scaling function and wavelets of the *synthesis bank*. Since symmetric filters are not orthogonal, the analysis and synthesis functions are not the same. The scaling function of the synthesis filter should be smooth and the wavelet should be short. The error is larger at higher subbands, assigning fewer bits to the wavelet coefficients. When the wavelet has length N , the error at the pixel (K, L) affects its neighbors $(K \pm k, L \pm \ell)$, where $k, \ell \leq N/2$.

Figure 10.6 shows the original image and reconstructed image at 0.5 bpp. The lengths of the synthesis filters are 11 and 17. Even at medium bit rate, the ringing effect is quite severe. This is especially true at strong edges.

Nonsmooth Functions: Checkerboard Artifacts

In all these compression examples, the filters are linear-phase factors of a maximally-flat half-band filter. All scaling functions and wavelets in the synthesis bank are smooth, which is essential for image reconstruction. When the functions are not smooth, what is the effect on reconstruction?

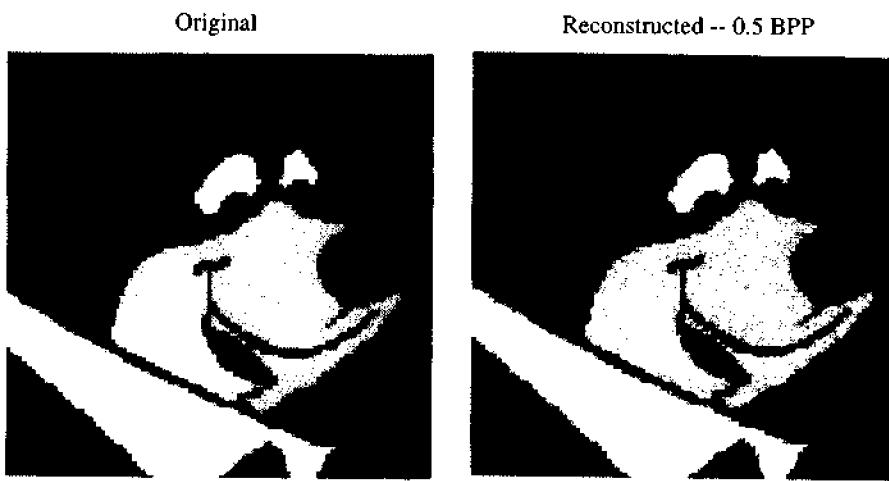


Figure 10.6: Ringing artifacts are clearly visible at edges.

The lowpass responses $H_0(z)$ and $F_0(z)$ in Figure 10.7 are designed such that the stopband error is equiripple (a desirable property in conventional digital signal processing). The scaling functions are not smooth. The reconstructed image suffers from the the **checkerboard effect**. This can be explained by observing that the scaling function from $F_0(z)$ has a large and narrow peak. The magnitudes near this narrow peak are about 10% below it. This explains the bright horizontal and vertical lines in the reconstructed image. The stronger intensity produces the checkerboard effect. This artifact can be attributed to the “roughness” of $\phi(t)$.

- | | |
|---|---|
| Lowpass synthesis:
Highpass synthesis: | <i>Long and smooth to avoid blocking and checkerboarding</i>
<i>Short to avoid ringing</i> |
|---|---|

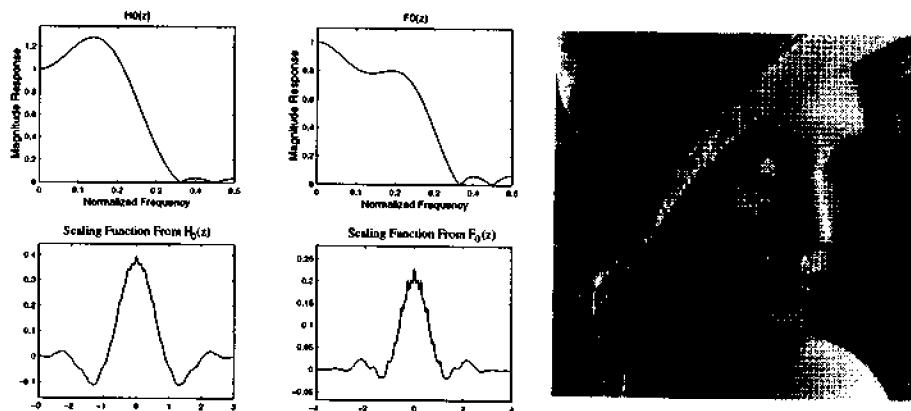


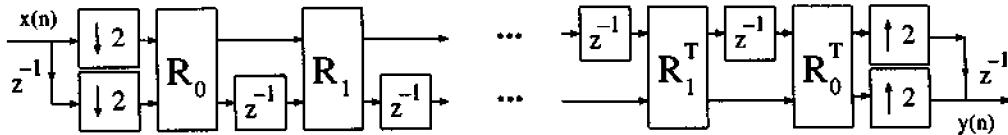
Figure 10.7: These lowpass equiripple filters are linear-phase factors of a halfband equiripple filter. The lengths are 9/7 and the rate is 0.25 bpp. The nonsmooth scaling functions create a checkerboard.

10.2 Design Methods—General Perspective

In all of our examples, one important factor for image quality is the choice of coefficients in the filter banks. For an objective function of coding gain or degree of smoothness or stopband attenuation, *one optimizes the filter coefficients*. This section will present several methods to design and optimize filter banks and wavelets. These methods are based on spectral factorizations, lattice structures and time-domain formulations. We will present several common choices for the objective function Φ , and its parameters, and the constraints to impose:

Design Problem: Minimize Φ (parameters) subject to constraints. (10.1)

The design parameters could be the lattice coefficients or the filter coefficients. Recall how a lattice structure can impose the PR property and also linear phase, pairwise-mirror-image or cosine modulation. The lattice structure of a two-channel paraunitary filter bank includes R_k = rotation by θ_k . There is a one-to-one correspondence between lattice coefficients $\theta_0, \theta_1, \dots, \theta_N$ and filter coefficients. The advantage of the lattice coefficients is that the quantized filters are still PR. The rotations are still orthogonal when the angles are rounded off.



Objective Function Φ

Useful objective functions are the *stopband attenuation*, *coding gain*, and *degree of smoothness*. Stopband attenuation is very important in audio compression, whereas coding gain and smoothness are more important in image compression.

Stopband Attenuation: Φ measures the passband and stopband errors of $H(z)$:

$$\Phi = \begin{cases} \int_{\Omega} W(e^{j\omega}) |H_d(e^{j\omega}) - H(e^{j\omega})|^2 d\omega, & L_2 \text{ norm (energy)} \\ \text{Max } [W(e^{j\omega}) |H_d(e^{j\omega}) - H(e^{j\omega})|], & L_{\infty} \text{ norm (maximum)}. \end{cases} \quad (10.2)$$

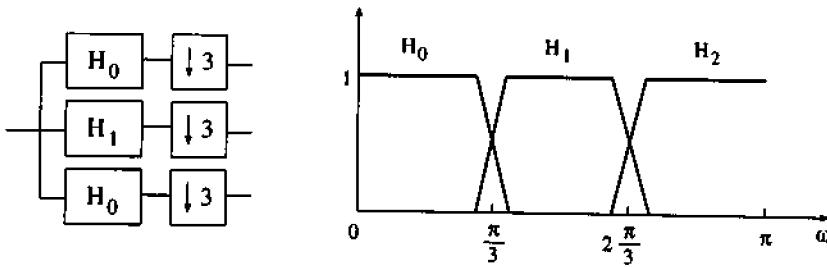
The frequency region Ω consists of the passband $0 \leq \omega \leq \omega_p$ and stopband $\omega_s \leq \omega \leq \pi$. $H_d(e^{j\omega})$ is the desired response of the lowpass filter. $W(e^{j\omega})$ is a positive weighting function. Examples of typical $H_d(e^{j\omega})$ and $W(e^{j\omega})$ are:

$$H_d(e^{j\omega}) = \begin{cases} e^{-jL\omega}, & 0 \leq \omega \leq \omega_p \\ 0, & \omega_s \leq \omega \leq \pi \end{cases} \quad W(e^{j\omega}) = \begin{cases} a, & 0 \leq \omega \leq \omega_p \\ b, & \omega_s \leq \omega \leq \pi. \end{cases} \quad (10.3)$$

If one increases the ratio b/a , the stopband error δ_s is much smaller than the passband error δ_p . In a filter bank with M channels, Φ has a term from each filter $H_k(e^{j\omega})$. The bandpass filters have two stopbands enclosing one passband.

The analysis filters in a paraunitary bank satisfy $\sum |H_k(e^{j\omega})|^2 = 1$. Consequently, it is sufficient to consider only the stopbands. With three filters, the figure shows $\frac{\pi}{3} \leq \omega \leq \frac{2\pi}{3}$ as a

stopband of H_0 and H_2 and as the passband of H_1 . In its passband, $H_1(e^{j\omega})$ must approximate unity. The passband error $\delta_{p1} = 1 - \sqrt{1 - \delta_{s0}^2 - \delta_{s2}^2}$ is very small, if both stopband errors δ_{s0} and δ_{s2} are small. Filter banks with high stopband attenuation are essential in audio compression and signal detection.



Example 10.1. Consider the design of a linear-phase filter bank with (9, 7) taps based on maximum error in the stopband. Figure 10.8(a) shows the equiripple stopband responses of $H_0(z)$ and $F_0(z)$. They are the factors of an equiripple halfband filter of length 15. There is no zero at $\omega = \pi$ and the scaling function and wavelets are *not smooth*. This pair of filters was used earlier to demonstrate checkerboard artifacts in image compression.

Figure 10.8(b) shows the frequency responses of a 9/7 maximally-flat halfband filter. *The stopband attenuation of both filters is bad.* However, each filter has four zeros at $\omega = \pi$. The scaling functions and wavelets are very smooth. They are the filters used in the FBI fingerprint image compression standard.

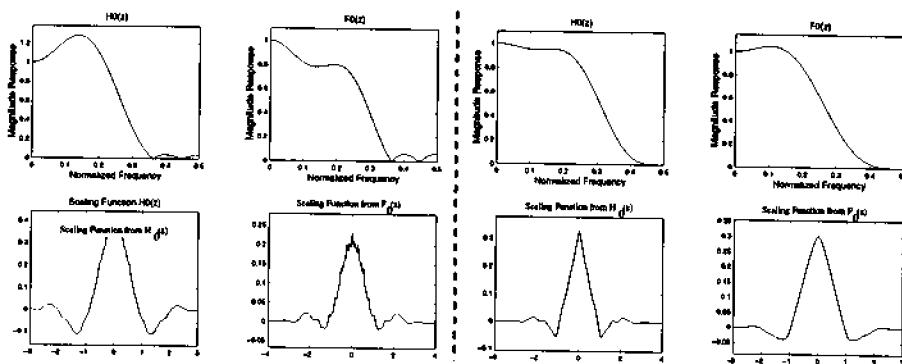


Figure 10.8: Example of equiripple design (left) and maximally-flat design (right).

Example 10.2. Figure 10.9 shows the filter responses of a cosine-modulated filter bank (8 channels, filter lengths 128). The objective function is $\int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega$, the stopband energy of the prototype filter.

Coding Gain: A uniform quantizer has an equivalent noise model. The input signal $x(n)$ is quantized to b bits to obtain $v(n)$. For large b , one can assume that the quantization error $q(n) = v(n) - x(n)$ is a random variable, uniformly distributed in $-2^b \leq q \leq 2^b$. The error variance of $q(n)$ is $\frac{1}{3}2^{-2b}$.

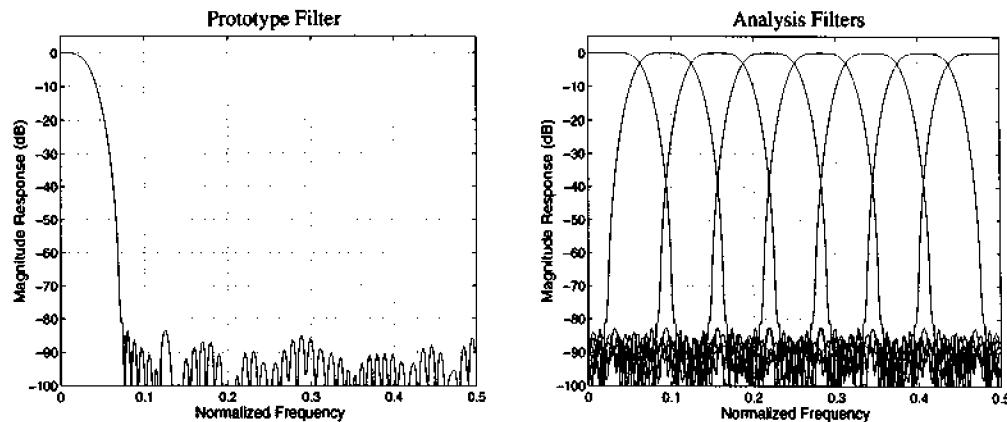
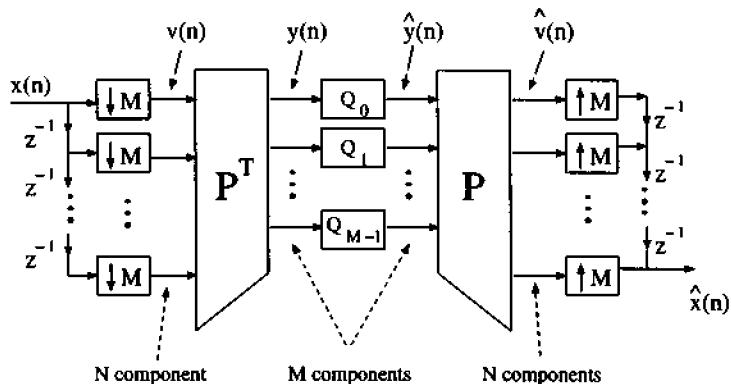


Figure 10.9: A cosine-modulated filter bank with stopband attenuation as objective Φ .

Instead of quantizing the input directly, one can transform it into another domain. The input signal is blocked into length N and transformed to length M by P^T . The subband signal $y_k(n)$ is then quantized to produce $\hat{y}_k(n)$. The transform should be designed so that the energy of x is concentrated in the first few coefficients of y_k . The number of bits in the quantizers Q_k should be proportional to the energy of $y_k(n)$.



The coding gain measures the energy compaction from the transform. An orthogonal transform with high coding gain yields reconstructed images with high quality at low bit rate. Chapter 11 will discuss coding gain in more detail. For biorthogonal systems, [Katto-Yasuda] derived a unified formula when the input signal $x(n)$ is modeled as a Markov process. Its inter-sample correlation factor is ρ . The coding gain is G when the sampling ratios are α_k :

$$G(\rho) = \prod_{k=0}^{M-1} (A_k B_k)^{-\alpha_k} \quad \text{where} \quad \begin{cases} A_k &= \sum_n \sum_m h_k(n) h_k(m) \rho^{|m-n|} \\ B_k &= \sum_n (f_k(n))^2 \end{cases}$$

For example, a 3-level wavelet transform has four analysis filters:

$$H_0(z) H_0(z^2) H_0(z^4) \quad H_0(z) H_0(z^2) H_1(z^4) \quad H_0(z) H_1(z^2) \quad H_1(z). \quad (10.4)$$

The coefficients in these combinations are used in the formula for coding gain. Assuming

$\rho = 0.95$, one can compute G and improve the performance. The sampling ratios are 1/8, 1/8, 1/4 and 1/2.

Example 10.3. Figure 10.10 shows the frequency responses of an 8-channel GenLOT with length 48, for maximum stopband attenuation and also for maximum coding gain. Note that the responses are different. These GenLOTs are compared at 50 : 1 compression. The objective performances with high stopband attenuation are MSE = 135, PSNR = 26.84 dB and Maximum error = 96. The corresponding numbers for the GenLOT with high coding gain are 130, 26.99 dB and 84. Even though these are comparable, maximizing the attenuation produced blocking artifact. This is not due to the block length 48, but it is due to the DC leakage. The bandpass and highpass filter responses are not zero at $\omega = 0$, so the DC term has leaked out of the lowpass.

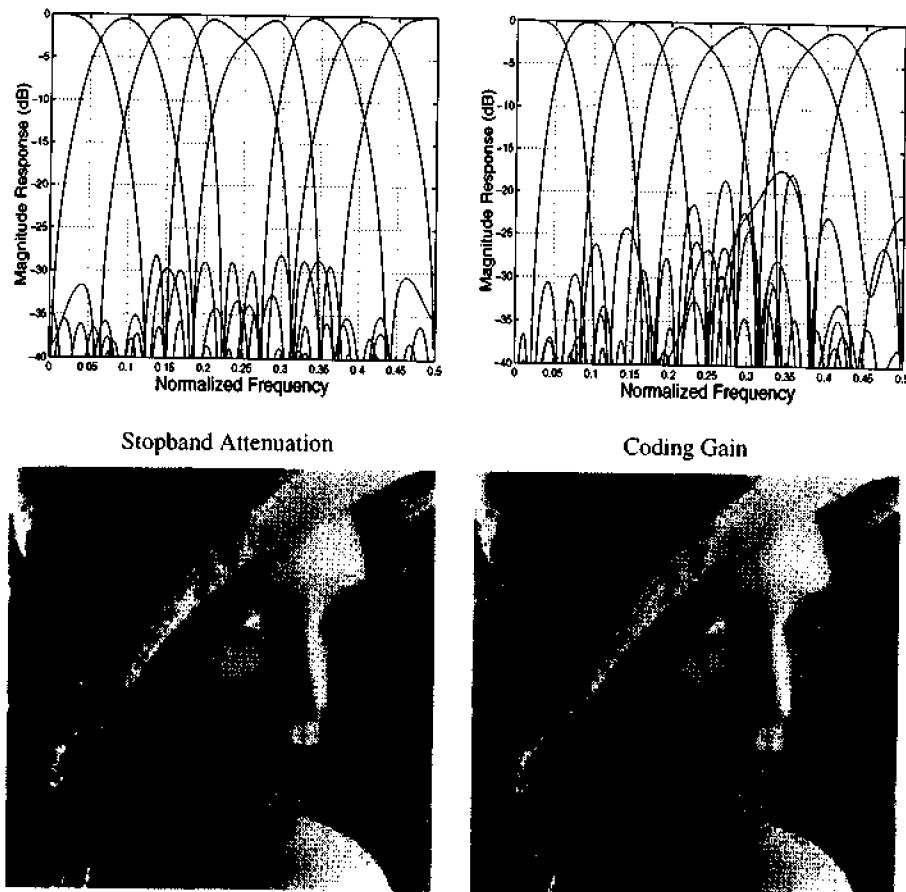


Figure 10.10: GenLOT designs optimized for stopband attenuation (left) and coding gain (right).

Constraints

PR constraints: The equivalent conditions for M -channel perfect reconstruction are

$$\begin{cases} \sum_n h_k(n) f_\ell(2m - n) = \delta(m)\delta(k - \ell) \\ F_p(z) H_p(z) = z^{-n_0} I. \end{cases} \quad (10.5)$$

How does one design $H_k(z)$ and $F_k(z)$ that satisfy (10.5)? One way is to design $h_k(n)$ first and then solve for $f_k(n)$. For arbitrary analysis filters, synthesis filters might not exist. The time domain approach in [Nayebi1] presents an iterative method to solve for $h_k(n)$ and $f_k(n)$.

For FIR filters, (10.5) is true only if the determinant of $H_p(z)$ is a delay. The special case of paraunitary $H_p(z)$ can be parametrized by a set of rotation angles or Householder matrices. This is the lattice structure. The PR constraints are *automatically satisfied*.

For two channels, a simple PR condition exists for FIR systems: $H_0(z)F_0(z)$ is a halfband filter. This is the result when aliasing is cancelled. Such simplicity does not exist for systems with more channels.

Smoothness constraints: From the image compression examples, it is clear that the synthesis functions should be smooth. This relates directly to the number p of zeros at $\omega = \pi$ of $F_0(z)$. The smoothness property is also known as the DC-leakage property in traditional image processing.

Example 10.4. The lowpass Daubechies orthogonal filter $H_0(z)$ is the minimum-phase spectral factor of a maximally-flat halfband filter $P(z)$. Since $P(z)$ has $2p$ zeros at $\omega = \pi$, $H_0(z)$ and $F_0(z)$ have p zeros. Note that the functions in Figure 10.11 of D_{24} are considerably smoother than D_4 , since D_{24} has 12 zeros at $\omega = \pi$.

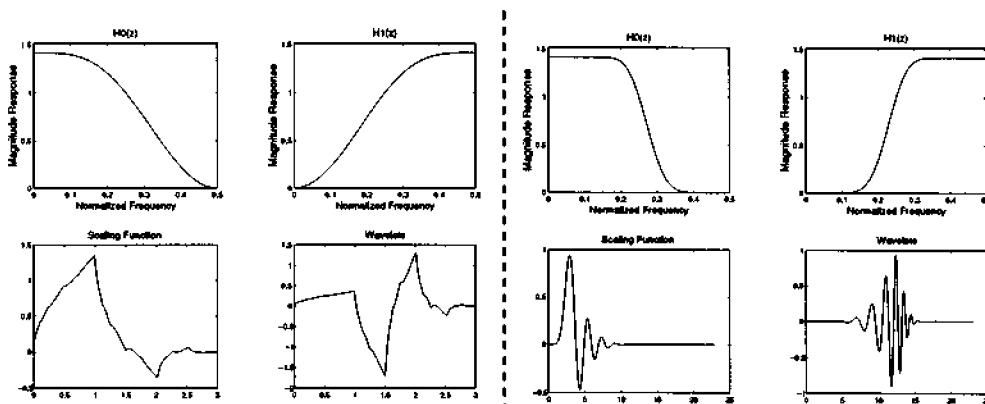


Figure 10.11: Magnitude response, scaling function and Daubechies wavelets D_4 and D_{24} .

10.3 Design of Perfect Reconstruction Filter Banks

The three main design methods extend in different ways to M channels:

$\left\{ \begin{array}{l} \text{Spectral factorization} \\ \text{Lattice structure method} \\ \text{Time-domain method} \end{array} \right.$	factor an M -th band filter minimize Φ (lattice coefficients) minimize Φ (filter coefficients) subject to PR.
--	---

The first approach uses spectral factorization. We find roots or factors $F_0(z)H_0(z)$ of a linear-phase polynomial $P(z)$. For applications that require large stopband attenuation, $P(z)$ has many

zeros on or near the unit circle. Its factorization becomes difficult. A special class that is very useful in function approximation is the halfband maxflat Daubechies filter (Section 5.5). One can distribute its zeros in several ways to $H_0(z)$ and $F_0(z)$ to obtain:

$$\left\{ \begin{array}{ll} \text{Paraunitary filter banks} & \text{--- Orthogonal Daubechies wavelets} \\ \text{Type A linear phase filter banks} & \text{--- Symmetric even-length wavelets} \\ \text{Type B linear phase filter banks} & \text{--- Symmetric odd-length wavelets.} \end{array} \right. \quad (10.6)$$

These wavelets have excellent approximation from the zeros at $\omega = \pi$.

The second approach is based on the lattice structure implementation of the filter bank. The unknown parameters are the lattice coefficients k_i . Since all PR filter banks are characterized by an appropriate set of k_i , the design problem reduces to minimizing the objective function $\Phi(k_i)$. Then the corresponding filters can be computed. The minimization is a challenging problem because of the *nonlinear relation* between the lattice and filter coefficients.

The time-domain approach expresses the PR conditions directly on the filter coefficients. Nonlinear quadratic constrained optimization yields good filters. One optimization method formulates the PR conditions in terms of matrices and the filter coefficients [Nayebi1]. The algorithm iteratively solves (10.5). An alternative is to formulate the objective function and constraints in quadratic forms:

$$\text{Minimize } \mathbf{h}^T \mathbf{P} \mathbf{h} \text{ subject to } \mathbf{h}^T \mathbf{Q}_k \mathbf{h} = c_k. \quad (10.7)$$

We may use Lagrange multipliers (explicitly or implicitly). The advantage of this constrained quadratic approach is that the derivatives and the Hessian of both the objective function and the PR conditions can be computed exactly.

Spectral Factorization Method

Given a polynomial of the form $P(z) = F_0(z)H_0(z)$, the objective here is to find the two unknown polynomials $F_0(z)$ and $H_0(z)$. We assume that all polynomials have real coefficients. Note that $F_0(z) = z^{-N}H_0(z^{-1})$ in a paraunitary filter bank, and the problem becomes a spectral factorization. For a low-degree polynomial $P(z)$, one can find its zeros and distribute them to $F_0(z)$ and $H_0(z)$. Since all polynomials are real-valued, the zeros have to be in complex-conjugate pairs.

For example, consider the maximally-flat halfband filter $P(z)$ of length 15. Its zeros are shown in Figure 10.12 with 8 zeros at $\omega = \pi$. Given the 14 zeros of $P(z)$, there are many ways of distributing its zeros to $F_0(z)$ and $H_0(z)$. One can assign half the zeros to $H_0(z)$ and half to $F_0(z)$ in such a way that $F_0(z) = z^{-7}H_0(z^{-1})$. This means that z_0 is a zero of $H_0(z)$ when z_0^{-1} is a zero of $F_0(z)$. The factorization assigns half of the zeros at π to $H_0(z)$, and the remaining half to $F_0(z)$. For minimum phase, $H_0(z)$ has all other zeros inside the unit circle, whereas $F_0(z)$ has all the zeros outside the unit circle. Both filters have length 8. This factorization yields a paraunitary filter bank and the Daubechies D_8 orthogonal wavelets.

Part b shows an alternate zero distribution. The zeros are chosen such that the resulting filters have odd length and linear phase. Note that $H_0(z)$ and $F_0(z)$ have the same number of zeros at $\omega = \pi$. This (9, 7) pair is used in the FBI fingerprint image compression standard.

Part c shows another zero distribution that yields linear phase with even length. Note that the zeros at π for $H_0(z)$ and $F_0(z)$ are not the same. The scaling function and wavelet from the

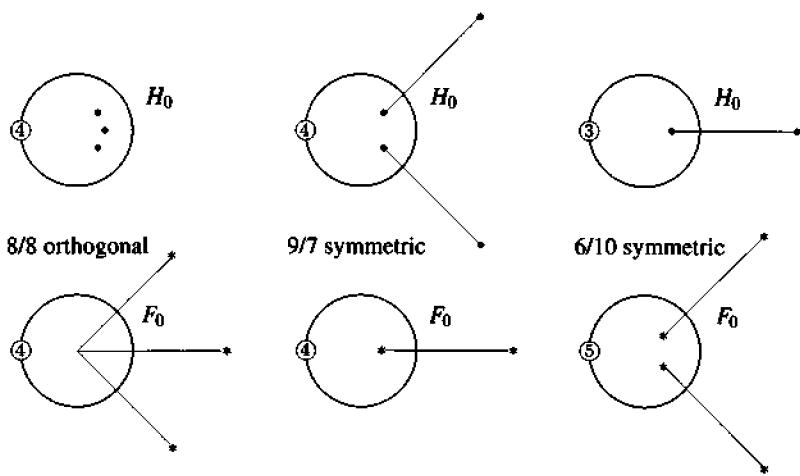
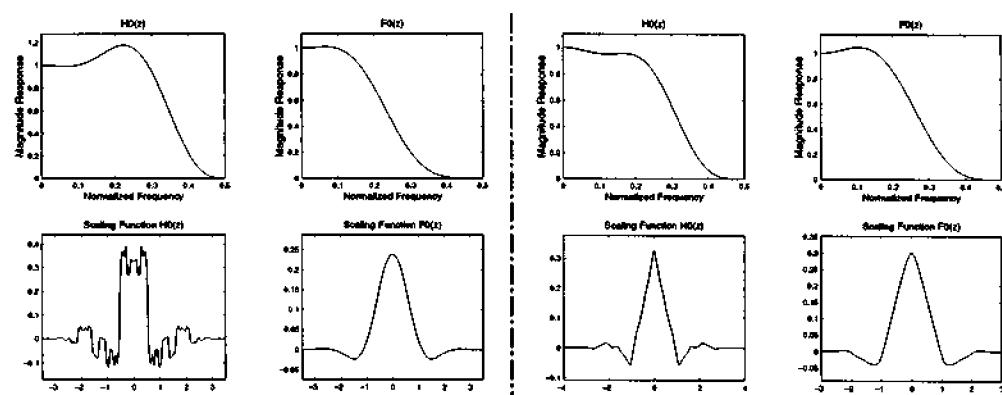


Figure 10.12: The zero distributions of $F_0(z)$ and $H_0(z)$. (a) Paraunitary filter bank (b) Linear phase filter bank (Type B, odd length) and (c) Linear phase filter bank (Type A, even length).

synthesis bank is smoother (it has more zeros at π). This is the (6, 10)-tap filter pair. Its performance is very good in image compression. The three factorizations demonstrate the essential ideas in designing two-channel paraunitary filter banks (orthogonal wavelets) and two-channel linear phase filter banks (symmetric wavelets).



Spectral factorization applies to the design of two-channel filter banks and Near-Perfect-Reconstruction cosine-modulated filter banks. In the two-channel filter bank case, $F_0(z)$ and $H_0(z)$ are the factors of $P(z)$. In the NPR cosine-modulated case, the prototype filter $H(z)$ is a spectral factor of a $2M$ -th band filter. For an arbitrary filter bank, such simple PR/NPR conditions do not exist. There is no optimization process involved in spectral factorization. In general, an optimal filter $P(z)$ does not guarantee optimal filters $H_0(z)$ and $F_0(z)$.

For orthogonal filters, the cepstral algorithm of [Mian-Nainer] is widely used. It computes the minimum phase spectral factor without finding the zeros. Section 5.4 describes this (additive) splitting of the logarithm of $P(e^{j\omega})$.

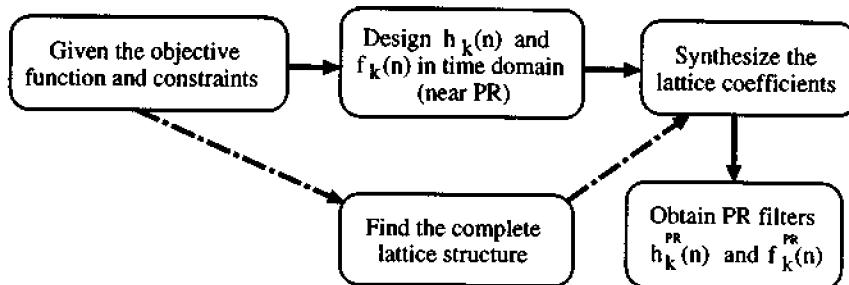
Lattice Structure

Most PR filter banks can be implemented based on lattice structure. For any set of lattice coefficients, the paraunitary, linear-phase, and cosine-modulation properties are structurally imposed. This means that those properties (and also PR) are enforced independent of the lattice coefficient values. The objective function could take many different forms depending on the applications. Examples are stopband attenuation, coding gain, and interference energy. The lattice method is used for two-channel paraunitary filter bank [VaidHoang], the two-channel biorthogonal linear-phase filter bank [V], the M-channel linear-phase paraunitary filter bank [Soman], and the paraunitary cosine-modulated filter bank [Koil2].

Another role for lattices begins in the time domain. Optimization yields a *Near Perfect Reconstruction (NPR)* solution:

$$\sum_n h_k(n) f_\ell(2m - n) < \epsilon < 10^{-10} \text{ for } k \neq \ell, m \neq 0. \quad (10.8)$$

From these NPR filters, one can find the lattice coefficients. Those give PR filters $h_k^{PR}(n)$ and $f_k^{PR}(n)$. If the reconstruction error ϵ is small, the NPR filters and PR filters are close and $\Phi^{PR} \approx \Phi$. If ϵ is not small, the filters are not close and Φ could change drastically.



There are two key points to ensure success:

- Existence of *complete* and *minimal lattice structure*. The *completeness* is crucial here since the lattice structure must cover all filters with the given properties. *Minimal* is desirable to ensure efficient implementation.
- The reconstruction error ϵ must be sufficiently small to guarantee that $\Phi^{PR} \approx \Phi$. A typical objective function is stopband attenuation.

