Chapter 5

Motivation for wavelets and some simple examples

In the first part of the book our focus was to approximate functions or vectors with trigonometric functions. We saw that the Discrete Fourier transform could be used to obtain a representation of a vector in terms of such functions, and that computations could be done efficiently with the FFT algorithm. This was useful for analyzing, filtering, and compressing sound and other discrete data. The approach with trigonometric functions has some limitations, however. One of these is that, in a representation with trigonometric functions, the frequency content is fixed over time. This is in contrast with most sound data, where the characteristics are completely different in different parts. We have also seen that, even if a sound has a simple representation in terms of trigonometric functions on two different parts, the representation of the entire sound may not be simple. In particular, if the function is nonzero only on a very small interval, a representation of it in terms of trigonometric functions is not so simple.

In this chapter we are going to introduce the basic properties of an alternative to Fourier analysis for representing functions. This alternative is called wavelets. Similar to Fourier analysis, wavelets are also based on the idea of expressing a function in some basis. But in contrast to Fourier analysis, where the basis is fixed, wavelets provide a general framework with many different types of bases. In this chapter we first give a motivation for wavelets, before we continue by introducing some very simple wavelets. The first wavelet we look at can be interpreted as an approximation scheme based on piecewise constant functions. The next wavelet we look at is similar, but with piecewise linear functions used instead. Following these examples we will establish a more general framework, based on experiences from the simple wavelets. In the following chapters we will interpret this framework in terms of filters, and use this connection to construct even more interesting wavelets.

The examples in this and the next chapters can be run from the notebook `applinalgnbchap5.ipynb`. Core functions are collected in a module called `dwt`.

5.1 Why wavelets?

The left image in Figure 5.1 shows a view of the entire Earth.

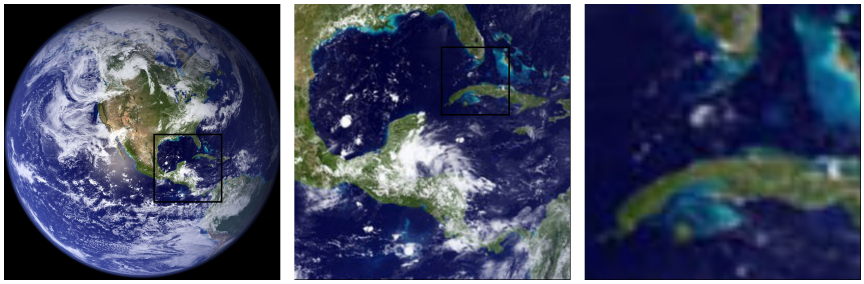

Figure 5.1: A view of Earth from space, together with versions of the image where we have zoomed in.

The startup image in Google Earth^{TM}, a program for viewing satellite images, maps and other geographic information, is very similar to this. In the middle image we have zoomed in on the Mexican Gulff, as marked with a rectangle in the left image. Similarly, in the right image we have further zoomed in on Cuba and a small portion of Florida, as marked with a rectangle in the middle image. There is clearly an amazing amount of information available behind a program like Google Earth^{TM}, since we there can zoom further in, and obtain enough detail to differentiate between buildings and even trees or cars all over the Earth. So, when the Earth is spinning in the opening screen of Google Earth^{TM}, all the Earth's buildings appear to be spinning with it! If this was the case the Earth would not be spinning on the screen, since there would just be so much information to process that a laptop would not be able to display a rotating Earth.

There is a simple reason that the globe can be shown spinning in spite of the huge amounts of information that need to be handled. We are going to see later that a digital image is just a rectangular array of numbers that represent the color at a dense set of points. As an example, the images in Figure 5.1 are made up of a grid of 1064×1064 points, which gives a total of 1 132 096 points. The color at a point is represented by three eight-bit integers, which means that the image files contain a total of 3 396 288 bytes each. So regardless of how close to the surface of the Earth our viewpoint is, the resulting image always contains the same number of points. This means that when we are far away from the Earth we can use a very coarse model of the geographic information that is being displayed, but as we zoom in, we need to display more details and therefore need a more accurate model.

Observation 5.1. *Images models.*

When discrete information is displayed in an image, there is no need to use a mathematical model that contains more detail than what is visible in the image.

A consequence of Observation 5.1 is that for applications like Google Earth$^{\text{TM}}$ we should use a mathematical model that makes it easy to switch between different levels of detail, or different resolutions. Such models are called *multiresolution models*, and wavelets are prominent examples of this kind of models. We will see that multiresolution models also provide us with means of approximating functions, just as Taylor series and Fourier series. Our new approximation scheme differs from these in one important respect, however: When we approximate with Taylor series and Fourier series, the error must be computed at the same data points as well, so that the error contains just as much information as the approximating function, and the function to be approximated. Multiresolution models on the other hand will be defined in such a way that the error and the "approximating function" each contain half of the information from the function we approximate, i.e. their amount of data is reduced. This property makes multiresolution models attractive for the problems at hand, when compared to approaches such as Taylor series and Fourier series.

When we zoom in with Google Earth$^{\text{TM}}$, it seems that this is done continuously. The truth is probably that the program only has representations at some given resolutions (since each representation requires memory), and that one interpolates between these to give the impression of a continuous zoom. In the coming chapters we will first look at how we can represent the information at different resolutions, so that only new information at each level is included.

We will now turn to how wavelets are defined more formally, and construct the simplest wavelet we have. Its construction goes in the following steps: First we introduce what we call resolution spaces, and the corresponding scaling function. Then we introduce the detail spaces, and the corresponding mother wavelet. These two functions will give rise to certain bases for these spaces, and we will define the Discrete Wavelet Transform as a change of coordinates between these bases.

5.2 A wavelet based on piecewise constant functions

Our starting point will be the space of piecewise constant functions on an interval $[0, N)$. This will be called a resolution space.

Definition 5.2. *The resolution space V_0.*

Let N be a natural number. The resolution space V_0 is defined as the space of functions defined on the interval $[0, N)$ that are constant on each subinterval $[n, n+1)$ for $n = 0, \ldots, N-1$.

Note that this also corresponds to piecewise constant functions which are periodic with period N. We will, just as we did in Fourier analysis, identify a function defined on $[0, N)$ with its (period N) periodic extension. An example of a function in V_0 for $N = 10$ is shown in Figure 5.2. It is easy to check that V_0 is a linear space, and for computations it is useful to know the dimension of the space and have a basis.

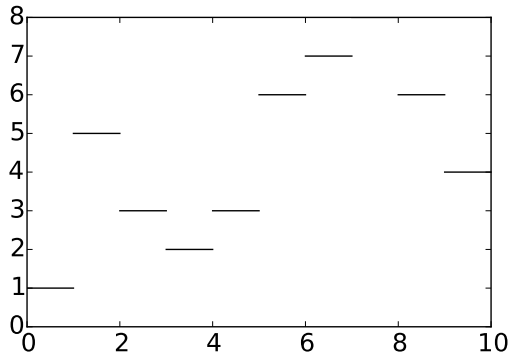

Figure 5.2: A piecewise constant function.

Lemma 5.3. *The function ϕ.*
Define the function $\phi(t)$ by

$$\phi(t) = \begin{cases} 1, & \text{if } 0 \le t < 1; \\ 0, & \text{otherwise;} \end{cases} \tag{5.1}$$

and set $\phi_n(t) = \phi(t-n)$ for any integer n. The space V_0 has dimension N, and the N functions $\{\phi_n\}_{n=0}^{N-1}$ form an orthonormal basis for V_0 with respect to the standard inner product

$$\langle f, g \rangle = \int_0^N f(t)g(t)\,dt. \tag{5.2}$$

In particular, any $f \in V_0$ can be represented as

$$f(t) = \sum_{n=0}^{N-1} c_n \phi_n(t) \tag{5.3}$$

for suitable coefficients $(c_n)_{n=0}^{N-1}$. The function ϕ_n is referred to as the *characteristic* function of the interval $[n, n+1)$

Note the small difference between the inner product we define here from the inner product we used for functions previously: Here there is no scaling $1/T$ involved. Also, for wavelets we will only consider real functions, and the inner product will therefore not be defined for complex functions. Two examples of the basis functions defined in Lemma 5.5 are shown in Figure 5.3.

Proof. Two functions ϕ_{n_1} and ϕ_{n_2} with $n_1 \neq n_2$ clearly satisfy $\int \phi_{n_1}(t)\phi_{n_2}(t)dt = 0$ since $\phi_{n_1}(t)\phi_{n_2}(t) = 0$ for all values of x. It is also easy to check that $\|\phi_n\| = 1$ for all n. Finally, any function in V_0 can be written as a linear combination the functions ϕ_0, ϕ_1, …, ϕ_{N-1}, so the conclusion of the lemma follows. □

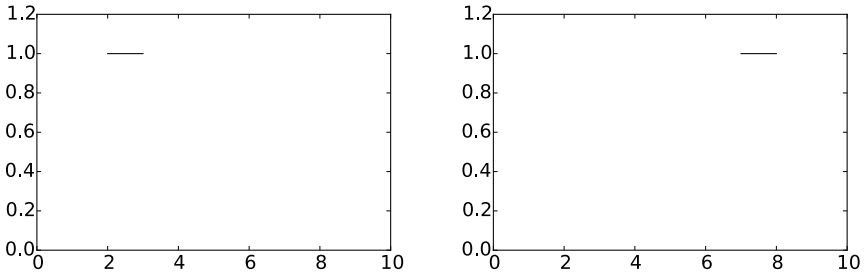

Figure 5.3: The basis functions ϕ_2 and ϕ_7 from $\boldsymbol{\phi}_0$.

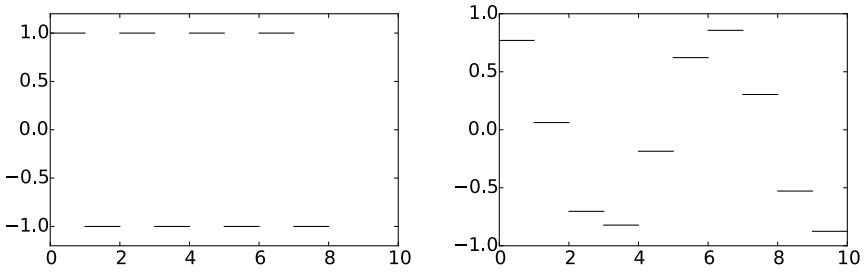

Figure 5.4: Examples of functions from V_0. The square wave in V_0 (left), and an approximation to $\cos t$ from V_0 (right).

In our discussion of Fourier analysis, the starting point was the function $\sin(2\pi t)$ that has frequency 1. We can think of the space V_0 as being analogous to this function: The function $\sum_{n=0}^{N-1}(-1)^n\phi_n(t)$ is (part of the) square wave that we discussed in Chapter 1, and which also oscillates regularly like the sine function, see the left plot in Figure 5.4. The difference is that we have more flexibility since we have a whole space at our disposal instead of just one function — the right plot in Figure 5.4 shows another function in V_0.

In Fourier analysis we obtained a linear space of possible approximations by including sines of frequency 1, 2, 3, ..., up to some maximum. We use a similar approach for constructing wavelets, but we double the frequency each time and label the spaces as V_0, V_1, V_2, ...

Definition 5.4. *Refined resolution spaces.*

The space V_m for the interval $[0, N)$ is the space of piecewise linear functions defined on $[0, N)$ that are constant on each subinterval $[n/2^m, (n+1)/2^m)$ for $n = 0, 1, \ldots, 2^m N - 1$.

Some examples of functions in the spaces V_1, V_2 and V_3 for the interval $[0, 10]$ are shown in Figure 5.5. As m increases, we can represent smaller details. In particular, the function in the rightmost is a piecewise constant function that oscillates like $\sin(2\pi 2^2 t)$ on the interval $[0, 10]$.

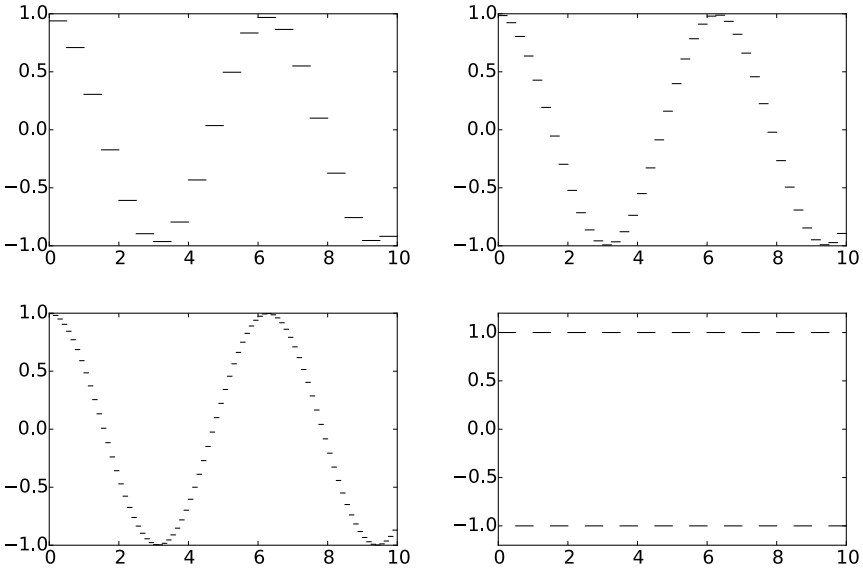

Figure 5.5: Piecewise constant approximations to $\cos t$ on the interval $[0, 10]$ in the spaces V_1, V_2, and V_3. The lower right plot shows the square wave in V_2.

It is easy to find a basis for V_m, we just use the characteristic functions of each subinterval.

Lemma 5.5. *Basis for V_m.*

Let $[0, N)$ be a given interval with N some positive integer. Then the dimension of V_m is $2^m N$. The functions

$$\phi_{m,n}(t) = 2^{m/2}\phi(2^m t - n), \quad \text{for } n = 0,\, 1,\, \ldots,\, 2^m N - 1. \tag{5.4}$$

$\{\phi_{m,n}\}_{n=0}^{2^m N-1}$ form an orthonormal basis for V_m, which we will denote by $\boldsymbol{\phi}_m$. Any function $f \in V_m$ can thus be represented uniquely as

$$f(t) = \sum_{n=0}^{2^m N-1} c_{m,n}\phi_{m,n}(t).$$

Proof. The functions given by Equation (5.4) are nonzero on the subintervals $[n/2^m, (n+1)/2^m)$ which we referred to in Definition 5.4, so that $\phi_{m,n_1}\phi_{m,n_2} = 0$ when $n_1 \neq n_2$, since these intervals are disjoint. The only mysterious thing may be the normalisation factor $2^{m/2}$. This comes from the fact that

$$\int_0^N \phi(2^m t - n)^2\, dt = \int_{n/2^m}^{(n+1)/2^m} \phi(2^m t - n)^2\, dt = 2^{-m}\int_0^1 \phi(u)^2\, du = 2^{-m}.$$

The normalisation therefore thus ensures that $\|\phi_{m,n}\| = 1$ for all m. □

In the following we will always denote the coordinates in the basis $\boldsymbol{\phi}_m$ by $c_{m,n}$. Note that our definition restricts the dimensions of the spaces we study to be on the form $N2^m$. In Chapter 6 we will explain how this restriction can be dropped, but until then the dimensions will be assumed to be on this form. In the theory of wavelets, the function ϕ is also called a *scaling function*. The origin behind this name is that the scaled (and translated) functions $\phi_{m,n}$ of ϕ are used as basis functions for the refined resolution spaces. Later on we will see that other scaling functions ϕ can be chosen, where the scaled versions $\phi_{m,n}$ will be used to define similar resolution spaces, with slightly different properties.

5.2.1 Function approximation property

Each time m is increased by 1, the dimension of V_m doubles, and the subinterval on which the functions in V_m are constant are halved in size. It therefore seems reasonable that, for most functions, we can find good approximations in V_m provided m is big enough.

Theorem 5.6. *Resolution spaces and approximation.*

Let f be a given function that is continuous on the interval $[0, N]$. Given $\epsilon > 0$, there exists an integer $m \geq 0$ and a function $g \in V_m$ such that

$$\big|f(t) - g(t)\big| \leq \epsilon$$

for all t in $[0, N]$.

Proof. Since f is (uniformly) continuous on $[0, N]$, we can find an integer m so that $\big|f(t_1) - f(t_2)\big| \leq \epsilon$ for any two numbers t_1 and t_2 in $[0, N]$ with $|t_1 - t_2| \leq 2^{-m}$. Define the approximation g by

$$g(t) = \sum_{n=0}^{2^m N-1} f\big(t_{m,n+1/2}\big)\phi_{m,n}(t),$$

where $t_{m,n+1/2}$ is the midpoint of the subinterval $\big[n2^{-m}, (n+1)2^{-m}\big)$,

$$t_{m,n+1/2} = (n + 1/2)2^{-m}.$$

For t in this subinterval we then obviously have $|f(t) - g(t)| \leq \epsilon$, and since these intervals cover $[0, N]$, the conclusion holds for all $t \in [0, N]$. □

Theorem 5.6 does not tell us how to find the approximation g although the proof makes use of an approximation that interpolates f at the midpoint of each subinterval. Note that if we measure the error in the L^2-norm, we have

$$\|f - g\|^2 = \int_0^N \big|f(t) - g(t)\big|^2 \, dt \leq N\epsilon^2,$$

so $\|f - g\| \leq \epsilon\sqrt{N}$. We therefore have the following corollary.

Corollary 5.7. *Resolution spaces and approximation.*

Let f be a given continuous function on the interval $[0, N]$. Then

$$\lim_{m\to\infty} \|f - \text{proj}_{V_m}(f)\| = 0.$$

Figure 5.6 illustrates how some of the approximations of the function $f(x) = x^2$ from the resolution spaces for the interval $[0, 1]$ improve with increasing m.

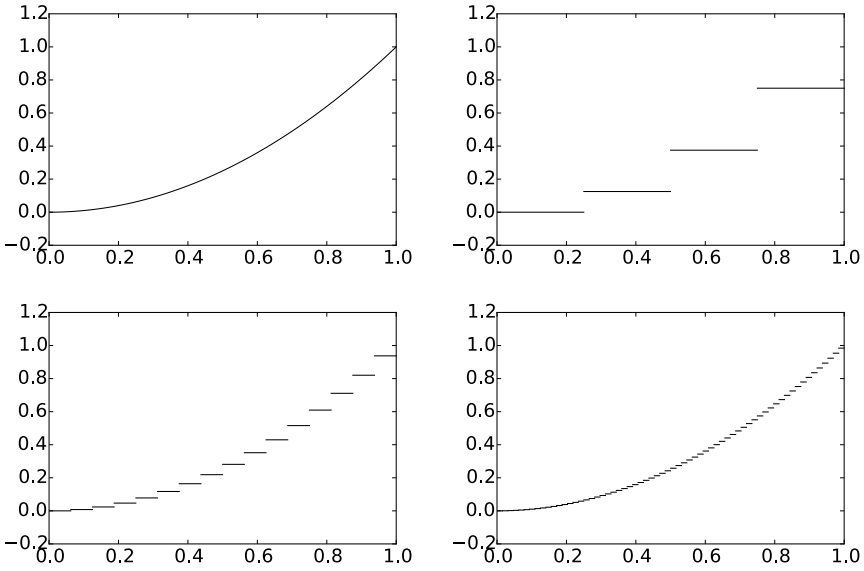

Figure 5.6: Comparison of the function defined by $f(t) = t^2$ on $[0, 1]$ with the projection onto V_2, V_4, and V_6, respectively.

5.2.2 Detail spaces and wavelets

So far we have described a family of function spaces that allow us to determine arbitrarily good approximations to a continuous function. The next step is to introduce the so-called detail spaces and the wavelet functions. We start by observing that since

$$[n, n+1) = [2n/2, (2n+1)/2) \cup [(2n+1)/2, (2n+2)/2),$$

we have

$$\phi_{0,n} = \frac{1}{\sqrt{2}}\phi_{1,2n} + \frac{1}{\sqrt{2}}\phi_{1,2n+1}.$$

This provides a formal proof of the intuitive observation that $V_0 \subset V_1$, for if $g \in V_0$, we can write

$$g(t) = \sum_{n=0}^{N-1} c_{0,n}\phi_{0,n}(t) = \sum_{n=0}^{N-1} c_{0,n}\big(\phi_{1,2n} + \phi_{1,2n+1}\big)/\sqrt{2},$$

and the right-hand side clearly lies in V_1. Since also

$$\begin{aligned}\phi_{m-1,n}(t) &= 2^{(m-1)/2}\phi(2^{m-1}t - n) = 2^{(m-1)/2}\phi_{0,n}(2^{m-1}t)\\ &= 2^{(m-1)/2}\frac{1}{\sqrt{2}}(\phi_{1,2n}(2^{m-1}t) + \phi_{1,2n+1}(2^{m-1}t))\\ &= 2^{(m-1)/2}(\phi(2^m t - 2n) + \phi(2^m t - (2n+1))) = \frac{1}{\sqrt{2}}(\phi_{m,2n}(t) + \phi_{m,2n+1}(t)),\end{aligned}$$

we also have that

$$\phi_{m-1,n} = \frac{1}{\sqrt{2}}\phi_{m,2n} + \frac{1}{\sqrt{2}}\phi_{m,2n+1}, \tag{5.5}$$

so that also $V_k \subset V_{k+1}$ for any integer $k \geq 0$.

Lemma 5.8. *Resolution spaces are nested.*
The spaces V_0, V_1, ..., V_m, ... are nested,

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_m \cdots .$$

This means that it is meaningful to project V_{k+1} onto V_k. The next step is to characterize the projection from V_1 onto V_0, and onto the orthogonal complement of V_0 in V_1. Before we do this, let us make the following definitions.

Definition 5.9. *Detail spaces.*
The orthogonal complement of V_{m-1} in V_m is denoted W_{m-1}. All the spaces W_k are also called detail spaces, or error spaces.

The name detail space is used since the projection from V_m onto V_{m-1} in considered as a (low-resolution) approximation, and the error, which lies in W_{m-1}, is the detail which is left out when we replace with this approximation. We will also write $g_m = g_{m-1} + e_{m-1}$ when we split $g_m \in V_m$ into a sum of a low-resolution approximation and a detail component. In the context of our Google Earth$^{\text{TM}}$example, in Figure 5.1 you should interpret g_0 as the left image, the middle image as an excerpt of g_1, and e_0 as the additional details which are needed to reproduce the middle image from the left image.

Since V_0 and W_0 are mutually orthogonal spaces they are also linearly independent spaces. When U and V are two such linearly independent spaces, we will write $U \oplus V$ for the vector space consisting of all vectors of the form $\boldsymbol{u} + \boldsymbol{v}$, with $\boldsymbol{u} \in U$, $\boldsymbol{v} \in V$. $U \oplus V$ is also called the *direct sum* of U and V. This also makes sense if we have more than two vector spaces (such as $U \oplus V \oplus W$),

and the direct sum clearly obeys the associate law $U \oplus (V \oplus W) = (U \oplus V) \oplus W$. Using the direct sum notation, we can first write

$$V_m = V_{m-1} \oplus W_{m-1}. \tag{5.6}$$

Since V_m has dimension $2^m N$, it follows that also W_m has dimension $2^m N$. We can continue the direct sum decomposition by also writing V_{m-1} as a direct sum, then V_{m-2} as a direct sum, and so on, and end up with

$$V_m = V_0 \oplus W_0 \oplus W_1 \oplus \cdots \oplus W_{m-1}, \tag{5.7}$$

where the spaces on the right hand side have dimension $N, N, 2N, \ldots, 2^{m-1}N$. This decomposition wil be important for our purposes. It says that the resolution space V_m acan be written as the sum of a lower order resolution space V_0, and m detail spaces $W_0, \ldots, W_{m-1}$. We will later interpret this splitting into a low-resolution component and m detail components.

It turns out that the following function will play the same role for the detail space W_k as the function ϕ plays for the resolution space V_k.

Definition 5.10. *The function ψ.*

We define

$$\psi(t) = \big(\phi_{1,0}(t) - \phi_{1,1}(t)\big)/\sqrt{2} = \phi(2t) - \phi(2t-1), \tag{5.8}$$

and

$$\psi_{m,n}(t) = 2^{m/2}\psi(2^m t - n), \quad \text{for } n = 0,\, 1,\, \ldots,\, 2^m N - 1. \tag{5.9}$$

The functions ϕ and ψ are shown in Figure 5.7.

