

Chapter 2

Wavelets

2.1 Introduction

Wavelets have attracted a lot of attention from people involved in time-frequency analysis of signals. The literature on wavelets, books, papers, is quite extensive. Many practical applications of wavelets have been found.

The purpose of this chapter is to introduce the fundamentals of wavelets. As in previous chapters, it is preferred to start with basic examples, and then to proceed with generalization and formalization of concepts.

Continuous periodic signals can be well represented as Fourier expansions. Other signals may be better represented with alternative expansions. Wavelets are functions that are localized both in time and in frequency, providing pieces for a new kind of expansions.

The main idea can be illustrated with the tiling of the time-frequency plane. When using the STFT, the tiling is as depicted in Fig. 2.1.

The tiles have an area $\Delta t \cdot \Delta f = A = \text{constant}$. Due to uncertainty, if you want to narrow Δt then you have to enlarge Δf , and vice-versa.

If you are analyzing a signal with significant low frequency components, then Δt must be relatively large (to accommodate at least one period of the low frequency component), and so you cannot study possible variations of high frequency components inside a Δt .

With wavelets the time-frequency tiling is as depicted in Fig. 2.2.

Tiles adapt to the signal components. Large Δt and narrow Δf for low-frequency components. Narrow Δt and wide Δf for high-frequency components. Therefore, a better study of the variations of high frequency components is allowed.

Now, let us introduce some formal concepts, which are obviously related with the tiling just described.

This chapter does not restrict to periodic functions but, instead, to the space L^2 of square-integrable functions. This refers to finite-energy signals.

A function $y(x)$ is said to have *support* in the interval $[x_1, x_2]$ if $y(x) = 0$ when x is outside the interval and $[x_1, x_2]$ is the smallest closed interval for which this

Fig. 2.1 Tiling of the TF plane corresponding to the STFT

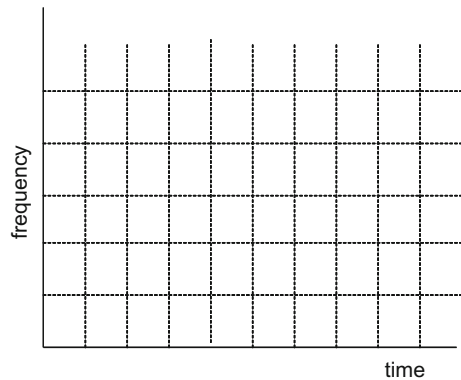
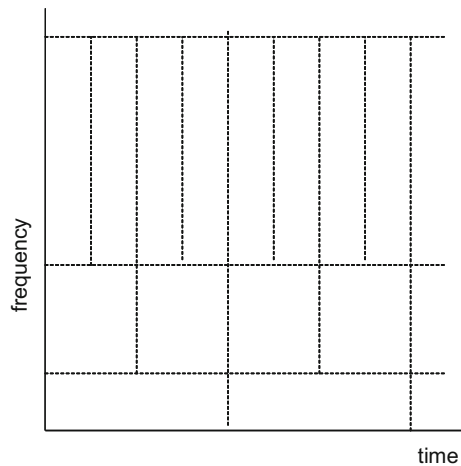


Fig. 2.2 Tiling of the TF plane corresponding to wavelets



holds. Compact support or finite support corresponds to finite intervals. This refers to finite-time signals, $y(t)$ in $[t_1, t_2]$, and to band-limited signals, $Y(\omega)$ in $[\omega_1, \omega_2]$.

Wavelets $\psi(t)$ are zero-mean real valued functions of time, which correspond in the frequency domain to band-pass behaviour:

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

It is convenient for many applications that $\psi(t)$ had several vanishing moments:

$$m_1(k) = \int_{-\infty}^{\infty} t^k \psi(t) dt = 0; k = 0, 1, \dots, m \quad (2.2)$$

As it will be described in this chapter, a signal can be decomposed into wavelets (analysis) and then this signal can be recovered from these wavelets (synthesis).

The wavelet analysis of signals can be obtained in real time with filter banks.

Filter banks provide a way of understanding wavelets. In particular, the analysis-synthesis of signals via wavelets is related with the perfect reconstruction (PR) using filter banks.

In the next section, the Haar wavelet will be introduced. It is directly related with the QMF filter bank, which is the only PR filter bank being FIR, orthogonal, and linear phase. The Haar wavelet is also unique in terms of compact support, orthogonality, and symmetry.

In addition to the Haar wavelet, the chapter will explore other orthogonal and biorthogonal wavelets, like it was done in the previous chapter with orthogonal and biorthogonal filter banks.

During the 1980s and 1990s, a lot of research work focused on the STFT. It was in this context that the wavelet analysis was introduced, in 1984, by Grossman and Morlet [49]. It soon awakened large interest, and then important contributions to a theory of wavelets came from Meyer [47], Daubechies [24], Mallat [45], and others. Initial efforts were oriented to orthogonal bases. During the 1990s the research departed from the limitations imposed by orthogonality, and many special versions and structures of wavelets appeared.

One of the main fields of applications is image processing. There are special versions of wavelets for this field. Next chapters will consider image processing, and will include pertinent wavelet variants.

There is a MATLAB toolbox for wavelets. This toolbox will be considered by the end of the chapter. For didactical reasons, it was preferred not to use this toolbox along the chapter, but instead to include our own MATLAB programs.

A lot of papers and books about wavelets have been published. This chapter is mostly based on [14, 25, 44, 48], with the complementary support of publications to be mentioned on proper places.

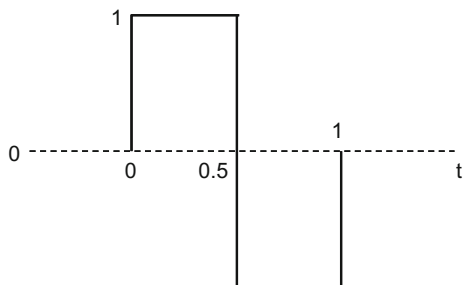
2.2 An Important Example: The Haar Wavelets

The Haar wavelets were introduced by Alfred Haar in 1909 (the term ‘*wavelet*’ was not coined till later on).

2.2.1 Definitions

The Haar *wavelet function* is:

$$\psi(t) = \begin{cases} 1, & 0 \leq t < 0.5 \\ -1, & 0.5 \leq t < 1 \\ 0, & elsewhere \end{cases} \quad (2.3)$$

Fig. 2.3 The Haar wavelet

The function $\psi(t)$ is a ‘mother wavelet’.

A plot of the Haar mother wavelet is shown in Fig. 2.3. Notice that the integral of $\psi(t)$ over its domain is 0.

‘Baby wavelets’ are obtained with scaling and shifting as follows:

$$\psi_{j,k}(t) = \sqrt{2^j} \psi(2^j t - k), \quad j = 0, 1, 2, \dots \text{ (scale)}; \quad k = 0, 1, 2, \dots, 2^j - 1 \text{ (shift)} \quad (2.4)$$

Figure 2.4 shows examples of scaling and shifting.

Every baby wavelet has:

$$\int_0^1 (\psi_{j,k}(t))^2 dt = 1 \quad (2.5)$$

It can be shown that the set $\psi_{j,k}(t)$ is an orthonormal basis for L^2 . A function $y(t)$ can be represented in terms of the Haar basis as follows:

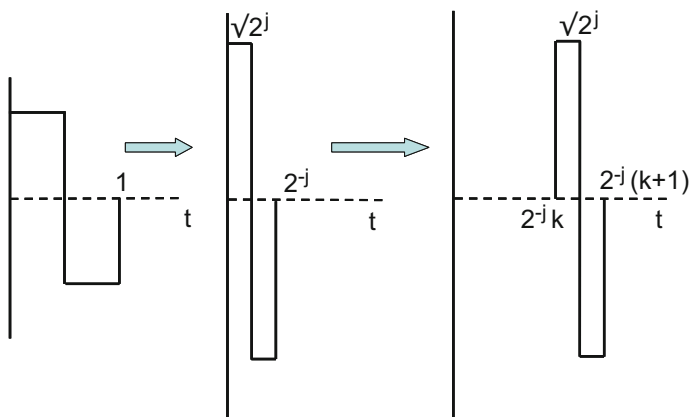
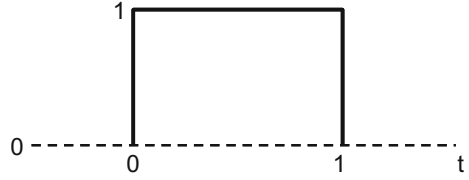
**Fig. 2.4** Baby wavelets: scaling and shifting

Fig. 2.5 The Haar scaling function



$$y(t) = \sum_{j,k} \langle y, \psi_{j,k} \rangle \cdot \psi_{j,k}(t) = \sum_{j,k} d_{j,k} \cdot \psi_{j,k}(t) \quad (2.6)$$

where $d_{j,k}$ are the Haar wavelet coefficients.

The Haar wavelet has a companion function, named *scaling function* $\varphi(t)$, which is defined by (Fig. 2.5).

$$\varphi(t) = \begin{cases} 1, & 0 \leq t < 1 \\ 0, & \text{elsewhere} \end{cases} \quad (2.7)$$

Also:

$$\varphi_{j,k}(t) = \sqrt{2^j} \varphi(2^j t - k), \quad j = 0, 1, 2, \dots; \quad k = 0, 1, 2, \dots, 2^j - 1 \quad (2.8)$$

and:

$$\int_0^1 (\varphi_{j,k}(t))^2 dt = 1 \quad (2.9)$$

The Haar wavelet and scaling function satisfy the identities:

$$\begin{aligned} \psi(t) &= \varphi(2t) - \varphi(2t - 1) \\ \varphi(t) &= \varphi(2t) + \varphi(2t - 1) \end{aligned} \quad (2.10)$$

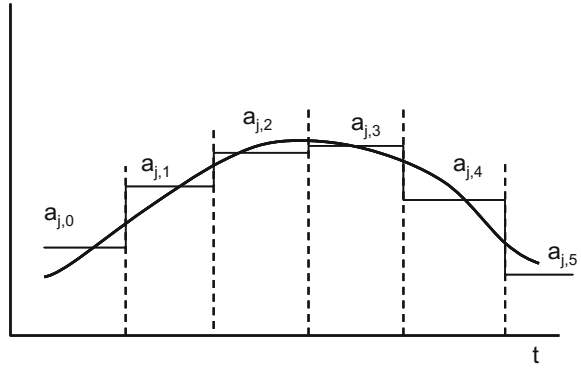
Suppose a signal $y(t)$, then:

$$a_{j,k} = \langle y, \varphi_{j,k} \rangle = \int y(t) \varphi_{j,k}(t) dt \quad (2.11)$$

is a set of ‘average samples’ of the signal (each $a_{j,k}$ is an average value of $y(t)$ over adjacent intervals). Figure 2.6 shows an example of average samples.

2.2.2 Multiresolution Analysis

There is a simple procedure to obtain function representations in terms of wavelets. The idea is to consider a set of subspaces corresponding to several resolution levels—that is multiresolution. Higher scale values, which corresponds to higher frequencies, allow for more time resolution (more values of k).

Fig. 2.6 Average samples

Let us advance step by step, with theory and examples.

Define the function space V_j as follows:

$$V_j = \text{span} \{ \varphi_{j,k} \}_{k=0,1,2,\dots,2^j-1} \quad (2.12)$$

where ‘span’ is the linear span.

It can be shown that (Fig. 2.7): $V_j \subseteq V_{j+1}$

The set:

$$\{ \varphi_{j,k}(t) \}_{k=0,1,2,\dots,2^j-1} \quad (2.13)$$

is an orthonormal basis for V_j .

But the set:

$$\{ \varphi_{j,k}(t) \}_{j=0,1,2,\dots; k=0,1,2,\dots,2^j-1} \quad (2.14)$$

is not an orthonormal basis for L^2 because the spaces V_j are not mutually orthogonal.

Define the wavelet subspace W_j as the orthogonal complement of V_j in V_{j+1} . That is (Fig. 2.8):

$$V_{j+1} = V_j \oplus W_j \quad (\oplus \text{ orthogonal sum}) \quad (2.15)$$

It can be shown that the set:

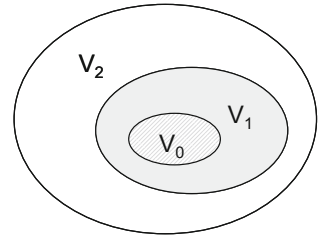
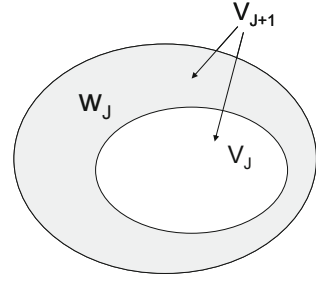
Fig. 2.7 Function spaces

Fig. 2.8 Complementary spaces V_j and W_j



$$\{\psi_{j,k}(t)\}_{k=0,1,2,\dots,2^j-1} \quad (2.16)$$

is an orthonormal basis for W_j .

In consequence it is possible to combine wavelets and scaling functions to generate V_{j+1} from V_j and W_j .

Notice that the decomposition could be iterated, so as:

$$V_{j+1} = V_0 \oplus W_0 \oplus W_1 \oplus W_2 \dots \oplus W_j \quad (2.17)$$

For example, suppose we have a signal y with 8 samples (2^3 samples). The number of samples gives the maximum resolution; thus $j+1 = 3$. Let us expand this signal y into $V_3 = V_0 \oplus W_0 \oplus W_1 \oplus W_2$:

$$\begin{aligned} y = & \langle y, \varphi_{0,0} \rangle \varphi_{0,0} + \langle y, \psi_{0,0} \rangle \psi_{0,0} + \langle y, \psi_{1,0} \rangle \psi_{1,0} + \\ & + \langle y, \psi_{1,1} \rangle \psi_{1,1} + \langle y, \psi_{2,0} \rangle \psi_{2,0} + \langle y, \psi_{2,1} \rangle \psi_{2,1} + \\ & + \langle y, \psi_{2,2} \rangle \psi_{2,2} + \langle y, \psi_{2,3} \rangle \psi_{2,3} \end{aligned} \quad (2.18)$$

Fortunately it is not necessary to evaluate each inner product in the previous equation using integrals. There is a useful alternative that will be described now with some theory continuation and a practical example.

To simplify the notation, the Eq. (2.18) is written as follows:

$$\begin{aligned} y = & a_{0,0} \varphi_{0,0} + d_{0,0} \psi_{0,0} + d_{1,0} \psi_{1,0} + d_{1,1} \psi_{1,1} + \\ & + d_{2,0} \psi_{2,0} + d_{2,1} \psi_{2,1} + d_{2,2} \psi_{2,2} + d_{2,3} \psi_{2,3} \end{aligned} \quad (2.19)$$

where $a_{j,k}$ correspond to average samples, and $d_{j,k}$ correspond to details or differences (this will be explained later on).

Substitute $2^j t - k$ in lieu of t in the identities (2.10). Also, multiply both sides of equations by $\sqrt{2^j}$:

$$\begin{aligned} \sqrt{2^j} \psi(2^j t - k) &= \frac{1}{\sqrt{2}} [\sqrt{2^{j+1}} \varphi(2^{j+1} t - 2k) - \sqrt{2^{j+1}} \varphi(2^{j+1} t - 2k - 1)] \\ \sqrt{2^j} \varphi(2^j t - k) &= \frac{1}{\sqrt{2}} [\sqrt{2^{j+1}} \varphi(2^{j+1} t - 2k) + \sqrt{2^{j+1}} \varphi(2^{j+1} t - 2k - 1)] \end{aligned} \quad (2.20)$$

Using (2.4) and (2.8) the above equation can be expressed in a simpler form:

$$\begin{aligned}\psi_{j,k}(t) &= \frac{1}{\sqrt{2}}[\varphi_{j+1,2k}(t) - \varphi_{j+1,2k+1}(t)] \\ \varphi_{j,k}(t) &= \frac{1}{\sqrt{2}}[\varphi_{j+1,2k}(t) + \varphi_{j+1,2k+1}(t)]\end{aligned}\quad (2.21)$$

Consider now the inner products of (2.18):

$$\begin{aligned}a_{j,k} &= \langle y, \varphi_{j,k} \rangle = \int y(t) \varphi_{j,k}(t) dt = \\ &= \int y(t) \frac{1}{\sqrt{2}}[\varphi_{j+1,2k}(t) + \varphi_{j+1,2k+1}(t)] dt = \\ &= \frac{1}{\sqrt{2}} a_{j+1,2k} + \frac{1}{\sqrt{2}} a_{j+1,2k+1}\end{aligned}\quad (2.22)$$

$$\begin{aligned}d_{j,k} &= \langle y, \psi_{j,k} \rangle = \int y(t) \psi_{j,k}(t) dt = \\ &= \int y(t) \frac{1}{\sqrt{2}}[\varphi_{j+1,2k}(t) - \varphi_{j+1,2k+1}(t)] dt = \\ &= \frac{1}{\sqrt{2}} a_{j+1,2k} - \frac{1}{\sqrt{2}} a_{j+1,2k+1}\end{aligned}\quad (2.23)$$

Equations (2.22) and (2.23) tell us that it is possible to obtain from the $a_{j,k}$ coefficients corresponding to scale j , the $a_{j-1,k}$ and $d_{j-1,k}$ coefficients corresponding to scale $j - 1$. Again, from the coefficients corresponding to scale $j - 1$ it is possible to obtain the coefficients for scale $j - 2$. And so on. This results in an easy iterative procedure to compute the wavelet expansion (2.18).

Let us illustrate the iterative method putting numbers for the 8 samples of the signal being considered so far:

$$y = \{3, 7, 1, 15, 2, 3, 6, 9\} \quad (2.24)$$

where integers have been used for simplicity.

The set of numbers (2.24) is also the set of $a_{j,k}$ coefficients corresponding to the scale $j = 3$. Let us compute the coefficients for the scale $j - 1$:

$$\begin{aligned}a_{2,0} &= \frac{1}{\sqrt{2}}(3 + 7) = \frac{10}{\sqrt{2}}; & a_{2,1} &= \frac{1}{\sqrt{2}}(1 + 15) = \frac{16}{\sqrt{2}} \\ a_{2,2} &= \frac{1}{\sqrt{2}}(2 + 3) = \frac{5}{\sqrt{2}}; & a_{2,3} &= \frac{1}{\sqrt{2}}(6 + 9) = \frac{15}{\sqrt{2}}\end{aligned}\quad (2.25)$$

$$\begin{aligned}d_{2,0} &= \frac{1}{\sqrt{2}}(3 - 7) = \frac{-4}{\sqrt{2}}; & d_{2,1} &= \frac{1}{\sqrt{2}}(1 - 15) = \frac{-14}{\sqrt{2}} \\ d_{2,2} &= \frac{1}{\sqrt{2}}(2 - 3) = \frac{-1}{\sqrt{2}}; & d_{2,3} &= \frac{1}{\sqrt{2}}(6 - 9) = \frac{-3}{\sqrt{2}}\end{aligned}\quad (2.26)$$

Expressions (2.25) and (2.26) show why we refer to the coefficients $a_{j,k}$ as averages and $d_{j,k}$ as differences.

Let us continue with the next lower scale:

$$a_{1,0} = \frac{1}{2}(10 + 16) = \frac{26}{2}; \quad a_{1,1} = \frac{1}{2}(5 + 15) = \frac{20}{2} \quad (2.27)$$

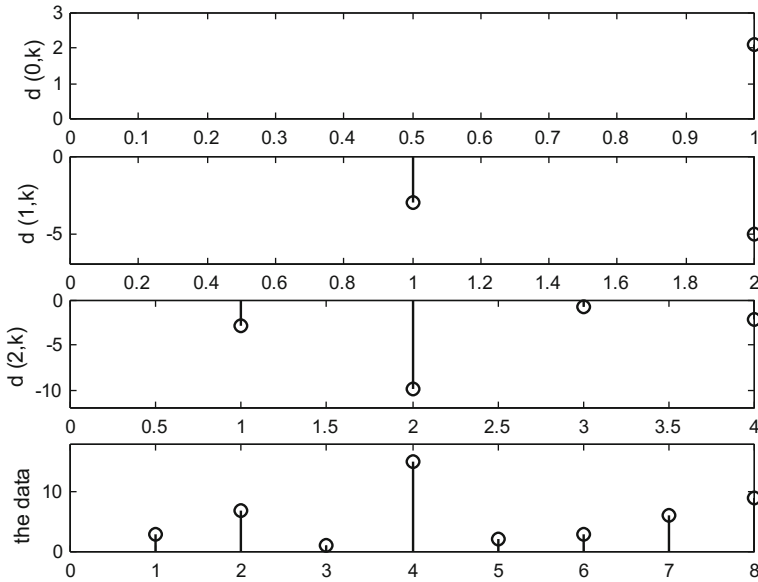


Fig. 2.9 Example of Haar transform

$$d_{1,0} = \frac{1}{2} (10 - 16) = \frac{-6}{2}; \quad d_{1,1} = \frac{1}{2} (5 - 15) = \frac{-10}{2} \quad (2.28)$$

and finally:

$$a_{0,0} = \frac{1}{2\sqrt{2}} (26 + 20) = \frac{46}{2\sqrt{2}} \quad (2.29)$$

$$d_{0,0} = \frac{1}{2\sqrt{2}} (26 - 20) = \frac{6}{2\sqrt{2}} \quad (2.30)$$

In this way, all coefficients in (2.19) have been determined.

The iterative coefficient computation method can be expressed in matrix format or with operators.

The Program 2.1 implements the procedure just described, with the same example. Figure 2.9 depicts the input data and the corresponding $d(j, k)$ coefficients.

Program 2.1 Haar wavelet transform of a data set

```
% Haar wavelet transform of a data set
y=[3,7,1,15,2,3,6,9]; %data set
Ns=8; %number of samples
K=3; %exponent, 8=2^3
wt=y; %space for the wavelet transform
d=zeros(K,Ns/2); %space for d(j,k) coefficients
for n=1:K,
```

```

aux1= wty(1:2:Ns-1) + wty(2:2:Ns);
aux2= wty(1:2:Ns-1) - wty(2:2:Ns);
wty(1:Ns)=[aux1,aux2]/sqrt(2);
d(K+1-n,1:Ns/2)=wty(1+(Ns/2):Ns); %fill d(j,k) coefficients
Ns=Ns/2;
end;
subplot(4,1,1)
stem(d(1,1),'k'); hold on; %the d(0,k) coefficients
axis([0 1 0 3]); ylabel('d(0,k)');
title('Example of Haar wavelet transform');
subplot(4,1,2)
stem(d(2,1:2),'k'); hold on; %the d(1,k) coefficients
axis([0 2 -7 0]); ylabel('d(1,k)');
subplot(4,1,3)
stem(d(3,1:4),'k'); hold on; %the d(2,k) coefficients
axis([0 4 -12 0]); ylabel('d(2,k)');
subplot(4,1,4)
stem(y,'k'); hold on; %the data
axis([0 8 0 18]);
ylabel('the data');
d
wty

```

It is possible to recover the original signal samples from the computed coefficients. Notice that adding of subtracting Eqs. (2.22) and (2.23) the following is obtained:

$$\begin{aligned}
 a_{j,k} + d_{j,k} &= \frac{2}{\sqrt{2}} a_{j+1,2k} \\
 a_{j,k} - d_{j,k} &= \frac{2}{\sqrt{2}} a_{j+1,2k+1}
 \end{aligned}
 \tag{2.31}$$

Consequently the iterative method can be reversed, computing the coefficients at scale 1 from coefficients at scale 0, coefficients at scale 2 from those at scale 1, etc. The last iteration gives the $a_{j,k}$ coefficients at scale j , which are the original signal samples.

The Program 2.2 recovers the original data from the Haar transform for the above example. Figure 2.10 depicts the $a(j, k)$ coefficients.

Program 2.2 Recover data set from Haar wavelet transform

```

% recover data set from Haar wavelet transform
%the wavelet transform data:
wty=[16.2635,2.1213,-3.0000,-5.0000,-2.8284,...
-9.8995,-0.7071,-2.1213];
K=3; %exponent, 8=2^3, for 8 samples
J=K+1; %to adapt to MATLAB indexing
y=wty; %space for the recovered data
a=zeros(J, (2^K)); %space for coefficients
m=1;
a(1,1)=y(1);
for n=1:K,

```

```

a(n+1,1:2:(2*m-1))=(a(n,1:m)+y((1+m):(2*m)))/sqrt(2);
a(n+1,2:2:(2*m))=(a(n,1:m)-y((1+m):(2*m)))/sqrt(2);
m=m*2;
end;
y=a(4,1:8); %the recovered data
subplot(4,1,1)
stem(a(1,1),'k'); hold on; %the a(0,k) coefficients
axis([0 1 0 20]); ylabel('a(0,k)');
title('Example of data recovery');
subplot(4,1,2)
stem(a(2,1:2),'k'); hold on; %the a(1,k) coefficients
axis([0 2 0 15]); ylabel('a(1,k)');
subplot(4,1,3)
stem(a(3,1:4),'k'); hold on; %the a(2,k) coefficients
axis([0 4 0 15]); ylabel('a(2,k)');
subplot(4,1,4)
stem(y,'k'); hold on; %the data
axis([0 8 0 18]);
ylabel('the data');

```

As an exercise let us select a sawtooth signal and obtain its Haar wavelet transform. The Program 2.3 repeats the transform procedure for this case. Figure 2.11 shows the input signal and the corresponding $d(j, k)$ coefficients. Most of the coefficients

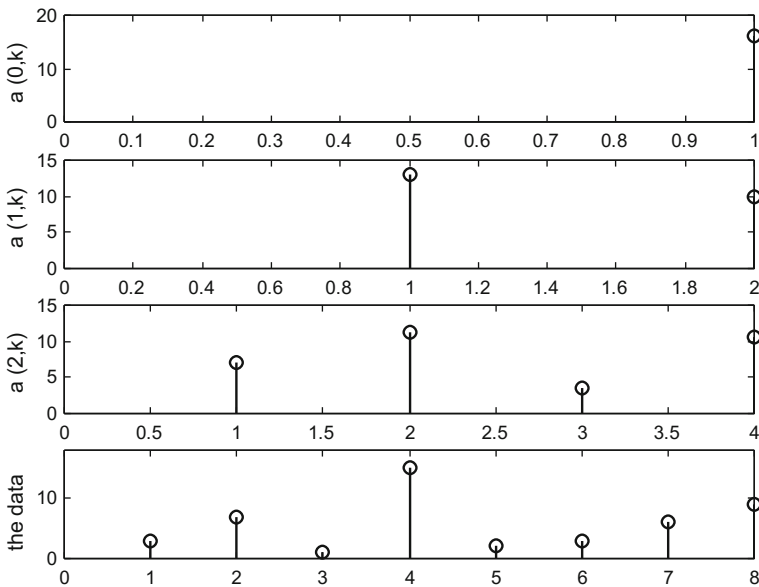
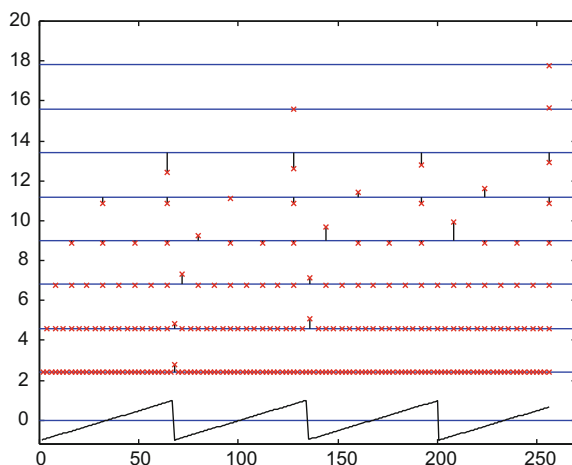
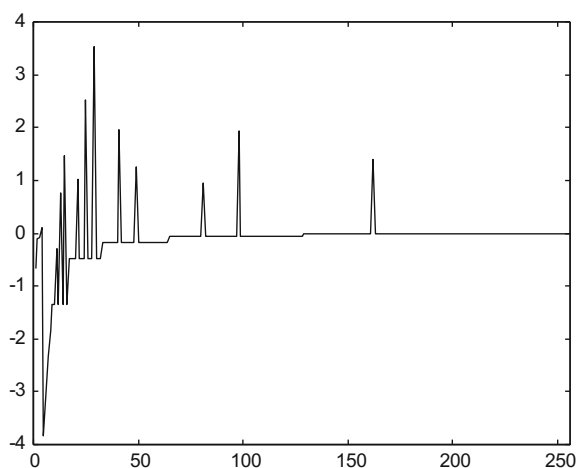


Fig. 2.10 Example of data recovery from Haar transform

are small and difficult to see, but the coefficients corresponding to the brisk changes of the sawtooth are clearly seen.

Program 2.3 Haar wavelet transform of a sawtooth signal

```
% Haar wavelet transform of a signal
% Sawtooth signal
fy=300; %signal frequency in Hz
wy=2*pi*fy; %signal frequency in rad/s
fs=20000; %sampling frequency in Hz
tiv=1/fs; %time interval between samples;
Ns=256; %let us take 256 signal samples
duy=Ns*tiv; %time for 256 samples
t=0:tiv:(duy-tiv); %time intervals set
y=sawtooth(wy*t); %signal data set (256 samples)
K=8; %exponent, 256=2^8
wty=y; %space for the wavelet transform
d=zeros(K,Ns/2); %space for d(j,k) coefficients
%the Haar wavelet transform
for n=1:K,
    aux1= wty(1:2:Ns-1) + wty(2:2:Ns);
    aux2= wty(1:2:Ns-1) - wty(2:2:Ns);
    wty(1:Ns)=[aux1,aux2]/sqrt(2);
    d(K+1-n,1:Ns/2)=wty(1+(Ns/2):Ns); %fill d(j,k) coefficients
    Ns=Ns/2;
end;
%figure
%scaling
dmax=max(max(d));
dmin=min(min(d)); abDMIN=abs(dmin);
if abDMIN>dmax, mh=abDMIN; else mh=dmax; end;
%area and signal
plot([0 270],[0 0],'b'); hold on;
plot(y,'k'); %the signal
axis([0 270 -1.2 20]);
%subplots
for nn=1:8,
    nx=2^(nn-1);
    ylevel=20-(2.2*nn);
    plot([0 270],[ylevel ylevel],'b'); %horizontal axes
    ydat=d(nn,1:nx)/mh; %data scaling
    yno=zeros(1,nx);
    ivx=256/nx; %horizontal interval
    for k=1:nx,
        plot([ivx*(k) ivx*(k)],[ylevel+yno(k) ylevel+ydat(k)], 'k');
        plot([ivx*(k) ivx*(k)],[ylevel+ydat(k)ylevel+ydat(k)], 'rx');
    end;
end;
title('Haar wavelet transform of sawtooth signal');
```

Fig. 2.11 Haar transform of sawtooth signal**Fig. 2.12** The vector $wty(n)$ corresponding to the sawtooth signal

Notice that in the wavelet transform examples just considered, a vector $wty(n)$ has been used to contain $a(1, 1)$, $d(1, 1)$, $d(2, 1)$, $d(2, 2)$, $d(3, 1)$... (indexes according to MATLAB). It is convenient to plot this vector, since it gives quick information of the values, variance, etc. Likewise it may be useful to compare wavelet transform alternatives.

Figure 2.12 shows the vector $wty(n)$ obtained by the Haar wavelet transform of the sawtooth signal just considered. The figure has been generated with the Program 2.4.

Program 2.4 Haar wavelet transform of a sawtooth signal: `wty`

```
% Haar wavelet transform of a signal
% Sawtooth signal
% Plot of wty
fy=300; %signal frequency in Hz
```

```

wy=2*pi*fy; %signal frequency in rad/s
fs=20000; %sampling frequency in Hz
tiv=1/fs; %time interval between samples;
Ns=256; %let us take 256 signal samples
duy=Ns*tiv; %time for 256 samples
t=0:tiv:(duy-tiv); %time intervals set
y=sawtooth(wy*t); %signal data set (256 samples)
K=8; %exponent, 256=2^8
wty=y; %space for the wavelet transform
%the Haar wavelet transform
for n=1:K,
aux1= wty(1:2:Ns-1) + wty(2:2:Ns);
aux2= wty(1:2:Ns-1) - wty(2:2:Ns);
wty(1:Ns)=[aux1,aux2]/sqrt(2);
Ns=Ns/2;
end;
%figure
plot(wty, 'k');
axis([0 256 -4 4]);
title('Vector wty obtained by sawtooth Haar wavelet transform')

```

Now let us try to recover the sawtooth signal from its Haar wavelet transform. This is made with the Program 2.5, repeating the procedure of Program 2.2. The recovered signal is shown in Fig. 2.13.

Program 2.5 Sawtooth signal recovery from Haar wavelet transform

```

% Haar wavelet transform of a signal
% Sawtooth signal recovery from transform
%-----
% First the wavelet transform to get data
fy=300; %signal frequency in Hz
wy=2*pi*fy; %signal frequency in rad/s
fs=20000; %sampling frequency in Hz
tiv=1/fs; %time interval between samples;
Ns=256; %let us take 256 signal samples
duy=Ns*tiv; %time for 256 samples
t=0:tiv:(duy-tiv); %time intervals set
y=sawtooth(wy*t); %signal data set (256 samples)
K=8; %exponent, 256=2^8
wty=y; %space for the wavelet transform
%the Haar wavelet transform
for n=1:K,
aux1= wty(1:2:Ns-1) + wty(2:2:Ns);
aux2= wty(1:2:Ns-1) - wty(2:2:Ns);
wty(1:Ns)=[aux1,aux2]/sqrt(2);
Ns=Ns/2;
end;
%-----
% Second the signal recovery from the wavelet transform data

```

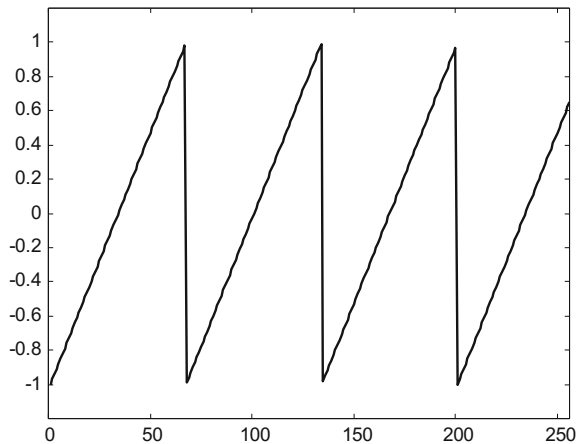
```

J=K+1;
z=wtY; %space for recovered data
a=zeros(J, (2^K)); %space for a(j,k) coefficients
m=1;
a(1,1)=z(1);
for n=1:K,
a(n+1,1:2:(2*m-1))=(a(n,1:m)+z((1+m):(2*m)))/sqrt(2);
a(n+1,2:2:(2*m))=(a(n,1:m)-z((1+m):(2*m)))/sqrt(2);
m=m*2;
end;
y=a(J,1:256); %the recovered data
figure
plot(y, 'k');
axis([0 256 -1.2 1.2]);
title('Recovered sawtooth signal, from Haar wavelet transform');

```

There is an intuitive visualization of wavelet coefficients called ‘*scalogram*’. The x-axis of this representation is discrete time (samples), and the y-axis is the scale. Figure 2.14 shows the scalogram of a sawtooth signal, corresponding to the decomposition of this signal into Haar wavelets. In strict terms, the scalogram should show squared (positive) values of the coefficients, but in our case we preferred to show the original values, negative or positive, letting to MATLAB normalization tasks. The values are painted in pseudocolors or in gray scale, for instance. The figure has been generated with the Program 2.6, which contains interesting code. The figure clearly shows the correspondence of signal peaks and the detection of high frequencies by the wavelets (the higher scales are associated to high frequencies).