Example 10.5. A two-channel linear phase system of length 32 is shown in Figure 10.13(a) (solid lines). These filters are designed with $\epsilon = 10^{-4}$. They are used in the above lattice synthesis procedure to obtain a PR filter bank (broken lines). The attenuation has changed from -42 dB to -25 dB . Apparently, 10^{-4} is not small enough. Figure 10.13(b) shows the frequency response of the NPR solution with $\epsilon = 10^{-14}$ (solid line) and the PR solution (broken line). The NPR filters are very close to PR. Practically, there are no changes in the frequency responses.

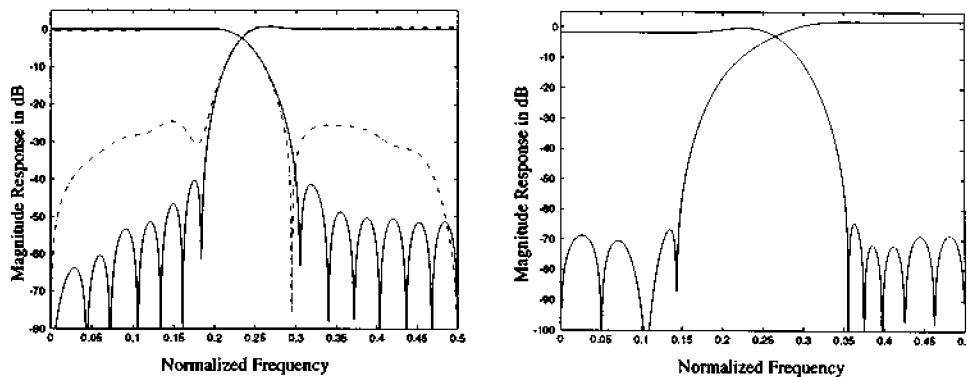


Figure 10.13: NPR and PR filters obtained by lattice synthesis for $\epsilon = 10^{-4}$ (left) and $\epsilon = 10^{-14}$ (right).

What went wrong in the lattice synthesis that changes the filter response drastically when ϵ is not small? Note the subtle difference between PR and NPR lattice structures in Figure 10.14. The delays are no longer z^{-2} , but z^{-1} . There are additional blocks $\hat{\Gamma}_k = \begin{bmatrix} 1 & \hat{\gamma}_k \\ \hat{\gamma}_k & 1 \end{bmatrix}$ between Γ_k . A large ϵ will yield nonzero $\hat{\gamma}_k$, and setting them to zero will change the filters dramatically. Since $\hat{\gamma}_k$ are very small if $\epsilon < 10^{-14}$, setting $\hat{\gamma}_k = 0$ does not change the frequency responses or the objective function.

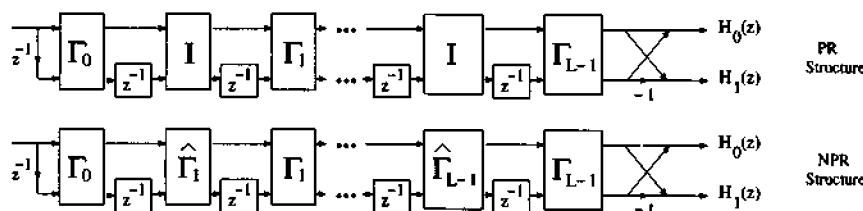


Figure 10.14: Lattice structures for PR and NPR Type-A linear phase filter banks.

Time-Domain Methods

Block matrix approach: This method is suggested by [Nayebi1] to solve the PR equations when the filter length is $N = LM$:

$$\underbrace{\begin{bmatrix} P_0^T & 0 & \cdots & 0 \\ P_1^T & P_0^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ P_{L-1}^T & P_{L-2}^T & \cdots & P_0^T \\ 0 & P_{L-1}^T & \cdots & P_1^T \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P_{L-1}^T \end{bmatrix}}_A \underbrace{\begin{bmatrix} Q_0^T \\ Q_1^T \\ \vdots \\ Q_{L-1}^T \\ Q \end{bmatrix}}_B = \underbrace{\begin{bmatrix} 0 \\ \vdots \\ 0 \\ J \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_C \quad (10.9)$$

J is the exchange matrix (reverse identity). Its location determines the delay of the system. The entries of the filter matrices are $P_{k\ell} = h_k(\ell)$ and $Q_{k\ell} = f_k(\ell)$. The $M \times M$ submatrices P_j and Q_j are defined by:

$$P = [P_0 \ P_1 \ \cdots \ P_{L-1}] \quad \text{and} \quad Q = [Q_0 \ Q_1 \ \cdots \ Q_{L-1}]. \quad (10.10)$$

Chapter 9 used this formulation for LOT ($N = 2M$ and $L = 2$), where (10.10) reduces to $P_1^T P_1 + P_0^T P_0 = I$ and $P_1^T P_0 = 0$.

In summary, the time domain PR constraints $AQ = B$ are general. Given the analysis filters $h_k(n)$ (and thus A), this is not always solvable. One can find “optimal” synthesis filters $Q = (A^T A)^{-1} A^T B$ by minimizing $\|A Q - B\|$. Then fix those filters $f_k(n)$ and reverse the equation to find optimal $h_k(n)$. This is the iterative block matrix approach.

Quadratic Constrained Least Squares method (QCLS): For an M -channel paraunitary filter bank, the PR condition requires no distortion ($k = 0$) and no aliasing ($k > 0$):

$$T_k(z) = \frac{1}{M} \sum_{\ell=0}^{M-1} z^{-N} H_\ell(z^{-1}) H_\ell(z W^k) = \delta(k) z^{-\ell}.$$

Let h be a column vector consisting of all the filter coefficients $h_k(n)$:

$$h = (h_0(0) \ \cdots \ h_0(N) \ \cdots \ h_{M-1}(0) \ \cdots \ h_{M-1}(N))^T.$$

The PR conditions can be written in the form $h^T Q_k h = 0$ and $h^T S_k h = 1$. Furthermore, the objective function for stopband attenuation can be formulated as a quadratic $\Phi = h^T Ph$, where P is symmetric and positive definite. The optimized filter is precisely h_{opt} such that

$$h_{\text{opt}} \text{ minimizes } h^T Ph \text{ subject to } \begin{cases} h^T Q_k h = 0, \\ h^T S_k h = 1. \end{cases} \quad (10.11)$$

Since Q_k is normally not positive definite, the solution can be difficult. However, there are effective optimization procedures that linearize the quadratic constraints [Schittkowski]. These procedures will yield an approximate solution (the constraints are not satisfied exactly). The errors are very small and can be ignored in most practical cases. The QCLS method is used to design the two-channel PR linear-phase filter bank [QCLS], the NPR Pseudo-QMF bank [Nguyen1], the PR cosine-modulated filter bank [QCLS], and the M -channel paraunitary linear-phase filter bank [Nguyen2].

10.4 Design of Two-Channel Filter Banks

This book has emphasized the construction of $H_0(z)$ and $F_0(z)$ as factors of a halfband filter $P_0(z)$. This method is conceptually simple, and good in practice. The other two approaches, based on lattice structure and quadratically constrained optimization, look awkward by comparison.

But those methods have their own advantages. The lattice structure is efficient to implement, once it is found. The optimization can yield “best” filters rather than “good” filters. This brief section will record the details for the two cases of greatest significance: *orthogonal filter banks* and *linear phase filter banks*.

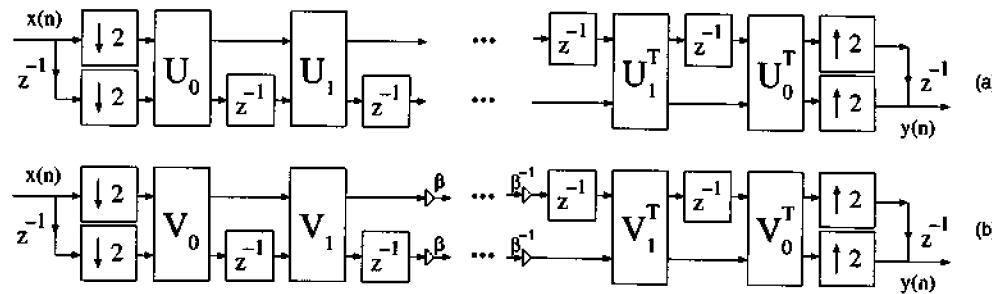


Figure 10.15: Lattice structures for the two-channel paraunitary filter bank. The first structure uses rotation angles as parameters and the second structure uses lattice coefficients.

Lattice Structure Method

Figure 10.15 shows two lattice structures for a paraunitary filter bank and orthogonal wavelets. The objective is to find the right set of angles θ_k in U_k (or lattice coefficients α_k in V_k) such that an objective function is minimized. The objective function considered by [VaidHo] is the stopband error (L_2 norm). There are several tables compiled for orthogonal filters with various frequency specifications.

Figure 10.16 shows the two lattice structures for linear phase filter banks. Recall that Type A yields filters with even length (one is symmetric and the other is antisymmetric). Type B yields symmetric filters with odd length. In Type A, the parameters are the lattice coefficients in Γ_k . [NgVa1] presents design examples for both Type A and B systems, designed using the nonlinear optimization where the parameters are the lattice coefficients.

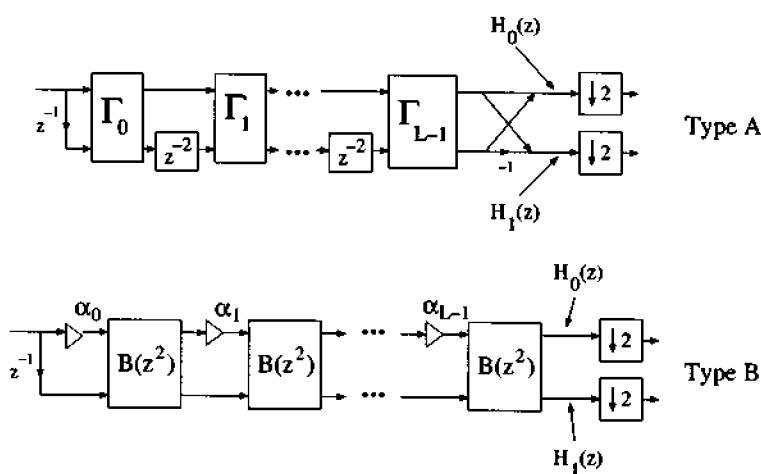


Figure 10.16: Lattice structures for symmetric filter banks, even and odd lengths.

Time-Domain Method—QCLS Formulation of Orthogonality

The orthogonality condition is that $P(z) = z^{-N} H_0(z^{-1}) H_0(z)$ is a halfband filter. We will show how to formulate this in quadratic form $\mathbf{h}^T \mathbf{Q}_k \mathbf{h} = c_k$ using

$$\mathbf{h} = [h_0(0) \ h_0(1) \ \cdots \ h_0(N)]^T \text{ and } \mathbf{e}(z) = [1 \ z^{-1} \ \cdots \ z^{-N}]^T \text{ and } H_0(z) = \mathbf{e}^T(z) \mathbf{h}.$$

Then $z^{-N} H_0(z^{-1}) = \mathbf{h}^T \mathbf{J} \mathbf{e}(z)$ where \mathbf{J} is the reverse identity matrix (antidiagonal). Therefore $P(z) = \mathbf{h}^T \mathbf{J} \mathbf{e}(z) \mathbf{e}^T(z) \mathbf{h}$. Let

$$\mathbf{e}(z) \mathbf{e}^T(z) = \sum z^{-k} \mathbf{D}_k \quad \text{where} \quad [\mathbf{D}_k]_{i,j} = \begin{cases} 1; & i+j=k \\ 0; & \text{otherwise.} \end{cases} \quad (10.12)$$

Then $P(z)$ can be rewritten as

$$P(z) = \sum z^{-k} \mathbf{h}^T \mathbf{J} \mathbf{D}_k \mathbf{h} = \sum p(k) z^{-k}. \quad (10.13)$$

In other words, the k th coefficient of $P(z)$ is $p(k) = \mathbf{h}^T \mathbf{J} \mathbf{D}_k \mathbf{h}$. The halfband condition on $P(z)$ gives the constraints in the design problem:

$$\min_{\mathbf{h}} \mathbf{h}^T \mathbf{P} \mathbf{h} \quad \text{subject to} \quad \mathbf{h}^T \mathbf{J} \mathbf{D}_{N-1-2n} \mathbf{h} = \delta(n).$$

Here $\mathbf{h}^T \mathbf{P} \mathbf{h}$ denotes the stopband energy. See [Nguyen3] for more examples.

Lagrange Multiplier Method

With the choice of $F_0(z) = H_1(-z)$ and $F_1(z) = -H_0(z)$ to cancel aliasing, the filters $H_0(z)$ and $H_1(z)$ must satisfy the PR condition $H_0(z)H_1(-z) - H_1(z)H_0(-z) = z^{-\ell}$. This is equivalent to $\mathbf{C} \mathbf{h}_1 = \mathbf{m}$, where \mathbf{C} is a matrix whose elements come from $\mathbf{h}_0(n)$. The highpass filter is \mathbf{h}_1 and the k -th element of \mathbf{m} is $\delta(k-l)$. Given $\mathbf{h}_0(n)$, the minimization problem becomes:

$$\text{Minimize } \Phi \text{ subject to } \mathbf{C} \mathbf{h}_1 = \mathbf{m}. \quad (10.14)$$

The minimization constructs a Lagrangian function

$$L(\mathbf{h}_1, \lambda) = \Phi - \lambda^T (\mathbf{C} \mathbf{h}_1 - \mathbf{m}). \quad (10.15)$$

The optimality conditions $\partial L / \partial \mathbf{h}_1 = 0$ and $\partial L / \partial \lambda = 0$ are

$$\frac{\partial \Phi}{\partial \mathbf{h}_1} - \mathbf{C}^T \lambda = \mathbf{0} \quad \text{and} \quad \mathbf{C} \mathbf{h}_1 - \mathbf{m} = \mathbf{0}. \quad (10.16)$$

In some cases Φ can be expressed explicitly in terms of \mathbf{h}_1 . When stopband error is the objective function, Φ has the form $\frac{1}{2} \mathbf{h}_1^T \mathbf{S}_1 \mathbf{h}_1 + \mathbf{S}_2^T \mathbf{h}_1 + s_3$. The optimality conditions become

$$\begin{bmatrix} -\mathbf{S}_1 & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{S}_2 \\ \mathbf{m} \end{bmatrix}. \quad (10.17)$$

QCLS formulation for Type A: The filter length is $2m$ and the PR condition is

$$T(z) = \sum_{n=0}^{4m-2} t(n)z^{-n} = -H_0(z)H_1(-z) + H_0(-z)H_1(z) = z^{-(2m-1)}. \quad (10.18)$$

Substituting $(-z)$ for z in (10.18), one obtains $t(n) = 0$ for even n . Consequently we have m conditions on the odd-numbered coefficients:

$$t(2k+1) = \begin{cases} 0, & 0 \leq k \leq m-2 \\ 1, & k = m-1. \end{cases} \quad (10.19)$$

Let $\mathbf{h} = (h_0(0) \ \dots \ h_0(m-1) \ \ h_1(0) \ \dots \ h_1(m-1))^T$. Our objective is to express the m conditions in (10.19) in terms of \mathbf{h} . The polynomials $H_0(z)$, $H_1(z)$, $H_0(-z)$ and $H_1(-z)$ can be written using $\mathbf{e}(z) = (1 \ z^{-1} \ \dots \ z^{-(m-1)})^T$:

$$\begin{cases} H_0(z) = \mathbf{h}^T \begin{pmatrix} \mathbf{e}(z) + z^{-m} \mathbf{J} \mathbf{e}(z) \\ \mathbf{0} \end{pmatrix}, & H_0(-z) = \mathbf{h}^T \begin{pmatrix} \mathbf{U} \mathbf{e}(z) + (-1)^{m/2} z^{-m} \mathbf{J} \mathbf{U} \mathbf{e}(z) \\ \mathbf{0} \end{pmatrix} \\ H_1(z) = \mathbf{h}^T \begin{pmatrix} \mathbf{0} \\ \mathbf{e}(z) - z^{-m} \mathbf{J} \mathbf{e}(z) \end{pmatrix}, & H_1(-z) = \mathbf{h}^T \begin{pmatrix} \mathbf{0} \\ \mathbf{U} \mathbf{e}(z) - (-1)^{m/2} z^{-m} \mathbf{J} \mathbf{U} \mathbf{e}(z) \end{pmatrix} \end{cases}$$

Here \mathbf{J} is the exchange matrix and \mathbf{U} is diagonal with $U_{kk} = (-1)^k$. Substituting into (10.18), $T(z)$ is simplified to

$$T(z) = \mathbf{h}^T \begin{pmatrix} \mathbf{0} & \Gamma(z) \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{h} \quad \text{where } \mathbf{E}(z) = \mathbf{e}(z)\mathbf{e}^T(z) \quad \text{and} \quad (10.20)$$

$$\begin{aligned} \Gamma(z) = & [\mathbf{U}\mathbf{E}(z) - \mathbf{E}(z)\mathbf{U}] + z^{-m} \left[(-1)^{m/2} (\mathbf{J}\mathbf{U}\mathbf{E}(z) + \mathbf{E}(z)\mathbf{U}\mathbf{J}) \right. \\ & \left. - (\mathbf{U}\mathbf{E}(z)\mathbf{J} + \mathbf{J}\mathbf{E}(z)\mathbf{U}) \right] + z^{-2m} (-1)^{m/2} [\mathbf{J}\mathbf{E}(z)\mathbf{U}\mathbf{J} - \mathbf{J}\mathbf{U}\mathbf{E}(z)\mathbf{J}]. \end{aligned}$$

Substituting (10.12) for $\mathbf{E}(z)$ produces a polynomial $\sum_{k=0}^{4m-2} z^{-k} \mathbf{h}^T \begin{pmatrix} \mathbf{0} & \Gamma_k \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{h}$, where the Γ_k are constant matrices \mathbf{D}_k , \mathbf{J} and \mathbf{U} . Comparing term by term in (10.20), (10.19) becomes

$$\begin{cases} \mathbf{h}^T \mathbf{Q}_{2k+1} \mathbf{h} = 0; & 0 \leq k \leq m-2 \\ \mathbf{h}^T \mathbf{Q}_{2m-1} \mathbf{h} = 1; & \end{cases} \quad \text{where } \mathbf{Q}_n = \begin{pmatrix} \mathbf{0} & \Gamma_n \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \quad \text{and} \quad (10.21)$$

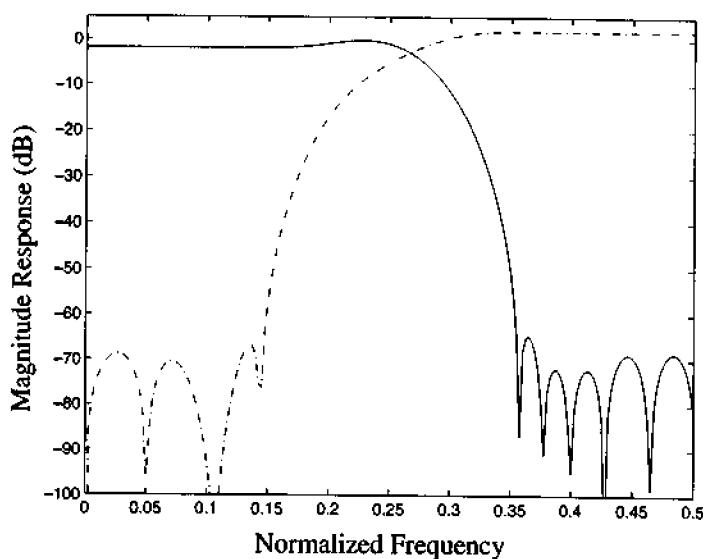
$$\Gamma_n = \begin{cases} \mathbf{U}\mathbf{D}_n - \mathbf{D}_n\mathbf{U}; & 0 \leq n \leq m-1 \\ \mathbf{U}\mathbf{D}_n - \mathbf{D}_n\mathbf{U} + (-1)^{m/2} (\mathbf{J}\mathbf{U}\mathbf{D}_{n-m} + \mathbf{D}_{n-m}\mathbf{U}\mathbf{J}) - (\mathbf{U}\mathbf{D}_{n-m}\mathbf{J} + \mathbf{J}\mathbf{D}_{n-m}\mathbf{U}); & m \leq n \leq 2m-2 \\ (-1)^{m/2} (\mathbf{J}\mathbf{U}\mathbf{D}_{m-1} + \mathbf{D}_{m-1}\mathbf{U}\mathbf{J}) - (\mathbf{U}\mathbf{D}_{m-1}\mathbf{J} + \mathbf{J}\mathbf{D}_{m-1}\mathbf{U}); & n = 2m-1. \end{cases}$$

In summary, the PR condition in (10.19) is rewritten as m quadratic constraints on \mathbf{h} .

Example 10.6. The QCLS formulation is used in the design of a Type A bank with filter lengths 30. The objective function Φ is

$$\int_0^{\omega_p} [|H_0(e^{j0}) - H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2] d\omega + \int_{\omega_s}^{\pi} [|H_0(e^{j\omega})|^2 + |H_1(e^{j\pi}) - H_1(e^{j\omega})|^2] d\omega$$

Note that both passband and stopband errors are included since this is not a power complementary system. This is an NPR filter bank with $\epsilon = 1.5 \times 10^{-15}$. The figure below shows the frequency responses of the filters.



We close by noting four other design methods for two channels:

- Biorthogonal filter banks with integer coefficients divided by 2^k [AH].
- LMS-based algorithm [Roy].
- Biorthogonal linear-phase filter bank [Phoong1, Kiya].
- Alias-free two-channel allpass-based IIR filter bank [V, Vaid1, Nguyen4, EkPr].

10.5 Design of Cosine-modulated Filter Banks

A cosine-modulated filter bank is very efficient. It can be implemented using *one prototype filter* $H(z)$ and a cosine-modulation block. The M filters $H_k(z)$ are cosine-modulated versions of $H(z)$. As a result, a 32-channel bank with length 512 has only 256 unknowns (assuming linear phase), instead of 16,384 unknowns in the general case. This helps the design algorithm tremendously in terms of convergence speed and results.

There are many design algorithms and they all use $H(z)$. What properties does $H(z)$ satisfy to obtain NPR solution or PR solution?

- **Nearly PR** If $H(z)$ is a spectral factor of a $2M$ -th band filter, amplitude and phase distortions are eliminated. Aliasing is suppressed by sufficient stopband attenuation.
- **Exactly PR** Let $H(z)$ be linear phase. The $2M$ polyphase components $G_k(z)$ of a paraunitary cosine-modulated filter bank have to satisfy

$$G_k(z^{-1}) G_k(z) + G_{M+k}(z^{-1}) G_{M+k}(z) = \frac{1}{2M}. \quad (10.22)$$

We begin with NPR methods that are based on spectral factorization. Lattice structure and time-domain approaches will also be discussed, for PR designs. We will see that NPR methods tend to give better stopband attenuation.

Traditional Pseudo-QMF Bank Design

Pseudo-QMF Banks are NPR and cosine-modulated. They were proposed for communication and audio processing applications. Aliasing of the adjacent bands is cancelled by appropriate phase terms in the modulation. The nonadjacent aliasing terms are minimized by designing a prototype with good stopband attenuation δ_s . The aliasing level is comparable to δ_s , and the reconstruction error consists of distortion (Φ_d) and aliasing (Φ_a):

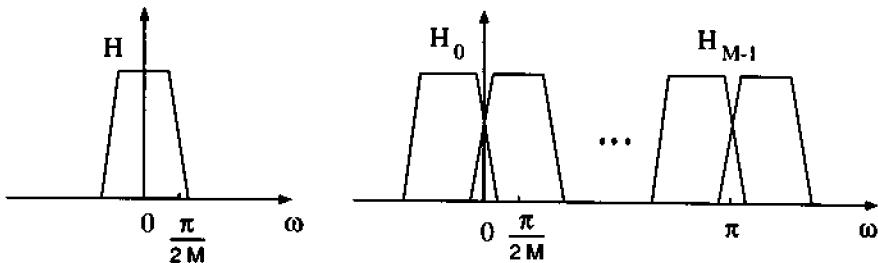
$$\Phi = \Phi_d + \Phi_a = (1 - \alpha) \int_0^{\pi} |T_0(e^{j\omega}) - e^{-j\ell\omega}|^2 d\omega + \alpha \int_{\omega_s}^{\pi} |H(e^{j\omega})|^2 d\omega.$$

The weighting factor has $0 \leq \alpha \leq 1$, and ℓ is the system delay. There is typically a tradeoff between distortion and aliasing. Distortion is eliminated if $H(z)$ is restricted to be a spectral factor of a $2M$ -th band filter $P(z)$. The objective function becomes simply Φ_a .

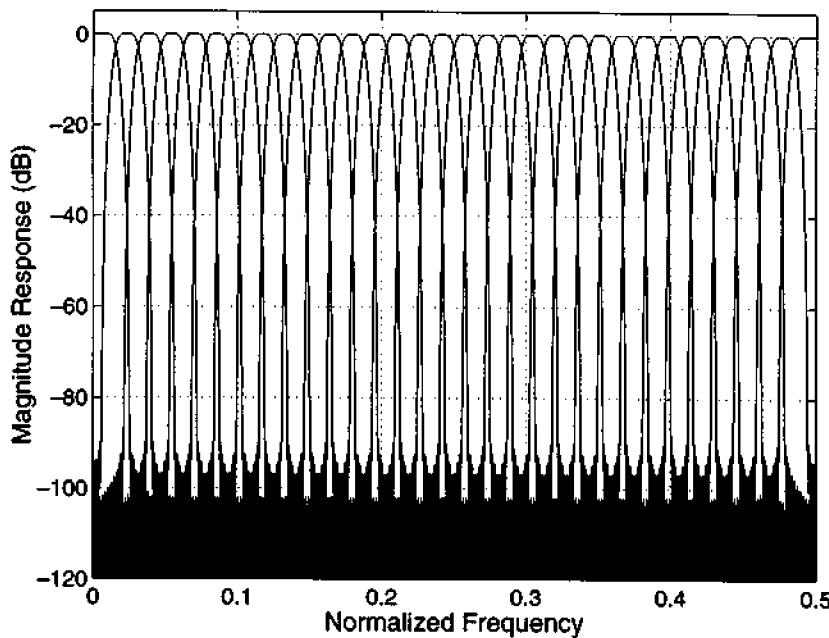
Spectral Factorization Evaluating $H(z)H(z^{-1})$ on the unit circle $z = e^{j\omega}$ yields a nonnegative function $P(e^{j\omega}) = |H(e^{j\omega})|^2$. How does one design $P(z)$ where $P(e^{j\omega}) \geq 0$? The design procedure is as follows:

- Design an equiripple $2M$ -th band filter with $\hat{P}(N) = \frac{1}{2M}$ and $\hat{P}(N - 2M\ell) = 0$ for $\ell \neq 0$. N is large since $\hat{P}(z)$ must also have good stopband attenuation δ_s .
- Form another $2M$ -th band filter $P(z)$ (nonnegative!) by $P(z) = \delta_s z^{-N} + \hat{P}(z)$. Factor $P(z)$.

The lowpass filter $H_0(z)$ and the highpass filter $H_{M-1}(z)$ are most affected by the choice of factor (minimum phase or near linear phase). Their passbands are combinations of two passbands and they have bumps or dips [Koil3] if the phase of $H(z)$ is not linear.



[Nguyen1] uses QCLS for linear-phase factorization. The resulting cosine-modulated filter bank has no distortion, small aliasing (comparable to stopband attenuation), and flat passband responses for H_0 and H_{M-1} . A 32-channel system of length 512 is shown, with distortion 5×10^{-11} and aliasing -83 dB.



QCLS in NPR Pseudo-QMF Design

The linear phase relation $H(z) = z^{-N}H(z^{-1})$ expresses the coefficients $p(n)$ as $\mathbf{h}^T \mathbf{D}(n) \mathbf{h}$. The matrix $\mathbf{D}(n)$ depends on the index n and the filter length [Nguyen1]. The design becomes a quadratic-constrained minimization:

Minimize the stopband error Φ_a subject to $\mathbf{p}(n) = \mathbf{h}^T \mathbf{D}(n) \mathbf{h} = 2M$ -th band filter.

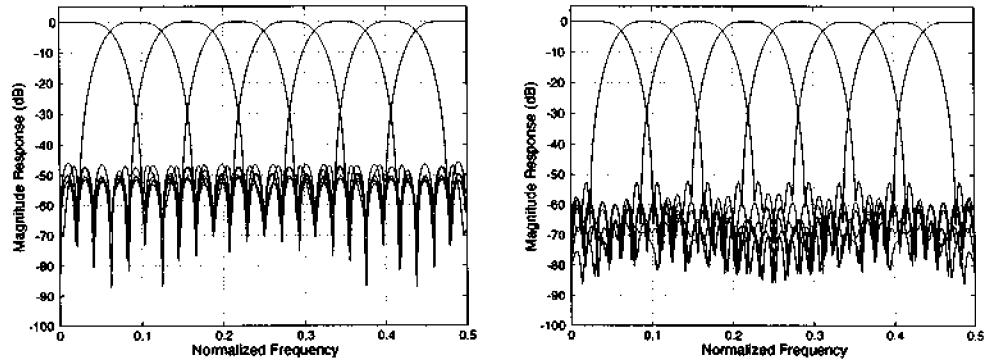
The quadratic forms have simple derivatives and Hessians, which makes the optimization search very fast. A high-order system design like the example above takes about 4 minutes on a Sun Workstation 10.

Lattice Structure Method

In a PR cosine-modulated filter bank of length $N = 2mM$, each polyphase component pair $(G_k(z), G_{M+k}(z))$ is paraunitary. The design procedure based on lattice structure is:

- Initialize the angles θ_k . Compute $H_k(z)$ and $\Phi = \int_{\omega_s}^{\pi} |H(e^{j\omega})|^2 d\omega$.
- Iterate on θ_k to minimize Φ .

This method works well for short filters. One needs a time domain approach to design long PR systems with high attenuation. We show the frequency responses of two eight-channel PR cosine-modulated filter banks with lengths 64 and 80. The stopband attenuation improves from ~ 46 dB to ~ 52 dB (no distortion or aliasing). The attenuation using the QCLS is ~ 58 dB, a significant improvement from ~ 52 dB.

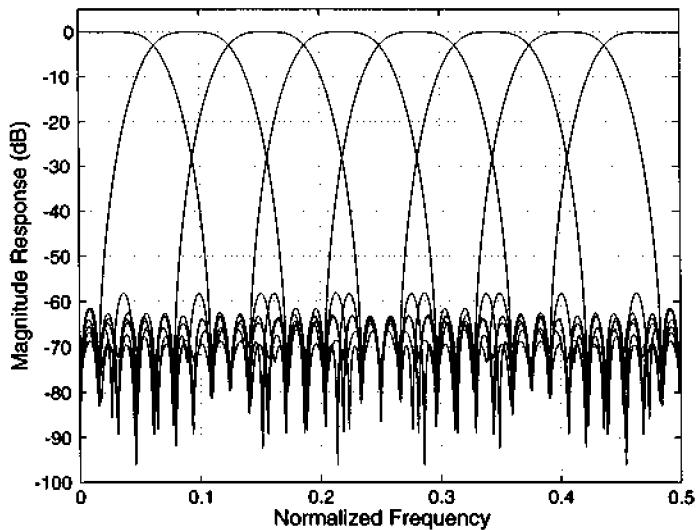


Time Domain Method in PR Design—QCLS Formulation

The objective is to design the linear phase prototype filter $H(z)$ such that its $2M$ polyphase components $G_k(z)$ satisfy the PR conditions (in pairs k and $M+k$). Let \mathbf{h} be a vector of length mM consisting of the first half of $\mathbf{h}(n)$. Let \mathbf{e} be a delay vector of length m with components z^{-k} . Then the polyphase components $G_k(z)$ can be written as $\mathbf{h}^T \mathbf{V}_k \mathbf{e}$, where $[\mathbf{V}_k]_{i,j} = 1$ only for $i = k + 2jM$. The PR condition (10.22) on the pair $G_k(z)$ and $G_{M+k}(z)$ becomes

$$\mathbf{h}^T [\mathbf{V}_k \mathbf{J} \mathbf{e}^T \mathbf{V}_k^T + \mathbf{V}_{M+k} \mathbf{J} \mathbf{e}^T \mathbf{V}_{M+k}^T] \mathbf{h} = \frac{1}{2M} z^{-(m-1)}. \quad (10.23)$$

Substituting $\mathbf{e} \mathbf{e}^T = \sum_{n=0}^{2m-2} z^{-n} \mathbf{D}_n$ where $[\mathbf{D}_n]_{i,j} = 1$ only for $i + j = n$, we obtain the



Example of an 8-channel cosine-modulated filter bank with length 80.