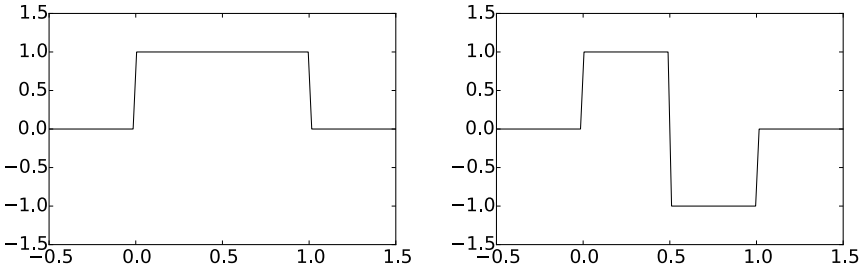

Figure 5.7: The functions ϕ and ψ we used to analyse the space of piecewise constant functions.

As in the proof for Equation (5.5), it follows that

$$\psi_{m-1,n} = \frac{1}{\sqrt{2}}\phi_{m,2n} - \frac{1}{\sqrt{2}}\phi_{m,2n+1}, \tag{5.10}$$

Clearly ψ is supported on $[0, 1)$, and $\|\psi\| = 1$. From this it follows as for $\boldsymbol{\phi}_0$ that the $\{\psi_{0,n}\}_{n=0}^{N-1}$ are orthonormal. In the same way as for $\boldsymbol{\phi}_m$, it follows

also that the $\{\psi_{m,n}\}_{n=0}^{2^m N-1}$ is orthonormal for any m. We will write $\boldsymbol{\psi}_m$ for the orthonormal basis $\{\psi_{m,n}\}_{n=0}^{2^m N-1}$, and we will always denote the coordinates in the basis $\boldsymbol{\psi}_m$ by $w_{m,n}$. The next result motivates the definition of ψ, and states how we can project from V_1 onto V_0 and W_0, i.e. find the low-resolution approximation and the detail component of $g_1 \in V_1$.

Lemma 5.11. *Orthonormal bases.*
For $0 \le n < N$ we have that

$$\operatorname{proj}_{V_0}(\phi_{1,n}) = \begin{cases} \phi_{0,n/2}/\sqrt{2}, & \text{if } n \text{ is even;} \\ \phi_{0,(n-1)/2}/\sqrt{2}, & \text{if } n \text{ is odd.} \end{cases} \tag{5.11}$$

$$\operatorname{proj}_{W_0}(\phi_{1,n}) = \begin{cases} \psi_{0,n/2}/\sqrt{2}, & \text{if } n \text{ is even;} \\ -\psi_{0,(n-1)/2}/\sqrt{2}, & \text{if } n \text{ is odd.} \end{cases} \tag{5.12}$$

In particular, $\boldsymbol{\psi}_0$ is an orthonormal basis for W_0. More generally, if $g_1 = \sum_{n=0}^{2N-1} c_{1,n}\phi_{1,n} \in V_1$, then

$$\operatorname{proj}_{V_0}(g_1) = \sum_{n=0}^{N-1} c_{0,n}\phi_{0,n}, \text{ where } c_{0,n} = \frac{c_{1,2n} + c_{1,2n+1}}{\sqrt{2}} \tag{5.13}$$

$$\operatorname{proj}_{W_0}(g_1) = \sum_{n=0}^{N-1} w_{0,n}\psi_{0,n}, \text{ where } w_{0,n} = \frac{c_{1,2n} - c_{1,2n+1}}{\sqrt{2}}. \tag{5.14}$$

Proof. We first observe that $\phi_{1,n}(t) \neq 0$ if and only if $n/2 \le t < (n+1)/2$. Suppose that n is even. Then the intersection

$$\left[\frac{n}{2}, \frac{n+1}{2}\right) \cap [n_1, n_1 + 1) \tag{5.15}$$

is nonempty only if $n_1 = \frac{n}{2}$. Using the orthogonal decomposition formula we get

$$\begin{aligned} \operatorname{proj}_{V_0}(\phi_{1,n}) &= \sum_{k=0}^{N-1} \langle \phi_{1,n}, \phi_{0,k} \rangle \phi_{0,k} = \langle \phi_{1,n}, \phi_{0,n_1} \rangle \phi_{0,n_1} \\ &= \int_{n/2}^{(n+1)/2} \sqrt{2}\, dt\, \phi_{0,n/2} = \frac{1}{\sqrt{2}}\phi_{0,n/2}. \end{aligned}$$

Using this we also get

$$\begin{aligned} \operatorname{proj}_{W_0}(\phi_{1,n}) &= \phi_{1,n} - \frac{1}{\sqrt{2}}\phi_{0,n/2} = \phi_{1,n} - \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\phi_{1,n} + \frac{1}{\sqrt{2}}\phi_{1,n+1}\right) \\ &= \frac{1}{2}\phi_{1,n} - \frac{1}{2}\phi_{1,n+1} = \psi_{0,n/2}/\sqrt{2}. \end{aligned}$$

This proves the expressions for both projections when n is even. When n is odd, the intersection (5.15) is nonempty only if $n_1 = (n-1)/2$, which gives the expressions for both projections when n is odd in the same way. In particular we get

$$\begin{aligned}\mathrm{proj}_{W_0}(\phi_{1,n}) &= \phi_{1,n} - \frac{\phi_{0,(n-1)/2}}{\sqrt{2}} = \phi_{1,n} - \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\phi_{1,n-1} + \frac{1}{\sqrt{2}}\phi_{1,n}\right) \\ &= \frac{1}{2}\phi_{1,n} - \frac{1}{2}\phi_{1,n-1} = -\psi_{0,(n-1)/2}/\sqrt{2}.\end{aligned}$$

$\boldsymbol{\psi}_0$ must be an orthonormal basis for W_0 since $\boldsymbol{\psi}_0$ is contained in W_0, and both have dimension N.

We project the function g_1 in V_1 using the formulas in (5.11). We first split the sum into even and odd values of n,

$$g_1 = \sum_{n=0}^{2N-1} c_{1,n}\phi_{1,n} = \sum_{n=0}^{N-1} c_{1,2n}\phi_{1,2n} + \sum_{n=0}^{N-1} c_{1,2n+1}\phi_{1,2n+1}. \tag{5.16}$$

We can now apply the two formulas in (5.11),

$$\begin{aligned}\mathrm{proj}_{V_0}(g_1) &= \mathrm{proj}_{V_0}\left(\sum_{n=0}^{N-1} c_{1,2n}\phi_{1,2n} + \sum_{n=0}^{N-1} c_{1,2n+1}\phi_{1,2n+1}\right) \\ &= \sum_{n=0}^{N-1} c_{1,2n}\,\mathrm{proj}_{V_0}(\phi_{1,2n}) + \sum_{n=0}^{N-1} c_{1,2n+1}\,\mathrm{proj}_{V_0}(\phi_{1,2n+1}) \\ &= \sum_{n=0}^{N-1} c_{1,2n}\phi_{0,n}/\sqrt{2} + \sum_{n=0}^{N-1} c_{1,2n+1}\phi_{0,n}/\sqrt{2} \\ &= \sum_{n=0}^{N-1} \frac{c_{1,2n} + c_{1,2n+1}}{\sqrt{2}}\phi_{0,n}\end{aligned}$$

which proves Equation (5.13). Equation (5.14) is proved similarly. □

In Figure 5.8 we have used Lemma 5.11 to plot the projections of $\phi_{1,0} \in V_1$ onto V_0 and W_0. It is an interesting exercise to see from the plots why exactly these functions should be least-squares approximations of $\phi_{1,n}$. It is also an interesting exercise to prove the following from Lemma 5.11:

Proposition 5.12. *Projections.*

Let $f(t) \in V_1$, and let $f_{n,1}$ be the value f attains on $[n, n+1/2)$, and $f_{n,2}$ the value f attains on $[n+1/2, n+1)$. Then $\mathrm{proj}_{V_0}(f)$ is the function in V_0 which equals $(f_{n,1} + f_{n,2})/2$ on the interval $[n, n+1)$. Moreover, $\mathrm{proj}_{W_0}(f)$ is the function in W_0 which is $(f_{n,1} - f_{n,2})/2$ on $[n, n+1/2)$, and $-(f_{n,1} - f_{n,2})/2$ on $[n+1/2, n+1)$.

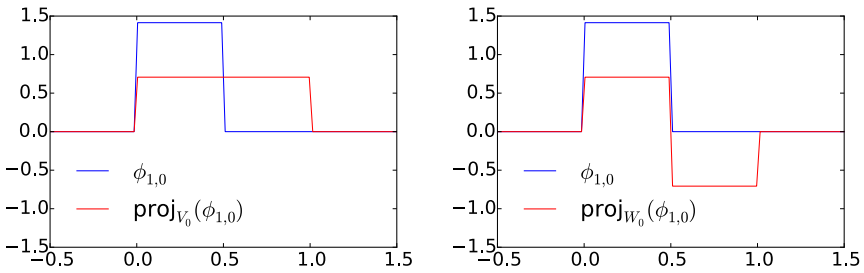

Figure 5.8: The projection of $\phi_{1,0} \in V_1$ onto V_0 and W_0.

In other words, the projection on V_0 is constructed by averaging on two subintervals, while the projection on W_0 is constructed by taking the difference from the mean. This sounds like a reasonable candidate for the least-squares approximations. In the exercise we generalize these observations.

In the same way as in Lemma 5.11, it is possible to show that

$$\text{proj}_{W_{m-1}}(\phi_{m,n}) = \begin{cases} \psi_{m-1,n/2}/\sqrt{2}, & \text{if } n \text{ is even;} \\ -\psi_{m-1,(n-1)/2}/\sqrt{2}, & \text{if } n \text{ is odd.} \end{cases} \tag{5.17}$$

From this it follows as before that $\boldsymbol{\psi}_m$ is an orthonormal basis for W_m. If $\{\mathcal{B}_i\}_{i=1}^n$ are mutually independent bases, we will in the following write $(\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n)$ for the basis where the basis vectors from $\mathcal{B}_i$ are included before $\mathcal{B}_j$ when $i < j$. With this notation, the decomposition in Equation (5.7) can be restated as follows

Theorem 5.13. *Bases for V_m.*
$\boldsymbol{\phi}_m$ and $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0, \boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_{m-1})$ are both bases for V_m.

The function ψ thus has the property that its dilations and translations together span the detail components. Later we will encounter other functions, which also will be denoted by ψ, and have similar properties. In the theory of wavelets, such ψ are called *mother wavelets*. There is one important property of ψ, which we will return to:

Observation 5.14. *Vanishing moment.*
We have that $\int_0^N \psi(t)dt = 0$.

This can be seen directly from the plot in Figure 5.7, since the parts of the graph above and below the x-axis cancel. In general we say that ψ has k vanishing moments if the integrals $\int t^l \psi(t)dt = 0$ for all $0 \leq l \leq k-1$. Due to Observation 5.14, ψ has one vanishing moment. In Chapter 7 we will show that mother wavelets with many vanishing moments are very desirable when it comes to approximation of functions.

We now have all the tools needed to define the Discrete Wavelet Transform.

Definition 5.15. *Discrete Wavelet Transform.*

The DWT (Discrete Wavelet Transform) is defined as the change of coordinates from $\boldsymbol{\phi}_1$ to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$. More generally, the m-level DWT is defined as the change of coordinates from $\boldsymbol{\phi}_m$ to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0, \boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_{m-1})$. In an m-level DWT, the change of coordinates from

$$(\boldsymbol{\phi}_{m-k+1}, \boldsymbol{\psi}_{m-k+1}, \boldsymbol{\psi}_{m-k+2}, \cdots, \boldsymbol{\psi}_{m-1}) \text{ to } (\boldsymbol{\phi}_{m-k}, \boldsymbol{\psi}_{m-k}, \boldsymbol{\psi}_{m-k+1}, \cdots, \boldsymbol{\psi}_{m-1}) \tag{5.18}$$

is also called the k'th stage. The (m-level) IDWT (Inverse Discrete Wavelet Transform) is defined as the change of coordinates the opposite way.

The DWT corresponds to replacing as many ϕ-functions as we can with ψ-functions, i.e. replacing the original function with a sum of as much detail at different resolutions as possible. We now can state the following result.

Theorem 5.16. *Expression for the DWT.*

If $g_m = g_{m-1} + e_{m-1}$ with

$$g_m = \sum_{n=0}^{2^m N-1} c_{m,n}\phi_{m,n} \in V_m,$$

$$g_{m-1} = \sum_{n=0}^{2^{m-1}N-1} c_{m-1,n}\phi_{m-1,n} \in V_{m-1} \quad e_{m-1} = \sum_{n=0}^{2^{m-1}N-1} w_{m-1,n}\psi_{m-1,n} \in W_{m-1},$$

then the change of coordinates from $\boldsymbol{\phi}_m$ to $(\boldsymbol{\phi}_{m-1}, \boldsymbol{\psi}_{m-1})$ (i.e. first stage in a DWT) is given by

$$\begin{pmatrix} c_{m-1,n} \\ w_{m-1,n} \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} c_{m,2n} \\ c_{m,2n+1} \end{pmatrix} \tag{5.19}$$

Conversely, the change of coordinates from $(\boldsymbol{\phi}_{m-1}, \boldsymbol{\psi}_{m-1})$ to $\boldsymbol{\phi}_m$ (i.e. the last stage in an IDWT) is given by

$$\begin{pmatrix} c_{m,2n} \\ c_{m,2n+1} \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} c_{m-1,n} \\ w_{m-1,n} \end{pmatrix} \tag{5.20}$$

Proof. Equations (5.5) and (5.10) say that

$$\phi_{m-1,n} = \phi_{m,2n}/\sqrt{2} + \phi_{m,2n+1}/\sqrt{2} \quad \psi_{m-1,n} = \phi_{m,2n}/\sqrt{2} - \phi_{m,2n+1}/\sqrt{2}.$$

The change of coordinate matrix from the basis $\{\phi_{m-1,n}, \psi_{m-1,n}\}$ to $\{\phi_{m,2n}, \phi_{m,2n+1}\}$ is thus $\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}$. This proves Equation (5.20). Equation (5.19) follows immediately since this matrix equals its inverse. □

Above we assumed that N is even. In Exercise 5.8 we will see how we can handle the case when N is odd.

From Theorem 5.16, we see that, if we had defined

$$\mathcal{C}_m = \{\phi_{m-1,0}, \psi_{m-1,0}, \phi_{m-1,1}, \psi_{m-1,1}, \cdots, \phi_{m-1,2^{m-1}N-1}, \psi_{m-1,2^{m-1}N-1}\}. \tag{5.21}$$

i.e. we have reordered the basis vectors in $(\boldsymbol{\phi}_{m-1}, \boldsymbol{\psi}_{m-1})$ (the subscript m is used since $\mathcal{C}_m$ is a basis for V_m), it is apparent from Equation (5.20) that $G = P_{\boldsymbol{\phi}_m \leftarrow \mathcal{C}_m}$ is the matrix where

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

is repeated along the main diagonal $2^{m-1}N$ times. Also, from Equation (5.19) it is apparent that $H = P_{\mathcal{C}_m \leftarrow \boldsymbol{\phi}_m}$ is the same matrix. Such matrices are called *block diagonal matrices*. This particular block diagonal matrix is clearly orthogonal. Let us make the following definition

Definition 5.17. *DWT and IDWT kernel transformations.*

The matrices $H = P_{\mathcal{C}_m \leftarrow \boldsymbol{\phi}_m}$ and $G = P_{\boldsymbol{\phi}_m \leftarrow \mathcal{C}_m}$ are called the *DWT and IDWT kernel transformations*. The DWT and the IDWT can be expressed in terms of these kernel transformations by

$$\text{DWT} = P_{(\boldsymbol{\phi}_{m-1}, \boldsymbol{\psi}_{m-1}) \leftarrow \mathcal{C}_m} H \text{ and IDWT} = G P_{\mathcal{C}_m \leftarrow (\boldsymbol{\phi}_{m-1}, \boldsymbol{\psi}_{m-1})},$$

respectively, where

- $P_{(\boldsymbol{\phi}_{m-1}, \boldsymbol{\psi}_{m-1}) \leftarrow \mathcal{C}_m}$ is a permutation matrix which groups the even elements first, then the odd elements,
- $P_{\mathcal{C}_m \leftarrow (\boldsymbol{\phi}_{m-1}, \boldsymbol{\psi}_{m-1})}$ is a permutation matrix which places the first half at the even indices, the last half at the odd indices.

Clearly, the kernel transformations H and G also invert each other. The point of using the kernel transformation is that they compute the output sequentially, similarly to how a filter does. Clearly also the kernel transformations are very similar to a filter, and we will return to this in the next chapter.

At each level in a DWT, V_k is split into one low-resolution component from V_{k-1}, and one detail component from W_{k-1}. We have illustrated this in figure 5.9, where the arrows represent changes of coordinates.

The detail component from W_{k-1} is not subject to further transformation. This is seen in the figure since $\boldsymbol{\psi}_{k-1}$ is a leaf node, i.e. there are no arrows going out from $\boldsymbol{\psi}_{m-1}$. In a similar illustration for the IDWT, the arrows would go the opposite way.

The Discrete Wavelet Transform is the analogue in a wavelet setting to the Discrete Fourier transform. When applying the DFT to a vector of length N,

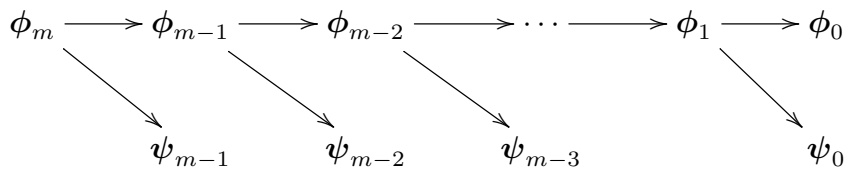

Figure 5.9: Illustration of a wavelet transform.

one starts by viewing this vector as coordinates relative to the standard basis. When applying the DWT to a vector of length N, one instead views the vector as coordinates relative to the basis $\boldsymbol{\phi}_m$. This makes sense in light of Exercise 5.1.

What you should have learned in this section.

- Definition of resolution spaces (V_m), detail spaces (W_m), scaling function (ϕ), and mother wavelet (ψ) for the wavelet based on piecewise constant functions.
- The nesting of resolution spaces, and how one can project from one resolution space onto a lower order resolution space, and onto its orthogonal complement.
- The definition of the Discrete Wavelet Transform as a change of coordinates, and how this can be written down from relations between basis functions.

Exercise 5.1: Samples are the coordinate vector

Show that the coordinate vector for $f \in V_0$ in the basis $\{\phi_{0,0}, \phi_{0,1}, \ldots, \phi_{0,N-1}\}$ is $(f(0), f(1), \ldots . f(N-1))$.

Exercise 5.2: Proposition 5.12

Prove Proposition 5.12.

Exercise 5.3: Computing projections

In this exercise we will consider the two projections from V_1 onto V_0 and W_0.

a) Consider the projection proj_{V_0} of V_1 onto V_0. Use Lemma 5.11 to write down the matrix for proj_{V_0} relative to the bases $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_0$.

b) Similarly, use Lemma 5.11 to write down the matrix for $\text{proj}_{W_0} : V_1 \to W_0$ relative to the bases $\boldsymbol{\phi}_1$ and $\boldsymbol{\psi}_0$.

Exercise 5.4: Computing projections 2

Consider again the projection proj_{V_0} of V_1 onto V_0.

a) Explain why $\text{proj}_{V_0}(\phi) = \phi$ and $\text{proj}_{V_0}(\psi) = 0$.

b) Show that the matrix of proj_{V_0} relative to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$ is given by the diagonal matrix where the first half of the entries on the diagonal are 1, the second half 0.

c) Show in a similar way that the projection of V_1 onto W_0 has a matrix relative to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$ given by the diagonal matrix where the first half of the entries on the diagonal are 0, the second half 1.

Exercise 5.5: Computing projections

Show that

$$\text{proj}_{V_0}(f) = \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)dt \right) \phi_{0,n}(t) \tag{5.22}$$

for any f. Show also that the first part of Proposition 5.12 follows from this.

Exercise 5.6: Finding the least squares error

Show that

$$\| \sum_n \left(\int_n^{n+1} f(t)dt \right) \phi_{0,n}(t) - f \|^2 = \langle f, f \rangle - \sum_n \left(\int_n^{n+1} f(t)dt \right)^2 .$$

This, together with the previous exercise, gives us an expression for the least-squares error for f from V_0 (at least after taking square roots). 2DO: Generalize to m

Exercise 5.7: Projecting on W_0

Show that

$$\text{proj}_{W_0}(f) = \sum_{n=0}^{N-1} \left(\int_n^{n+1/2} f(t)dt - \int_{n+1/2}^{n+1} f(t)dt \right) \psi_{0,n}(t) \tag{5.23}$$

for any f. Show also that the second part of Proposition 5.12 follows from this.

Exercise 5.8: When N is odd

When N is odd, the (first stage in a) DWT is defined as the change of coordinates from $(\phi_{1,0}, \phi_{1,1}, \ldots, \phi_{1,N-1})$ to

$$(\phi_{0,0}, \psi_{0,0}, \phi_{0,1}, \psi_{0,1}, \ldots, \phi_{0,(N-1)/2}, \psi_{(N-1)/2}, \phi_{0,(N+1)/2}).$$

Since all functions are assumed to have period N, we have that

$$\phi_{0,(N+1)/2} = \frac{1}{\sqrt{2}}(\phi_{1,N-1} + \phi_{1,N}) = \frac{1}{\sqrt{2}}(\phi_{1,0} + \phi_{1,N-1}).$$

From this relation one can find the last column in the change of coordinate matrix from $\boldsymbol{\phi}_0$ to $(\boldsymbol{\phi}_1, \boldsymbol{\psi}_1)$, i.e. the IDWT matrix. In particular, when N is odd, we see that the last column in the IDWT matrix circulates to the upper right corner. In terms of coordinates, we thus have that

$$c_{1,0} = \frac{1}{\sqrt{2}}(c_{0,0} + w_{0,0} + c_{0,(N+1)/2}) \qquad c_{1,N-1} = \frac{1}{\sqrt{2}} c_{0,(N+1)/2}. \tag{5.24}$$

a) If $N = 3$, the DWT matrix equals $\frac{1}{\sqrt{2}} \begin{pmatrix} 1\&1 & 1 \\ 1 & -1\&0 \\ 0\&0 & 1 \end{pmatrix}$, and the inverse of this is $\frac{1}{\sqrt{2}} \begin{pmatrix} 1\&1 & -1 & \\ 1 & -1 & -1 \\ 0\&0 & 2 & \end{pmatrix}$. Explain from this that, when N is odd, the DWT matrix can be constructed by adding a column on the form $\frac{1}{\sqrt{2}}(-1, -1, 0, \ldots, 0, 2)$ to the DWT matrices we had for N even (in the last row zeros are also added). In terms of the coordinates, we thus have the additional formulas

$$c_{0,0} = \frac{1}{\sqrt{2}}(c_{1,0+c_{1,1}-c_{1,N-1}}) \quad w_{0,0} = \frac{1}{\sqrt{2}}(c_{1,0-c_{1,1}-c_{1,N-1}}) \quad c_{0,(N+1)/2} = \frac{1}{\sqrt{2}} 2c_{1,N-1}. \tag{5.25}$$

b) Explain that the DWT matrix is orthogonal if and only if N is even. Also explain that it is only the last column which spoils the orthogonality.

5.3 Implementation of the DWT and examples

The DWT is straightforward to implement: One simply needs to iterate Equation (5.19) in the compendium for $m, m-1, \ldots, 1$. We will use a DWT kernel function which takes as input the coordinates $(c_{m,0}, c_{m,1}, \ldots)$, and returns the coordinates $(c_{m-1,0}, w_{m-1,0}, c_{m-1,1}, w_{m-1,1}, \ldots)$, i.e. computes one stage of the DWT. This is a different order for the coordinates than that given by the basis $(\boldsymbol{\phi}_m, \boldsymbol{\psi}_m)$. The reason is that it is easier with this new order to compute the DWT in-place. As

an example, the kernel transformation for the Haar wavelet can be implemented as follows. For simplicity this first version of the code assumes that N is even:

```
def DWTKernelHaar(x, symm, dual):
    x /= sqrt(2)
    for k in range(2,len(x) - 1,2):
        a, b = x[k] + x[k+1], x[k] - x[k+1]
        x[k], x[k+1] = a, b
```

Note that the code above accepts two-dimensional data, just as our function `FFTImpl` did. Thus, the function may be applied simultaneously to all channels in a sound. The mysterious parameters `symm` and `dual` will be explained in Chapter 6. For now they have no role in the code, but will still appear several places in the code in this section. When N is even, `IDWTKernelHaar` can be implemented with the exact same code. When N is odd, we can use the results from Exercise 5.8 (see also Exercise 5.24). The reason for using a general kernel function will be apparent later, when we change to different types of wavelets.