Fig. 2.13 Recovering the sawtooth signal from its Haar wavelet transform



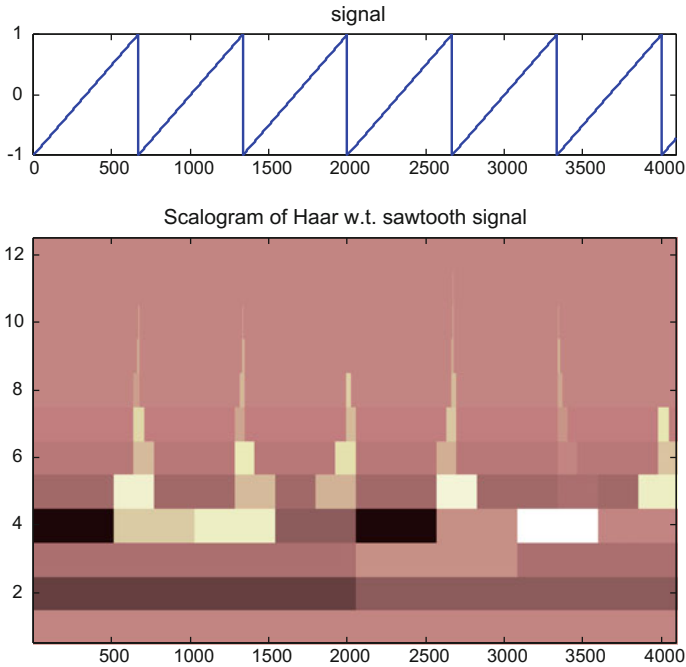


Fig. 2.14 Scalogram of sawtooth signal using Haar wavelets

Program 2.6 Scalogram of sawtooth signal, Haar wavelet

```
% Haar wavelet transform of a signal
% SCALOGRAM
% Sawtooth signal
fy=600; %signal frequency in Hz
wy=2*pi*fy; %signal frequency in rad/s
fs=4*(10^5); %sampling frequency in Hz
tiv=1/fs; %time interval between samples;
K=12; %exponent
Ns=2^K; %let us take 2^K signal samples
duy=Ns*tiv; %time for 2^K samples
t=0:tiv:(duy-tiv); %time intervals set
y=sawtooth(wy*t); %signal data set (256 samples)
wty=y; %space for the wavelet transform
d=zeros(K,Ns/2); %space for d(j,k) coefficients
NN=Ns;
%the Haar wavelet transform
for n=1:K,
    aux1= wty(1:2:NN-1) + wty(2:2:NN);
    aux2= wty(1:2:NN-1) - wty(2:2:NN);
    wty(1:NN)=[aux1,aux2]/sqrt(2);
    d(K+1-n,1:NN/2)=wty(1+(NN/2):NN); %fill d(j,k) coefficients
```



```

NN=NN/2;
end;



---



```

2.2.3 Wavelets and Filter Banks

The iterative method just explained can be expressed as filtering the data. The computation of $a_{j,k}$ coefficients implies averaging, which is low pass filtering. Let us denote this low pass filter as $LP(z)$. Likewise, the computation of $d_{j,k}$ coefficients implies differencing, which is high pass filtering with the filter $HP(z)$. Both filters are moved along the input data. The expressions of these Haar filters are the following:

$$LP(z) = \frac{1}{\sqrt{2}} (z^{-1} + 1) \quad (2.32)$$

$$HP(z) = \frac{1}{\sqrt{2}} (z^{-1} - 1) \quad (2.33)$$

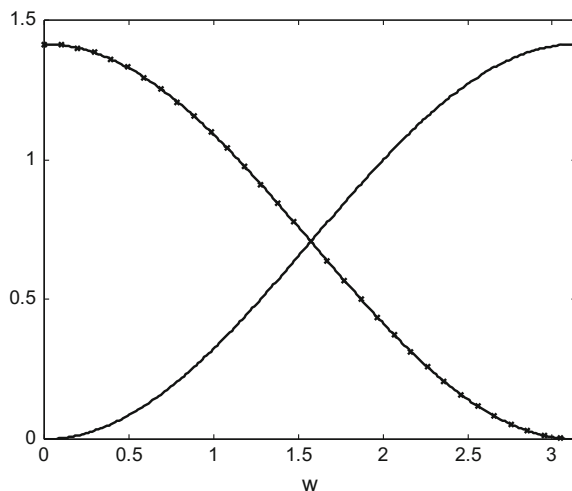
The frequency responses of the Haar filters are:

$$|LP(\omega)| = \sqrt{2} \cos\left(\frac{\omega}{2}\right) \quad (2.34)$$

$$|HP(\omega)| = \sqrt{2} \sin\left(\frac{\omega}{2}\right) \quad (2.35)$$

Figure 2.15 shows the frequency magnitude responses of the Haar LP and HP filters. According with the nomenclature of the filter banks, LP is $H_0(z)$ and HP is $H_1(z)$.

Fig. 2.15 Frequency magnitude responses of the LP and HP Haar filters



The Fig. 2.15 has been generated with the Program 2.7, which is almost the same as the program used in the previous chapter for the QMF filter bank (Sect. 1.4.2).

Program 2.7 Frequency magnitude response of Haar filters H0 and H1

```
% Frequency magnitude response of Haar filters H0 and H1
c=1/sqrt(2);
h0=[c c]; %low-pass filter
h1=[-c c]; %high-pass filter
w=0:(2*pi/511):pi;
H0=real(fft(h0,512)); %discrete Fourier transform
H1=real(fft(h1,512)); % ""
plot(w,H0(1:256),'k',w(1:8:256),H0(1:8:256),'kx'); hold on;
plot(w,abs(H1(1:256)),'k');
axis([0 pi 0 1.5]);
title('frequency response of QMF H0 and H1 filters');
xlabel('w');
```

Consider again the two-channel filter bank, as in Fig. 2.16.

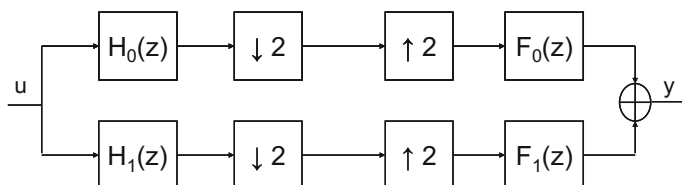


Fig. 2.16 A two-channel filter bank

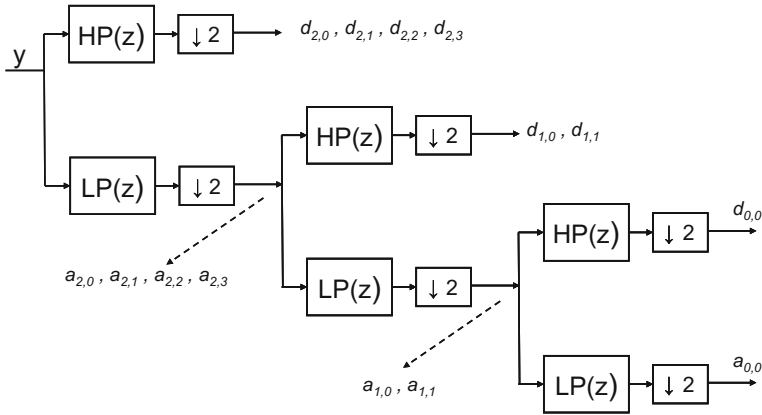


Fig. 2.17 Wavelet expansion using filter bank

If the Haar filters are chosen for $H_0(z)$ and $H_1(z)$, and the other two filters are designed as:

$$\begin{aligned} F_0(z) &= H_1(-z) \\ F_1(z) &= -H_0(-z) \end{aligned} \quad (2.36)$$

Then a QMF filter bank is obtained, with perfect reconstruction (see Sect. 1.4.2).

Actually, the iterative method for the wavelet decomposition can be expressed with two-channel filter banks. Figure 2.17 shows a filter bank corresponding to the example considered in Eq. (2.24) and thereafter. This is in clear connection with the topics of the previous chapter.

Let us visit again the example of the sawtooth signal for comparison purposes. The Program 2.8 applies Haar filters to obtain the wavelet transform of the sawtooth signal. Figure 2.18 shows the vector $wty(n)$ that is obtained: it is exactly the same obtained by Program 2.4 (Fig. 2.12).

Program 2.8 Haar wavelet transform of sawtooth signal using filters

```
% Haar wavelet transform of a signal using filters
% Sawtooth signal
% Plot of wty
% The Haar filters
c=1/sqrt(2);
h0=[c c]; %low-pass filter
h1=[-c c]; %high-pass filter
%The sawtooth signal
fy=300; %signal frequency in Hz
wy=2*pi*fy; %signal frequency in rad/s
duy=0.03; %signal duration in seconds
fs=20000; %sampling frequency in Hz
tiv=1/fs; %time interval between samples;
Ns=256; %let us take 256 signal samples
```

```

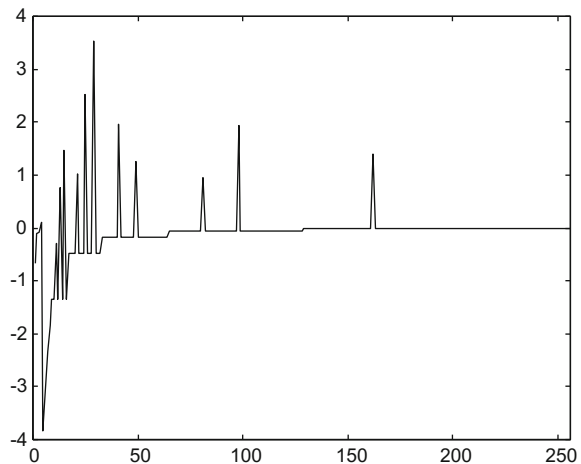
duy=Ns*tiv; %time for 256 samples
t=0:tiv:(duy-tiv); %time intervals set
y=sawtooth(wy*t); %signal data set (256 samples)
%Haar wavelet transform using filters
K=8; %exponent, 256=2^8
wtv=y; %space for the wavelet transform
d=zeros(K,Ns/2); %space for d(j,k) coefficients
a=zeros(K,Ns/2); %space for a(j,k) coefficients
My=y; %auxiliar vector
Ls=zeros(1,128); %""
Hs=zeros(1,128); %""
for nn=K:-1:1,
m=2^nn;
lx=filter(h0,1,My); Ls(1:(m/2))=lx(2:2:m);
a(nn,1:(m/2))=Ls(1:(m/2)); %LP and subsampling
hx=filter(h1,1,My); Hs(1:(m/2))=hx(2:2:m);
d(nn,1:(m/2))=Hs(1:(m/2)); %HP and subsampling
My=Ls(1:(m/2));
wtv((1+(m/2)):m)=d(nn,1:(m/2)); %append to wtv
end;
wtv(1)=a(1,1);
%figure
plot(wtv,'k');
axis([0 256 -4 4]);
title('Vector wtv with Haar wavelet transform using filters');

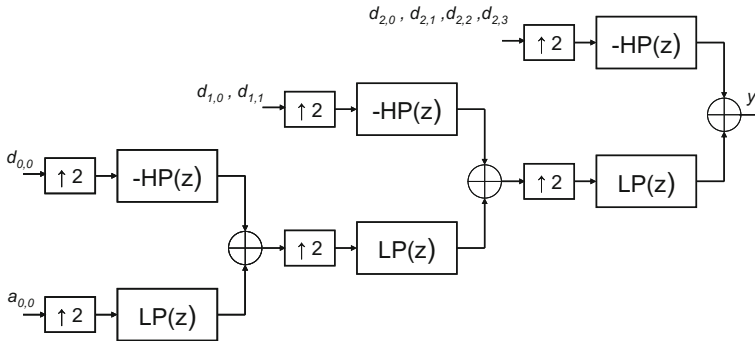
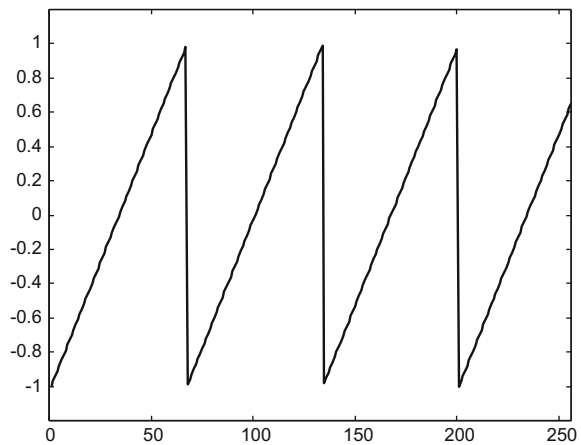
```

The original data can be recovered with another filter bank. Figure 2.19 shows the filter bank that reverses the action of the filter bank represented in Fig. 2.17.

The Program 2.9 applies Haar filters to recover the sawtooth signal from its wavelet transform. Figure 2.20 shows the result.

Fig. 2.18 The vector $wtv(n)$ corresponding to the sawtooth signal



**Fig. 2.19** Recovering original data from wavelet expansion**Fig. 2.20** Recovering the sawtooth signal from its Haar wavelet transform**Program 2.9** Recovering of sawtooth signal from its Haar wavelet transform, using filters

```
% Haar wavelet transform of a signal using filters
% Recovering of sawtooth signal from its transform
% The Haar filters, analysis part
c=1/sqrt(2);
h0=[c c]; %low-pass filter
h1=[-c c]; %high-pass filter
%-----
% First the wavelet transform to get the data
%The sawtooth signal
fy=300; %signal frequency in Hz
wy=2*pi*fy; %signal frequency in rad/s
duy=0.03; %signal duration in seconds
fs=20000; %sampling frequency in Hz
tiv=1/fs; %time interval between samples;
Ns=256; %let us take 256 signal samples
duy=Ns*tiv; %time for 256 samples
```

```

t=0:tiv:(duy-tiv); %time intervals set
y=sawtooth(wy*t); %signal data set (256 samples)
%Haar wavelet transform using filters
K=8; %exponent, 256=2^8
wt=y; %space for the wavelet transform
d=zeros(K,Ns/2); %space for d(j,k) coefficients
a=zeros(K,Ns/2); %space for a(j,k) coefficients
My=y; %auxiliar vector
Ls=zeros(1,128); %""
Hs=zeros(1,128); %""
for nn=K:-1:1,
m=2^nn;
lx=filter(h0,1,My); Ls(1:(m/2))=lx(2:2:m);
a(nn,1:(m/2))=Ls(1:(m/2)); %LP and subsampling
hx=filter(h1,1,My); Hs(1:(m/2))=hx(2:2:m);
d(nn,1:(m/2))=Hs(1:(m/2)); %HP and subsampling
My=Ls(1:(m/2));
wt((1+(m/2)):m)=d(nn,1:(m/2)); %append to wty
end;
wt(1)=a(1,1);
%-----
% Second the recovery from the wavelet transform
% The Haar filters, synthesis part
c=1/sqrt(2);
f0=[c c]; %low-pass filter
f1=[c -c]; %high-pass filter
J=K+1;
z=wt; %space for recovered data
a=zeros(J,(2^K)); %space for a(j,k) coefficients
Ma=zeros(1,(2^K)); Md=Ma; %auxiliar vectors
m=1;
a(1,1)=z(1); d(1,1)=z(2);
Ma(1)=a(1,1); Md(1)=d(1,1);
for nn=1:(K-1),
m=2^nn;
lx=filter(f0,1,Ma); hx=filter(f1,1,Md);
a(nn+1,1:m)=lx(1:m)+hx(1:m);
Ma(1:2:(m*2))=a(nn+1,1:m); %upsampling
Md(1:2:(m*2))=z((1+m):(m*2)); %""
end;
nn=8; m=2^nn;
lx=filter(f0,1,Ma); hx=filter(f1,1,Md);
a(nn+1,1:m)=lx(1:m)+hx(1:m);
y=a(J,1:256); %the recovered data
%figure
plot(y,'k');
axis([0 256 -1.2 1.2]);
title('Recovered sawtooth from Haar wavelet transform
using filters');

```

2.3 The Multiresolution Analysis Equation

After the introductory example of Haar wavelets, it is convenient to generalize the main ideas.

Signal processing with wavelets is based on the use of scaling functions and wavelet functions. A scaling function $\varphi(t)$ is given by the recursive equation:

$$\varphi(t) = \sum_n h_0(n) \sqrt{2} \varphi(2t - n); \quad \varphi(t) \in L^2; \quad n = 0, 1, 2, \dots, N-1 \quad (2.37)$$

This is an important equation, which receives several names like refinement equation, multiresolution analysis equation (MAE), dilation equation, etc.

In general, given a certain type of signals to be processed, certain properties of the wavelets and the scaling functions are sought. This usually implies bounds for the values of the coefficients $h_0(n)$. Once a set of $h_0(n)$ values is specified, the scaling function $\varphi(t)$ can be determined by recursion.

A family of functions is generated from the scaling function as follows:

$$\varphi_{j,k}(t) = \sqrt{2^j} \varphi(2^j t - k) \quad (2.38)$$

Denote the Fourier transform of $\varphi(t)$ as $\Phi(\omega)$, and:

$$H_0(\omega) = \sum_{n=-\infty}^{\infty} h_0(n) e^{j\omega n} \quad (2.39)$$

(the frequency response of the LP(z) filter)

Then the frequency domain version of the MAE is:

$$\Phi(\omega) = \frac{1}{\sqrt{2}} H_0\left(\frac{\omega}{2}\right) \Phi\left(\frac{\omega}{2}\right) \quad (2.40)$$

which after iteration becomes:

$$\Phi(\omega) = \prod_{k=1}^{\infty} \left[\frac{1}{\sqrt{2}} H_0\left(\frac{\omega}{2^k}\right) \right] \Phi(0) \quad (2.41)$$

2.3.1 Solving the MAE

2.3.1.1 Necessary Conditions

There is a set of necessary conditions for $\varphi(t)$ to be a solution of the MAE. These conditions can be used to deduce the $h_0(n)$ coefficients. Let us list some of them:

- If $\int \varphi(t) dt \neq 0$:

$$\sum_n h_0(n) = H_0(0) = \sqrt{2} \quad (2.42)$$

Notice that $H_0(0)$ is the frequency response of the filter $LP(z)$ at DC.

- If $\int \varphi(t) dt = 1$ and

$$\sum_k \varphi(t - k) = \sum_k \varphi(k) = 1 \quad (2.43)$$

then

$$\sum_n h_0(2n) = \sum_n h_0(2n + 1) \quad (2.44)$$

and

$$H_0(\pi) = 0 \quad (2.45)$$

Equation (2.43) is called a partitioning of unity . Also, in the opposite direction, if (2.44) is true then (2.43) is true.

- If the integer translates of $\varphi(t)$ are **orthogonal**:

$$\sum_n h_0(n) h_0(n - 2k) = \begin{cases} 1, & \text{if } k = 0 \\ 0, & \text{otherwise} \end{cases} \quad (2.46)$$

$$\sum_n |h_0(n)|^2 = 1 \quad (2.47)$$

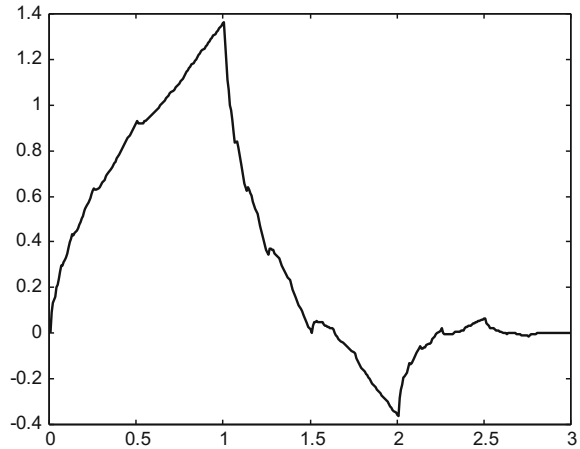
$$\sum_n |\Phi(\omega + 2\pi k)|^2 = 1 \quad (2.48)$$

$$\sum_n h_0(2n) = \sum_n h_0(2n + 1) = \frac{1}{\sqrt{2}} \quad (2.49)$$

$$|H_0(\omega)|^2 + |H_0(\omega + \pi)|^2 = 2 \quad (2.50)$$

The coefficients that satisfy (2.46) are coefficients of a QMF filter.

Fig. 2.21 Daubechies 4 scaling function



2.3.1.2 Iterative Computation of the Scaling Function

Usually there is no need for using scaling functions or wavelets explicitly. Instead wavelet transforms are applied by filtering, so the important issue is to know the values of $h_0(n)$ and $h_1(n)$.

Anyway, it is interesting to introduce two iterative approaches to compute the scaling function.

The first method is called the cascade algorithm, and consists of using the MAE for a series of iterations:

$$\varphi^{(l+1)}(t) = \sum_n h_0(n) \sqrt{2} \varphi^{(l)}(2t - n), \quad l = 0, 1, 2, \dots \quad (2.51)$$

An initial $\varphi^{(0)}(t)$ must be given.

Program 2.10 implements the cascade algorithm for the Daubechies 4 case. It generates the Fig.2.21, which shows the famous Daubechies 4 scaling function. Notice its fractal nature.

Program 2.10 Compute a scaling function from MAE

```
% Compute a scaling function
% from MAE
% MAE coefficients
hden=4*sqrt(2); %coeff. denominator
hsq=sqrt(3); %factor
%Daubechies 4:
h=[(1+hsq)/hden, (3+hsq)/hden, (3-hsq)/hden, (1-hsq)/hden];
hN=(h*2)/sum(h); %normalization
K=length(hN);
Ns=128; %number of fi samples
```

```

fi=[ones(1,3*K*Ns),0]/(3*K); %initial iteration
%upsample hN, inserting Ns-1 zeros between samples
hup=[hN;zeros(Ns-1,K)];
hup=hup(1:(Ns*K));
%iteration
for nn=0:12,
aux=conv(hup,fi);
fi=aux(1:2:length(aux)); %downsampling by 2
end
%result
fi=fi(1:(K-1)*Ns); %the supported part
x=(1:length(fi))/Ns;
plot(x,fi,'k'); %plots the scaling function
title('Daubechies 4 scaling function');

```

The frequency domain form of the iterative method is:

$$\Phi^{(l+1)}(\omega) = \frac{1}{\sqrt{2}} H\left(\frac{\omega}{2}\right) \Phi^{(l)}\left(\frac{\omega}{2}\right), \quad l = 0, 1, 2, \dots \quad (2.52)$$

If there is a solution for the equation, the result of the iterations is the Fourier transform of the scaling function, and this can be written as follows:

$$\Phi(\omega) = \left\{ \prod_{k=1}^{\infty} \left(\frac{1}{\sqrt{2}} H\left(\frac{\omega}{2^k}\right) \right) \right\} \Phi(0) \quad (2.53)$$

Based on this last equation, a very effective procedure can be implemented, which proceeds from the largest value of k to the lowest. Figure 2.22 shows the first six successive iterations, converging to the target scaling function. The figure has been generated with the Program 2.11. Notice that $\text{fft}()$ is required only once.

Program 2.11 Computing a scaling function using FFT and MAE

```

% Computing a scaling function using FFT and MAE
% Display of first iterations
%Daubechies 4, coeffs.
hden=4*sqrt(2); %coeff. denominator
hsq=sqrt(3); %factor
%Daubechies 4:
h=[(1+hsq)/hden, (3+hsq)/hden, (3-hsq)/hden, (1-hsq)/hden];
hN=(h*2)/sum(h); %normalization
Ns=2^12; %number of samples
Ffi=fft(hN,Ns); %Initial Fourier transform
aux=Ffi;
for nn=1:6,
%display
subplot(2,3,nn);
plot(abs(aux(1:(Ns/2))), 'k'); axis([0 Ns/2 0 2*nn]);

```

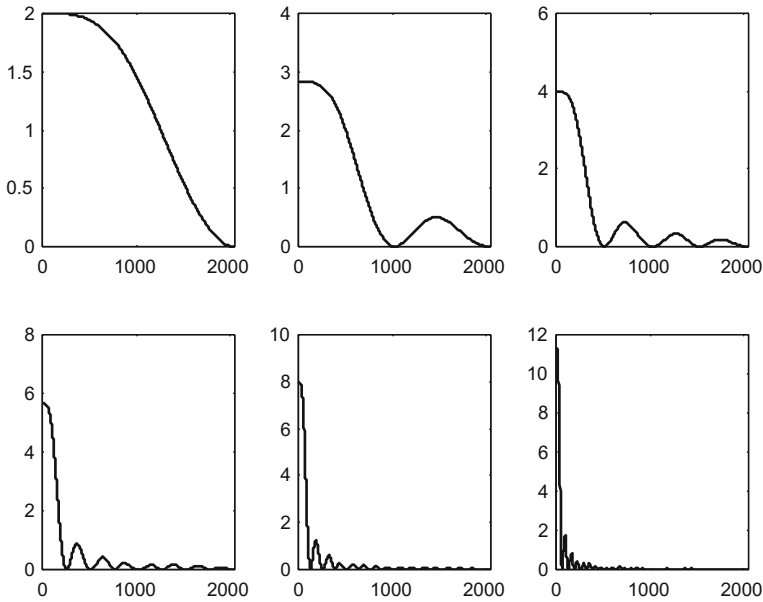


Fig. 2.22 Successive approximations to the scaling function (Daubechies case)

```
%iteration
Ffi=[Ffi(1:2:Ns),Ffi(1:2:Ns)];
aux=(aux.*Ffi)/sqrt(2);
end;
```

Once a good approximation of the frequency domain scaling function is obtained, it is a matter of inverse Fourier transform to get the time domain scaling function. Program 2.12 implements this method, and the result is shown in Fig. 2.23. The result is similar to Fig. 2.21, with a definition improvement since we get 1537 curve points instead of 384 points.

Program 2.12 Computing a scaling function using FFT and MAE: final result

```
% Computing a scaling function using FFT and MAE
% Display of final result
%Daubechies 4, coeffs.
hden=4*sqrt(2); %coeff. denominator
hsq=sqrt(3); %factor
%Daubechies 4:
h=[(1+hsq)/hden, (3+hsq)/hden, (3-hsq)/hden, (1-hsq)/hden];
hN=(h*2)/sum(h); %normalization
Ns=2^12; %number of samples
Ffi=fft(hN,Ns); %Initial Fourier transform
aux=Ffi;
for nn=1:8,
```

```

Ffi=[Ffi(1:2:Ns),Ffi(1:2:Ns)];
aux=(aux.*Ffi)/sqrt(2);
end;
fi=real(ifft(aux));
L=6*(2^8); %the supported part
t=((0:L-1)*3)/L;
ct=2^4; fi=ct*fi; %scaling
plot(t,fi(1:L),'k')
title('Daubechies 4 scaling function');

```

The other method is based on a dyadic expansion . The idea is to obtain the values of $\varphi(t)$ on the integers, by solving a set of simultaneous equations, and then calculate the values of $\varphi(t)$ at the half integers, then at the quarter integers, etc. Let us consider an example, with $h_0(n) \neq 0$ for $n = 0, 1, 2, 3$ and 4. The MAE at integers l is:

$$\varphi(l) = \sum_n h_0(n) \sqrt{2} \varphi(2l - n) \quad (2.54)$$

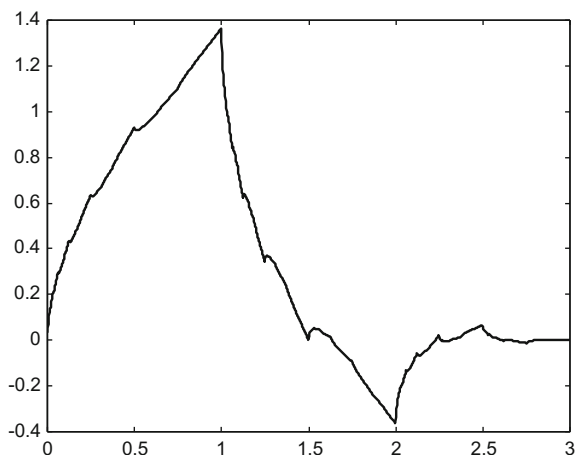
Thus:

$$\begin{pmatrix} h_0(0) & 0 & 0 & 0 & 0 \\ h_0(2) & h_0(1) & h_0(0) & 0 & 0 \\ h_0(4) & h_0(3) & h_0(2) & h_0(1) & h_0(0) \\ 0 & 0 & h_0(4) & h_0(3) & h_0(2) \\ 0 & 0 & 0 & 0 & h_0(4) \end{pmatrix} \begin{pmatrix} \varphi(0) \\ \varphi(1) \\ \varphi(2) \\ \varphi(3) \\ \varphi(4) \end{pmatrix} = \begin{pmatrix} \varphi(0) \\ \varphi(1) \\ \varphi(2) \\ \varphi(3) \\ \varphi(4) \end{pmatrix} \quad (2.55)$$

This equation can be written in matrix form:

$$M_0 \bar{\varphi} = \bar{\varphi} \quad (2.56)$$

Fig. 2.23 Daubechies 4 scaling function



where the matrix M_0 has an eigenvalue of unity; this is guaranteed by the condition:

$$\sum_n h_0(2n) = \sum_n h_0(2n+1) \quad (2.57)$$

The system of equations can be solved to obtain $\varphi(0), \dots, \varphi(4)$. Now, the MAE can also be written as:

$$\varphi(l/2) = \sum_n h_0(n) \sqrt{2} \varphi(l-n) \quad (2.58)$$

Consequently:

$$\begin{pmatrix} h_0(1) & h_0(0) & 0 & 0 & 0 \\ h_0(3) & h_0(2) & h_0(1) & h_0(0) & 0 \\ 0 & 0 & h_0(4) & h_0(3) & h_0(2) \\ 0 & 0 & 0 & 0 & h_0(4) \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \varphi(0) \\ \varphi(1) \\ \varphi(2) \\ \varphi(3) \\ \varphi(4) \end{pmatrix} = \begin{pmatrix} \varphi(1/2) \\ \varphi(3/2) \\ \varphi(5/2) \\ \varphi(7/2) \\ \varphi(9/2) \end{pmatrix} \quad (2.59)$$

(the last row is not needed)

In matrix form:

$$M_1 \bar{\varphi} = \bar{\varphi}_2 \quad (2.60)$$

The equation can be iterated to obtain the values of the scaling functions at quarter integers, etc.

Figure 2.24 shows the first iterations of the procedure, for the Daubechies 4 example again. The figure has been generated with the Program 2.13.

Program 2.13 Compute a scaling function with dyadic approach

```
% Compute a scaling function with dyadic approach
% display of first iterations
% Daubechies 4, coeffs.
hden=4*sqrt(2); %coeff. denominator
hsq=sqrt(3); %factor
%Daubechies 4:
h=[(1+hsq)/hden, (3+hsq)/hden, (3-hsq)/hden, (1-hsq)/hden];
hN=(h*2)/sum(h); %normalization
K=length(hN);
hrev=hN(K:-1:1); %reverse hN
%M0 matrix
MA=[hrev,zeros(1,(2*K)-2)];
MB=MA;
for nn=1:K-1,
MA=[0,0,MA(1:(3*K)-4)];
MB=[MB; MA];
end
M0=MB(:,K:(2*K)-1);
```

```

%Solving the first system of equations, for fi(0)..fi(3)
MF=M0-eye(K);
MG=[MF(1:K-1,:);ones(1,K)];
nfi=MG\[zeros(K-1,1);1];
%display
subplot(2,3,1);
x=(1:length(nfi))*(K-1)/length(nfi);
plot(x,nfi,'k',x,nfi,'dk');
axis([0 3 -0.5 1.5]);
%getting middle {&} quarter values
fi=nfi(2:length(nfi)-1);
fi=conv(hN,fi);
aux=fi(1:2:length(fi)); %downsampling by 2
%quarter values
y=conv(hN,aux);
%merge y and fi
aux=[y;fi,0]; aux=aux(:)';
fi=aux(1:length(aux)-1);
%display
subplot(2,3,2);
x=(1:length(fi))*(K-1)/length(fi); plot(x,fi,'k',x,fi,'dk');
axis([0 3 -0.5 1.5]);
%iteration
hup=hN; Nx=4;
for nn=1:Nx,
%upsample by 2
L=length(hup);
aux=[hup;zeros(1,L)]; aux=aux(:)';
hup=aux(1:2*L-1);
%intermediate terms
y=conv(hup,y);
%merge y and fi
aux=[y;fi,0]; aux=aux(:)';
fi=aux(1:length(aux)-1);
%display
subplot(2,3,2+nn);
x=(1:length(fi))*(K-1)/length(fi); plot(x,fi,'k',x,fi,'.k');
axis([0 3 -0.5 1.5]);
end

```

To complete the example, a program has been written (Program A.3) that implements the procedure to obtain 6143 curve points for the Daubechies 4 scaling function. Figure 2.25 shows the result. The program has been included in Appendix A.

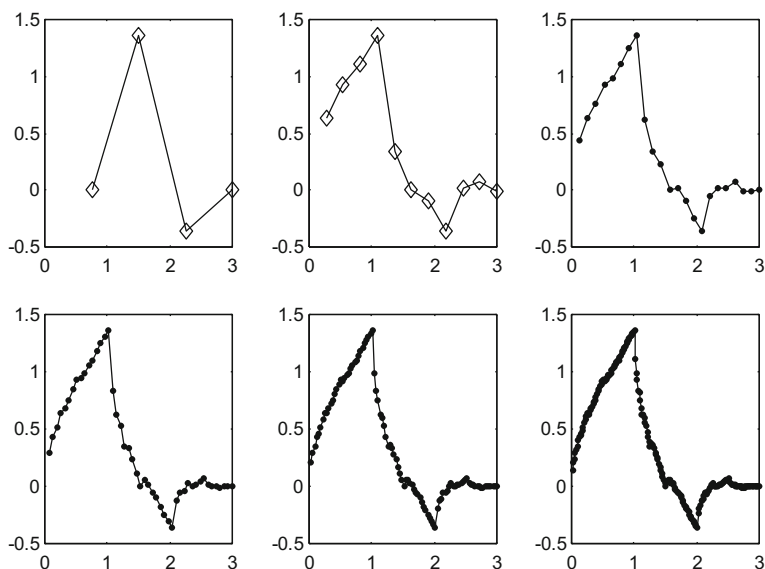
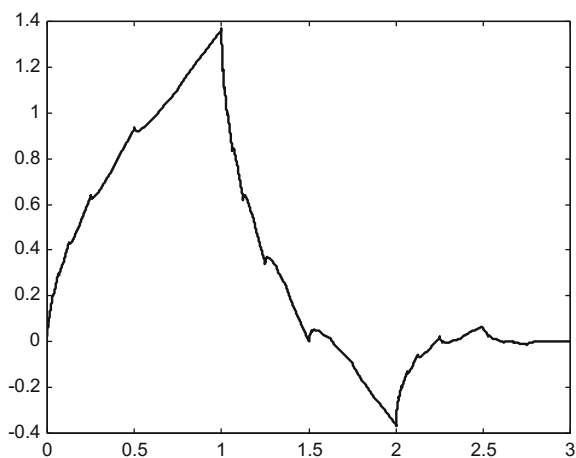


Fig. 2.24 First dyadic iterations to compute the Daubechies 4 scaling function

Fig. 2.25 Daubechies 4 scaling function, again



2.3.2 Scaling Functions, Wavelets, and Function Expansions

A wavelet function $\psi(t)$ can be obtained from scaling functions with the following equation:

$$\psi(t) = \sum_n h_1(n) \sqrt{2} \varphi(2t - n); \quad n = 0, 1, 2, \dots, N-1 \quad (2.61)$$

where $\psi(t)$ is a ‘mother wavelet’.

The designer can decide certain relationships of wavelet functions and scaling functions—for example to be orthogonal-, and this may determine both the coefficients $h_1(n)$ and the wavelet function $\psi(t)$, from $h_0(n)$ and $\varphi(t)$.

Another family of functions is generated from the ‘mother wavelet’:

$$\psi_{j,k}(t) = \sqrt{2^j} \psi(2^j t - k) \quad (2.62)$$

The goal is to generate a set of functions $\psi_{j,k}(t)$ such that any function $y(t)$ of interest could be written as an expansion in terms of wavelets and perhaps scaling functions. Actually there are two ways that we could expand $y(t)$:

- A low-resolution approximation plus its wavelet details:

$$y(t) = \sum_k a_{j_0,k} \varphi_{j_0,k}(t) + \sum_k \sum_{j=j_0}^{\infty} d_{j,k} \psi_{j,k}(t) \quad (2.63)$$

this is the most useful in practice; also in practice the upper value of j is finite ($= M - 1$).

- Only the wavelet details:

$$y(t) = \sum_{j,k} d_{j,k} \psi_{j,k}(t) \quad (2.64)$$

the set of coefficients $d_{j,k}$ is the discrete wavelet transform of $y(t)$.

The coefficients can be recursively computed with the following equations:

$$a_{j,k} = \sum_m h_0(m - 2k) a_{j+1,m} \quad (2.65)$$

$$d_{j,k} = \sum_m h_1(m - 2k) a_{j+1,m} \quad (2.66)$$

with:

$$m = 2k, 2k + 1, 2k + 2, 2k + N - 1 \quad (2.67)$$

The filter banks of Fig. 2.17 can be used, with $LP(z)$ corresponding to (2.65) and $HP(z)$ corresponding to (2.66). The series of values $h_0(m - 2k)$ and $h_1(m - 2k)$ are filter impulse responses.