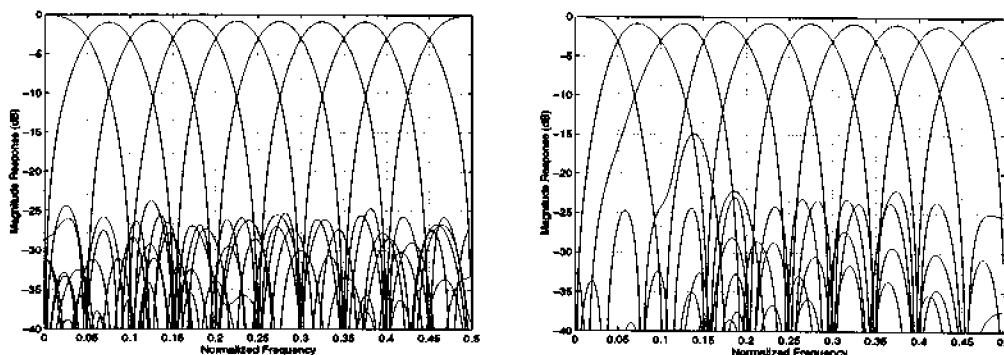
PR condition in terms of $\mathbf{h}(n)$:

$$\mathbf{h}^T [\mathbf{V}_k \mathbf{J} \mathbf{D}_n \mathbf{V}_k^T + \mathbf{V}_{M+k} \mathbf{J} \mathbf{D}_n \mathbf{V}_{M+k}^T] \mathbf{h} = \frac{1}{2M} \delta(n - m + 1). \quad (10.24)$$

The design problem has these *quadratic constraints* [QCLS]. The objective function Φ is also quadratic in \mathbf{h} . As an example, we design an 8-channel cosine-modulated filter bank with length 80. The frequency responses of the analysis filters are shown above.

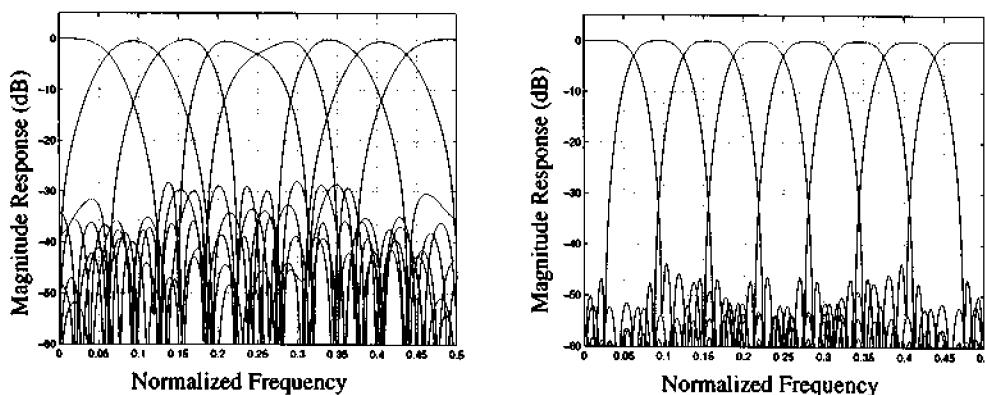
Final Design Example: M Channels with Linear Phase (GenLOT)

Linear phase filter banks are not cosine-modulated. For $M = 2$, they are constructed by factorization of the Daubechies polynomial or any symmetric halfband $P(z)$. They can also be designed by optimizing the lattice coefficients in the lattice structures [NgVaid1]. For $M > 2$, there is no simple spectral factorization method and one has to rely on lattice structure or time-domain approaches.



Frequency responses of a 10-channel linear-phase system with length 40.

Lattice structures exist for even M , when the length is a multiple of M (see Chapter 9 and [Soman, deQueiroz]). The figure above shows the frequency responses of a 10-channel linear-phase system with length 40. These are designed by optimizing the lattice coefficients for stop-band attenuation (left) and coding gain (right).



An NPR design (also linear phase) gave attenuation of 43.7dB.

One notices from those GenLOT designs that the stopband attenuation does not improve significantly as the filter length increases. One way to obtain high attenuation is to *trade off the reconstruction error and stopband attenuation* [Nguyen2]. An 8-channel PR design gave us attenuation of 28dB using the lattice structure. On the other hand, an NPR design (also linear phase) gave attenuation of 43.7dB in the last graph shown. The price was amplitude distortion of 0.0365 and aliasing of -47dB . The acceptability of a near PR design depends on the applications!

Chapter 11

Applications

11.1 Digitized Fingerprints and the FBI

The FBI has 30 million sets of fingerprints (300 million fingers) from felony arrests. I always assumed that they have my fingerprints too, because I once applied for a security clearance. (I believe it was granted. So far I have not been arrested. This is the first author, still at large.) The 40,000 sets of prints that arrive each day are divided into three groups:

- 5000 new prints to be saved
- 15000 repeat prints (recidivists) to be compared
- 20000 security clearances to check and return to other agencies.

Up to now, the prints are on cards in Washington. The file cabinets fill a whole floor in the FBI building. The final identification is always done by trained examiners. The “minutiae” that they watch for are ridge endings and bifurcations, which form permanently in childhood. In a British court, 12 of these indicators provide a legal match (fewer than 12 in most US courts — there are about 150 minutiae per finger). The FBI stores the cards in a special order, which begins with an automated classification using arches, loops, and whorls. Then come finer details until manual search is possible. Multiresolution is natural for fingerprints.

This data has to be digitized, and transmitted more quickly. The prints will be captured by electronic “live scans” instead of ink. At present most cards are simply mailed to the FBI. The criminal has gone from the booking station long before the fingerprints reach Washington. The new turnaround time, for digital information sent to the West Virginia office, is to be two hours (if requested).

At present, the rules do not allow the FBI to keep fingerprint files for juveniles. That group is responsible for a large fraction of breaking and entering felonies. They leave fingerprints all over the place! Some states do maintain files on juveniles, and it seems likely that changes in technology will bring changes in the law. The digital revolution is reaching criminology.

Our focus is on the specifications for the compression of gray scale images (256 levels of gray). The FBI chose a *wavelet/scalar quantization* (WSQ) algorithm. It was initially expected that the JPEG standard would win, but at compression of 15:1 and 20:1 the blocking from the DCT was severe. Ridges that are separated in the true image were found to merge during compression. This is unacceptable. It did not happen for wavelets. The linear phase 9/7 filter bank appeared just in time to be compared with the Daubechies 8/8 bank, and 9/7 was better for two main reasons:

1. Symmetry (and symmetric extension at the boundaries)
2. The image contents do not shift between the subbands.

A ridge has the same position within each subband, when filters have zero phase. The FBI confirmed that symmetric extension is better than circular convolution.

At 500 pixels per inch, ridges typically repeat every 10 to 16 pixels. The dominant frequency band is $\omega = \pi/8$ to $\omega = \pi/4$. The subband tree (Figure 11.1) goes down four levels in this range. All WSQ encoders will use the same tree. Symmetric filters up to 31 or 32 taps are permitted — and the 9/7 pair is recommended.

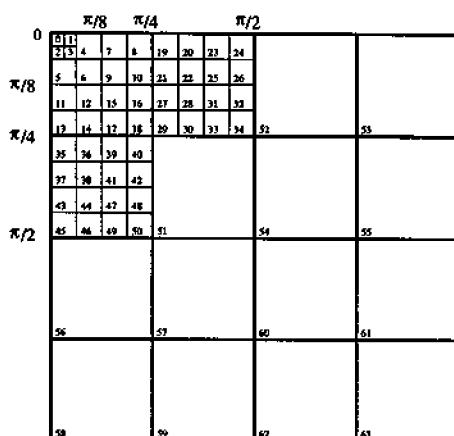


Figure 11.1: The WSQ subbands.

Each of the 64 subbands has its own uniform scalar quantization. The coefficient is set to $p = 0$ if it falls between $-Z/2$ and $Z/2$. A high percentage of the compression comes from this zeroth bin. The compressed image has strings of zeros, and the entropy coding step (Huffman coding) transmits a long string by giving its length. The other bin widths Q are smaller (about $.8Z$). Then the coefficient $a > 0$ falls into the p th bin in Figure 11.2 if $\frac{1}{2}Z + (p - 1)Q \leq a < \frac{1}{2}Z + pQ$.

Thus a is coded by the small integer p . The coding rule is similar for $a < 0$. The *decoder* (inverse direction) assigns to p the number a at the midpoint of the bin. The decoder also has an option to shift away from the bin center.

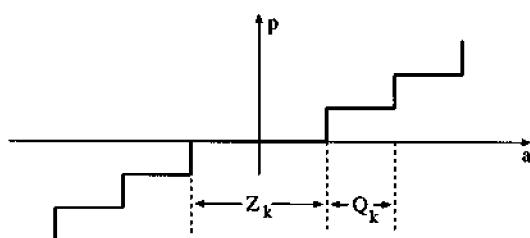


Figure 11.2: The quantization map from real numbers a to integers p .

The key question is the choice of Z and Q for each subband. This is the problem of *bit allo-*

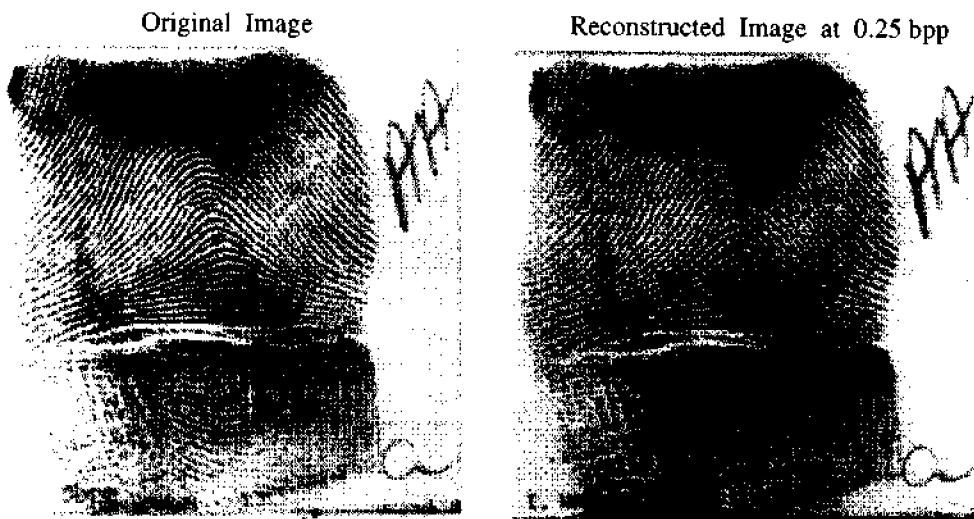
cation in Section 11.2. The FBI uses an empirical relationship $Q_k = qC_k / \log \sigma_k$ controlled by the variance σ_k of the k th subband. Flat signals, with small variance, contain little information. The bin widths are correspondingly larger. The constant C_k can give extra emphasis to subbands that happen to be at the scale of ridges and pores. Then the constant q controls the overall compression. In the FBI system, q is partly determined by the size of the blank background that surrounds the print. The ten megabytes of data per card are compressed by 15 : 1.

Theoretically, all bands should be at the same stage on their rate-distortion curves. The difficulty is to measure the distortion that our eyes actually perceive. It is not the L^2 norm! Energy is a convenient measure of error, but it does not agree well with human perception.

Experiments indicate that the improvement over the JPEG algorithm (the DCT in 8×8 blocks) *increases with the compression ratio*. At high compression, the blocking effects in the JPEG output overwhelm the signal. It becomes impossible to follow the ridge lines, and the FBI examiners frankly said *no*. One effort of Tom Hopper, who is leading the move to digital, is to convert all the groups who deal with fingerprints — including state and local police and the registry of motor vehicles. They have to believe in the investment, and the quality of the compressed signal is what they go by. Then they build the systems.

Entropy Coding

A set of 254 symbols captures bin values p from -73 to 74 and the lengths of zero runs up to 100 . A few escape symbols account for all remaining possibilities. In a highpass channel, the symbols $p = \pm 1$ might occur with frequency up to 25% each, and $p = \pm 2$ up to 5% each. Zero runs might account for 12% (length 1) and 6% (length 2) and 3% (length 3). The lowpass channels, with smaller bin widths, have more information in higher values of p . A block of subbands will use the same Huffman table to encode the quantized values. This yields the compressed image data in the figure below. The decoder reverses the entropy coding, then the quantization, and finally the wavelet transform.



11.2 Image and Video Compression

Image Compression

“A picture is worth a thousand words.” This English aphorism reminds us of the importance of images. It is especially true in the age of information highway and multimedia. Computers, fax machines, videophones, teleconferencing systems and storage devices impact our workplace. Text, data, sound, image and video clip are grouped together to send over data networks or to store. The amount of data is astronomical. Compression increases the throughput of the network and the capacity of the storage device. For satellite transmission, compression greatly reduces cost.

A 24-bit color picture with 256 by 256 pixels needs more than 0.2 MByte of storage. A high density diskette with capacity of 1.4 MB can store about 7 pictures. If the picture can be compressed by 50 to 1 without any perceptual distortion, the capacity of the diskette increases to 350 pictures. This is significant. The key point here is the notion of *perceptual* lossless compression. A good coding algorithm should study and incorporate the human visual system to exploit redundancy in the image.

There are many techniques for image coding. Subband coding is today the most successful. Pyramid coding was and is effective for high bit-rate compression. Transform coding based on the Discrete Cosine Transform became popular in the 1980s because of low complexity and effective bit allocation. This became the JPEG standard in image coding. For gray-scale images, it performs well for compression ratios up to 16 to 1. At 24 to 1 compression, JPEG’s synthesized image suffers from blocking, which manifests the short basis functions used in reconstruction.

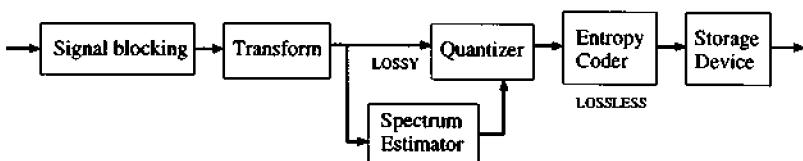


Figure 11.3: The steps of a transform-based image coder.

Subband coding using wavelets (this means tree-structured filter banks) avoids blocking at medium bit-rate, because its basis functions have variable length. Long basis functions represent flat background (low frequency). Short basis functions represent regions with texture. At a reasonable distance, one can not detect the errors easily. At low bit-rate, wavelet coding suffers from *ringing* when high frequencies (textures) are deleted. The ringing artifacts are significant around edges with high intensity. They have the shape of the basis functions that are emphasized in synthesis.

Part of the “Barbara” image and its Discrete Cosine Transform are in Figure 11.4. The image is blocked (8 by 8) and then transformed. The transformed subimage also has size 8 by 8. The intensities of the first few coefficients (the upper left corner of the transform) are the largest. The DCT preserves energy and the essential information is in those few coefficients. Quantization assigns more bits to these pixels. The objective of bit allocation is to minimize the distortion. The quantized subbands are then scanned and coded using lossless compression. This *entropy coder* watches for runs of zeros and transmits their length (roughly 3 : 1 compression for free).

Similar procedures are used in the wavelet-based transform coder. Figure 11.5 shows the

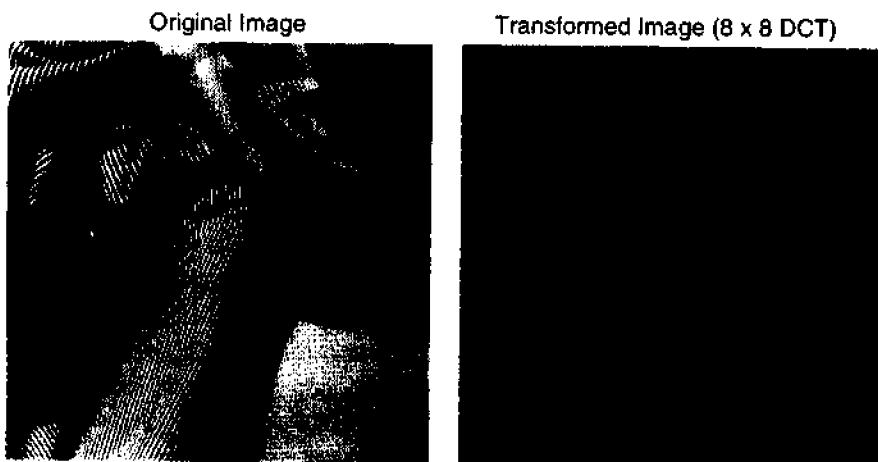


Figure 11.4: The discrete cosine transform shows most energy in the DC coefficients (bright points).

transform using a two-channel filter bank. The upper left subimage is obtained by lowpass filtering in both the horizontal and vertical directions, indicated by *LL*. The other three subimages (*much lower intensity*) have details involving high frequencies. The bit allocation algorithm will assign many bits to *LL* and few bits to *HH*. The normal number of iterations on the *LL* subimages is 4 or 5, as shown on the right side of Figure 11.5.

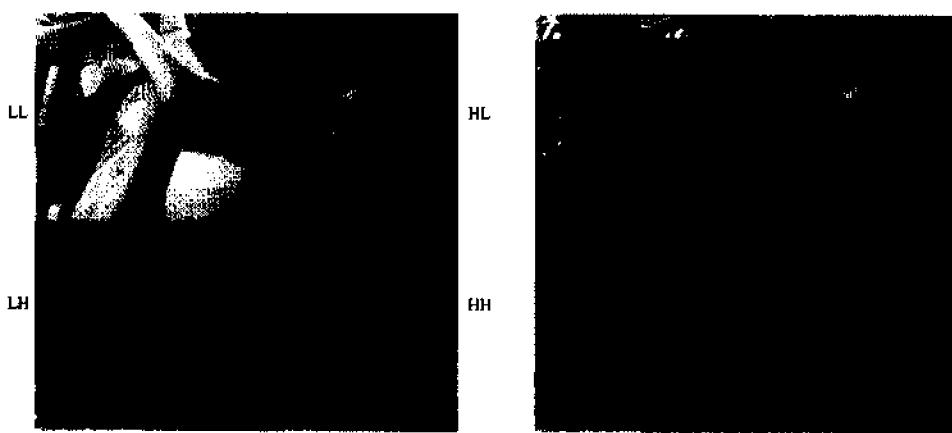


Figure 11.5: The discrete wavelet transform of "Barbara" (one level and four levels).

The subsections below discuss these building blocks in detail. The organization is:

- Choice of transformations and their effects on image coding.
- Bit allocation algorithm and quantization.
- Entropy coding algorithms.

- Error measures.
- Comparison of block transforms and wavelets.

Choice of Transformations and Their Effects on Image Coding

Consider the block transform coder in Figure 11.6. The input blocks are

$$\mathbf{v}(n) = [\quad x(nM) \quad x(nM - 1) \quad \cdots \quad x(nM - N + 1) \quad]^T.$$

Each block is transformed to length M by $y(n) = \mathbf{P}^T \mathbf{v}(n)$. The figure shows a uniform filter bank with M channels. The k th row of \mathbf{P}^T contains the filter coefficients $h_k(n)$. Examples are the DCT ($N = M$), the Lapped Orthogonal Transform ($N = 2M$), and the Generalized LOT ($N = LM$).

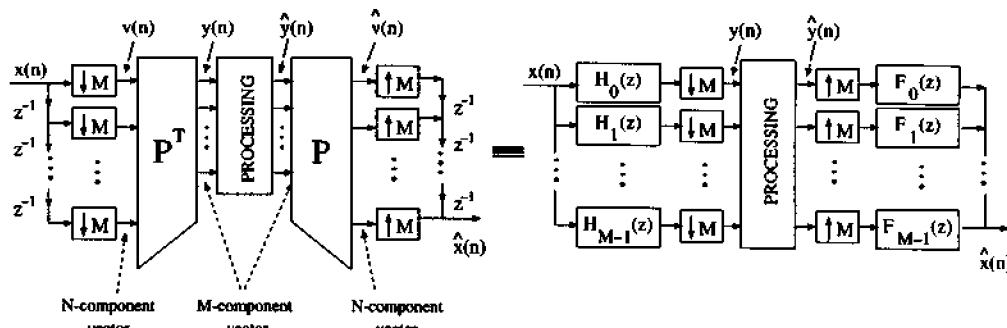


Figure 11.6: Block transform and the equivalent filter bank. The k th row of \mathbf{P}^T contains $h_k(n)$.

The subband signal $y(n)$ is quantized, entropy coded, and stored. In synthesis, the stored signal is decoded (to produce $\hat{y}(n)$) and then transformed by the matrix P . In the absence of quantization, $P = [P_0 \ P_1 \ \cdots \ P_{L-1}]$ gives an orthogonal transform:

$$\sum_{m=0}^{L-1-\ell} P_m P_{m+\ell}^T = \delta(\ell) I; \quad 0 \leq \ell \leq L - 1. \quad (11.1)$$

The synthesis filters are time-reversed versions of the analysis filters. The reconstructed signal $\hat{x}(n)$ is a linear combination of the basis vectors. What properties of those vectors (columns of P) will produce a good perceptual image coder?

- *They should be smooth and symmetric (or antisymmetric).* Smoothness controls the noise in a region with constant background. Symmetry allows the use of symmetric extension to process the image's borders.
- *They should decay to zero smoothly at both ends.* Figure 11.7(a) shows the basis functions of the DCT. Note that they do not decay to zero. This creates discontinuity between blocks (subimages) when the image is compressed. This *blocking artifact* can be seen from the reconstructed image in Figure 11.7(b). The compression rate is 32 to 1 (0.25 bpp).

- The bandpass and highpass filters should have no DC leakage. Higher frequency bands will be quantized severely. It is desirable for the lowpass band to contain all of the DC information. Otherwise, if the bandpass and highpass responses to $\omega = 0$ are not zero, we see the *checkerboard artifact* in Figure 11.8(b).

An equivalent form of the DC leakage condition can be derived for the lowpass filter: $H_0(2k\pi/M) = \delta(k)$ for $k < M$. These ω_k are the mirror frequencies.

- The basis functions should be chosen to maximize coding gain.
- Their lengths should be reasonably short to avoid excessive ringing and reasonably long to avoid blocking. Figure 11.9 shows transforms of length 8 (DCT) and 48 (GenLOT) used in 32:1 compression. The first transform has blocking and the second transform has ringing. Intuitively, one would like to represent texture by short functions and background by long functions. This is offered by wavelets.

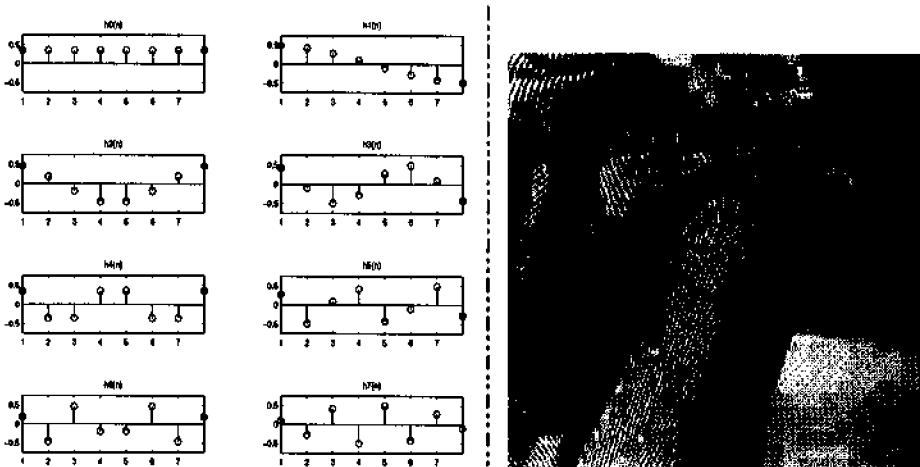


Figure 11.7: Blocking at 32:1 comes from discontinuities of the 8 DCT basis functions at edges.

- In the frequency range $|\omega| \leq \frac{\pi}{M}$, the bandpass and highpass responses should be small. This minimizes the quantization effect on bandpass and highpass.

Wavelet-based transform One attractive property of wavelets is their ability to adjust the lengths of basis functions. A four-level wavelet decomposition and its equivalent nonuniform filter bank are in Figure 11.10. The low frequency basis function is a cascade of interpolated versions of the lowpass filter H_0 . Its effective length is large. Higher frequencies are iterated less; the basis functions become shorter. The signal is approximated by a few basis functions. After four levels in Figure 11.5, most of the energy is in the lowpass subband. This upper left subimage is a coarse approximation of the original. The other bands add details. *Bit allocation becomes crucial*. Clearly, subimages with low energy levels should have fewer bits.

We comment further on desirable properties of a two-channel filter bank, now emphasizing what becomes important with *iteration*.

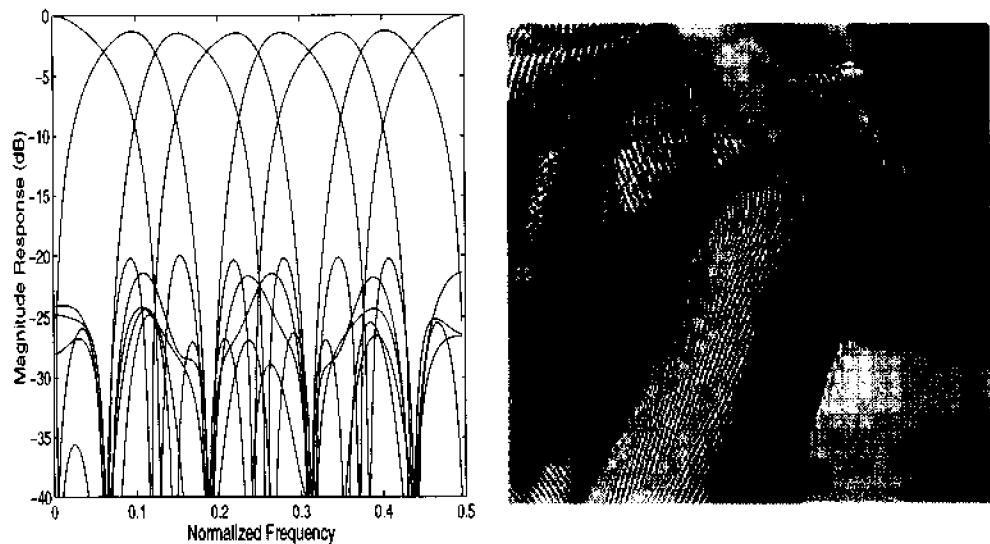


Figure 11.8: Frequency responses of GenLOT with $M = 8$ and $N = 24$. The bandpass responses at $\omega = 0$ (DC) are not zero. The DC component leaks over to produce *checkerboarding*.



Figure 11.9: Short DCT basis vectors produce blocking. Long GenLOT vectors produce ringing.

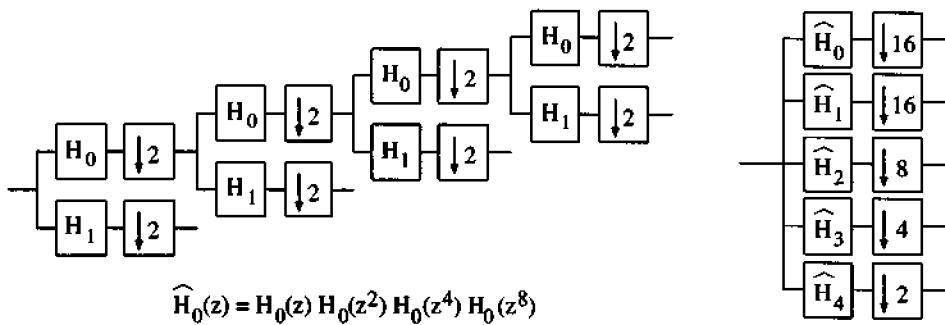


Figure 11.10: A four-level discrete wavelet transform and its equivalent nonuniform filter bank.

- *The synthesis scaling function should be smooth and symmetric.* Figure 11.11 shows several sawtooth subsections, plus two large jumps. Those jumps in the synthesis scaling function produce “blocking-like artifacts” in the reconstructed image.
- *The highpass analysis filter should have no DC leakage.* When output from this filter is quantized, we do not want to affect the DC component of the signal. It is desirable that the lowpass subband contains all the DC energy. Figure 11.12 shows the filter banks, their scaling functions, and the reconstructed image (at 1 bpp). Note the checkerboard artifact and the nonzero responses at $\omega = \pi$. Wavelet theory requires the lowpass filters to have at least one zero. This is also beneficial for image coding.
- *The analysis filters should be chosen to maximize coding gain.*
- *$F_0(z)$ should be long and $F_1(z)$ should be short.* A long $F_0(z)$ will help the coder to represent flat regions. A short $F_1(z)$ will minimize ringing due to quantization. Figure 11.13(a) shows blocking at 0.25 bpp from the short Haar filter. Figure 11.13(b) shows ringing for a longer 9-tap $F_1(z)$.
- *$H_1(z)$ should have good stopband attenuation to minimize the leakage of quantization noise into low frequencies.*
- How does one factor a given halfband $P(z)$ into $H_0(z)$ $F_0(z)$? One should choose $H_0(z)$ short (and therefore $F_1(z)$ short) and $F_0(z)$ long. $H_0(z)$ should have at least one zero at $\omega = \pi$. $F_0(z)$ should have many zeros at π to obtain a smooth scaling function. An example is the filter bank with length (2, 14) (factors of the maxflat filter). The total of eight zeros at π is the same as the (9, 7) FBI filter bank. Figure 11.14 shows the reconstructed images at 0.5 bpp using the (2, 14) and FBI systems. The PSNR is 24.68 dB and 25.30 dB for (2, 14) and (9, 7). The image qualities are comparable, although (2, 14) has smaller coding gain (9.45 dB) compared to 9.787 dB of the FBI system.

Bit Allocation and Quantization

After transformation the subband signals are quantized, entropy coded and stored (or transmitted). Figure 11.15 shows typical distributions for the subband signals. For bandpass and high-pass subbands, zero-mean Gaussian is a good approximation. The lowpass subband is image-dependent and its distribution is roughly uniform. Given M subimages and a fixed bit rate R ,

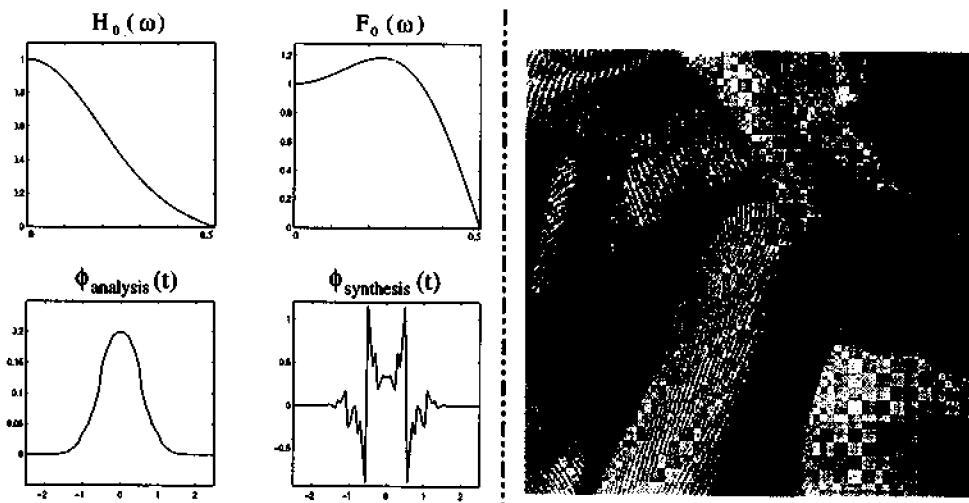


Figure 11.11: Frequency responses and scaling functions. The synthesis scaling function is rough and has large jumps. This produces blocking-like artifacts.