Since the code above does not give the coordinates in the same order as $(\boldsymbol{\phi}_m, \boldsymbol{\psi}_m)$, an implementation of the DWT needs to organize the DWT coefficients in the right order, in addition to calling the kernel function for each stage, and applying the kernel to the right coordinates. Clearly, the coordinates from $\boldsymbol{\phi}_m$ end up at indices $k2^m$, where m represents the current stage, and k runs through the indices. The following function, called `DWTImpl`, follows this procedure. It takes as input the number of levels, `nres`, as well as the input vector $\boldsymbol{x}$, runs the DWT on $\boldsymbol{x}$ with the given number of resolutions, and returns the result:

```
def DWTImpl(x, nres, f, symm=True, dual=False):
    for res in range(nres):
        f(x[0::2**res], symm, dual)
    reorganize_coefficients(x, nres, True)
```

Again note that the code is applied to all columns if the data is two-dimensional. Note also that here the kernel function `f` is first invoked, one time for each resolution. Finally, the coefficients are reorganized so that the $\boldsymbol{\phi}_m$ coordinates come first, followed by the coordinates from the different levels. We have provided a function `reorganize_coefficients` which does this reorganization, and you will be spared the details in this implementation. In Exercise 5.25 we go through some aspects of this implementation. Note that, although the DWT requires this reorganization, this reorganization may not be required in practice, as further processing is needed, for which the coefficients can be accessed where they have been placed after the in-place operations. Note also the two last arguments, `symm` and `dual`, which we have not commented on. We will return to these in Chapter 6. This implementation is not recursive, as the for-loop runs through the different stages. Inside the loop we perform the change of coordinates from $\boldsymbol{\phi}_k$ to $(\boldsymbol{\phi}_{k-1}, \boldsymbol{\psi}_{k-1})$ by applying Equation (5.19). This works on the first coordinates, since the coordinates from $\boldsymbol{\phi}_k$ are stored first in

$$(\boldsymbol{\phi}_k, \boldsymbol{\psi}_k, \boldsymbol{\psi}_{k+1}, \cdots, \boldsymbol{\psi}_{m-2}, \boldsymbol{\psi}_{m-1}).$$

Finally, the $\boldsymbol{c}$-coordinates are stored before the $\boldsymbol{w}$-coordinates. In this implementation, note that the first levels require the most multiplications, since the latter levels leave an increasing part of the coordinates unchanged. Note also that the change of coordinates matrix is a very sparse matrix: At each level a coordinate can be computed from only two of the other coordinates, so that this matrix has only two nonzero elements in each row/column. The algorithm clearly shows that there is no need to perform a full matrix multiplication to perform the change of coordinates.

The corresponding function for the IDWT, called IDWTImpl, goes as follows:

```
def IDWTImpl(x, nres, f, symm=True, dual=False):
    reorganize_coefficients(x, nres, False)
    for res in range(nres - 1, -1, -1):
        f(x[0::2**res], symm, dual)
```

Here the steps are simply performed in the reverse order, and by iterating Equation (5.20). You may be puzzled by the names `DWTKernelHaar` and `IDWTKernelHaar`. In the next sections we will consider other cases, where the underlying function ϕ may be a different function, not necessarily piecewise constant. It will turn out that much of the analysis we have done makes sense for other functions ϕ as well, giving rise to other structures which we also will refer to as wavelets. The wavelet resulting from piecewise constant functions is thus simply one example out of many, and it is commonly referred to as the *Haar wavelet*.

Let us round off this section with some important examples.

Example 5.18. *Computing the DWT by hand.*

In some cases, the DWT can be computed by hand, keeping in mind its definition as a change of coordinates. As an example, consider the simple vector $\boldsymbol{x}$ of length $2^{10} = 1024$ defined by

$$x_n = \begin{cases} 1 & \text{for } n < 512 \\ 0 & \text{for } n \geq 512, \end{cases}$$

and let us compute the 10-level DWT of this vector by first visualizing the function with these coordinates. Since $m = 10$ here, we should view $\boldsymbol{x}$ as coordinates in the basis $\boldsymbol{\phi}_{10}$ of a function $f(t) \in V_{10}$. This is $f(t) = \sum_{n=0}^{511} \phi_{10,n}$, and since $\phi_{10,n}$ is supported on $[2^{-10}n, 2^{-10}(n+1))$, the support of f has width $512 \times 2^{-10} = 1/2$ (512 translates, each with width 2^{-10}). Moreover, since $\phi_{10,n}$ is $2^{10/2} = 2^5 = 32$ on $[2^{-10}n, 2^{-10}(n+1))$ and 0 elsewhere, it is clear that

$$f(t) = \begin{cases} 32 & \text{for } 0 \leq t < 1/2 \\ 0 & \text{for } 0t \geq 1/2. \end{cases}$$

This is by definition a function in V_1: f must in fact be a multiplum of $\phi_{1,0}$, since this also is supported on $[0, 1/2)$. We can thus write $f(t) = c\phi_{1,0}(t)$ for some c. We can find c by setting $t = 0$. This gives that $32 = 2^{1/2}c$ (since $f(0) = 32$,

$\phi_{1,0}(0) = 2^{1/2}$), so that $c = 32/\sqrt{2}$. This means that $f(t) = \frac{32}{\sqrt{2}}\phi_{1,0}(t)$, f is in V_1, and with coordinates $(32/\sqrt{2}, 0, \ldots, 0)$ in $\boldsymbol{\phi}_1$.

When we run a 10-level DWT we make a change of coordinates from $\boldsymbol{\phi}_{10}$ to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0, \cdots, \boldsymbol{\psi}_9)$. The first 9 levels give us the coordinates in $(\boldsymbol{\phi}_1, \boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_9)$, and these are $(32/\sqrt{2}, 0, \ldots, 0)$ from what we showed. It remains thus only to perform the last level in the DWT, i.e. perform the change of coordinates from $\boldsymbol{\phi}_1$ to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$. Since $\phi_{1,0} = \frac{1}{\sqrt{2}}(\phi_{0,0} + \psi_{0,0})$, so that we get

$$f(t) = \frac{32}{\sqrt{2}}\phi_{1,0}(t) = \frac{32}{\sqrt{2}}\frac{1}{\sqrt{2}}(\phi_{0,0} + \psi_{0,0}) = 16\phi_{0,0} + 16\psi_{0,0}.$$

From this we see that the coordinate vector of f in $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0, \cdots, \boldsymbol{\psi}_9)$, i.e. the 10-level DWT of $\boldsymbol{x}$, is $(16, 16, 0, 0, \ldots, 0)$. Note that here V_0 and W_0 are both 1-dimensional, since V_{10} was assumed to be of dimension 2^{10} (in particular, $N = 1$).

It is straightforward to verify what we found using the algorithm above:

```
x = hstack([ones(512), zeros(512)])
DWTImpl(x, 10, DWTKernelHaar)
print x
```

The reason why the method from this example worked was that the vector we started with had a simple representation in the wavelet basis, actually it equaled the coordinates of a basis function in $\boldsymbol{\phi}_1$. Usually this is not the case, and our only possibility then is to run the DWT on a computer.

Example 5.19. *DWT and sound.*

When you run a DWT you may be led to believe that coefficients from the lower order resolution spaces may correspond to lower frequencies. This sounds reasonable, since the functions $\phi(2^m t - n) \in V_m$ change more quickly than $\phi(t - n) \in V_0$. However, the functions $\phi_{m,n}$ do not correspond to pure tones in the setting of wavelets. But we can still listen to sound from the different resolution spaces. In Exercise 5.19 you will be asked to implement a function which runs an m-level DWT on the first samples of the sound file `castanets.wav`, extracts the coefficients from the lower order resolution spaces or the detail spaces, transforms the values back to sound samples with the IDWT, and plays the result. When you listen to the result the sound is clearly recognizable for lower values of m, but is degraded for higher values of m. The explanation is that too much of the detail is omitted when you use a higher m. To be more precise, when listening to the sound by throwing away everything from the detail spaces $W_0, W_1, \ldots, W_{m-1}$, we are left with a 2^{-m} share of the data. Note that this procedure is mathematically not the same as setting some DFT coefficients to zero, since the DWT does not operate on pure tones.

It is of interest to plot the samples of our test audio file `castanets.wav`, and compare it with the first order DWT coefficients of the same samples. This is shown in Figure 5.10. The first half part of the plot represents the low-resolution approximation of the sound, the second half part represents the detail/error.

We see that the detail is quite significant in this case. This means that the first order wavelet approximation does not give a very good approximation to the sound. In the exercises we will experiment more on this.

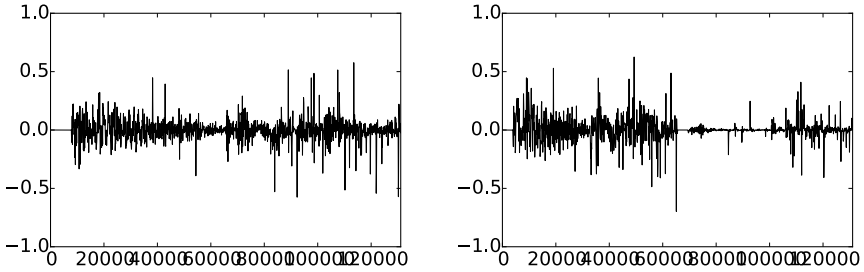

Figure 5.10: The 2^17 first sound samples (left) and the DWT coefficients (right) of the sound `castanets.wav`.

It is also interesting to plot only the detail/error in the sound, for different resolutions. For this, we must perform a DWT so that we get a representation in the basis $(\phi_0, \psi_0, \psi_1, \ldots, \psi_{m-1})$, set the coefficients from V_0 to zero, and transform back with the IDWT. In figure 5.11 the error is shown for the test audio file `castanets.wav` for $m = 1$, $m = 2$. This clearly shows that the error is larger when two levels of the DWT are performed, as one would suspect. It is also seen that the error is larger in the part of the file where there are bigger variations. This also sounds reasonable.

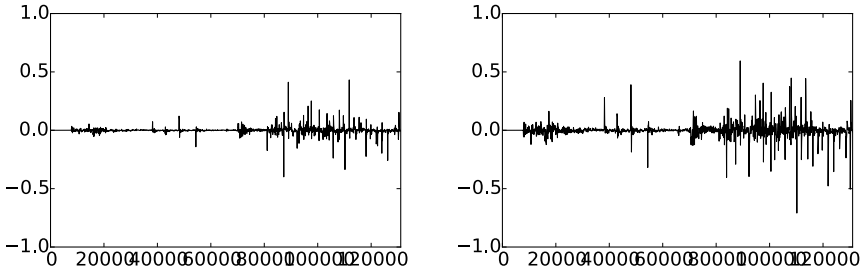

Figure 5.11: The error (i.e. the contribution from $W_0 \oplus W_1 \oplus \cdots \oplus W_{m-1}$) in the sound file `castanets.wav`, for $m = 1$ and $m = 2$, respectively.

The previous example illustrates that wavelets as well may be used to perform operations on sound. As we will see later, however, our main application for wavelets will be images, where they have found a more important role than for sound. Images typically display variations which are less abrupt than the ones found in sound. Just as the functions above had smaller errors in the corresponding resolution spaces than the sound had, images are thus more suited for for use with wavelets. The main idea behind why wavelets are so useful

comes from the fact that the detail, i.e., wavelet coefficients corresponding to the spaces W_k, are often very small. After a DWT one is therefore often left with a couple of significant coefficients, while most of the coefficients are small. The approximation from V_0 can be viewed as a good approximation, even though it contains much less information. This gives another reason why wavelets are popular for images: Detailed images can be very large, but when they are downloaded to a web browser, the browser can very early show a low-resolution of the image, while waiting for the rest of the details in the image to be downloaded. When we later look at how wavelets are applied to images, we will need to handle one final hurdle, namely that images are two-dimensional.

Example 5.20. *DWT on the samples of a mathematical function.*

Above we plotted the DWT coefficients of a sound, as well as the detail/error. We can also experiment with samples generated from a mathematical function. Figure 5.12 plots the error for different functions, with $N = 1024$.

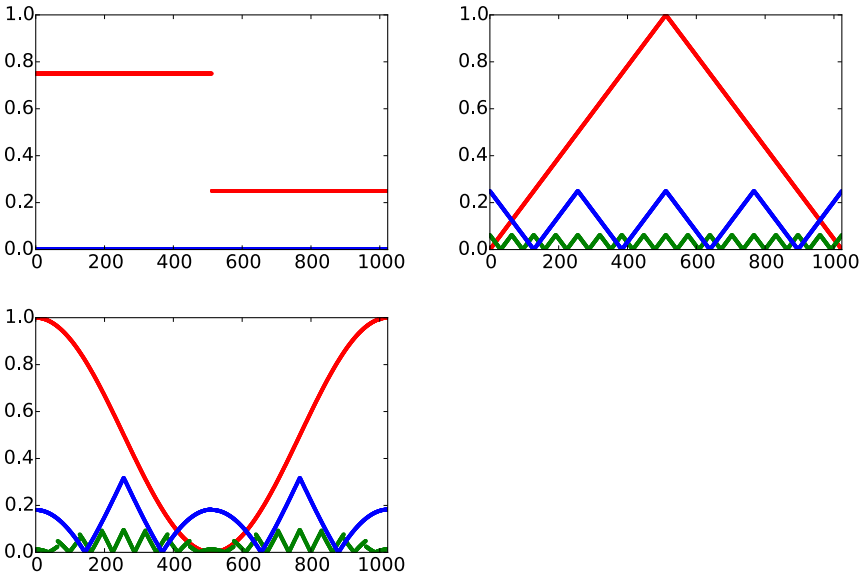

Figure 5.12: The error (i.e. the contribution from $W_0 \oplus W_1 \oplus \cdots \oplus W_{m-1}$) for $N = 1024$ when f is a square wave, the linear function $f(t) = 1 - 2|1/2 - t/N|$, and the trigonometric function $f(t) = 1/2 + \cos(2\pi t/N)/2$, respectively. The detail is indicated for $m = 6$ and $m = 8$.

In these cases, we see that we require large m before the detail/error becomes significant. We see also that there is no error for the square wave. The reason is that the square wave is a piecewise constant function, so that it can be represented exactly by the ϕ-functions. For the other functions, however, this is not the case, so we here get an error.

Above we used the functions `DWTImpl`, `IDWTImpl` to plot the error. For the functions we plotted in the previous example it is also possible to compute the wavelet coefficients, which we previously have denoted by $w_{m,n}$, exactly. You will be asked to do this in exercises 5.21 and 5.22. The following example shows the general procedure which can be used for this:

Example 5.21. *Computing the wavelet coefficients.*

Let us consider the function $f(t) = 1 - t/N$. This function decreases linearly from 1 to 0 on $[0, N]$, so that it is not piecewise constant, and does not lie in any of the spaces V_m. We can instead consider $\text{proj}_{V_m} f \in V_m$, and apply the DWT to this. Let us compute the $\boldsymbol{\psi}_m$-coordinates $w_{m,n}$ of $\text{proj}_{V_m} f$ in the orthonormal basis $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0, \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_{m-1})$. The orthogonal decomposition theorem says that

$$w_{m,n} = \langle f, \psi_{m,n} \rangle = \int_0^N f(t)\psi_{m,n}(t)dt = \int_0^N (1 - t/N)\psi_{m,n}(t)dt.$$

Using the definition of $\psi_{m,n}$ we see that this can also be written as

$$2^{m/2} \int_0^N (1 - t/N)\psi(2^m t - n)dt = 2^{m/2} \left(\int_0^N \psi(2^m t - n)dt - \int_0^N \frac{t}{N}\psi(2^m t - n)dt \right).$$

Using Observation 5.14 we get that $\int_0^N \psi(2^m t - n)dt = 0$, so that the first term above vanishes. Moreover, $\psi_{m,n}$ is nonzero only on $[2^{-m}n, 2^{-m}(n+1))$, and is 1 on $[2^{-m}n, 2^{-m}(n+1/2))$, and -1 on $[2^{-m}(n+1/2), 2^{-m}(n+1))$. We therefore get

$$\begin{aligned}
w_{m,n} &= -2^{m/2} \left(\int_{2^{-m}n}^{2^{-m}(n+1/2)} \frac{t}{N}dt - \int_{2^{-m}(n+1/2)}^{2^{-m}(n+1)} \frac{t}{N}dt \right) \\
&= -2^{m/2} \left(\left[\frac{t^2}{2N} \right]_{2^{-m}n}^{2^{-m}(n+1/2)} - \left[\frac{t^2}{2N} \right]_{2^{-m}(n+1/2)}^{2^{-m}(n+1)} \right) \\
&= -2^{m/2} \left(\left(\frac{2^{-2m}(n+1/2)^2}{2N} - \frac{2^{-2m}n^2}{2N} \right) - \left(\frac{2^{-2m}(n+1)^2}{2N} - \frac{2^{-2m}(n+1/2)^2}{2N} \right) \right) \\
&= -2^{m/2} \left(-\frac{2^{-2m}n^2}{2N} + \frac{2^{-2m}(n+1/2)^2}{N} - \frac{2^{-2m}(n+1)^2}{2N} \right) \\
&= -\frac{2^{-3m/2}}{2N} \left(-n^2 + 2(n+1/2)^2 - (n+1)^2 \right) = \frac{1}{N2^{2+3m/2}}.
\end{aligned}$$

We see in particular that $w_{m,n} \to 0$ when $m \to \infty$. Also, all coordinates were equal, i.e. $w_{m,0} = w_{m,1} = w_{m,2} = \cdots$. It is not too hard to convince oneself that this equality has to do with the fact that f is linear. We see also that there were a lot of computations even in this very simple example. For most functions we therefore usually do not compute $w_{m,n}$ symbolically, but instead run implementations like `DWTImpl`, `IDWTImpl` on a computer.

What you should have learned in this section.

- Definition of the m-level Discrete Wavelet Transform.
- Implementation of the Haar wavelet transform and its inverse.
- Experimentation with wavelets on sound.

Exercise 5.9: Implement IDWT for The Haar wavelet

Write a function `IDWTKernelHaar` which uses the formulas (5.24) in the compendium to implement the IDWT, similarly to how the function `DWTKernelHaar` implemented the DWT using the formulas (5.25) in the compendium.

Exercise 5.10: Computing projections

Generalize Exercise 5.4 to the projections from V_{m+1} onto V_m and W_m.

Exercise 5.11: Scaling a function

Show that $f(t) \in V_m$ if and only if $g(t) = f(2t) \in V_{m+1}$.

Exercise 5.12: Direct sums

Let $C_1, C_2 \ldots, C_n$ be independent vector spaces, and let $T_i : C_i \to C_i$ be linear transformations. The direct sum of T_1, T_2,...,T_n, written as $T_1 \oplus T_2 \oplus \ldots \oplus T_n$, denotes the linear transformation from $C_1 \oplus C_2 \oplus \cdots \oplus C_n$ to itself defined by

$$T_1 \oplus T_2 \oplus \ldots \oplus T_n(\boldsymbol{c}_1 + \boldsymbol{c}_2 + \cdots + \boldsymbol{c}_n) = T_1(\boldsymbol{c}_1) + T_2(\boldsymbol{c}_2) + \cdots + T_n(\boldsymbol{c}_n)$$

when $\boldsymbol{c}_1 \in C_1$, $\boldsymbol{c}_2 \in C_2$, ..., $\boldsymbol{c}_n \in C_n$. Similarly, when $A_1, A_2, \ldots, A_n$ are square matrices, $A_1 \oplus A_2 \oplus \cdots \oplus A_n$ is defined as the block matrix where the blocks along the diagonal are $A_1, A_2, \ldots, A_n$, and where all other blocks are 0. Show that, if $\mathcal{B}_i$ is a basis for C_i then

$$[T_1 \oplus T_2 \oplus \ldots \oplus T_n]_{(\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n)} = [T_1]_{\mathcal{B}_1} \oplus [T_2]_{\mathcal{B}_2} \oplus \cdots \oplus [T_n]_{\mathcal{B}_n},$$

Here two new concepts are used: a direct sum of matrices, and a direct sum of linear transformations.

Exercise 5.13: Eigenvectors of direct sums

Assume that T_1 and T_2 are matrices, and that the eigenvalues of T_1 are equal to those of T_2. What are the eigenvalues of $T_1 \oplus T_2$? Can you express the eigenvectors of $T_1 \oplus T_2$ in terms of those of T_1 and T_2?

Exercise 5.14: Invertibility of direct sums

Assume that A and B are square matrices which are invertible. Show that $A \oplus B$ is invertible, and that $(A \oplus B)^{-1} = A^{-1} \oplus B^{-1}$.

Exercise 5.15: Multiplying direct sums

Let A, B, C, D be square matrices of the same dimensions. Show that $(A \oplus B)(C \oplus D) = (AC) \oplus (BD)$.

Exercise 5.16: Finding N

Assume that you run an m-level DWT on a vector of length r. What value of N does this correspond to? Note that an m-level DWT performs a change of coordinates from $\boldsymbol{\phi}_m$ to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0, \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_{m-2}, \boldsymbol{\psi}_{m-1})$.

Exercise 5.17: Different DWTs for similar vectors

In Figure 5.13 we have plotted the DWT's of two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. In both vectors we have 16 ones followed by 16 zeros, and this pattern repeats cyclically so that the length of both vectors is 256. The only difference is that the second vector is obtained by delaying the first vector with one element.

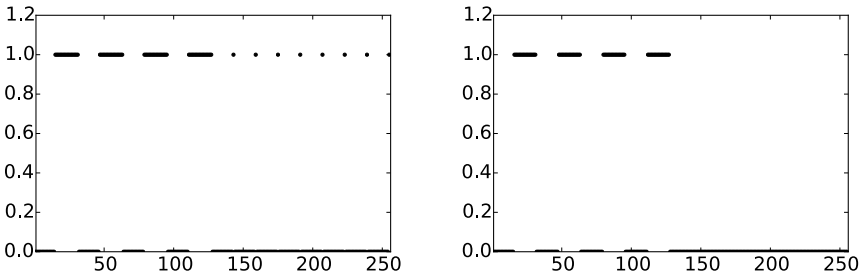

Figure 5.13: 2 vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ which seem equal, but where the DWT's are very different.

You see that the two DWT's are very different: For the first vector we see that there is much detail present (the second part of the plot), while for the second vector there is no detail present. Attempt to explain why this is the case. Based on your answer, also attempt to explain what can happen if you change the point of discontinuity for the piecewise constant function in Figure 5.20(a) to something else.

Exercise 5.18: Plotting the DWT on a sound

Run a 2-level DWT on the first 2^{17} sound samples of the audio file `castanets.wav`, and plot the values of the resulting DWT-coefficients. Compare the values of the coefficients from V_0 with those from W_0 and W_1.

Exercise 5.19: Zeroing out DWT coefficients

In this exercise we will experiment with applying an m-level DWT to a sound file.

a) Write a function `playDWT` which takes m, a DWT kernel `f`, an IDWT kernel `invf`, and a variable `lowres` as input, and

- reads the audio file `castanets.wav`,
- performs an m-level DWT to the first 2^{17} sound samples of $\boldsymbol{x}$ using the function `DWTImpl` with DWT kernel `f`,
- sets all wavelet coefficients representing detail to zero if `lowres` is true (i.e. keep only the coordinates from $\boldsymbol{\phi}_0$ in the basis $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0, \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_{m-2}, \boldsymbol{\psi}_{m-1})$),
- sets all low-resolution coefficients to zero if `lowres` is false (i.e. zero out the coordinates from $\boldsymbol{\phi}_0$ and keep the others),
- performs an IDWT on the resulting coefficients using the function `IDWTImpl` with IDWT kernel `invf`,
- plays the resulting sound.

b) Do the sound samples returned by `playDWT` lie in $[-1, 1]$?

c) Run the function `playDWT` with `DWTKernelHaar` and `IDWTKernelHaar` as inputs, and for different values of m, with 'lowres' set to true (i.e. with the low-resolution approximation chosen). For which m can you hear that the sound gets degraded? How does it get degraded? Compare with what you heard through the function `playDFT` in Example 2.27, where you performed a DFT on the sound sample instead, and set some of the DFT coefficients to zero.

d) Repeat the listening experiment from c., but this time with `lowres` set to false (i.e. keep only the detail from W_0, $W_1, \ldots$. What kind of sound do you hear? Can you recognize the original sound in what you hear?