2.3.2.1 Wavelet Properties

If the integer translates of $\varphi(t)$ are orthogonal the corresponding wavelet has the following properties:

- The wavelet is orthogonal to the scaling function at the same scale if and only if:

$$h_1(n) = \pm(-1)^n h_0(L - n) \quad (2.68)$$

where L is an arbitrary odd integer.

Orthogonal at the same scale is:

$$\int \varphi(t - n) \psi(t - m) dt = 0 \quad (2.69)$$

- If the wavelet is orthogonal to the scaling function at the same scale then:

$$\sum_n h_0(n) h_1(n - 2k) = 0 \quad (2.70)$$

- Also,

$$\sum_n h_1(n) = H_1(0) = 0 \quad (2.71)$$

$$|H_1(\omega)| = |H_0(\omega + \pi)| \quad (2.72)$$

$$|H_0(\omega)|^2 + |H_1(\omega)|^2 = 2 \quad (2.73)$$

2.3.3 Examples

Coming back to the multiresolution equation, let us comment some solutions.

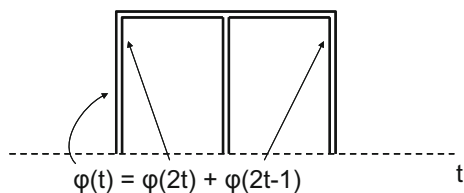
2.3.3.1 For $M = 2$

The necessary conditions imply that:

$$h_0(0) + h_0(1) = \sqrt{2} \quad (2.74)$$

$$h_0(0) = h_0(1) \quad (2.75)$$

Fig. 2.26 Haar scaling functions



Then:

$$h_0(0) = \frac{1}{\sqrt{2}}, \quad h_0(1) = \frac{1}{\sqrt{2}} \quad (2.76)$$

and the solution is given by the Haar scaling function.

Figure 2.26 shows how $\varphi(t)$ can be obtained with $\varphi(2t)$ and $\varphi(2t - 1)$ according with Eq. (2.37) and the coefficients (2.76). Notice that $\varphi(2t)$ and $\varphi(2t - 1)$ (or in general $\varphi(2t - k)$) are orthogonal since they do not overlap.

The Haar wavelets are obtained with Eq. (2.61) using:

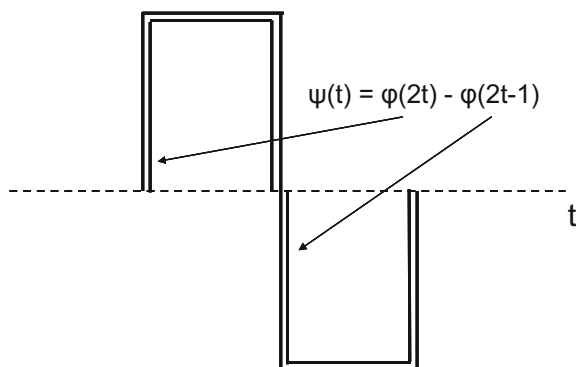
$$h_1(0) = \frac{1}{\sqrt{2}}, \quad h_1(1) = -\frac{1}{\sqrt{2}} \quad (2.77)$$

The values of $h_1(0)$ and the $h_1(1)$ are easily derived from orthogonality conditions. Figure 2.27 shows how $\psi(t)$ can be obtained with $\varphi(2t)$ and $\varphi(2t - 1)$ according with Eq. (2.61) and the coefficients (2.77).

The Haar wavelets have the following properties:

- Orthogonal
- Compact time-domain support
- The scaling function is symmetric
- The wavelet function is anti-symmetric
- The wavelet function has only one vanishing moment

Fig. 2.27 Haar wavelet and scaling functions



Actually, the Haar wavelet is the only one having the first three properties.

2.3.3.2 For $M = 3$

The necessary conditions imply that:

$$h_0(0) + h_0(1) + h_0(2) = \sqrt{2} \quad (2.78)$$

$$h_0(0) + h_0(2) = h_0(1) \quad (2.79)$$

A solution is provided by the triangle scaling function, with the following values:

$$h_0(0) = \frac{1}{2\sqrt{2}}, \quad h_0(1) = \frac{1}{\sqrt{2}}, \quad h_0(2) = \frac{1}{2\sqrt{2}} \quad (2.80)$$

Figure 2.28 shows how $\varphi(t)$ can be obtained with $\varphi(2t)$, $\varphi(2t - 1)$, and $\varphi(2t - 2)$ according with Eq. (2.37) and the coefficients (2.80). Notice that $\varphi(2t)$ and $\varphi(2t - 1)$ (or in general $\varphi(2t - k)$) are *not* orthogonal.

The triangle wavelets are obtained with Eq. (2.61) using:

$$h_1(0) = -\frac{1}{2\sqrt{2}}, \quad h_1(1) = \frac{1}{\sqrt{2}}, \quad h_1(2) = -\frac{1}{2\sqrt{2}} \quad (2.81)$$

As in the previous example, Fig. 2.29 shows how $\psi(t)$ can be obtained with $\varphi(2t)$, $\varphi(2t - 1)$, and $\varphi(2t - 2)$ according with Eq. (2.61) and the coefficients of the triangle wavelet (2.81).

Triangle scaling functions have compact support and are symmetric, but they are not orthogonal.

Fig. 2.28 Triangle scaling functions

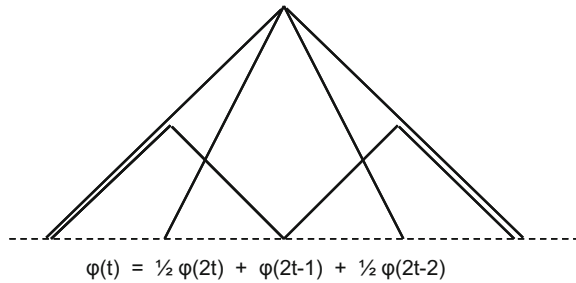
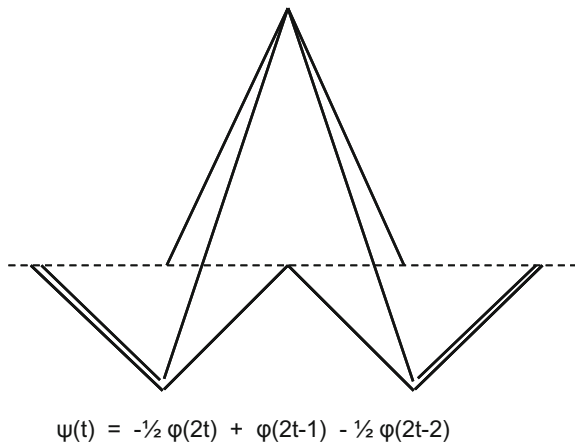


Fig. 2.29 Triangle wavelet and scaling functions



2.3.3.3 For $M = 4$

The necessary conditions, for the case that integer translates of $\varphi(t)$ are orthogonal, imply that:

$$h_0(0) + h_0(1) + h_0(2) + h_0(3) = \sqrt{2} \quad (2.82)$$

$$h_0(0) + h_0(2) = h_0(1) + h_0(3) \quad (2.83)$$

$$h_0^2(0) + h_0^2(1) + h_0^2(2) + h_0^2(3) = 1 \quad (2.84)$$

A solution was found by Daubechies, with the following values

$$\begin{aligned} h_0(0) &= \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_0(1) = \frac{3+\sqrt{3}}{4\sqrt{2}}, \\ h_0(2) &= \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_0(3) = \frac{1-\sqrt{3}}{4\sqrt{2}} \end{aligned} \quad (2.85)$$

Further details on the Daubechies scaling functions and wavelets will be given in the next section. Scaling functions have compact support and are orthogonal, but they are not symmetric.

2.3.4 Shannon Wavelets

The Shannon scaling function has a well-known expression, related to the frequency response of a brickwall filter:

$$\varphi(t) = \text{sinc}(\pi t) = \frac{\sin(\pi t)}{\pi t} \quad (2.86)$$

with value 1 for $t = 0$.

The corresponding LP(z) coefficients are the following:

$$h_0(0) = \frac{1}{2} \quad (2.87)$$

$$h_0(2n) = 0, n \neq 0 \quad (2.88)$$

$$h_0(2n + 1) = \frac{(-1)^n \sqrt{2}}{(2n + 1)\pi}, n \neq 0 \quad (2.89)$$

Cardinal functions have zero value at integer t values (except at $t = 0$). The Shannon scaling function $\varphi(t)$ is a cardinal function and this makes it useful for interpolation (actually it is the function used by the Shannon sampling theorem to reconstruct a signal from its samples).

- The Shannon scaling functions are orthogonal
- The Shannon wavelet is:

$$\psi(t) = 2\varphi(2t) - \varphi(t) = \frac{\sin(2\pi t) - \sin(\pi t)}{\pi t} \quad (2.90)$$

- The Shannon wavelet has infinite number of vanishing moments

Both $\varphi(t)$ and $\psi(t)$ are symmetric, with infinite time support and slow decay (Fig. 2.30).

Program 2.14 Display of Shannon scaling function and wavelet

```
% Display of Shannon scaling function and wavelet
t=-10:0.01:10; %time vector
phi=sinc(t) ; %the scaling function
psi=(2*sinc(2*t))-sinc(t); %the wavelet
figure(1)
subplot(2,1,1)
plot(t,phi,'k');
axis([-10 10 -0.4 1.2]);
xlabel('t')
title('Shannon scaling function');
subplot(2,1,2)
plot(t,psi,'k');
axis([-10 10 -1.2 1.2]);
xlabel('t')
title('Shannon wavelet');
```

The Haar and the Shannon scaling functions are Fourier duals. The Haar wavelet system is good for time localization, but not for frequency localization. The Shannon wavelet system is the opposite.

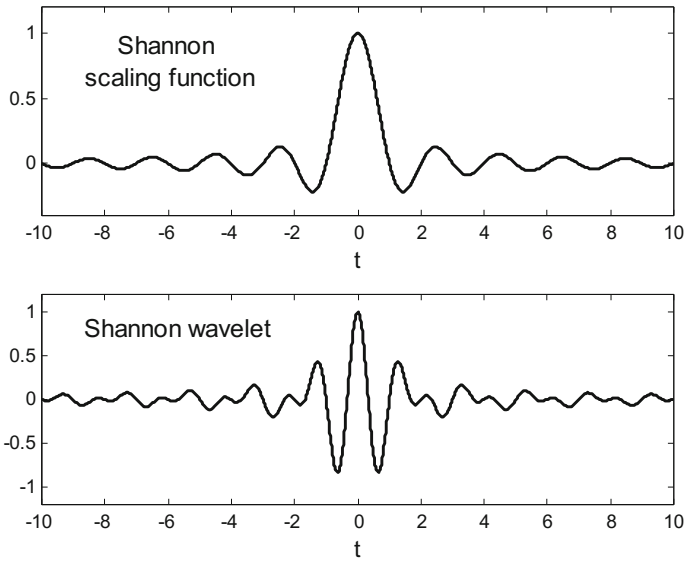


Fig. 2.30 Shannon scaling function and wavelet

2.3.5 Splines

Now it is convenient to introduce splines. They are important in signal processing, for instance for interpolation applications, and in the context of wavelets.

Suppose several consecutive times:

$$a = t_0 < t_1 < t_2 < \dots < t_{k-1} = b \quad (2.91)$$

A spline $l(t)$ is a piecewise polynomial function, such that:

$$l(t) = P_0(t), \quad t_0 \leq t < t_1, \quad (2.92)$$

$$l(t) = P_1(t), \quad t_1 \leq t < t_2, \quad (2.93)$$

$$l(t) = P_{k-2}(t), \quad t_{k-2} \leq t < t_{k-1}, \quad (2.94)$$

where $P_i(t)$ are polynomials.

The points t_i are called 'knots'.

There are many types of splines. For instance, if the knots are equidistant the spline is 'uniform', otherwise is 'non-uniform'.

If all $P_i(t)$ have degree at most n , then the spline is of degree n .

For a given knot t_i , $P_{i-1}(t)$ and $P_i(t)$ may share a certain number r_i of common derivatives, which indicates how smooth is the spline at t_i .

‘Natural’ splines have zero second derivatives at a and b . ‘Interpolating’ splines should have a given set of data values.

2.3.5.1 B-Splines

B-splines (basis splines) are bell-shaped functions generated with the $(n + 1)$ -fold convolution of a rectangular pulse β^0 :

$$\beta^0(t) = \begin{cases} 1, & -1/2 < t < 1/2 \\ 1/2, & |t| = 1/2 \\ 0, & \text{otherwise} \end{cases} \quad (2.95)$$

$$\beta^n(t) = \underbrace{\beta^0 * \beta^0 * \dots * \beta^0}_{n+1 \text{ times}}(t) \quad (2.96)$$

where the asterisk (*) means convolution.

The Fourier transform of $\beta^n(t)$ is:

$$B^n(\omega) = \left(\frac{\sin(\omega/2)}{\omega/2} \right)^{n+1} \quad (2.97)$$

B-splines satisfy the MAE equation, so they are scaling functions. In particular:

- For $n = 1$:

$$\beta^1(t) = \frac{1}{2} (\beta^1(2t + 1) + 2\beta^1(2t) + \beta^1(2t - 1)) \quad (2.98)$$

That corresponds to:

$$h_0(0) = \frac{1}{2\sqrt{2}}, h_0(1) = \frac{1}{\sqrt{2}}, h_0(2) = \frac{1}{2\sqrt{2}} \quad (2.99)$$

The B-spline of degree 1 is the triangle scaling function seen before.

- For $n = 2$:

$$\beta^1(t) = \frac{1}{4} (\beta^2(2t + 1) + 3\beta^2(2t) + 3\beta^2(2t - 1) + \beta^2(2t - 2)) \quad (2.100)$$

That corresponds to:

$$h_0(0) = \frac{1}{4\sqrt{2}}, h_0(1) = \frac{3}{4\sqrt{2}}, h_0(2) = \frac{3}{4\sqrt{2}}, h_0(3) = \frac{1}{4\sqrt{2}} \quad (2.101)$$

- For $n = 3$ we have the popular cubic B-spline, with:

$$\begin{aligned} h_0(0) &= \frac{1}{16\sqrt{2}}, h_0(1) = \frac{1}{4\sqrt{2}}, \\ h_0(2) &= \frac{3}{8\sqrt{2}}, h_0(3) = \frac{1}{4\sqrt{2}}, h_0(4) = \frac{1}{16\sqrt{2}} \end{aligned} \quad (2.102)$$

- In general:

$$\beta^n(t) = \frac{1}{2^n} \sum_{j=0}^{n+1} \binom{n+1}{j} \beta^n(2t + \frac{n}{2} - j), \quad n \text{ even} \quad (2.103)$$

$$\beta^n(t) = \frac{1}{2^n} \sum_{j=0}^{n+1} \binom{n+1}{j} \beta^n(2t + \frac{n+1}{2} - j), \quad n \text{ odd} \quad (2.104)$$

The B-splines are *not* orthogonal over integer translations.

Schoenberg has shown, in 1946, that all uniform splines with unit spacing are uniquely characterized in terms of a B-spline expression:

$$f(t) = \sum_l c(l) \beta^n(t - l) \quad (2.105)$$

with l an integer and $\beta^n(t)$ the central B-spline of degree n .

Influential authors prefer a slightly different definition of B-splines, being uniform splines starting from:

$$b_i^0(t) = \begin{cases} 1, & t_i \leq t < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (2.106)$$

and then the others can be obtained with a recursive relation:

$$b_i^n(t) = \frac{t - t_i}{t_{i+n} - t_i} b_{i,n-1}(t) + \frac{t_{i+n+1} - t}{t_{i+n+1} - t_{i+1}} b_{i+1,n-1}(t) \quad (2.107)$$

Based on this recursion, the Program 2.15 computes the B-splines corresponding to $i = 0$ with degrees $n = 0$ to 5. Figure 2.31 shows the B-splines. Notice how the support of the B-splines grows as degree increases (actually this support is $n + 1$). For a B-spline of degree n , the program only has to consider the previously computed B-spline of degree $n - 1$, and a right-shifted version (by 1 second) of this B-spline of degree $n - 1$.

The difference between the b and the β versions of B-splines is that the centre of β versions is the origin, and the b versions are shifted to the right.

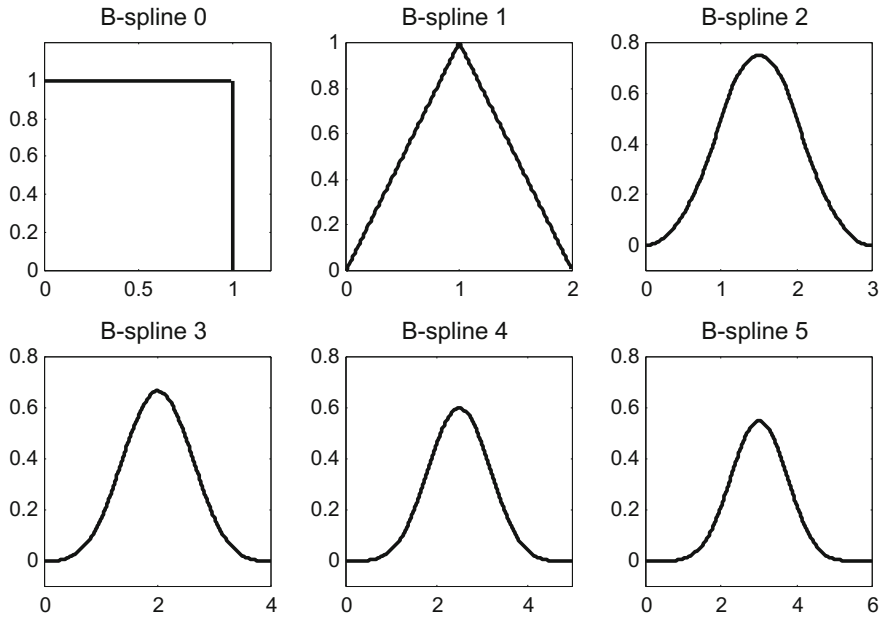


Fig. 2.31 B-splines of degrees 0 to 5

Program 2.15 Display of B-splines

```
% Display of B-splines
%space for splines
nz=100; tiv=1/nz;
b=zeros(4,5*nz); %space for splines
% the box spline (n=0)
b0=ones(1,nz);
% spline n=1
N=1;
b1L=(0:tiv:(N-tiv)).*b0; b1D=(N:-tiv:tiv).*b0;
b(N,1:((N+1)*nz))=[b1L b1D];
% splines 2..5
for N=2:5,
    bL=(0:tiv:(N-tiv)).*b(N-1,1:N*nz);
    bD=(N:-tiv:tiv).*b(N-1,1:N*nz);
    b(N,1:((N+1)*nz))=(1/N)*[bL(1:nz) bL((nz+1):N*nz)+...
    bD(1:((N-1)*nz)) bD(((N-1)*nz)+1):N*nz]];
end;
figure(1)
subplot(2,3,1)
t=(0:tiv:(1-tiv)); plot(t,b0,'k',[1 1],[1 0],'k');
title('B-spline 0');
axis([0 1.2 0 1.2]);
```

```

subplot(2,3,2)
t=(0:tiv:(2-tiv)); plot(t,b(1,1:2*nz), 'k');
title('B-spline 1');
axis([0 2 0 1]);
subplot(2,3,3)
t=(0:tiv:(3-tiv));plot(t,b(2,1:3*nz), 'k');
title('B-spline 2');
axis([0 3 -0.1 0.8]);
subplot(2,3,4)
t=(0:tiv:(4-tiv));plot(t,b(3,1:4*nz), 'k');
title('B-spline 3');
axis([0 4 -0.1 0.8]);
subplot(2,3,5)
t=(0:tiv:(5-tiv));plot(t,b(4,1:5*nz), 'k');
title('B-spline 4');
axis([0 5 -0.1 0.8]);
subplot(2,3,6)
t=(0:tiv:(6-tiv));plot(t,b(5,1:6*nz), 'k');
title('B-spline 5');
axis([0 6 -0.1 0.8]);

```

2.4 Orthogonal Wavelets

Orthogonal scaling functions can be obtained by design, or by orthogonalization of non-orthogonal scaling functions.

Given a non-orthogonal scaling function, with Fourier transform $\Phi(\omega)$ a orthogonal scaling function with Fourier transform $\Phi_{on}(\omega)$ can be obtained as follows:

$$\Phi_{on}(\omega) = \frac{\Phi(\omega)}{\sqrt{E(\omega)}} \quad (2.108)$$

with:

$$E(\omega) = \sum_k |\Phi(\omega + 2\pi k)|^2 \quad (2.109)$$

where $E(\omega)$ is an Euler-Frobenius polynomial.

To see that $\Phi_{on}(\omega)$ is orthogonal, recall from Sect. 2.3.2 the necessary condition given in frequency domain, that in this case reads as:

$$\sum_n |\Phi_{on}(\omega + 2\pi k)|^2 = 1 \quad (2.110)$$

A version of the frequency domain version of the MAE, for the orthogonal scaling function, is:

$$\Phi_{on}(\omega) = \frac{1}{\sqrt{2}} H_0\left(\frac{\omega}{2}\right) \Phi_{on}\left(\frac{\omega}{2}\right) \quad (2.111)$$

Then, the corresponding LP filter is:

$$H_0(\omega) = \sqrt{2} \frac{\Phi_{on}(2\omega)}{\Phi_{on}(\omega)} \quad (2.112)$$

Being filters orthogonal, then:

$$H_1(\omega) = -e^{-j\omega} H_0^*(e^{j(\omega+\pi)}) \quad (2.113)$$

And, by a frequency domain version of (2.61):

$$\Psi(\omega) = \frac{1}{\sqrt{2}} H_1\left(\frac{\omega}{2}\right) \Phi_{on}\left(\frac{\omega}{2}\right) \quad (2.114)$$

The coefficients of the LP or HP filters can be obtained with:

$$h_k = \sqrt{2} \frac{1}{2\pi} \int_0^{2\pi} H(\omega) e^{jk\omega} d\omega \quad (2.115)$$

2.4.1 Meyer Wavelet

The Meyer wavelet was one of the first wavelets, mid 1980s. Meyer wavelet can be viewed as an improvement with respect to the Shannon wavelet. Instead of a brickwall shape of $\Phi(\omega)$ with vertical transitions, discontinuities and transitions are smoothed to improve the time-domain localization.

The scaling function proposed by Meyer is, in the frequency domain:

$$\Phi(\omega) = \begin{cases} 1, & |\omega| \leq \frac{2\pi}{3} \\ \cos\left[\frac{\pi}{2} v\left(\frac{3}{2\pi} |\omega| - 1\right)\right], & \frac{2\pi}{3} \leq |\omega| \leq \frac{4\pi}{3} \\ 0, & \text{otherwise} \end{cases} \quad (2.116)$$

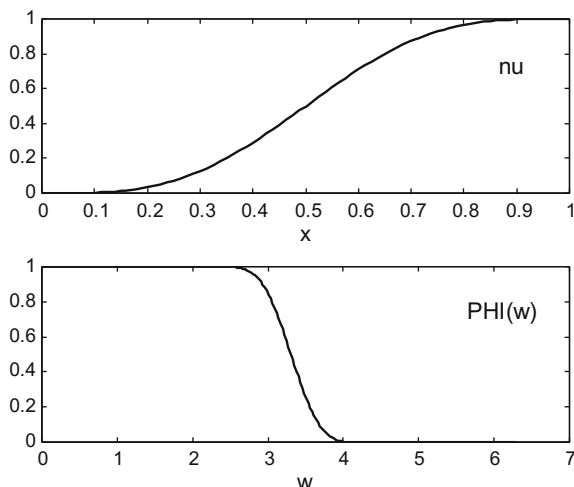
where $v(x)$ is a smooth polynomial interpolating function that:

$$\begin{aligned} v(x) &= \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x \geq 1 \end{cases} \\ v(x) + v(1-x) &= 1 \end{aligned} \quad (2.117)$$

For example:

$$v(x) = x^4 (35 - 84x + 70x^2 - 20x^3), \text{ with } x \in [0, 1] \quad (2.118)$$

Fig. 2.32 The $v(x)$ function and the Meyer $\Phi(\omega)$



The scaling functions $\varphi(t - k)$ form an orthonormal basis.

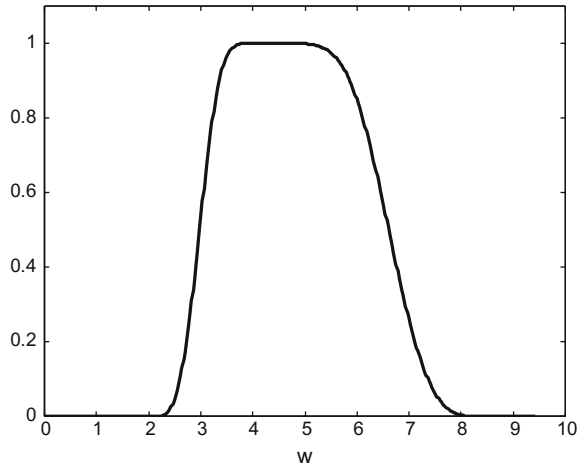
The frequency response of the corresponding LP(z) filter is:

$$H_0(\omega) = \sum_k \Phi(2(\omega + 2\pi k)) \quad (2.119)$$

Figure 2.32 depicts $v(x)$ and (half) $\Phi(\omega)$ as given by Eqs. (2.117) and (2.116). The figure has been generated with the Program 2.16. Notice the difference with respect to a Sannon's $\Phi(\omega)$ brickwall shape.

Program 2.16 Display of Meyer nu function and 1/2 PHI(w) (right side)

```
% Display of Meyer nu function and 1/2 PHI(w) (right side)
Np=120; %samples per pi
w=0:(pi/Np):(2*pi); %frequencies
L=length(w);
n23=1+((2*Np)/3); %samples for 0..2/3 pi
n43=1+((4*Np)/3); %samples for 0..4/3 pi
wc=w(n23:n43); %central zone
x=((3/(2*pi))*abs(wc))-1;
nu=35-(84*x)+(70*(x.^2))-(20*(x.^3));
nu=(x.^4).*nu;
PHI=zeros(1,L);
PHI(1:(n23-1))=1;
PHI(n23:n43)=cos((pi/2)*nu);
figure(1)
subplot(2,1,1)
plot(x,nu,'k');
xlabel('x'); title('nu');
subplot(2,1,2)
plot(w,PHI,'k');
xlabel('w'); title('PHI(w)');
```

Fig. 2.33 The Meyer $\Psi(\omega)$ 

The wavelet is, in the frequency domain:

$$\Psi(\omega) = e^{j\omega/2} (\Phi(\omega - 2\pi) + \Phi(\omega + 2\pi)) \Phi\left(\frac{\omega}{2}\right) \quad (2.120)$$

That is:

$$\Psi(\omega) = \begin{cases} e^{j(\omega/2)} \sin\left[\frac{\pi}{2} v\left(\frac{3}{2\pi}|\omega| - 1\right)\right], & \frac{2\pi}{3} \leq |\omega| \leq \frac{4\pi}{3} \\ e^{j(\omega/2)} \cos\left[\frac{\pi}{2} v\left(\frac{3}{4\pi}|\omega| - 1\right)\right], & \frac{4\pi}{3} \leq |\omega| \leq \frac{8\pi}{3} \\ 0, & \text{otherwise} \end{cases} \quad (2.121)$$

Figure 2.33, which has been generated with the Program 2.17, depicts the $\Psi(\omega)$ corresponding to $v(x)$ as in (2.117). Notice how we extended the range of frequencies above 2π . The plot shows clearly the pass-band nature of the wavelet.

Program 2.17 Display of Meyer 1/2 PSI(w) (right side)

```
% Display of Meyer 1/2 PSI(w) (right side)
Np=120; %samples per pi
w=0:(pi/Np):(3*pi); %frequencies
L=length(w);
n23=1+((2*Np)/3); %samples for 0..2/3 pi
n43=1+((4*Np)/3); %samples for 0..4/3 pi
n83=1+((8*Np)/3); %samples for 0..8/3 pi
wc1=w(n23:n43); %zone 1
wc2=w((n43+1):n83); %zone 2
x1=((3/(2*pi))*abs(wc1))-1;
x2=((3/(4*pi))*abs(wc2))-1;
nu1=35-(84*x1)+(70*(x1.^2))-(20*(x1.^3));
nu1=(x1.^4).*nu1;
```

```

nu2=35-(84*x2)+(70*(x2.^2))-(20*(x2.^3));
nu2=(x2.^4).*nu2;
PSI=zeros(1,L);
PSI(1:(n23-1))=0;
PSI(n23:n43)=(exp(j*wc1/2).*sin((pi/2)*nu1));
PSI((n43+1):n83)=(exp(j*wc2/2).*cos((pi/2)*nu2));
figure(1)
plot(w,abs(PSI),'k');
xlabel('w'); title('PSI(w)');
axis([0 10 0 1.1]);

```

Both the scaling function and the wavelet are symmetric. They have infinite time support, but faster decaying than the *sinc* function, and they are infinitely many times differentiable.

Figure 2.34 shows the Meyer scaling function $\varphi(t)$ corresponding to $v(x)$ as in (2.117). The figure has been generated with the Program 2.18.

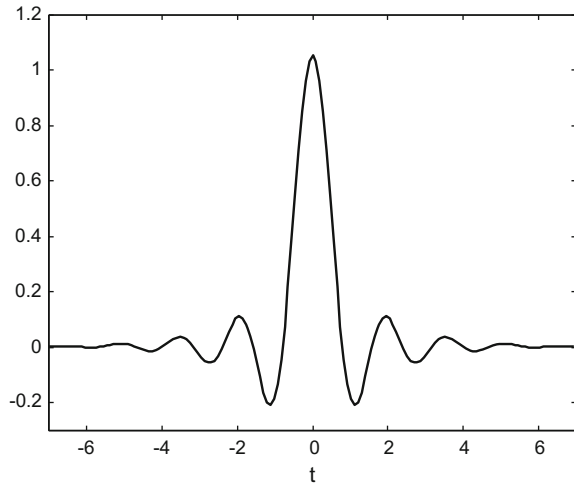
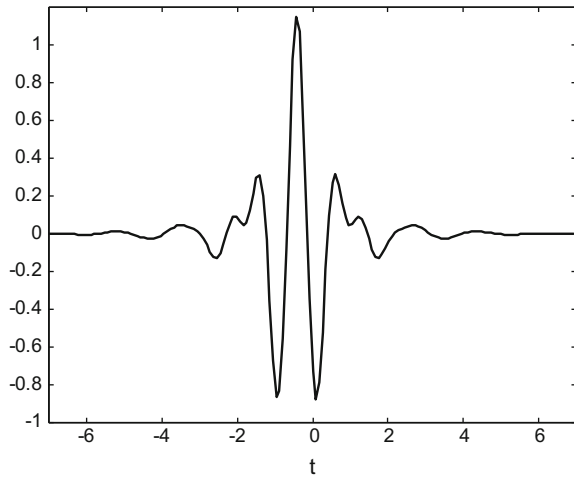
Program 2.18 Display of Meyer scaling function phi(t)

```

% Display of Meyer scaling function phi(t)
Np=120; %samples per pi
Nc=18; %number of data cycles
%frequencies extended for better time resolution:
w=0:(pi/Np):(Nc*pi);
L=length(w);
n23=1+((2*Np)/3); %samples for 0..2/3 pi
n43=1+((4*Np)/3); %samples for 0..4/3 pi
wc=w(n23:n43); %central zone
x=((3/(2*pi))*abs(wc))-1;
nu=35-(84*x)+(70*(x.^2))-(20*(x.^3));
nu=(x.^4).*nu;
PHI=zeros(1,L);
PHI(1:(n23-1))=1;
PHI(n23:n43)=cos((pi/2)*nu);
yphi=ifftshift(ifft(PHI));
ct=L/Np; %scaling according with sampling rate
phi=ct*yphi;
figure(1)
tiv=100/(Nc*Np);
tx=(-50):tiv:(50);
t=tx*(Np/60);
Mt=floor(L/2); My=1+(Mt); Mw=4*(Nc+1);
plot(t((Mt-Mw):(Mt+Mw)),real(phi((My-Mw):(My+Mw))), 'k');
axis([-7 7 -0.3 1.2]);
xlabel('t'); title('phi(t)');

```

Figure 2.35 shows the Meyer wavelet $\psi(t)$ corresponding to $v(x)$ as in (2.117). The figure has been generated with the Program 2.19.

Fig. 2.34 The Meyer scaling function**Fig. 2.35** The Meyer wavelet**Program 2.19** Display of Meyer wavelet $\psi(t)$

```
% Display of Meyer wavelet psi(t)
Np=120; %samples per pi
Nc=19; %number of data cycles
%frequencies extended for better time resolution:
w=0:(pi/Np):(Nc*pi);
L=length(w);
n23=1+((2*Np)/3); %samples for 0..2/3 pi
n43=1+((4*Np)/3); %samples for 0..4/3 pi
n83=1+((8*Np)/3); %samples for 0..8/3 pi
wc1=w(n23:n43); %zone 1
wc2=w((n43+1):n83); %zone 2
```

```

x1=((3/(2*pi))*abs(wc1))-1;
x2=((3/(4*pi))*abs(wc2))-1;
nu1=35-(84*x1)+(70*(x1.^2))-(20*(x1.^3));
nu1=(x1.^4).*nu1;
nu2=35-(84*x2)+(70*(x2.^2))-(20*(x2.^3));
nu2=(x2.^4).*nu2;
PSI=zeros(1,L);
PSI(1:(n23-1))=0;
PSI(n23:n43)=(exp(j*wc1/2).*sin((pi/2)*nu1));
PSI((n43+1):n83)=(exp(j*wc2/2).*cos((pi/2)*nu2));
ypsi=ifftshift(ifft(PSI));
ct=L/Np; %scaling according with sampling rate
psi=ct*ypsi;
figure(1)
tiv=100/(Nc*Np);
tx=(-50):tiv:(50);
t=tx*(Np/60);
Mt=floor(L/2); My=1+(Mt); Mw=4*(Nc+1);
plot(t((Mt-Mw):(Mt+Mw)),real(psi((My-Mw):(My+Mw))), 'k');
axis([-7 7 -1 1.2]);
xlabel('t'); title('psi(t)');

```

2.4.2 Battle-Lemarié Wavelet

The Battle-Lemarié scaling functions are obtained by orthogonalization of B-splines. In this case one can use the following relationship:

$$B^{(2N+1)}(\omega) = \sum_k |B^N(\omega + 2k\pi)|^2 \quad (2.122)$$

where $B^{(2N+1)}$ is the DFT of the sampled version of the continuous time $b^{(2N+1)}$ B-spline.

The orthogonal scaling function is obtained with:

$$\Phi_{on}(\omega) = \frac{B^N(\omega)}{\sqrt{B^{(2N+1)}(\omega)}} \quad (2.123)$$

Using the MAE equation in the frequency domain, we can write:

$$\begin{aligned}
H_0(\omega) &= \sqrt{2} \frac{\Phi_{on}(2\omega)}{\Phi_{on}(\omega)} = \sqrt{2} \left(\frac{\sin \omega}{\omega} \frac{(\omega/2)}{\sin(\omega/2)} \right)^{N+1} \frac{\sqrt{B^{(2N+1)}(\omega)}}{\sqrt{B^{(2N+1)}(2\omega)}} = \\
&= \sqrt{2} (\cos(\omega/2))^{N+1} \frac{\sqrt{B^{(2N+1)}(\omega)}}{\sqrt{B^{(2N+1)}(2\omega)}}
\end{aligned} \quad (2.124)$$

The Battle-Lemarié scaling functions and the corresponding wavelets have infinite time-domain support, but decay exponentially fast.