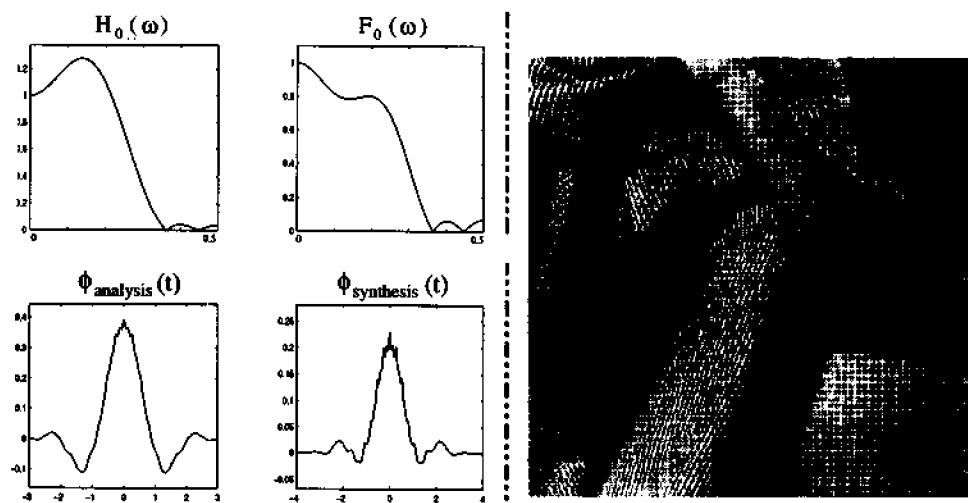


Figure 11.12: Frequency responses and scaling functions of a filter bank. The lowpass responses are not zero at $\omega = \pi$. This DC leakage yields a checkerboard in the reconstructed image.

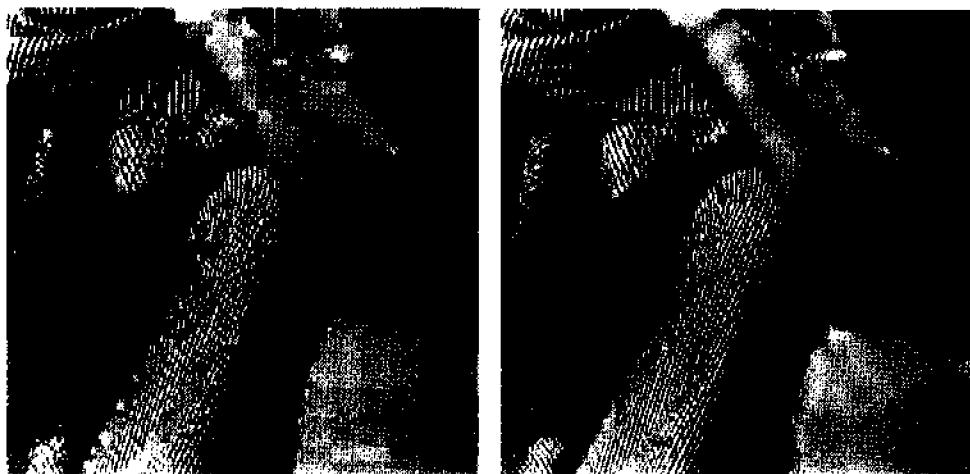


Figure 11.13: Blocking from a short synthesis lowpass filter. Ringing from a long synthesis high-pass filter.

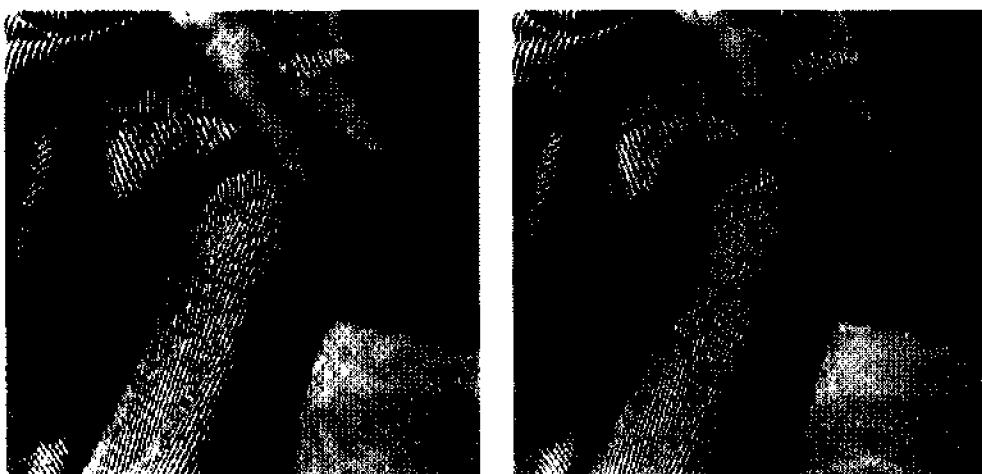


Figure 11.14: Image compression at 0.5 bpp using (2, 14) and (9, 7) filter lengths.

how does one assign bits to the subimages? The bit allocation algorithm needs a cost function D . Examples of D are distortion, energy and entropy. We discuss the minimization of distortion for a uniform quantizer with variable bit length next.

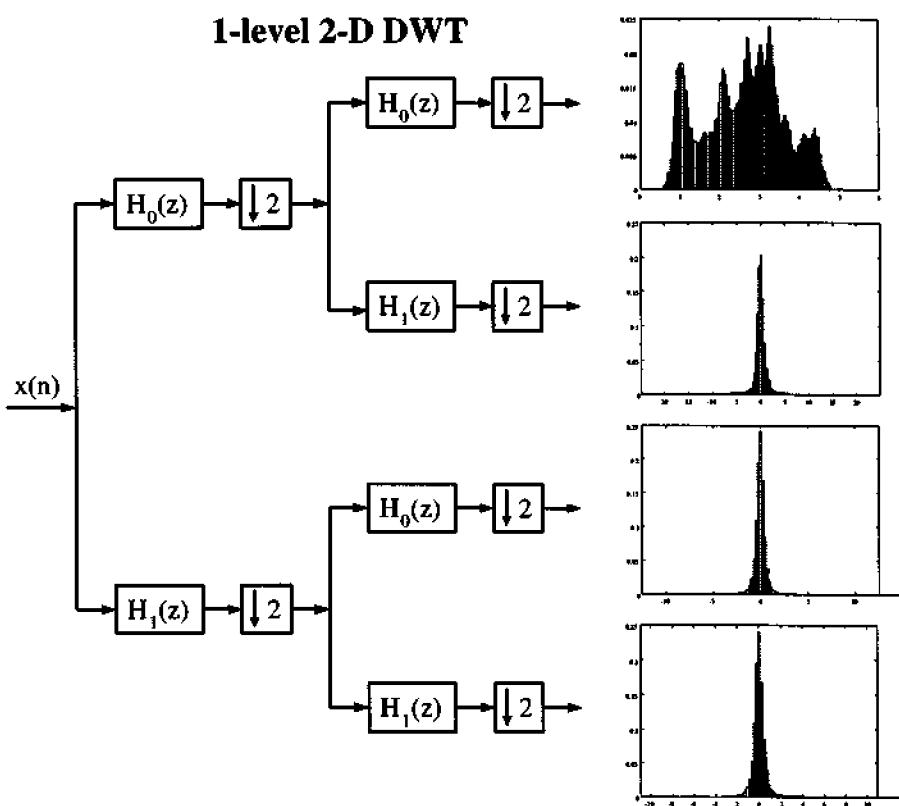


Figure 11.15: Typical distributions of subband coefficients.

Consider the 4-level wavelet decomposition in Figure 11.5. The original image size is 256 by 256 with 8 bits per pixel, for a total of 524,288 bits. How does one assign 16,384 bits to the subbands for a 32 to 1 compression? The numbers of pixels in the subbands go from 16 by 16 up to 128 by 128. Assigning one extra bit to the coefficients in the small subband does not increase the total number of bits as much as assigning the extra bit to the large subbands. Let

- N be the total number of bits in the original image
- M be the number of subbands ($M = 13$ for four levels)
- $\alpha_k = N_k/N$ be the relative subband size
- $b = (b_1, \dots, b_M)$ contain the bit lengths allocated to the subbands
- ϵ_k be the quantizer performance index
- σ_k^2 be the subband variance
- w_k be the weighting factor (for perceptual coding).

Define the quantization error and the total bit rate as:

$$D(\mathbf{b}) = \sum_{k=1}^M \alpha_k w_k \epsilon_k^2 2^{-2b_k} \sigma_k^2 \quad \text{and} \quad R(\mathbf{b}) = \sum_{k=1}^M \alpha_k b_k.$$

The bit allocation problem is to minimize $D(\mathbf{b})$ subject to $R(\mathbf{b}) = R_c$ = fixed bit rate. This can be solved using a Lagrange multiplier λ :

$$\min \{D(\mathbf{b}) + \lambda R(\mathbf{b})\} = \min \sum \alpha_k (w_k \epsilon_k^2 2^{-2b_k} \sigma_k^2 + \lambda b_k). \quad (11.2)$$

Assume that all $\epsilon_k = \epsilon$, and set $\tilde{\lambda} = \lambda/\epsilon^2$. Then (11.2) reduces to

$$\min \{w_k 2^{-2b_k} \sigma_k^2 + \tilde{\lambda} b_k\} \quad \forall k. \quad (11.3)$$

Differentiate (11.3) with respect to b_k to obtain a closed form solution:

$$b_k = \frac{1}{2} \log_2 \left(\frac{(2 \log_e 2) w_k \sigma_k^2}{\tilde{\lambda}} \right). \quad (11.4)$$

Then the bit-rate constraint $R(\mathbf{b}) = R_c$ becomes

$$\sum_{k=1}^M \alpha_k b_k = \frac{1}{2} \sum_{k=1}^M \alpha_k \log_2 \left(\frac{(2 \log_e 2) w_k \sigma_k^2}{\tilde{\lambda}} \right) = R_c.$$

This yields the multiplier $\tilde{\lambda} = \lambda/\epsilon^2$:

$$\tilde{\lambda} = 2^{\left\lceil \sum_{k=1}^M \alpha_k \log_2 \left(\frac{(2 \log_e 2) w_k \sigma_k^2}{2R_c} \right) \right\rceil} \quad (11.5)$$

In summary, the average subband variances σ_k^2 lead to $\tilde{\lambda}$, subject to the constraint. Then the allocated bit length b_k is computed from (11.4). When R_c is too small, the resulting bit length can be negative for subbands with very small signal variances. Since one can not assign negative bits, the bit allocation algorithm is restarted with these subbands removed (0 bits). The iterative algorithm is:

1. Find bit lengths b_k and set all negative b_k to zero.
2. Reduce the number of subbands appropriately, $M_{new} = M_{old} - M_{zero}$.
3. Adjust $R_c = R_{c,new}$ and repeat step 1. Stop if all $b_k \geq 0$.

For a linear-phase filter bank without orthogonality, the signal energy is not preserved. Quantization noise introduced in the subbands is amplified by the synthesis bank. Assuming that the quantization errors are uncorrelated and defining $B_k = \alpha_k \sum_n f_k^2(n)$, the reconstruction error variance is $\sigma^2 = \sum_{k=0}^{M-1} B_k \sigma_k^2$. B_k normally ranges from $0.9\alpha_k$ to $1.1\alpha_k$ for practical systems, and it is reasonable to assume that $B_k = 1$ (as for an orthogonal filter bank).

From b_k and σ_k , we compute the step size Δ_k for the quantizer. The maximum allowable quantization error for the k th subband is $T_k = c_k \sigma_k / 2^{b_k}$, where $c_k = 8$ is a reasonable choice. The step size Δ_k is normally chosen to be

$$\Delta_k = \max(2T_k, \Delta_{k,min}) \quad \text{for } b_k > 0 \quad \text{and} \quad \Delta_k = 2X_{k,max} + \epsilon \quad \text{for } b_k = 0. \quad (11.6)$$

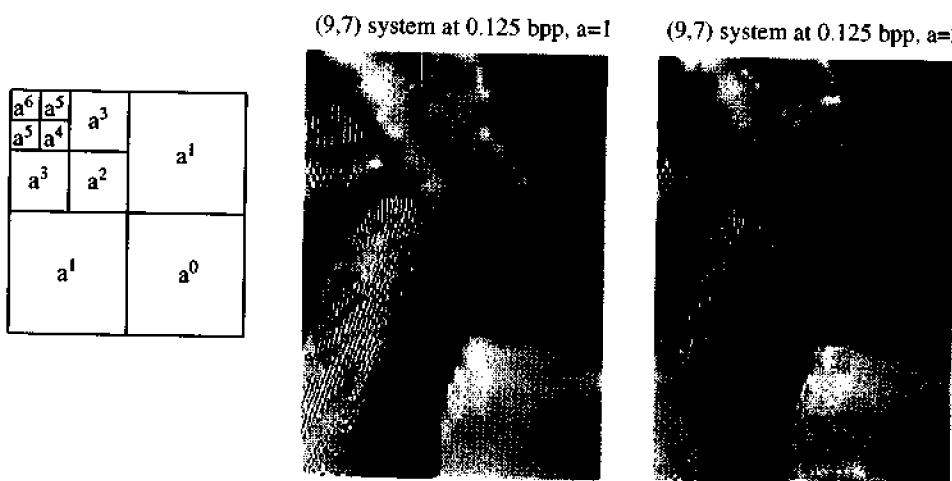
Here $X_{k,\max}$ is the largest nonzero value, $b_{k,\max}$ is the maximum number of bits for coding a nonzero normalized coefficient, and $\Delta_{k,\min}$ is the minimum step size to guarantee that $b_{k,\max}$ is not overridden for coding $X_{k,\max}$. The Huffman table of the sequential baseline coder has a limited size and should be kept small. The table is also quantized and transmitted. Therefore, Δ_k must be greater than or equal to the smallest δ_k that is not quantized to zero. Summarizing,

$$\Delta_{k,\min} = \max \left\{ \frac{X_{k,\max}}{2^{b_{k,\max}} - 1}, \delta_k \right\}; \quad \epsilon \geq \delta_k.$$

The exception for the case where $b_k = 0$ in (11.6) assures that all coefficients are quantized to zero if no bit is assigned. $\epsilon \geq \delta_k$ is necessary to keep all normalized coefficients $\tilde{y}_k = \text{round}(y_k/\Delta_k)$ smaller than 0.5.

The lowpass channel variance strongly depends on the signal. Its distribution is approximately uniform. Δ_1 computed from (11.6) would be too large since the uniform quantizer is more effective for uniform distribution (rather than Gaussian). This error is equalized appropriately by scaling the lowpass subband variance by $\tilde{\sigma}_1^2 = c_1 \sigma_1^2$ before bit allocation. c_1 is determined experimentally and is different for every signal. It is about 1.5 for the image Lenna.

Perceptual weighting for wavelet-based image coding It is well known that minimizing the squared error does not guarantee optimal results in the perceptual sense. At medium and low bit-rates, human eyes are less sensitive to loss of high frequencies. One needs to weight the quantization noise in the subband using the sensitivity of the human eye. But perceptual quality is complicated. We summarize here a simple and effective weighting scheme for the estimated quantization noise before bit allocation. More bits are allocated to low and medium bands, and high frequency noise is increased. The figure below shows the weight factors for three levels with $a > 1$ and the improvement for $a = 2$.

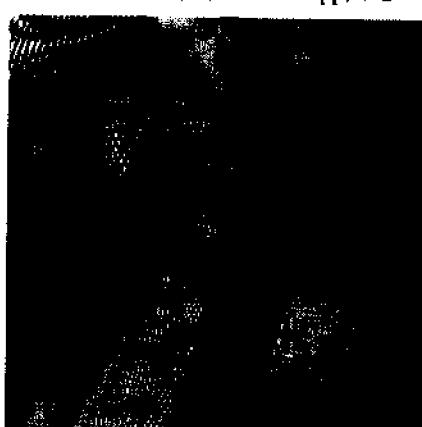


The choice of weight for an M -channel GenLOT is $w = [a^{M-1} \dots a \ 1]^T$. Extending to 2-D yields $W = w w^T$. Here are the reconstructed images using GenLOT of length 48, with $a = 1$ and $a = 2$. Note again the improvement with $a = 2$.

GenLOT(48) at 0.125 bpp, a=1



GenLOT (48) at 0.125 bpp, a=2



Since PSNR is maximal for $\alpha = 1$, this weighting is a tradeoff between perceptual quality and error measure of the reconstructed image. At lower rates, perceptual quality is more important. The weighting scheme is simple, efficient, and image independent. It does not guarantee perceptually optimal results! A reliable measure for perceptual quality (and the best weight) is an ongoing research problem.

Entropy Coding

After bit allocation and quantization, we have subimages with discrete levels represented by integers. How do we store or transmit these subimages? Many highpass coefficients are zero after quantization. These coefficients should be grouped so that the entropy coder can take full advantage of long strings of zeros. This is accomplished by *scanning*.

Run-length coding or Huffman coding or a combination should be used to reduce the redundancy of the images [PM]. We will discuss the baseline entropy coder which is a combination. JPEG also uses the baseline coding method.

Scanning of the discrete wavelet coefficients To demonstrate scanning, consider the three-level wavelet transform in Figure 11.16. Subbands 2, 5 and 8 are highly correlated since 2 is the coarse approximation of 5 and 5 is the coarse approximation of 8. Suppose the pixel at the upper left corner of subband 2 is zero. *Then it is very likely that the pixels in a 2×2 shaded square of subband 5 are zero.* Similarly, the pixels in a 4×4 shaded square of subband 8 are probably zero. One can group these pixels into an “AC sequence” of length 21 ($= 1+4+16$) by vertically scanning the shaded squares. Figure 11.16 also shows the scanning patterns for subbands 3, 6 and 9 (horizontal) and for subbands 4, 7 and 10 (diagonal). When the original image has size 32, the 16 pixels each in subbands 2, 3, 4 give 48 AC sequences. The low frequency band is scanned horizontally and grouped into the DC sequence of length 16. This scanning method is similar to the *zero-tree coder* proposed by [Shapiro].

Scanning of the block transform Consider an image of size 32×32 transforms to 16 blocks of size 8×8 , using an 8-channel GenLOT. The quantized coefficients are scanned and entropy-coded. The ℓ coefficients in block k are correlated with the ℓ coefficients in blocks $k \pm m$. There-

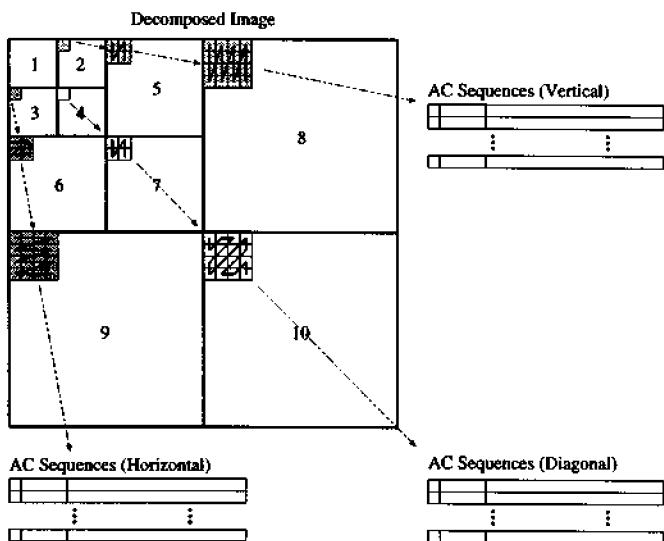


Figure 11.16: Scanning method used in the Discrete Wavelet Decomposition.

fore one should scan them in a zig-zag pattern, as shown in the figure. These scanned coefficients are grouped to sequences of length 16. There are 64 such sequences.

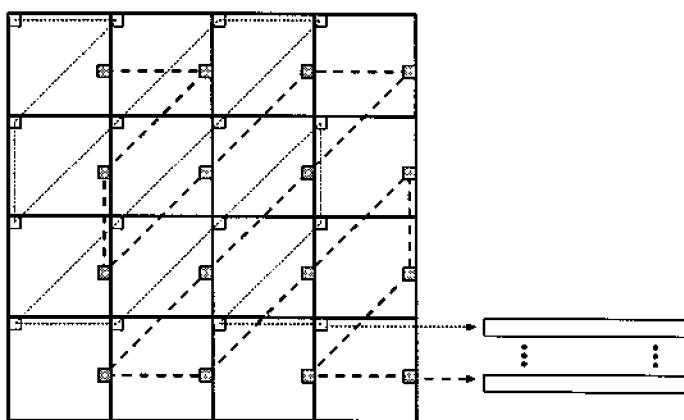


Figure 11.17: Scanning method used in the block transform.

Sequential Baseline Coding After scanning the quantized subimages, we have a set of DC and AC sequences to be stored. The baseline coder takes advantage of the correlation in the AC sequences. This algorithm combines Sequential Baseline Coding and Huffman entropy coding. The basic principle is similar to the JPEG coder, but does not restrict to the DCT.

Coding of the AC sequence: The sequence 9 0 0 2 0 1 0 0 0 – 3 0 0 3 0 – 1 – 1 has strings of zeros interlacing with nonzeros. An efficient representation remembers the number of zeros before each nonzero. **Symbol-1** is the pair (*Runlength*, *Size*) where *Runlength* specifies the

number of preceding zeros. *Size* determines the number of bits to encode the current nonzero. *Symbol-2* gives the (*Amplitude*) of the nonzero: *Size* = n corresponds to *Amplitude* less than 2^n (but not less than 2^{n-1}).

This representation of the example gives a string of *symbol-1* and *symbol-2*:

$$(0, 1)4 \ (2, 2)2 \ (1, 1)1 \ (3, 2)-3 \ (2, 2)3 \ (1, 1)-1 \ (0, 1)-1 \ (0, 0).$$

Note the terminal *symbol-1* (0, 0) at the end. Also (1, 1) and (0, 1) and (2, 2) occur twice in the string. Huffman entropy coding (see Figure 11.18) can exploit this redundancy in *symbol-1*. A simple way to code *symbol-2* “*a*” in binary is:

$$b = \begin{cases} a - 2^{\text{Size}-1} & \text{if } a > 0 \\ |a| & \text{otherwise.} \end{cases}$$

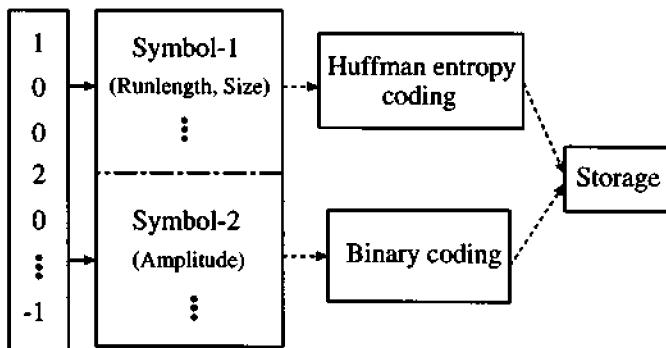


Figure 11.18: Steps in a Sequential Baseline Coder.

Coding of the DC sequence: DC coefficients measure the average energy of the input signal. There is usually a strong correlation between neighboring coefficients. For efficiency we use *differential coding*: save the first coefficient and then the *differences between successive coefficients*. These are coded as for AC coefficients. Since one would not expect long zero strings, *Runlength* is not used. *Symbol-1* only gives *Size*. An excellent source for Sequential Baseline Coder is [PM].

Three error measures are often used to compare coders and perceptual quality:

Mean Square Error Peak Signal Noise Ratio Maximum Error	$MSE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m, n) - \hat{x}(m, n) ^2$ $PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right)$ $MaxError = \text{Max } x(m, n) - \hat{x}(m, n) $	(11.7)
---	--	--------

The image is $M \times N$. The MSE and PSNR are directly related, and one normally uses PSNR to measure the coder's objective performance. At high rate, images with PSNR above 32 dB are considered to be perceptually lossless. At medium and low rates, the PSNR does not agree with the quality of the image.

Comparison of image coder based on block transforms and wavelets

Block Transform Image Coder: GenLOT with M channels Figure 11.19(a) shows the coding gain as a function of the overlapping factor N , for various values of M . Notice the large improvement from 4 channels to 8 channels. The added improvement for $M = 10$ and 16 channels is less. For fixed M , one observes a steady increase in coding gain as N increases. GenLOT with 8 channels is a good compromise between the implementation complexity and performance.

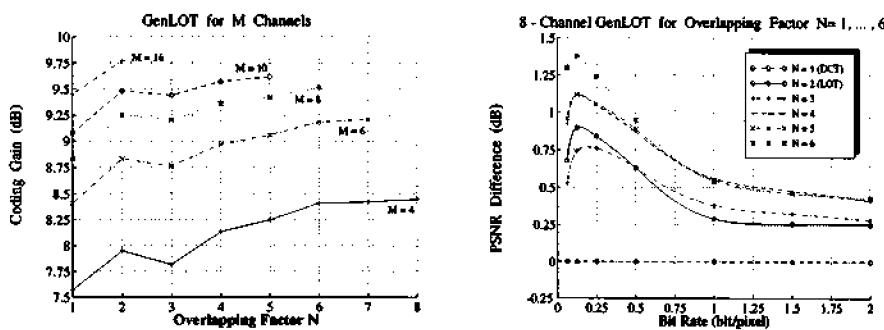


Figure 11.19: (a) Plots of coding gains as functions of overlapping factor. (b) Plots of PSNR differences for the “Lenna” image using 8-channel systems for various bit rates.

Figure 11.19(b) shows the PSNR difference between GenLOT and DCT, for several overlapping factors N . At high rate (≥ 1 bpp), the improvement is not significant. All block image coders would work well at high rate. At medium rate the improvement is larger, since DCT algorithms suffer from blocking. At low rate, blurring and ringing artifacts control the performance of the image coder.

One should also compare the *subjective* performance. Figure 11.20 shows the reconstructed images for DCT and GenLOT (length 48) at 1 bpp and 0.125 bpp. Although the PSNR improvement at 0.125 bpp is only 0.52 dB, one observes a large difference in subjective quality.

Wavelet-based Image Coder with L levels and p zeros at π All filters used in this study are factors of a halfband maximally-flat $P(z)$. One way to form symmetric $F_0(z)$ and $H_0(z)$ from $P(z)$ of length 15 is to assign four zeros at π and the four complex zeros to $F_0(z)$. Then $H_0(z)$ will have the remaining four zeros at π and the two real zeros. This yields the (9, 7) system used by the FBI. To obtain filter banks with even length (Type A), move one zero at π from $H_0(z)$ to $F_0(z)$. This yields the (10, 6) system.

Figure 11.21(a) shows the unified coding gain as function of the number of levels L and the p zeros at π in $P(z)$. For a given $P(z)$, we compute the unified coding gain for all linear-phase systems and plot the best value. See also [Maj1]. This plot assumes that the image is an AR(1) model with $\rho = 0.95$. The unified coding gain saturates around 4 or 5 levels of decomposition.

More than 5 levels of decomposition help very little. Type A and Type B systems show a similar coding gain.

The best filter bank for a given $P(z)$ is used in a wavelet-based study with four levels of decomposition for the “Lenna” image. The PSNRs of the reconstructed images are computed for various rates and plotted in Figure 11.22 for Type A systems. Notice how Haar wavelets are worst. Odd-length B systems show a similar PSNR performance.

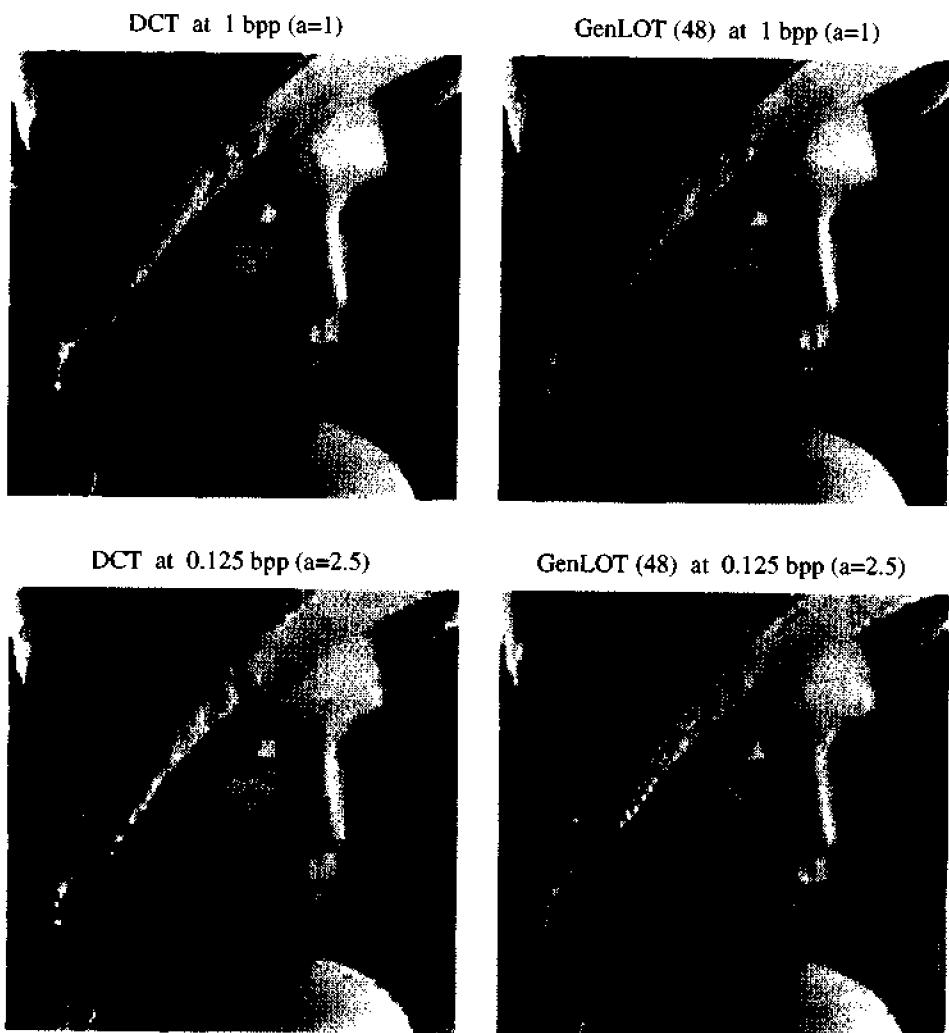


Figure 11.20: Reconstructed images using DCT and GenLOT (length 48) for 1 bpp and 0.125 bpp.

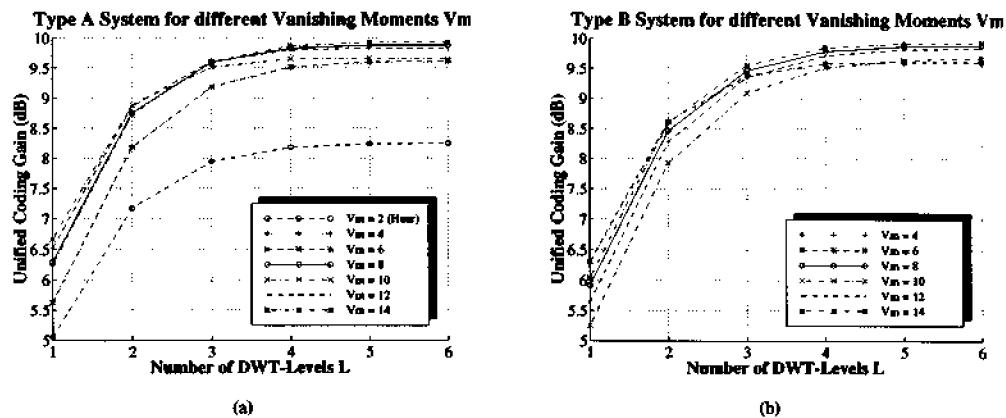


Figure 11.21: Unified coding gain for the best even-length Type-A systems and odd-length Type-B systems. V_m is the number of vanishing moments of the halfband filter $P(z)$.