Exercise 5.20: Construct a sound

Attempt to construct a (nonzero) sound where the function `playDWT` form the previous exercise does not change the sound for $m = 1, 2$.

Exercise 5.21: Exact computation of wavelet coefficients 1

Compute the wavelet detail coefficients analytically for the functions in Example 5.20, i.e. compute the quantities $w_{m,n} = \int_0^N f(t)\psi_{m,n}(t)dt$ similarly to how this was done in Example 5.21.

Exercise 5.22: Exact compution of wavelet coefficients 2

Compute the wavelet detail coefficients analytically for the functions $f(t) = \left(\frac{t}{N}\right)^k$, i.e. compute the quantities $w_{m,n} = \int_0^N \left(\frac{t}{N}\right)^k \psi_{m,n}(t)dt$ similarly to how this was done in Example 5.21. How do these compare with the coefficients from the Exercise 5.21?

Exercise 5.23: Computing the DWT of a simple vector

Suppose that we have the vector $\boldsymbol{x}$ with length $2^{10} = 1024$, defined by $x_n = 1$ for n even, $x_n = -1$ for n odd. What will be the result if you run a 10-level DWT on $\boldsymbol{x}$? Use the function `DWTImpl` to verify what you have found.

Hint. We defined ψ by $\psi(t) = (\phi_{1,0}(t) - \phi_{1,1}(t))/\sqrt{2}$. From this connection it follows that $\psi_{9,n} = (\phi_{10,2n} - \phi_{10,2n+1})/\sqrt{2}$, and thus $\phi_{10,2n} - \phi_{10,2n+1} = \sqrt{2}\psi_{9,n}$. Try to couple this identity with the alternating sign you see in $\boldsymbol{x}$.

Exercise 5.24: The Haar wavelet when N is odd

Use the results from Exercise 5.8 to rewrite the implementations `DWTKernelHaar` and `IDWTKernelHaar` so that they also work in the case when N is odd.

Exercise 5.25: in-place DWT

Show that the coordinates in $\boldsymbol{\phi}_m$ after an in-place m-level DWT end up at indices $k2^m$, $k = 0, 1, 2, \ldots$. Show similarly that the coordinates in $\boldsymbol{\psi}_m$ after an in-place m-level DWT end up at indices $2^{m-1} + k2^m$, $k = 0, 1, 2, \ldots$. Find these indices in the code for the function `reorganize_coefficients`.

5.4 A wavelet based on piecewise linear functions

Unfortutately, piecewise constant functions are too simple to provide good approximations. In this section we are going to extend the construction of wavelets to piecewise linear functions. The advantage is that piecewise linear functions are better for approximating smooth functions and data than piecewise constants, which should translate into smaller components (errors) in the detail spaces in many practical situations. As an example, this would be useful if we are interested in compression. In this new setting it turns out that we loose the

orthonormality we had for the Haar wavelet. On the other hand, we will see that the new scaling functions and mother wavelets are symmetric functions. We will later see that this implies that the corresponding DWT and IDWT have simple implementations with higher precision. Our experience from deriving Haar wavelets will guide us in the construction of piecewise linear wavelets. The first task is to define the new resolution spaces.

Definition 5.22. *Resolution spaces of piecewise linear functions.*

The space V_m is the subspace of continuous functions on $\mathbb{R}$ which are periodic with period N, and linear on each subinterval of the form $[n2^{-m}, (n+1)2^{-m})$.

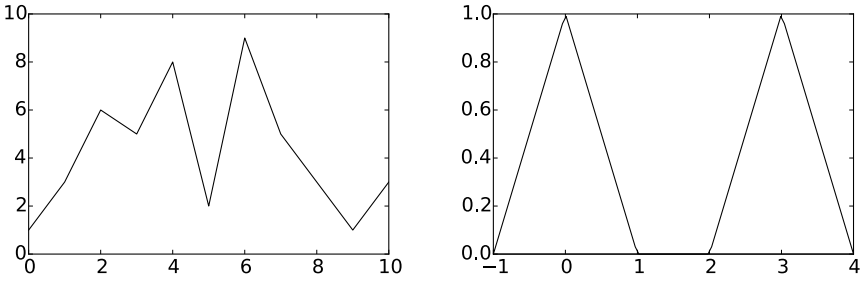

Figure 5.14: A piecewise linear function and the two functions $\phi(t)$ and $\phi(t-3)$.

Any $f \in V_m$ is uniquely determined by its values in the points $\{2_{-m}n\}_{n=0}^{2^m N-1}$. The linear mapping which sends f to these samples is thus an isomorphism from V_m onto R^{N2^m}, so that the dimension of V_m is $\mathbb{N}2^m$. The lft plot in Figure 5.14 shows an example of a piecewise linear function in V_0 on the interval $[0, 10]$. We note that a piecewise linear function in V_0 is completely determined by its value at the integers, so the functions that are 1 at one integer and 0 at all others are particularly simple and therefore interesting, see the right plot in Figure 5.14. These simple functions are all translates of each other and can therefore be built from one scaling function, as is required for a multiresolution analysis.

Lemma 5.23. *The function ϕ.*

Let the function ϕ be defined by

$$\phi(t) = \begin{cases} 1 - |t|, & \text{if } -1 \le t \le 1; \\ 0, & \text{otherwise;} \end{cases} \tag{5.26}$$

and for any $m \ge 0$ set

$$\phi_{m,n}(t) = 2^{m/2}\phi(2^m t - n) \quad \text{for } n = 0, 1, \ldots, 2^m N - 1,$$

and $\boldsymbol{\phi}_m = \{\phi_{m,n}\}_{n=0}^{2^m N-1}$. $\boldsymbol{\phi}_m$ is a basis for V_m, and $\phi_{0,n}(t)$ is the function in V_0 with smallest support that is nonzero at $t = n$.

Proof. It is clear that $\phi_{m,n} \in V_m$, and

$$\phi_{m,n'}(n2^{-m}) = 2^{m/2}\phi(2^m(2^{-m}n) - n') = 2^{m/2}\phi(n - n').$$

Since ϕ is zero at all nonzero integers, and $\phi(0) = 1$, we see that $\phi_{m,n'}(n2^{-m}) = 2^{m/2}$ when $n' = n$, and 0 if $n' \neq n$. Let $L_m : V_m \to R^{N2^m}$ be the isomorphism mentioned above which sends $f \in V_m$ to the samples in the points $\{2_{-m}n\}_{n=0}^{2^m N-1}$. Our calculation shows that $L_m(\phi_{m,n}) = 2^{m/2}e_n$. Since L_m is an isomorphism it follows that $\boldsymbol{\phi}_m = \{\phi_{m,n}\}_{n=0}^{2^m N-1}$ is a basis for V_m.

Suppose that the function $g \in V_0$ has smaller support than $\phi_{0,n}$, but is nonzero at $t = n$. We must have that $L_0(g) = c\boldsymbol{e}_n$ for some c, since g is zero on the integers different from n. But then g is a multiple of $\phi_{0,n}$, so that it is the function in V_0 with smallest support that is nonzero at $t = n$. □

The function ϕ and its translates and dilates are often referred to as hat functions for obvious reasons. Note that the new function ϕ is nonzero for small negative x-values, contrary to the ϕ we defined in Chapter 5. If we plotted the function on $[0, N)$, we would see the nonzero parts at the beginning and end of this interval, due to the period N, but we will mostly plot on an interval around zero, since such an interval captures the entire support of the function. Also for the piecewise linear wavelet the coordinates of a basis function is given by the samples:

Lemma 5.24. *Writing in terms of the samples.*
A function $f \in V_m$ may be written as

$$f(t) = \sum_{n=0}^{2^m N-1} f(n/2^m)2^{-m/2}\phi_{m,n}(t). \tag{5.27}$$

An essential property also here is that the spaces are nested.

Lemma 5.25. *Resolution spaces are nested.*
The piecewise linear resolution spaces are nested,

$$V_0 \subset V_1 \subset \cdots \subset V_m \subset \cdots .$$

Proof. We only need to prove that $V_0 \subset V_1$ since the other inclusions are similar. But this is immediate since any function in V_0 is continuous, and linear on any subinterval in the form $[n/2, (n + 1)/2)$. □

In the piecewise constant case, we saw in Lemma 5.5 that the scaling functions were automatically orthogonal since their supports did not overlap. This is not the case in the linear case, but we could orthogonalise the basis $\boldsymbol{\phi}_m$ with the Gram-Schmidt process from linear algebra. The disadvantage is that we lose the nice local behaviour of the scaling functions and end up with basis functions that are nonzero over all of $[0, N]$. And for most applications, orthogonality is not essential; we just need a basis. The next step in the derivation of wavelets is to find formulas that let us express a function given in the basis $\boldsymbol{\phi}_0$ for V_0 in terms of the basis $\boldsymbol{\phi}_1$ for V_1.

Lemma 5.26. *The two-scale equation.*
The functions $\phi_{0,n}$ satisfy the relation

$$\phi_{0,n} = \frac{1}{\sqrt{2}}\left(\frac{1}{2}\phi_{1,2n-1} + \phi_{1,2n} + \frac{1}{2}\phi_{1,2n+1}\right). \tag{5.28}$$

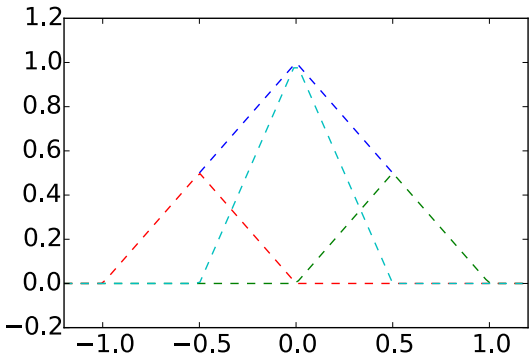

Figure 5.15: How $\phi(t)$ can be decomposed as a linear combination of $\phi_{1,-1}$, $\phi_{1,0}$, and $\phi_{1,1}$.

Proof. Since $\phi_{0,n}$ is in V_0 it may be expressed in the basis $\boldsymbol{\phi}_1$ with formula (5.27),

$$\phi_{0,n}(t) = 2^{-1/2}\sum_{k=0}^{2N-1}\phi_{0,n}(k/2)\phi_{1,k}(t).$$

The relation (5.28) now follows since

$$\phi_{0,n}\big((2n-1)/2\big) = \phi_{0,n}\big((2n+1)/2\big) = 1/2, \quad \phi_{0,n}(2n/2) = 1,$$

and $\phi_{0,n}(k/2) = 0$ for all other values of k. □

The relationship given by Equation (5.28) is shown in Figure 5.15.

5.4.1 Detail spaces and wavelets

The next step in our derivation of wavelets for piecewise linear functions is the definition of the detail spaces. We need to determine a space W_0 that is linearly independent from V_0, and so that $V_1 = V_0 \oplus W_0$. In the case of piecewise constant functions we started with a function g_1 in V_1, computed the least squares approximation g_0 in V_0, and then defined the error function $e_0 = g_1 - g_0$, with $e_0 \in W_0$ and W_0 as the orthogonal complement of V_0 in V_1.

It turns out that this strategy is less appealing in the case of piecewise linear functions. The reason is that the functions $\phi_{0,n}$ are not orthogonal anymore (see Exercise 5.26). Due to this we have no simple, orthogonal basis for the set of piecewise linear functions, so that the orthogonal decomposition theorem fails to give us the projection onto V_0 in a simple way. It is therefore no reason to use the orthogonal complement of V_0 in V_1 as our error space, since it is hard to write a piecewise linear function as a sum of two other piecewise linear functions which are orthogonal. Instead of using projections to find low-resolution approximations, and orthogonal complements to find error functions, we will attempt the following simple approximation method:

Definition 5.27. *Alternative projection.*
Let g_1 be a function in V_1 given by

$$g_1 = \sum_{n=0}^{2N-1} c_{1,n}\phi_{1,n}. \tag{5.29}$$

The approximation $g_0 = P(g_1)$ in V_0 is defined as the unique function in V_0 which has the same values as g_1 at the integers, i.e.

$$g_0(n) = g_1(n), \quad n = 0, 1, \ldots, N-1. \tag{5.30}$$

It is easy to show that $P(g_1)$ actually is different from the projection of g_1 onto V_0: If $g_1 = \phi_{1,1}$, then g_1 is zero at the integers, and then clearly $P(g_1) = 0$. But in Exercise 5.27 you will be asked to compute the projection onto V_0 using different means than the orthogonal decomposition theorem, and the result will be seen to be nonzero. It is also very easy to see that the coordinates of g_0 in $\boldsymbol{\phi}_0$ can be obtained by dropping every second coordinate of g_0 in $\boldsymbol{\phi}_1$. To be more precise, the following holds:

Lemma 5.28. *Expression for the alternative projection.*
We have that

$$P(\phi_{1,n}) = \begin{cases} \sqrt{2}\phi_{0,n/2}, & \text{if } n \text{ is an even integer;} \\ 0, & \text{otherwise.} \end{cases}$$

Once this approximation method is determined, it is straightforward to determine the detail space as the space of error functions.

Lemma 5.29. *Resolution spaces.*
Define

$$W_0 = \{f \in V_1 \mid f(n) = 0, \quad \text{for } n = 0, 1, \ldots, N-1\},$$

and

$$\psi(t) = \frac{1}{\sqrt{2}}\phi_{1,1}(t) \qquad \psi_{m,n}(t) = 2^{m/2}\psi(2^m t - n). \tag{5.31}$$

Suppose that $g_1 \in V_1$ and that $g_0 = P(g_1)$. Then

- the error $e_0 = g_1 - g_0$ lies in W_0,
- $\boldsymbol{\psi}_0 = \{\psi_{0,n}\}_{n=0}^{N-1}$ is a basis for W_0.
- V_0 and W_0 are linearly independent, and $V_1 = V_0 \oplus W_0$.

Proof. Since $g_0(n) = g_1(n)$ for all integers n, $e_0(n) = (g_1 - g_0)(n) = 0$, so that $e_0 \in W_0$. This proves the first statement.

For the second statement, note first that

$$\psi_{0,n}(t) = \psi(t-n) = \frac{1}{\sqrt{2}}\phi_{1,1}(t-n) = \phi(2(t-n)-1) = \phi(2t-(2n+1)) = \frac{1}{\sqrt{2}}\phi_{1,2n+1}(t). \tag{5.32}$$

$\boldsymbol{\psi}_0$ is thus a linearly independent set of dimension N, since it corresponds to a subset of $\boldsymbol{\phi}_1$. Since $\phi_{1,2n+1}$ is nonzero only on $(n, n+1)$, it follows that all of $\boldsymbol{\psi}_0$ lies in W_0. Clearly then $\boldsymbol{\psi}_0$ is also a basis for W_0, since W_0 also has dimension N (its image under L_1 consists of points where every second component is zero).

Consider finally a linear combination from $\boldsymbol{\phi}_0$ and $\boldsymbol{\psi}_0$ which gives zero:

$$\sum_{n=0}^{N-1} a_n \phi_{0,n} + \sum_{n=0}^{N-1} b_n \psi_{0,n} = 0.$$

If we evaluate this at $t = k$, we see that $\psi_{0,n}(k) = 0$, $\phi_{0,n}(k) = 0$ when $n \neq k$, and $\phi_{0,k}(k) = 1$. When we evaluate at k we thus get a_k, which must be zero. If we then evaluate at $t = k + 1/2$ we get in a similar way that all $b_n = 0$, and it follows that V_0 and W_0 are linearly independent. That $V_1 = V_0 \oplus W_0$ follows from the fact that V_1 has dimension $2N$, and V_0 and W_0 both have dimension N. □

We can define W_m in a similar way for $m > 0$, and generalize the lemma to W_m. We can thus state the following analog to Theorem 5.16 for writing $g_m \in V_m$ as a sum of a low-resolution approximation $g_{m-1} \in V_{m-1}$, and a detail/error component $e_{m-1} \in W_{m-1}$.

Theorem 5.30. *Decomposing V_m.*

The space V_m can be decomposed as the direct sum $V_m = V_{m-1} \oplus W_{m-1}$ where

$$W_{m-1} = \{f \in V_m \mid f(n/2^{m-1}) = 0, \quad \text{for } n = 0, 1, \ldots, 2^{m-1}N - 1\}.$$

W_m has the base $\boldsymbol{\psi}_m = \{\psi_{m,n}\}_{n=0}^{2^m N-1}$, and V_m has the two bases

$$\boldsymbol{\phi}_m = \{\phi_{m,n}\}_{n=0}^{2^m N-1}, \text{ and } (\boldsymbol{\phi}_{m-1}, \boldsymbol{\psi}_{m-1}) = \big(\{\phi_{m-1,n}\}_{n=0}^{2^{m-1}N-1}, \{\psi_{m-1,n}\}_{n=0}^{2^{m-1}N-1}\big).$$

With this result we can define the DWT and the IDWT with their stages as before, but the matrices thesemselves are now different. For the IDWT (i.e. $P_{\boldsymbol{\phi}_1 \leftarrow (\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)}$), the columns in the matrix can be found from equations (5.28) and (5.32), i.e.

$$\begin{aligned} \phi_{0,n} &= \frac{1}{\sqrt{2}} \left(\frac{1}{2}\phi_{1,2n-1} + \phi_{1,2n} + \frac{1}{2}\phi_{1,2n+1} \right) \\ \psi_{0,n} &= \frac{1}{\sqrt{2}}\phi_{1,2n+1}. \end{aligned} \tag{5.33}$$

For the DWT we can find the columns in the matrix by rewriting these equations to

$$\begin{aligned} \frac{1}{\sqrt{2}}\phi_{1,2n} &= \phi_{0,n} - \frac{1}{2\sqrt{2}}\phi_{1,2n-1} - \frac{1}{2\sqrt{2}}\phi_{1,2n+1} \\ \frac{1}{\sqrt{2}}\phi_{1,2n+1} &= \psi_{0,n}, \end{aligned}$$

so that

$$\phi_{1,2n} = \sqrt{2}\phi_{0,n} - \frac{1}{2}\phi_{1,2n-1} - \frac{1}{2}\phi_{1,2n+1} = -\frac{\sqrt{2}}{2}\psi_{0,n-1} + \sqrt{2}\phi_{0,n} - \frac{\sqrt{2}}{2}\psi_{0,n} \tag{5.34}$$

$$\phi_{1,2n+1} = \sqrt{2}\psi_{0,n}. \tag{5.35}$$

Example 5.31. *DWT on sound.*

Later we will write algorithms which performs the DWT/IDWT for the piecewise linear wavelet, similarly to how we implemented the Haar wavelet transformation in the previous chapter. This gives us new kernel transformations, which we will call `DWTKernelpwl0`, `IDWTKernelpwl0` (The 0 stands for 0 vanishing moments. We defined vanishing moments after Observation 5.14. We will have more to say about vanishing moments later). Using these new kernels, let us plot the detail/error in the test audio file `castanets.wav` for different resolutions, as we did in Example 5.19. The result is shown in Figure 5.16. When comparing with Figure 5.11 we see much of the same, but it seems here that the error is bigger than before. In the next section we will try to explain why this is the case, and construct another wavelet based on piecewise linear functions which remedies this.

Example 5.32. *DWT on the samples of a mathematical function.*

Let us also repeat Exercise 5.20, where we plotted the detail/error at different resolutions, for the samples of a mathematical function. Figure 5.17 shows the new plot.

With the square wave we see now that there is an error. The reason is that a piecewise constant function can not be represented exactly by piecewise linear

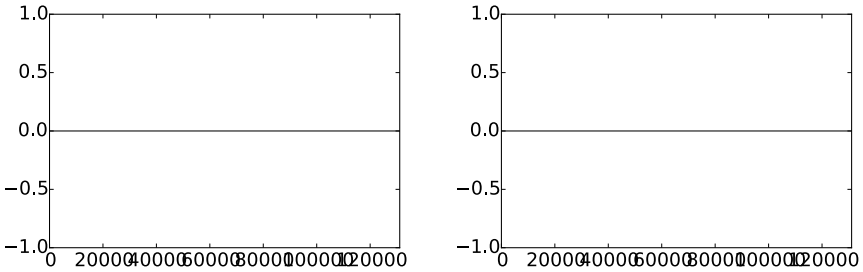

Figure 5.16: The error (i.e. the contribution from $W_0 \oplus W_1 \oplus \cdots \oplus W_{m-1}$) in the sound file `castanets.wav`, for $m = 1$ and $m = 2$, respectively.

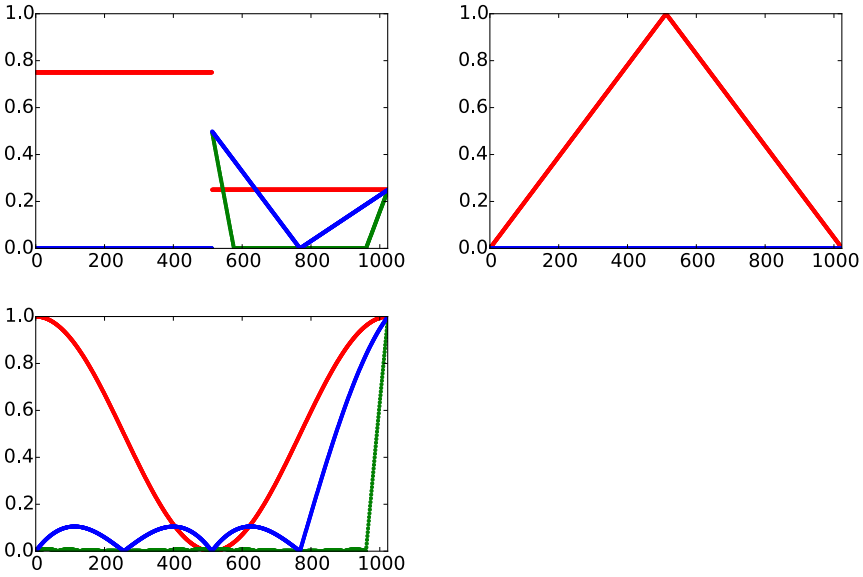

Figure 5.17: The error (i.e. the contribution from $W_0 \oplus W_1 \oplus \cdots \oplus W_{m-1}$) for $N = 1025$ when f is a square wave, the linear function $f(t) = 1 - 2|1/2 - t/N|$, and the trigonometric function $f(t) = 1/2 + \cos(2\pi t/N)/2$, respectivey. The detail is indicated for $m = 6$ and $m = 8$.

functions, due to discontinuity. For the second function we see that there is no error. The reason is that this function is piecewise linear, so there is no error when we represent the function from the space V_0. With the third function, hoewever, we see an error.

What you should have learned in this section.

- Definition of scaling function, mother wavelet, resolution spaces, and detail spaces for the wavelet of piecewise linear functions.

Exercise 5.26: The sample values are coordinates

Show that, for $f \in V_0$ we have that $[f]_{\boldsymbol{\phi}_0} = (f(0), f(1), \ldots, f(N-1))$. This generalizes the result for piecewise constant functions.