For example, in the case of the triangular B-spline $B^1(\omega)$:

$$B^{(3)}(\omega) = \frac{2}{3} + \frac{1}{3} \cos(\omega) = 1 - \frac{2}{3} \sin^2\left(\frac{\omega}{2}\right) = \frac{1}{3} (1 + 2 \cos^2\left(\frac{\omega}{2}\right)) \quad (2.125)$$

Denote:

$$V = \frac{1}{3} (1 + 2 \cos^2\left(\frac{\omega}{2}\right)) \quad (2.126)$$

Then:

$$\Phi_{on}(\omega) = \frac{\sin^2(\omega/2)}{(\omega/2)^2 \sqrt{V}} = \sqrt{3} \frac{4 \sin^2(\omega/2)}{\omega^2 \sqrt{1 + 2 \cos^2(\omega/2)}} \quad (2.127)$$

And,

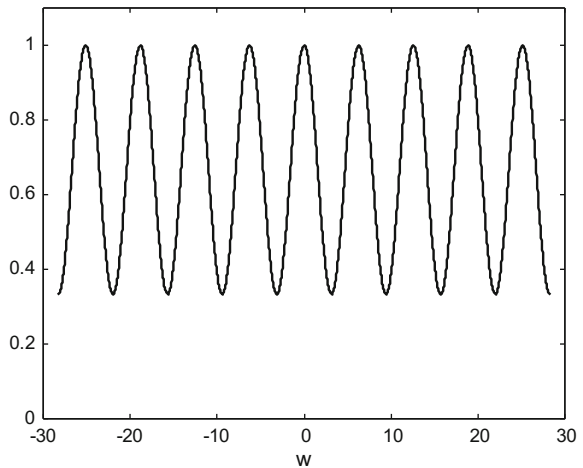
$$H_0(\omega) = \sqrt{2} \frac{\cos^2(\omega/2) \sqrt{V}}{\sqrt{1 - (2/3) \sin^2(\omega)}} \quad (2.128)$$

$$H_1(\omega) = -e^{-j\omega} H_0(\omega + \pi) \quad (2.129)$$

$$\Psi(\omega) = -e^{-j(\omega/2)} \frac{\sin^4(\omega/4)}{(\omega/4)^2 \sqrt{V}} \sqrt{\frac{1 - (2/3) \cos^2(\omega/4)}{1 - (2/3) \sin^2(\omega/4)}} \quad (2.130)$$

Figure 2.36 shows V.

Fig. 2.36 The values of V along frequency



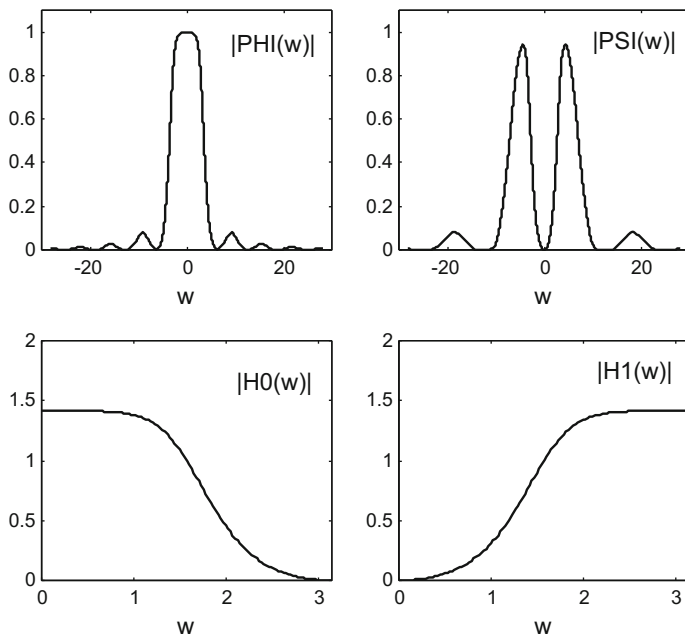


Fig. 2.37 The Battle-Lemarié first order scaling function $\Phi(\omega)$, wavelet $\Psi(\omega)$, and magnitude frequency response of filters H_0 and H_1

Figure 2.37 shows in frequency domain the Battle-Lemarié scaling function, $H_0(\omega)$, $H_1(\omega)$, and wavelet corresponding to the triangular B-spline $B^1(\omega)$. This figure and the previous one have been generated with the Program 2.20.

Program 2.20 Display of Battle-Lemarié $\Phi(w)$, $H_0(w)$, $H_1(w)$ and $\Psi(w)$

```
% Display of Battle-Lemarié \{'e\}  $\Phi(w)$ ,  $H_0(w)$ ,  $H_1(w)$  and  $\Psi(w)$ 
Np=120; %samples per pi
w=(-9*pi):(pi/Np):(9*pi); %frequencies
L=length(w); M=ceil(L/2);
w(M)=0.000001; %to avoid w=0 and division by zero
aux1=(cos(w/2)).^2;
V1=(1+(2*aux1))/3;
PHI=((sin(w/2)).^2)/(((w/2).^2).*sqrt(V1));
aux2=(sin(w)).^2;
H0=(sqrt(2)*aux1.*sqrt(V1))./sqrt(1-(2*aux2/3));
aux1=(cos((w+pi)/2)).^2;
V2=(1+(2*aux1))/3;
aux2=(sin(w+pi)).^2;
aux=sqrt(2)*aux1.*sqrt(V2)./sqrt(1-(2*aux2/3));
H1=(-exp(-j*w)).*aux;
aux0=(exp(-j*(w/2))).*((sin(w/4)).^4)/(((w/4).^2).*sqrt(V1));
aux1=(1-((2/3)*((cos(w/4)).^2)))/(1-((2/3)*((sin(w/4)).^2)));
```

```

PSI=aux0.*sqrt(aux1);
figure(1)
plot(w,V1,'k');
xlabel('w'); title('V');
axis([-30 30 0 1.1]);
figure(2)
subplot(2,2,1)
plot(w,abs(PHI),'k');
xlabel('w'); title(' | PHI (w) | ');
axis([-30 30 0 1.1]);
subplot(2,2,2)
plot(w,abs(PSI),'k');
xlabel('w'); title(' | PSI (w) | ');
axis([-30 30 0 1.1]);
subplot(2,2,3)
plot(w(M:(M+Np)),abs(H0(M:(M+Np))), 'k');
xlabel('w'); title(' | H0 (w) | ');
axis([0 pi 0 2]);
subplot(2,2,4)
plot(w(M:(M+Np)),abs(H1(M:(M+Np))), 'k');
xlabel('w'); title(' | H1 (w) | ');
axis([0 pi 0 2]);

```

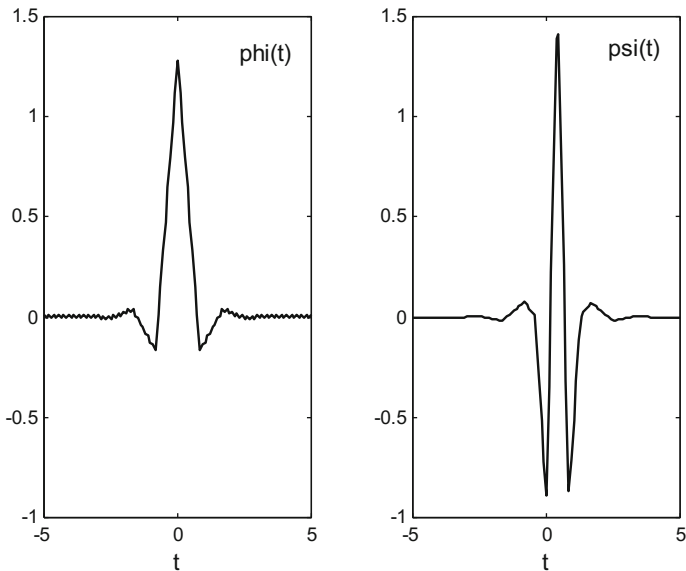


Fig. 2.38 The Battle-Lemarié first order scaling function $\varphi(t)$

Figure 2.38 shows in time domain the Battle-Lemarié scaling function and wavelet corresponding to the triangular B-spline $B^1(\omega)$. The figure has been generated with the Program 2.21.

Program 2.21 Display of Battle-Lemarié $\phi(t)$, and $\psi(t)$

```
% Display of Battle-Lemarié \phi(t), and \psi(t)
Np=120; %samples per pi
Nc=18; %number of data cycles
wiv=pi/Np; %frequency interval
%frequencies extended for better time resolution:
w=wiv:wiv:(Nc*pi)+wiv;
L=length(w);M=ceil(L/2);
w(M)=0.000001; %to avoid w=0 and division by zero
V=(1+(2*(cos(w/2)).^2))/3;
PHI=((sin(w/2)).^2)/((w/2).^2).*sqrt(V);
aux0=(exp(-j*(w/2)).*(sin(w/4)).^4)/((w/4).^2).*sqrt(V);
aux1=(1-((2/3)*(cos(w/4)).^2))/(1-((2/3)*(sin(w/4)).^2));
PSI=aux0.*sqrt(aux1);
yphi=ifftshift(ifft(PHI));
ct=L/Np; %scaling according with sampling rate
phi=ct*yphi;
ypsi=ifftshift(ifft(PSI));
psi=ct*yphi;
figure(1)
tiv=100/(Nc*Np);
tx=(-50):tiv:(50);
t=tx*(Np/60);
Mt=floor(L/2); My=1+(Mt); Mw=4*(Nc+1);
subplot(1,2,1)
plot(t((Mt-Mw):(Mt+Mw)),real(phi((My-Mw):(My+Mw))), 'k');
axis([-5 5 -1 1.5]);
xlabel('t'); title('phi(t)');
subplot(1,2,2)
plot(t((Mt-Mw):(Mt+Mw)),real(psi((My-Mw):(My+Mw))), 'k');
axis([-5 5 -1 1.5]); xlabel('t'); title('psi(t)');
```

2.4.3 Daubechies Wavelets

2.4.3.1 Preliminaries

A convenient way to introduce the Daubechies wavelets is by considering an orthogonal scaling function and its corresponding filter $LP(z)$. This filter is said to be K -regular if $H_0(z)$ has K zeros at $z = \exp(j\pi)$, so:

$$H_0(z) = \left(\frac{1 + z^{-1}}{2} \right)^K Q(z) \quad (2.131)$$

where $Q(z)$ has no poles or zeros at $z = \exp(j\pi)$.

Suppose that $H_0(z)$ is polynomial with degree $N - 1$. It can be shown that:

$$1 \leq K \leq \frac{N}{2} \quad (2.132)$$

The Daubechies design aims at maximum regularity. To see the implications consider the moments of $\psi(t)$ and $\varphi(t)$:

$$m_0(k) = \int t^k \varphi(t) dt \quad (2.133)$$

$$m_1(k) = \int t^k \psi(t) dt \quad (2.134)$$

and the discrete moments:

$$\mu_0(k) = \sum_n n^k h_0(n) \quad (2.135)$$

$$\mu_1(k) = \sum_n n^k h_1(n) \quad (2.136)$$

It can be shown that the filter $H_0(z)$ is K -regular if and only if the following equivalent statements are true:

$$\forall \mu_1(k) = 0, \quad k = 0, 1, 2, \dots, K - 1 \quad (2.137)$$

$$\forall m_1(k) = 0, \quad k = 0, 1, 2, \dots, K - 1 \quad (2.138)$$

Recall from Sect. 2.3 that one of the necessary conditions for $\varphi(t)$ to be a solution of the MAE, if the integer translates of $\varphi(t)$ are orthogonal, is that:

$$|H_0(\omega)|^2 + |H_0(\omega + \pi)|^2 = 2 \quad (2.139)$$

2.4.3.2 The Daubechies Wavelets

Daubechies wavelets are an important alternative for many practical applications. These wavelets allow for localization in both time and frequency.

Daubechies obtained orthonormal wavelets with compact support and the maximum number of vanishing moments, with:

$$H_0(\omega) = \sqrt{2} \left(\frac{1 + e^{-j\omega}}{2} \right)^K D(\omega), \quad K \leq \frac{N}{2} \quad (2.140)$$

($D(\omega)$ is a trigonometric polynomial).

The filter must satisfy the orthogonality condition (2.139).

Let us work on one of the squared terms that appear in the orthogonality condition:

$$\begin{aligned} |H_0(\omega)|^2 &= 2 \left| \frac{1 + e^{-j\omega}}{2} \right|^{2K} |D(\omega)|^2 = 2 \left| \cos^2\left(\frac{\omega}{2}\right) \right|^K |D(\omega)|^2 = \\ &= 2 \left| \cos^2\left(\frac{\omega}{2}\right) \right|^K P(\sin^2(\frac{\omega}{2})) = 2(1 - y)^K P(y) \end{aligned} \quad (2.141)$$

(change of variable: $y = \sin^2(\omega/2)$)

Therefore the orthogonality condition (2.139) can be written as:

$$(1 - y)^K P(y) + y^K P(1 - y) = 1 \quad (2.142)$$

which is a Bezout's identity. In this case there are explicit solutions. If we set $K = N/2$, the solution is:

$$P(y) = \sum_{l=0}^{K-1} \binom{K-1+l}{l} y^l \quad (2.143)$$

If we want $K < N/2$, then the solution is:

$$P(y) = \sum_{l=0}^{K-1} \binom{K-1+l}{l} y^l + y^K R\left(\frac{1}{2} - y\right) \quad (2.144)$$

where $R(y)$ is an odd polynomial such that:

$$P(y) \geq 0 \text{ for } 0 \leq y \leq 1 \quad (2.145)$$

Here are some examples of $P(y)$ for several values of $K = N/2$:

$$\begin{aligned} K = 2, \quad P(y) &= 1 + 2y \\ K = 3, \quad P(y) &= 1 + 3y + 6y^2 \\ K = 4, \quad P(y) &= 1 + 4y + 10y^2 + 20y^3 \\ K = 5, \quad P(y) &= 1 + 5y + 15y^2 + 35y^3 + 70y^4 \end{aligned} \quad (2.146)$$

Once $|H_0(\omega)|^2$ is obtained, its “square root” can be computed via factorization of $P(y)$. It is convenient for this purpose to use $z = \exp(j\omega)$. The target is to get a factorization in the form $P(z) = L(z) L(z^{-1})$. First notice that:

$$y = \sin^2\left(\frac{\omega}{2}\right) = \frac{1}{2} - \frac{z + z^{-1}}{4} = \frac{2 - (z + z^{-1})}{4} \quad (2.147)$$

Introduce an auxiliary polynomial $Q(z)$ such that:

$$P\left(\frac{2 - (z + z^{-1})}{4}\right) = z^{-(K-1)} Q(z) \quad (2.148)$$

Now compute the roots a_m of the polynomial $Q(z)$. Then:

$$L(z) L(z^{-1}) = r^2 \prod_{m=1}^{K-1} (1 - a_m z) (1 - a_m^{-1} z^{-1}) = z^{-(K-1)} Q(z) \quad (2.149)$$

with:

$$r = 1 / \left(\prod_{m=1}^{K-1} (1 - a_m) \right) \quad (2.150)$$

Factorization becomes easy with the help of $Q(z)$. For example, in the case of $K = 2$:

$$\begin{aligned} P(y) &= 1 + 2y \\ P\left(\frac{2 - (z + z^{-1})}{4}\right) &= 1 + 2\left(\frac{2 - (z + z^{-1})}{4}\right) = \frac{4 - (z + z^{-1})}{2} = z^{-1} Q(z) \end{aligned} \quad (2.151)$$

$$Q(z) = \frac{1}{2} (-z^2 + 4z - 1) \quad (2.152)$$

The roots of $Q(z)$ are $2 + \sqrt{3}$, $2 - \sqrt{3}$. Let us choose the root inside the unit circle ($|a_m| < 1$) for $L(z)$. Therefore:

$$\begin{aligned} L(z) &= r (1 - a_1 z) = \frac{1}{1 - a_1} (1 - a_1 z) = \frac{1}{\frac{\sqrt{3}-1}{2}} (1 - (2 - \sqrt{3}) z) = \\ &= \frac{\sqrt{3}+1}{2} (1 - (2 - \sqrt{3}) z) = \frac{\sqrt{3}+1}{2} - \frac{\sqrt{3}-1}{2} z \end{aligned} \quad (2.153)$$

Likewise:

$$L(z^{-1}) = \frac{\sqrt{3}+1}{2} - \frac{\sqrt{3}-1}{2} z^{-1} \quad (2.154)$$

with it the desired result is obtained:

$$\begin{aligned} H_0(z) &= \sqrt{2} \left(\frac{1+z}{2}\right)^2 L(z) = \frac{\sqrt{2}}{4} (1 + 2z + z^2) \left(\frac{\sqrt{3}+1}{2} - \frac{\sqrt{3}-1}{2} z\right) = \\ &= \frac{1}{4\sqrt{2}} (1 + \sqrt{3} + (3 + \sqrt{3}) z + (3 - \sqrt{3}) z^2 + (1 - \sqrt{3}) z^3) \end{aligned} \quad (2.155)$$

(this gives the filter coefficients already obtained in Sect. 2.3.3. for $M = 4$).

Other Daubechies orthonormal wavelets can be obtained for higher values of K , following the same factorization procedure, which gives minimal phase H_0 filters (since roots inside the unit circle were chosen for $L(z)$). A colateral result of the minimal phase is the marked asymmetry of the scaling functions and wavelets. This asymmetry denotes that the filter H_0 is non-linear phase.

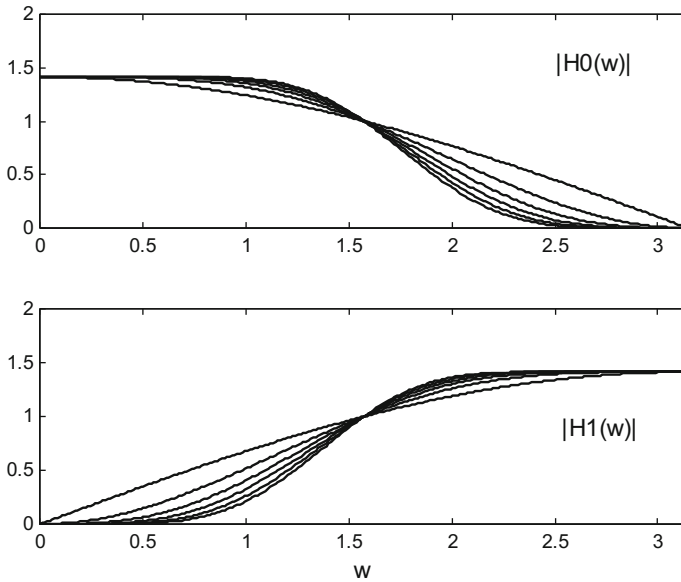


Fig. 2.39 Frequency magnitude responses of H_0 and H_1 Daubechies filters for $N = 2, 4, 6, \dots, 12$

Other factorizations are possible. For instance a maximal phase factorization, choosing the roots outside the unit circle for $L(z)$. Or for instance mixed solutions, taking roots inside the unit circle and others outside the unit circle. Not always a factorization leads to an orthonormal wavelet basis; but it is sufficient that $H_0(\omega)$ has no zeros in the band $[0, \pi/2]$.

Daubechies has shown that, except by the Haar basis, there is no factorization yielding symmetric scaling functions and wavelets (for orthonormal wavelets having compact support).

Figure 2.39 shows the frequency magnitude responses of the filters H_0 and H_1 for $N = 2, 4, 6 \dots 12$ and $K = N/2$. The figure has been generated with the Program 2.22, which contains interesting code.

Program 2.22 Compute Daubechies filters for $N = 2, 4, 6 \dots 12$

```
% Compute Daubechies filters for N=2,4,6...12
w=0:(2*pi/511):pi;
for K=1:6,
a=1; p=1; q=1;
h=[1 1];
M=2*K; %length of the filter
% the h0(n) coefficients
for nn=1:K-1,
h=conv(h,[1,1]);
a=-a*0.25*(nn+K-1)/nn;
p=conv(p,[1,-2,1]);
```

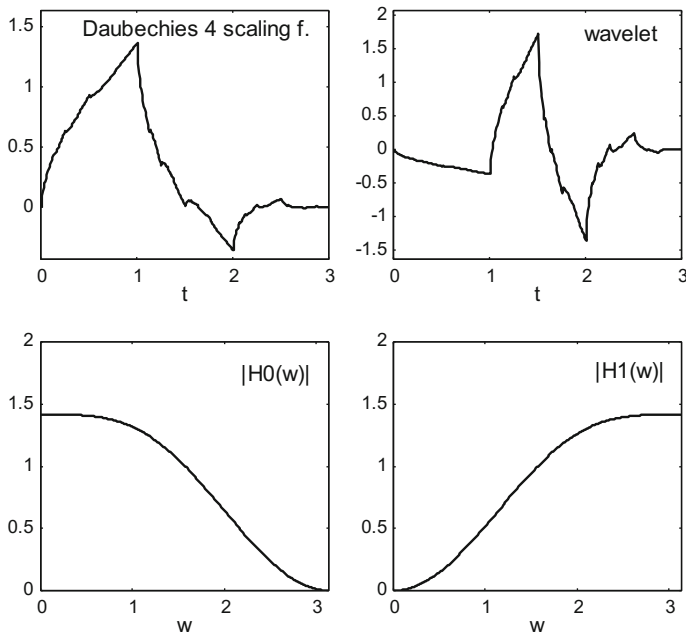



Fig. 2.40 Daubechies $N = 4$ scaling function, wavelet, and frequency magnitude responses of H_0 and H_1 filters

```

q=[0 q 0] + a*p;
end;
q=sort(roots(q));
aux=real(poly(q(1:K-1)));
h=conv(h,aux);
h0=(h*sqrt(2))/sum(h); %normalization
H0=fft(h0,512); %LP filter frequency response
%the h1(n) coefficients
h1=fliplr(h0); h1(1:2:end)=-h1(1:2:end);
H1=fft(h1,512); %HP filter frequency response
%display
subplot(2,1,1)
plot(w,abs(H0(1:256)), 'k'); hold on;
axis([0 pi 0 2]);
title(' |H0(w) | ');
subplot(2,1,2)
plot(w,abs(H1(1:256)), 'k'); hold on;
axis([0 pi 0 2]);
title(' |H1(w) | '); xlabel('w');
end;

```

Once the $h_0(n)$ have been determined, it is possible to compute the scaling function, the $h_1(n)$ coefficients by orthogonality, and the wavelet. The Program 2.25 demonstrates this, using as example Daubechies with $N = 4$. Fig. 10.4.12 shows the results: the scaling function, the wavelet, and the frequency magnitude responses of the corresponding H_0 and H_1 filters. Notice the flatness of the filters.

The Program 2.23 can easily be modified to study what happens with other values of N . Simply change the line stating $N = 4$.

Notice in the figure the finite support of the scaling function and the wavelet (Fig. 2.40).

Program 2.23 Compute Daubechies $h_0(n)$, $h_1(n)$, $\phi(t)$, $\psi(t)$

```
% Compute Daubechies h0(n), h1(n), phi(t), psi(t)
% for M=4 (length of the filter)
a=1; p=1; q=1;
h=[1 1];
M=4; %length of the filter
K=M/2;
% the h0(n) coefficients
for nn=1:K-1,
h=conv(h,[1,1]);
a=-a*0.25*(nn+K-1)/nn;
p=conv(p,[1,-2,1]);
q=[0 q 0] + a*p;
end;
q=sort(roots(q));
aux=real(poly(q(1:K-1)));
h=conv(h,aux);
h0=(h*sqrt(2))/sum(h); %normalization
H0=fft(h0,512); %LP filter frequency response
%the scaling function phi(t), using cascade algorithm
Ns=128; %number of fi samples
hN=sqrt(2)*h0;
phi=[ones(1,3*M*Ns),0]/(3*M); %initial iteration
%upsample hN, inserting Ns-1 zeros between samples
hup=[hN;zeros(Ns-1,M)];
hup=hup(1:(Ns*M));
%iteration
for nn=0:12,
aux=conv(hup,phi);
phi=aux(1:2:length(aux)); %downsampling by 2
end
%the h1(n) coefficients
h1=flipr(h0); h1(1:2:end)=-h1(1:2:end);
H1=fft(h1,512); %HP filter frequency response
%the wavelet psi(t), using definition
%upsample by K
hN=-sqrt(2)*h1;
h1up=[hN;zeros(Ns-1,M)];
```

```

hlup=hlup(1:Ns*M-1);
%downsample by 2
aux=conv(hlup,phi);
psi=aux(1:2:length(aux));
%display
subplot(2,2,1)
phi=phi(1:(M-1)*Ns); %the supported part
t=(1:length(phi))/Ns;
plot(t,phi,'k'); %plots the scaling function
axis([0 max(t) 1.2*min(phi) 1.2*max(phi)]);
title('Daubechies 4 scaling f. '); xlabel('t');
subplot(2,2,2)
psi=psi(1:(M-1)*Ns); %the supported part
plot(t,psi,'k'); %plots the wavelet
axis([0 max(t) 1.2*min(psi) 1.2*max(psi)]);
title('wavelet '); xlabel('t');
w=0:(2*pi/511):pi;
subplot(2,2,3)
plot(w,abs(H0(1:256)), 'k');
axis([0 pi 0 2]);
title(' |H0(w)| '); xlabel('w');
subplot(2,2,4)
plot(w,abs(H1(1:256)), 'k');
axis([0 pi 0 2]);
title(' |H1(w)| '); xlabel('w');

```

The code of the previous programs has been re-used to study the changes on scaling functions and wavelets as N grows up. Figure 2.41 shows the scaling function (left) and the wavelet (right) for $N = 4, 8, 12, \dots, 26$. The figure has been generated with the Program A.4 that has been included in Appendix A.

It is interesting to note that Daubechies obtains the Haar wavelet for $N = 2$. This is not shown in the figure, but it should be easy for the reader to plot this case.

Notice again that for $N > 2$ the Daubechies scaling function and wavelet are non-symmetrical.

2.4.3.3 Symlets

Symmlets are the result of factorizations leading to the least asymmetric scaling function and wavelet. In other words, one tries to get as linear phase as possible.

Each of the roots chosen for $L(z)$ has a corresponding phase contribution to the phase θ of $H_0(\omega)$. A measure of the nonlinear part of θ was defined by Daubechies as follows:

$$\eta(\omega) = \theta(\omega) - \frac{\omega}{2\pi}\theta(2\pi) \quad (2.156)$$

The idea is then to choose the roots minimizing $\eta(\omega)$.

Next table gives the coefficients of symlets 2, 3 and 4.

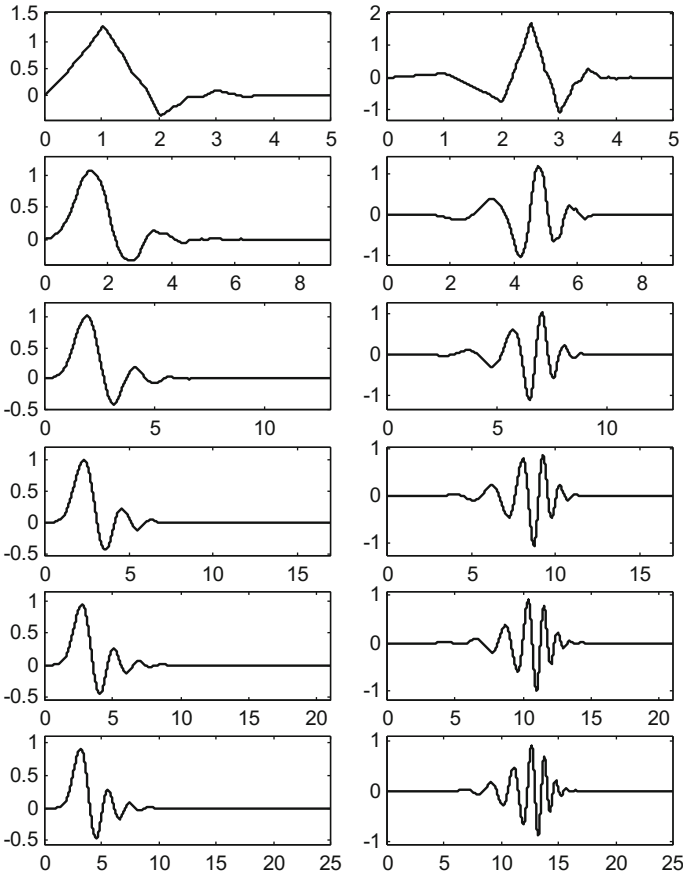


Fig. 2.41 Daubechies scaling function (*left*) and wavelet (*right*) for $N = 4, 8, 12, \dots, 26$

h_0 (symlet 2)	h_0 (symlet 3)	h_0 (symlet 4)
0.482962913145	0.332670552951	0.032223100604
0.836516303737	0.806891509313	-0.012603967262
0.224143868042	0.459877502119	-0.099219543577
-0.129409522551	-0.135011020010	0.297857795606
	-0.085441273882	0.803738751807
	0.035226291882	0.497618667633
		-0.029635527646
		-0.075765714789

A more complete table, with a different normalization of coefficients, is given in [25], (p. 198).

Figure 2.42 shows the symlet 4 scaling function and wavelet. The figure has been generated with Program 2.24.

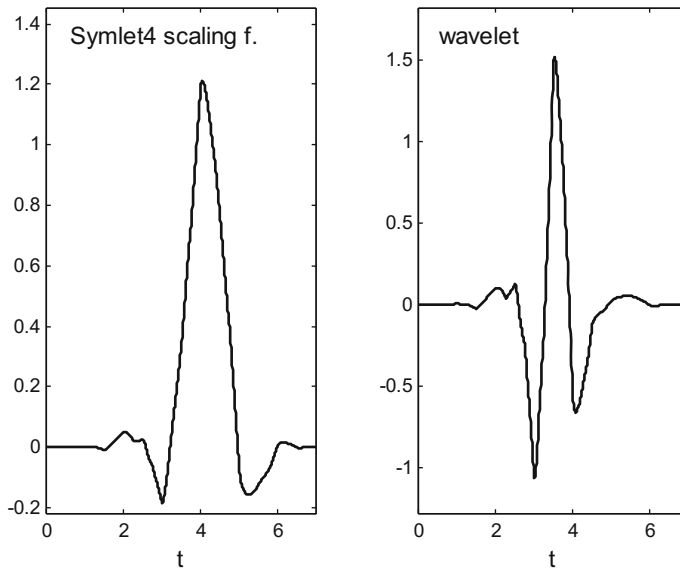


Fig. 2.42 Symlet 4 scaling function (*left*) and wavelet (*right*)

Program 2.24 Symlet4 $\phi(t)$, $\psi(t)$

```
% Symlet4 phi(t), psi(t)
%coefficients
h0=[0.032223100604,-0.012603967262,-0.099219543577...
,0.297857795606,0.803738751807,0.497618667633...
,-0.029635527646,-0.075765714789];
Ns=128; %number of function samples
%scaling function
%using cascade algorithm
M=length(h0);
hN=sqrt(2)*h0;
phi=[ones(1,3*M*Ns),0]/(3*M); %initial iteration
%upsample hN, inserting Ns-1 zeros between samples
hup=[hN;zeros(Ns-1,M)];
hup=hup(1:(Ns*M));
%iteration
for nn=0:12,
aux=conv(hup,phi);
phi=aux(1:2:length(aux)); %downsampling by 2
end
%wavelet
%the h1(n) coefficients
h1=flipplr(h0); h1(1:2:end)=-h1(1:2:end);
%the wavelet psi(t), using definition
%upsample
```

```

hN=sqrt(2)*h1;
hlup=[hN;zeros(Ns-1,M)];
hlup=hlup(1:Ns*M-1);
%downsample by 2
aux=conv(hlup,phi);
psi=aux(1:2:length(aux));
%display
subplot(1,2,1)
phi=phi(1:(M-1)*Ns); %the supported part
t=(1:length(phi))/Ns;
plot(t,phi,'k'); %plots the scaling function
axis([0 max(t) 1.2*min(phi) 1.2*max(phi)]);
title('Symlet4 scaling f. '); xlabel('t');
subplot(1,2,2)
psi=psi(1:(M-1)*Ns); %the supported part
t=(1:length(psi))/Ns;
plot(t,psi,'k'); %plots the wavelet
axis([0 max(t) 1.2*min(psi) 1.2*max(psi)]);
title('wavelet '); xlabel('t');

```

2.4.3.4 Coiflets

In 1989 R. Coifman suggested to Daubechies to construct wavelet bases with vanishing moments not only for the wavelet, but also for the scaling function. That is:

$$\forall m_0(k) = 0, \quad k = 0, 1, 2, \dots, L-1 \quad (2.157)$$

$$\forall m_1(k) = 0, \quad k = 0, 1, 2, \dots, L-1 \quad (2.158)$$

where L is the order of the *coiflet*.

According with [25], in order to have the specified wavelet vanishing moments, $H_0(\omega)$ should be:

$$H_0(\omega) = \sqrt{2} \left(\frac{1 + e^{-j\omega}}{2} \right)^L D(\omega) \quad (2.159)$$

In addition, to have the specified scaling function vanishing moments, $H_0(\omega)$ should be:

$$H_0(\omega) = \sqrt{2} (1 + (1 - e^{-j\omega})^L) E(\omega) \quad (2.160)$$

($D(\omega)$ and $E(\omega)$ are trigonometric polynomials).

Suppose that L is even, so $L = 2K$.

Since

$$\begin{aligned} \left(\frac{1 + e^{-j\omega}}{2} \right)^{2K} &= e^{-j\omega K} (\cos^2(\omega/2))^K, \\ (1 - e^{-j\omega})^{2K} &= e^{-j\omega K} (2j \sin^2(\omega/2))^{2K} \end{aligned} \quad (2.161)$$

Then, to satisfy (2.159) and (2.160), one has to find two trigonometric polynomials, $P_1(\omega)$, $P_2(\omega)$, such that:

$$(\cos^2(\omega/2))^K P_1(\omega) = 1 + (\sin^2(\omega/2))^K P_2(\omega) \quad (2.162)$$

This is again the Bezout equation. The solution for $P_1(\omega)$ is:

$$P_1(\omega) = \sum_{l=0}^{K-1} \binom{K-1+l}{l} (\sin^2(\omega/2))^l + (\sin^2(\omega/2))^K f(\omega) \quad (2.163)$$

In this expression a $f(\omega)$ must be found to satisfy the orthogonality condition. An approach for this is to take:

$$f(\omega) = \sum_{n=0}^{L-1} f_n e^{-jn\omega} \quad (2.164)$$

Then a system of K quadratic equations can be set.

Next table gives the coefficients of the coiflets for $K = 1$ and $K = 2$.

$h_0/\sqrt{2}$ (for $K = 1$)	$h_0/\sqrt{2}$ (for $K = 2$)
-0.051429728471	0.011587596739
0.238929728471	-0.029320137980
0.602859456942	-0.047639590310
0.272140543058	0.273021046535
-0.051429972847	0.574682393857
-0.011070271529	0.294867193696
	-0.054085607092
	-0.042026480461
	0.016744410163
	0.003967883613
	-0.001289203356
	-0.000509505399

A more complete table can be found in [25] (p. 261).

Figure 2.43 shows the Coiflet1 scaling function and wavelet. The figure has been generated with Program A.5 which is similar to the previous program, the only change is in the coefficients. This program has been included in Appendix A.

2.4.3.5 Example of Signal Analysis and Recovery with Daubechies 4

To conclude this section it seems rewarding to see an example of wavelet decomposition of a signal (analysis), and then a recovery of the signal from the wavelets (synthesis). We use for this example the Daubechies 4 wavelets.

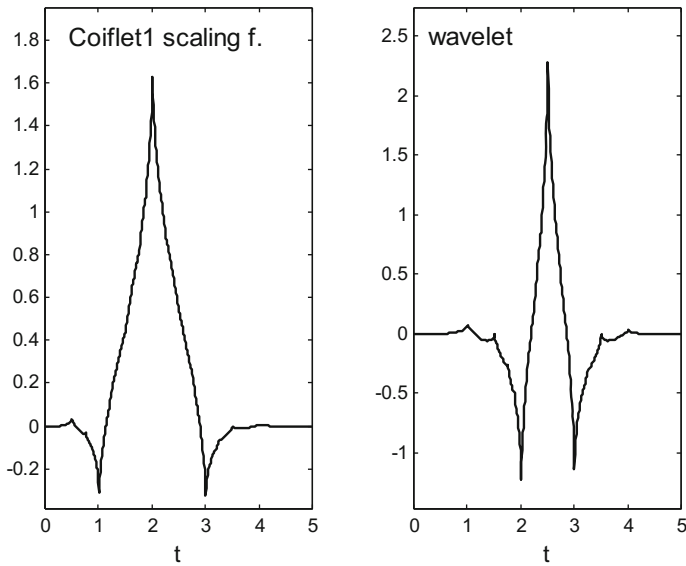


Fig. 2.43 Coiflet1 scaling function (*left*) and wavelet (*right*)

The signal selected for this example is a pattern visual evoked potential (see the web page of Rodrigo Quian Quiroga). It was recorded with a left occipital electrode.