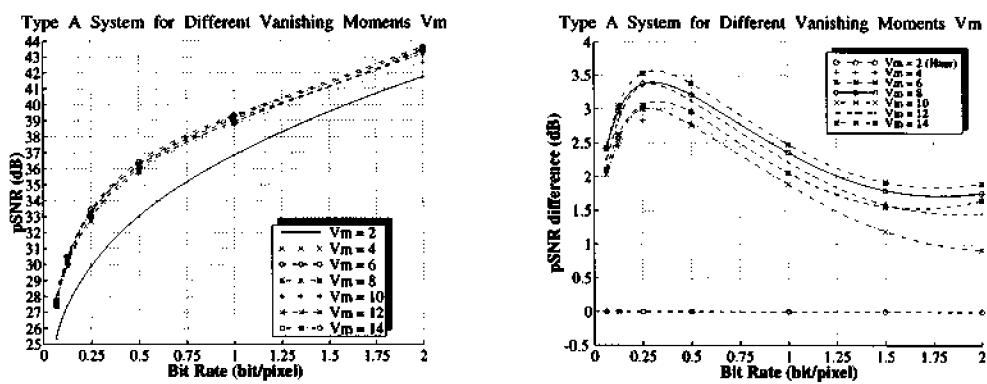


Figure 11.22: PSNR for even length filter. Difference from the PSNR of a DCT transform.

Discussion Given an image and a compression ratio, should one use a block transform (GenLOT) or a wavelet-based transform? At high rate (1 bpp), the perceptual quality is very similar. At medium rate, wavelet-based transforms perform a little better (GenLOT has ringing). At low rate, all transforms have their drawbacks and it is hard to pick one over the others. We can only display the outputs and tabulate the error measures as shown in Figure 11.23.

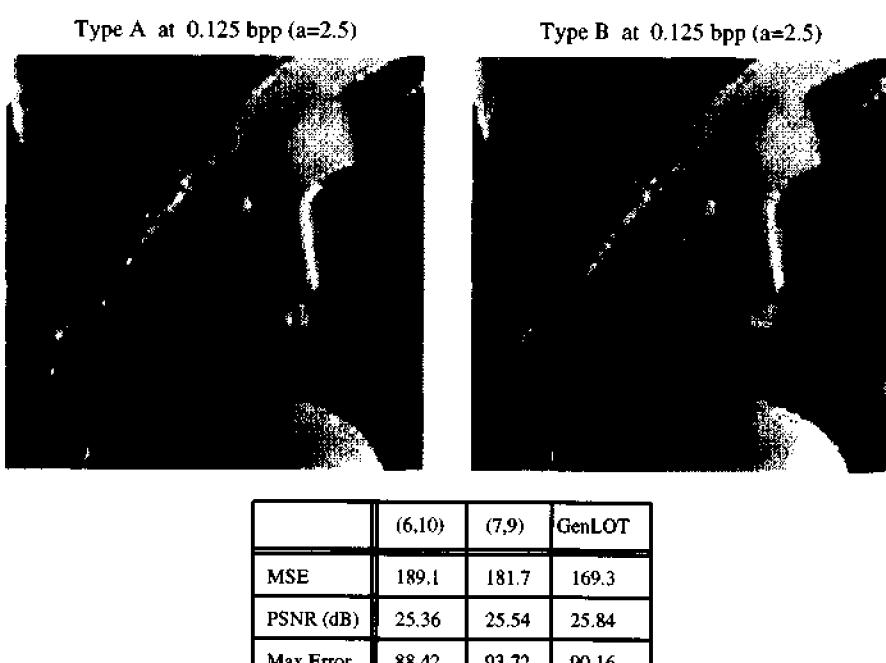


Figure 11.23: Reconstructed images using Type A (6, 10) and Type B (9, 7). Error measures include GenLOT (length 48) from Figure 11.21.

Software for image compression is available at

<http://saigon.ece.wisc.edu/~waveweb/QMF.html>

Video Compression

Commercial systems such as video-on-demand, videophone, video conferencing and multimedia applications are being planned and manufactured for home use. The key to their success is video compression. Video on the Internet will be at a low bit-rate!

Video signals are sequences of 2D images updating at about 30 frames per second. *The new dimension is time.* One can extend separable processing from 2D to 3D. Then a video compression system would use a 3D separable filter bank in the front end. The transformed sequences are quantized and entropy coded. A bit allocation algorithm based on rate distortion theory can be used to find the optimal bit assignment.

Another approach to video compression is based on *motion estimation*. At 30 frames/second, the information in frames m and $m \pm 1$ is highly correlated. Suppose that one can estimate the

motion vectors for all pixels, to indicate where each part of the image moves in the following frame. Then it is sufficient to send the first frame (compressed) and the *motion vectors*. At the synthesis bank, the first frame is reconstructed and subsequent frames are formed by using the motion vectors (plus small corrections to the image). The quality of the reconstructed image depends on the accuracy of the estimated motion vectors.

Consider the image coder based on 8×8 blocks transformed by the DCT. The first frame is quantized, entropy coded and transmitted. The second frame is transformed into blocks. For a specific block (K, L) , the search algorithm considers the neighbor blocks $(K \pm 1, L \pm 1)$ to estimate the motion vectors. These are also coded and transmitted. However, an imperfect estimate reduces the quality of the reconstructed second frame. Consequently, one also needs to transmit the expected residual error. The MPEG video standard [MPEG2] employs both forward and backward predictions for motion vector estimation.

Similar algorithms based on the wavelet transform are being developed. Where MPEG deals with subblocks, the wavelet algorithm has blocks with different sizes at different resolutions. Motion estimation is more complicated since there are several scales to search. We *estimate motion first on a coarse scale, then on finer scales*. The support regions also depend on the filter lengths. The memory requirement for MPEG is lower, but multiresolution is more powerful. An early comparison is in [Zafar].

Problem Set 11.1

- If A, B, C, D occur with probabilities 0.4, 0.3, 0.2, 0.1, a Huffman entropy coder will create a binary tree:

A, B, C, D have codes 0, 10, 110, 111

What are the Huffman codes for X, Y, Z with probabilities 0.2, 0.5, 0.3?

11.3 Speech, Audio, and ECG Compression

Psychoacoustic model of the ear To design effective algorithms for speech and audio compression, one needs to know how hearing works. The more one knows, the better the algorithms. Psychoacoustics is the study of hearing, from which quantitative models are built. The models are based on years of extensive tests on humans, which led to these conclusions:

- Hearing is associated with critical bands.* These nonuniform frequency bands can be approximated by tree-structured filter banks. In speech compression, the filter bank is a four-level dyadic tree. Audio compression is based on either M -band uniform or M -band tree-structure cosine-modulated filter banks (Figure 11.24). There is always a tradeoff between complexity and accuracy in the approximation of critical bands. The filter bank in Figure 11.24(c) is more complicated than the other two. However, it approximates the critical bands more accurately.
- Around any frequency f_m there is masking.* An adjacent frequency f with magnitude below $T(f_m, f)$ is masked by f_m and is not audible:

$$T(f_m, f) = \begin{cases} M(f_m) \left(\frac{f}{f_m}\right)^{28}; & f \leq f_m \\ M(f_m) \left(\frac{f}{f_m}\right)^{-10}; & f > f_m. \end{cases} \quad (11.8)$$

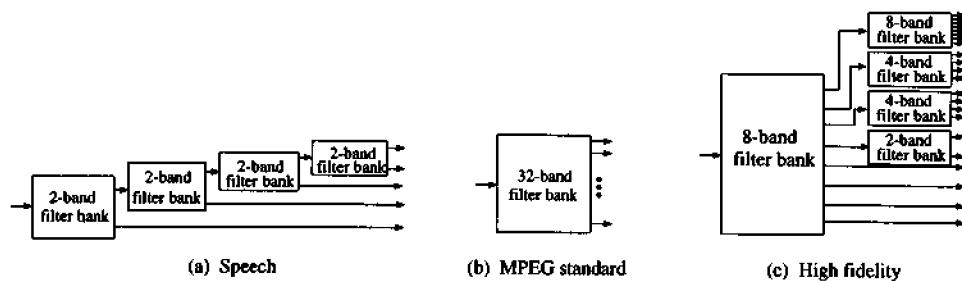
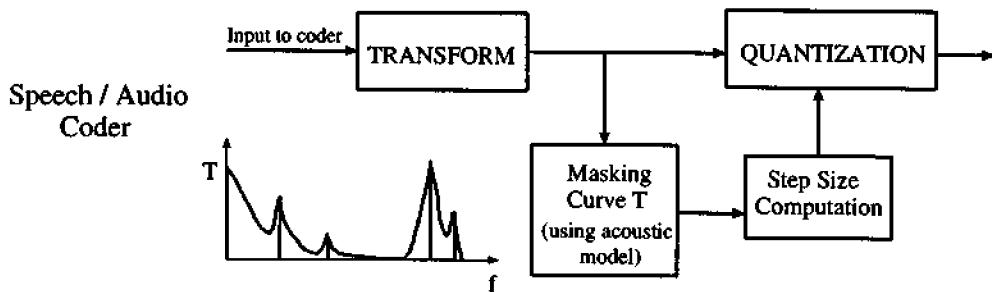


Figure 11.24: Tree-structured filter banks used to approximate the critical bands.

The quantity $M(f_m)$ is the masking threshold at frequency f_m , independent of the signal. The graph of T is linear on a log-log scale. Quantization noise can be freely added to the signal as long as it is below $T(f_m, f)$.



In a speech/audio compression system, the signal is transformed by a tree-structured filter bank. The frequency allocation approximates the critical bands of the human ear. The frequencies f_m with significant power are detected, and their masking envelopes $T(f_m, f)$ are computed. The combined masking envelope forms the *masking curve* of the signal. The quantization noise level is kept below the masking curve. Out-of-band masking noise is negligible as long as the frequency responses of the filter banks have high attenuation.

The allocation of B bits to the signal, assigning b_k bits to subband k , tries to keep the quantization noise inside the masking curve:

$$\text{Minimize} \quad \sum_{k=1}^M \frac{1}{12} \left(\frac{2 p_k}{2^{b_k} - 1} \right)^2 \frac{1}{\sigma_{m,k}^2} \quad \text{such that} \quad \sum_{k=1}^M b_k = B.$$

Here $\sigma_{m,k}^2$ is the masking power of the k th subband and p_k is its peak value.

Speech Compression

Speech compression is important in mobile communications, to reduce transmission time. Digital answering machines also depend on compression. The bit-rates are low, typically 2.4 kbit/s per second to 9.6 kbit/s/second. The best algorithms use either linear predictive models or sinusoidal models.

Speech is classified into *voiced* and *unvoiced* sounds. Voiced sounds are mainly low frequency. In CELP (code excitation linear predictor) the voiced sound is modeled as the output of an all-pole IIR filter with white noise as input. The filter coefficients are found by linear prediction. This filter represents the transfer function of the vocal tract. In a sinusoidal transform, the voiced sounds use a sinusoidal basis. Unvoiced sounds (like sss) have components in all frequency bands and resemble white noise. Model-based techniques achieve reasonable performance at low rates.

At more than 16 kbits/second, subband coding is effective and compatible with the models. Psychoacoustics has associated human hearing to nonuniform critical bands. These bands can be realized roughly as a four-level dyadic tree (Figure 11.24a). For sampling at 8 kHz, the frequency bands of the dyadic tree are: 0–250 Hz, 250–500 Hz, 500–1000 Hz, 1000–2000 Hz and 2000–4000 Hz. These bands can be quantized and coded depending on subband energy; the average signal to noise ratio is maximized. And the *noise masking property* is used.

High-Fidelity Audio Compression

Music signals have no models. The frequency spectrum of a harp is not similar to that of a piano. Subband coding, which makes no assumption on the signal model, is a natural choice. Consider an audio signal of CD quality, sampled at 44.1 kHz with 16 bits of resolution. The total bit-rate is 705.6 kbytes/second. For multimedia applications, one would like to compress it to a range from 64 to 192 kbytes/second (11:1 to 4:1). High-fidelity audio compression implies that there is no perceptual loss in the reconstructed signal. This is crucial for digital audio broadcast and satellite TV, where sound quality is the most important feature. Applications of audio compression systems are:

- Digital audio broadcasting
- Satellite TV, High-Definition TV
- Contribution and Distribution links
- Production (tapeless studio, editing systems)
- Storage devices (studio & consumer market)
- Multimedia applications

The noise floor (stopband attenuation) of the filter banks should be below –96 dB (16 bits at 6 dB per bit), although lower attenuation is always preferred. To minimize noise from adjacent subbands, the stopband attenuation is required to have steep roll-off rate. Interested readers should consult [Dehery, Stautner, Brandenburg].

Electrocardiogram Compression

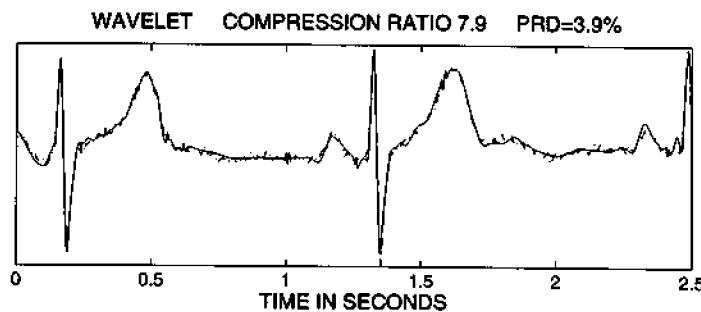
ECG waveforms are signals measured from the heart. They provide essential information to the cardiologist. Normally, a 24-hour recording is desirable to detect heart abnormalities or disorders. Storage requirements can range from 26 Mbytes, with one lead to the heart and 12 bit resolution sampled at 200 Hz, to 138 Mbytes for a system with two leads and 16 bit resolution sampled at 400 Hz. Compression is needed at a low bit-rate for both storage and telemedicine.

An ECG compression algorithm is judged by its ability to minimize the distortion while retaining all significant features of the signal. An accepted error measure is the percent root-mean-square difference (PRD). Let x_{or} and x_{re} be the original and reconstructed signals of length N :

$$\text{PRD} = \left\{ \sum [x_{or}(n) - x_{re}(n)]^2 / \sum [x_{or}(n)]^2 \right\}^{1/2} \times 100\%$$

Reconstruction with low PRD does not necessarily mean clinical acceptance. The crucial requirement is not to distort the diagnostic information used by the physician. This is a challenging problem since GenLOT and wavelet compression typically produce blocking and ringing. Moreover, ECG is normally used as input to classification. If compression does not change the outcome of the detection and classification algorithms, it is acceptable.

An ECG waveform (dashed line) is shown with its reconstruction (solid line). The compression ratio is 7.9 to 1, using the (13, 11)-tap filter bank. The reconstructed image preserves the significant features of the original and the PRD is 3.9%.



Techniques such as the turning point algorithm, amplitude zone time epoch coding (AZTEC), coordinate reduction time encoding system (CORTES), and the FAN algorithm compress the data by discarding relatively insignificant information [T]. The reconstructed signal is obtained by interpolating the stored samples. These algorithms are simple to implement, but they can produce significant distortion. Figure 11.25 shows the original and reconstructed waveforms using the AZTEC and FAN algorithms. We also show the PRD as function of average bit rate for the AZTEC, FAN and wavelet algorithms. The original signal has 12 bits per sample. In all our tests, wavelet-based compression gave the best objective measures.

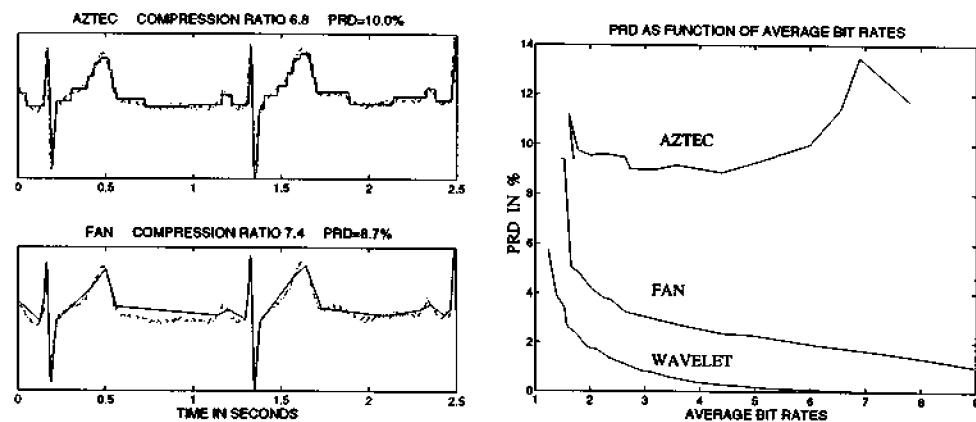


Figure 11.25: ECG Compression using the AZTEC, FAN and wavelet-based algorithms.

The blocks of the wavelet-based ECG algorithm are the same as for image coding. The input is divided into segments of N samples and these vectors are transformed (four levels, linear phase). The spectrum estimator computes a bit allocation to the subbands. They are quantized (scalar uniform) and entropy coded. The segment size N is important in determining the com-

pression ratio and the corresponding PRD. A large N increases the variance of the subband signals and the distortion. By experimenting with many sets of ECG data, we conclude that a block size of 2000 is reasonable.

Software for ECG compression is available at

<http://saigon.ece.wisc.edu/~waweb/QMF.html>

11.4 Shrinkage, Denoising, and Feature Detection

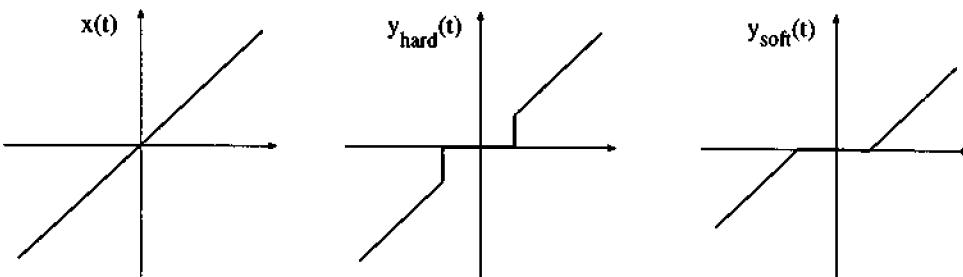
Wavelet Shrinkage

In the L -level wavelet decomposition of a signal, the number of wavelet coefficients with significant energy is small. This is a direct consequence of the approximation property of the wavelets, assuming a sufficient number of vanishing moments (say $p > 2$). The signal can be accurately represented by a small number of coefficients. *Wavelet shrinkage*, developed by Johnstone and Donoho, selects these coefficients based on *thresholding*. The wavelet shrinkage algorithm decomposes the signal into L levels and then:

- For each level, we select a threshold and apply “hard thresholding”. This will zero out many small coefficients, which results in efficient representation. Thresholding is a lossy algorithm; the original signal can not be reconstructed exactly. An alternative is soft thresholding at level δ , chosen for compression performance or relative error. The outputs $y_{\text{hard}}(t)$ and $y_{\text{soft}}(t)$ with threshold δ are

$$y_{\text{hard}}(t) = \begin{cases} x(t), & |x(t)| > \delta \\ 0, & |x(t)| \leq \delta \end{cases} \quad \text{Hard thresholding}$$

$$y_{\text{soft}}(t) = \begin{cases} \text{sign}(x(t))(|x(t)| - \delta), & |x(t)| > \delta \\ 0, & |x(t)| \leq \delta \end{cases} \quad \text{Soft thresholding.}$$



The signal in Figure 11.26 is basically lowpass with noise added. We show the reconstructed signal with threshold levels $\delta = 35$ and 100 . After thresholding at these levels, 92.8% and 93.4% of the coefficients are zero. The L_2 norm recoveries are 99.99% and 99.98%, respectively. Reconstruction with $\delta = 35$ is closer to the original since the threshold is lower. But it uses more terms.

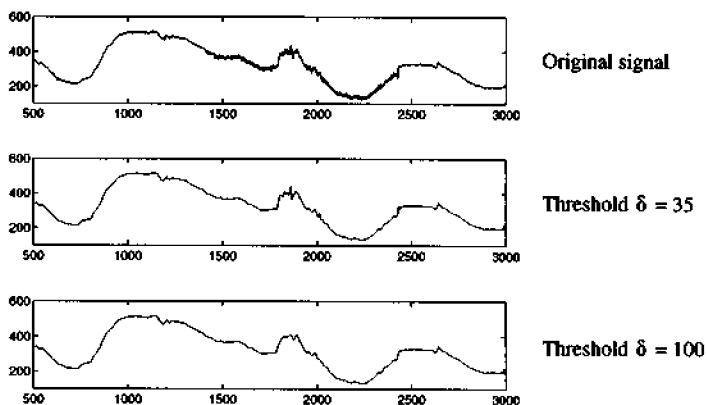


Figure 11.26: Hard thresholding generated using the *Wavelet Toolbox* from *MathWorks*.

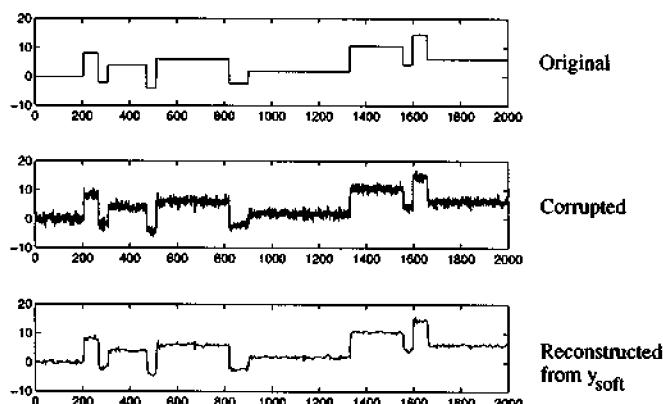
Denoising

Thresholding generally gives a lowpass version of the original signal. By selecting δ , one can suppress the noise $w(n)$ in the signal $x(n) = u(n) + \sigma w(n)$. A simple example of $w(n)$ is Gaussian white noise $N(0, 1)$. *For denoising we use soft thresholding.*

The noise power σ^2 is assumed to be much smaller than the signal power. Moreover, the signal is assumed to have low frequency components, whereas the noise source is white. Thresholding the detailed coefficients will also remove some of signal's power. It is generally impossible to filter out all the noise without affecting the signal. Algorithms selecting the threshold levels include Stein's Unbiased Risk Estimate, fixed threshold, Minimax criteria or a combination [Donbf1].

The piecewise constant signal below is corrupted by Gaussian white noise. The corrupted signal is decomposed using the Daubechies wavelet D_6 . The coefficients at level 4 are thresholded using Stein's Unbiased Risk Estimate. Notice that the reconstruction consists of the original signal and *some* of the noise.

In both wavelet shrinkage and denoising, the output is a cleaned-up version of the input. This works only when one knows the signal characteristics in advance. The algorithm will distort the desired signals when thresholding is applied.



Discontinuity Detection

A discontinuity in the signal can be detected by Haar wavelets. A discontinuity in the first derivative can be detected by D_4 . The signal in Figure 11.27(a) is linear, except for an interval of zero slope. The D_4 wavelet can represent straight lines exactly ($p = 2$). The two discontinuities are being captured in the wavelet coefficients. Although wavelets with longer supports could be used, D_4 is the best choice since it has good time localization.

Figure 11.27(b) shows a signal with discontinuity in the second derivative. The wavelet used is D_8 , which can represent parabolas exactly (so could D_6). Even though the discontinuity can not be seen from the signal, it is clearly detected by the coefficients in the first level of the decomposition.

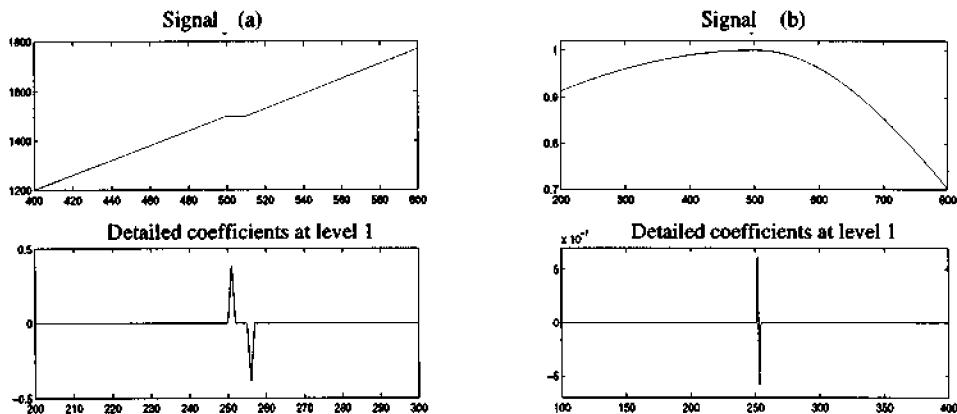


Figure 11.27: Detecting discontinuity in the first derivative and second derivative. (These plots are generated by the Wavelet Toolbox.)

Detecting discontinuities using a wavelet transform works well for signals with no noise. When noise dominates, one will obtain many false alarms (wrong detections). For reliable detection, one needs to follow the coefficients for several scales. Mallat pioneered this important application in continuous time for the case with no noise. The section below elaborates on the multiscale feature detection and points out its advantage comparing to the conventional feature detection.

Multiscale Feature Detection

The step edge in Figure 11.28 can be detected from the detailed coefficients using Haar wavelets. An alternative approach uses the Canny edge detector in image processing [Canny]. Both approaches work well for a signal with no noise. The output is clean and the edges can be detected. For the same input with added noise, on the right side, there are many false alarms (wrong detections).

Figure 11.29 shows the same noisy signal with its multiscale representation. Note the occurrence of false alarms at each scale. *The consistency between the peaks at all scales allows the edge locations to be detected.*

There are several issues in a multiscale feature detector. Our example uses *peak* as an indicator for the *edge* feature. However, one could also use *zero crossing* as an indicator. Given a

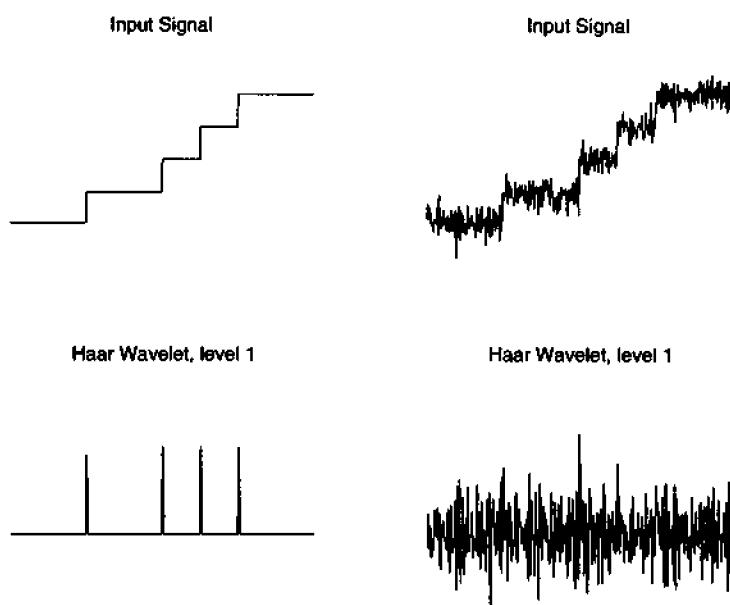


Figure 11.28: Step edge detection using Haar wavelets. The Canny edge detector is very similar. The clean step signal (left) is easy. The signal with noise (right) is difficult at one level.

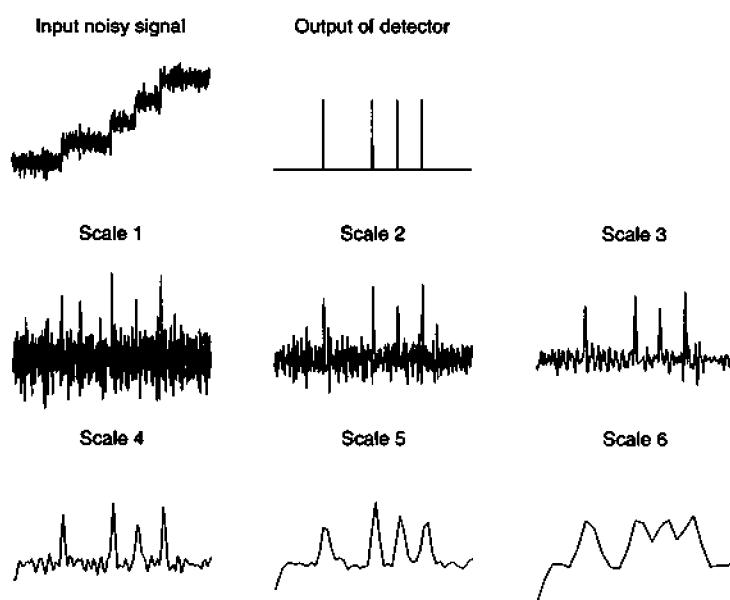


Figure 11.29: Step edge detection using the outputs at different scales.

feature, which indicator is the best one to use? The second issue is the choice of filter bank. How can one design filters that give a multiresolution representation of the indicators? The third issue is the design of an optimal detector for various noise models. Finally, many algorithms are derived in continuous time and can not be used for discrete-time implementation [Mt]. [HaNgCh] presents a general approach to discrete-time multiscale feature detection.

Problem Set 11.4

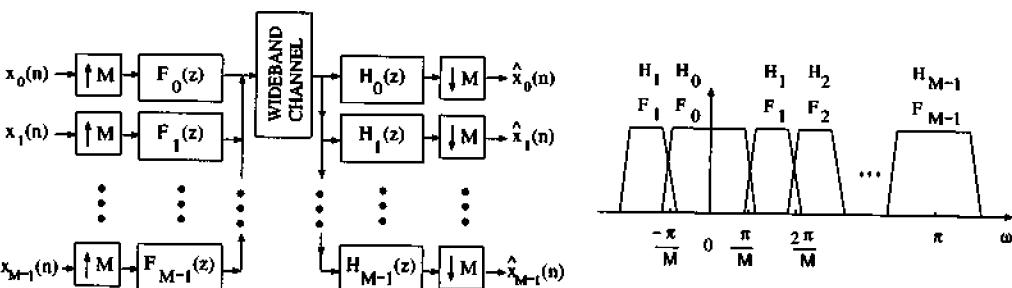
- For the signal $x(t) = \sin t$, draw the graphs of $y_{hard}(t)$ and $y_{soft}(t)$ after thresholding with $\delta = \frac{1}{2}$.

11.5 Communication Applications and Adaptive Systems

Transmultiplexers

A *transmultiplexer* is a filter bank with synthesis first. This reverse order is depicted in the figure below. M signals come in, and they are combined into *one* transmitted signal. Then the receiver has to reconstruct the M separate signals.

Low-bandwidth input signals $x_k(n)$ are upsampled and filtered. The combined signal goes through a high-bandwidth channel. The received signal is filtered and downsampled to the original rates. The outputs $\hat{x}_k(n)$ may suffer from *distortion* and *crosstalk* because of the decimation, interpolation, and non-ideal filtering.



The output $\hat{x}_k(n)$ depends on all M inputs by $\hat{X}_k(z) = \sum S_{kk}(z)X_k(z)$. The transfer functions $S_{kk}(z)$ give the distortion. The remaining $S_{k\ell}(z)$, $\ell \neq k$, are the crosstalk functions. The objective is to find $H_k(z)$ and $F_k(z)$ such that $S(z)$ is a diagonal matrix, $S_{kk}(z) = \delta_{kk} z^{-N_k}$. This would be a *perfect reconstruction transmultiplexer*. The design solutions for the PR transmultiplexer are closely related to those in the PR filter bank!

Let $H_k(z)$ and $F_k(z)$ be the analysis and synthesis filters of a perfect reconstruction filter bank. [Vet3, KoNgVa] show how the filters $H_k(z)$ and $z^{-1}F_k(z)$ yield a perfect reconstruction transmultiplexer (for an ideal communication channel).