Exercise 5.27: Computing projections

In this exercise we will show how the projection of $\phi_{1,1}$ onto V_0 can be computed. We will see from this that it is nonzero, and that its support is the entire $[0, N]$. Let $f = \text{proj}_{V_0}\phi_{1,1}$, and let $x_n = f(n)$ for $0 \leq n < N$. This means that, on $(n, n+1)$, $f(t) = x_n + (x_{n+1} - x_n)(t-n)$.

a) Show that $\int_n^{n+1} f(t)^2 dt = (x_n^2 + x_n x_{n+1} + x_{n+1}^2)/3$.

b) Show that

$$\int_0^{1/2} (x_0 + (x_1 - x_0)t)\phi_{1,1}(t)dt = 2\sqrt{2}\left(\frac{1}{12}x_0 + \frac{1}{24}x_1\right)$$
$$\int_{1/2}^{1} (x_0 + (x_1 - x_0)t)\phi_{1,1}(t)dt = 2\sqrt{2}\left(\frac{1}{24}x_0 + \frac{1}{12}x_1\right).$$

c) Use the fact that

$$\int_0^N (\phi_{1,1}(t) - \sum_{n=0}^{N-1} x_n \phi_{0,n}(t))^2 dt$$
$$= \int_0^1 \phi_{1,1}(t)^2 dt - 2\int_0^{1/2} (x_0 + (x_1 - x_0)t)\phi_{1,1}(t)dt - 2\int_{1/2}^1 (x_0 + (x_1 - x_0)t)\phi_{1,1}(t)dt$$
$$+ \sum_{n=0}^{N-1} \int_n^{n+1} (x_n + (x_{n-1} - x_n)t)^2 dt$$

and a) and b) to find an expression for $\|\phi_{1,1}(t) - \sum_{n=0}^{N-1} x_n \phi_{0,n}(t)\|^2$.

d) To find the minimum least squares error, we can set the gradient of the expression in c. to zero, and thus find the expression for the projection of $\phi_{1,1}$ onto V_0. Show that the values $\{x_n\}_{n=0}^{N-1}$ can be found by solving the equation $S\boldsymbol{x} = \boldsymbol{b}$, where $S = \frac{1}{3}\{1, \underline{4}, 1\}$ is an $N \times N$ symmetric filter, and $\boldsymbol{b}$ is the vector with components $b_0 = b_1 = \sqrt{2}/2$, and $b_k = 0$ for $k \geq 2$.

e) Solve the system in d. for some values of N to verify that the projection of $\phi_{1,1}$ onto V_0 is nonzero, and that its support covers the entire $[0, N]$.

Exercise 5.28: Non-orthogonality for the piecewise linear wavelet

Show that

$$\langle \phi_{0,n}, \phi_{0,n} \rangle = \frac{2}{3} \qquad \langle \phi_{0,n}, \phi_{0,n\pm 1} \rangle = \frac{1}{6} \qquad \langle \phi_{0,n}, \phi_{0,n\pm k} \rangle = 0 \text{ for } k > 1.$$

As a consequence, the $\{\phi_{0,n}\}_n$ are neither orthogonal, nor have norm 1.

Exercise 5.29: Wavelets based on polynomials

The convolution of two functions defined on $(-\infty, \infty)$ is defined by

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt.$$

Show that we can obtain the piecewise linear ϕ we have defined as $\phi = \chi_{[-1/2,1/2)} * \chi_{[-1/2,1/2)}$ (recall that $\chi_{[-1/2,1/2)}$ is the function which is 1 on $[-1/2, 1/2)$ and 0 elsewhere). This gives us a nice connection between the piecewise constant scaling function (which is similar to $\chi_{[-1/2,1/2)}$) and the piecewise linear scaling function in terms of convolution.

5.5 Alternative wavelet based on piecewise linear functions

For the scaling function used for piecewise linear functions, $\{\phi(t-n)\}_{0 \le n < N}$ were not orthogonal anymore, contrary to the case for piecewise constant functions. We were still able to construct what we could call resolution spaces and detail spaces. We also mentioned that having many vanishing moments is desirable for a mother wavelet, and that the scaling function used for piecewise constant functions had one vanishing moment. It is easily checked, however, that the mother wavelet we now introduced for piecewise linear functions (i.e. $\psi(t) = \frac{1}{\sqrt{2}}\phi_{1,1}(t)$) has no vanishing moments. Therefore, this is not a very good choice of mother wavelet. We will attempt the following adjustment strategy to construct an alternative mother wavelet $\hat{\psi}$ which has two vanishing moments, i.e. one more than the Haar wavelet.

Idea 5.33. *Adjusting the wavelet construction.*
Adjust the wavelet construction in Theorem 5.30 to

$$\hat{\psi} = \psi - \alpha\phi_{0,0} - \beta\phi_{0,1} \tag{5.36}$$

and choose α, β so that

$$\int_0^N \hat{\psi}(t)\,dt = \int_0^N t\hat{\psi}(t)\,dt = 0, \tag{5.37}$$

and define $\boldsymbol{\psi}_m = \{\hat{\psi}_{m,n}\}_{n=0}^{N2^m-1}$, and W_m as the space spanned by $\boldsymbol{\psi}_m$.

We thus have two free variables α, β in Equation (5.36), to enforce the two conditions in Equation (5.37). In Exercise 5.30 you are taken through the details of solving this as two linear equations in the two unknowns α and β, and this gives the following result:

Lemma 5.34. *The new function ψ.*
The function

$$\hat{\psi}(t) = \psi(t) - \frac{1}{4}\big(\phi_{0,0}(t) + \phi_{0,1}(t)\big) \tag{5.38}$$

satisfies the conditions (5.37).

Using Equation (5.28), which stated that

$$\phi_{0,n} = \frac{1}{\sqrt{2}}\left(\frac{1}{2}\phi_{1,2n-1} + \phi_{1,2n} + \frac{1}{2}\phi_{1,2n+1}\right), \tag{5.39}$$

we get

$$\begin{aligned}
\hat{\psi}_{0,n} &= \psi_{0,n} - \frac{1}{4}\big(\phi_{0,n} + \phi_{0,n+1}\big) \\
&= \frac{1}{\sqrt{2}}\phi_{1,2n+1} - \frac{1}{4}\frac{1}{\sqrt{2}}\left(\frac{1}{2}\phi_{1,2n-1} + \phi_{1,2n} + \frac{1}{2}\phi_{1,2n+1}\right) \\
&- \frac{1}{4}\frac{1}{\sqrt{2}}\left(\frac{1}{2}\phi_{1,2n+1} + \phi_{1,2n+2} + \frac{1}{2}\phi_{1,2n+3}\right) \\
&= \frac{1}{\sqrt{2}}\left(-\frac{1}{8}\phi_{1,2n-1} - \frac{1}{4}\phi_{1,2n} + \frac{3}{4}\phi_{1,2n+1} - \frac{1}{4}\phi_{1,2n+2} - \frac{1}{8}\phi_{1,2n+3}\right)
\end{aligned} \tag{5.40}$$

Note that what we did here is equivalent to finding the coordinates of $\hat{\psi}$ in the basis $\boldsymbol{\phi}_1$: Equation (5.38) says that

$$[\hat{\psi}]_{(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)} = (-1/4, -1/4, 0, \ldots, 0) \oplus (1, 0, \ldots, 0). \tag{5.41}$$

Since the IDWT is the change of coordinates from $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$ to $\boldsymbol{\phi}_1$, we could also have computed $[\hat{\psi}]_{\boldsymbol{\phi}_1}$ by taking the IDWT of $(-1/4, -1/4, 0, \ldots, 0) \oplus (1, 0, \ldots, 0)$. In the next section we will consider more general implementations of the DWT and the IDWT, which we thus can use instead of performing the computation above.

In summary we have

$$\begin{aligned}
\phi_{0,n} &= \frac{1}{\sqrt{2}}(\frac{1}{2}\phi_{1,2n-1} + \phi_{1,2n} + \frac{1}{2}\phi_{1,2n+1}) \\
\hat{\psi}_{0,n} &= \frac{1}{\sqrt{2}}\left(-\frac{1}{8}\phi_{1,2n-1} - \frac{1}{4}\phi_{1,2n} + \frac{3}{4}\phi_{1,2n+1} - \frac{1}{4}\phi_{1,2n+2} - \frac{1}{8}\phi_{1,2n+3}\right),
\end{aligned} \tag{5.42}$$

which gives us the change of coordinate matrix $P_{\boldsymbol{\phi}_1 \leftarrow (\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)}$. The new function $\hat{\psi}$ is plotted in Figure 5.18.

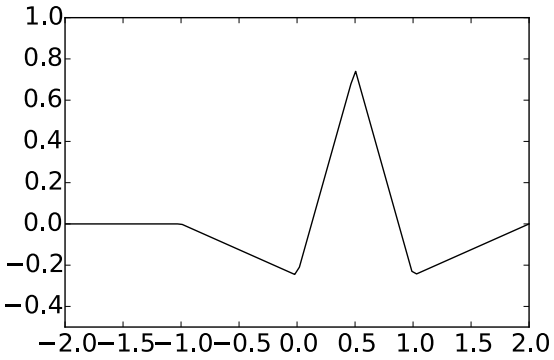

Figure 5.18: The function $\hat{\psi}$ we constructed as an alternative wavelet for piecewise linear functions.

We see that $\hat{\psi}$ has support $(-1, 2)$, and consist of four linear segments glued together. This is in contrast with the old ψ, which was simpler in that it had the shorther support $(0, 1)$, and consisted of only two linear segments glued together. It may therefore seem surprising that $\hat{\psi}$ is better suited for approximating functions than ψ. This is indeed a more complex fact, which may not be deduced by simply looking at plots of the functions.

Example 5.35. *DWT on sound.*

Also in this case we will see later how to write kernel transformations for the alternative piecewise wavelet. We will call these `DWTKernelpwl2` and `IDWTKernelpwl2` (2 stands for 2 vanishing moments). Using these we can plot the detail/error in the test audio file `castanets.wav` for different resolutions for our alternative wavelet, as we did in Example 5.19. The result is shown in Figure 5.19. Again, when comparing with Figure 5.11 we see much of the same. It is difficult to see an improvement from this figure. However, this figure also clearly shows a smaller error than the wavelet of the preceding section. A partial explanation is that the wavelet we now have constructed has two vanishing moments, while the previous one had not.

Example 5.36. *DWT on the samples of a mathematical function.*

Let us also repeat Exercise 5.20 for our alternative wavelet, where we plotted the detail/error at different resolutions, for the samples of a mathematical function. Figure 5.20 shows the new plot.

Again for the square wave there is an error, which seems to be slightly lower than for the previous wavelet. For the second function we see that there is no error, as before. The reason is the same as before, since the function is piecewise

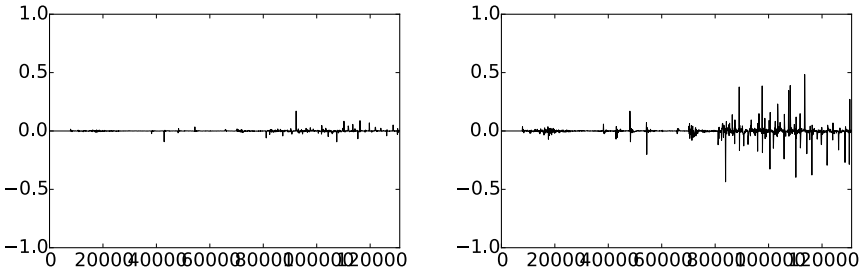

Figure 5.19: The error (i.e. the contribution from $W_0 \oplus W_1 \oplus \cdots \oplus W_{m-1}$) in the sound file `castanets.wav`, for $m = 1$ and $m = 2$, respectively.

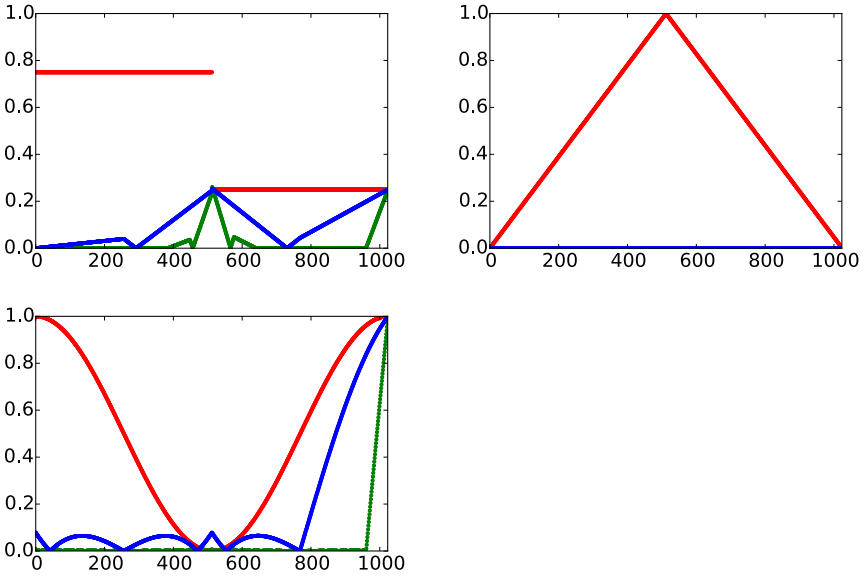

Figure 5.20: The error (i.e. the contribution from $W_0 \oplus W_1 \oplus \cdots \oplus W_{m-1}$) for $N = 1025$ when f is a square wave, the linear function $f(t) = 1 - 2|1/2 - t/N|$, and the trigonometric function $f(t) = 1/2 + \cos(2\pi t/N)/2$, respectivey. The detail is indicated for $m = 6$ and $m = 8$.

linear. With the third function there is an error. The error seems to be slightly lower than for the previous wavelet, which fits well with tha fact that this new wavelet has a bigger number of vanishing moments.

Example 5.37. *Playing sound.*

In Exercise 5.19 we implemented a function `playDWT` which could play the low resolution part in a sound, and we tested this for the Haar wavelet. Let us now also test this for the two piecewise linear wavelets we have constructed,

and the new wavelet kernels we have implemented. Code which plays the low resolution part for all three wavelet kernels can look as follows:

```
playDWT(m, DWTKernelHaar, IDWTKernelHaar, True)
playDWT(m, DWTKernelpwl0, IDWTKernelpwl0, True)
playDWT(m, DWTKernelpwl2, IDWTKernelpwl2, True)
```

The first call to `playDWT` plays the low-resolution part using the Haar wavelet. The code then moves on to the two piecewise linear wavelets. We clearly hear different sounds when we run this code for different m, so that the three wavelets act differently on the sound (if you want, you can here write a `for`-loop around the code, running through different m). Perhaps the alternative piecewise wavelet gives a bit better quality.

What you should have learned in this section.

- How one alters the mother wavelet for piecewise linear functions, in order to add a vanishing moment.

Exercise 5.30: Two vanishing moments

In this exercise we will show that there is a unique function on the form fiven by Equation (5.36) in the compendium which has two vanishing moments.

a) Show that, when $\hat{\psi}$ is defined by Equation (5.36) in the compendium, we have that

$$\hat{\psi}(t) = \begin{cases} -\alpha t - \alpha & \text{for } -1 \leq t < 0 \\ (2+\alpha-\beta)t - \alpha & \text{for } 0 \leq t < 1/2 \\ (\alpha-\beta-2)t - \alpha + 2 & \text{for } 1/2 \leq t < 1 \\ \beta t - 2\beta & \text{for } 1 \leq t < 2 \\ 0 & \text{for all other } t \end{cases}$$

b) Show that

$$\int_0^N \hat{\psi}(t)dt = \frac{1}{2} - \alpha - \beta, \qquad \int_0^N t\hat{\psi}(t)dt = \frac{1}{4} - \beta.$$

c) Explain why there is a unique function on the form given by Equation (5.36) in the compendium which has two vanishing moments, and that this function is given by Equation (5.38) in the compendium.

Exercise 5.31: Implement finding ψ with vanishing moments

In the previous exercise we ended up with a lot of calculations to find α, β in Equation (5.36) in the compendium. Let us try to make a program which does this for us, and which also makes us able to generalize the result.

a) Define

$$a_k = \int_{-1}^{1} t^k(1-|t|)dt, \quad b_k = \int_{0}^{2} t^k(1-|t-1|)dt, \quad e_k = \int_{0}^{1} t^k(1-2|t-1/2|)dt,$$

for $k \geq 0$. Explain why finding α, β so that we have two vanishing moments in Equation (5.36) in the compendium is equivalent to solving the following equation:

$$\begin{pmatrix} a_0 & b_0 \\ a_1 & b_1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} e_0 \\ e_1 \end{pmatrix}$$

Write a program which sets up and solves this system of equations, and use this program to verify the values for α, β we previously have found.

Hint. you can integrate functions in Python with the function `quad` in the package `scipy.integrate`. As an example, the function $\phi(t)$, which is nonzero only on $[-1, 1]$, can be integrated as follows:

```
res, err = quad(lambda t: t**k*(1-abs(t)), -1, 1)
```

b) The procedure where we set up a matrix equation in a) allows for generalization to more vanishing moments. Define

$$\hat{\psi} = \psi_{0,0} - \alpha\phi_{0,0} - \beta\phi_{0,1} - \gamma\phi_{0,-1} - \delta\phi_{0,2}. \tag{5.43}$$

We would like to choose $\alpha, \beta, \gamma, \delta$ so that we have 4 vanishing moments. Define also

$$g_k = \int_{-2}^{0} t^k(1-|t+1|)dt, \qquad d_k = \int_{1}^{3} t^k(1-|t-2|)dt$$

for $k \geq 0$. Show that $\alpha, \beta, \gamma, \delta$ must solve the equation

$$\begin{pmatrix} a_0 & b_0 & g_0 & d_0 \\ a_1 & b_1 & g_1 & d_1 \\ a_2 & b_2 & g_2 & d_2 \\ a_3 & b_3 & g_3 & d_3 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix},$$

and solve this with your computer.

c) Plot the function defined by (5.43) in the compendium, which you found in b.

Hint. If `t` is the vector of t-values, and you write

```
(t >= 0)*(t <= 1)*(1-2*abs(t-0.5))
```

you get the points $\phi_{1,1}(t)$.

d) Explain why the coordinate vector of $\hat{\psi}$ in the basis $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$ is

$$[\hat{\psi}]_{(\boldsymbol{\phi}_0,\boldsymbol{\psi}_0)} = (-\alpha, -\beta, -\delta, 0, \ldots, 0 - \gamma) \oplus (1, 0, \ldots, 0).$$

Hint. You can also compare with Equation (5.41) in the compendium here. The placement of $-\gamma$ may seem a bit strange here, and has to with that $\phi_{0,-1}$ is not one of the basis functions $\{\phi_{0,n}\}_{n=0}^{N-1}$. However, we have that $\phi_{0,-1} = \phi_{0,N-1}$, i.e. $\phi(t+1) = \phi(t-N+1)$, since we always assume that the functions we work with have period N.

e) Sketch a more general procedure than the one you found in b., which can be used to find wavelet bases where we have even more vanishing moments.

Exercise 5.32: ψ for the Haar wavelet with two vanishing moments

Let $\phi(t)$ be the function we used when we defined the Haar-wavelet.

a) Compute $\text{proj}_{V_0}(f(t))$, where $f(t) = t^2$, and where f is defined on $[0, N)$.

b) Find constants α, β so that $\hat{\psi}(t) = \psi(t) - \alpha\phi_{0,0}(t) - \beta\phi_{0,1}(t)$ has two vanishing moments, i.e. so that $\langle\hat{\psi}, 1\rangle = 0$, and $\langle\hat{\psi}, t\rangle = 0$. Plot also the function $\hat{\psi}$.

Hint. Start with computing the integrals $\int \psi(t)dt$, $\int t\psi(t)dt$, $\int \phi_{0,0}(t)dt$, $\int \phi_{0,1}(t)dt$, and $\int t\phi_{0,0}(t)dt$, $\int t\phi_{0,1}(t)dt$.

c) Express ϕ and $\hat{\psi}$ with the help of functions from $\boldsymbol{\phi}_1$, and use this to write down the change of coordinate matrix from $(\boldsymbol{\phi}_0, \hat{\boldsymbol{\psi}}_0)$ to $\boldsymbol{\phi}_1$.

Exercise 5.33: More vanishing moments for the Haar wavelet

It is also possible to add more vanishing moments to the Haar wavelet. Define

$$\hat{\psi} = \psi_{0,0} - a_0\phi_{0,0} - \cdots - a_{k-1}\phi_{0,k-1}.$$

Define also $c_{r,l} = \int_l^{l+1} t^r dt$, and $e_r = \int_0^1 t^r \psi(t)dt$.

a) Show that $\hat{\psi}$ has k vanishing moments if and only if $a_0, \ldots, a_{k-1}$ solves the equation

$$\begin{pmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,k-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,k-1} \\ \vdots & \vdots & \vdots & \vdots \\ c_{k-1,0} & c_{k-1,1} & \cdots & c_{k-1,k-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{pmatrix} = \begin{pmatrix} e_0 \\ e_1 \\ \vdots \\ e_{k-1} \end{pmatrix} \tag{5.44}$$

b) Write a function `vanishingmomshaar` which takes k as input, solves Equation (5.44) in the compendium, and returns the vector $\boldsymbol{a} = (a_0, a_1, \ldots, a_{k-1})$.

Exercise 5.34: Listening experiments

Run the function `playDWT` for different m for the Haar wavelet, the piecewise linear wavelet, and the alternative piecewise linear wavelet, but listen to the detail components $W_0 \oplus W_1 \oplus \cdots \oplus W_{m-1}$ instead. Describe the sounds you hear for different m, and try to explain why the sound seems to get louder when you increase m.

5.6 Multiresolution analysis: A generalization

Let us summarize the properties of the spaces V_m. In both our examples we showed that they were nested, i.e.

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_m \subset \cdots .$$

We also showed that continuous functions could be approximated arbitrarily well from V_m, as long as m was chosen large enough. Moreover, the space V_0 is closed under all translates, at least if we view the functions in V_0 as periodic with period N. In the following we will always identify a function with this periodic extension, just as we did in Fourier analysis. When performing this identification, we also saw that $f(t) \in V_m$ if and only if $g(t) = f(2t) \in V_{m+1}$. We have therefore shown that the scaling functions we have considered fit into the following general framework.

Definition 5.38. *Multiresolution analysis.*

A Multiresolution analysis, or MRA, is a nested sequence of function spaces

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_m \subset \cdots , \tag{5.45}$$

called resolution spaces, so that

Any function can be approximated arbitrarily well from V_m, as long as m is large enough,

$f(t) \in V_0$ if and only if $f(2^m t) \in V_m$,

$f(t) \in V_0$ if and only if $f(t - n) \in V_0$ for all n.

There is a function ϕ, called a scaling function, so that $\boldsymbol{\phi} = \{\phi(t-n)\}_{0\leq n<N}$ is a basis for V_0.

When $\boldsymbol{\phi}$ is an orthonormal basis we say that the MRA is *orthonormal*.

The wavelet of piecewise constant functions was an orthonormal MRA, while the wavelets for piecewise linear functions were not. Although the definition above states that any function can be approximated with MRA's, in practice one needs to restrict to certain functions: Certain pathological functions may be difficult to approximate. In the literature one typically requires that the function is in $L^2(\mathbb{R})$, and also that the scaling function and the spaces V_m are in $L^2(\mathbb{R})$. MRA's are much used, and one can find a wide variety of functions ϕ, not only piecewise constant functions, which give rise to MRA's.

In the examples we have considered we also chose a mother wavelet. The term wavelet is used in very general terms. However, the term mother wavelet is quite concrete, and is what gives rise to the theory of wavelets. This was necessary in order to efficiently decompose the $g_m \in V_m$ into a low resolution approximation $g_{m-1} \in V_{m-1}$, and a detail/error e_{m-1} in a detail space we called W_{m-1}. We have freedom in how we define these detail spaces, as well as how we define a mother wavelet whose translates span the detail space (in general we choose a mother wavelet which simplifies the computation of the decomposition $g_m = g_{m-1} + e_{m-1}$, but we will see later that it also is desirable to choose a ψ with other properties). Once we agree on the detail spaces and the mother wavelet, we can perform a change of coordinates to find detail and low resolution approximations. We thus have the following general recipe.

Idea 5.39. *Recipe for constructing wavelets.*

In order to construct MRA's which are useful for practical purposes, we need to do the following:

- Find a function ϕ which can serve as the scaling function for an MRA,
- Find a function ψ so that $\boldsymbol{\psi} = \{\psi(t-n)\}_{0\leq n<N}$ and $\boldsymbol{\phi} = \{\phi(t-n)\}_{0\leq n<N}$ together form an orthonormal basis for V_1. The function ψ is also called a mother wavelet.

With V_0 the space spanned by $\boldsymbol{\phi} = \{\phi(t-n)\}_{0\leq n<N}$, and W_0 the space spanned by $\boldsymbol{\psi} = \{\psi(t-n)\}_{0\leq n<N}$, ϕ and ψ should be chosen so that we easily can compute the decomposition of $g_1 \in V_1$ into $g_0 + e_0$, where $g_0 \in V_0$ and $e_0 \in W_0$. If we can achieve this, the Discrete Wavelet Transform is defined as the change of coordinates from $\boldsymbol{\phi}_1$ to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$.