The scalogram obtained with the wavelet analysis is shown in Fig. 2.44. Program 2.25 has been used for the analysis and graphical representation.

Program 2.25 Visual Evoked Potential analysis

```
% Visual Evoked Potential analysis
% Daubechies 4 Wavelet
% Plot of signal and scalogram
%The EVP signal
fs=250; %samplig frequency in Hz
tiv=1/fs; %time interval between samples
Ts=tiv; %sampling period
%read signal file
fer=0;
while fer==0,
    fid2=fopen('EVP_short.txt','r');
    if fid2==~-1, disp('read error')
    else sg=fscanf(fid2,'%f \r\n'); fer=1;
end;
end;
fclose('all');
Nss=length(sg); %number of signal samples
duy=(Nss-1)*tiv; %duration of signal
tss=0:tiv:duy; %time intervals set
```



```

%analysis of the signal with wavelets
y=sg;
%scaling filter
hden=4*sqrt(2); %coeff. denominator
hsq=sqrt(3); %factor
%Daubechies 4:
h=[(1+hsq)/hden, (3+hsq)/hden, (3-hsq)/hden, (1-hsq)/hden];
N=length(h);
K=9; %number of scales (512=2^9)
dd=zeros(K,Nss/2); %space for coefficients
a=y';
aux=0;
h0=flipplr(h);
h1=h; h1(1:2:N)=-h1(1:2:N);
%wavelet calculus using filters
NN=Nss;
for n=1:K,
L=length(a);
a=[a(mod((- (N-1):-1),L)+1) a];
d=conv(a,h1);
d=d(N:2:(N+L-2));
a=conv(a,h0);
a=a(N:2:(N+L-2));
aux=[d,aux];
dd(K+1-n,1:NN/2)=d;
NN=NN/2;
end;
wty=[a,aux(1:end-1)];
%preparing for scalogram
S=zeros(K,Nss); %space for S(j,k) scalogram coefficients
for n=1:K,
q=2^(n-1); L=Nss/q;
for m=1:q,
R=(1+(L*(m-1))):(L*m); %index range
S(n,R)=dd(n,m);
end;
end;
%figure
subplot('position',[0.04 0.77 0.92 0.18])
plot(y);
axis([0 512 1.2*min(y) 1.2*max(y)]);
title('signal');
subplot('position',[0.04 0.05 0.92 0.6])
imagesc(S); colormap('bone');
title('Scalogram of Daubechies w.t. Evoked Potential signal');
h=gca; set(h,'YDir','normal');

```

Program 2.26 does the recovery of the evoked potential signal, with Daubechies 4 wavelet synthesis. Figure 2.45 shows the recovered signal.

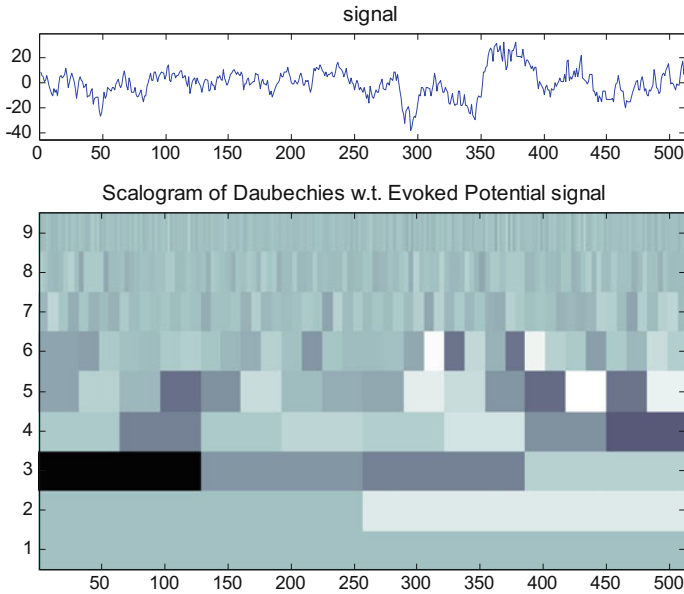
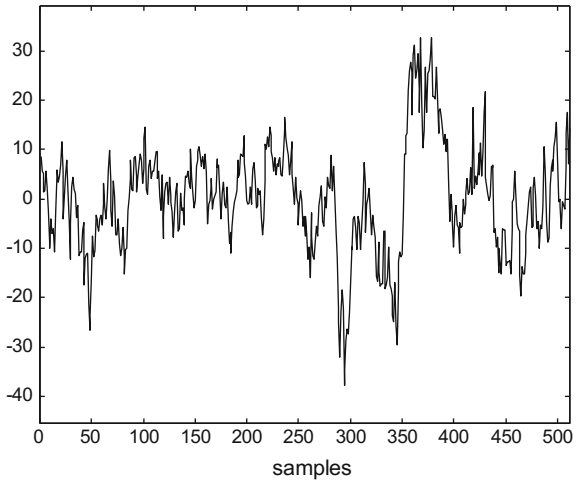


Fig. 2.44 Scalogram of Evoked Potential signal using Daubechies 4

Fig. 2.45 Recovered
Evoked Potential signal



Program 2.26 Inverse of the Daubechies DWT of Evoked Potential signal

```
% inverse of the Daubechies DWT
% Visual Evoked Potential signal
L=length(wty); %length of the DWT
%scaling filter
hden=4*sqrt(2); %coeff. denominator
```

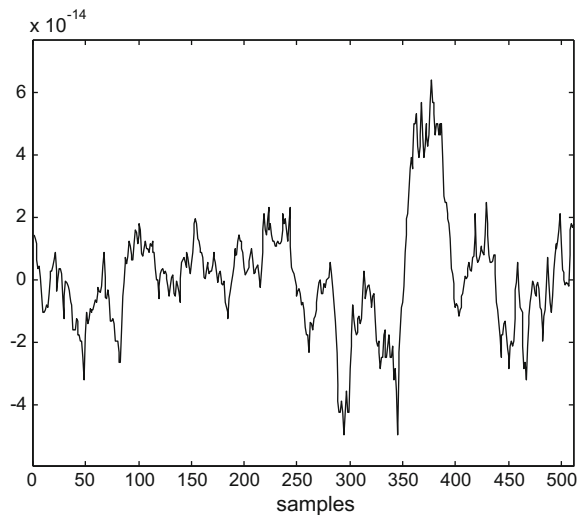
```

hsq=sqrt(3); %factor
%Daubechies 4:
h=[(1+hsq)/hden, (3+hsq)/hden, (3-hsq)/hden, (1-hsq)/hden];
hN=h;
N=length(hN);
K=9; %number of scales
aux=0;
h0=hN;
h1=flipplr(hN); h1(2:2:N)=-h1(2:2:N);
Ln=1;
a=wt(1);
for n=1:K,
aux= 1+mod(0:N/2-1,Ln);
d=wt(Ln+1:2*Ln);
ax(1:2:2*Ln+N)= [a a(1,aux)];
dx(1:2:2*Ln+N)= [d d(1,aux)];
a=conv(ax,h0)+ conv(dx,h1);
a=a(N:(N+2*Ln-1));
Ln=2*Ln;
end;
figure(1)
plot(a,'k');
axis([0 512 1.2*min(a) 1.2*max(a)]);
xlabel('samples');
title('the recovered signal');

```

A first intent of comparing the original evoked potential signal, and the recovered signal, was to plot one on top the other. But both signals are practically identical, so

Fig. 2.46 Difference between original and recovered signals



it is not possible to discern one from the other. What has been done is to subtract both signals and depict—Fig. 2.46—the result. Notice that the scale of the vertical axis is 10^{-14} .

2.5 Biorthogonal Wavelets

The use of orthogonal wavelet systems cause limitations in design freedom, and prevent linear phase analysis and synthesis filter banks. Biorthogonal wavelet systems offer greater flexibility.

In a biorthogonal system, a primal scaling function $\varphi(t)$ and a dual scaling function $\tilde{\varphi}(t)$ are defined, such that:

$$\varphi(t) = \sum_n h_0(n) \sqrt{2} \varphi(2t - n) \quad (2.165)$$

$$\tilde{\varphi}(t) = \sum_n \tilde{h}_0(n) \sqrt{2} \tilde{\varphi}(2t - n) \quad (2.166)$$

In order for $\varphi(t)$ and $\tilde{\varphi}(t)$ to exist:

$$\sum_n h_0(n) = \sum_n \tilde{h}_0(n) = \sqrt{2} \quad (2.167)$$

A primal and a dual wavelet are also defined, such that:

$$\psi(t) = \sum_n h_1(n) \sqrt{2} \varphi(2t - n) \quad (2.168)$$

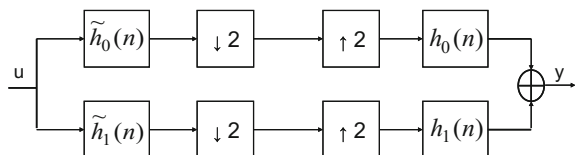
$$\tilde{\psi}(t) = \sum_n \tilde{h}_1(n) \sqrt{2} \tilde{\varphi}(2t - n) \quad (2.169)$$

The corresponding biorthogonal filter bank is shown in Fig. 2.47.

For perfect reconstruction:

$$\sum_k (h_0(2k - m) \tilde{h}_0(2k - n) + h_1(2k - m) \tilde{h}_1(2k - n)) = \delta_{m,n} \quad (2.170)$$

Fig. 2.47 A biorthogonal filter bank



where m, n , are integers.

In order for this condition to hold, the four filters have to be:

$$\tilde{h}_1(n) = \pm(-1)^n h_0(L - n) \quad (2.171)$$

$$\tilde{h}_0(n) = \pm(-1)^n h_1(L - n) \quad (2.172)$$

where L is an arbitrary odd integer.

Using these relationships, the condition (2.170) becomes:

$$\sum_n \tilde{h}_0(n) h_0(n + 2k) = \delta_k \quad (2.173)$$

Primal and dual baby families are also defined:

$$\varphi_{j,k}(t) = \sqrt{2^j} \varphi(2^j t - k) \quad (2.174)$$

$$\tilde{\varphi}_{j,k}(t) = \sqrt{2^j} \tilde{\varphi}(2^j t - k) \quad (2.175)$$

$$\psi_{j,k}(t) = \sqrt{2^j} \psi(2^j t - k) \quad (2.176)$$

$$\tilde{\psi}_{j,k}(t) = \sqrt{2^j} \tilde{\psi}(2^j t - k) \quad (2.177)$$

If Eq.(2.170) is satisfied and some other conditions are satisfied by the primal and dual scaling functions, the primal baby wavelets constitute a frame in L^2 , and the dual baby wavelets constitute the dual frame if and only if:

$$\int \varphi(t) \tilde{\varphi}(t - k) dt = \delta_k \quad (2.178)$$

Then the primal and dual baby wavelets constitute two Riesz bases, with:

$$\langle \psi_{j,k}, \tilde{\psi}_{j',k'} \rangle = \delta_{j,j'} \delta_{k,k'} \quad (2.179)$$

Concerning multiresolution formulations, we have:

$$V_j \subset V_{j+1}, \quad \tilde{V}_j \subset \tilde{V}_{j+1} \quad (2.180)$$

$$V_j \perp \tilde{W}_j, \quad \tilde{V}_j \perp W_j \quad (2.181)$$

The coefficients corresponding to the expansion of a signal $y(t)$ in terms of the primal baby scaling functions and wavelets, can be obtained with:

$$a_{j,k} = \langle y, \tilde{\varphi}_{j,k} \rangle \quad (2.182)$$

$$d_{j,k} = \langle y, \tilde{\psi}_{j,k} \rangle \quad (2.183)$$

And:

$$a_{j,k} = \sum_m \tilde{h}_0(m - 2k) a_{j+1,m} \quad (2.184)$$

$$d_{j,k} = \sum_m \tilde{h}_1(m - 2k) a_{j+1,m} \quad (2.185)$$

All filters can be symmetric (linear phase). This is a great advantage of biorthogonal wavelets.

2.5.1 Daubechies Approach

Orthogonal wavelets can be considered as a particular case of biorthogonal wavelets. In fact the approach of K -regular filters can be used to obtain orthogonal or biorthogonal wavelets.

In orthogonal wavelets we needed to get the “square root” of $|H_0(\omega)|^2$, in the case of biorthogonal wavelets we only need a factorization into two filters (the other two filters are obtained according with (2.171) and (2.172)).

As required for perfect reconstruction in a two-filter bank, the distortion term (see Sect. 1.4.1.) must be in this case the following:

$$\tilde{H}_0(\omega) H_0(\omega)^* \cdot \tilde{H}_0(\omega + \pi) H_0(\omega + \pi)^* = 2 \quad (2.186)$$

(this also corresponds to condition (2.173)).

Symmetry of filters can be obtained if one chooses $H_0(\omega)$, $\tilde{H}_0(\omega)$ of the form;

$$\sqrt{2}(\cos(\omega/2))^L q_0(\cos \omega) \quad (2.187)$$

for L even.

or

$$\sqrt{2} e^{-j\omega/2} (\cos(\omega/2))^L q_0(\cos \omega) \quad (2.188)$$

for L odd.

Where $q()$ is a polynomial. Substitution into (2.186) gives:

$$\begin{aligned} & (\cos(\omega/2))^K \tilde{q}_0(\cos \omega) q_0(\cos \omega)^* + \\ & + (\sin(\omega/2))^K \tilde{q}_0(-\cos \omega) q_0(-\cos \omega)^* = 1 \end{aligned} \quad (2.189)$$

where $K = L_0 + L_1$ for L_0 even, or $K = L_0 + L_1 + 1$ for L_0 odd. The integers L_0 and L_1 correspond to $\tilde{H}_0(\omega)$ and $H_0(\omega)$.

Now, if you define:

$$P(\sin^2(\omega/2)) = \tilde{q}_0(\cos \omega) \cdot q_0(\cos \omega)^* \quad (2.190)$$

Then:

$$(1 - y)^K P(y) + y^K P(1 - y) = 1 \quad (2.191)$$

and so, the solution is once again:

$$P(y) = \sum_{l=0}^{K-1} \binom{K-1+l}{l} y^l + y^K R\left(\frac{1}{2} - y\right) \quad (2.192)$$

Daubechies [25] obtained several families of biorthogonal wavelets, using different choices for $R()$ and for the factorization of $P()$.

A first alternative is to choose $R() = 0$. In this case, the design focuses on the summatorial term. Several families of biorthogonal wavelets have been obtained, including the spline biorthogonal wavelets.

2.5.1.1 Spline Biorthogonal Wavelets

From the previous equations, and taking $R() = 0$, a family of spline biorthogonal wavelets is directly obtained as follows.

- For odd-length filters, the analysis and synthesis low-pass filters are:

$$\tilde{H}_0(\omega) = \sqrt{2} (\cos^2(\omega/2))^{L_0} \sum_{n=0}^{K-1} \binom{K+n-1}{n} (\sin^2(\omega/2))^n \quad (2.193)$$

$$H_0(\omega) = \sqrt{2} (\cos^2(\omega/2))^{L_1} \quad (2.194)$$

with $K = L_0 + L_1$.

- For even-length filters, the analysis and synthesis low-pass filters are:

$$\tilde{H}_0(\omega) = \sqrt{2} e^{-j\omega/2} (\cos^2(\omega/2))^{L_0} \sum_{n=0}^{K-1} \binom{K+n}{n} (\sin^2(\omega/2))^n \quad (2.195)$$

$$H_0(\omega) = \sqrt{2} e^{-j\omega/2} (\cos^2(\omega/2))^{L_1} \quad (2.196)$$

with $K = L_0 + L_1 + 1$.

Here is a table with the coefficients of some members of the Cohen-Daubechies-Feauveau family of biorthogonal spline wavelets [20]:

A more complete table is given in [25], (p. 277), and in [20].

$h_0/\sqrt{2}$	$\tilde{h}_0/\sqrt{2}$
1/2, 1/2	1/2, 1/2
1/2, 1/2	-1/16, 1/16, 1/2, 1/2, 1/16, -1/16
1/4, 1/2, 1/4	-1/8, 1/4, 3/4, 1/4, -1/8
1/4, 1/2, 1/4	3/128, -3/64, -1/8, 19/64, 45/65, 19/64, -1/8, -3/64, 3/128
1/8, 3/8, 3/8, 1/8	-1/4, 3/4, 3/4, -1/4
1/8, 3/8, 3/8, 1/8	3/64, -9/64, -7/64, 45/64, 45/64, -7/64, -9/64, 3/64
1/8, 3/8, 3/8, 1/8	-5/512, 15/512, 19/512, -97/512, -13/256, 175/256, 175/256, -13/256, -97/512, 19/512, 15/512, -5/512

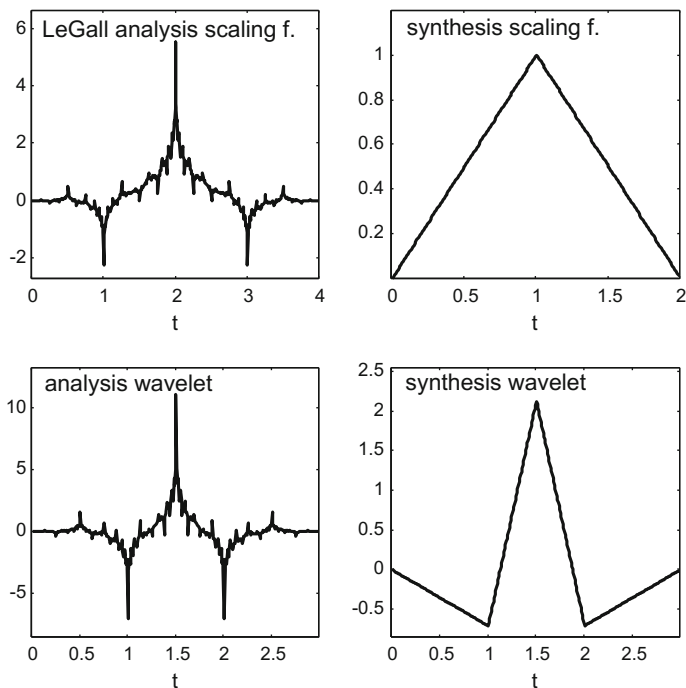


Fig. 2.48 LeGall scaling functions and wavelets

The third row of coefficients corresponds to the LeGall 5/3 wavelet [75]. Figure 2.48 shows the scaling functions and the wavelets of LeGall 5/3. The figure has been generated with Program 2.27.

Program 2.27 LeGall $\phi(t)$, $\psi(t)$

```
% LeGall phi(t), psi(t)
%coefficients
ah0=[-1/8, 1/4, 3/4, 1/4, -1/8];
sh0=[1/4, 1/2, 1/4];
Ns=128; %number of function samples
%analysis scaling function
%using cascade algorithm
```



```

Ma=length(ah0);
ah0=2*ah0/sum(ah0); %normalization
aphi=[ones(1,3*Ma*Ns),0]/(3*Ma); %initial iteration
%upsample hN, inserting Ns-1 zeros between samples
hup=[ah0;zeros(Ns-1,Ma)];
hup=hup(1:(Ns*Ma));
%iteration
for nn=0:12,
aux=conv(hup,aphi);
aphi=aux(1:2:length(aux)); %downsampling by 2
end
%synthesis scaling function
%using cascade algorithm
Ms=length(sh0);
sh0=2*sh0/sum(sh0); %normalization
sphi=[ones(1,3*Ms*Ns),0]/(3*Ms); %initial iteration
%upsample hN, inserting Ns-1 zeros between samples
hup=[sh0;zeros(Ns-1,Ms)];
hup=hup(1:(Ns*Ms));
%iteration
for nn=0:12,
aux=conv(hup,sphi);
sphi=aux(1:2:length(aux)); %downsampling by 2
end
%analysis wavelet
%the ah1(n) coefficients
ah1=fliplr(sh0); ah1(1:2:end)=-ah1(1:2:end);
%the wavelet psi(t), using definition
%upsample
hN=sqrt(2)*ah1;
hlup=[hN;zeros(Ns-1,Ms)];
hlup=hlup(1:Ns*Ms-1);
%downsample by 2
aux=conv(hlup,aphi);
apsi=aux(1:2:length(aux));
%synthesis wavelet
%the sh1(n) coefficients
sh1=fliplr(ah0); sh1(1:2:end)=-sh1(1:2:end);
%the wavelet psi(t), using definition
%upsample
hN=-sqrt(2)*sh1;
hlup=[hN;zeros(Ns-1,Ma)];
hlup=hlup(1:Ns*Ma-1);
%downsample by 2
aux=conv(hlup,sphi);
spsi=aux(1:2:length(aux));
%display
subplot(2,2,1)
aphi=aphi(1:(Ma-1)*Ns); %the supported part

```

```

t=(1:length(aphi))/Ns;
plot(t,aphi,'k'); %plots the scaling function
axis([0 max(t) 1.2*min(aphi) 1.2*max(aphi)]);
title('LeGall analysis scaling f. '); xlabel('t');
subplot(2,2,3)
su=round(0.75*length(apsi));
t=(1:su)/Ns;
plot(t,apsi(1:su),'k'); %plots the wavelet
axis([0 max(t) 1.2*min(apsi) 1.2*max(apsi)]);
title('analysis wavelet'); xlabel('t');
subplot(2,2,2)
sphi=sphi(1:(Ms-1)*Ns); %the supported part
t=(1:length(sphi))/Ns;
plot(t,sphi,'k'); %plots the scaling function
axis([0 max(t) 1.2*min(sphi) 1.2*max(sphi)]);
title('synthesis scaling f. '); xlabel('t');
subplot(2,2,4)
su=round(0.75*length(spsi));
t=(1:su)/Ns;
plot(t,spsi(1:su),'k'); %plots the wavelet
axis([0 max(t) 1.2*min(spsi) 1.2*max(spsi)]);
title('synthesis wavelet'); xlabel('t');

```

2.5.1.2 Filters with Nearly Same Lengths

Notice that in the spline biorthogonal wavelets the part with the summatorial is entirely assigned to one of the filters, thus causing disparate filter lengths.

The filter lengths could be more balanced if $P(y)$ was factorized into two polynomials with similar lengths. Daubechies proposed to write $P(y)$ as a product of first and second order polynomial factors, based on real and complex zeros, and then build the two polynomials with these factors in appropriated form.

There is a table in [25], (p. 279) with some biorthogonal wavelets constructed in this way. One of these wavelets is the popular CDF 9/7, which has the following filter coefficients:

$h_0/\sqrt{2}$	$\tilde{h}_0/\sqrt{2}$
0.026748757411	-0.045635881557
-0.016864118443	-0.028771763114
-0.078223266529	0.295635881557
0.266864118443	0.557543526229
0.602949018236	0.295635881557
0.266864118443	-0.028771763114
-0.078223266529	-0.045635881557
-0.016864118443	
0.026748757411	

A main reason for the CDF 9/7 to be popular is that it has been adopted by the FBI for fingerprint image compression [13]. Also, the JPEG 2000 compression standard uses CDF 9/7, and LeGall 5/3 [64, 75].

Figure 2.49 shows the scaling functions and the wavelets of cdf 9/7. The figure has been generated with Program A.6 that is very similar to Program 2.27, with only changes of coefficients. Anyway, the Program A.6 has been included in Appendix A.

To complete the view of the CDF 9/7 filter bank, Fig. 2.50 shows the frequency responses of the four filters. The figure has been generated with Program 2.28.

Program 2.28 CDF 9/7 frequency response of filters

```
% CDF 9/7 frequency response of filters
%coefficients
ah0=[-0.045635881557,-0.028771763114,0.295635881557...
,0.557543526229,0.295635881557,-0.028771763114...
,-0.045635881557];
sh0=[0.026748757411,-0.016864118443,-0.078223266529...
,0.266864118443,0.602949018236,0.266864118443...
,-0.078223266529,-0.016864118443,0.026748757411];
ah0n=(ah0*sqrt(2))/sum(ah0); %normalization
sh0n=(sh0*sqrt(2))/sum(sh0); %normalization
ah0=fft(ah0n,512); %frequency response
sH0=fft(sh0n,512); %frequency response
%the ah1(n) coefficients
ah1=fliplr(sh0n); ah1(1:2:end)=-ah1(1:2:end);
%the sh1(n) coefficients
sh1=fliplr(ah0n); sh1(1:2:end)=-sh1(1:2:end);
aH1=fft(ah1,512); %frequency response
sH1=fft(sh1,512); %frequency response
%display
w=(0:(2*pi/511)):pi;
subplot(2,2,1)
plot(w,abs(aH0(1:256)),'k');
axis([0 pi 0 2]);
title(' | aH0(w) | ');xlabel('w');
subplot(2,2,2)
plot(w,abs(aH1(1:256)),'k');
axis([0 pi 0 2]);
title(' | aH1(w) | ');xlabel('w');
subplot(2,2,3)
plot(w,abs(sH0(1:256)),'k');
axis([0 pi 0 2]);
title(' | sH0(w) | ');xlabel('w');
subplot(2,2,4)
plot(w,abs(sH1(1:256)),'k');
axis([0 pi 0 2]);
title(' | sH1(w) | ');xlabel('w');
```

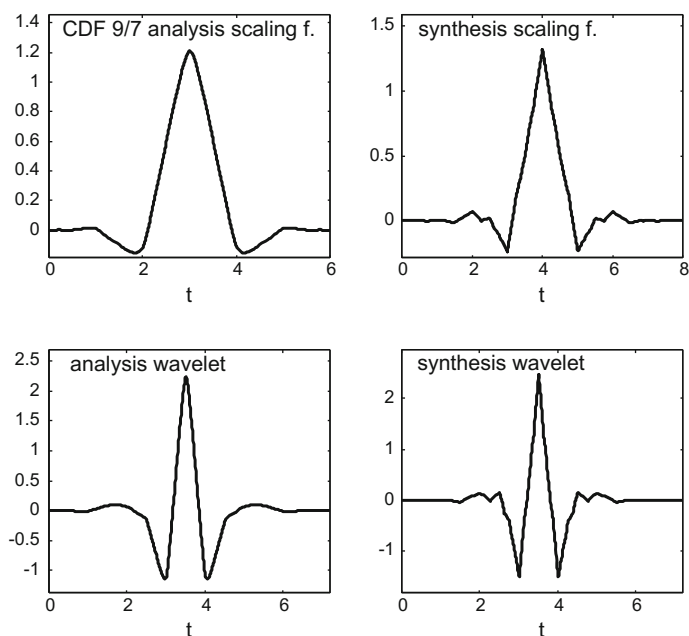


Fig. 2.49 CDF 9/7 scaling functions and wavelets

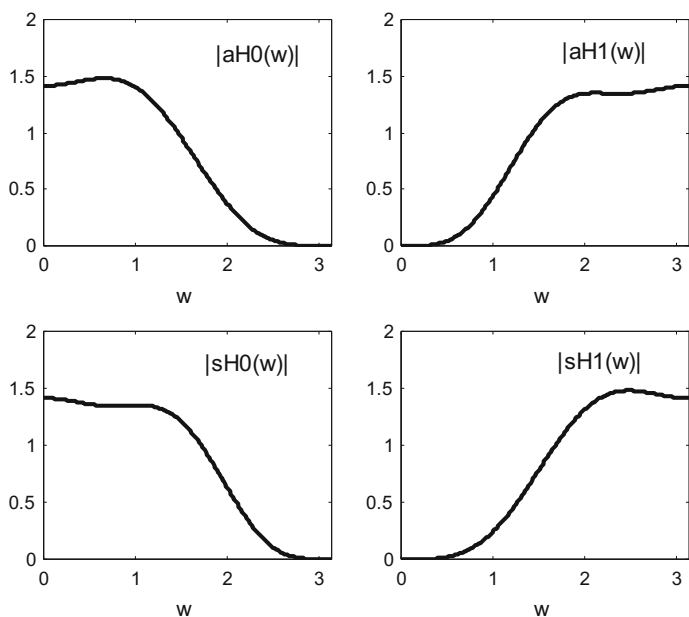


Fig. 2.50 Frequency responses of the CDF 9/7 filter banks

2.5.1.3 Almost Orthonormal Filters

Following the work of Daubechies in [25], it was considered that, based on a Laplacian pyramid filter [15], an almost orthonormal biorthogonal wavelet could be found. We will give more details of this type of filters in a next chapter.

The analysis filter was chosen to be a Laplacian pyramid filter:

$$\tilde{H}_0(\omega) = \sqrt{2} (\cos^2(\omega/2)) (1 + \frac{4}{5} \sin^2(\omega/2)) \quad (2.197)$$

Then, using (2.186):

$$H_0(\omega) = \sqrt{2} (\cos^2(\omega/2)) (1 + \frac{6}{5} \sin^2(\omega/2) - \frac{24}{35} \sin^4(\omega/2)) \quad (2.198)$$

The result is a biorthogonal wavelet that is very close to one of the orthonormal coiflets.

A second initiative of [25] was to drive near orthonormal wavelets using optimization procedures. A proximity criterion was defined, and some requisites were added, like to have rational filter coefficients. The analysis filter was chosen to be:

$$\begin{aligned} \tilde{H}_0(\omega) = & \sqrt{2} (\cos^2(\omega/2))^{2K} \cdot \\ & \cdot \left[\sum_{n=0}^{K-1} \binom{K+k-1}{k} (\sin^2(\omega/2))^{2k} + a \cdot (\sin^2(\omega/2))^{2K} \right] \end{aligned} \quad (2.199)$$

This expression has one parameter, a , to be optimized. Once its optimal value is found, the corresponding synthesis filter is obtained using (2.186) (actually it is similar to $\tilde{H}_0(\omega)$).

2.5.2 More Ways to Find Biorthogonal Wavelets

2.5.2.1 Still More About Coifman Biorthogonal Wavelets

There is more to say about Coifman and biorthogonal wavelets. In the extensive work on coiflet-type wavelets presented in the Thesis [78], there are two chapters on biorthogonal wavelets that detail its construction using Vandermonde matrices. The results are similar to those presented in [70]. More recently, [36] completed the results.

According to [36], if $1 \leq a \leq N$, the Coifman biorthogonal wavelet system of order N with minimum length $2N + 1$ synthesis filter, is obtained as follows:

- The nonzero coefficients of the synthesis filter are given by:

$$h_0 = \frac{1}{\sqrt{2}}, \quad h_{1-2a+2k} = \frac{\prod_{m=0, m \neq k}^N (2(m-a) + 1)}{2^N \sqrt{2} \prod_{m=0, m \neq k}^N (m-k)}, \quad k = 0, \dots, N \quad (2.200)$$

where a is an integer.

- The coefficients of the analysis filter are calculated in recursively form:

$$\tilde{h}_{2k} = \sqrt{2} \left[\delta_{k0} - \sum_n h_{1+2n} h_{1+2(n-k)} \right], \quad \tilde{h}_{2k+1} = h_{2k+1} \quad (2.201)$$

With this procedure, all the list of biorthogonal Coifman wavelet systems given in [70] can be obtained, and other new examples can be found.

An important aspect of the results attained by [36, 70, 78] is that all the filter coefficients are in the form of fractions of integers.

2.5.2.2 Filter Banks and Biorthogonal Wavelets

Clearly, wavelets are linked to perfect reconstruction filter banks. Actually, there are books that arrive to the construction of orthogonal and biorthogonal wavelets in the framework of filter banks. For instance, [77] presents in this way a fairly complete treatment of orthogonal, biorthogonal, etc. wavelets, and other related topics. It extends the scenario by considering the use of IIR filters, instead of the FIR filters. As one interesting example, a wavelet system based on the Butterworth filter is discussed. With IIR filters, orthogonal symmetric wavelets (not Haar) can be designed.

2.5.2.3 Getting Rational Coefficients

In most of the tables of coefficients already shown, the coefficients are irrational. This is not convenient for the use of digital processors. The research has proposed some solutions, for instance the procedure of [68]. This article belongs to the filter bank context. It starts with analysis H_0 and synthesis F_0 low-pass filters, and high-pass filters given by:

$$H_1(z) = z^{-1} F_0(-z); \quad F_1(z) = z H_0(-z) \quad (2.202)$$

The product of the low-pass filters $D(z) = H_0(z) F_0(z)$ must satisfy the PR condition, $D(z) + D(-z) = 2$. Many wavelet systems have been obtained by factorizing the *Lagrange half-band filter (LHBF)*:

$$D(z) = L_K(z) = z^K \left(\frac{1 + z^{-1}}{2} \right)^{2K} R_K(z) \quad (2.203)$$

where:

$$R_K(z) = \sum_{n=0}^{K-1} \binom{K+n-1}{n} \left(\frac{2 - (z + z^{-1})}{4} \right)^n \quad (2.204)$$

The transition bandwidth of the LHBF is proportional to \sqrt{N} , where $N = 2K - 1$. The LHBF is a maximally flat filter. The LHBF is linked to Lagrange interpolation formulae. Obviously, the orthogonal and biorthogonal wavelets of Daubechies are related to factorizations of the LHBF. Using the change of variable: $x = \cos(\omega) = (1/2)(z + z^{-1})$, the LHBF can be written as follows:

$$L'_K(x) = (x + 1)^K R'_K(x) \quad (2.205)$$

The presence of irrational coefficients is due to $R'_K(x)$. The idea of [68] is to allow one degree of freedom by using:

$$L_K^M(x) = (x + 1)^{K-1} (x + \alpha) R_K^M(x) \quad (2.206)$$

and then try to get rational coefficients. For instance, let us factorize the $K=4$ LHBF:

$$H_0 = k_1 (x + 1)(x^3 + Ax^2 + Bx + C) \quad (2.207)$$

$$F_0 = k_2 (x + 1)^2(x + \alpha) \quad (2.208)$$

(k_1, k_2 are normalizing constants)

Then, one gets the product $D(x) = H_0(x) F_0(x)$ and applies the PR condition. The terms in x^2, x^4, x^6 , must be zero. Three equations are obtained, and finally:

$$A = -(3 + \alpha) \quad (2.209)$$

$$B = \frac{9\alpha^3 + 35\alpha^2 + 45\alpha + 24}{3\alpha^2 + 9\alpha + 8} \quad (2.210)$$

$$C = -\frac{8(1 + \alpha)^3}{3\alpha^2 + 9\alpha + 8} \quad (2.211)$$

Setting $\alpha = -1.6848$ will give the CDF 9/7. But with values such $-3/2$, or $-5/3$, or $-8/5$, or -2 , biorthogonal filters can be obtained with all coefficients rational. In particular, with $\alpha = -5/3$ the corresponding filter bank has responses close to the CDF 9/7.

2.5.2.4 Getting Families by Using One Parameter

In the previous paragraphs the introduction of one parameter, α , has proved to be useful. This idea could be extended for the parameterized construction of many families of biorthogonal wavelets. The contribution of [43] proposes to use the following low-pass analysis filter, which has $2l$ vanishing moments (case I), or $2l + 1$ vanishing moments (case II):

- (case I):

$$H_0(\omega) = \cos^{2k}(\omega/2) (\alpha + (1 - \alpha) \cos \omega) \quad (2.212)$$

- (case II):

$$H_0(\omega) = e^{-j\omega/2} \cos^{2k+1}(\omega/2) (\alpha + (1 - \alpha) \cos \omega) \quad (2.213)$$

The dual low-pass filter is found to be:

- (case I):

$$F_0(\omega) = \cos^{2m}(\omega/2) Q(\sin^2(\omega/2)) \quad (2.214)$$

- (case II):

$$F_0(\omega) = e^{-j\omega/2} \cos^{2m+1}(\omega/2) Q(\sin^2(\omega/2)) \quad (2.215)$$

The filter has m vanishing moments.

Now, it is shown that, if we define:

$$Q() = \sum_{n=0}^L q_n (\sin^2(\omega/2))^n \quad (2.216)$$

Then the filter bank satisfies the PR condition if:

$$q_n = \sum_{k=0}^L \binom{L+n-k-1}{L-1} (2(1-\alpha))^k, \quad n = 0, 1, \dots, L-1 \quad (2.217)$$

and:

$$q_L = \frac{1}{2\alpha} \left\{ \sum_{k=0}^L \binom{2L-k-1}{L-1} (2(1-\alpha))^k + (1-2\alpha) \sum_{n=0}^{L-1} q_n \right\} \quad (2.218)$$

where $L = l + m$ in case I, or $L = l + m + 1$ in case II.

Once this parameterization is created, [43] constructs ten families of biorthogonal wavelets. The CDF 9/7 is obtained with $\alpha = 2.460348$. Families with rational coefficients are found, and also families with binary coefficients.