Discrete Multitone Modulation Transceivers

Transmultiplexers can be used in conjunction with discrete multitone modulation transceivers. Discrete multitone (DMT) or multicarrier modulation is a class of orthogonal frequency modulations. The transmitted data is split into several bit streams and used to modulate several carriers. The spectrum is divided into parallel orthogonal and narrowband subchannels. Each carrier occupies one subchannel. This concept was proposed thirty years ago, but did not receive much attention because of the difficulty in implementation using analog technology.

The advance of digital signal processors and related hardware has made the multicarrier modulation a preferred structure over the single-carrier QAM system. There is important interest in multicarrier modulation for high-speed data transmission over the twisted pair channel of a digital subscriber line. As a result, discrete multitone has been proposed as a standard for high-speed digital subscriber line (HDSL) and asymmetric digital subscriber line (ADSL) communication.

Consider a typical channel where the attenuation variation could be as large as 40 dB. A linear equalizer for a single-carrier QAM system would increase the noise, whereas a decision-feedback equalizer would be too complex. For many channels, multicarrier modulation approximates a constant transfer function in each subchannel. Equalization becomes very simple. Other advantages of multicarrier modulation are the reduced effect of impulse noise as a result of large symbol duration; the flexibility of not transmitting in the corrupted subchannels in case of narrowband interference; and the flexibility of transmitting important data in subchannels with high SNRs.

One disadvantage is the large peak-to-rms ratio of the transmitted signals, which might lead to nonlinear distortion. And the implementation complexity must be controlled. When the frequency separation Δf is $\frac{1}{T}$, where T is the symbol duration, the multicarrier system can use the DFT. That avoids implementing several carriers using analog technology. This is one of the main attractive features of DFT-based multicarrier modulation.

Given an ideal channel, orthogonality is ensured by the $\frac{1}{T}$ spacing between subcarriers, and they can be independently demodulated. In a dispersive channel, the carriers are no longer orthogonal. Interference depends on the spectral overlap between subchannels. DFT-based multicarrier systems, with rectangular pulse shape and -13 dB sidelobes, can have significant interchannel interference. One could employ pulse shaping to reduce spectral overlap, but the resulting transmultiplexer is not necessarily orthogonal (nor perfect reconstruction).

An alternative is to use a *cosine-modulated* orthogonal transmultiplexer. Its spectral containment leads to superior robustness against narrowband radio frequency interference (RFI). Interested readers should consult [RiPrNg, SaTz, TzTzPh, TzTzRe] for algorithms and comparisons between DFT and cosine-modulated multicarrier systems.

Adaptive Systems

Adaptive systems are important when the signals or environments are changing with time. Typical applications are inverse filtering, room acoustics modelling, and echo cancellation. We also mention adaptive array processing (beamforming and direction finding) and channel equalization. In these applications, an FIR filter models the signal or the environment. Its coefficients are adapted in time to minimize an error measure. Popular adaptive algorithms are LMS (Least-Mean-Square) and RLS (Recursive-Least-Square). A detailed treatment of adaptive filter theory is in [H, PRCN].

The adaptive system in Figure 11.30 is used in inverse filtering. The objective is to find $f(n)$ such that the error $e(n)$ is as small as possible. Here, the desired signal $d(n)$ is distorted by an unknown system $h(n)$ and is corrupted by noise. The z -transform of $e(n)$ is

$$E(z) = [F(z) H(z) - z^{-N}] D(z).$$

Assuming that $H(z)$ is an FIR filter, the ideal choice for $F(z)$ is $z^{-N}/H(z)$. But normally $F(z)$ is constrained to be FIR. Its length affects the convergence rate and the error.

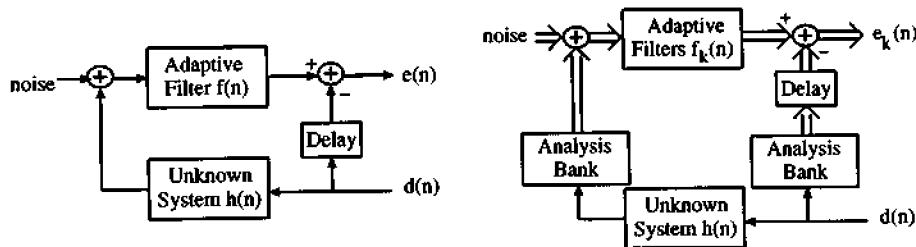


Figure 11.30: Adaptive systems in inverse filtering (left) and using a filter bank (right).

If the spectrum of $H(e^{j\omega})$ is complicated, the inverse spectrum is equally complicated and the resulting $f(n)$ is long. By separating $H(e^{j\omega})$ into many bands $H_k(e^{j\omega})$, the inverse filter in the k -th subband would not be too large. These M subbands in Figure 11.30 are adapted using M FIR adaptive filters $f_k(n)$, much shorter than $f(n)$ since the analyzing spectrum is simplified band by band. The convergence rates are faster. This is a tradeoff between hardware complexity and convergence rate.

Figure 11.31(a) shows a very different adaptive system. From the error $[H(z) - F(z)] D(z)$, one observes that $F(z)$ is no longer an inverse but is an approximation to $H(z)$. The length of $f(n)$ is comparable to that of $h(n)$. This system is used to cancel telephone echo coming from an impedance mismatch of the residence line and the switching center. It is most noticeable and annoying for long distance calls. Here, $s(n)$ is the speech signal, $d(n)$ is the reference signal, $y(n)$ is the echo signal received at the handset, and you hear $x(n)$.

The objective of $f(n)$ is to reproduce $s(n)$ while suppressing the echo $y(n)$. For a complicated and nonlinear echo path, an adequate $f(n)$ is too long. **A bank of shorter filters $f_k(n)$ is better.** The signals $s(n)$ and $d(n)$ are decomposed into M subbands, on which adaptation is performed.

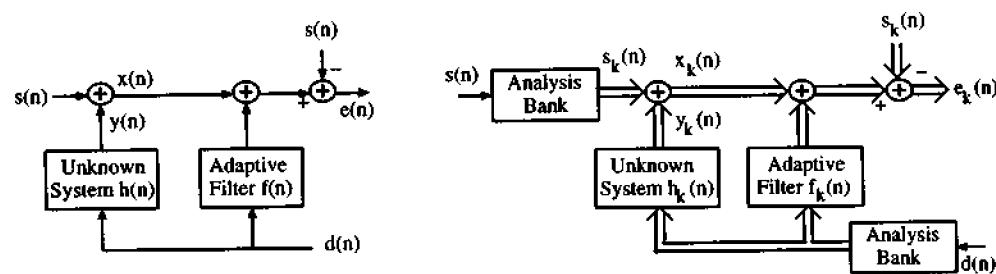


Figure 11.31: Adaptive systems for echo cancellation (left) and using a filter bank (right).

Filter bank adaptive systems give an improved convergence rate and error reduction [GilVet,

JinWoLu]. Subbands with large energy can be adapted by long filters, whereas subbands with insignificant energy can be either discarded or adjusted using short filters. The price is the additional complexity (minimized by using lattice structure or polyphase). Two-channel adaptive banks are designed in [JinWoLu] and compared with spline and Daubechies wavelets.

11.6 Wavelet Integrals for Differential Equations

The possible application of wavelets to differential equations is important. In principle, this application is available. *In practice, it is not yet a real success.* The overall approach is familiar, but the competition with other methods is severe. We do not necessarily predict that wavelets will win.

The unknown function $u(t)$ (or $u(x)$ or $u(x, y)$) is to be approximated by a finite combination $b_1\phi_1(t) + \dots + b_N\phi_N(t)$. These “trial functions” are chosen in advance. They are polynomials or sinusoids in the spectral method. The $\phi_k(t)$ are piecewise polynomials in the finite element method. Presumably they are scaling functions and wavelets in the “wavelet method.”

A key point is the option to use different scales $\Delta t = 2^{-j}$. The grid becomes a *multigrid*. The number of scale levels can vary with the position t or x , to produce an *adaptive mesh*. Flexibility comes from multiresolution. But the wavelet mesh does not achieve the total flexibility of the finite element method.

Up to this point in the book, all signals were assumed known. Now the coefficients b_1, \dots, b_N are unknowns because $u(t)$ is unknown. There are two outstanding rules for producing N equations for b_1, \dots, b_N :

1. *Collocation:* Apply the differential equation at N points t_1, \dots, t_N .
2. *Galerkin method:* Choose N test functions $g_1(t), \dots, g_N(t)$. Replace an equation like $u''(t) = f(t)$ by N “weighted residual equations:”

$$\int (b_1\phi_1''(t) + \dots + b_N\phi_N''(t)) g_j(t) dt = \int f(t) g_j(t) dt, \quad 1 \leq j \leq N. \quad (11.9)$$

When the test functions g_j are delta functions $\delta(t - t_j)$, this is collocation. When the g_j are the same as the ϕ_j , this is the Rayleigh-Ritz method. When the g_j are powers of t , this is the moment method. The trial functions and test functions should satisfy all *essential* boundary conditions in the problem—these are Dirichlet conditions like $u(0) = 0$. They need not satisfy *natural* boundary conditions like $u'(1) = 0$.

Collocation seems doubtful for wavelets, which are not smooth. The Galerkin method may also look doubtful, because of the integrals in (11.9). But these integrals can be computed by a quadrature formula, in which the coefficients are found once and for all. Then an integral of $f(t)\phi(t)$ does not require point values of $\phi(t)$, only of the smoother function $f(t)$:

$$\int_a^b f(t)\phi(t) dt \approx \sum_{k=1}^r a_k f(t_k). \quad (11.10)$$

This section discusses several quadrature rules. We have written $\phi(t)$ but similar rules are needed also for the wavelet $w(t)$. We will choose equally spaced evaluation points, so that overlapping integrals can share the same t_k (and $f(t)$ may be known only by those samples anyway).

Note that integrals are often computed by parts. Derivatives are moved away from rough functions and onto smooth functions. When the trial and test functions are the same ($\phi_j = g_j$), integration by parts changes $\int \phi''(t)g(t) dt$ into $-\int \phi'(t)g'(t) dt$. This needs only one derivative, not two.

We expect exact integrals for functions chosen in advance, and quadrature rules for integrals involving inputs $f(t)$. We discuss quadrature rules and then exact integrals.

Quadrature Rules

The usual test for accuracy is appropriate for smooth functions $f(t)$. The idea is to try $f(t) = \text{polynomial}$. The rule (11.10) has accuracy d if it gives the correct integral for all polynomials of degree less than d . The first d terms in the Taylor series of $f(t)$ are integrated correctly, so the error is from the d th derivative:

$$\int f(t)\phi(t) dt = \sum a_k f(t_k) + O(\|f^{(d)}\|).$$

Changing from t to $2^j t$, this error will be of order $(\Delta t)^d$ when the scale is $\Delta t = 2^{-j}$:

$$\int f(t)2^{j/2}\phi(2^j t) dt = \sum a_k 2^{-j/2} f(2^{-j} t_k) + O((\Delta t)^d). \quad (11.11)$$

We study quadrature formulas at unit scale. Then (11.11) gives the error at other scales.

Example 11.1. One-point quadrature. The rule has only one term:

$$\int f(t)w(t) dt \approx 0 \quad \text{and} \quad \int f(t)\phi(t) dt \approx f(t_1). \quad (11.12)$$

The first rule is strange but its accuracy is $d = p$, because the wavelet has p vanishing moments. The exact integral is zero up to $f(t) = t^{p-1}$.

The second rule is correct for $f(t) \equiv 1$ provided $\int \phi(t) dt = 1$. Thus $d = 1$ at least. The accuracy increases to $d = 2$ when the evaluation point is $t_1 = \int t\phi(t) dt$. (The rule is then exact for $f(t) = t$.) This is a low-level example of the Gauss idea, to double the accuracy by choosing intelligent t_k as well as a_k . But higher-order Gauss has unequally spaced t_k and is not ideal for these applications. We follow instead the excellent paper [Sweldens-Piessens], which also gives the coefficients a_k for high order quadrature rules.

How can you find this number $\int t\phi(t) dt$? By definition, it is the first moment M_1 of $\phi(t)$. It equals the first moment $m_1 = \sum kh(k)$ of the filter coefficients! There is a simple recursion that gives the moments $M_n = \int t^n\phi(t) dt$ exactly from the moments $m_n = \sum k^n h(k)$. As always, the answer is in those coefficients $h(k)$, when we use the dilation equation for $\phi(t)$:

$$\begin{aligned} 2^n \int t^n \phi(t) dt &= \sum 2h(k) \int (2t)^n \phi(2t - k) dt \\ &= \sum h(k) \int t^n \phi(t - k) dt \\ &= \sum h(k) \int (t + k)^n \phi(t) dt \\ &= \sum_{j=0}^n \binom{n}{j} \sum h(k) k^j \int t^{n-j} \phi(t) dt. \end{aligned}$$

This says that $2^n M_n = \sum_{j=0}^n \binom{n}{j} m_j M_{n-j}$. Bring M_n to the left side:

$$\text{Moment formula } M_n = \frac{1}{2^n - 1} \sum_{j=1}^n \binom{n}{j} m_j M_{n-j}. \quad (11.13)$$

In particular $M_1 = m_1$. This is the best evaluation point for wavelet integrals — but it is not generally an integer. The D_4 scaling function has $M_1 = m_1 = (1 + \sqrt{3})/4$.

Three comments. First, orthogonal scaling functions have $M_2 = M_1^2$ (Problem 10). The one-point quadrature $\int f(t)\phi(t) dt \cong f(M_1)$ is *third-order* accurate. It is correct for $f(t) = 1, t, t^2$. Second, the Daubechies polynomials are nearly “interpolating” at the points $t = M_1 + k$:

$$\phi(M_1) \approx 1.0002 \quad \text{and} \quad \phi(M_1 + 1) \approx -0.0004 \quad \text{and} \quad \phi(M_1 + 2) \approx 0.0002.$$

See [K] for graphs and discussion. Third, by using $3p$ instead of $2p$ coefficients we can construct “coiflets” that have *zero moments* M_1, M_2, \dots, M_{p-1} . Then the one-point rule $\int f(t)\phi(t)dt \approx f(0)$ is p th order accurate.

Exact Wavelet Integrals

We have already computed the inner products $a(k) = \int \phi(t)\phi(t+k) dt$ between translates of the scaling function. We will review the key idea, which gives $a(k)$ in terms of the filter coefficients $h(k)$. We never want to do an exact integral any other way. Generally we can’t! The applications to differential equations require many more integrals, for example

$$\int w(t)w(t+k) dt \quad \text{and} \quad \int \phi'(t)\phi'(t+k) dt \quad \text{and} \quad \int t^m \phi(t)\phi(t+k) dt.$$

The last integrand has three factors. Freely available software now allows *four factors* [Kunoth]. These integrals may be needed on an interval I (usually of length 1 or 2^{-n}) as well as on the whole line. Then the integral has a one-zero box function $\chi_I(t)$ as an extra factor.

In all cases the key idea is to use the two-scale equation for each factor. There are two-scale equations for $\phi(t)$ and $\phi'(t)$ and $w(t)$ and the box function $\chi_I(t)$. Recall what happens when two-scale equations are substituted into typical integrals:

$$\begin{aligned} a(k) &= \int \phi(t)\phi(t+k) dt = \int \left(\sum 2h(n)\phi(2t-n) \right) \left(\sum 2h(n)\phi(2t+2k-n) \right) dt \\ a_w(k) &= \int \phi(t)w(t+k) dt = \int \left(\sum 2h(n)\phi(2t-n) \right) \left(\sum 2h_1(n)\phi(2t+2k-n) \right) dt \end{aligned}$$

The next step is to replace $2t$ by t on the right side. Then dt becomes $\frac{1}{2}dt$. The results are very different in our two examples:

$a(k)$ becomes a combination of $a(k+\ell)$: *eigenvalue problem* $\mathbf{a} = \mathbf{T}\mathbf{a}$

$a_w(k)$ also becomes a combination of $a(k+\ell)$: *explicit equation* $\mathbf{a}_w = \mathbf{T}_w\mathbf{a}$.

The equation $\mathbf{a} = \mathbf{T}\mathbf{a}$ is homogeneous. Its solution must be suitably normalized. The equation $\mathbf{a}_w = \mathbf{T}_w\mathbf{a}$ gives \mathbf{a}_w in terms of the known \mathbf{a} . The matrices are doubled-shifted Toeplitz:

$$\mathbf{T} = (\downarrow 2)2\mathbf{H}\mathbf{H}^T \quad \text{and} \quad \mathbf{T}_w = (\downarrow 2)2\mathbf{H}_1\mathbf{H}^T. \quad (11.14)$$

This pattern is quite typical. Section 7.2 carried out the steps leading to \mathbf{T} . We now carry out the same steps for other important integrals (with more factors). Then we deal with integrals involving $w(t)$, which do *not* yield eigenvalue problems. For integrals involving $w(2t)$, the highpass matrix with diagonals $\mathbf{h}_1(0), 0, \mathbf{h}_1(1), 0, \mathbf{h}_1(2), 0, \dots$ replaces \mathbf{H}_1 .

Integrals of Products of $\phi_i(t)$ and $\phi'_i(t)$

We will describe the exact computation of a much wider class of integrals $\mathbf{b}(k)$. All functions to be integrated must be *refinable*. They satisfy two-scale dilation equations (also called refinement equations). The coefficients for $\phi_0(t), \phi_1(t), \phi_2(t)$ are $2\mathbf{h}_0(k), 2\mathbf{h}_1(k), 2\mathbf{h}_2(k)$. The dilation equation is substituted for each $\phi_i(t)$ and for any derivatives. (Each derivative of $2t$ produces an extra 2.) Changing the variable of integration brings $2t$ back to t and yields a two-scale equation for the integral:

$$\mathbf{b}(n) = 2^\alpha \sum g(n) \mathbf{b}(2k - n). \quad (11.15)$$

That is an eigenvalue problem $\mathbf{b} = 2^\alpha (\downarrow 2)\mathbf{G}\mathbf{b}$. The eigenvector is \mathbf{b} , and the operator $(\downarrow 2)$ changes k to $2k$. Our first problem is to find the coefficients in g from the coefficients in $\mathbf{h}_0, \mathbf{h}_1$, and \mathbf{h}_2 . *The two-scale equation for the integral \mathbf{b} follows from the two-scale equations for $\phi_0(t), \phi_1(t), \phi_2(t)$.*

Our presentation follows [Kunoth], who has created a valuable and freely available code. She works in d dimensions with $x = (x_1, \dots, x_d)$ and allows a product of $\ell + 1 = 4$ functions. In principle any product of functions and their derivatives and dilations and translations is workable. We will begin in one dimension with $\ell = 2$ — thus a product of three functions inside the integral. The letters \mathbf{b} and g are different from hers, since her conventions with 2's are not the same.

The integrals to be computed are sometimes called *connection coefficients*:

$$\mathbf{b}(k) = \mathbf{b}(k_1, k_2) = \int_{-\infty}^{\infty} \phi_0(t) D^{\mu_1} \phi_1(t - k_1) D^{\mu_2} \phi_2(t - k_2) dt.$$

The function $\phi_0(t)$ might be the standard box with dilation coefficients $\frac{1}{2}, \frac{1}{2}$. The functions $\phi_1(t)$ and $\phi_2(t)$ might be scaling functions, when the integral arises from Galerkin's method. If the differential equation is nonlinear, or has variable coefficients, that gives more factors ($\ell > 2$) in the integral. We are not now integrating wavelets! It is the scaling function that has a homogeneous dilation equation and leads to an eigenvalue problem. Integrals involving $w(t)$ are computed afterward. The filters $\mathbf{h}_0, \mathbf{h}_1, \mathbf{h}_2$ are all *lowpass*.

Our standard example is $a(n) = \int \phi(t)\phi(t+k) dt = \int \phi(t-k)\phi(t) dt$. (From now on we have $t - k$ instead of $t + k$; no problem.) It has $\ell = 1$ with $\mu_1 = 0$; no derivatives. Or it has $\ell = 2$ with $\phi_0(t) \equiv 1$ and $(\mu_1, \mu_2) = (0, 0)$. In this familiar example, the matrix is $\mathbf{G} = \mathbf{H}\mathbf{H}^T$ and the exponent is $\alpha = 1$. The double-shift matrix is $\mathbf{T} = (\downarrow 2)2\mathbf{H}\mathbf{H}^T$ as usual. This example will be a useful check, when the coefficients $g(n)$ come from the autocorrelation $g = \mathbf{h} * \mathbf{h}^T$:

$$g(n) = \sum \mathbf{h}(k)\mathbf{h}(k - n). \quad (11.16)$$

We jump directly to the key formula with three factors ϕ_0, ϕ_1, ϕ_2 :

$$g(n_1, n_2) = \sum \mathbf{h}_0(k)\mathbf{h}_1(k - n_1)\mathbf{h}_2(k - n_2). \quad (11.17)$$

To establish that formula, and to allow derivatives of the functions $\phi_i(t)$, substitute the dilation equations into the integral $b(k_1, k_2)$. The derivatives bring 2^{μ_1} and 2^{μ_2} :

$$\int_{-\infty}^{\infty} \left[\sum 2h_0(k)\phi_0(2t-k) \right] \left[2^{\mu_1} \sum 2h_1(m)\phi_1(2t-2k_1-m) \right] \left[2^{\mu_2} \sum 2h_2(p)\phi_2(2t-2k_2-p) \right] dt.$$

Change $2t - k$ to T and $2dt$ to dT . Set $\alpha = 2 + \mu_1 + \mu_2$. Our integral is

$$2^\alpha \sum_{k,m,p} h_0(k)h_1(m)h_2(p) \int_{-\infty}^{\infty} \phi_0(T)\phi_1(T-(2k_1+m-k))\phi_2(T-(2k_2+p-k))dT. \quad (11.18)$$

That inner integral is $b(2k_1 + m - k, 2k_2 + p - k)$. Set $n_1 = k - m$ and $n_2 = k - p$:

$$b(k_1, k_2) = 2^\alpha \sum_{n_1} \sum_{n_2} \sum_k h_0(k)h_1(k-n_1)h_2(k-n_2)b(2k_1-n_1, 2k_2-n_2). \quad (11.19)$$

The coefficients $g(n_1, n_2)$ anticipated in (11.17) have appeared (take the sum on k). The eigenvalue equation for b is

$$b(k_1, k_2) = 2^\alpha \sum_{n_1} \sum_{n_2} g(n_1, n_2)b(2k_1-n_1, 2k_2-n_2). \quad (11.20)$$

In general $\alpha = \ell + \mu_1 + \dots + \mu_\ell$. The matrix $(\downarrow 2)G$ must have the eigenvalue $2^{-\alpha}$. In case of multiple eigenvectors, there is a correct choice to be made for b . It is given by moment conditions (11.22) discovered in [DaMi]:

Theorem 11.1 Suppose the filters H_0, H_1, H_2 have at least 1, $1+\mu_1$, $1+\mu_2$ zeros at π . There is a unique finite length eigenvector b satisfying (11.21) and (11.22):

$$\sum g(2k-n)b(n) = 2^{-\alpha}b(k) \text{ which is } (\downarrow 2)Gb = 2^{-\alpha}b \quad (11.21)$$

$$\sum k^{v_1} k_2^{v_2} b(k_1, k_2) = \mu_1! \mu_2! \delta(v_1 - \mu_1) \delta(v_2 - \mu_2) \text{ for } v_1 \leq \mu_1, v_2 \leq \mu_2. \quad (11.22)$$

Example 11.2. Compute $b(k) = \int \phi(t)\phi'(t-k)dt$ with $\ell = 1$ and $\mu_1 = 1$ (first derivative).

The eigenvalue problem will be $2Tb = b$. The familiar matrix $T = (\downarrow 2)2HH^T$ has an extra 2 because of the derivative ϕ' . We have $\ell = 1$ and $\mu_1 = 1$ (there is no μ_2). Then b is the eigenvector of T with $\lambda = \frac{1}{2}$. The hat function example has eigenvector $(-0.5, 0, 0.5)$ when normalized by (11.22):

$$Tb = \frac{1}{8} \begin{bmatrix} \cdots & 6 & 4 & 1 \\ & 1 & 4 & 6 & 4 & 1 \\ & & 1 & 4 & 6 & \cdots \end{bmatrix} \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix} = \frac{1}{2}b.$$

The integral of the hat function times its derivative (which is +1 then -1) is $b(0) = 0$. The inner products with shifts are -.5 and .5 in agreement with b .

Example 11.3. Compute $b(k) = \int \phi(t)\phi''(t-k)dt = -\int \phi'(t)\phi'(t-k)dt$.

We can take $\ell = 1$ and choose $\mu_1 = 2$, or we can take $\ell = 2$ and choose $\phi_0(t) \equiv 1$ and $(\mu_1, \mu_2) = (1, 1)$. In either case b is an eigenvector of T with eigenvalue $\lambda = \frac{1}{4}$. For the hat function matrix above, that eigenvector is $b = (-1, 2, -1)$. It is pleasant to go on to third derivatives (Problem 7). The eigenvalue for $\int \phi(t)\phi'''(t-k)dt = -\int \phi'(t)\phi''(t-k)dt$ is

$\lambda = \frac{1}{8}$. The matrix T has this eigenvalue. But we must use the 5×5 central submatrix instead of 3×3 .

Example 11.4. Integrate the Daubechies D_4 function $\phi(t)$ over each interval $[0, 1]$, $[1, 2]$, $[2, 3]$.

Solution. We integrate $\phi(t)$ times translates of a box function, which has dilation coefficients $\frac{1}{2}$ and $\frac{1}{2}$. The Daubechies coefficients a, b, c, d add to 1 and are convolved with $(\frac{1}{2}, \frac{1}{2})$ to find $\frac{1}{2}(a, a+b, b+c, c+d, d)$. After double shift, the eigenvector b for $\lambda = 1$ gives the integrals of $\phi(t)$ over the three intervals:

$$\begin{bmatrix} 0 & d & c+d \\ c+d & b+c & a+b \\ a+b & a & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \quad \text{yields} \quad \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \frac{1}{12} \begin{bmatrix} 5+3\sqrt{3} \\ 2 \\ 5-3\sqrt{3} \end{bmatrix}.$$

The theory extends directly from a single variable t to d variables $x = (x_1, \dots, x_d)$. Each dilation equation involves integer vectors $k = (k_1, \dots, k_d)$:

$$\phi(x) = 2^d \sum h(k) \phi(2x - k). \quad (11.23)$$

The convention is still $\sum h(k) = 1$. The ordinary derivatives $D^\mu \phi(t)$ become mixed partial derivatives $D^\mu \phi(x) = (\partial/\partial x_1)^{\mu_1} \cdots (\partial/\partial x_d)^{\mu_d} \phi(x)$ of order $|\mu| = \mu_1 + \cdots + \mu_d$. Substitute the dilation equations into the desired integral, which is

$$b(k_1, \dots, k_\ell) = \int \cdots \int \phi_0(x) D^{\mu_1} \phi_1(x - k_1) \cdots D^{\mu_\ell} \phi_\ell(x - k_\ell) dx. \quad (11.24)$$

Each integer vector k_1, \dots, k_ℓ has d components. When $2x$ in the dilation equation is replaced by x , the multiplying factor 2^α has $\alpha = \ell d + |\mu_1| + \cdots + |\mu_\ell|$. The two-scale equation for the integrals is

$$b(k_1, \dots, k_\ell) = 2^\alpha \sum g(n_1, \dots, n_\ell) b(2k_1 - n_1, \dots, 2k_\ell - n_\ell). \quad (11.25)$$

The coefficients g follow the model (11.16). Now k is (k_1, \dots, k_d) and each vector n_1, \dots, n_ℓ has d components:

$$g(n_1, \dots, n_\ell) = \sum_k h_0(k) h_1(k - n_1) \cdots h_\ell(k - n_\ell). \quad (11.26)$$

Integrals with Wavelets

Integrals involving wavelets come directly from the integrals $b(k)$ involving scaling functions. The user must provide the highpass coefficients for any wavelets $w_0(t), w_1(t), \dots, w_\ell(t)$ that are in the integrals (staying for now at unit scale $j = 0$). Make those highpass replacements in $g(n)$ to get $g_w(n)$, leaving the lowpass coefficients whenever $\phi_i(t)$ is left in the integral. Then the wavelet integrals are $b_w = 2^\alpha (\downarrow 2) g_w * b$.

To emphasize: Wavelets bring no new eigenvalue problem, just a trivial matrix multiplication (trivial if you manage all the indices). Start with a basic example to see the pattern. When $w(t)$ replaces $\phi(t)$, the matrix H_1 replaces H :

Theorem 11.2 The wavelet integrals $\mathbf{a}_w(k) = \int \phi(t)w(t+k) dt$ and $\mathbf{a}_{ww}(k) = \int w(t)w(t+k) dt$ are computed from \mathbf{a} by the equations

$$\mathbf{a}_w = \mathbf{T}_w \mathbf{a} = (\downarrow 2) 2\mathbf{H}_1 \mathbf{H}_1^T \mathbf{a} \quad \text{and} \quad \mathbf{a}_{ww} = \mathbf{T}_{ww} \mathbf{a} = (\downarrow 2) 2\mathbf{H}_1 \mathbf{H}_1^T \mathbf{a}. \quad (11.27)$$

Proof: The wavelet equation in vector form is $\mathbf{W}(t) = (\downarrow 2) 2\mathbf{H}_1 \Phi(2t)$, where

$$\Phi(t) = \begin{bmatrix} \cdot & \cdot & \cdot \\ \phi(t-1) & \phi(t) & \phi(t+1) \\ \cdot & \cdot & \cdot \end{bmatrix} \quad \text{and} \quad \mathbf{W}(t) = \begin{bmatrix} \cdot & \cdot & \cdot \\ w(t-1) & w(t) & w(t+1) \\ \cdot & \cdot & \cdot \end{bmatrix}. \quad (11.28)$$

This vector form leads directly to the inner product vectors:

$$\mathbf{a}_w = \int_{-\infty}^{\infty} \phi(t) \mathbf{W}(t) dt \quad \text{and} \quad \mathbf{a}_{ww} = \int_{-\infty}^{\infty} w(t) \mathbf{W}(t) dt.$$

To compute \mathbf{a}_w substitute the dilation and wavelet equations:

$$\mathbf{a}_w = \int_{-\infty}^{\infty} \left(\sum 2h(k)\phi(2t-k) \right) (\downarrow 2) 2\mathbf{H}_1 \Phi(2t) dt. \quad (11.29)$$

For the k th term in the sum, change variables to $u = 2t - k$ (so that $du = 2dt$). That k th term becomes

$$2h(k)(\downarrow 2)\mathbf{H}_1 \int_{-\infty}^{\infty} \phi(u) \Phi(u+k) du = 2h(k)(\downarrow 2)\mathbf{H}_1 S^{-k} \mathbf{a}. \quad (11.30)$$

The k -step shift gave $\Phi(u+k) = S^{-k}\Phi(u)$, and then $\int \phi(t)\Phi(t) dt$ is exactly \mathbf{a} . Now sum on k to complete the formula:

$$\mathbf{a}_w = \sum_k 2h(k)(\downarrow 2)\mathbf{H}_1 S^{-k} \mathbf{a} = (\downarrow 2) 2\mathbf{H}_1 \mathbf{H}_1^T \mathbf{a}. \quad (11.31)$$

Notice that $\sum h(k)S^{-k}$ is the *upper triangular Toeplitz matrix* \mathbf{H}^T . The same steps for \mathbf{a}_{ww} , wavelet times wavelet, produce the double-shifted Toeplitz matrix $\mathbf{T}_{ww} = (\downarrow 2) 2\mathbf{H}_1 \mathbf{H}_1^T$.