More generally, if

$$f(t) = \sum_n c_{m,n}\phi_{m,n} = \sum_n c_{0,n}\phi_{0,n} + \sum_{m'<m,n} w_{m',n}\psi_{m',n},$$

then the m-level DWT is defined by $\mathrm{DWT}(\boldsymbol{c}_m) = (\boldsymbol{c}_0, \boldsymbol{w}_0, \ldots, \boldsymbol{w}_{m-1})$. It is useful to interpret m as frequency, n as time, and $w_{m,n}$ as the contribution

at frequency m and time n. In this sense, wavelets provide a *time-frequency representation* of signals. This is what can make them more useful than Fourier analysis, which only provides frequency representations.

While there are in general many possible choices of detail spaces, in the case of an orthonormal wavelet we saw that it was natural to choose the detail space W_{m-1} as the orthogonal complement of V_{m-1} in V_m, and obtain the mother wavelet by projecting the scaling function onto the detail space. Thus, for orthonormal MRA's, the low-resolution approximation and the detail can be obtained by computing projections, and the least squares approximation of f from V_m can be computed as

$$\text{proj}_{V_m}(f) = \sum_n \langle f, \phi_{m,n}\rangle \phi_{m,n}(t).$$

5.6.1 Working with the samples of f instead of f

In the MRA-setting it helps to think about the continuous-time function $f(t)$ as the model for an image, which is the object under study. f itself may not be in any V_m, however (this corresponds to that detail is present in the image for infinitely many m), and increasing m corresponds to that we also include the detail we see when we zoom in on the image. But how can we obtain useful approximations to f from V_m? In case of an orthonormal MRA we can compute the least squares approximation as above, but we then need to compute the integrals $\langle f, \phi_{m,n}\rangle$, so that all function values are needed. However, as before we have only access to some samples $f(2^{-m}n)$, $0 \le n < 2^m N$. These are called pixel values in the context of images, so that we can only hope to obtain a good approximation to $f^{(m)}$ (and thus f) from the pixel values. The following result explains how we can obtain this.

Theorem 5.40. *Using the samples.*

If f is continuous, and ϕ has compact support, we have that, for all t,

$$f(t) = \lim_{m\to\infty} \sum_{n=0}^{2^m N-1} \frac{2^{-m}}{\int_0^N \phi_{m,0}(t)dt} f(n/2^m)\phi_{m,n}(t).$$

Proof. We have that

$$2^{-m} \sum_{n=0}^{2^m N-1} \phi_{m,n} = \sum_{n=0}^{2^m N-1} 2^{-m}\phi_{m,0}(t - 2^{-m}n).$$

We recognize this as a Riemann sum for the integral $\int_0^N \phi_{m,0}(t)dt$. Therefore,

$$\lim_{m\to\infty} \sum_{n=0}^{2^m N-1} 2^{-m}\phi_{m,n} = \int_0^N \phi_{m,0}(t)dt.$$

Also, finitely many n contribute in this sum since ϕ has compact support. We now get that

$$\begin{aligned}\sum_{n=0}^{2^m N-1} 2^{-m} f(n/2^m)\phi_{m,n}(t) &= \sum_{n \text{ so that } 2^{-m}n \approx t} 2^{-m} f(n/2^m)\phi_{m,n}(t) \\ &\approx \sum_{n \text{ so that } 2^{-m}n \approx t} 2^{-m} f(t)\phi_{m,n}(t) \\ &= f(t) \sum_{n \text{ so that } 2^{-m}n \approx t} 2^{-m}\phi_{m,n}(t) \approx f(t)\int_0^N \phi_{m,0}(t)dt.\end{aligned}$$

where we have used the continuity of f and that

$$\lim_{m\to\infty} \sum_{n=0}^{2^m N-1} 2^{-m}\phi_{m,n} = \int_0^N \phi_{m,0}(t)dt.$$

The result follows. Note that here we have not used the requirement that $\{\phi(t-n)\}_n$ are orthogonal. □

The coordinate vector $\boldsymbol{x} = \left(\frac{2^{-m}}{\int_0^N \phi_{m,0}(t)dt} f(n/2^m)\right)_{n=0}^{2^m N-1}$ in $\boldsymbol{\phi}_m$ is therefore a candidate to an approximation of both f and $f^{(m)}$ from V_m, using only the pixel values. Normally one drops the leading constant $\frac{2^{-m}}{\int_0^N \phi_{m,0}(t)dt}$, so that one simply considers the sample values $f(n/2^m)$ as a coordinate vector in $\boldsymbol{\phi}_m$. This is used as the input to the DWT.

5.6.2 Increasing the precision of the DWT

Even though the samples of f give a good approximation to f as above, the approximation and f are still different, so that we obtain different output from the DWT. In Section 7.1 we will argue that the output from the DWT is equivalent to sampling the output from certain analog filters. We would like the difference in the output from these analog filters to be as small as possible. If the functions ϕ, ψ are symmetric around 0, we will also see that the analog filters are symmetric (a filter is symmetric if and only if the convolution kernel is symmetric around 0), in which case we know that such a high precision implementation is possible using the simple technique of symmetric extension. Let us summarize this as the following idea.

Idea 5.41. *Symmetric wavelets.*

If the functions ϕ, ψ in a wavelet are symmetric around 0, then we can obtain an implementation of the DWT with higher precision when we consider symmetric extensions of the input.

Unfortunately, the piecewise constant scaling function we encountered was not symmetric. However, the piecewise linear scaling function was, and so are also many other interesting scaling functions we will encounter later. For a

symmetric function, denote as before the symmetric extension of the input f with $\breve{f}$. If the input $\boldsymbol{x}$ to the DWT are the samples $(f(n/2^m))_{n=0}^{2^mN-1}$, we create a vector $\breve{\boldsymbol{x}}$ representing the samples of $\breve{f}$. It is clear that this vector should be

$$\breve{\boldsymbol{x}} = \left((f(n/2^m))_{n=0}^{2^mN-1}, \lim_{t \to N_-} f(t), (f((2^mN-n)/2^m))_{n=1}^{2^mN-1} \right).$$

In this vector there is symmetry around entry 2^mN, so that the vector is determined from the 2^mN+1 first elements. Also the boundary is not duplicated, contrary to the previous symmetric extension given by Definition 4.1. We are thus lead to define a symmetric extension in the following way instead:

Definition 5.42. *Symmetric extension of a vector.*

By the *symmetric extension* of $\boldsymbol{x} \in \mathbb{R}^N$, we mean $\breve{\boldsymbol{x}} \in \mathbb{R}^{2N-2}$ defined by

$$\breve{\boldsymbol{x}}_k = \begin{cases} x_k & 0 \leq k < N \\ x_{2N-2-k} & N \leq k < 2N-3 \end{cases} \tag{5.46}$$

With this notation, $N-1$ is the symmetry point in all symmetric extensions. This is illustrated in Figure 5.21

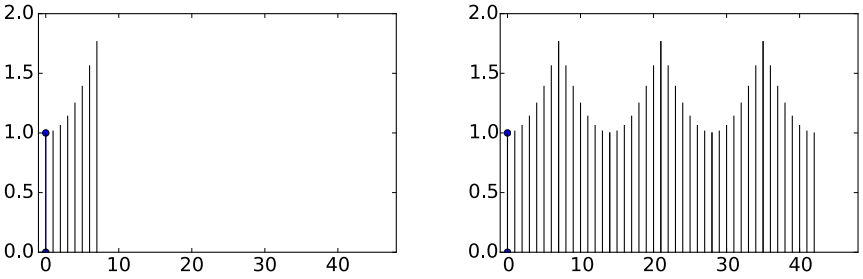

Figure 5.21: A vector and its symmetric extension. Note that the period of the vector is now $2N-2$, while it was $2N$ for the vector shown in Figure 4.1.

From Chapter 4 it follows that symmetric filters preserve the symmetry around $N-1$ when applied to such vectors. We can now define the symmetric restriction S_r as before, with the definition of symmetric extension replaced with the above. We now have the following analog to Theorem 4.9. The proof of this is left as an exercise.

Theorem 5.43. *Expression for S_r.*

With $S = \begin{pmatrix} S_1 \& S_2 \\ S_3 \& S_4 \end{pmatrix} \in \mathbb{R}^{2N-2} \times \mathbb{R}^{2N-2}$ a symmetric filter, with $S_1 \in \mathbb{R}^N \times \mathbb{R}^N$, $S_2 \in \mathbb{R}^N \times \mathbb{R}^{N-2}$, we have that $S_r = S_1 + \begin{pmatrix} 0 & (S_2)^f \& 0 \end{pmatrix}$.

With the Haar wavelet we succeeded in finding a function ψ which could be used in the recipe above. Note, however, that there may be many other ways to

define a function ψ which can be used in the recipe. In the next chapter we will follow the recipe in order to contruct other wavelets, and we will try to express which pairs of function ϕ, ψ are most interesting, and which resolution spaces are most interesting.

What you should have learned in this section.

- Definition of a multiresolution analysis.

Exercise 5.35: Prove expression for S_r

Prove Theorem 5.43. Use the proof of Theorem 4.9 as a guide.

Exercise 5.36: Orthonormal basis for the symmetric extensions

In this exercise we will establish an orthonormal basis for the symmetric extensions, as defined by Definition 5.42. This parallels Theorem 4.6.

a) Explain why, if $\boldsymbol{x} \in \mathbb{R}^{2N-2}$ is a symmetric extension (according to Definition 4.1), then $(\widehat{\boldsymbol{x}})_n = z_n e^{-\pi i n}$, where $\boldsymbol{z}$ is a real vectors which satisfies $z_n = z_{2N-2-n}$

b) Show that

$$\left\{ \boldsymbol{e}_0, \left\{ \frac{1}{\sqrt{2}}(\boldsymbol{e}_i + \boldsymbol{e}_{2N-2-i}) \right\}_{n=1}^{N-2}, \boldsymbol{e}_{N-1} \right\} \tag{5.47}$$

is an orthonormal basis for the vectors on the form $\widehat{\boldsymbol{x}}$ with $\boldsymbol{x} \in \mathbb{R}^{2N-2}$ a symmetric extension.

c) Show that

$$\begin{gathered} \frac{1}{\sqrt{2N-2}} \cos\left(2\pi \frac{0}{2N-2} k\right) \\ \left\{ \frac{1}{\sqrt{N-1}} \cos\left(2\pi \frac{n}{2N-2} k\right) \right\}_{n=1}^{N-2} \\ \frac{1}{\sqrt{2N-2}} \cos\left(2\pi \frac{N-1}{2N-2} k\right) \end{gathered} \tag{5.48}$$

is an orthonormal basis for the symmetric extensions in $\mathbb{R}^{2N-2}$.

d) Assume that S is symmetric. Show that the vectors listed in (5.48) in the compendium are eigenvectors for S_r, when the vectors are viewed as vectors in $\mathbb{R}^N$, and that they are linearly independent. This shows that S_r is diagonalizable.

Exercise 5.37: Diagonalizing S_r

Let us explain how the matrix S_r can be diagonalized, similarly to how we previously diagonalized using the DCT. In Exercise 5.36 we showed that the vectors

$$\left\{\cos\left(2\pi\frac{n}{2N-2}k\right)\right\}_{n=0}^{N-1} \tag{5.49}$$

in $\mathbb{R}^N$ is a basis of eigenvectors for S_r when S is symmetric. S_r itself is not symmetric, however, so that this basis can not possibly be orthogonal (S is symmetric if and only if it is orthogonally digonalizable). However, when the vectors are viewed in $\mathbb{R}^{2N-2}$ we showed in Exercise 5.36c) an orthogonality statement which can be written as

$$\sum_{k=0}^{2N-3}\cos\left(2\pi\frac{n_1}{2N-2}k\right)\cos\left(2\pi\frac{n_2}{2N-2}k\right) = (N-1)\times\begin{cases}2 & \text{if } n_1=n_2\in\{0,N-1\}\\ 1 & \text{if } n_1=n_2\notin\{0,N-1\}\\ 0 & \text{if } n_1\neq n_2\end{cases}. \tag{5.50}$$

a) Show that

$$\begin{aligned}&(N-1)\times\begin{cases}1 & \text{if } n_1=n_2\in\{0,N-1\}\\ \frac{1}{2} & \text{if } n_1=n_2\notin\{0,N-1\}\\ 0 & \text{if } n_1\neq n_2\end{cases}\\ =&\frac{1}{\sqrt{2}}\cos\left(2\pi\frac{n_1}{2N-2}\cdot 0\right)\frac{1}{\sqrt{2}}\cos\left(2\pi\frac{n_2}{2N-2}\cdot 0\right)\\ &+\sum_{k=1}^{N-2}\cos\left(2\pi\frac{n_1}{2N-2}k\right)\cos\left(2\pi\frac{n_2}{2N-2}k\right)\\ &+\frac{1}{\sqrt{2}}\cos\left(2\pi\frac{n_1}{2N-2}(N-1)\right)\frac{1}{\sqrt{2}}\cos\left(2\pi\frac{n_2}{2N-2}(N-1)\right).\end{aligned}$$

Hint. Use that $\cos x=\cos(2\pi-x)$ to pair the summands k and $2N-2-k$.

Now, define the vector $\boldsymbol{d}_n^{(\mathrm{I})}$ as

$$d_{n,N}\left(\frac{1}{\sqrt{2}}\cos\left(2\pi\frac{n}{2N-2}\cdot 0\right),\left\{\cos\left(2\pi\frac{n}{2N-2}k\right)\right\}_{k=1}^{N-2},\frac{1}{\sqrt{2}}\cos\left(2\pi\frac{n}{2N-2}(N-1)\right)\right),$$

and define $d_{0,N}^{(\mathrm{I})}=d_{N-1,N}^{(\mathrm{I})}=1/\sqrt{N-1}$, and $d_{n,N}^{(\mathrm{I})}=\sqrt{2/(N-1)}$ when $n>1$. The orthogonal $N\times N$ matrix where the rows are $\boldsymbol{d}_n^{(\mathrm{I})}$ is called the DCT-I,

and we will denote it by $D_N^{(\mathrm{I})}$. DCT-I is also much used, just as the DCT-II of Chapter 4. The main difference from the previous cosine vectors is that $2N$ has been replaced by $2N-2$.

b) Explain that the vectors $\boldsymbol{d}_n^{(\mathrm{I})}$ are orthonormal, and that the matrix

$$\sqrt{\frac{2}{N-1}}\begin{pmatrix} 1/\sqrt{2} & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1/\sqrt{2} \end{pmatrix}\left(\cos\left(2\pi\tfrac{n}{2N-2}k\right)\right)\begin{pmatrix} 1/\sqrt{2} & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1/\sqrt{2} \end{pmatrix}$$

is orthogonal.

c) Explain from b. that $\left(\cos\left(2\pi\tfrac{n}{2N-2}k\right)\right)^{-1}$ can be written as

$$\frac{2}{N-1}\begin{pmatrix} 1/2 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1/2 \end{pmatrix}\left(\cos\left(2\pi\tfrac{n}{2N-2}k\right)\right)\begin{pmatrix} 1/2 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1/2 \end{pmatrix}$$

With the expression we found in c., S_r can now be diagonalized as

$$\left(\cos\left(2\pi\tfrac{n}{2N-2}k\right)\right) D \left(\cos\left(2\pi\tfrac{n}{2N-2}k\right)\right)^{-1}.$$

5.7 Summary

We started this chapter by motivating the theory of wavelets as a different function approximation scheme, which solved some of the shortcomings of Fourier series. While one approximates functions with trigonometric functions in Fourier theory, with wavelets one instead approximates a function in several stages, where one at each stage attempts to capture information at a given resolution, using a function prototype. This prototype is localized in time, contrary to the Fourier basis functions, and this makes the theory of wavelets suitable for time-frequency representations of signals. We used an example based on Google Earth to illustrate that the wavelet-based scheme can represent an image at different resolutions in a scalable way, so that passing from one resolution to another simply mounts to adding some detail information to the lower resolution version of the image. This also made wavelets useful for compression, since the images at different resolutions can serve as compressed versions of the image.

We defined the simplest wavelet, the Haar wavelet, which is a function approximation scheme based on piecewise constant functions, and deduced its properties. We defined the Discrete Wavelet Transform (DWT) as a change of coordinates corresponding to the function spaces we defined. This transform is the crucial object to study when it comes to more general wavelets also, since it is the object which makes wavelets useful for computation. In the following chapters, we will see that reordering of the source and target bases of the DWT will aid in expressing connections between wavelets and filters, and in constructing optimized implementations of the DWT.

We then defined another wavelet, which corresponded to a function approximation scheme based on piecewise linear functions, instead of piecewise constant functions. There were several differences with the new wavelet when compared to the previous one. First of all, the basis functions were not orthonormal, and we did not attempt to make them orthonormal. The resolution spaces we now defined were not defined in terms of orthogonal bases, and we had some freedom on how we defined the detail spaces, since they are not defined as orthogonal complements anymore. Similarly, we had some freedom on how we define the mother wavelet, and we mentioned that we could define it so that it is more suitable for approximation of functions, by adding what we called vanishing moments.

From these examples of wavelets and their properties we made a generalization to what we called a multiresolution analysis (MRA). In an MRA we construct successively refined spaces of functions that may be used to approximate functions arbitrarily well. We will continue in the next chapter to construct even more general wavelets, within the MRA framework.

The book [21] goes through developments for wavelets in detail. While wavelets have been recognized for quite some time, it was with the important work of Daubechies [8, 9] that they found new arenas in the 80's. Since then they found important applications. The main application we will focus on in later chapters is image processing.

Chapter 6

The filter representation of wavelets

Previously we saw that analog filters restricted to the Fourier spaces gave rise to digital filters. These digital filters sent the samples of the input function to the samples of the output function, and are easily implementable, in contrast to the analog filters. We have also seen that wavelets give rise to analog filters. This leads us to believe that the DWT also can be implemented in terms of digital filters. In this chapter we will prove that this is in fact the case.

There are some differences between the Fourier and wavelet settings, however:

- The DWT is not constructed by looking at the samples of a function, but rather by looking at coordinates in a given basis.
- The function spaces we work in (i.e. V_m) are different from the Fourier spaces.
- The DWT gave rise to two different types of analog filters: The filter defined by Equation (7.16) for obtaining $c_{m,n}$, and the filter defined by Equation (7.17) for obtaining $w_{m,n}$. We want both to correspond to digital filters.

Due to these differences, the way we realize the DWT in terms of filters will be a bit different. Despite the differences, this chapter will make it clear that the output of a DWT can be interpreted as the combined output of two different filters, and each filter will have an interpretation in terms of frequency representations. We will also see that the IDWT has a similar interpretation in terms of filters.

In this chapter we will also see that expressing the DWT in terms of filters will also enable us to define more general transforms, where even more filters are used. It is fruitful to think about each filter as concentrating on a particular frequency range, and that these transforms thus simply splits the input into different frequency bands. Such transforms have important applications to the

processing and compression of sound, and we will show that the much used MP3 standard for compression of sound takes use of such transforms.

6.1 The filters of a wavelet transformation

We will make the connection with digital filters by looking again at the different examples of wavelet bases we have seen: The ones for piecewise constant and piecewise linear functions. For the Haar wavelet we have noted that G and H are block-diagonal with $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$ repeated along the diagonal. For the piecewise linear wavelet, Equation (5.33) gives that the first two columns in $G = P_{\phi_m \leftarrow \mathcal{C}_m}$ take the form

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 1/2 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1/2 & 0 \end{pmatrix}. \tag{6.1}$$

The remaining columns are obtained by shifting this, as in a circulant Toeplitz matrix. Similarly, Equation (5.35) gives that the first two columns in $H = P_{\mathcal{C}_m \leftarrow \phi_m}$ take the form

$$\sqrt{2} \begin{pmatrix} 1 & 0 \\ -1/2 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ -1/2 & 0 \end{pmatrix}. \tag{6.2}$$

Also here, the remaining columns are obtained by shifting this, as in a circulant Toeplitz matrix. For the alternative piecewise linear wavelet, Equation (5.42) give all columns in the change of coordinate matrix $G = P_{\phi_m \leftarrow \mathcal{C}_m}$ also. In particular, the first two columns in this matrix are

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1/4 \\ 1/2 & 3/4 \\ 0 & -1/4 \\ 0 & -1/8 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1/2 & -1/8 \end{pmatrix}. \tag{6.3}$$

The first column is the same as before, since there was no change in the definition of ϕ. The remaining columns are obtained by shifting this, as in a circulant Toeplitz matrix. We will explain later how the change of coordinate matrix $H = P_{\mathcal{C}_m \leftarrow \boldsymbol{\phi}_m}$ also can be computed.

In each case above it turned out that the kernel transformations $G = P_{\boldsymbol{\phi}_m \leftarrow \mathcal{C}_m}$, $H = P_{\mathcal{C}_m \leftarrow \boldsymbol{\phi}_m}$ had a special structure: They were obtained by repeating the first two columns in a circulant way, similarly to how we did in a circulant Toeplitz matrix. The matrices were not exactly circulant Toeplitz matrices, however, since there are two different columns repeating. The change of coordinate matrices occuring in the stages in a DWT are thus not digital filters, but they seem to be related. Let us start by giving these new matrices names:

Definition 6.1. *MRA-matrices.*

An $N \times N$-matrix T, with N even, is called an MRA-matrix if the columns are translates of the first two columns in alternating order, in the same way as the columns of a circulant Toeplitz matrix.

From our previous calculations it is clear that, once ϕ and ψ are given through an MRA, the corresponding change of coordinate matrices will always be MRA-matrices. The MRA-matrices is our connection between filters and wavelets. Let us make the following definition:

Definition 6.2. *H_0 and H_1.*

We denote by H_0 the (unique) filter with the same first row as H, and by H_1 the (unique) filter with the same second row as H. H_0 and H_1 are also called the *DWT filter components*.

Using this definition it is clear that

$$(H\boldsymbol{c}_m)_k = \begin{cases} (H_0\boldsymbol{c}_m)_k & \text{when } k \text{ is even} \\ (H_1\boldsymbol{c}_m)_k & \text{when } k \text{ is odd,} \end{cases}$$

since the left hand side depends only on row k in the matrix H, and this is equal to row k in H_0 (when k is even) or row k in H_1 (when k is odd). This means that $H\boldsymbol{c}_m$ can be computed with the help of H_0 and H_1 as follows:

Theorem 6.3. *DWT expressed in terms of filters.*

Let $\boldsymbol{c}_m$ be the coordinates in $\boldsymbol{\phi}_m$, and let H_0, H_1 be defined as above. Any stage in a DWT can ble implemented in terms of filters as follows:

- Compute $H_0\boldsymbol{c}_m$. The even-indexed entries in the result are the cordinates $\boldsymbol{c}_{m-1}$ in $\boldsymbol{\phi}_{m-1}$.
- Compute $H_1\boldsymbol{c}_m$. The odd-indexed entries in the result are the coordinates $\boldsymbol{w}_{m-1}$ in $\boldsymbol{\psi}_{m-1}$.

This gives an important connection between wavelets and filters: The DWT corresponds to applying two filters, H_0 and H_1, and the result from the DWT is produced by assembling half of the coordinates from each. Keeping only every second coordinate is called *downsampling* (with a factor of two). Had we not performed downsampling, we would have ended up with twice as many coordinates as we started with. Downsampling with a factor of two means that we end up with the same number of samples as we started with. We also say that the output of the two filters is *critically sampled*. Due to the critical sampling, it is inefficient to compute the full application of the filters. We will return to the issue of making efficient implementations of critically sampled filter banks later.

We can now complement Figure 5.9 by giving names to the arrows as follows:

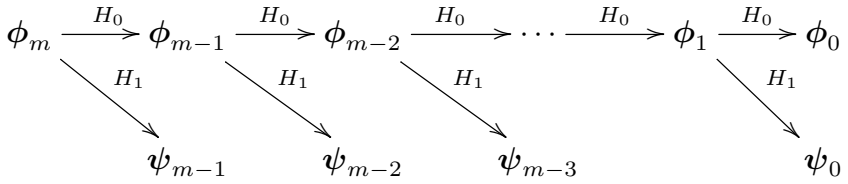

Figure 6.1: Detailed illustration of a wavelet transform.

Let us make a similar anlysis for the IDWT, and let us first make the following definition:

Definition 6.4. *G_0 and G_1.*

We denote by G_0 the (unique) filter with the same first column as G, and by G_1 the (unique) filter with the same second column as G. G_0 and G_1 are also called the *IDWT filter components*.