2.5.2.5 Using Bernstein Polynomials

Given a function $f(x)$ on the interval $[0,1]$, the N th order Bernstein polynomial approximation to $f(x)$ is:

$$B_N(f, x) = \sum_{k=0}^N f(k/N) \cdot \binom{N}{k} x^k (1-x)^{N-k} \quad (2.219)$$

Notice that the approximation uses only $(N + 1)$ samples of $f(x)$ at equally spaced values of x . The approximation has nice mathematical properties.

Now suppose that $f(x)$ express the frequency response of a desired low-pass filter. The description of that response could be done as follows:

$$f\left(\frac{k}{2N-1}\right) = \begin{cases} 1 & k = 0 \\ 1 - \alpha_k & 1 \leq k \leq N-1 \\ \alpha_k & N \leq k \leq 2(N-1) \\ 0 & k = 2N-1 \end{cases} \quad (2.220)$$

where $0 \leq \alpha_k < 0.5$, and $\alpha_k = \alpha_{2N-k-1}$.

The approximation by Bernstein polynomial would be:

$$\begin{aligned} B_{2N-1}(f, x) &= \sum_{k=0}^{N-1} \binom{2N-1}{k} x^k (1-x)^{2N-1-k} - \\ &- \sum_{k=1}^{N-1} \alpha_k \binom{2N-1}{k} x^k (1-x)^{2N-1-k} + \\ &+ \sum_{k=N}^{2(N-1)} \alpha_k \binom{2N-1}{k} x^k (1-x)^{2N-1-k} \end{aligned} \quad (2.221)$$

By substituting $x = -(1/4)z(1-z^{-1})^2$ a half band filter $R(z)$ is obtained. A PR filter bank can be easily found by factorization.

It has been shown (see for instance [79]) that the number of zeros in the Bernstein coefficients (that is, the α_k) determines the vanishing moments of wavelets. Also, the number of ones in Bernstein coefficients determines the filter flatness.

This method has been introduced by [16]. It would be convenient for the reader to see [12] in order to know more about Bernstein polynomials. The article [22] extends the application of the method, designing orthonormal and biorthogonal wavelet filters. Relevant theoretical aspects are covered in [79]. Indeed, the frequency response of $R(z)$ should be nonnegative. This is ensured when using $0 \leq \alpha_k < 0.5$. If you

try other values, nonnegativity is not guaranteed. The research has proposed some alternatives to overcome this problem [69, 80, 81].

Once the way is open for the use of interpolation functions, new proposals could be expected. For instance, using Lagrange interpolation, etc.

2.5.2.6 Factorization of a General Half-Band Filter

A more radical proposal is to design the biorthogonal filter bank starting from a general half-band filter, to be factorized. The target is to have more control on the frequency response of the filters.

An example of this approach is [53]. The general half-band filter would be:

$$D(z) = a_0 + a_2 z^{-2} + \dots + a_{(K/2)-1} z^{-(K/2)-1} + z^{-K/2} + a_{(K/2)-1} z^{-(K/2)+1} + \dots + a_0 z^{-K} \quad (2.222)$$

A M th order flatness is imposed:

$$\frac{d^i}{d\omega^i} D(\omega)|_{\omega=\pi} = 0, \quad i = 0, 1, \dots, M \quad (2.223)$$

with $M < ((K/2) + 1)$. We do not impose the maximum number of zeros (which would mean maximum flatness). In this way, some design freedom is gained. After imposing the specified flatness, $D(z)$ can be expressed in terms of some independent parameters.

Then, $D(z)$ is factorized into $H_0(z)$ and $F_0(z)$. An optimization criterion is established, considering the desired frequency response of the two filters (like, for instance, not to be far from 1 in the pass band). Finally, a standard optimization method is applied to get the parameters that optimize the criterion.

2.6 Continuous Wavelets

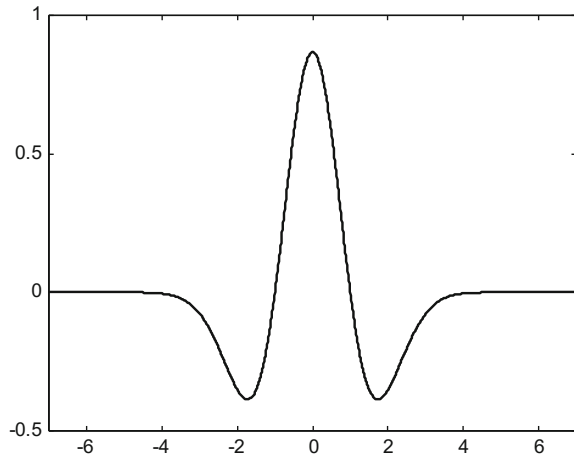
Some examples of continuous wavelets have already been considered in this chapter, like for instance the Shannon wavelet. Now, let us see other instances of this type of wavelets.

An important advantage of the next wavelets is that they have explicit definition formulae.

2.6.1 The Mexican Hat Wavelet

The Mexican hat wavelet, also denoted as Ricker wavelet, is proportional to the second derivative of the Gaussian density function:

Fig. 2.51 Mexican hat wavelet



$$\psi(t) = \left(\frac{2}{\sqrt{3}} \cdot \frac{1}{\sqrt[4]{\pi}} \right) (1 - t^2) e^{-\frac{t^2}{2}} \quad (2.224)$$

Figure 2.51 shows the Mexican hat wavelet. The figure has been generated with the Program 2.29.

Program 2.29 Display of Mexican hat wavelet

```
% Display of Mexican hat wavelet
t=-7:0.01:7;
C=2/(sqrt(3)*sqrt(sqrt(pi)));
psi=C*(1-t.^2).*exp(-0.5*t.^2);
plot(t,psi,'k'); %plots the wavelet
axis([-7 7 -0.5 1]);
title('Mexican hat wavelet');
```

There is a complex Mexican hat wavelet, proposed by Add. Its Fourier transform is:

$$\Psi(\omega) = \left(\frac{2\sqrt{2}}{\sqrt{3}} \cdot \frac{1}{\sqrt[4]{\pi}} \right) \omega^2 e^{-\frac{\omega^2}{2}} \quad (2.225)$$

for $\omega \geq 0$; and $\Psi(\omega) = 0$ otherwise.

The Mexican hat wavelet is a special case of *Hermitian wavelets*, which are derivatives of a Gaussian.

2.6.2 The Morlet Wavelet

Consider as wavelet candidate a sine wave multiplied by a Gaussian envelope (this is similar to the complex Gaussian modulated pulse (GMP) used in the Gabor expan-

sion):

$$f(t) = e^{j\omega_0 t} e^{-\frac{t^2}{2}} \quad (2.226)$$

The Fourier transform is:

$$F(\omega) = e^{-\frac{(\omega - \omega_0)^2}{2}} \quad (2.227)$$

Strictly speaking, the above candidate does not satisfy the admissibility condition, since $F(0)$ is not zero ($f(t)$ does not have zero mean).

Let us introduce a correction factor to get zero mean:

$$\psi(t) = (e^{-j\omega_0 t} - e^{-\frac{\omega_0^2}{2}}) e^{-\frac{t^2}{2}} \quad (2.228)$$

This is the complex Morlet wavelet. Its Fourier transform is:

$$\Psi(\omega) = e^{-\frac{(\omega - \omega_0)^2}{2}} - e^{-\frac{\omega_0^2}{2}} e^{-\frac{\omega^2}{2}} \quad (2.229)$$

Since the Morlet wavelet is complex, the wavelet transform of a signal is plotted as in Bode diagrams, magnitude and phase. The frequency ω_0 is usually chosen equal to five, to make the ratio of main peak and neighbour peak of the wavelet be near 2. For $\omega_0 = 5$ the correction factor is very small and can be neglected. Some authors take the real part, and say that the Morlet wavelet is:

$$\psi(t) = C e^{-t^2/2} \cos(5t) \quad (2.230)$$

Figure 2.52 shows the magnitude, real part and imaginary part of the complex Morlet wavelet. The figure has been generated with the Program 2.30.

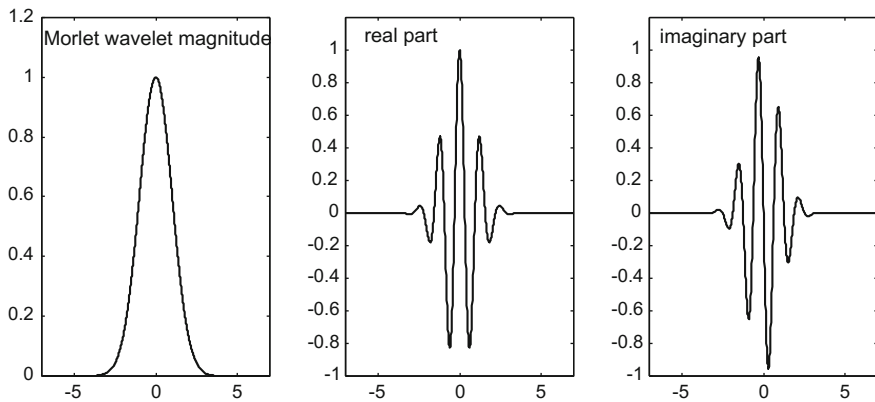


Fig. 2.52 The complex Morlet wavelet

Program 2.30 Display of complex Morlet wavelet

```
% Display of complex Morlet wavelet
t=-7:0.01:7;
w0=5;
cc=exp(-0.5*(w0^2));
aux=exp(-j*w0*t)-cc;
psi=aux.*exp(-0.5*t.^2);
%plots the wavelet
subplot(1,3,1)
plot(t,abs(psi),'k'); %magnitude
axis([-7 7 0 1.2]);
title('Morlet wavelet magnitude');
subplot(1,3,2)
plot(t,real(psi),'k'); %magnitude
axis([-7 7 -1 1.2]);
title('real part');
subplot(1,3,3)
plot(t,imag(psi),'k'); %magnitude
axis([-7 7 -1 1.2]);
title('imaginary part');
```

2.6.3 Complex B-Spline Wavelets

Complex B-spline wavelets are obtained with the following formula:

$$\psi(t) = \sqrt{\frac{\omega_b}{2\pi m}} \left(\frac{\sin(\frac{\omega_b t}{2m})}{(\frac{\omega_b t}{2m})} \right)^m e^{j\omega_c t} \quad (2.231)$$

where m is the order of the wavelet, ω_c is the central frequency of the wavelet and ω_b is a window parameter.

For $m = 1$ we obtain the Shannon wavelet.

Figure 2.53 shows the magnitude, real part and imaginary part of the complex B-spline wavelet. The figure has been generated with the Program 2.31.

Program 2.31 Display of complex B-spline wavelet

```
% Display of complex B-spline wavelet
t=-9:0.01:9;
m=2; wc=8; wb=5;
cc=sqrt(wb/(2*pi*m));
aa=(wb*t)/(2*m); aux=sin(aa)./aa;
psi=cc*(aux.^m).*exp(j*wc*t);
%plots the wavelet
subplot(1,3,1)
plot(t,abs(psi),'k'); %magnitude
```

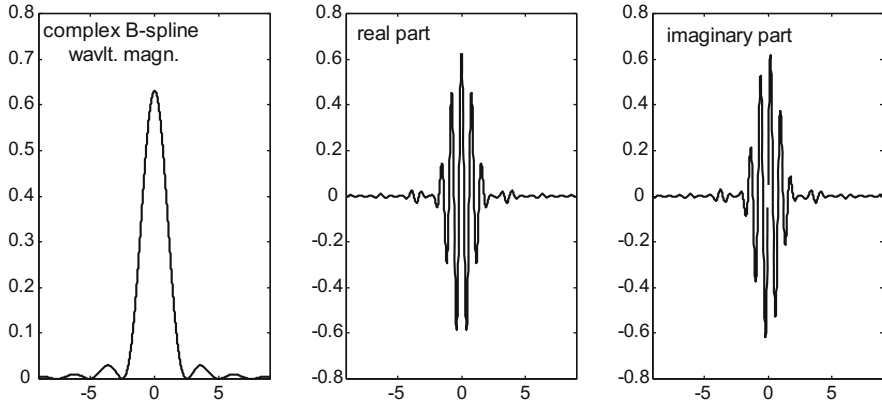


Fig. 2.53 The complex B-spline wavelet

```
axis([-9 9 0 0.8]);
title('complex B-spline wavlt. magn. ');
subplot(1,3,2)
plot(t,real(psi),'k'); %magnitude
axis([-9 9 -0.8 0.8]);
title('real part');
subplot(1,3,3)
plot(t,imag(psi),'k'); %magnitude
axis([-9 9 -0.8 0.8]);
title('imaginary part');
```

2.7 Continuous Wavelet Transform (CWT)

The continuous Wavelet transform (CWT) is:

$$W_y(\tau, s) = \langle y(t), \psi(t) \rangle = \int_{-\infty}^{\infty} y(\tau) \frac{1}{\sqrt{|s|}} \psi^*\left(\frac{t-\tau}{s}\right) dt \quad (2.232)$$

The original signal can be reconstructed with the inverse transform:

$$y(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_y(\tau, s) \frac{1}{\sqrt{|s|}} \psi\left(\frac{t-\tau}{s}\right) d\tau \frac{ds}{s^2} \quad (2.233)$$

where:

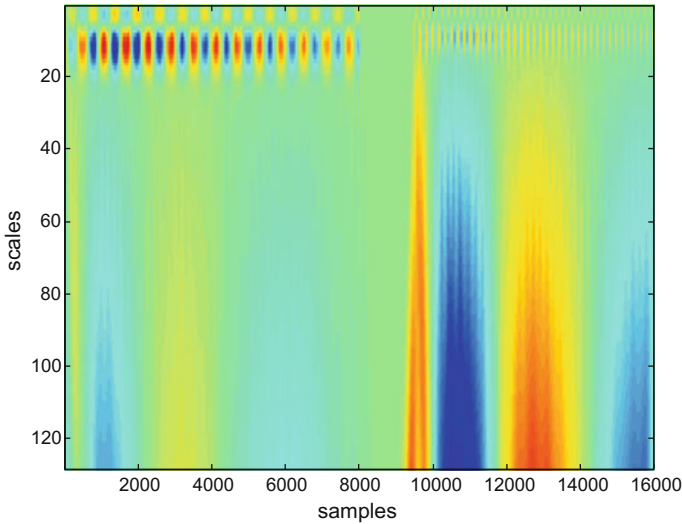


Fig. 2.54 Scalogram of doorbell using mexican hat

$$C\psi = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega \quad (2.234)$$

and Ψ is the Fourier transform of ψ .

See [3] for an extensive introduction of CWT.

Since MATLAB has optimized so much the speed of the Fourier transform, it is convenient to implement the CWT in the Fourier domain. This is demonstrated in the next example, where a typical DIIN-DOON doorbell is analyzed.

Figure 2.54 shows the analysis result in the form of a continuous scalogram. This figure has been generated with the Program 2.32. The Mexican hat wavelet has been chosen for this example; the reader is invited to use any other wavelet.

The execution of the program requires about one minute.

Program 2.32 Continuous wavelet analysis with Mexican hat

```
%Continuous wavelet analysis with Mexican hat
%harp signal
[y1,fs1]=wavread('doorbell1.wav'); %read wav file
y1=y1(1:16000); %select part of the signal
Nss=length(y1);
soundsc(y1,fs1); %hear wav
t=(-Nss/2):(Nss/2)-1; %normalized time intervals set
C=2/(sqrt(3)*sqrt(sqrt(pi))); %Mexican hat constant
NS=128; %number of scales
CC=zeros(NS,Nss); %space for wavelet coeffs.
for ee=1:NS,
```

```

s=(ee*1); %scales
%the scaled Mexican hat
ts=t/s;
psi=C*(1-(ts.^2).*exp(-0.5*ts.^2));
%CWT
CC(ee,:)=abs(iff(fft(psi).*fft(y1')));
end
figure(1)
imagesc(CC);
title('Scalogram of doorbell');
xlabel('samples'); ylabel('scales');

```

2.8 The Lifting Method and the Second Generation Wavelets

The lifting method is a recent alternative for the construction of wavelet bases, or for the computation of wavelet transforms. It constitutes the basis for the development of the so-called second generation wavelets [35, 67].

Let us first introduce the main ideas. Consider a signal $y(k)$. Let us split it into two sets, one is formed by the even indexed samples and the other by the odd indexed samples:

$$\bar{y}_e = y(2k), \text{ even indexed samples} \quad (2.235)$$

$$\bar{y}_o = y(2k + 1), \text{ odd indexed samples} \quad (2.236)$$

Notice that these are the polyphase components (see Sect. 1.3.1.).

Usually both sets are correlated. In fact the even samples could be used to predict the intermediate odd samples, or vice versa.

Suppose the even samples are used, with an operator $P()$, to predict the odd samples. There would be differences (details) between the actual and the predicted values:

$$\bar{d} = \bar{y}_o - P(\bar{y}_e) \quad (2.237)$$

Given the details and the even samples, it is always possible to recover the odd samples:

$$\bar{y}_o = P(\bar{y}_e) + \bar{d} \quad (2.238)$$

The operation of computing a prediction and recording the details is a ‘lifting step’.

A simple predictor is the average of the even neighbours:

$$d_k = y_{2k+1} - \frac{(y_{2k} + y_{2k+2})}{2} \quad (2.239)$$

A second lifting step, which obtains smoothed data, could be the following:

$$\bar{a} = \bar{y}_e + U(\bar{d}) \quad (2.240)$$

where $U()$ is an update operator.

The original values can always be recovered with:

$$\bar{y}_e = \bar{a} - U(\bar{d}) \quad (2.241)$$

An update example, which is able to keep the moving average of the signal, is:

$$a_k = y_{2k} + \frac{(d_{k-1} + d_k)}{4} \quad (2.242)$$

Figure 2.55 shows a conceptual diagram of the complete scheme.

It is clear that the scheme is invertible, so it can be used for a perfect reconstruction filter bank. Let us specify more about this filter. The separation of even and odd samples is done with the so-called Lazy wavelet, as shown in the next Fig. 2.56.

The combination of Lazy wavelet, and the predict and update lifting steps, yields the PR filter bank represented in Fig. 2.57.

The filter bank just represented can be also seen as a polyphase transform. Let us translate the filtering steps to matrix algebra. To begin with, the next expression corresponds to Fig. 2.58.

$$\begin{pmatrix} x_e \\ x_o \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -P & 1 \end{pmatrix} \begin{pmatrix} y_e \\ y_o \end{pmatrix} \quad (2.243)$$

And the next expression corresponds to Fig. 2.59.

Fig. 2.55 Conceptual diagram of the lifting method

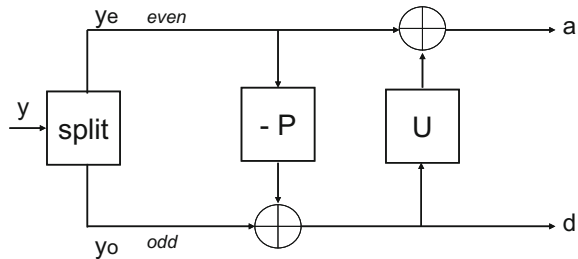
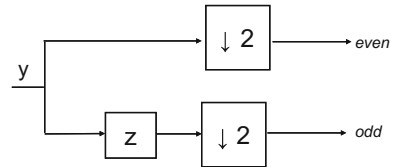


Fig. 2.56 The Lazy wavelet filter bank



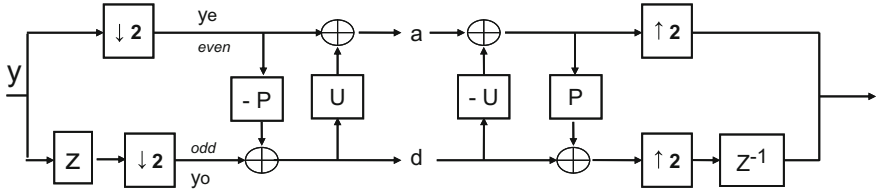


Fig. 2.57 The complete filter bank

Fig. 2.58 A prediction branch

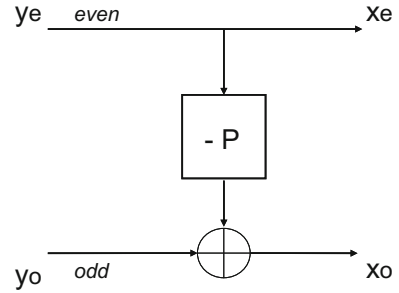
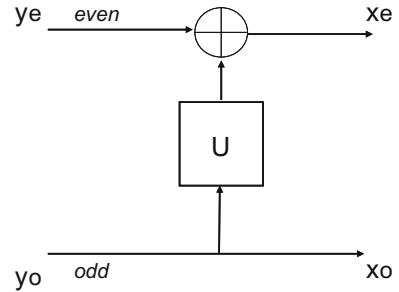


Fig. 2.59 An update branch



$$\begin{pmatrix} x_e \\ x_o \end{pmatrix} = \begin{pmatrix} 1 & U \\ 0 & 1 \end{pmatrix} \begin{pmatrix} y_e \\ y_o \end{pmatrix} \quad (2.244)$$

Usually a signal scaling step is added, so the combination of prediction, update and scaling yields the following expression:

$$\begin{pmatrix} x_e \\ x_o \end{pmatrix} = \begin{pmatrix} c & 0 \\ 0 & 1/c \end{pmatrix} \begin{pmatrix} 1 & U \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -P & 1 \end{pmatrix} \begin{pmatrix} y_e \\ y_o \end{pmatrix} = H_p \begin{pmatrix} y_e \\ y_o \end{pmatrix} \quad (2.245)$$

where H_p is called the 'polyphase matrix'. Likewise, in the synthesis part of the filter bank, there will be re-scaling, and the inverse of update and prediction. The corresponding expression is:

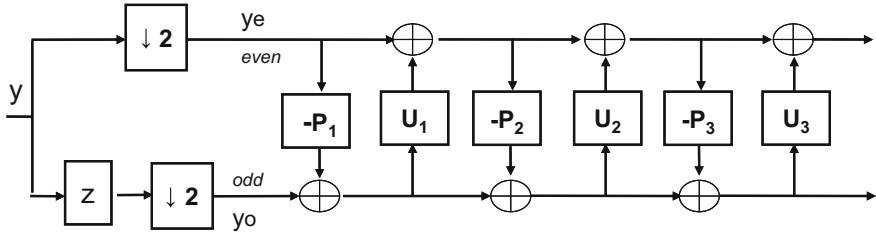


Fig. 2.60 Lifting steps

$$\begin{pmatrix} y_e \\ y_o \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P & 1 \end{pmatrix} \begin{pmatrix} 1 & -U \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1/c & 0 \\ 0 & c \end{pmatrix} \begin{pmatrix} x_e \\ x_o \end{pmatrix} = F_p \begin{pmatrix} x_e \\ x_o \end{pmatrix} \quad (2.246)$$

It is easy to check that the product of the F_p and H_p matrices is the identity matrix. Also $\det(H_p) = \det(F_p) = 1$.

In general it is possible to continue with the chain of lifting steps, as illustrated in Fig. 2.60:

Again, the chain of lifting steps can be represented with a polyphase matrix H_p , so the analysis part of the filter bank can be drawn as in Fig. 2.61 (similarly, *mutatis mutandis*, with the synthesis part and the matrix F_p):

Let us write the polyphase matrix as:

$$H_p(z^2) = \begin{pmatrix} H_{00}(z^2) & H_{01}(z^2) \\ H_{10}(z^2) & H_{11}(z^2) \end{pmatrix} \quad (2.247)$$

It is possible to relate the filters with the lifting branches (FLB) and the filter banks with parallel structure (FPS) (as depicted in Fig. 2.62).

It can be shown that both filters, FLB and FPS , are equal when:

Fig. 2.61 Analysis part of the filter

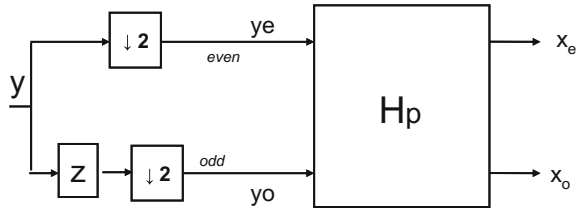
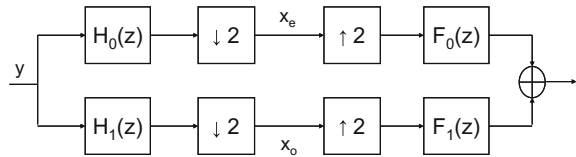


Fig. 2.62 A FPS filter bank



$$H_0(z) = H_{00}(z^2) + z H_{01}(z^2) \quad (2.248)$$

$$H_1(z) = H_{10}(z^2) + z H_{11}(z^2) \quad (2.249)$$

$$F_0(z) = F_{00}(z^2) + z^{-1} F_{01}(z^2) \quad (2.250)$$

$$F_1(z) = F_{10}(z^2) + z^{-1} F_{11}(z^2) \quad (2.251)$$

In consequence the wavelet transform can be given in three equivalent forms: lifting steps, polyphase matrix, or *FPS* filter bank. If a perfect reconstruction (PR) filter can be represented by a polyphase matrix H_p , then it can be implemented with lifting steps, which are obtained by factorization of H into triangle matrices.

The filter pair $(H_0(z), H_1(z))$ is *complementary* if the corresponding polyphase matrix H_p has $\det(H_p) = 1$.

If $(H_0(z), H_1(z))$ is complementary, $(F_0(z), F_1(z))$ is also complementary.

2.8.1 Example

Let us take as example the Daubechies wavelet introduced in Sect. 2.4.3.2. Figure 2.63 shows how it is implemented using lifting steps.

The lifting equations are the following:

$$a_k^{(1)} = y_{2k} + \sqrt{3} y_{2k+1} \quad (2.252)$$

$$d_k^{(1)} = y_{2k+1} - \frac{\sqrt{3}}{4} a_k^{(1)} - \frac{(\sqrt{3}-2)}{4} a_{k-1}^{(1)} \quad (2.253)$$

$$a_k^{(2)} = a_k^{(1)} - d_{k+1}^{(1)} \quad (2.254)$$

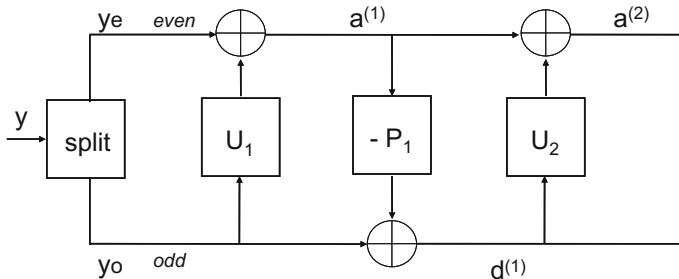


Fig. 2.63 Daubechies 4 wavelet lifting steps

Then, the polyphase matrix is:

$$H_p(z) = \begin{pmatrix} \frac{\sqrt{3}-1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}+1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & -z \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{\sqrt{3}}{4} & -\frac{\sqrt{3}-2}{4} z^{-1} \end{pmatrix} \begin{pmatrix} 1 & \sqrt{3} \\ 0 & 1 \end{pmatrix} \quad (2.255)$$

(with scaling $c = \frac{\sqrt{3}-1}{\sqrt{2}}$)

Develop the equation:

$$\begin{aligned} H_p(z) &= \begin{pmatrix} \frac{3-\sqrt{3}}{4\sqrt{2}} z + \frac{1+\sqrt{3}}{4\sqrt{2}} & \frac{1-\sqrt{3}}{4\sqrt{2}} z + \frac{3+\sqrt{3}}{4\sqrt{2}} \\ -\frac{(3+\sqrt{3})}{4\sqrt{2}} & -\frac{(1-\sqrt{3})}{4\sqrt{2}} z^{-1} \end{pmatrix} = \\ &= \begin{pmatrix} H_{00}(z^2) & H_{01}(z^2) \\ H_{10}(z^2) & H_{11}(z^2) \end{pmatrix} \end{aligned} \quad (2.256)$$

In particular:

$$\begin{aligned} H_0(z) &= H_{00}(z^2) + z H_{01}(z^2) = \\ &= \frac{3-\sqrt{3}}{4\sqrt{2}} z^2 + \frac{1+\sqrt{3}}{4\sqrt{2}} + \frac{1-\sqrt{3}}{4\sqrt{2}} z^3 + \frac{3+\sqrt{3}}{4\sqrt{2}} z = \\ &= \frac{1+\sqrt{3}}{4\sqrt{2}} + \frac{3+\sqrt{3}}{4\sqrt{2}} z + \frac{3-\sqrt{3}}{4\sqrt{2}} z^2 + \frac{1-\sqrt{3}}{4\sqrt{2}} z^3 \end{aligned} \quad (2.257)$$

$$\begin{aligned} H_1(z) &= H_{10}(z^2) + z H_{11}(z^2) = \\ &= -\frac{3+\sqrt{3}}{4\sqrt{2}} - \frac{1-\sqrt{3}}{4\sqrt{2}} z^{-2} + \frac{1+\sqrt{3}}{4\sqrt{2}} z + \frac{3-\sqrt{3}}{4\sqrt{2}} z^{-1} = \\ &= -\frac{1-\sqrt{3}}{4\sqrt{2}} z^{-2} + \frac{3-\sqrt{3}}{4\sqrt{2}} z^{-1} - \frac{3+\sqrt{3}}{4\sqrt{2}} + \frac{1+\sqrt{3}}{4\sqrt{2}} z \end{aligned} \quad (2.258)$$

Note that the coefficients in the expression of $H_0(z)$ are:

$$\begin{aligned} h_0(0) &= \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_0(1) = \frac{3+\sqrt{3}}{4\sqrt{2}}, \\ h_0(2) &= \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_0(3) = \frac{1-\sqrt{3}}{4\sqrt{2}} \end{aligned} \quad (2.259)$$

(these values already given in Sect. 2.3.3).

A segment of a long sonar signal has been chosen to present now an example of wavelet analysis using lifting for Daubechies 4. Program 2.33 shows a MATLAB implementation of the lifting scheme, based on [35]. The result is depicted in Fig. 2.64.

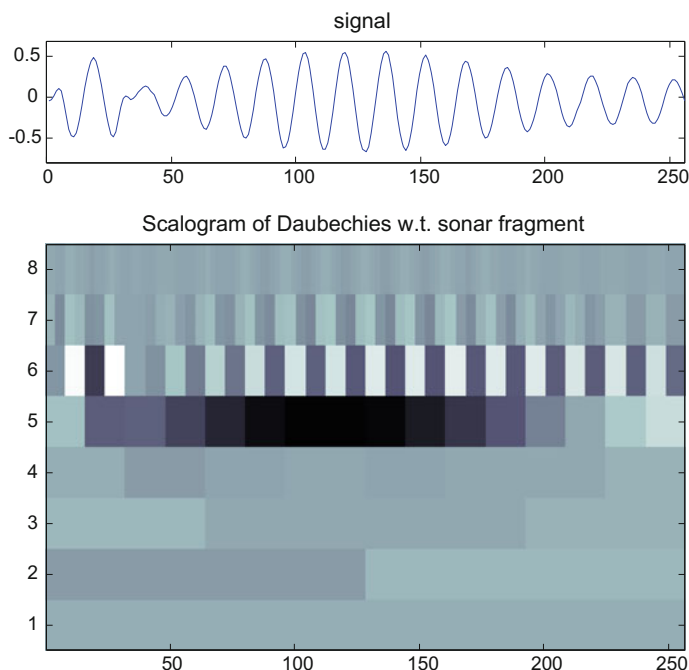


Fig. 2.64 Scalogram of sonar fragment using lifting Daubechies 4

Program 2.33 Lifting example: analysis of sonar signal

```
% Lifting example
% analysis of sonar signal
% Daubechies 4 Wavelet
% Plot of signal and scalogram
%The sonar signal
[y1,fs1]=wavread('sonar1.wav'); %read wav file
ta=7815; tb=ta+255;
sg=y1(ta:tb);
Nss=length(sg);%number of signal samples
tiv=1/fs1;
duy=(Nss-1)*tiv; %duration of signal
tss=0:tiv:duy; %time intervals set
Ts=tiv; %sampling period
%analysis of the signal with wavelets
y=sg;
a=y';
aux=0; cq=sqrt(3);
K=8; %number of scales (256=2^8)
%wavelet calculus using lifting
NN=Nss;
```

```

for n=1:K,
L=length(a); L2=L/2;
a1=a(1:2:L-1)+(cq*a(2:2:L));
d1=a(2:2:L)-((cq/4)*a1)-(((cq-2)/4)*[a1(L2) a1(1:(L2-1))]);
a2=a1-[d1(2:L2) d1(1)];
a=((cq-1)/sqrt(2))*a2;
d=((cq+1)/sqrt(2))*d1;
aux=[d,aux];
dd(K+1-n,1:NN/2)=d;
NN=NN/2;
end;
wty=[a,aux(1:(end-1))];
%preparing for scalogram
S=zeros(K,Nss); %space for S(j,k) scalogram coefficients
for n=1:K,
q=2^(n-1); L=Nss/q;
for m=1:q,
R=(1+(L*(m-1))):(L*m); %index range
S(n,R)=dd(n,m);
end;
end;
%figure
subplot('position',[0.04 0.77 0.92 0.18])
plot(y);
axis([0 256 1.2*min(y) 1.2*max(y)]);
title('signal');
subplot('position',[0.04 0.05 0.92 0.6])
imagesc(S); colormap('bone');
title('Scalogram of Daubechies w.t. sonar fragment');
h=gca; set(h,'YDir','normal');

```

To complete the example, the original signal is recovered by the Program 2.34. It is clearly seen in this program how easy is to invert the lifting process, re-using and re-ordering the lifting equations. Figure 2.65 shows the recovered sonar signal, which is practically identical to the original signal.

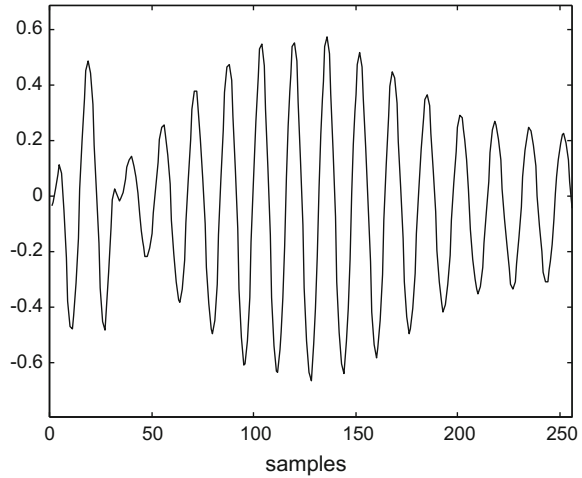
Program 2.34 Inverse of the Lifting

```

% inverse of the Lifting
% Sonar signal
% Daubechies 4 wavelet
% use after lifting analysis (it needs wty)
L=length(wty); %length of the DWT
sg=zeros(1,L);
K=8; %number of scales
cq=sqrt(3);
a=wty(1);
for n=1:K,
wd=2^(n-1);
bg=1+wd; fl=bg+wd-1;

```

Fig. 2.65 Recovered sonar fragment using lifting Daubechies 4



```
d=wty(bg:fl);
d1=d/((cq+1)/sqrt(2));
a2=a/((cq-1)/sqrt(2));
a1=a2+[d1(2:end) d1(1)];
sg(2:2:(2^n))=d1+((cq/4)*a1)+(((cq-2)/4)*[a1(end) a1(1:end-1)]);
sg(1:2:(2^n)-1)=a1-(cq*sg(2:2:(2^n)));
a=sg(1:(2^n));
end;
figure(1)
plot(sg, 'k');
axis([0 256 1.2*min(sg) 1.2*max(sg)]);
xlabel('samples');
title('the recovered signal');
```

2.8.2 Decomposition into Lifting Steps

By writing any FIR wavelet or PR filter bank in the polyphase form, it can be confirmed that in every case it can be decomposed into lifting steps (this is an indication of the power of the lifting method) [23, 27, 46, 76].

The decomposition into lifting steps can be done by applying the Euclidean algorithm on Laurent polynomials.

A Laurent polynomial is an expression of the form:

$$p(z) = \sum_{k=a}^b c_k z^{-k} \quad (2.260)$$

The degree of a Laurent polynomial is $b - a$. Thus the following z-transform:

$$h(z) = \sum_{k=a}^b h_k z^{-k} \quad (2.261)$$

is a Laurent polynomial.