Example 11.5. For $\mathbf{h}(k) = \left(\frac{1}{4}, \frac{2}{4}, \frac{1}{4}\right)$ the piecewise linear hat function on $[0, 2]$ is $\phi(t)$. Its inner products $\mathbf{a}(k)$ come from \mathbf{T} :

$$\mathbf{T} = \frac{1}{8} \begin{bmatrix} 4 & 1 & 0 \\ 4 & 6 & 4 \\ 0 & 1 & 4 \end{bmatrix} \quad \text{has eigenvector } \mathbf{a} = \frac{1}{6} \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix}.$$

For the wavelets, we choose the highpass coefficients $\mathbf{h}_1(k) = \frac{1}{8}(-1, -2, 6, -2, -1)$. Then $H(z)\mathbf{H}_1(-z)$ is the 6th degree Daubechies halfband filter and the synthesis functions $\widehat{\phi}(t)$ are biorthogonal to the hats. From the wavelet equation, $w(t)$ is also piecewise linear and it is supported on $[0, 3]$. The inner products with wavelet use matrix entries from $H_1(z)H(z^{-1})$ and $H_1(z)H_1(z^{-1})$:

$$\mathbf{a}_w = \frac{1}{16} \begin{bmatrix} -2 & -1 & & & \\ -2 & 6 & -2 & -1 & \\ & -1 & -2 & 6 & -2 \\ & & -1 & -2 & \end{bmatrix} \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 1 & 2 & 1 & \\ & 1 & 2 & \\ & & 1 & \end{bmatrix} \begin{bmatrix} 1/6 \\ 4/6 \\ 1/6 \\ 1 \end{bmatrix} = \begin{bmatrix} -1/12 \\ 1/12 \\ 1/12 \\ -1/12 \end{bmatrix}$$

$$\mathbf{a}_{ww} = \frac{1}{32} \begin{bmatrix} 4 & 1 & & \\ -20 & -8 & 4 & \\ -20 & 46 & -20 & \\ 4 & -8 & -20 & \\ 1 & 4 & & \end{bmatrix} \begin{bmatrix} 1/6 \\ 4/6 \\ 1/6 \end{bmatrix} = \begin{bmatrix} 1/24 \\ -6/24 \\ 18/24 \\ -6/24 \\ 1/24 \end{bmatrix}.$$

Integrals with multiple scales are also possible. For $a_1(k) = \int \phi(t)\phi(2t-k) dt$ we get a different eigenvalue problem $a_1 = (\downarrow 2)2UH^T a_1$. The reason is that the dilation equation for $\phi(2t)$ involves $4t$. On the right side is $h(k)\phi(4t-k)$ which is $u(2k)\phi(4t-k)$, provided $u = (\uparrow 2)h$. The matrix U with diagonals $h(0), 0, h(1), 0, h(2), 0, \dots$ replaces H when the integral involves $\phi(2t)$.

Problem Set 11.6

1. Integrate the hat function on $[0, 2]$ times the box function on $[0, 1]$, using their dilation coefficients $(\frac{1}{4}, \frac{2}{4}, \frac{1}{4})$ and $(\frac{1}{2}, \frac{1}{2})$.
2. For the quadratic spline that comes from $h = \frac{1}{6}(1, 3, 3, 1)$, find the integrals over $[0, 1]$ and $[1, 2]$ and $[2, 3]$.

Compute the integrals in 3.–6. by creating the coefficients $g(n)$ from h_0, h_1, h_2 and solving the eigenvalue problem $(\downarrow 2)Gb = 2^{-n}b$. Normalize by (11.22) and check by direct integration.

$$3. \int_{-\infty}^{\infty} (\text{hat function})^3 dt \quad 5. \int_{-\infty}^{\infty} (D_4 \text{ function } \phi(t))^2 dt$$

$$4. \int_0^1 (\text{hat function})^2 dt \quad 6. \int_0^1 (D_4 \text{ function } \phi(t))^2 dt$$

7. Follow Example 3 to integrate $\phi(t)\phi''(t-k)$ for $\phi(t) = \text{hat function}$.
8. The bilinear hat function in two variables $x = (x_1, x_2)$ has what formula and what coefficients $h(n)$? Find its inner products with its translates.
9. The frequency response $\sum h(k) e^{-i\omega k}$ gives the moments $H(0) = m_0 = 1$ and $H'(0) = -im_1$, and $H''(0) = -m_2$. Prove that an orthonormal filter with $H(\pi) = H'(\pi) = 0$ has $m_2 = m_1^2$, by setting $\omega = 0$ in the second derivative of $|H(\omega)|^2 + |H(\omega + \pi)|^2 = 1$.
10. Put $m_2 = m_1^2$ in equation (11.13) to show that $M_2 = M_1^2$ for orthonormal scaling functions with accuracy $p > 1$. Then the one-point rule $\int f(t)\phi(t)dt \approx f(M_1)$ has second-order accuracy.

Appendix 1

Wavelets

American Scientist 82 (April 1994) 250-255

Gilbert Strang

I was listening to Mozart, but my mind was wandering. Normally one doesn't admit such a thing. Possibly this was not Mozart at his best. Symphony Hall was full, we were late, and parking had been very illegal. But the real reason for inattention was an idle thought, and it led me toward this article. The thought was really a question: *How did he write it all down?*

For Mozart that must have been a serious problem. Musical notation is fairly efficient for its purpose, but a full score for all instruments equals a small book. The *notation* is crucial. To a musician it may seem the only possible notation—but it's not. When that signal is recorded (say digitally), the same symphony is expressed in a totally different way. The score that Mozart wrote is not transmitted.

The complete signal is not only the sequence of notes, but the nuances that the performer adds. The touch on the keys or the pedal or the bow—these distinguish a master. There may be an extra word saying *forte* or *pianissimo*. This is not too precise! Mathematically, the notes are an indication but not a complete code.

Our world is full of sounds and images and bit strings that have to be compressed and transmitted and recovered:

- Audio—music and speech
- Video—still images and television
- Data—numbers, letters, books, even zip codes.

Zip codes seem trivial, but this short signal illustrates the choices to be made. I think that England and Canada overdid the compression. Strings like M5S 1A4 and V6T 1W5 give extra precision, but they are too hard to list and remember. (Does any reader know how those countries chose alphanumeric?) Compression is delicate, because we have to *process* the signal, not just code it. It is the same for a student taking notes in class: compress the lecture but stay readable.

Television has been avoiding compression, but that is about to change. Each “pixel” on the TV screen needs 24 zeros or ones to mix three colors. The shading levels go from 0 to 255, making 2^8 possibilities—eight bits for each color. A very sharp image needs more pixels with more bits than we can afford to send or receive. Compression is required; information has to be thrown away. Otherwise the TV can't keep up with real time!

The problem is really severe for *high definition television*. It has an enormous bit rate. We will soon see an amazingly clear picture with many more pixels, provided the compression is well done. The processing of HDTV signals is a billion-dollar decision that the biggest companies are competing for. At the end I will mention a personal experience with this mathematical and engineering and increasingly political competition.

Fourier Transform and Wavelet Transform

This article is about several methods — none perfect, all being improved — to analyze and synthesize a signal. I mention three ways to transform a symphony:

1. Into cosine waves (by Fourier transform)
2. Into pieces of cosines (short time Fourier transform)
3. Into wavelets (the new way).

The classical method is to separate the whole symphony into pure harmonics. Each instrument plays one steady note. The signal becomes a sum $a_0 + a_1 \cos t + \dots$ of cosine waves. This is the *Fourier transform*, published 172 years ago by Joseph Fourier in Paris. He assumed an infinite orchestra: Conductor not needed, musicians totally bored.

All music and all signals can be analyzed into pure harmonic waves by the Fourier transform. Engineers do that almost instinctively — the signal strength at each moment in time is replaced by the amplitude of each wave. The big question is how many frequencies are needed for a high-fidelity signal. Probably too many for good compression.

A second possibility is the *Short Time Fourier Transform*. Short segments of the symphony are transformed separately. In each segment, the signal is separated into cosine waves as before. The musicians still play one note each, but they change amplitude in each segment. This is the way most long signals are carved up.

One disadvantage: There are sudden breaks between segments. You might or might not hear that. In a visual image you could probably see it as an edge. This “blocking effect” is the nemesis of the Short Time Fourier Transform.

A new idea in signal processing is closer to the score that Mozart originally wrote. Instead of cosine waves that go on forever or get chopped off, the new building blocks are “*wavelets*”. These are little waves that start and stop. They all come from one basic wavelet $w(t)$, which gives the sound level of the standard tune at time t . The basses spread that tune over the longest time. The cellos play the same tune but in half the time, at doubled frequency. Mathematically, this speed-up replaces the time variable t by $2t$. The first bass plays $b_1 w(t)$ and the first cello plays $c_1 w(2t)$, both starting at time $t = 0$.

The next bass and cello play $b_2 w(t - 1)$ and $c_2 w(2t - 1)$, with amplitudes b_2 and c_2 . The bass starts when $t - 1 = 0$, at time $t = 1$. The cello starts earlier, when $2t - 1 = 0$ and $t = \frac{1}{2}$. There are twice as many cellos as basses, to fill the total length of the symphony. Violas and violins play the same passage but faster and faster and all overlapping. At every level the frequency is doubled (up one octave) and there is new richness of detail. “Hyperviolins” are playing at 16 and 32 and 64 times the bass frequency, probably with very small amplitudes. Somehow these wavelets add up to a complete symphony. This article will attempt to show how.

Summary: A long signal is broken into a *basis of possible signals*. Those can be notes or waves or wavelets. The wavelets come from a single function $w(t)$ by speed-ups and time delays. Amplitudes are sent to the receiver, which reconstructs the symphony. (Very small amplitudes are discarded; more later about this crucial compression step.) We now explain the idea of a basis.

Choosing the Best Basis

Some readers will know about vectors and matrices. This section is for you. You remember that every vector (x, y) is a combination of the basis vectors $(1, 0)$ and $(0, 1)$. The vector $(4, 2)$ is 4 times the first basis vector plus 2 times the second.

Another pair of basis vectors is $(1, 1)$ and $(1, -1)$. Again they produce all plane vectors, including $(4, 2)$. The requirement met by both bases is this: *Each vector in the space can be expressed in one way only as a combination of the basis vectors.* The vectors $(1, 0)$ and $(2, 0)$ would not be a basis. They lie on the same line; no combination can give $(4, 2)$. The vectors $(1, 0)$ and $(0, 1)$ and $(3, 1)$ are not a basis—three vectors is too many. The signal $(4, 2)$ can be produced from any two of them (and it is also the sum of all three). Any two vectors in different directions, like $(5, 1)$ and $(3, 1)$, form a basis for the whole plane.

The best bases have a valuable extra property: The vectors are perpendicular. For the standard basis $(1, 0)$ and $(0, 1)$, across is perpendicular to up. The vectors $(1, 1)$ and $(1, -1)$ also pass the test for a 90° angle: *their dot product is zero.* The dot product of (x, y) with (X, Y) is $xX + yY$. The dot product of $(1, 1)$ with $(1, -1)$ is $1 - 1 = 0$. The dot product of $(5, 1)$ and $(3, 1)$ is 16—not perpendicular. More to the point, a vector with 1000 components separates quickly into 1000 perpendicular pieces—speed is crucial. These steps are at the heart of linear algebra.

For engineers and social and physical scientists, *linear algebra* now fills a place that is often more important than calculus. My generation of students, and certainly my teachers, did not see this change coming. It is partly the move from analog to digital; functions are replaced by vectors. Linear algebra combines the insight of n -dimensional space with the applications of matrices.

Higher Dimensions

Move now to four dimensions. Think of the components (x_1, x_2, x_3, x_4) as strengths of a signal. If the signal is $x = (1, 1, 1, 1)$ you are producing a steady note. With $x = (1, 2, 4, 8)$ the volume is increasing.

The easy way to send the signal is to give its four components. Implicitly, you have chosen the standard basis in four-dimensional space. The basis vectors are $(1, 0, 0, 0)$ and $(0, 1, 0, 0)$ and $(0, 0, 1, 0)$ and $(0, 0, 0, 1)$. You are sending the numbers x_1, x_2, x_3, x_4 that multiply your basis vectors. How could this be less than optimal? The answer depends on the signal.

Suppose you hold the same note. This vector $(1, 1, 1, 1)$ is very possible and very common, especially if the time interval is short. Good sound reproduction will certainly use short intervals. With the standard basis, you have to send 1 and 1 and 1 and 1. It would be better if the useful vector $(1, 1, 1, 1)$ was actually in the basis, so one signal would do. *A good basis allows accurate reproduction with only a few vectors.*

Compression discards basis vectors that are absent (or barely present) in the signal. When the vectors represent different frequencies—the *Fourier basis*—high frequencies can often be discarded. A “lowpass filter” removes them. With the standard basis, we can discard $(0, 1, 0, 0)$ only when the second note is not played. Otherwise there is a very audible (or inaudible) gap in the broadcast.

Musical notation has met this problem. It represents $(1, 1, 1, 1)$ by a full note and also by two half-notes. The same signal has several forms. This isn’t a basis! Full notes and half-notes and quarter-notes are fine for the musician, but the mathematician only wants four vectors in the basis.

It is quite pleasant to devise a basis that includes the constant vector $(1, 1, 1, 1)$. The second basis vector can be $(1, 1, -1, -1)$. This is a square wavelet, once up and once down. The German mathematician Alfred Haar completed a perpendicular basis in 1910, by *dilation* (or squeezing) and *translation* (or shifting). He squeezed $(1, 1, -1, -1)$ to produce the third vector $(1, -1, 0, 0)$. Then he shifted by two time intervals to produce the fourth vector $(0, 0, 1, -1)$. The basis vectors go into

the columns of the four-by-four “Haar matrix”:

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix}.$$

Can you guess the matrix H_8 ? Its first column is the constant vector $(1, 1, 1, 1, 1, 1, 1, 1)$. Electrical engineers would call this the zeroth column; they start from zero, like English lifts. The next column is $(1, 1, 1, 1, -1, -1, -1, -1)$, the wavelet that is played by the bass. It is squeezed to $(1, 1, -1, -1, 0, 0, 0, 0)$ for the first cello, and shifted for the second cello. The last four columns of H_8 are viola notes like $(1, -1, 0, 0, 0, 0, 0, 0)$, with higher frequency over a shorter time. All these vectors are perpendicular!

The violins enter when the dimension reaches 16. I don’t recommend writing down the matrix H_{16} , but you would know how. This is typical of linear algebra: When four dimensions are understood, higher dimensions are no problem. There are four violas and eight violins. The count $1 + 2 + 4 + 8$ equals 15, one short. The constant vector $(1, 1, \dots, 1, 1)$ makes the 16th or zeroth basis vector, which is the scaling function.

Fast Wavelet Transform

The very smallest wavelet transform we saw already. $(4, 2)$ is 3 times $(1, 1)$ plus 1 times $(1, -1)$. From the amplitudes 3 and 1, we recover the vector $(4, 2)$. To find those amplitudes for longer signals, we use *averages* and *differences*. The averages move up in a pyramid, to be averaged again. The differences give the fine details at each level. When a signal changes quickly, those details are important. When the signal is almost steady, we might not transmit them. That is how compression works, by ignoring what the eye can’t see and the ear can’t hear.

The test case is in four dimensions, for a vector like $(4, 2, 5, 5)$. We know how 4,2 in the first half yields 3 and 1. The second half 5,5 yields 5 (the average) and 0 (the difference). The two averages 3 and 5 move up the pyramid. The two differences 1 and 0 are the details that go with the basis vectors $(1, -1, 0, 0)$ and $(0, 0, 1, -1)$.

The averages give a coarse signal $(3, 5)$. Treat that the usual way. Compute $\frac{1}{2}(3 + 5) = 4$ and the difference $\frac{1}{2}(3 - 5) = -1$. Consistency is the key to a good algorithm! The wavelet transform of $(4, 2, 5, 5)$ is the set of averages and differences $(4, -1, 1, 0)$. This is transmitted to the receiver, which does an inverse transform back to $(4, 2, 5, 5)$. The decoder goes down the pyramid, first using the coarse parts 4 and -1 to recover 3 and 5. Those are combined with the fine details 1 and 0 to produce 4,2 and 5,5. We have reconstructed the signal.

This gives hope for a signal with 256 numbers. The pyramid has 8 levels because $256 = 2^8$. the lowest level starts as always with $A = \frac{1}{2}(x_1 + x_2)$ and $D = \frac{1}{2}(x_1 - x_2)$. The average A moves up to the next level. The difference D is the detail at this finest level.

The key word behind this pyramid is **multiresolution**. We are seeing the signal at multiple scales—fine, medium, and coarse. We resolve it at each scale by details and averages. Possibly our eyes do the same, to see the target and then the bull’s-eye. Certainly our ears are designed to pick off higher frequencies as the sound travels into the cochlea. The art of the engineer is imitating nature.

Note: We called this wavelet transform “fast”. A vector of N numbers is to be expressed as a combination of N basis vectors. In matrix language, we are multiplying by the Haar matrix H

or its inverse. Normally such a multiplication takes N^2 steps. The pyramid does it with only N averages and N differences.

Question: Is there also a fast Fourier transform? Can we quickly find the amplitudes of N cosine waves? Instead of H we use the Fourier matrix F , to transform between “time domain” and “frequency domain”. This fast transform is the most important numerical algorithm in our lifetime, a very tough competitor for wavelets. It is wired into computers and explained in linear algebra textbooks. The key is to produce F out of matrices whose entries are almost all zeros. The heart of numerical linear algebra has become matrix factorizations— H and F are outstanding examples.

The Daubechies Wavelets

Wavelets are a hot topic, but the Haar wavelets are cold. They were invented in 1910. Their graphs are made from flat pieces; anything that simple has already been thought of. Approximation to most signals is very poor. We need many many flat pieces to represent even a sloping line to decent accuracy. The basis will not give compression ratios of 20:1 or 100:1, as desired. So we turn to a better basis.

The new wavelets are more intricate, and their only formula is an infinite product, but eventually mathematics had to find them. I will touch on this advance by describing the discovery made by Ingrid Daubechies. In 1988, at AT&T Bell Laboratories, she found a pulse that starts and stops and is perpendicular to all its dilations and shifts. It is based on four “magic” numbers h_0, h_1, h_2, h_3 . Her scaling vector S uses them in that order. Her wavelet uses them in the order $W = (h_3, -h_2, h_1, -h_0)$. Do you see why W is perpendicular to S ? Multiply and add: the dot product $S \cdot W$ cancels itself to give zero. She also wanted $(1, 1, 1, 1)$ and $(1, 2, 3, 4)$ to have zero component along W , so constant and linear signals can be greatly compressed. Their dot products with W must be zero:

$$h_3 - h_2 + h_1 - h_0 = 0 \quad \text{and} \quad h_3 - 2h_2 + 3h_1 - 4h_0 = 0.$$

Those are two equations for the h 's; we need two more. The third equation makes the first bass tune $(h_3, -h_2, h_1, -h_0, 0, 0)$ perpendicular to the second bass tune $(0, 0, h_3, -h_2, h_1, -h_0)$. Their dot product is required to be $h_1h_3 + h_0h_2 = 0$. Then a fourth equation $h_0 + h_1 + h_2 + h_3 = 2$ sets the size of the h 's. Her numbers from solving the four equations give a much better filter than Haar: $4h_0 = 1 + \sqrt{3}$, $4h_1 = 3 + \sqrt{3}$, $4h_2 = 3 - \sqrt{3}$, $4h_3 = 1 - \sqrt{3}$. I must mention that this is not the ultimate. Six numbers or eight numbers can be even better, but also more work. Video filters tend to be short. Audio filters are amazingly long, because music is generally smoother than a picture.

The key step from discrete vectors to continuous functions is the *dilation equation*. This uses the magic numbers h_0, h_1, h_2, h_3 in a curious but natural way. The equation for the scaling function $\phi(t)$ involves t and $2t$ and the magic h 's:

$$\phi(t) = h_0\phi(2t) + h_1\phi(2t - 1) + h_2\phi(2t - 2) + h_3\phi(2t - 3).$$

Substitute $t = 1$ and $t = 2$ to find $\phi(1)$ and $\phi(2)$. Then the equation gives ϕ at $t = \frac{1}{2}, \frac{3}{2}, \frac{5}{2}$ because $2t$ on the right side is a whole number. From these half-integer times we go to quarter-integers. Eventually we have enough values to draw the graph from $t = 0$ to $t = 3$. This has become famous (to some people) as a function with entirely new properties, all built into the dilation equation.

Where Fourier used a cosine wave and Haar used a square wave, Daubechies begins with this scaling function. Her wavelet $w(t)$ has the same right side as the equation above, but with coefficients $h_3, -h_2, h_1, -h_0$. It has a highly irregular graph, a type of fractal. Squeezing and shifting it gives the complete wavelet basis. But all computations go back to the four numbers.

High Definition Television

“The shot heard round the world” was a metaphor in 1775. The signal seen around the world is now a reality. It is television. You may think that the signal could be improved (by better programs). Engineers also want to improve it (by more pixels and better compression). Probably this second improvement is the one we will get, with HDTV.

The battle to set the standard for high definition television has been played out in a surprisingly public way. Europe invested a billion dollars, then stopped. It was decided to follow the North American standard. A competition was organized by the Federal Communications Commission, and there were four finalists. One big issue is *motion estimation*, to predict where the picture will move. Then you only have to transmit a small correction, and you can keep up with all the pixels in real time.

The New York Times expected that the Japanese entry would win, because HDTV is already available in Japan. But it is based on analog signals, not 0’s and 1’s. Digital filters are now well developed, to separate the high frequencies and compress that information. But sharp edges have to stay sharp. Experts know exactly what to look for, like tasting wine or tea.

To the great credit of the New York Times, the competition was explained to the public. Every month it took a new turn. The FCC finished its tests, but one group of companies wanted a makeup test. They had stayed up the night before and their system wasn’t feeling too well. (All professors recognize these symptoms, but the government is softer. They allowed a retest.) We were expecting one winner, but the FCC urged cooperation in the end. We all hope it works.

I must explain how I learned about this. As usual, it was from a student. He came into my office and said “Do you remember me?” Well, I tried to think fast. Too many students, too weak a memory, and better to be absolutely honest. “Not completely,” I replied — never expecting to hear what he told me next. “You are my thesis advisor.”

What do you say to a thesis student you don’t remember? In that position I suggest something very short: “Tell me more.” The most amazing part was his thesis topic. “I am designing the filter bank for MIT’s entry in the HDTV competition.” Some days you can’t lose, even if you deserve to.

It is true that I am his advisor. He is a mathematics student who looked for a thesis in electrical engineering. He found Jae Lim, who masterminded MIT’s joint entry with General Instruments. A formal committee was required by the mathematics department. I must have agreed, since my signature was there on the paper. But my very own student would not tell me the coefficients h_0, \dots, h_6 in his filters. He expressed confidence in his advisor, but not enough for a billion-dollar secret.

Summary

A family of wavelets is an orthogonal basis. By combining the wavelets we represent any signal. Music has one independent variable: time. A photograph has two variables, across and up. Then wavelets in x multiply wavelets in y . A video image has three variables, x , y , and t — a flow of pictures with very important predictability. The competition between bases is fierce. Where computers race for faster calculations, mathematics races for quicker algorithms. An idea that cuts in half the *number* of steps is as good as a chip that doubles the speed. I don’t know if the idea of wavelets is in that league, but it might be.

The HDTV standard is based on Fourier transforms. Wavelets came too late to have a real chance in that video race, but they are highly competitive for audio. Four stereo channels need only a small fraction of a TV band. In a completely different competition, run by the FBI and worth describing here, wavelets have won.

The FBI has 30 million sets of fingerprints, more every day. They need to be digitized. When a driver is stopped, or a thief is apprehended, fingerprints will be recorded electronically. A central

computer will look for a match. As it is now, with thumb prints in enormous files in Washington, your chance to escape has been pretty good. Soon the “minutiae” of ridge endings and bifurcations will catch you.

It was expected that the older Fourier methods would succeed here too. But with 20:1 compression, the fingerprint ridges couldn't be followed. Lines were broken between one 8 by 8 square and the next. The short time Fourier transform introduced too much blocking. Wavelets gave a better picture. Tom Hopper at the FBI has chosen that new basis.

So when you get caught, you can blame it on wavelets.

Appendix 2

Wavelets and Dilation Equations: A Brief Introduction

Siam Review 31 (1989) 613-627

Gilbert Strang

Abstract

Wavelets are new families of basis functions that yield the representation $f(x) = \sum b_{ij} W(2^j x - k)$. Their construction begins with the solution $\phi(x)$ to a dilation equation with coefficients c_k . Then W comes from ϕ , and the basis comes by translation and dilation of W . It is shown in Part 1 how conditions on the c_k lead to approximation properties and orthogonality properties of the wavelets. Part 2 describes the recursive algorithms (also based on the c_k) that decompose and reconstruct f . The object of wavelets is to localize as far as possible in both time and frequency, with efficient algorithms.

Wavelets are based on translation ($W(x) \rightarrow W(x + 1)$) and above all on dilation ($W(x) \rightarrow W(2x)$). It is remarkable how long it has taken for “*dilation equations*” to be mentioned beside differential equations and difference equations. True, they are hardly in the same league. But ideas about wavelets are coming fast. The mathematics is attractive and several important applications seem to fit—I hope this survey will be helpful. You should know that its author is neither an expert nor an evangelist.

The goal is a new way to represent functions—especially functions that are local in time and frequency (or space and wave number). Compare with Fourier series. Sines and cosines are perfectly local in frequency, but global in x or t . A short pulse has slowly decaying coefficients that are hard to measure. To reconstruct the pulse, a Fourier series depends heavily on cancellation. The whole of Fourier analysis, relating properties of functions to properties of coefficients, is made difficult (some say interesting) by the nonlocal support of $\sin x$.

In achieving local support we lose the greatest property of the basis $\{e^{inx}\}$. With respect to a wavelet basis the differentiation operator is not diagonal. Wavelets are not eigenfunctions of $\partial/\partial x$, and frequencies are mixed up. The uncertainty principle imposes limits on what is possible in x and ξ together. The commutator $(\partial/\partial x)(\partial/\partial \xi) - (\partial/\partial \xi)(\partial/\partial x)$ is a multiple of the identity (since $(\partial/\partial x)(xu) - x(\partial u/\partial x) = u$), so we cannot diagonalize both operators. But a good “microlocalization” leaves $\partial/\partial x$ nearly diagonal, and at the same time nearly diagonalizes $\partial/\partial \xi$ (which is multiplication by x). To connect dilation with multiplication by x , differentiate $f(cx)$ with respect to c at $c = 1$.

The second important property of $\{e^{inx}\}$ is orthogonality. That can be saved. Wavelets can be made orthogonal to their own dilations (as well as their translations). Then $\int W(x)W(2^j x - k)dx = 0$ for all integers j and k . The wavelet basis has two indices, in which k is translation and j is dilation or compression. It suggests multigrid. A wavelet expansion $\sum b_{jk} W_{jk}(x)$ is a *multiresolution* of $f(x)$, in which b_{jk} carries information about f near $\xi = 2^j$ and $x = 2^{-j}k$. The sum on k is the detail at the scaling level $h = 2^{-j}$.

Orthogonality is not easy to achieve with local support. Truncated at zero and 2π , a sine wave $\phi(x)$ is orthogonal to $\phi(2x)$ but not to $\phi(4x)$. The “windowed Fourier transform” combines

smoothness with local support by bringing $e^{i\xi x}$ gradually to zero, but it is not fully satisfactory. The price of orthogonality with compact support is irregular basis functions. We live with these wavelets by doing all computations recursively (this subject is recursion heaven). And it is important to recognize that orthogonality and even linear independence (!) are not essential in the representation of functions. Wavelets need not be orthogonal.

This brief introduction cannot do justice to the applications. Nor can we attempt a proper history—it would be mostly in French. The idea of wavelets grew out of seismic analysis. Their development has been led by Yves Meyer, whose book will describe a new chapter in harmonic analysis (connecting to work of Calderòn, Grossmann, Morlet, Coifman, Weiss, and many others). The interest in wavelets is both pure and applied—like the interest in splines.

Part 1 of this paper establishes the properties of wavelets—approximation through Condition A and orthogonality through Condition O. Since we never see wavelets as functions (only recursively), their properties have to be discovered indirectly. We absolutely need these properties in order to have any idea what the algorithms are producing. Then Part 2 begins with a piecewise constant example (ϕ is a box function, the wavelet is Haar's). The example reveals a lot with no deep analysis. You could go directly to Part 2, about algorithms, and then return to dilation equations.

1. Dilation equations: Construction of ϕ . *The basic dilation equation is a two-scale difference equation:*

$$\phi(x) = \sum c_k \phi(2x - k). \quad (\text{A.1})$$

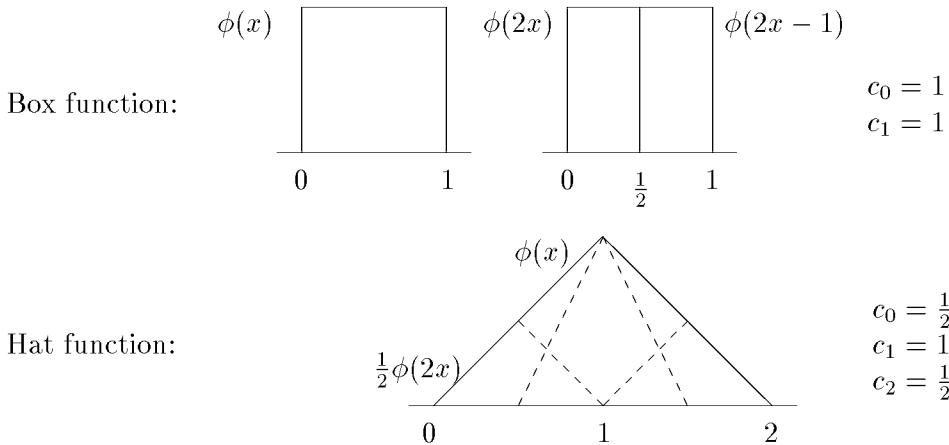
We look for a solution normalized by $\sum \phi dx = 1$. The first requirement on the coefficients c_k comes from multiplying by 2 and integrating:

$$2 \int \phi dx = \sum c_k \int \phi(2x - k) d(2x - k) \quad \text{yields} \quad \sum c_k = 2.$$

Uniqueness of ϕ is ensured by $\sum c_k = 2$. A smooth solution is not ensured. For a striking example, set $c_0 = 2$:

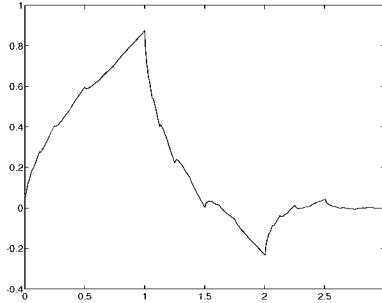
$$\text{The delta function } \phi = \delta \text{ satisfies } \delta(x) = 2\delta(2x).$$

That dilation of δ is unfamiliar (but somehow very pleasing). For other c 's, spline functions appear:



We now outline three constructions of the “scaling function” ϕ . Those constructions display very clearly the mathematics of dilation. Then we turn to wavelets, their properties and their purpose. A wavelet $W(x)$ is a second combination (involving the same recursion coefficients c_k) of the translates $\phi(2x - k)$.

Construction 1. Iterate $\phi_j(x) = \sum c_k \phi_{j-1}(2x - k)$ with the box function as $\phi_0(x)$. When $c_0 = 2$ the boxes get taller and thinner, approximating the delta function. For $c_0 = c_1 = 1$ the box is invariant: $\phi_j = \phi_0$. For $\frac{1}{2}, 1, \frac{1}{2}$ the hat function appears as $j \rightarrow \infty$, and $\frac{1}{8}, \frac{4}{8}, \frac{6}{8}, \frac{4}{8}, \frac{1}{8}$ yields the cubic B-spline. An example that will be important (an inspiration of Daubechies—we propose the notation D_4) has coefficients $\frac{1}{4}(1 + \sqrt{3}), \frac{1}{4}(3 + \sqrt{3}), \frac{1}{4}(3 - \sqrt{3}),$ and $\frac{1}{4}(1 - \sqrt{3})$:



This scaling function D_4 leads to orthogonal wavelets. *It is not as smooth as it looks.* Note that the Weierstrass nowhere differentiable function, which is $\sum b^n \cos(3^n x)$, involves dilation by 3. So does de Rham's function, which has $c_k = \frac{2}{3}, \frac{1}{3}, 1, \frac{1}{3}, \frac{2}{3}$ adding to 3. Resnikoff has found a connection between Weierstrass functions and wavelets.