These filters are uniquely determined, since any filter is uniquely determined from one of its columns. We can now write

$$
\begin{aligned}
\boldsymbol{c}_m = G\begin{pmatrix} c_{m-1,0} \\ w_{m-1,0} \\ c_{m-1,1} \\ w_{m-1,1} \\ \cdots \\ c_{m-1,2^{m-1}N-1} \\ w_{m-1,2^{m-1}N-1} \end{pmatrix} &= G\left(\begin{pmatrix} c_{m-1,0} \\ 0 \\ c_{m-1,1} \\ 0 \\ \cdots \\ c_{m-1,2^{m-1}N-1} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ w_{m-1,0} \\ 0 \\ w_{m-1,1} \\ \cdots \\ 0 \\ w_{m-1,2^{m-1}N-1} \end{pmatrix}\right) \\
&= G\begin{pmatrix} c_{m-1,0} \\ 0 \\ c_{m-1,1} \\ 0 \\ \cdots \\ c_{m-1,2^{m-1}N-1} \\ 0 \end{pmatrix} + G\begin{pmatrix} 0 \\ w_{m-1,0} \\ 0 \\ w_{m-1,1} \\ \cdots \\ 0 \\ w_{m-1,2^{m-1}N-1} \end{pmatrix} \\
&= G_0\begin{pmatrix} c_{m-1,0} \\ 0 \\ c_{m-1,1} \\ 0 \\ \cdots \\ c_{m-1,2^{m-1}N-1} \\ 0 \end{pmatrix} + G_1\begin{pmatrix} 0 \\ w_{m-1,0} \\ 0 \\ w_{m-1,1} \\ \cdots \\ 0 \\ w_{m-1,2^{m-1}N-1} \end{pmatrix}.
\end{aligned}
$$

Here we have split a vector into its even-indexed and odd-indexed elements, which correspond to the coefficients from $\boldsymbol{\phi}_{m-1}$ and $\boldsymbol{\psi}_{m-1}$, respectively. In the last equation, we replaced with G_0, G_1, since the multiplications with G depend only on the even and odd columns in that matrix (due to the zeros inserted), and these columns are equal in G_0, G_1. We can now state the following characterization of the inverse Discrete Wavelet transform:

Theorem 6.5. *IDWT expressed in terms of filters.*

Let G_0, G_1 be defined as above. Any stage in an IDWT can be implemented in terms of filters as follows:

$$
\boldsymbol{c}_m = G_0\begin{pmatrix} c_{m-1,0} \\ 0 \\ c_{m-1,1} \\ 0 \\ \cdots \\ c_{m-1,2^{m-1}N-1} \\ 0 \end{pmatrix} + G_1\begin{pmatrix} 0 \\ w_{m-1,0} \\ 0 \\ w_{m-1,1} \\ \cdots \\ 0 \\ w_{m-1,2^{m-1}N-1} \end{pmatrix}. \tag{6.4}
$$

Making a new vector where zeroes have been inserted in this way is also called *upsampling* (with a factor of two). We can now also complement Figure 5.9 for the IDWT with named arrows. This has bee done in Figure 6.2

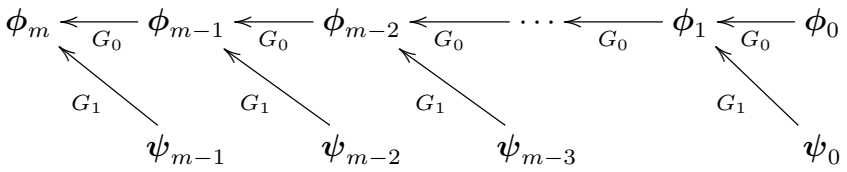

Figure 6.2: Detailed illustration of an IDWT.

Note that the filters G_0, G_1 were defined in terms of the columns of G, while the filters H_0, H_1 were defined in terms of the rows of H. This difference is seen from the computations above to come from that the change of coordinates one way splits the coordinates into two parts, while the inverse change of coordinates performs the opposite. Let us summarize what we have found as follows.

Fact 6.6. *Computing DWT/IDWT through filters.*

The DWT can be computed with the help of two filters H_0, H_1, as explained in Theorem 6.3. Any linear transformation computed from two filters H_0, H_1 in this way is called a *forward filter bank transform*. The IDWT can be computed with the help of two filters G_0, G_1 as explained in Theorem 6.5. Any linear transformation computed from two filters G_0, G_1 in this way is called a *reverse filter bank transform*.

In Chapter 8 we will go through how any forward and reverse filter bank transform can be implemented, once we have the filters H_0, H_1, G_0, and G_1. When we are in a wavelet setting, the filter coefficients in these four filters can be found from the relations between the bases $\boldsymbol{\phi}_1$ and $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$. The filters H_0, H_1, G_0, G_1 can also be constructed from outside a wavelet setting, i.e. that they do not originate from change of coordinate matrices between certain function bases. The important point is that the matrices invert each other, but in a signal processing setting it may also be meaningful to allow for the reverse transform not to invert the forward transform exactly. This corresponds to some loss of information when we attempt to reconstruct the original signal using the reverse transform. A small such loss can, as we will see at the end of this chapter, be acceptable.

That the reverse transform inverts the forward transform means that $GH = I$. If we transpose this expression we get that $H^T G^T = I$. Clearly H^T is a reverse filter bank transform with filters $(H_0)^T, (H_1)^T$, and G^T is a forward filter bank transform with filters $(G_0)^T, (G_1)^T$. Due to their usefulness, these transforms have their own name:

Definition 6.7. *Dual filter bank transforms.*

Assume that H_0, H_1 are the filters of a forward filter bank transform, and that G_0, G_1 are the filters of a reverse filter bank transform. By the *dual transforms* we mean the forward filter bank transform with filters $(G_0)^T, (G_1)^T$, and the reverse filter bank transform with filters $(H_0)^T, (H_1)^T$.

In Section 5.3 we used a parameter `dual` in our call to the DWT amd IDWT kernel functions. This parameter can now be explained as follows:

Fact 6.8. *The `dual`-parameter in DWT kernel functions..*

- If the `dual` parameter is false, the DWT is computed as the forward filter bank transform with filters H_0, H_1, and the IDWT is computed as the reverse filter bank transform with filters G_0, G_1.
- If the `dual` parameter is true, the DWT is computed as the forward filter bank transform with filters $(G_0)^T, (G_1)^T$, and the IDWT is computed as the reverse filter bank transform with filters $(H_0)^T, (H_1)^T$.

Note that, even though the reverse filter bank transform G can be associated with certain function bases, it is not clear if the reverse filter bank transform H^T also can be associated with such bases. We will see in the next chapter that such bases can in many cases be found. We will also denote these bases as *dual bases.*

Note that Figure 6.1 and 6.2 do not indicate the additional downsampling and upsampling steps described in Theorem 6.3 and 6.5. If we indicate downsampling with $\downarrow_2$, and upsampling with $\uparrow_2$, the algorithms given in Theorem 6.3 and 6.5 can be summarized as in Figure 6.3.

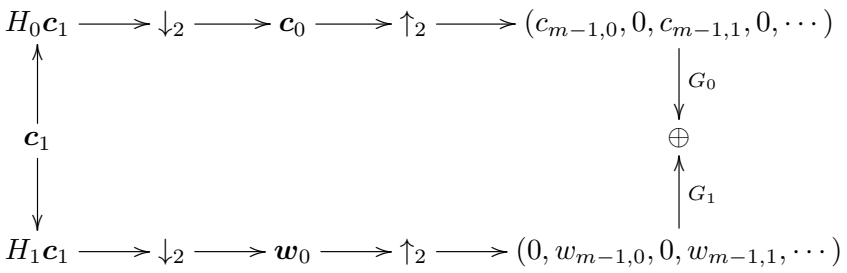

Figure 6.3: Detailed illustration of a DWT.

Here $\oplus$ represents summing the elements which point inwards to the plus sign. In this figure, the left side represents the DWT, the right side the IDWT. In the literature, wavelet transforms are more often illustrated in this way using filters, since it makes alle steps involved in the process more clear. This type of figure also opens for generalization. We will shortly look into this.

There are several reasons why it is smart to express a wavelet transformation in terms of filters. First of all, it enables us to reuse theoretical results from the world of filters in the world of wavelets, and to give useful interpretations of the wavelet transform in terms of frequencies. Secondly, and perhaps most important, it enables us to reuse efficient implementations of filters in order to compute wavelet transformations. A lot of work has been done in order to establish efficient implementations of filters, due to their importance.

In Example 5.19 we argued that the elements in V_{m-1} correspond to frequencies at lower frequencies than those in V_m, since $V_0 = \text{Span}(\{\phi_{0,n}\}_n)$ should be interpreted as content of lower frequency than the $\phi_{1,n}$, with $W_0 = \text{Span}(\{\psi_{0,n}\}_n)$ the remaining high frequency detail. To elaborate more on this, we have that

$$\phi(t) = \sum_{n=0}^{2N-1} (G_0)_{n,0}\phi_{1,n}(t) \tag{6.5}$$

$$\psi(t) = \sum_{n=0}^{2N-1} (G_1)_{n,1}\phi_{1,n}(t)., \tag{6.6}$$

where $(G_k)_{i,j}$ are the entries in the matrix G_k. Similar equations are true for $\phi(t-k), \psi(t-k)$. Due to Equation (6.5), the filter G_0 should have lowpass characteristics, since it extracts the information at lower frequencies. Similarly, G_1 should have highpass characteristics due to Equation (6.6).

Let us verify these lowpass/highpass characteristics of G_0 and G_1 for the wavelets we have considered up to now by plotting their frequency responses. In order to do this we should make a final remark on how these frequency responses can be plotted. For all wavelets we look at the filter coefficients are computed, so that the frequency responses can be easily calculated. However, when we use a wavelet for computation, we applied it by means of a kernel transformation. We will later see that the most efficient such kernel transformations do not apply the filter coefficients directly, but rather a factorization into smaller components (but see Exercise 6.12 on how we can produce kernel transformations which use the filter coefficients directly). So how can we find and plot the frequency response when only the kernel transformation is known? First of all, since the first column of G is identical to the first column of G_0, the first column of G_0 can be obtained by applying the IDWT kernel to the vector $\boldsymbol{e}_0$. We can then use Theorem 3.14 to find the vector frequency response of the filter (i.e. applying an FFT), and then Theorem 3.21 to find the values of the continuous frequency response in the points $2\pi n/N$ for $0 \leq n < N$. The following code can thus be used to plot the frequency response of G_0, when only the IDWT kernel (called `idwtkernel` below) is known.

```
omega = 2*pi*arange(0,N)/float(N)
g0 = concatenate([[1], zeros(N - 1)]) # Creates e_0
idwtkernel(g0, 0, 0)                  # Find the first column of G_0
plot(omega, abs(fft.fft(g0)))
```

A similar procedure can be applied in order to plot the frequency response of G_1 (just replace $\boldsymbol{e}_0$ with $\boldsymbol{e}_1$ in order o exract the second column of G instead). The frequency responses of H_0 and H_1 can be found by considering a dual wavelet transform, since the reverse transform for the dual wavelet has filters $(H_0)^T$ and $(H_1)^T$. In most of the following examples in this book this procedure will be applied to plot all frequency responses. We start with the Haar wavelet.

Example 6.9. *The Haar wavelet.*

For the Haar wavelet we saw that, in G, the matrix

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{6.7}$$

repeated along the diagonal. The filters G_0 and G_1 can be found directly from these columns:

$$G_0 = \{\underline{1/\sqrt{2}}, 1/\sqrt{2}\}$$
$$G_1 = \{1/\sqrt{2}, \underline{-1/\sqrt{2}}\}.$$

We have seen these filters previously: G_0 is a movinge average filter of two elements (up to multiplication with a constant). This is a lowpass filter. G_1 is a bass-reducing filter, which is a highpass filter. Up to a constant, this is also an approximation to the derivative. Since G_1 is constructed from G_0 by adding an alternating sign to the filter coefficients, we know from before that G_1 is the highpass filter corresponding to the lowpass filter G_0, so that the frequency response of the second is given by a shift of frequency with π in the first. The frequency responses are

$$\lambda_{G_0}(\omega) = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} e^{-i\omega} = \sqrt{2} e^{-i\omega/2} \cos(\omega/2)$$
$$\lambda_{G_1}(\omega) = \frac{1}{\sqrt{2}} e^{i\omega} - \frac{1}{\sqrt{2}} = \sqrt{2} i e^{i\omega/2} \sin(\omega/2).$$

The magnitude of these are plotted in Figure 6.4, where the lowpass/highpass characteristics are clearly seen.

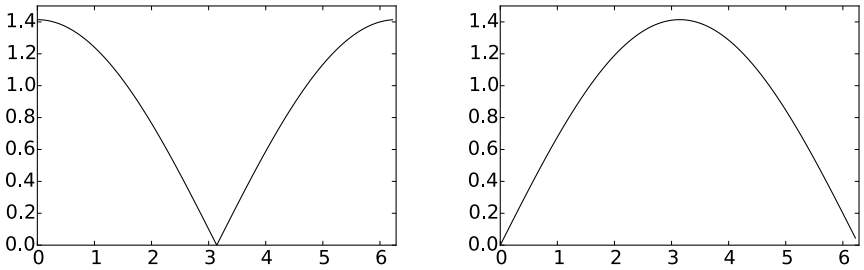

Figure 6.4: The frequency responses $\lambda_{G_0}(\omega)$ (left) and $\lambda_{G_1}(\omega)$ (right) for the Haar wavelet.

By considering the filters where the rows in Equation (6.7), it is clear that

$$H_0 = \{1/\sqrt{2}, \underline{1/\sqrt{2}}\}$$
$$H_1 = \{\underline{-1/\sqrt{2}}, 1/\sqrt{2}\},$$

so that the frequency responses for the DWT have the same lowpass/highpass characteristics.

It turns out that this connection between G_0 and G_1 as lowpass and highpass filters corresponding to each other can be found in all orthonormal wavelets. We will prove this in the next chapter.

Example 6.10. *Wavelet for piecewise linear functions.*

For the wavelet for piecewise linear functions we looked at in the previous section, Equation (6.1) gives that

$$\begin{aligned} G_0 &= \frac{1}{\sqrt{2}}\{1/2, \underline{1}, 1/2\} \\ G_1 &= \frac{1}{\sqrt{2}}\{\underline{1}\}. \end{aligned} \tag{6.8}$$

G_0 is again a filter we have seen before: Up to multiplication with a constant, it is the treble-reducing filter with values from row 2 of Pascal's triangle. We see something different here when compared to the Haar wavelet, in that the filter G_1 is not the highpass filter corresponding to G_0. The frequency responses are now

$$\begin{aligned} \lambda_{G_0}(\omega) &= \frac{1}{2\sqrt{2}}e^{i\omega} + \frac{1}{\sqrt{2}} + \frac{1}{2\sqrt{2}}e^{-i\omega} = \frac{1}{\sqrt{2}}(\cos\omega + 1) \\ \lambda_{G_1}(\omega) &= \frac{1}{\sqrt{2}}. \end{aligned}$$

$\lambda_{G_1}(\omega)$ thus has magnitude $\frac{1}{\sqrt{2}}$ at all points. The magnitude of $\lambda_{G_0}(\omega)$ is plotted in Figure 6.5.

Comparing with Figure 6.4 we see that here also the frequency response has a zero at π. The frequency response seems also to be flatter around π. For the DWT, Equation (6.2) gives us

$$\begin{aligned} H_0 &= \sqrt{2}\{\underline{1}\} \\ H_1 &= \sqrt{2}\{-1/2, \underline{1}, -1/2\}. \end{aligned} \tag{6.9}$$

Even though G_1 was not the highpass filter corresponding to G_0, we see that, up to a constant, H_1 is (it is a bass-reducing filter with values taken from row 2 of Pascals triangle).

Note that the role of H_1 as the highpass filter corresponding to G_0 is the case in both previous examples. We will prove in the next chapter that this is a much more general result which holds for all wavelets, not only for the orthonormal ones.

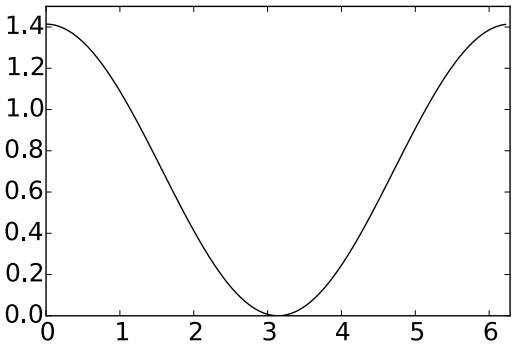

Figure 6.5: The frequency response $\lambda_{G_0}(\omega)$ for the first choice of wavelet for piecewise linear functions.

For the alternative wavelet for piecewise linear functions, we are only able to find expressions for the filters G_0, G_1 at this stage (these can be extracted from Equation (6.3)). In the next chapter we will learn a general technique of computing the transformations the opposite way from these, so this will be handled in the next chapter.

6.1.1 The support of the scaling function and the mother wavelet

The scaling functions and mother wavelets we encounter will turn out to always be functions with compact support. An interesting consequence of equations (6.5) and (6.6) is that we can find the size of these supports from the number of filter coefficients in G_0 and G_1:

Theorem 6.11. *Support size.*
Assume that the filters G_0, G_1 have N_0, N_1 nonzero filter coefficients, respectively, and that ϕ and ψ have compact support. Then the support size of ϕ is $N_0 - 1$, and the support size of ψ is $(N_0 + N_1)/2 - 1$. Moreover, when all the filters are symmetric, the support of ϕ is symmetric around 0, and the support of ψ is symmetric around $1/2$.

Proof. Let q be the support size of ϕ. Then the functions $\phi_{1,n}$ all have support size $q/2$. On the right hand side of Equation (6.5) we thus add N_0 functions, all with support size $q/2$. These functions are translated with $1/2$ with respect to oneanother, so that the sum has support size $q/2 + (N_0 - 1)/2$. Comparing with the support of the left hand side we get the equation $q = q/2 + (N_0 - 1)/2$, so that $q = N_0 - 1$. Similarly, with Equation (6.6), the function on the right hand side has support size $q/2 + (N_1 - 1)/2 = (N_0 + N_1)/2 - 1$, which thus is the support of ψ.

Assume now also that all filters are symmetric, so that the nonzero filter coefficients of G_0 have indices $-(N_0-1)/2, \ldots, (N_0-1)/2$. If ϕ has support $[q_1, q_2]$, $\phi_{1,n}$ has support $[(q_1+n)/2, (q_2+n)/2]$. It follows that the right hand side of Equation (6.5) has support $[(q_1-(N_0-1)/2)/2, (q_2+(N_0-1)/2)/2]$, so that we obtain the equations

$$q_1 = (q_1 - (N_0-1)/2)/2, \text{ and } q_2 = (q_2 + (N_0-1)/2)/2.$$

Solving these we obtain that $q_1 = -(N_0-1)/2$, $q_2 = (N_0-1)/2$, so that the support of ϕ is symmetric around 0. Similarly, the right hand side of Equation (6.6) has support

$$\begin{aligned}\left[\frac{q_1 - \frac{N_1-3}{2}}{2}, \frac{q_2 + \frac{N_1+1}{2}}{2}\right] &= \left[\frac{-\frac{N_0-1}{2} - \frac{N_1-3}{2}}{2}, \frac{\frac{N_0-1}{2} + \frac{N_1+1}{2}}{2}\right] \\ &= \left[-\frac{\frac{N_0+N_1}{2} - 1}{2}, \frac{\frac{N_0+N_1}{2} - 1}{2}\right] + 1.\end{aligned}$$

From this it is clear that ψ has support symmetric around $1/2$. □

Let us use this theorem to verify the supports for the scaling functions and mother wavelets we have already encountered:

- For the Haar wavelet, we know that both filters have 2 coefficients. From Theorem 6.11 it follows that both ϕ and ψ have support size 1, which clearly is true.
- For the the piecewise linear wavelet, the filters were symmetric. G_0 has 3 filter coefficients so that ϕ has support size $3-1=2$. G_1 has one filter coefficient, so that the support size of ψ is $(3+1)/2-1=1$. We should thus have that $\text{supp}(\phi) = [-1, 1]$, and $\text{supp}(\psi) = [0, 1]$. This is clearly true from our previous plots of these functions.
- From Equation (6.3) we see that, for the alternative piecewise linear wavelet, G_0 and G_1 have 3 and 5 filter coefficients, respectively. ψ has thus support size $(3+5)/2-1=3$, so that the support is $[-1, 2]$, which also can be seen to be the case from Figure 5.18.

6.1.2 Wavelets and symmetric extensions

In practice we want to apply the wavelet transform to a symmetric extension, since then symmetric filters can give a better approximation to the underlying analog filters. In order to achieve this, the following result says that we only need to replace the filters H_0, H_1, G_0, and G_1 in the wavelet transform with $(H_0)_r$, $(H_1)_r$, $(G_0)_r$, and $(G_1)_r$.

Theorem 6.12. *Symmetric filters and symmetric extensions.*

If the filters H_0, H_1, G_0, and G_1 in a wavelet transform are symmetric, then the DWT/IDWT preserve symmetric extensions (as defined in Definition 5.42). Also, applying the filters H_0, H_1, G_0, and G_1 to $\breve{\boldsymbol{x}} \in \mathbb{R}^{2N-2}$ in the DWT/IDWT is equivalent to applying $(H_0)_r$, $(H_1)_r$, $(G_0)_r$, and $(G_1)_r$ to $\boldsymbol{x} \in \mathbb{R}^N$ in the same way.

Proof. Since H_0 and H_1 are symmetric, their output from $\breve{\boldsymbol{x}}$ is also a symmetric vector, and by assembling their outputs as the even- and odd-indexed entries, we see that the output $(c_0, w_0, c_1, w_1, \ldots)$ of the MRA-matrix H also is a symmetric vector. The same then applies for the matrix G, since it inverts the first. This proves the first part.

Now, assume that $\boldsymbol{x} \in \mathbb{R}^N$. By definition of $(H_i)_r$, $(H_i\breve{\boldsymbol{x}})_n = ((H_i)_r\boldsymbol{x})_n$ for $0 \leq n \leq N-1$. This means that we get the same first N output elements in a wavelet transform if we repace H_0, H_1 with $(H_0)_r, (H_1)_r$. Since the vectors $(c_0, 0, c_1, 0, \ldots)$ and $(0, w_0, 0, w_1, \ldots)$ also are symmetric vectors when $(c_0, w_0, c_1, w_1, \ldots)$ is, it follows that $(G_0)_r, (G_1)_r$ will reproduce the same first N elements as G_0, G_1 also. In conclusion, for symmetric vectors, the wavelet transform restricted to the first N elements produces the same result when we replace H_0, H_1, G_0, and G_1 with $(H_0)_r$, $(H_1)_r$, $(G_0)_r$, and $(G_1)_r$. This proves the result. □

As in Chapter 4, it follows that when the filters of a wavelet are symmetric, applying $(H_0)_r$, $(H_1)_r$, $(G_0)_r$, and $(G_1)_r$ to the input better approximates an underlying analog filter.

In Section 5.3 we used a parameter `symm` in our call to the DWT amd IDWT kernel functions. This parameter can now also be explained:

idxDWT kernel parameter `symm`

Fact 6.13. *The `symm`-parameter in DWT kernel functions.*

Assume that the filters H_0, H_1, G_0, and G_1 are symmetric. If the `symm` parameter is true, the symmetric versions $(H_0)_r$, $(H_1)_r$, $(G_0)_r$, and $(G_1)_r$ should be applied in the DWT and IDWT, rather than the filters H_0, H_1, G_0, and G_1 themselves. If `symm` is false, the filters H_0, H_1, G_0, and G_1 are applied

In Chapter 8 we will also see how the symmetric versions $(H_0)_r$, $(H_1)_r$, $(G_0)_r$ can be implemented.

What you should have learned in this section.

- How one can find the filters of a wavelet transformation by considering its matrix and its inverse.
- Forward and reverse filter bank transforms.
- How one can implement the DWT and the IDWT with the help of the filters.

- Plot of the frequency responses for the filters of the wavelets we have considered, and their interpretation as lowpass and highpass filters.

Exercise 6.1: Compute filters and frequency responses 1

Write down the corresponding filters G_0 og G_1 for Exercise 5.32. Plot their frequency responses, and characterize the filters as lowpass- or highpass filters.

Exercise 6.2: Symmetry of MRA matrices vs. symmetry of filters 1

Find two symmetric filters, so that the corresponding MRA-matrix, constructed with alternating rows from these two filters, is not a symmetric matrix.

Exercise 6.3: Symmetry of MRA matrices vs. symmetry of filters 2

Assume that an MRA-matrix is symmetric. Are the corresponding filters H_0, H_1, G_0, G_1 also symmetric? If not, find a counterexample.