The Euclidean algorithm is a method for finding the greatest common divisor (gcd) of two integers. Suppose two integers a and b , with $|a| > |b|$ and $b \neq 0$. By division one can obtain a quotient q_1 and a remainder r_1 (if $r_1 = 0$, then $\gcd(a, b) = b$):

$$a = b q_1 + r_1 \quad (2.262)$$

Rewriting this:

$$r_1 = a - b q_1 \quad (2.263)$$

It is evident that any common factor of a and b is a factor of r_1 , and it is also a common factor of r_1 and b : $\gcd(a, b) = \gcd(b, r_1)$

Let us continue with the division:

$$b = r_1 q_2 + r_2 \quad (2.264)$$

It is clear that $\gcd(b, r_1) = \gcd(r_1, r_2)$.

Iterating the process, a $r_i = 0$ will be obtained, and:

$$r_{i-2} = r_{i-1} q_i \quad (2.265)$$

So $\gcd(a, b) = \gcd(r_{i-2}, r_{i-1}) = r_{i-1}$.

For example, the gcd of 182 and 34:

$$\begin{aligned} 182 &= 34 \cdot [5] + 12 \\ 34 &= 12 \cdot [2] + 10 \\ 12 &= 10 \cdot [1] + 2 \\ 10 &= 2 \cdot [5] \end{aligned} \quad (2.266)$$

The gcd is 2.

The procedure can be implemented in the following computational recursive form:

$$\begin{aligned} q_{i+1}, r_{i+1} &\leftarrow a_i / b_i \\ a_{i+1} &\leftarrow b_i \\ b_{i+1} &\leftarrow r_{i+1} \end{aligned} \quad (2.267)$$

For Laurent polynomials division with remainder is possible, so given two non-zero Laurent polynomials $a(z)$, with degree n , and $b(z)$, with degree m ($n \geq m$), we can write:

$$a(z) = b(z)q(z) + r(z) \quad (2.268)$$

where $q(z)$ and $r(z)$ are also Laurent polynomials. Notice that the division is not necessarily unique.

The Euclidean algorithm can be applied to Laurent polynomials, starting from $a_0(z) = a(z)$ and $b_0(z) = b(z)$, $i = 0$, and according with the following iteration:

$$\begin{aligned} q_{i+1}(z), r_{i+1}(z) &\leftarrow a_i(z) / b_i(z) \\ a_{i+1}(z) &\leftarrow b_i(z) \\ b_{i+1}(z) &\leftarrow r_{i+1}(z) = a_i(z) - b_i(z)q_{i+1}(z) \end{aligned} \quad (2.269)$$

which in matrix form is:

$$\begin{bmatrix} a_{i+1}(z) \\ b_{i+1}(z) \end{bmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_{i+1}(z) \end{pmatrix} \begin{bmatrix} a_i(z) \\ b_i(z) \end{bmatrix} \quad (2.270)$$

After N iterations we obtain a $b_N(z) = 0$, so $a_N(z)$ is the gcd of $a(z)$ and $b(z)$. Therefore:

$$\begin{bmatrix} a_N(z) \\ 0 \end{bmatrix} = \prod_{i=N}^1 \begin{pmatrix} 0 & 1 \\ 1 & -q_i(z) \end{pmatrix} \begin{bmatrix} a(z) \\ b(z) \end{bmatrix} \quad (2.271)$$

With matrix inversion and side exchanging:

$$\begin{bmatrix} a(z) \\ b(z) \end{bmatrix} = \prod_{i=1}^N \begin{pmatrix} q_i(z) & 1 \\ 1 & 0 \end{pmatrix} \begin{bmatrix} a_N(z) \\ 0 \end{bmatrix} \quad (2.272)$$

Let us apply the Euclidean algorithm to our lifting factorization problem. Now take a complementary filter pair and substitute $a(z) = H_{00}(z)$, $b(z) = H_{01}(z)$:

$$\begin{bmatrix} H_{00}(z) \\ H_{01}(z) \end{bmatrix} = \prod_{i=1}^N \begin{pmatrix} q_i(z) & 1 \\ 1 & 0 \end{pmatrix} \begin{bmatrix} K \\ 0 \end{bmatrix} \quad (2.273)$$

The constant K appears because $\det(Hp) = 1$:

$$H_{00}(z)H_{11}(z) - H_{01}(z)H_{10}(z) = 1 \quad (2.274)$$

so the gcd of $H_{00}(z)$ and $H_{01}(z)$ must be a monomial (Kz^n) , and the quotients can be chosen to have $n = 0$.

To complete a complementary filter pair, ensuring a determinant 1, we can use:

$$\begin{pmatrix} H_{00}(z) & H'_{10}(z) \\ H_{01}(z) & H'_{11}(z) \end{pmatrix} = \prod_{i=1}^N \begin{pmatrix} q_i(z) & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} K & 0 \\ 0 & 1/K \end{pmatrix} \quad (2.275)$$

Considering that:

$$\begin{pmatrix} q_i(z) & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & q_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ q_i(z) & 1 \end{pmatrix} \quad (2.276)$$

and transposing both sides:

$$\begin{pmatrix} H_{00}(z) & H_{01}(z) \\ H'_{10}(z) & H'_{11}(z) \end{pmatrix} = \begin{pmatrix} K & 0 \\ 0 & 1/K \end{pmatrix} \prod_{i=N/2}^1 \begin{pmatrix} 1 & q_{2i}(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ q_{2i-1}(z) & 1 \end{pmatrix} \quad (2.277)$$

The original filter with polyphase matrix $Hp(z)$ can be obtained with:

$$H_p(z) = \begin{pmatrix} 1 & 0 \\ -K^2 t(z) & 1 \end{pmatrix} \begin{pmatrix} K & 0 \\ 0 & 1/K \end{pmatrix} \prod_{i=N/2}^1 \begin{pmatrix} 1 & q_{2i}(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ q_{2i-1}(z) & 1 \end{pmatrix} \quad (2.278)$$

where $t(z)$ is a lifting term, which is found by comparison between both sides of the equation.

The expression above is a factorization of the filter into lifting steps.

2.8.3 Examples

2.8.3.1 A Filter Branch is Half-Band

Suppose that in the complementary pair, $H_0(z)$ is half-band. That means that $H_{00}(z) = 1$ and so the polyphase matrix is:

$$H_p(z) = \begin{pmatrix} 1 & H_{01}(z) \\ H_{10}(z) & 1 + H_{01}(z) H_{10}(z) \end{pmatrix} \quad (2.279)$$

(since $\det(Hp(z)) = 1$)

This polyphase matrix can be easily factorized:

$$H_p(z) = \begin{pmatrix} 1 & 0 \\ H_{10}(z) & 1 \end{pmatrix} \begin{pmatrix} 1 & H_{01}(z) \\ 0 & 1 \end{pmatrix} \quad (2.280)$$

This expression is connected with a family of symmetric biorthogonal wavelets with Deslauriers-Dubuc scaling functions.

2.8.3.2 The (9–7) Filter Pair

The (9–7) filter pair is frequently being used in important applications. The analysis filter has 9 coefficients, and the synthesis filter has 7 coefficients. Thanks to the work of Daubechies and Sweldens, the following factorization has been obtained:

$$H_p(z) = \begin{pmatrix} 1 & \alpha(1+z^{-1}) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \beta(1+z) & 1 \end{pmatrix} \begin{pmatrix} 1 & \gamma(1+z^{-1}) \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ \delta(1+z) & 1 \end{pmatrix} \begin{pmatrix} \varphi & 0 \\ 0 & 1/\varphi \end{pmatrix} \quad (2.281)$$

with: $\alpha = -1.58613$, $\beta = -0.05298$, $\gamma = 0.88298$, $\delta = 0.4435$, $\varphi = 1.1496$

Then, the lifting implementation of the filter is:

$$a_k^{(0)} = y_{2k} \quad (2.282)$$

$$d_k^{(0)} = y_{2k+1} \quad (2.283)$$

$$d_k^{(1)} = d_k^{(0)} + \alpha (a_k^{(0)} + a_{k+1}^{(0)}) \quad (2.284)$$

$$a_k^{(1)} = a_k^{(0)} + \beta (d_k^{(1)} + d_{k-1}^{(1)}) \quad (2.285)$$

$$d_k^{(2)} = d_k^{(1)} + \gamma (a_k^{(1)} + a_{k+1}^{(1)}) \quad (2.286)$$

$$a_k^{(2)} = a_k^{(1)} + \delta (d_k^{(2)} + d_{k-1}^{(2)}) \quad (2.287)$$

$$a_k = \varphi a_k^{(2)} \quad (2.288)$$

$$d_k = 1/\varphi d_k^{(2)} \quad (2.289)$$

Presuming that the reader would be tempted to check the CDF 9/7, the same example as before (the analysis and recovery of the sonar signal) is again selected for an exercise of CDF 9/7 coding in MATLAB.

Program 2.35 does the wavelet analysis of the sonar signal using CDF 9/7. The result is shown in Fig. 2.66.

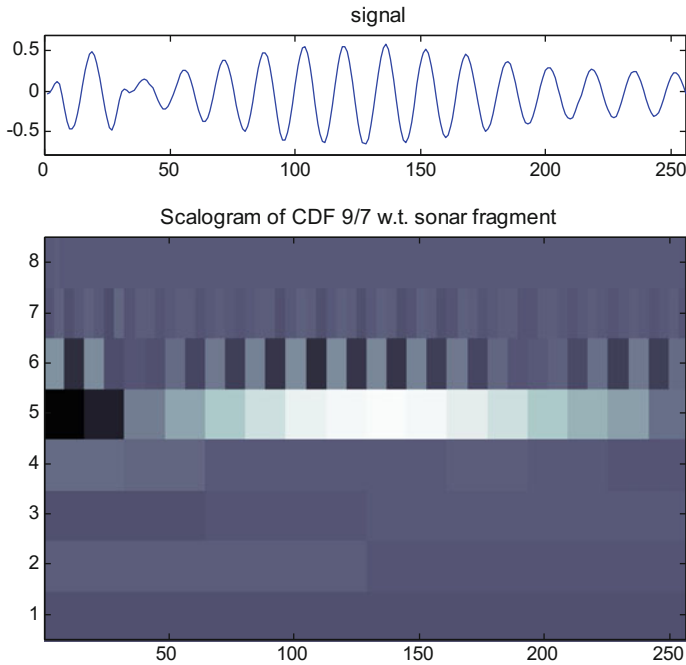


Fig. 2.66 Scalogram of sonar fragment using lifting CDF 9/7

Program 2.35 Lifting example (sonar signal), CDF 9/7

```
% Lifting example
% analysis of sonar signal
% CDF 9/7 Wavelet
% Plot of signal and scalogram
%The sonar signal
[y1,fs1]=wavread('sonar1.wav'); %read wav file
ta=7815; tb=ta+255;
sg=y1(ta:tb);
Nss=length(sg);%number of signal samples
tiv=1/fs1;
duy=(Nss-1)*tiv; %duration of signal
tss=0:tiv:duy; %time intervals set
Ts=tiv; %sampling period
%analysis of the signal with wavelets
y=sg;
a=y';
aux=0;
%CDF coeffs.
pa=-1.58613; pb=-0.05298; pg=0.88298; pd=0.4435; pp=1.1496;
K=8; %number of scales (256=2^8)
%wavelet calculus using lifting
```

```

NN=Nss;
for n=1:K,
L=length(a); L2=L/2;
a0=a(1:2:L-1);
d0=a(2:2:L);
d1=d0+(pa*(a0+[a0(2:L2) a0(1)]));
a1=a0+(pb*(d1+[d1(L2) d1(1:(L2-1))]));
d2=d1+(pg*(a1+[a1(2:L2) a1(1)]));
a2=a1+(pd*(d2+[d2(L2) d2(1:(L2-1))]));
a=pp*a2;
d=d2/pp;
aux=[d,aux];
dd(K+1-n,1:NN/2)=d;
NN=NN/2;
end;
wty=[a,aux(1:(end-1))];
%preparing for scalogram
S=zeros(K,Nss); %space for S(j,k) scalogram coefficients
for n=1:K,
q=2^(n-1); L=Nss/q;
for m=1:q,
R=(1+(L*(m-1))):(L*m); %index range
S(n,R)=dd(n,m);
end;
end;
%figure
subplot('position',[0.04 0.77 0.92 0.18])
plot(y);
axis([0 256 1.2*min(y) 1.2*max(y)]);
title('signal');
subplot('position',[0.04 0.05 0.92 0.6])
imagesc(S); colormap('bone');
title('Scalogram of CDF 9/7 w.t. sonar fragment');
h=gca; set(h,'YDir','normal');

```

The signal recovery is done with the Program 2.36, which inverts the application of the lifting equations. The recovered signal is shown in Fig. 2.67; it is close to the original signal.

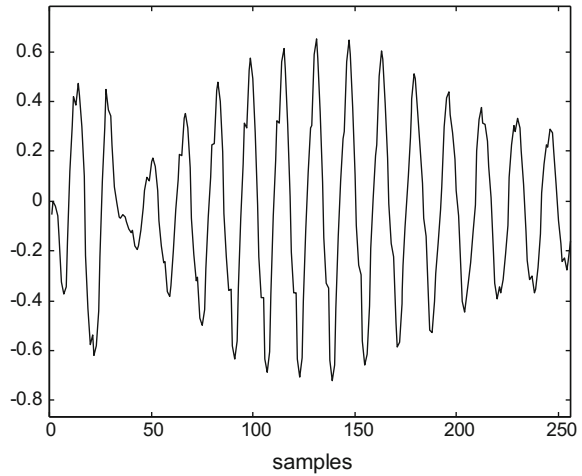
Program 2.36 Inverse of the Lifting, CDF 9/7

```

% inverse of the Lifting
% Sonar signal
% CDF 9/7 wavelet
% use after lifting analysis (it needs wty)
L=length(wty); %length of the DWT
sg=zeros(1,L);
%CDF coeffs.
pa=-1.58613; pb=-0.05298; pg=0.88298; pd=0.4435; pp=1.1496;
K=8; %number of scales

```

Fig. 2.67 Recovered sonar fragment using lifting CDF 9/7



```

cq=sqrt(3);
a=wt(1);
for n=1:K,
    wd=2^(n-1);
    bg=1+wd; fl=bg+wd-1;
    d=wt(bg:fl);
    d2=d*pp;
    a2=a/pp;
    a1=a2-(pd*(d2+[d2(end) d2(1:(end-1))]));
    d1=d2-(pg*(a1+[a1(2:end) a1(1)]));
    a0=a1-(pb*(d1+[d1(end) d1(1:(end-1))]));
    d0=d1-(pa*(a0+[a0(2:end) a0(1)]));
    sg(2:2:(2^n))=d0;
    sg(1:2:(2^n)-1)=a0;
    a=sg(1:(2^n));
end;
figure(1)
plot(sg, 'k');
axis([0 256 1.2*min(sg) 1.2*max(sg)]);
xlabel('samples');
title('the recovered signal');

```

2.9 More Analysis Flexibility

There are applications that require more processing flexibility. This in particular concerns to images, as it will be clearly seen in the next chapter. In general two approaches have been proposed. One is to extend the architecture of filter banks, and

the other is to design new wavelets with suitable capabilities. In this section the first approach is concisely introduced, based on [14] and other publications.

2.9.1 *M-Band Wavelets*

In order to get a more flexible tiling of the time-frequency plane it has been proposed [7] to use divisions into M bands, instead of the standard division into two bands. In terms of filter banks, this idea is illustrated with Fig. 2.68 for the case $M = 4$.

The frequency responses of the filters in Fig. 2.68 are depicted in Fig. 2.69.

With this approach, the multiresolution equation is the following:

$$\varphi(t) = \sum_n h_0(n) \sqrt{M} \varphi(Mt - n) \quad (2.290)$$

The frequency domain version of this MAE is:

$$\Phi(\omega) = \frac{1}{\sqrt{M}} H_0\left(\frac{\omega}{M}\right) \Phi\left(\frac{\omega}{M}\right) \quad (2.291)$$

which after iteration becomes:

$$\Phi(\omega) = \prod_{k=1}^{\infty} \left[\frac{1}{\sqrt{M}} H_0\left(\frac{\omega}{M^k}\right) \right] \Phi(0) \quad (2.292)$$

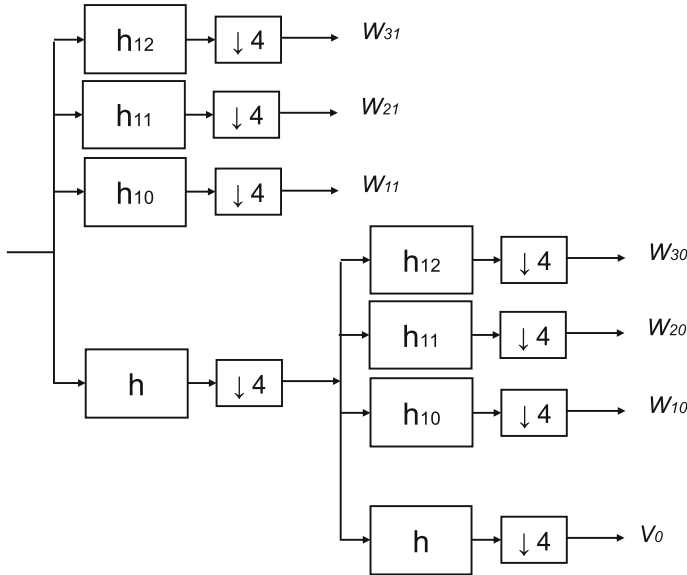


Fig. 2.68 4-band wavelet system

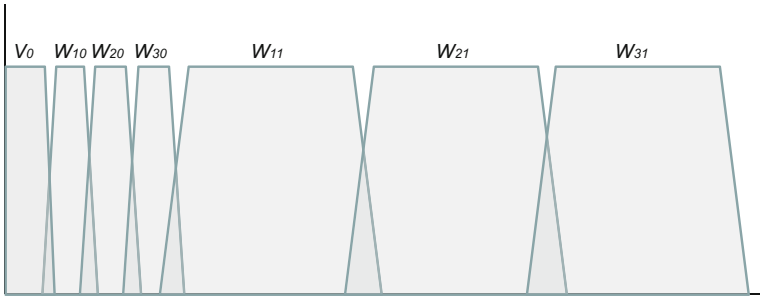


Fig. 2.69 Frequency bands of 4-band wavelet system

The computation of filter coefficients could be involved, especially for odd values of M . The article [41] shows an algebraic computation for orthonormal M-band wavelets, with several examples. In [39] the M-band wavelets are applied to audio signals; a case of adaptation to musical scales is considered. There are versions for image texture studies.

2.9.2 Wavelet Packets

In 1992, in a frequently cited article [21], Coifman proposed the ‘*wavelet packet*’ system. Figure 2.70 depicts the idea in terms of filter bank. Like for the low frequencies, there are filters dividing into bands the high frequencies. This allows for a better analysis of high frequencies.

When wavelet packets were introduced, it was noticed in real applications that the general architecture could be simplified and optimized for each particular case. The signal power is distributed among branches. It is not practical to have branches with negligible signal energy flow.

The issue can be regarded in terms of assignments of frequency bands to signal characteristics.

The research has formulated this problem as finding a *best basis*. An objective function should be defined, and then an optimization algorithm should be applied. For instance, in [21] an additive cost function was suggested; it is an entropy-based cost function that is minimized if the energy is concentrated in a few nodes. A better basis employs fewer components to represent a signal.

Supposing that a signal has been analyzed with a complete structure, then a pruning process can be applied. Start with terminal nodes, and prune bottom-up. Consider the following example of node costs (only the two last levels of a structure):

0.23	0.77
[0.18 0.19	[0.4 0.2

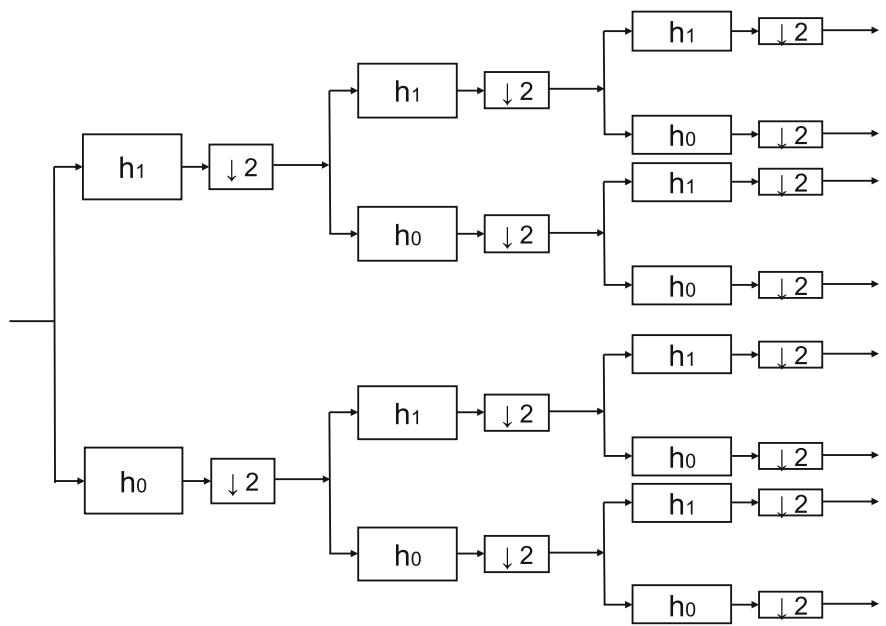


Fig. 2.70 A wavelet packet example

If the sum of costs of children (S) is greater than the parent’s cost, the children are pruned. Otherwise, the parent’s cost is replaced with S . Therefore, after the first step of pruning the two last levels become:

0.23	0.6
0.18 0.19	0.4 0.2

Bold numbers correspond to selected members of the best basis. The algorithm continues towards the top of the hierarchy.

Several cost functions have been proposed based on thresholds, the number of bits to represent the signal, and other measures.

2.9.3 Multiwavelets

Whereas wavelets have an associated scaling function and wavelet function, *multi-wavelets* have two or more scaling and wavelet functions. In this way more analysis freedom—with good properties—is allowed.

The set of scaling functions can be written in vector notation as follows:

$$\Phi(t) = [\varphi_1(t), \varphi_2(t), \dots, \varphi_R(t)]^T \quad (2.293)$$

and the set of wavelet functions

$$\Psi(t) = [\psi_1(t), \psi_2(t), \dots, \psi_R(t)]^T \quad (2.294)$$

The multiresolution equation is the following:

$$\Phi(t) = \sqrt{2} \sum_n H(n) \Phi(2t - n) \quad (2.295)$$

where $H(n)$ is a $R \times R$ matrix.

Likewise, the construction of wavelets is:

$$\Psi(t) = \sqrt{2} \sum_n G(n) \Phi(2t - n) \quad (2.296)$$

where $G(n)$ is another $R \times R$ matrix.

2.9.3.1 A Simple Example

Most published multiwavelet systems include just two scaling functions. For example [8] one Haar scaling function as $\varphi_1(t)$ and a second scaling function such as:

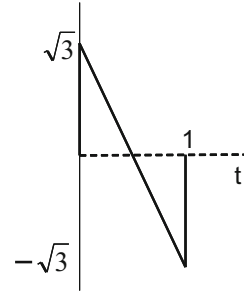
$$\varphi_2(t) = \frac{\sqrt{3}}{2} \varphi_1(2t) + \frac{1}{2} \varphi_2(2t) - \frac{\sqrt{3}}{2} \varphi_1(2t - 1) + \frac{1}{2} \varphi_2(2t - 1) \quad (2.297)$$

Fig. 2.71 shows $\varphi_2(t)$:

The MAE for this case can be written as follows:

$$\begin{bmatrix} \varphi_1(t) \\ \varphi_2(t) \end{bmatrix} = \begin{pmatrix} 10 \\ \frac{\sqrt{3}}{2} \frac{1}{2} \end{pmatrix} \begin{bmatrix} \varphi_1(2t) \\ \varphi_2(2t) \end{bmatrix} + \begin{pmatrix} 10 \\ -\frac{\sqrt{3}}{2} \frac{1}{2} \end{pmatrix} \begin{bmatrix} \varphi_1(2t - 1) \\ \varphi_2(2t - 1) \end{bmatrix} \quad (2.298)$$

Fig. 2.71 The second scaling function $\varphi_2(t)$



2.9.3.2 The Geronimo, Hardin, and Massopust Multiwavelet System

The dilation and wavelet equations of the multiwavelet system proposed by Geronimo, Hardin and Massopust (see [65] and references therein), have the following expressions:

$$\begin{aligned} \Phi(t) = \begin{bmatrix} \varphi_1(t) \\ \varphi_2(t) \end{bmatrix} &= H(0) \Phi(2t) + H(1) \Phi(2t - 1) \\ &+ H(2) \Phi(2t - 2) + H(3) \Phi(2t - 3) \end{aligned} \quad (2.299)$$

with:

$$H(0) = \begin{pmatrix} -\frac{3}{10\sqrt{2}} & \frac{4\sqrt{2}}{5} \\ \frac{3}{10} & -\frac{3}{10} \end{pmatrix}; \quad H(1) = \begin{pmatrix} \frac{3}{10\sqrt{2}} & 0 \\ \frac{9}{10\sqrt{2}} & 1 \end{pmatrix} \quad (2.300)$$

$$H(2) = \begin{pmatrix} 0 & 0 \\ \frac{9}{10\sqrt{2}} & -\frac{3}{10} \end{pmatrix}; \quad H(3) = \begin{pmatrix} 0 & 0 \\ -\frac{1}{10\sqrt{2}} & 0 \end{pmatrix} \quad (2.301)$$

and,

$$\begin{aligned} \Psi(t) = \begin{bmatrix} \psi_1(t) \\ \psi_2(t) \end{bmatrix} &= G(0) \Phi(2t) + G(1) \Phi(2t - 1) \\ &+ G(2) \Phi(2t - 2) + G(3) \Phi(2t - 3) \end{aligned} \quad (2.302)$$

with:

$$G(0) = \frac{1}{10} \begin{pmatrix} -\frac{1}{\sqrt{2}} & -3 \\ 1 & -3\sqrt{2} \end{pmatrix}; \quad G(1) = \frac{1}{10} \begin{pmatrix} \frac{9}{\sqrt{2}} & -10 \\ -9 & 0 \end{pmatrix} \quad (2.303)$$

$$G(2) = \frac{1}{10} \begin{pmatrix} \frac{9}{\sqrt{2}} & -3 \\ 9 & -3\sqrt{2} \end{pmatrix}; \quad G(3) = \frac{1}{10} \begin{pmatrix} -\frac{1}{\sqrt{2}} & 0 \\ -1 & 0 \end{pmatrix} \quad (2.304)$$

This multiwavelet system has notable properties: the scaling functions have short support, both scaling functions are symmetric, the wavelets form a symmetric/antisymmetric pair, and all integer translates of the scaling functions are orthogonal.

2.9.3.3 Other Examples

Let us include two more examples. The first is the symmetric pair determined by the following three coefficients [65]:

$$H(0) = \begin{pmatrix} 0 & \frac{2+\sqrt{7}}{4} \\ 0 & \frac{2-\sqrt{7}}{4} \end{pmatrix}; \quad H(1) = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{4} \end{pmatrix} \quad (2.305)$$

$$H(2) = \begin{pmatrix} \frac{2-\sqrt{7}}{4} & 0 \\ \frac{2+\sqrt{7}}{4} & 0 \end{pmatrix} \quad (2.306)$$

The other example is the multiwavelet system proposed by Chui-Lian (see [65] and references therein):

$$H(0) = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{\sqrt{7}}{4} & -\frac{\sqrt{7}}{4} \end{pmatrix}; \quad H(1) = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \quad (2.307)$$

$$H(2) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{\sqrt{7}}{4} & -\frac{\sqrt{7}}{4} \end{pmatrix} \quad (2.308)$$

2.10 Experiments

Next experiments consider the most popular applications of wavelets, which are signal analysis, denoising, and compression.

2.10.1 ECG Analysis Using the Morlet Wavelet

One of the most important fields of wavelet application is medicine. In this experiment a fragment of a normal electrocardiogram (ECG) signal has been selected. The fragment is shown in the upper plot in Fig. 2.72. It includes two consecutive heartbeats.

The scalograms corresponding to continuous wavelet analysis are very expressive regarding local frequency contents. In this experiment the Morlet wavelet has been

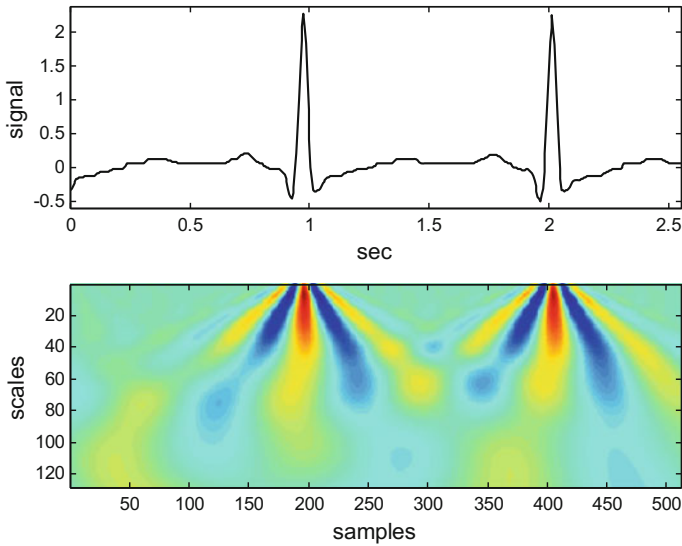


Fig. 2.72 Signal analysis using the Morlet wavelet

used. The scalogram of the two heartbeats is shown in Fig. 2.72. The pictured signal energies at each time and scale correspond well to the peaks and the valleys of heartbeats. The figure has been obtained with the Program 2.37.

Notice important aspects of the Program 2.37. Zero padding is applied for both extremes of the signal. The program has a first vectorized part to compute a large wavelet tensor. Then, the program continues with the CWT as a multiplication of signal and tensor. The execution of the program may take around one or two minutes. Notice that the result is a large matrix, corresponding to the scalogram pixels.

Program 2.37 ECG analysis by continuous Morlet wavelet transform

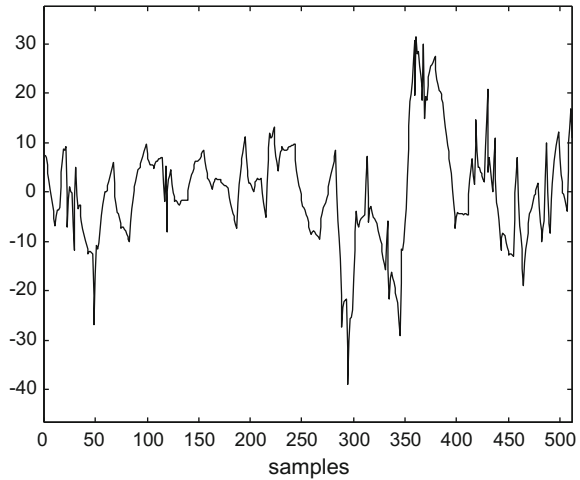
```
% ECG analysis by continuous wavelet transform
% Morlet Wavelet
% Plot of signal and scalogram
clear all;
disp('please wait'); %ask for patience
%The ECG signal
fs=200; %sampling frequency in Hz
tiv=1/fs; %time interval between samples
Ts=tiv; %sampling period
%read signal file
fer=0;
while fer==0,
fid2=fopen('ECG_short.txt','r');
if fid2==~-1, disp('read error')
else sg=fscanf(fid2,'%f \r\n'); fer=1;
end;
```

```

end;
fclose('all');
Nss=length(sg); %number of signal samples
duy=(Nss-1)*tiv; %duration of signal
tss=0:tiv:duy; %time intervals set
y=[sg(1)*ones(1,128) sg' sg(end)*ones(1,128)]; %padding
y=y-mean(y); %zero-mean signal
ND=length(y); %number of data
NS=128; %number of scales
CC=zeros(NS,ND); %for output (coeffs)
% Pre-compute wavelet tensor-----
PSIa=zeros(NS,ND,ND); %array
nn=1:ND;
t=Ts*(nn-1);
for ee=1:NS,
s=(ee*0.006)+0.05; %scales
for rr=1:ND, %delays
a=Ts*(rr-1);
val=0;
%vectorized part (t)
x=(t-a)/s;
PSIa(ee,rr,:)=(1/sqrt(s))*(exp(-(x.^2)/2)).*cos(5*x));
end;
end;
disp('wavelet tensor is now ready')
%CWT-----
nd=1:ND;
for ne=1:NS,
aux=squeeze(PSIa(ne,nd,:));
val=(y)*aux;
CC(ne,nd)=val;
end;
%display-----
figure(1)
subplot(2,1,1)
plot(tss,y(129:(128+Nss)),'k');
axis([0 (Nss-1)*tiv min(y)-0.1 max(y)+0.1]);
xlabel('sec'); ylabel('signal');
title('wavelet analysis');
subplot(2,1,2)
imagesc(CC(:,129:(128+Nss)));
colormap('jet');
xlabel('samples'); ylabel('scales');

```

Fig. 2.73 Denoised Evoked Potential signal



2.10.2 Signal Denoising

Once a signal is decomposed into wavelet coefficients at several scales, there is the opportunity of attenuating or deleting high-frequency components often related to noise.

The idea is to choose a value, a threshold, and modify wavelet coefficients being smaller than the threshold. Two alternatives have been proposed: hard and soft thresholding. Both substitute by zeros the smaller coefficients. Hard thresholding keeps untouched the rest of coefficients. Soft thresholding subtracts the threshold to the rest of coefficients.

The thresholding is usually applied to the highest scales. However there are alternatives: global, or scale level by scale level.

Of course there is a key issue: what threshold value should be chosen. The research has proposed several ways to take into account estimated noise.

Another issue is strategic: to keep a constant threshold, or to change it along the signal. Again, the research proposes some alternatives, being adaptive or not, local in time or by blocks, etc.

Opportune references on signal denoising are [29, 56].

Our experiment is to denoise the evoked potential signal that has been studied in Sect. 2.4.3. A hard thresholding has been applied using the Program 2.38. The result is shown in Fig. 2.73.

Program 2.38 Denoising example with Daubechies DWT

```
% Denoising example
% with inverse of the Daubechies DWT
% Visual Evoked Potential signal
% Use after analysis routine (to get wty)
L=length(wty); %length of the DWT
```



```

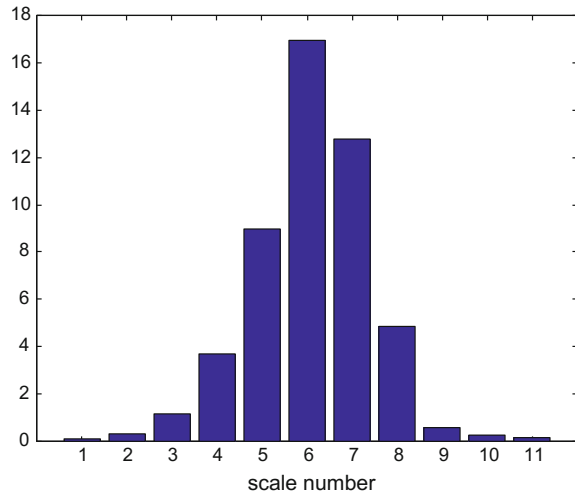
%scaling filter
hden=4*sqrt(2); %coeff. denominator
hsq=sqrt(3); %factor
%Daubechies 4:
h=[(1+hsq)/hden, (3+hsq)/hden, (3-hsq)/hden, (1-hsq)/hden];
hN=h;
N=length(hN);
K=9; %number of scales
aux=0;
h0=hN;
h1=fliplr(hN); h1(2:2:N)=-h1(2:2:N);
Ln=1;
a=wt(1);
th=8; %noise threshold
for n=1:K,
    aux= 1+mod(0:N/2-1, Ln);
    d=wt(Ln+1:2*Ln);
    %denoising at the higher scales
    if n>K-6,
        mm=length(d);
        for m=1:mm,
            if abs(d(m))<th,
                d(m)=0;
            end;
        end;
    end;
    ax(1:2:2*Ln+N)= [a a(1,aux)];
    dx(1:2:2*Ln+N)= [d d(1,aux)];
    a=conv(ax,h0)+ conv(dx,h1);
    a=a(N:(N+2*Ln-1));
    Ln=2*Ln;
end;
figure(1)
plot(a, 'k');
axis([0 512 1.2*min(a) 1.2*max(a)]);
xlabel('samples');
title('the denoised signal');

```

2.10.3 Compression

It has been noticed in many cases that if you delete the wavelet coefficients of the highest scales, and then you recover the signal, this recovered signal has acceptable characteristics. Even, it may happen that this signal has less noise than the original. Therefore, if you have say a 10 Mb signal and you apply wavelet analysis using 12 scale levels, it can be reduced to 5 Mb just by deleting the 12th scale coefficients;

Fig. 2.74 Energy of the signal to be compressed



or even smaller sizes if you continue deleting coefficients of 11th, 10th, etc., scale levels. Obviously you lose signal details: it is lossy compression.