Construction 2. The second construction takes the Fourier transform of (1):

$$\begin{aligned}\hat{\phi}(\xi) &= \sum c_k \int \phi(2x - k) e^{i\xi x} dx \\ &= \frac{1}{2} \left(\sum c_k e^{ik\xi/2} \right) \int \phi(y) e^{iy\xi/2} dy = P\left(\frac{\xi}{2}\right) \hat{\phi}\left(\frac{\xi}{2}\right).\end{aligned}\tag{A.2}$$

The symbol $P(\xi) = \frac{1}{2} \sum c_k e^{ik\xi}$ is the crucial function in this theory. Note that $P(0) = 1$. Now repeat (2) at $\xi/2, \xi/4, \dots$ and recall $\hat{\phi}(0) = \int \phi dx = 1$:

$$\hat{\phi}(\xi) = \left[\prod_1^n P\left(\frac{\xi}{2^j}\right) \right] \hat{\phi}\left(\frac{\xi}{2^N}\right) \quad \text{approaches} \quad \prod_1^\infty P\left(\frac{\xi}{2^j}\right).\tag{A.3}$$

For $c_0 = 2$ we find $P \equiv 1$ and $\hat{\phi} \equiv 1$, the transform of the delta function. For $c_0 = c_1 = 1$ the products of the P 's are geometric series:

$$P\left(\frac{\xi}{2}\right) P\left(\frac{\xi}{4}\right) = \frac{1}{4} (1 + e^{i\xi/2})(1 + e^{i\xi/4}) = \frac{1 - e^{i\xi}}{4(1 - e^{i\xi/4})}.$$

As $N \rightarrow \infty$ this approaches the infinite product $(1 - e^{i\xi}) / (-i\xi)$. This is $\int_0^1 e^{i\xi x} dx$, the transform of the box function. The hat function comes from squaring $P(\xi)$ which by (3) also squares $\hat{\phi}(\xi)$. (Multiplication of P 's is $\frac{1}{2}$ times convolution of c 's). The cubic B-spline comes from squaring again.

Construction 3. This construction of ϕ works directly with the recursion. Suppose ϕ is known at the integers $x = j$. The recursion (1) gives ϕ at the half-integers. Then it gives ϕ at the quarter-integers, and ultimately at all dyadic points $x = k/2^j$. This is fast to program. *All good wavelet calculations use recursion.*

The values of ϕ at the integers come from an eigenvector. With the four Daubechies coefficients, set $x = 1$ and $x = 2$ in the dilation equation (1) and use the fact that $\phi = 0$ unless $0 < x < 3$:

$$\begin{aligned}\phi(1) &= \frac{1}{4} (3 + \sqrt{3}) \phi(1) + \frac{1}{4} (1 + \sqrt{3}) \phi(2) \\ \phi(2) &= \frac{1}{4} (1 - \sqrt{3}) \phi(1) + \frac{1}{4} (3 - \sqrt{3}) \phi(2).\end{aligned}\tag{A.4}$$

This is $\phi = L\phi$, with matrix entries $L_{ij} = c_{2i-j}$. Compare with c_{i-j} for an ordinary difference equation. The eigenvalues are 1 and $\frac{1}{2}$. The eigenvector for $\lambda = 1$ has components $\phi(1) = \frac{1}{2}(1 + \sqrt{3})$ and $\phi(2) = \frac{1}{2}(1 - \sqrt{3})$, which are the heights on our graph of D_4 . The other eigenvalue $\lambda = \frac{1}{2}$ means that the recursion can be differentiated: $\phi'(x) = \sum c_k 2\phi'(2x - k)$ leads similarly to $\phi'(1)$ and $\phi'(2)$. In some weak sense, $\phi = D_4$ has a “dilational derivative.” For the hat function, the recursion matrix (see below) again has $\lambda = 1, \frac{1}{2}$. For the cubic spline the eigenvalues are 1, $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$.

To repeat for emphasis: From $\phi(1)$ and $\phi(2)$ the recursion gives everything.

In these constructions the properties of $P(\xi) = \frac{1}{2} \sum c_k e^{ik\xi}$ are decisive. The precise hypotheses are in flux, and infinitely many c_k can be allowed. One basic property will bring together the theory of dilation equations, before we go on to wavelets.

1.1. Dilation equations: Fundamental theorem. The accuracy of piecewise polynomial approximation, by splines or finite elements, depends on the answer to this question: To what degree $p - 1$ can the polynomials $1, x, x^2, \dots, x^{p-1}$ be reproduced exactly by the approximating functions? When the polynomials are “in the space,” the approximation error is of order h^p . In our case, the approximating functions are $\phi(x)$ and its translates. Splines are the best at approximation, and finite elements have the narrowest support—but both are weeded out when we require orthogonality. There is already a theory of approximation by translates. It connects p with the properties of $\hat{\phi}$. The link is the Poisson summation formula. When ϕ solves a dilation equation, that throws new questions into the theory—it is extremely satisfying that these new questions have the same answers.

For approximation with accuracy h^p , the Fourier transform $\hat{\phi}$ must have zeros of order p at all points $\xi = 2\pi n$ (except at $\xi = 0$ where $\hat{\phi} = 1$). Notice how easily that converts to a condition on the symbol P . According to (3), the transform $\hat{\phi}$ is the infinite product of $P(\xi/2^j)$. At $\xi = 2\pi$ the first factor is $P(\pi)$. At $\xi = 4\pi$ the second factor becomes $P(\pi)$. At $\xi = 6\pi$ the first factor is $P(3\pi)$, which by periodicity is the same as $P(\pi)$. The zeros of P produce zeros of $\hat{\phi}$:

Condition A. The symbol $P = \frac{1}{2} \sum c_k e^{ik\xi}$ has a zero of order p at $\xi = \pi$. Equivalently, the coefficients c_k satisfy the sum rules that yield $P^{(m)}(\pi) = 0$:

$$\sum (-1)^k k^m c_k = 0, \quad m = 0, 1, \dots, p-1.\tag{A.5}$$

The box function has $P = \frac{1}{2}(1 + e^{i\xi})$ and $p = 1$. The hat function has $p = 2$ and so does D_4 . The cubic spline has $p = 4$.

A zero at $\xi = \pi/2$ (instead of π) would also produce the desired zeros in the product $\hat{\phi}$. Thus Condition A is not strictly necessary in what follows. Choosing $c_0 = 1$ and $c_2 = 1$ and $P = \frac{1}{2}(1 + e^{2i\xi})$ stretches out the box function—it becomes $\phi = \frac{1}{2}$ on the double interval $0 < x \leq 2$. But $P(\pi/2) = 0$ produces instability and linear dependence—the alternating sum of stretched boxes is $\sum (-1)^k \phi(x - k) = 0$. With the added requirement of stability, the condition is exactly right.

The fundamental theorem states the consequences of Condition A:

1. The polynomials $1, x, \dots, x^{p-1}$ are linear combinations of the translates $\phi(x - k)$.
2. Smooth functions can be approximated with error $O(h^p)$ by combinations at every scale $h = 2^{-j}$:

$$\left\| f - \sum_k a_k \phi(2^j x - k) \right\| \leq C 2^{-jp} \|f^{(p)}\| \quad \text{for suitable } a_k.$$

3. The first p moments of the wavelet $W(x)$ (see below) are zero:

$$\int x^m W(x) dx = 0 \quad \text{for } m = 0, \dots, p-1.$$

4. The wavelet coefficients of a smooth function decay like $|\int f(x) W(2^j x) dx| \leq C 2^{-jp}$.

5. The recursion matrix M_N that determines ϕ at the integers has the eigenvalues $1, \frac{1}{2}, \dots, \left(\frac{1}{2}\right)^{p-1}$.

1 and **2** come from approximation theory. The combination of ϕ 's at scale j is also a combination $\sum b_{jk} W(2^j x - k)$ down to scale j . **3** and **4** are easy once wavelets are defined. Mallat gives a sharp result, with properly stated requirements on the smoothness and decay of ϕ : The H^p norm of f is equivalent to the corresponding norm of its coefficients b_{jk} . Wavelets lead to unconditional bases, suitable for a wide range of function spaces.

It is **5** that makes $\phi(x)$ smoother as p increases and also makes the constructions successful. The smoothness is weaker than $\phi \in C^{p-1}$, but it is striking that “dilational derivatives” come at the same time as higher degrees of approximation. What remains to be studied is orthogonality—which imposes an entirely different condition on the c_k .

Remark 1. Suppose the basic recursion has coefficients c_0, \dots, c_N . Then ϕ is zero outside the interval $[0, N]$. With continuity it follows that $\phi(0) = 0$ and $\phi(N) = 0$. Those were assumed in (4) when we determined $\phi = D_4$ at the integers. For the box function with $N = 1$, $\phi(0)$ and $\phi(N)$ cannot both be dropped. Our recursion matrix will be $(M_N)_{ij} = c_{2i-j}$ with $i, j = 0, \dots, N-1$. For the box function $M_1 = [1]$ has eigenvalue $\lambda = 1$, as expected in **5** above.

The spectrum of the infinite matrix M (allowing all i, j) is an attractive problem in operator theory. Notice that M is *convolution followed by decimation*—multiplication by the matrix c_{i-j} followed by projection onto even-numbered coordinates. By contrast with the usual Toeplitz case, eigenfunctions can have compact support! Homogeneous difference equations with zero boundary conditions lead to $\phi = 0$, but not so for dilation equations.

Remark 2. The minimum requirement is $p = 1$. Then $P(\pi) = 0$, which means that $\sum c_{2k} = \sum c_{2k+1}$. Since $\sum c_k = 2$, the columns of M add to 1:

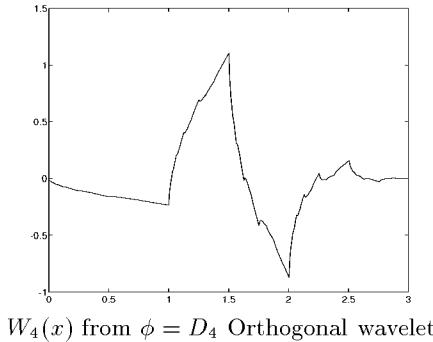
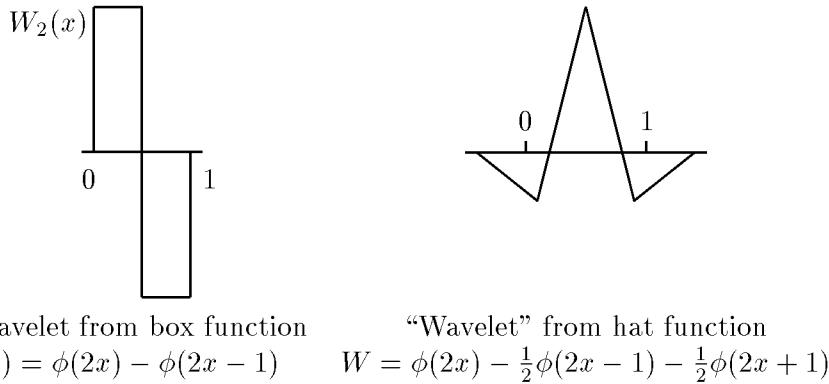
$$M_N = \begin{bmatrix} c_0 & & & \\ c_2 & c_1 & c_0 & \\ & c_3 & c_2 & c_1 \\ & & & c_3 \end{bmatrix} \quad \begin{array}{l} \text{steps of 2 down columns} \\ \text{steps of 1 across rows} \\ \text{here } N = 4 \end{array}$$

$(1, 1, 1, 1)$ is a left eigenvector with $\lambda = 1$. The right eigenvector yields the values $\phi(0), \dots, \phi(N-1)$ at the integers. The recursion determines ϕ at all dyadic points. Values at other points are never used.

1.2. Wavelets and orthogonality. Finally we define a wavelet. It comes from the scaling function ϕ by taking “differences”:

$$W(x) = \sum (-1)^k c_{1-k} \phi(2x - k). \tag{A.6}$$

We write W in place of the usual ψ , to distinguish more clearly from ϕ . Notice $2x$ on the right, and especially $(-1)^k$. Examples show the effect of alternating signs:



The wavelet from the hat function does not belong here. It is not orthogonal to $W(x + 1)$. The point is that the other two do belong. The Haar function is orthogonal to its own translations and dilations. Historically it was the original wavelet (but with $p = 1$ and poor approximation). The orthogonal wavelet W_4 has $p = 2$ and second-order approximation.

Without formulas for D_4 and W_4 , how is the orthogonality of their translates known? We need a test that applies to the recursion coefficients c_k , or to the symbol $P(\xi) = \frac{1}{2} \sum c_k e^{ik\xi}$.

Condition O.

$$|P(\xi)|^2 + |P(\xi + \pi)|^2 \equiv 1 \quad \text{or} \quad \sum c_k c_{k-2m} = 2\delta_{0m}.$$

With this condition, the infinite matrix L^*L in Part 2 is an orthogonal projection. To see now the role of Condition O, suppose the functions $\phi_0(2x - k)$ are orthogonal. Then so are the translates of $\phi_1(x) = \sum c_k \phi_0(2x - k)$:

$$\begin{aligned} \int \phi_1(x) \phi_1(x - m) dx &= \int \left(\sum c_k \phi_0(2x - k) \right) \left(\sum c_l \phi_0(2x - 2m - l) \right) dx \\ &= \sum c_k c_{k-2m} \int \phi_0^2(2x) dx = 0 \quad \text{for } m \neq 0. \end{aligned} \tag{A.7}$$

Construction 1 creates ϕ by iteration from the box function, which is orthogonal to its translates. Therefore (as Daubechies observed) so is ϕ .

The wavelet $W(x)$ in (6) is also orthogonal to $\phi(x - m)$. This is simple but neat, not involving Condition O. The sum in (7) changes to

$$\sum (-1)^k c_{1-k} c_{k-2m} \quad \text{which is identically zero!} \tag{A.8}$$

Just replace k by $1 - n + 2m$. This identity is $HL^* = 0$ in Part 2. Then (6) makes $W(x)$ orthogonal to $W(2x - m)$. The orthogonality of $W(x)$ and $W(x - m)$ comes back to Condition O.

The goal in constructing wavelets is to satisfy Conditions A and O. The basic family W_2, W_4, W_6, \dots was discovered by Daubechies, following Haar's W_2 . The accuracies are $p = 1, 2, 3, \dots$ and there are $2, 4, 6, \dots$ nonzero coefficients c_k . The smoothness also increases with p —but only by about $\frac{1}{2}$ derivative each time. D_4 and W_4 are Hölder continuous with exponent $.550\dots$. In Galerkin's method for solving differential equations, it is natural for these wavelets to be the trial functions—broader support than splines, nonsymmetric but orthogonal, multigrid built in, all computations based on recursion, difficulty to be expected at boundaries. The first experiments by Glowinski, Lawton, and Ravachol are particularly interesting for Burgers' equation.

2. Algorithms for wavelet expansions. Now comes a change of direction. Instead of discussing the properties of wavelets, we describe algorithms. The main question is how to *decompose* a signal into its wavelet coefficients, and how to *reconstruct* the signal from the coefficients. There is a “tree algorithm” or “pyramid algorithm” that makes these steps simple and fast. It does for the discrete wavelet transform what the Fast Fourier Transform (FFT) does for the discrete Fourier transform. The algorithm is fully recursive.

The user chooses a specific wavelet. We begin with the simplest choice, based on the box function. It satisfies the orthogonality property (Condition O), so all pieces of the decomposition are orthogonal. The approximation property (Condition A which preserves polynomials) determines how quickly the coefficients decay—for efficiency we want to stop the decomposition early. In that respect the box function is poor. Efficiency is the reason for working with higher wavelets W_4, W_6, W_8, \dots , and simplicity is the reason for starting with W_2 . This is Haar's wavelet [1 − 1].

The discussion will be discrete—for vectors not functions. We are given $n = 2^j$ values f_1, \dots, f_n . They may be equally spaced values of a function $f(x)$ on a unit interval. The goal is to split this vector f into its components at different scales, indexed by j . At each new level the meshwidth h is cut in half and the number of wavelet coefficients is doubled. The decomposition is

$$f = f^\phi + f^{(0)} + \dots + f^{(J-1)}.$$

The “detail” $f^{(j)}$ is a combination of 2^j wavelets at scale 2^{-j} , and f^ϕ is a multiple of the scaling function ϕ . For a numerical example take $J = 2$. Then the finest detail $f^{(1)}$ is the sum of two terms, here with coefficients $b_{11} = 4$ and $b_{12} = 1$:

$$f = \begin{bmatrix} 9 \\ 1 \\ 2 \\ 0 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} + 4 \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}. \quad (\text{A.9})$$

Notice that the four components are mutually orthogonal. There are $1 + 2 + \dots + 2^{J-1}$ wavelet coefficients, and the one from f^ϕ makes 2^J .

How are the coefficients 3, 2, 4, 1 computed from f ? *On the finest scale first.* As in the FFT, the decomposition begins with a “butterfly”:

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \\ & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 9 \\ 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 1 \\ 1 \end{bmatrix}. \quad (\text{A.10})$$

This is followed by a permutation, in which high frequencies go to the bottom:

$$\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 4 \\ 1 \end{bmatrix}. \quad (\text{A.11})$$

The next step is another butterfly, on low frequencies only:

$$\begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & & \\ \frac{1}{2} & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 1 \end{bmatrix}. \quad (\text{A.12})$$

The result is the set of wavelet coefficients 3, 2, 4, 1. The product of the three matrices in (10–12) is the decomposition matrix D . Its inverse is the reconstruction matrix R :

$$D = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \quad \text{has} \quad D^{-1} = R = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix}.$$

The coefficients 3, 2, 4, 1 enter the vector $b = (b_\phi, b_{01}, b_{11}, b_{12})$. The wavelet expansion in (9) is $f = R b$. The coefficients are $b = R^{-1}f = Df$. This product Df was computed recursively, from two butterfly matrices with a permutation between. In general there will be J matrices with permutations between.

The reconstruction is also recursive. It inverts (12) then (11) then (10). The global matrix R is the product of these local inverse matrices.

Notice that the operation count is proportional to n . It is best possible (the FFT count is $n \log_2 n$). There are only $n - 1$ individual 2-by-2 matrix multiplications, since high frequency coefficients (here 4 and 1) are settled and not reused. The Walsh functions give a different piecewise constant representation, in which the last two basis vectors are $(1, -1, 1, -1)$ and $(1, -1, -1, 1)$. In that case 4 and 1 enter another butterfly to produce the Walsh coefficients $\frac{5}{2}$ and $\frac{3}{2}$. The Walsh basis is global. The wavelet basis is local, but scaled—its support has width $O(2^{-J})$ at the finest scale and $O(1)$ at the coarsest scale.

Notice also the normalizing factors $\frac{1}{2}$ in decomposition (and 1's in reconstruction). The alternative is to introduce $1/\sqrt{2}$ for both. This has the advantage of normalizing the wavelets $W_{jk} = 2^{j/2}W(2^j x - k)$ at every scale. The whole basis is orthonormal (when $\|W\| = 1$). In the discrete case R and D become orthogonal matrices:

$$\hat{D} = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 & -1/2 \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \quad \text{has } \hat{R} = \hat{D}^{-1} = \text{transpose of } \hat{D}.$$

Based on the Haar example, we now start on Mallat's beautiful *tree algorithm* for wavelets. The simple average from $\left[\frac{1}{2} \frac{1}{2}\right]$ is replaced by a discrete filter based on ϕ . The difference $\left[\frac{1}{2} - \frac{1}{2}\right]$ is replaced by a filter based on W . The filters use the same recursion coefficients c_k that led to ϕ and W in the first place.

Decomposition. The given n -vector f is on the finest scale $h = 2^{-J}$. The *fine-to-coarse filter* (the “restriction operator” in multigrid language, the lowpass filter in signal processing language) is L . It produces a vector with half as many entries:

$$(Lf)_i = \frac{1}{2} \sum c_{2i-j} f_j, \quad i = 1, \dots, \frac{n}{2}. \quad (\text{A.13})$$

In the Haar example with $c_0 = c_1 = 1$, the entries of Lf are $\frac{1}{2}(f_1 + f_2)$ and $\frac{1}{2}(f_3 + f_4)$. The recursion continues to coarser scales, and after J steps it reaches a single number—the coefficient b_ϕ in f^ϕ at the coarsest scale $h = 1$. Here $b_\phi = \frac{1}{4}(f_1 + f_2 + f_3 + f_4)$.

The dual to L is the *coarse-to-fine map* L^* (the “interpolation operator” in multigrid language). Notice the change of index and the disappearance of $\frac{1}{2}$:

$$(L^*g)_j = \sum c_{2i-j} g_i, \quad j = 1, \dots, n. \quad (\text{A.14})$$

In the Haar example L^*Lf has entries $\frac{1}{2}(f_1 + f_2), \frac{1}{2}(f_1 - f_2), \frac{1}{2}(f_3 + f_4), \frac{1}{2}(f_3 - f_4)$. It is the projection of f onto the subspace that is piecewise constant at scale $2h$. It gives a blurred picture, with details lost.

The decomposition picks out these details, orthogonal to the average. The projection onto the wavelet subspace is the high frequency component:

$$f^{(J-1)} = f - L^*Lf. \quad (\text{A.15})$$

This repeats at every stage. There is an “average” or “blurred picture” $a^{(j-1)} = La^{(j)}$, starting from $a^{(J)} = f$. The detail lost in that average is the component of f at that stage:

$$f^{(j-1)} = (I - L^*L)a^{(j)} = a^{(j)} - L^*a^{(j-1)}. \quad (\text{A.16})$$

This is a first statement of the decomposition algorithm. We will see how Condition O simplifies the formula.

Reconstruction. To produce f from its details $f^{(j)}$, run the recursion (16) in reverse:

$$a^{(j)} = f^{(j-1)} + L^*a^{(j-1)}. \quad (\text{A.17})$$

This starts from the coarsest detail $f^{(0)}$ and the totally blurred picture $a^{(0)} = f^\phi$. It returns to $f = a^{(J)}$.

Apply orthogonality. The most elegant part of the algorithm is still to come. It is not necessary to compute the detail vector $f^{(j)}$ from (16), and then to compute its wavelet coefficients b_{jk} . Those are the numbers we want (4 and 1 in the example at level $j = 1$). *These numbers can be found directly from $a^{(j)}$.*

Review the Haar example first. The lowpass filter gave $a^{(1)}$ from $f = a^{(2)}$:

$$Lf = \frac{1}{2} \begin{bmatrix} c_1 & c_0 & & \\ & c_1 & c_0 & \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & \\ & \frac{1}{2} & \frac{1}{2} & \end{bmatrix} \begin{bmatrix} 9 \\ 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \end{bmatrix}.$$

The blurred picture is $a^{(1)} = (5, 5, 1, 1)$. At the next level the low-pass filter leaves 3, the coefficient of $(1, 1, 1, 1)$. *We now want the orthogonal filter—the highpass filter H.* In the Haar example it

produces

$$Hf = \frac{1}{2} \begin{bmatrix} c_0 & -c_1 \\ & c_0 & -c_1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 9 \\ 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}.$$

Those coefficients 4 and 1 represent the detail $f^{(1)} = (4, -4, 1, -1)$, which is lost when $a^{(2)}$ is blurred to $a^{(1)}$. At the next level H is applied to $a^{(1)}$. That produces $\frac{1}{2}(5) - \frac{1}{2}(1) = 2$. This is the coefficient b_{01} , representing the detail $(2, 2, -2, -2)$ lost when $a^{(1)}$ is blurred to $a^{(0)}$. We now put these pieces together into *Mallat's pyramid algorithm*:

Decomposition. Initialize $a^J = f$. For $j = J, \dots, 1$ compute

$$a^{j-1} = La^j \quad \text{and} \quad b^{j-1} = Ha^j. \quad (\text{A.18})$$

Reconstruction. Start with a^0 and b^0, \dots, b^{J-1} . For $j = 1, \dots, J$ compute

$$a^j = L^*a^{j-1} + H^*b^{j-1}. \quad (\text{A.19})$$

The full decomposition is represented by a tree of filters:

$$\begin{array}{ccccccc} a^J & \xrightarrow{L} & a^{J-1} & \xrightarrow{L} & a^{J-2} & \cdots & \xrightarrow{L} & a^0 \\ H \searrow & & H \searrow & & H \searrow & & H \searrow & \\ b^{J-1} & & b^{J-2} & & & & & b^0. \end{array}$$

The reconstruction goes from the branches of the tree back to the root:

$$\begin{array}{ccccccc} a^0 & \xrightarrow{L^*} & a^1 & \xrightarrow{L^*} & a^2 & \cdots & \xrightarrow{L^*} & a^J = f \\ b^0 & \nearrow_{H^*} & b^1 & \nearrow_{H^*} & & & \nearrow_{H^*} & \end{array}$$

The next step is to identify these filter matrices L and H for examples other than ‘‘box and Haar.’’

Note. The filter matrices L and H have half as many rows as columns. By dropping the parentheses around j , we distinguish the vector a^j with only 2^j components from the vector $a^{(j)}$ with the full $w^J = n$ components. The vector a^j contains the expansion coefficients of $a^{(j)}$ with respect to the translates $\phi(2^j x - k)$. See the example above and the multiresolution below!

2.1. The filter matrices L and H . The matrix L is known from the first part of the paper. Its entries $L_{ij} = c_{2i-j}$ are the recursion coefficients for the scaling function. Rows 1, 2 and columns $-1, 0, 1, 2$ are displayed with $N = 3$:

$$L = \frac{1}{2} \begin{bmatrix} c_3 & c_2 & c_1 & c_0 \\ & c_3 & c_2 & c_1 & c_0 \end{bmatrix}.$$

The beautiful thing is that the highpass filter (strictly speaking it is band-pass) uses the same coefficients. H is associated with the wavelet W just as L is associated with the scaling function ϕ . Equation (6) for W uses the same c_k , but with alternating signs and reversed order. The wavelet filter has

$$H_{ij} = (-1)^{j+1}c_{j+1-2i}. \quad (\text{A.20})$$

Rows 1, 2 and columns 1, 2, 3, 4 are displayed:

$$H = \frac{1}{2} \begin{bmatrix} c_0 & -c_1 & c_2 & -c_3 \\ & c_0 & -c_1 & c_2 & -c_3 \end{bmatrix}.$$

The indices were chosen to match the Haar example (variants are possible). The transposed matrices, without the factor $\frac{1}{2}$, represent the dual filters L^* and H^* . The important points now come quickly, and matrix multiplication is the best proof.

Theorem 1. *By their construction the filters are orthogonal:*

$$HL^* = 0. \quad (\text{A.21})$$

This multiplication is the reason behind the construction of H —alternating signs, reversed order, index shifted by one. See equation (8).

We finally come to the reward for Condition O: $\sum c_k c_{k+2m} = 2\delta_{0m}$. The reason for that condition is in the reward. Remember that the box function and D_4 satisfied this requirement but not the hat function or the cubic spline. Condition O can be stated and understood in transform space, but I believe that the matrix interpretation is again the clearest.

Theorem 2. *If condition O holds then*

1. $LL^* = I$ and $HH^* = I$.
2. L^*L and H^*H are mutually orthogonal projections with

$$L^*L + H^*H = I. \quad (\text{A.23})$$

Remember that L and H map into subspaces half as large as the original. L^* and H^* map back. The identity operators in (22) are on the half-sized subspaces.

The proof of (22) is by direct matrix manipulation. Condition O gives the result. Then it follows that $L^*LL^*L = L^*L$, so L^*L is a projection—and similarly for H^*H . The property $HL^* = 0$ in (21) yields $H(L^*L + H^*H) = H$. The transpose $LH^* = 0$ yields $L(L^*L + H^*H) = L$. The operator in (23) is the identity on both orthogonal components—the ranges of L and H —so it is the identity. We have an orthogonal decomposition by “quadrature mirror filters” L and H at every step.

2.2. Multiresolution of L^2 . The last paragraphs changed quietly from functions to vectors. That was for the sake of algorithms, which use values of ϕ and W at dyadic points $k/2^J$. The Haar example began with f at equally spaced points on $(0, 1]$. But the filter matrices really apply to discrete values along the whole line—they are infinite matrices. More than that, the decomposition $f = \sum f^{(j)}$ is just as valuable for functions in L^2 as for vectors in l^2 .

This multiresolution yields the details of f at all scalings 2^{-j} . On the whole line we take $j = 0, \pm 1, \pm 2, \dots$. The decomposition develops an idea that was already present in approximation theory—to put frequencies together in “octaves.” (Besov spaces combine frequencies $2^j \leq \xi < 2^{j+1}$. It seems that the ear also receives frequencies on a logarithmic scale.) For functional analysis the starting point is the subspace S_j spanned by the translates $\phi(2^j x - k)$. If a function $g(x)$ is in S_j , then $g(2x)$ is in S_{j+1} . The dilation equation writes $\phi(x)$ as a combination of $\phi(2x - k)$, which assures that $S_0 \subset S_1$. At all scales we have

$$\cdots S_{-1} \subset S_0 \subset S_1 \subset S_2 \cdots \text{ with } \cup S_j \text{ dense in } L^2 \text{ and } \cap S_j = \{0\}.$$

Now turn to the *wavelet subspace* W_j . It is spanned by the translates $W(2^j x - k)$. It is invariant under translation by multiples of 2^{-j} . If $g(x)$ is in W_j then $g(2x)$ is in W_{j+1} . The construction

$W(x) = \sum (-1)^k x_{1-k} \phi(2x - k)$ puts W and its translates into S_1 , and makes them orthogonal to S_0 . In fact, W_0 and S_0 are orthogonal complements in S_1 . At every scale $W_j \oplus S_j = S_{j+1}$. The spaces S_j give the “partial sums” of the differences W_j :

$$\cdots \bigoplus W_{-1} \bigoplus W_0 \bigoplus \cdots \bigoplus W_j = S_{j+1} \quad \text{and} \quad \bigoplus_{-\infty}^{\infty} W_j = L^2.$$

The multiresolution of f is a splitting into components $f^{(j)} \in W_j$:

$$f = \sum_{-\infty}^{\infty} f^{(j)} \quad \text{or} \quad f = f^\phi + \sum_0^{\infty} f^{(j)}, \quad f^\phi \in S_0. \quad (\text{A.24})$$

This is a very satisfying decomposition of L^2 functions, classical but with new subspaces. The coefficients b^j in Mallat’s pyramid algorithm corresponded to $f^{(j)} \in W_j$, and a^j corresponded to $a^{(j)} \in S_j$.

The analogue of the discrete Fourier transform was in the algorithm. The analogue of ordinary Fourier series is (24). The analogue of the Fourier integral formula is the *integral wavelet transform*. Representations of different groups give rise to different transforms.

2.3. Applications. Image processing works with $F(x, y)$, so it is natural to look for *two-dimensional wavelets*. The simplest construction uses the products $\phi(x)\phi(y)$, $\phi(x)W(y)$, $W(x)\phi(y)$, $W(x)W(y)$. Orthogonality is clear. New constructions have been invented that are genuinely two-dimensional, but it is useful to start with the tensor products of “box and Haar.” The given two-dimensional array F yields a two-dimensional array B of wavelet coefficients.

For pattern recognition, a major difficulty with the wavelet transform B is the lack of translation invariance. If the pattern is shifted by a fraction of h , its wavelet model is changed. A higher sampling rate is possible but expensive. Mallat studies instead the *zero-crossings* of the wavelet transform, which locate the signal edges. Now the difficulty is to make the reconstruction stable. In edge detection the first wavelets were Laplacians of shifted Gaussians, introduced by Gabor. The orthogonal wavelets of Meyer are C^∞ with polynomial decay, the Battle-Lemarié wavelets based on splines are C^n with exponential decay, and the Daubechies wavelets are C^n (smaller n) with compact support.

In closing we recall the original problem—to localize in time and frequency. Geophysics needs to represent short high-frequency pulses. Physics needs to divide up phase space. The coherent states $g_{pq} = e^{ipx} g(x-q)$ give a “Weyl-Heisenberg” frame, with some redundancy—but still f can be reconstructed from $\int f(g_{pq}) g_{pq} dp dq$. Mathematics needs (or wants) an orthogonal decomposition, better than g_{pq} at high frequencies and with no redundancy. The answer for now is wavelets.

It is a pleasure to thank Ingrid Daubechies and Howard Resnikoff for introducing me to wavelets.