A piece of Bizet music has been chosen for our signal compression experiment. Since the signal is long in terms of number of samples, the signal is divided into segments, and a Haar wavelet analysis has been repeatedly applied.

The idea has been to obtain with the wavelet analysis a set of vectors $wty(n,:)$ containing the wavelet coefficients of each signal segment. This is an uncompressed representation of the original signal. Then, the upper half part of each $wty(n,:)$ (with $n = 1, 2, \dots$) is eliminated. Hence, a 50% signal compression is achieved.

Program 2.39 implements this simple procedure. The result of compression is a set of vectors $cwty(n,:)$. This is supposed to be stored on a file for further use.

Then, after compression a signal is recovered by the second part of the program. Here the simple idea is to append zeros in substitution of the deleted wavelet coefficients, getting a set of vectors $rwty(n,:)$, and then apply Haar synthesis to obtain an uncompressed signal.

When running the Program 2.39 the user can hear the original music, and then, after some moments, hear the recovered uncompressed signal for comparison. The program also generates the Fig. 2.74 that shows the energy distribution of the music signal among scale levels. The reason for the compressing procedure to be acceptable is that it only touches the highest scale, where there is not much signal energy. The figure also suggests that more compressing could be tried.

Program 2.39 Audio compression example with Haar wavelet

```
% Audio compression example
% analysis of music signal
% Haar Wavelet
% Hear original and compressed signals
```

```

%The music signal
[y,fs]=wavread('Bizet1.wav'); %read wav file
y1=y(:,1); %mono channel
y1=y1-mean(y1); %zero mean
Nss=300*2048; %number of signal samples
sg=y1(1:Nss);
tiv=1/fs;
disp('the original music');
soundsc(sg,fs);
pause(16);
disp('now, wavelets in action');
%analysis of the signal with wavelets
%divide the signal into 230 segments
%apply wavelet analysis to each segment
K=11; %number of scales (2048=2^11)
cq=sqrt(3);
wty=zeros(300,2048);
En=zeros(1,K); %for energy measurement
for nst=1:300,
    bg=( (nst-1)*2048)+1;
    y=sg(bg:(bg+2047)); %music segment
    %Haar wavelet
    NN=2048;
    for n=1:K,
        aux1= y(1:2:NN-1) + y(2:2:NN);
        aux2= y(1:2:NN-1) - y(2:2:NN);
        y(1:NN)=[aux1,aux2]/sqrt(2);
        En(n)=En(n)+sum(y((NN/2)+1:NN).^2);
        NN=NN/2;
    end;
    wty(nst,:)=y';
end;
%-----
%compress the signal by deleting highest scale data
aux=zeros(300,1024);
cwtly=zeros(300,1024); %50% smaller
for nn=1:300,
    aux(nn,:)=wty(nn,1:(2^10)); %delete the upper half of wty
end;
cwtly=aux; %compressed audio (to be stored)
%-----
% read compressed audio and recover the signal
aux=zeros(300,1024);
rwty=zeros(300,2048);
%append zeros for the upper half of rwty
for nn=1:300,
    rwty(nn,:)= [cwtly(nn,:) aux(nn,:)];
end;
%-----

```

```

%conventional signal recovery
ry=zeros(1,Nss);
J=K+1;
a=zeros(J,(2^K)); %space for a(j,k) coefficients
%signal recovering (wavelet synthesis)
for nst=1:300,
m=1;
z=rwty(nst,:);
a(1,1)=z(1);
for n=1:K,
a(n+1,1:2:(2*m-1))=(a(n,1:m)+z((1+m):(2*m)))/sqrt(2);
a(n+1,2:2:(2*m))=(a(n,1:m)-z((1+m):(2*m)))/sqrt(2);
m=m*2;
end;
bg=((nst-1)*2048)+1;
ry(bg:(bg+2047))=a(J,:); %signal recovery
end;
disp('and the decompressed music');
soundsc(ry,fs);
%display of signal energy vs scale-----
figure(1)
bar(En);
title('signal energy vs. scale');
xlabel('scale number');

```

2.11 Applications

The area of wavelet applications is continuously expanding. Possibly, two main application directions could be identified. One is linked to signal/data compression, and denoising. The other comes from the origins of wavelets: the analysis of signals/data.

In this section a short overview of different fields of application will be done. Most cited references are intended as starting points for the reader, and so they include abundant bibliographies.

A review of emerging applications of wavelets is [4]. The book [37] includes a set of case studies. There are several scientific journals focusing on wavelets, like the International Journal of Wavelets, Multiresolution and Information Processing, the Journal of Wavelet Theory and Applications, and others.

2.11.1 Earth Sciences

During the last 1970s, Morlet introduced what he called “wavelets of constant shapes”. Mr. Morlet was a geophysical engineer at an oil company. Daubechies,

in her view of the wavelet history [26], tells that, soon after, the research adopted the term *wavelet* letting implicit the mention of constant shape.

Hence, the origin of wavelets is in geophysical signal analysis. It is not strange that now there is an extensive literature on geophysical wavelet applications.

Seismic signals are of evident interest for at least two reasons. One is earthquakes, and the other is detection and characterization of hydrocarbon reservoirs, minerals, etc.

With respect to earthquakes, the article [61] provides pertinent information and references.

And with respect to natural resources the key topic is seismic inversion. The web page of the Wavelet Seismic Inversion Lab, and the publications of the Society of Exploration Geophysicists offer a good ingress on this field. In [62] the use of wavelets and the like on the sphere, with application to geophysical data, is treated; see also [63] about the use of continuous wavelets.

There are many other branches of Earth sciences interested on the use of wavelets. This is revealed by research papers like [19] on tsunami warning, [32] on ocean waves, [28] on atmospheric sciences and space, etc. The article [34] presents an overview of wavelet applications in earthquake, wind and ocean engineering.

There are some books on wavelets and geophysics, like [33, 38]. Denoising of geophysical data is considered in a number of papers, see [71] and references therein.

2.11.2 *Medicine, Biology*

The denoising and compressing capabilities of wavelets have found great acceptance for biomedical applications. In addition, the research has proposed several applications of wavelets for the signal/data processing and analysis in this area. There are some books about the use of wavelets in medicine and biology [6, 18]. Other books include chapters on wavelet biomedical applications [51, 52, 58]. The book [11] deals with wavelet applications in biology and geosciences. Articles with reviews of this topic are [1, 5, 74].

A lot of research effort has been directed to image processing. This aspect will be covered in the next chapter.

Regarding physiological signals, a main source of information is the web page Physionet

The application of wavelet techniques in electrocardiograms (ECG) is reviewed in [50]. The paper [30] deals with sleep stage classification on the basis of electroencephalogram. As an aspect of robotics, the recognition of emotions is treated in the Thesis [10].

The paper [42] presents the state of the art in bioinformatics.

Since natural sounds, like bird singing or other animal sounds, are transients, it is pertinent to use wavelets for their study; [40] is a frequently cited article about it.

2.11.3 Chemical

According with the perspective given in [59] there is an increasing number of wavelet applications in chemistry. In particular for the study of spectra, chromatography, or in case of oscillatory signals.

Some examples are [31], which describes the applications of wavelets to analytical chemistry for denoising, compression, etc. The article [60] on the study of pressure in a fluidized bed. And [17] on Raman spectrography, and [57] on spectral libraries and remote environmental sensing.

2.11.4 Industrial

In the reviews [72, 73] of industrial applications of wavelets, the authors cite applications related to acoustical signal processing, power production and transport, power electronics, non-destructive testing, chemical processes, stochastic signal analysis, image compression, satellite imagery, machine vision, bioinformatics, and flow analysis.

Vibration and mechanical damage is an obvious field of application for wavelets. Two illustrative papers concerned with this topic are [66] on structural damage, and [9] on bridge integrity assessment.

Another kind of application is machine condition monitoring. The article [54] presents a review of this topic, with extensive bibliography.

In [2] the authors deal with crack propagation in rotating shafts. The article of [55] uses wavelet analysis of acoustic emissions for material characterization. There is a number of papers on wavelets and materials fatigue, considering cases related to railways, ships, airplanes, etc.

2.12 The MATLAB Wavelet Toolbox

With the help of the MATLAB Wavelet Toolbox, wavelets can be applied and studied in a fast and easy way. This Toolbox has extensive and well organized documentation.

The purpose of this section is a quick introduction to the Toolbox.

Since the Toolbox includes many wavelets, the user can recall names and short descriptions using the functions *waveinfo()* and *waveletfamilies()*.

With *wavedemo()*, the user can get first impressions of wavelet possibilities.

An important general tool is *wavemenu()*. It is an entrance to the graphical tools. The following screen appears (Fig. 2.75):

The Toolbox provides functions for 1-D continuous and discrete wavelets, and 2-D discrete wavelets. In addition the Toolbox includes wavelet packets, lifting wavelets, multiwavelets, denoising and compression and other wavelet applications.

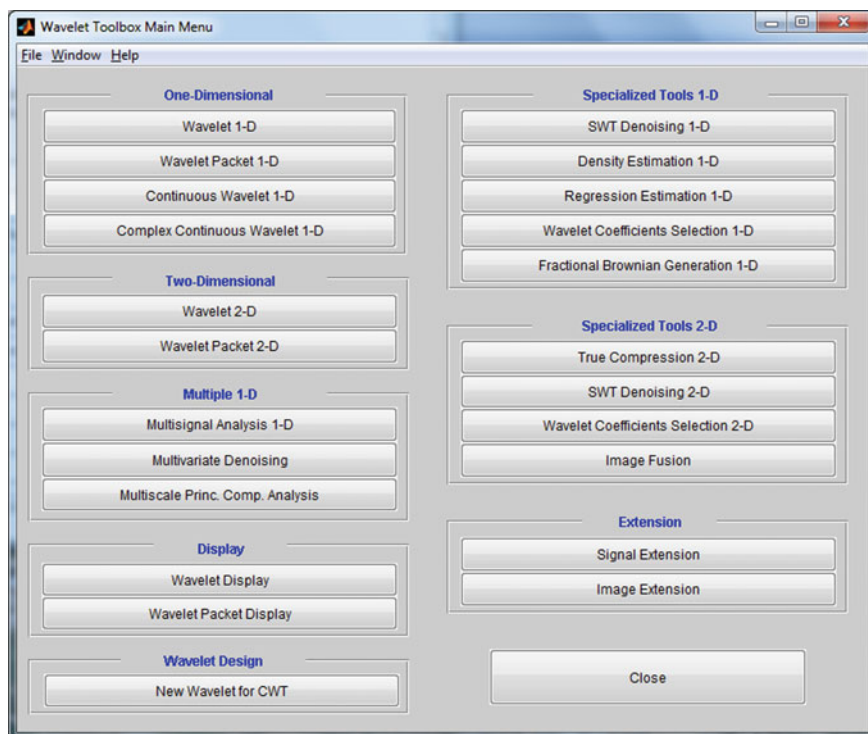


Fig. 2.75 Screen corresponding to `wavemenu()`

2.12.1 1-D Continuous Wavelet

The main function for 1-D continuous wavelets is `cwt()`. The scalogram can be obtained in two ways: one by using `wscalogram()` after `cwt()`, or directly from `cwt()` with 'plot' option.

If you use the [Continuous Wavelet 1-D] option after `wavemenu()`, a graphical tool opens. You import a file with the signal to analyze (for instance in `.wav` format), then choose a wavelet, and then select display options (colors, subplots, etc.). The following screen (Fig. 2.76) has been obtained using the Morlet wavelet for the analysis of the cow moaning already studied in Chap. 6 with the spectrogram. The central plot is the scalogram.

2.12.2 1-D Discrete Wavelet

The main functions for 1-D discrete wavelets are `dwt()` and `idwt()` for single-level transform, and `wavedec()` and `waverec()` for multiple level transform.

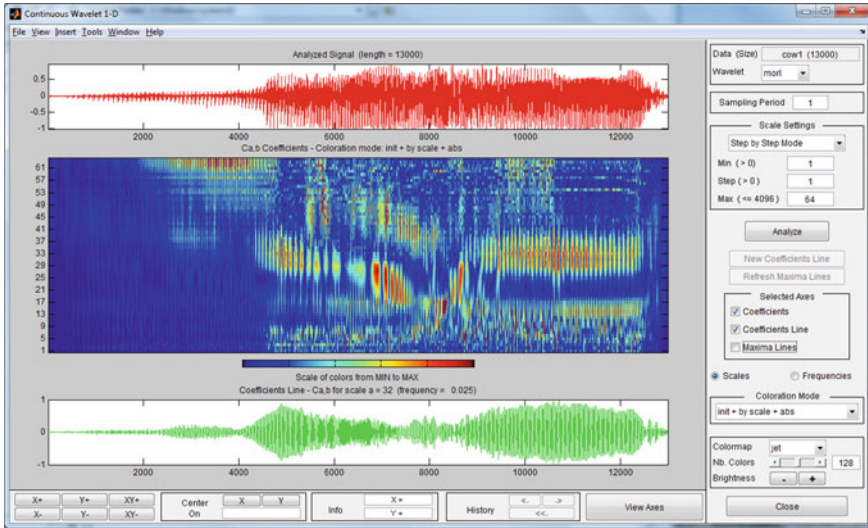


Fig. 2.76 Screen corresponding to cow moaning

If you use the [Wavelet 1-D] option after *wavemenu()*, then a screen similar to the previous example opens. Figure 2.77 shows the result of analyzing a duck quack. The subplots show the decomposition into scales using the Haar wavelet (the numbering of scales is the opposite to the numbering in this chapter).

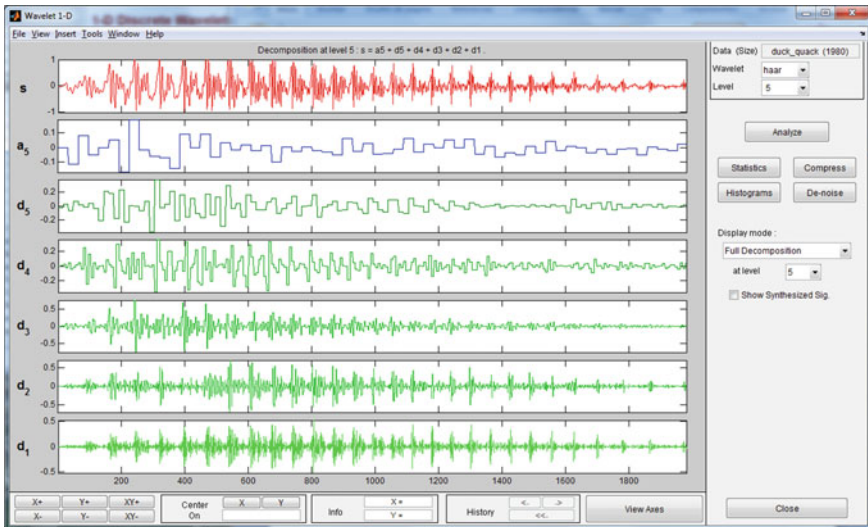


Fig. 2.77 Screen corresponding to duck quack

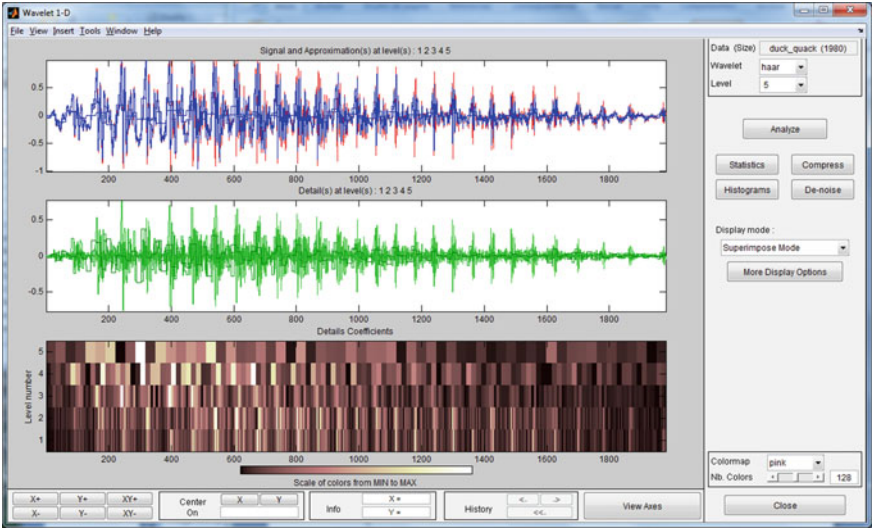


Fig. 2.78 Another screen corresponding to duck quack

Another visualization option leads to the screen shown in Fig. 2.78. It includes an approximation to the signal and the scalogram.

2.12.3 Wavelet Packets

The main functions for 1-D wavelet packets are *wpdec()* and *wprec()*. To find best basis one uses *bestlevt()* and *besttree()*.

If you use the [Wavelet Packet 1-D] option after *wavemenu()*, then a screen opens. This is a different type of screen, specific for wavelet packets. Figure 2.79 shows the screen when a harp signal was analyzed with Haar wavelet. The left plot shows the structure of the packet decomposition. If you click on one of the nodes, a plot appears at bottom left with the part of the analyzed signal that crosses this node. Another plot, bottom right, shows the coefficients at the terminal nodes. Taken as a whole, the screen is a tool that helps to find a best basis.

2.12.4 Lifting

The main functions for 1-D lifting wavelet transforms are *lwt()* and *ilwt()*. Information about lifting schemes is given by *lsinfo()* and *displs()*, the lifting scheme for

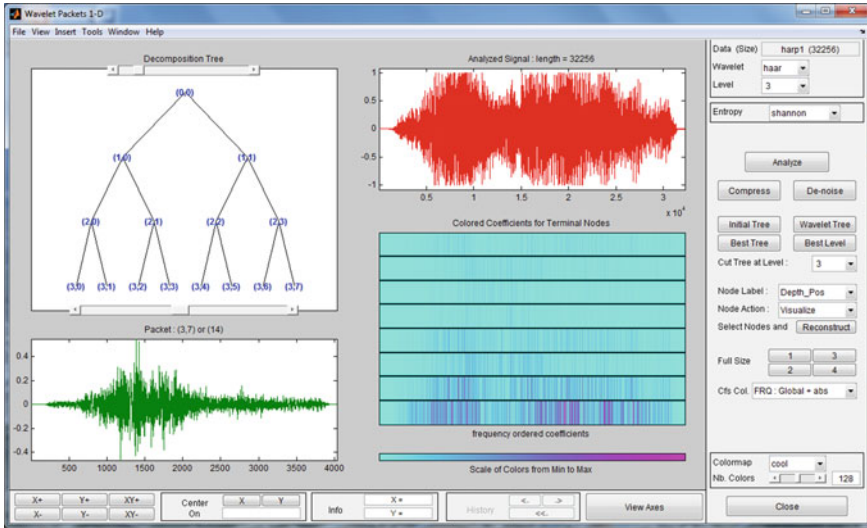


Fig. 2.79 Screen corresponding to harp and wavelet packet analysis

a particular wavelet is given by *liftwave()*, the names of wavelets is provided by *wavenames()*. The lifting schemes can be translated to filter banks using *ls2filt()*.

2.12.4.1 Multiwavelets (multisignal)

The main functions for 1-D multisignal wavelet transforms are *mdwtdec()* and *mdwtrec()*. There are functions for denoising and compression using multisignal transform.

2.13 Resources

2.13.1 MATLAB

Usually the toolboxes include tables of coefficients for the different wavelet families.

2.13.1.1 Toolboxes

- WaveLab:
<http://www-stat.stanford.edu/~wavelab/>
- Uvi-Wave Wavelet Toolbox:
<http://www.gts.tsc.uvigo.es/~wavelets/>

- Rice Wavelet Toolbox:
<http://dsp.rice.edu/software/rice-wavelet-toolbox>
- UST Wavelets:
<http://cam.mathlab.stthomas.edu/wavelets/packages.php>
- WavBox:
<http://www.toolsmiths.com/wavelet/WavBox>
- WAVOS Toolkit:
<http://sourceforge.net/projects/wavos/files/>
- WMTSA Wavelet Toolkit:
<http://www.atmos.washington.edu/~wmtsa/>
- Wavekit Toolkit:
<http://www.math.rutgers.edu/ojanen/wavekit/>

2.13.1.2 Matlab Code

- Mathworks file exchange:
<http://www.mathworks.com/matlabcentral/fileexchange/5104-toolbox-wavelets>
- Gabriel Peyre:
<http://www.mathworks.es/matlabcentral/fileexchange/authors/14044>
- Ripples in Mathematics:
<http://www.control.auc.dk/alc/ripples.html>
- Matthew Roughan:
<http://www.maths.adelaide.edu.au/matthew.roughan/>
- Pascal Getreuer:
<http://www.getreuer.info/>
(see also): <http://www.mathworks.com/matlabcentral/fileexchange/authors/14582>
- Brooklin Poly:
<http://eeweb.poly.edu/iselesni/WaveletSoftware/standard1D.html>
- Joseph Salmon:
<http://josephsalmon.eu/enseignement/M1/ondelettes.sci>

2.13.2 Internet

2.13.2.1 Web Sites

- Wavelet.org:
<http://www.wavelet.org/>
- Wim Sweldens' Homepage:
www.cm.bell-labs.com/who/wim/
- Jacket's Wavelets:
<http://gtwavelet.bme.gatech.edu/>

- Rodrigo Quian, Evoked Potential data:
<http://www2.le.ac.uk/departments/engineering/research/bioengineering/neuroengineering-lab/software/>
- The Wavelet Seismic Inversion Lab:
<http://timna.mines.edu/~zmeng/waveletlab/waveletlab.html>
- Physionet:
www.physionet.org
- The BioSig Project:
<http://biosig.sourceforge.net/>
- JPEG Homepage:
<http://www.jpeg.org/jpeg/index.html>

2.13.2.2 Link Lists

- Wavelet Software:
<http://www.amara.com/current/wavesoft.html>
- SIVA links:
<http://www.laurent-duval.eu/siva-signal-image-links.html>
- Baum links:
<http://stommel.tamu.edu/~baum/wavelets.html>

References

1. P.S. Addison, J. Walker, R.C. Guido, Time-frequency analysis of biosignals. *IEEE Eng. Med. Biol. Mgz.* pp. 14–29 (2009)
2. S.A. Adewuai, B.O. Al-Bedoor, Wavelet analysis of vibration signals of an overhang rotor with a propagating transverse crack. *J. Sound Vibr.* **246**(5), 777–793 (2001)
3. L. Aguiar-Conraria, M.J. Soares, The continuous wavelet transform: Moving beyond uni- and bivariate analysis. *J. Econ. Surv.* **28**(2), 344–375 (2014)
4. A.N. Akansu, W.A. Serdijn, I.W. Selesnick, Emerging applications of wavelets: A review. *Phys. Commun.* **3**, 1–18 (2010)
5. M. Akay, Wavelet applications in medicine. *IEEE Spectrum* **34**(5), 50–56 (1997)
6. A. Aldroubi, M. Unser, *Wavelets in Medicine and Biology* (CRC Press, 1996)
7. O. Alkin, H. Caglar, Design of efficient M-band coders with linear-phase and perfect-reconstruction properties. *IEEE Trans. Sign. Process.* **43**(7), 1579–1590 (1995)
8. B.K. Alpert, A class of bases in L_2 for the sparse representation of integral operators. *SIAM J. Math. Anal.* **24**(1), 246–262 (1993)
9. A. Alvandi, J. Bastien, E. Gregoire, M. Jolin, Bridge integrity assessment by continuous wavelet transforms. *Intl. J. Struct. Stab. Dyn.* **9**(11) (2009)
10. V. Aniket, *Biosignal Processing Challenges in Emotion Recognition for Adaptive Learning*. PhD thesis (Univ. Central Florida, 2010)
11. D. Balenau, *Wavelet Transforms and Their Recent Applications in Biology and Geoscience* (InTech., 2012)
12. R.P. Boyer, Generalized Bernstein polynomials and symmetric functions. *Adv. Appl. Math.* **28**, 17–39 (2002)

13. J. Bradley, C. Brislawn, T. Hopper, The FBI wavelet/scalar quantization standard for gray-scale fingerprint image compression, in *SPIE v.1961: Visual Image Processing* (1993), pp. 293–304
14. C.S. Burrus, R.A. Gopinath, H. Guo, *Wavelets and Wavelet Transforms* (Prentice-Hall, 1998)
15. P. Burt, E. Adelson, The Laplacian pyramid as a compact image code. *IEEE Trans. Commun.* **31**, 482–540 (1983)
16. H. Caglar, A.N. Akansu, A generalized parametric PR-QMF design technique based on Bernstein polynomial approximation. *IEEE Trans. Sign. Process.* **41**(7), 2314–2321 (1993)
17. T.T. Cai, D. Zhang, D. Ben-Amotz, Enhanced chemical classification of Raman images using multiresolution wavelet transformation. *Appl. Spectrosc.* **55**(9), 1124–1130 (2001)
18. R. Capobianco, *Emergent Applications of Fractals and Wavelets in Biology and Biomedicine* (Elsevier, 2009)
19. A. Chamoli, V.S. Rani, K. Srivastava, D. Srinagesh, V.P. Dimri, Wavelet analysis of the seismograms for tsunami warning. *Nonlinear Process. Geophys.* **17**, 569–574 (2010)
20. A. Cohen, I. Daubechies, J.C. Feauveau, Biorthogonal bases of compactly supported wavelets. *Commun. Pure Appl. Math.* **45**, 485–560 (1992)
21. R.R. Coifman, M.V. Wickerhauser, Entropy-based algorithms for best basis selection. *IEEE Trans. Inform. Theory* **38**(2), 713–718 (1992)
22. T. Cooklev, A. Nishihara, M. Sablatash, Regular orthonormal and biorthogonal wavelet filters. *Sign. Process.* **57**, 121–137 (1997)
23. A. Cour-Harbo, A. Jensen. Wavelets and the lifting scheme, in *Encyclopedia of Complexity and Systems Science*, ed. by R.A. Meyers (Springer, 2009), pp. 10007–10031
24. I. Daubechies, The wavelet transform, time-frequency localization and signal analysis. *IEEE Trans. Inform. Theory* **36**(5), 961–1005 (1990)
25. I. Daubechies, *Ten Lectures on Wavelets* (SIAM, Philadelphia, 1992)
26. I. Daubechies, Where do wavelets come from?—A personal point of view. *Proc. IEEE* **84**(4), 510–513 (1996)
27. I. Daubechies, W. Sweldens, Factoring wavelet and subband transforms into lifting steps. Technical report, TechnicalBell Laboratories, Lucent Technologies (1996)
28. M.O. Domingues, O. Jr. Mendes, A. Mendes da Costa, On wavelet techniques in atmospheric sciences. *Adv. Space Res.* **35**, 831–842 (2005)
29. D.L. Donoho, Denoising by soft-thresholding. *IEEE Trans. Inform. Theory* **41**(3), 613–627 (1995)
30. F. Ebrahimi, M. Mikaeili, E. Estrada, H. Nazeran, Automatic sleep stage classification based on EEG signals by using neural networks and wavelet packet coefficients. *Proc. IEEE Int. Conf. EMBS* 1151–1154 (2008)
31. F. Ehrentreich, Wavelet transform applications in analytical chemistry. *Anal. Bioanal. Chem.* **372**(1), 115–121 (2002)
32. M. Elsayed, An overview of wavelet analysis and its application to ocean wind waves. *J. Coast. Res.* **26**(3), 535–540 (2010)
33. Foufola-Georgiou, E., P. Kumar, *Wavelets in Geophysics* (Academic Press, 1994)
34. K. Gurley, A. Kareem, Applications of wavelet transforms in earthquakes, wind and ocean engineering. *Eng. Struct.* **21**, 149–167 (1999)
35. A. Jensen, A. la Cour-Harbo, *Ripples in Mathematics* (Springer, 2001)
36. Z. Jiang, X. Guo, A note on the extension of a family of biorthogonal Coifman wavelet systems. *The ANZIAM J.* **46**, 111–120 (2004)
37. M. Kobayashi, Wavelets and their applications: Case studies. Technical report, IBM Tokyo Research Lab (1998)
38. P. Kumar, Wavelet analysis for geophysical applications. *Rev. Geophys.* **35**(4), 385–412 (1997)
39. F. Kurth, M. Clausen, Filter bank tree and M-band wavelet packet algorithms in audio signal processing. *IEEE Trans. Sign. Process.* **47**(2), 549–554 (1999)
40. M.S. Lewicki, Efficient coding of natural sounds. *Nat. Neurosci.* **5**(4), 356–363 (2002)
41. T. Lin, S. Xu, Q. Shi, P. Hao, An algebraic construction of orthonormal M-band wavelets with perfect reconstruction. *Appl. Math. Comput.* **172**, 717–730 (2006)

42. P. Lio, Wavelets in bioinformatics and computational biology: State of art and perspectives. *Bioinform. Rev.* **19**(1), 2–9 (2003)
43. Z. Liu, N. Zheng, Parametrization construction of biorthogonal wavelet filter banks for image coding. *Sign. Image Video Process.* **1**, 63–76 (2007)
44. S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way* (Academic Press, 2008)
45. S.G. Mallat, A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **11**(7), 674–693 (1989)
46. M. Maslen, P. Abbott, Automation of the lifting factorization of wavelet transforms. *Comput. Phys. Commun.* **127**, 309–326 (2000)
47. Y. Meyer, *Principe D'incertitude, Bases Hilbertiennes Et Algebres D'operateurs* (1985). *Seminaire Bourbaki*, n. 662. http://archive.numdam.org/article/SB_1985-1986_28_209_0.pdf
48. M. Misiti, Y. Misiti, G. Oppenheim, J.M. Poggi, *Wavelets and Their Applications* (ISTE, London, 2007)
49. J. Morlet, A. Grossman, Decomposition of Hardy functions into square integrable wavelets of constant shape. *SIAM J. Math. Anal.* **15**, 723–736 (1984)
50. H. Nagendra, S. Mukherjee, V. Kumar, Application of wavelet techniques in ECG signal processing: An overview. *Int. J. Eng. Sci. Technol.* **3**(10), 7432–7443 (2011)
51. A. Nait-Ali, *Advanced Biosignal Processing* (Springer, 2009)
52. K. Najarian, R. Splinter, *Biomedical Signal and Image Processing* (CRC Press, 2012)
53. B.D. Patil, P.G. Patwardhan, V.M. Gadre, On the design of FIR wavelet filter banks using factorization of a halfband polynomial. *IEEE Sign. Process. Lett.* **15**, 485–488 (2008)
54. Z.K. Peng, F.L. Chu, Application of the wavelet transform in machine condition monitoring and fault diagnostics: A review with bibliography. *Mech. Syst. Sign. Process.* **18**, 199–221 (2004)
55. G. Qi, Wavelet-based AE characterization of composite materials. *NDT E Int.* **33**(3), 133–144 (2000)
56. M. Raphan, E.P. Simoncelli, Optimal denoising in redundant representations. *IEEE Trans. Image Process.* **17**(8), 1342–1352 (2008)
57. B. Rivard, J. Feng, A. Gallie, A. Sanchez-Azofeifa, Continuous wavelets for the improved use of spectral libraries and hyperspectral data. *Remote Sens. Environ.* **112**, 2850–2862 (2008)
58. J.L. Semmlow, *Biosignal and Biomedical Image Processing* (CRC Press, 2008)
59. X.G. Shao, A.K. Leung, F.T. Chau, Wavelet: A new trend in chemistry. *Acc. Chem. Res.* **36**(4), 276–283 (2003)
60. M.C. Shou, L.P. Leu, Energy of power spectral density function and wavelet analysis of absolute pressure fluctuation measurements in fluidized beds. *Chem. Eng. Res. Des.* **83**(5), 478–491 (2005)
61. F.J. Simons, B.D.E. Dando, R.M. Allen, Automatic detection and rapid determination of earthquake magnitude by wavelet multiscale analysis of the primary arrival. *EarthPlanet. Sci. Lett.* **250**, 214–223 (2006)
62. F.J. Simons, I. Loris, E. Brevdo, I. Daubechies, Wavelets and wavelet-like transforms on the sphere and their application to geophysical data inversion. *Proc. SPIE* **8138**, 1–15 (2011)
63. S. Sinha, P.S. Routh, P.D. Anno, J.P. Castagna, Spectral decomposition of seismic data with continuous-wavelet transform. *Geophysics* **70**, 19–25 (2005)
64. A.N. Skodras, C.A. Christopoulos, T. Ebrahimi, JPEG2000: The upcoming still image compression standard. *Pattern Recogn. Lett.* **22**(12), 1337–1345 (2001)
65. V. Strela, P.N. Heller, G. Strang, P. Topiwala, C. Heil, The application of multiwavelet filter-banks to image processing. *IEEE T. Image Proc.* **8**(4), 548–563 (1999)
66. Z. Sun, C.C. Chang, Structural damage assessment based on wavelet packet transform. *J. Struct. Eng.* **128**(10), 1354–1361 (2002)
67. W. Sweldens, The lifting scheme: A construction of second generation wavelets. *SIAM. J. Math. Anal.* **29**(2), 511–546 (1997)
68. D.B.H. Tay, Rationalizing the coefficients of popular biorthogonal wavelet filters. *IEEE Trans. Circ. Syst. Video Technol.* **10**(6), 998–1005 (2000)

69. D.B.H. Tay, M. Palaniswami, A novel approach to the design of the class of triplet halfband filterbanks. *IEEE Trans. Circ. Syst.-II: Express Briefs* **51**(7), 378–383 (2004)
70. J. Tian, R.O. Wells Jr, Vanishing moments and biorthogonal wavelet systems, in *Mathematics in Signal Processing IV*, ed. by Mc.Whirter (Oxford University Press, 1997)
71. A.C. To, J.R. Moore, S.D. Glaser, Wavelet denoising techniques with applications to experimental geophysical data. *Sign. Process.* **89**, 144–160 (2009)
72. F. Truchelet, O. Laligant, Wavelets in industrial applications: A review, in *Proceedings SPIE*, vol. 5607 (2004), pp. 1–14
73. F. Truchelet, O. Laligant, Review of industrial applications of wavelet and multiresolution-based signal and image processing. *J. Electron. Imaging* **17**(3) (2008)
74. M. Unser, A. Aldroubi, A review of wavelets in biomedical applications. *Proc. IEEE* **84**(4), 626–638 (1996)
75. M. Unser, T. Blu, Mathematical properties of the JPEG2000 wavelet filters. *IEEE Trans. Image Process.* **12**(9), 1080–1090 (2003)
76. G. Uytterhoeven, D. Roose, A. Bultheel, Wavelet transforms using the lifting scheme. Technical report, Katholieke Universiteit Leuven, 1997. ITA-Wavelets-WP.1.1
77. M. Vetterli, J. Kovacevic. *Wavelets and Subband Coding* (Prentice Hall, 1995)
78. D. Wei. *Coiflet-type Wavelets: Theory, Design, and Applications*. PhD thesis, University of Texas at Austin (1998)
79. X. Yang, Y. Shi, B. Yang, General framework of the construction of biorthogonal wavelets based on Bernstein bases: Theory analysis and application in image compression. *IET Comput. Vision* **5**(1), 50–67 (2011)
80. R. Yu, A. Baradarani, Design of halfband filters for orthogonal wavelets via sum of squares decomposition. *IEEE Sign. Process.* **15**, 437–440 (2008)
81. X. Zhang, Design of FIR halfband filters for orthonormal wavelets using Remez exchange algorithm. *IEEE Sign. Proces. Lett.* **16**(9), 814–817 (2009)

<http://www.springer.com/978-981-10-2536-5>

Digital Signal Processing with Matlab Examples, Volume
2

Decomposition, Recovery, Data-Based Actions

Giron-Sierra, J.M.

2017, XXXIX, 913 p. 659 illus., 279 illus. in color.,

Hardcover

ISBN: 978-981-10-2536-5