Exercise 6.4: Finding H_0, H_1 from the H

Assume that one stage in a DWT is given by the MRA-matrix

$$H = \begin{pmatrix} 1/5 & 1/5 & 1/5 & 0 & 0 & 0 & \cdots & 0 & 1/5 & 1/5 \\ -1/3 & 1/3 & -1/3 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & -1/3 & 1/3 & -1/3 & 0 & \cdots 0 & 0 & & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Write down the compact form for the corresponding filters H_0, H_1, and compute and plot the frequency responses. Are the filters symmetric?

Exercise 6.5: Finding G_0, G_1 from the G

Assume that one stage in the IDWT is given by the MRA-matrix

$$G = \begin{pmatrix} 1/2 & -1/4 & 0 & 0 & \cdots \\ 1/4 & 3/8 & 1/4 & 1/16 & \cdots \\ 0 & -1/4 & 1/2 & -1/4 & \cdots \\ 0 & 1/16 & 1/4 & 3/8 & \cdots \\ 0 & 0 & 0 & -1/4 & \cdots \\ 0 & 0 & 0 & 1/16 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots \\ 1/4 & 1/16 & 0 & 0 & \cdots \end{pmatrix}$$

Write down the compact form for the filters G_0, G_1, and compute and plot the frequency responses. Are the filters symmetric?

Exercise 6.6: Finding H from H_0, H_1

Assume that $H_0 = \{1/16, 1/4, \underline{3/8}, 1/4, 1/16\}$, and $H_1 = \{-1/4, \underline{1/2}, -1/4\}$. Plot the frequency responses of H_0 and H_1, and verify that H_0 is a lowpass filter, and that H_1 is a highpass filter. Also write down the change of coordinate matrix $P_{\mathcal{C}_1 \leftarrow \boldsymbol{\phi}_1}$ for the wavelet corresponding to these filters.

Exercise 6.7: Finding G from G_0, G_1

Assume that $G_0 = \frac{1}{3}\{1, \underline{1}, 1\}$, and $G_1 = \frac{1}{5}\{1, -1, \underline{1}, -1, 1\}$. Plot the frequency responses of G_0 and G_1, and verify that G_0 is a lowpass filter, and that G_1 is a highpass filter. Also write down the change of coordinate matrix $P_{\boldsymbol{\phi}_1 \leftarrow \mathcal{C}_1}$ for the wavelet corresponding to these filters.

Exercise 6.8: Computing by hand

In Exercise 5.17 we computed the DWT of two very simple vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, using the Haar wavelet.

a) Compute $H_0\boldsymbol{x}_1$, $H_1\boldsymbol{x}_1$, $H_0\boldsymbol{x}_2$, and $H_1\boldsymbol{x}_2$, where H_0 and H_1 are the filters used by the Haar wavelet.

b) Compare the odd-indexed elements in $H_1\boldsymbol{x}_1$ with the odd-indexed elements in $H_1\boldsymbol{x}_2$. From this comparison, attempt to find an explanation to why the two vectors have very different detail components.

Exercise 6.9: Comment code

Suppose that we run the following algorithm on the sound represented by the vector `x`:

```
c = (x[0::2] + x[1::2])/sqrt(2)
w = (x[0::2] - x[1::2])/sqrt(2)

newx = concatenate([c, w])
newx /= abs(newx).max()
play(newx,44100)
```

a) Comment the code and explain what happens. Which wavelet is used? What do the vectors `c` and `w` represent? Describe the sound you believe you will hear.

b) Assume that we add lines in the code above which sets the elements in the vector `w` to 0 before we compute the inverse operation. What will you hear if you play the new sound you then get?

Exercise 6.10: Computing filters and frequency responses 1

Let us return to the piecewise linear wavelet from Exercise 5.31.

a) With $\hat{\psi}$ as defined as in Exercise 5.31b), compute the coordinates of $\hat{\psi}$ in the basis $\boldsymbol{\phi}_1$ (i.e. $[\hat{\psi}]_{\boldsymbol{\phi}_1}$) with $N = 8$, i.e. compute the IDWT of

$$[\hat{\psi}]_{(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)} = (-\alpha, -\beta, -\delta, 0, 0, 0, 0, -\gamma) \oplus (1, 0, 0, 0, 0, 0, 0, 0),$$

which is the coordinate vector you computed in Exercise 5.31d). For this, you should use the function `IDWTImpl`, with the kernel of the piecewise linear wavelet without symmetric extension as input. Explain that this gives you the filter coefficients of G_1.

b) Plot the frequency response of G_1.

Exercise 6.11: Computing filters and frequency responses 2

Repeat the previous exercise for the Haar wavelet as in Exercise 5.33, and plot the corresponding frequency responses for $k = 2, 4, 6$.

Exercise 6.12: Implementing with symmetric extension

In Exercise 3.6 we implemented a symmetric filter applied to a vector, i.e. when a periodic extension is assumed. The corresponding function was called `filterS(t, x)`, and used the function `numpy.convolve`.

a) Reimplement the function `filterS` so that it also takes a third parameter `symm`. If `symm` is false a periodic extension of `x` should be performed (i.e. filtering as we have defined it, and as the previous version of `filterS` performs it). If `symm` is true, symmetric extensions should be used (as given by Definition 5.42).

b) Implement functions `DWTKernelFilters(H0, H1, G0, G1, x, symm, dual)` and `IDWTKernelFilters(H0, H1, G0, G1, x, symm, dual)` which compute the DWT and IDWT kernels using theorems 6.3 and 6.5, respectively. This function thus bases itself on that the filters of the wavelet are known. The functions should call the function `filterS` from a). Recall also the definition of the parameter `dual` from this section.

With the functions defined in b. you can now define standard DWT and IDWT kernels in the following way, once the filters are known.

```
f    = lambda x, symm, dual: DWTKernelFilters(H0,H1,G0,G1,x,symm,dual)
invf = lambda x, symm, dual: IDWTKernelFilters(H0,H1,G0,G1,x,symm,dual)
```

6.2 Properties of the filter bank transforms of a wavelet

We have now described the DWT/IDWT as linear transformations G, H so that $GH = I$, and where two filters G_0, G_1 characterize G, two filters H_0, H_1 characterize H. G and H are not Toeplitz matrices, however, so they are not filters. Since filters produce the same output frequency from an input frequency, we must have that G and H produce other (undesired) frequencies in the output than those that are present in the input. We will call this phenomenon *aliasing*. In order for $GH = I$, the undesired frequencies must cancel each other, so that we end up with what we started with. Thus, GH must have what we will refer to as *alias cancellation*. This is the same as saying that GH is a filter. In order for $GH = I$, alias cancellation is not enough: We also need that the amount at the given frequency is unchanged, i.e. that $GH\phi_n = \phi_n$ for any Fourier basis vector ϕ_n. We then say that we have *perfect reconstruction*. Perfect reconstruction is always the case for wavelets by construction, but in signal processing many interesting examples (G_0, G_1, H_0, H_1) exist, for which we do not have perfect reconstruction. Historically, forward and reverse filter bank transforms have been around long before they appeared in a wavelet context. Operations where $GH\phi_n = c_n\phi_n$ for all n may also be useful, in particular when c_n is close to 1 for all n. If c_n is real for all n, we say that we have *no phase distortion*. If we have no phase distortion, the output from GH has the same phase, even if we do not have perfect reconstruction. Such "near-perfect reconstruction systems" have also been around long before many perfect reconstruction wavelet systems were designed. In signal processing, these transforms also exist in more general variants, and we will define these later. Let us summarize as follows.

Definition 6.14. *Alias cancellation, phase distortion, and perfect reconstruction.*
We say that we have *alias cancellation* if, for any n,

$$GH\phi_n = c_n\phi_n,$$

for some constant c_n (i.e. GH is a filter). If all c_n are real, we say that we *no phase distortion*. If $GH = I$ (i.e. $c_n = 1$ for all n) we say that we have

perfect reconstruction. If all c_n are close to 1, we say that we have *near-perfect reconstruction*.

In signal processing, one also says that we have perfect- or near-perfect reconstruction when GH equals E_d, or is close to E_d (i.e. the overall result is a delay). The reason why a delay occurs has to do with that the transforms are used in real-time processing, for which we may not be able to compute the output at a given time instance before we know some of the following samples. Clearly the delay is unproblematic, since one can still can reconstruct the input from the output. We will encounter a useful example of near-perfect reconstruction soon in the MP3 standard.

Let us now find a criterium for alias cancellation: When do we have that $GHe^{2\pi irk/N}$ is a multiplum of $e^{2\pi irk/N}$, for any r? We first remark that

$$H(e^{2\pi irk/N}) = \begin{cases} \lambda_{H_0,r} e^{2\pi irk/N} & k \text{ even} \\ \lambda_{H_1,r} e^{2\pi irk/N} & k \text{ odd.} \end{cases}$$

The frequency response of $H(e^{2\pi irk/N})$ is

$$\begin{aligned} &\sum_{k=0}^{N/2-1} \lambda_{H_0,r} e^{2\pi ir(2k)/N} e^{-2\pi i(2k)n/N} + \sum_{k=0}^{N/2-1} \lambda_{H_1,r} e^{2\pi ir(2k+1)/N} e^{-2\pi i(2k+1)n/N} \\ &= \sum_{k=0}^{N/2-1} \lambda_{H_0,r} e^{2\pi i(r-n)(2k)/N} + \sum_{k=0}^{N/2-1} \lambda_{H_1,r} e^{2\pi i(r-n)(2k+1)/N} \\ &= (\lambda_{H_0,r} + \lambda_{H_1,r} e^{2\pi i(r-n)/N}) \sum_{k=0}^{N/2-1} e^{2\pi i(r-n)k/(N/2)}. \end{aligned}$$

Clearly, $\sum_{k=0}^{N/2-1} e^{2\pi i(r-n)k/(N/2)} = N/2$ if $n = r$ or $n = r + N/2$, and 0 else. The frequency response is thus the vector

$$\frac{N}{2}(\lambda_{H_0,r} + \lambda_{H_1,r})\boldsymbol{e}_r + \frac{N}{2}(\lambda_{H_0,r} - \lambda_{H_1,r})\boldsymbol{e}_{r+N/2},$$

so that

$$H(e^{2\pi irk/N}) = \frac{1}{2}(\lambda_{H_0,r} + \lambda_{H_1,r})e^{2\pi irk/N} + \frac{1}{2}(\lambda_{H_0,r} - \lambda_{H_1,r})e^{2\pi i(r+N/2)k/N}. \tag{6.10}$$

Let us now turn to the reverse filter bank transform. We can write

$$\begin{aligned} (e^{2\pi ir\cdot 0/N}, 0, e^{2\pi ir\cdot 2/N}, 0, \ldots, e^{2\pi ir(N-2)/N}, 0) &= \frac{1}{2}(e^{2\pi irk/N} + e^{2\pi i(r+N/2)k/N}) \\ (0, e^{2\pi ir\cdot 1/N}, 0, e^{2\pi ir\cdot 3/N}, \ldots, 0, e^{2\pi ir(N-1)/N}) &= \frac{1}{2}(e^{2\pi irk/N} - e^{2\pi i(r+N/2)k/N}). \end{aligned}$$

This means that

$$\begin{aligned}
&G(e^{2\pi irk/N}) = G_0\left(\frac{1}{2}\left(e^{2\pi irk/N} + e^{2\pi i(r+N/2)k/N}\right)\right) + G_1\left(\frac{1}{2}\left(e^{2\pi irk/N} - e^{2\pi i(r+N/2)k/N}\right)\right)\\
=&\frac{1}{2}(\lambda_{G_0,r}e^{2\pi irk/N} + \lambda_{G_0,r+N/2}e^{2\pi i(r+N/2)k/N}) + \frac{1}{2}(\lambda_{G_1,r}e^{2\pi irk/N} - \lambda_{G_1,r+N/2}e^{2\pi i(r+N/2)k/N})\\
=&\frac{1}{2}(\lambda_{G_0,r} + \lambda_{G_1,r})e^{2\pi irk/N} + \frac{1}{2}(\lambda_{G_0,r+N/2} - \lambda_{G_1,r+N/2})e^{2\pi i(r+N/2)k/N}. \quad (6.11)
\end{aligned}$$

Now, if we combine equations (6.10) and (6.11), we get

$$\begin{aligned}
&GH(e^{2\pi irk/N})\\
&= \frac{1}{2}(\lambda_{H_0,r} + \lambda_{H_1,r})G(e^{2\pi irk/N}) + \frac{1}{2}(\lambda_{H_0,r} - \lambda_{H_1,r})G(e^{2\pi i(r+N/2)k/N})\\
&= \frac{1}{2}(\lambda_{H_0,r} + \lambda_{H_1,r})\left(\frac{1}{2}(\lambda_{G_0,r} + \lambda_{G_1,r})e^{2\pi irk/N} + \frac{1}{2}(\lambda_{G_0,r+N/2} - \lambda_{G_1,r+N/2})e^{2\pi i(r+N/2)k/N})\right)\\
&+ \frac{1}{2}(\lambda_{H_0,r} - \lambda_{H_1,r})\left(\frac{1}{2}(\lambda_{G_0,r+N/2} + \lambda_{G_1,r+N/2})e^{2\pi i(r+N/2)k/N} + \frac{1}{2}(\lambda_{G_0,r} - \lambda_{G_1,r})e^{2\pi irk/N})\right)\\
&= \frac{1}{4}\left((\lambda_{H_0,r} + \lambda_{H_1,r})(\lambda_{G_0,r} + \lambda_{G_1,r}) + (\lambda_{H_0,r} - \lambda_{H_1,r})(\lambda_{G_0,r} - \lambda_{G_1,r})\right)e^{2\pi irk/N}\\
&+ \frac{1}{4}\left((\lambda_{H_0,r} + \lambda_{H_1,r})(\lambda_{G_0,r+N/2} - \lambda_{G_1,r+N/2}) + (\lambda_{H_0,r} - \lambda_{H_1,r})(\lambda_{G_0,r+N/2} + \lambda_{G_1,r+N/2})\right)e^{2\pi i(r+N/2)k/N}\\
&= \frac{1}{2}(\lambda_{H_0,r}\lambda_{G_0,r} + \lambda_{H_1,r}\lambda_{G_1,r})e^{2\pi irk/N} + \frac{1}{2}(\lambda_{H_0,r}\lambda_{G_0,r+N/2} - \lambda_{H_1,r}\lambda_{G_1,r+N/2})e^{2\pi i(r+N/2)k/N}.
\end{aligned}$$

If we also replace with the continuous frequency response, we obtain the following:

Theorem 6.15. *Expression for aliasing.*
We have that

$$\begin{aligned}
GH(e^{2\pi irk/N}) =&\frac{1}{2}(\lambda_{H_0,r}\lambda_{G_0,r} + \lambda_{H_1,r}\lambda_{G_1,r})e^{2\pi irk/N}\\
&+ \frac{1}{2}(\lambda_{H_0,r}\lambda_{G_0,r+N/2} - \lambda_{H_1,r}\lambda_{G_1,r+N/2})e^{2\pi i(r+N/2)k/N}.
\end{aligned} \quad (6.12)$$

In particular, we have alias cancellation if and only if

$$\lambda_{H_0}(\omega)\lambda_{G_0}(\omega+\pi) = \lambda_{H_1}(\omega)\lambda_{G_1}(\omega+\pi). \quad (6.13)$$

We will refer to this as the *alias cancellation condition*. If in addition

$$\lambda_{H_0}(\omega)\lambda_{G_0}(\omega) + \lambda_{H_1}(\omega)\lambda_{G_1}(\omega) = 2, \quad (6.14)$$

we also have perfect reconstruction. We will refer to as the *condition for perfect reconstruction*.

No phase distortion means that we have alias cancellation, and that

$$\lambda_{H_0}(\omega)\lambda_{G_0}(\omega) + \lambda_{H_1}(\omega)\lambda_{G_1}(\omega) \text{ is real.}$$

Now let us turn to how we can construct wavelets/perfect reconstruction systems from FIR-filters (recall from Chapter 3 that FIR filters where filters with a finite number of filter coefficients). We will have use for some theorems which allow us to construct wavelets from prototype filters. In particular we show that, when G_0 and H_0 are given lowpass filters which satisfy a certain common property, we can define unique (up to a constant) highpass filters H_1 and G_1 so that the collection of these four filters can be used to implement a wavelet. We first state the following general theorem.

Theorem 6.16. *Criteria for perfect reconstruction.*
The following statements are equivalent for FIR filters H_0, H_1, G_0, G_1:

- H_0, H_1, G_0, G_1 give perfect reconstruction,
- there exist $\alpha \in \mathbb{R}$ and $d \in \mathbb{Z}$ so that

$$(H_1)_n = (-1)^n \alpha^{-1} (G_0)_{n-2d} \tag{6.15}$$
$$(G_1)_n = (-1)^n \alpha (H_0)_{n+2d} \tag{6.16}$$
$$2 = \lambda_{H_0,n}\lambda_{G_0,n} + \lambda_{H_0,n+N/2}\lambda_{G_0,n+N/2} \tag{6.17}$$

Let us translate this to continuous frequency responses. We first have that

$$\begin{aligned}\lambda_{H_1}(\omega) &= \sum_k (H_1)_k e^{-ik\omega} = \sum_k (-1)^k \alpha^{-1} (G_0)_{k-2d} e^{-ik\omega} \\ &= \alpha^{-1} \sum_k (-1)^k (G_0)_k e^{-i(k+2d)\omega} = \alpha^{-1} e^{-2id\omega} \sum_k (G_0)_k e^{-ik(\omega+\pi)} \\ &= \alpha^{-1} e^{-2id\omega} \lambda_{G_0}(\omega + \pi).\end{aligned}$$

We have a similar computation for $\lambda_{G_1}(\omega)$. We can thus state the following:

Theorem 6.17. *Criteria for perfect reconstruction.*
The following statements are equivalent for FIR filters H_0, H_1, G_0, G_1:

- H_0, H_1, G_0, G_1 give perfect reconstruction,
- there exist $\alpha \in \mathbb{R}$ and $d \in \mathbb{Z}$ so that

$$\lambda_{H_1}(\omega) = \alpha^{-1} e^{-2id\omega} \lambda_{G_0}(\omega + \pi) \tag{6.18}$$
$$\lambda_{G_1}(\omega) = \alpha e^{2id\omega} \lambda_{H_0}(\omega + \pi) \tag{6.19}$$
$$2 = \lambda_{H_0}(\omega)\lambda_{G_0}(\omega) + \lambda_{H_0}(\omega + \pi)\lambda_{G_0}(\omega + \pi) \tag{6.20}$$

Proof. Let us prove first that equations (6.18)- (6.20) for a FIR filter implies that we have perfect reconstruction. Equations (6.18)-(6.19) mean that the alias cancellation condition (6.13) is satisfied, since

$$\begin{aligned}\lambda_{H_1}(\omega)\lambda_{G_1}(\omega+\pi) =& \alpha^{-1}e^{-2id\omega}\lambda_{G_0}(\omega+\pi)(\alpha)e^{2id(\omega+\pi}\lambda_{H_0}(\omega)\\ =& \lambda_{H_0}(\omega)\lambda_{G_0}(\omega+\pi).\end{aligned}$$

Inserting this in the perfect reconstruction condition (6.20), we get

$$\begin{aligned}2 &= \lambda_{H_0}(\omega)\lambda_{G_0}(\omega)+\lambda_{G_0}(\omega+\pi)\lambda_{H_0}(\omega+\pi)\\ &= \lambda_{H_0}(\omega)\lambda_{G_0}(\omega)+\alpha^{-1}e^{-2id\omega}\lambda_{G_0}(\omega+\pi)\alpha e^{2id\omega}\lambda_{H_0}(\omega+\pi)\\ &= \lambda_{H_0}(\omega)\lambda_{G_0}(\omega)+\lambda_{H_1}(\omega)\lambda_{G_1}(\omega),\end{aligned}$$

which is Equation (6.14), so that equations (6.18)- (6.20) imply perfect reconstruction. We therefore only need to prove that any set of FIR filters which give perfect reconstruction, also satisfy these equations. Due to the calculation above, it is enough to prove that equations (6.18)-(6.19) are satisfied. The proof of this will wait till Section 8.1, since it uses some techniques we have not introduced yet. □

Note that, even though conditions (6.18) and (6.19) together ensure that the alias cancellation condition is satisfied, alias cancellation can occur also if these conditions are not satisfied. Conditions (6.18) and (6.19) thus give a stronger requirement than alias cancellation. We will be particularly concerned with wavelets where the filters are symmetric, for which we can state the following corollary.

Corollary 6.18. *Criteria for perfect reconstruction .*
The following statements are equivalent:

- H_0, H_1, G_0, G_1 are the filters of a symmetric wavelet,
- $\lambda_{H_0}(\omega), \lambda_{H_1}(\omega), \lambda_{G_0}(\omega), \lambda_{G_1}(\omega)$ are real functions, and

$$\lambda_{H_1}(\omega) = \alpha^{-1}\lambda_{G_0}(\omega+\pi) \tag{6.21}$$
$$\lambda_{G_1}(\omega) = \alpha\lambda_{H_0}(\omega+\pi) \tag{6.22}$$
$$2 = \lambda_{H_0}(\omega)\lambda_{G_0}(\omega)+\lambda_{H_0}(\omega+\pi)\lambda_{G_0}(\omega+\pi). \tag{6.23}$$

Thw delay d is thus 0 for symmetric wavelets.

Proof. Since H_0 is symmetric, $(H_0)_n = (H_0)_{-n}$, and from equations (6.15) and (6.16) it follows that

$$\begin{aligned}(G_1)_{n-2d} &= (-1)^{n-2d}\alpha(H_0)_n = (-1)^n\alpha^{-1}(H_0)_{-n}\\ &= (-1)^{(-n-2d)}\alpha^{-1}(H_0)_{(-n-2d)+2d} = (G_1)_{-n-2d}\end{aligned}$$

This shows that G_1 is symmetric about both $-2d$, in addition to being symmetric about 0 (by assumption). We must thus have that $d = 0$, so that $(H_1)_n = (-1)^n \alpha (G_0)_n$ and $(G_1)_n = (-1)^n \alpha^{-1} (H_0)_n$. We now get that

$$\begin{aligned}
\lambda_{H_1}(\omega) &= \sum_k (H_1)_k e^{-ik\omega} = \alpha^{-1} \sum_k (-1)^k (G_0)_k e^{-ik\omega} \\
&= \alpha^{-1} \sum_k e^{-ik\pi} (G_0)_k e^{-ik\omega} = \alpha^{-1} \sum_k (G_0)_k e^{-ik(\omega+\pi)} \\
&= \alpha^{-1} \lambda_{G_0}(\omega + \pi),
\end{aligned}$$

which proves Equation (6.21). Equation (6.21) follows similarly. □

When constructing a wavelet it may be that we know one of the two pairs (G_0, G_1), (H_0, H_1), and that we would like to construct the other two. This can be achieved if we can find the constants d and α from above. If the filters are symmetric we just saw that $d = 0$. If G_0, G_1 are known, it follows from from equations (6.15) and(6.16) that

$$1 = \sum_n (G_1)_n (H_1)_n = \sum_n (G_1)_n \alpha^{-1} (-1)^n (G_0)_n = \alpha^{-1} \sum_n (-1)^n (G_0)_n (G_1)_n,$$

so that $\alpha = \sum_n (-1)^n (G_0)_n (G_1)_n$. On the other hand, if H_0, H_1 are known instead, we must have that

$$1 = \sum_n (G_1)_n (H_1)_n = \sum_n \alpha (-1)^n (H_0)_n (H_1)_n = \alpha \sum_n (-1)^n (H_0)_n (H_1)_n,$$

so that $\alpha = 1/(\sum_n (-1)^n (H_0)_n (H_1)_n)$. Let us use these observations to state the filters for the alternative wavelet of piecewise linear functions, which is the only wavelet we have gone through we have not computed the filters and the frequency response for.

Example 6.19. *The alternative piecewise linear wavelet.*

In Equation (6.3) we wrote down the first two columns in $P_{\phi_m \leftarrow \mathcal{C}_m}$ for the alternative piecewise linear wavelet. This gives us that the filters G_0 ans G_1 are

$$\begin{aligned}
G_0 &= \frac{1}{\sqrt{2}} \{1/2, \underline{1}, 1/2\} \\
G_1 &= \frac{1}{\sqrt{2}} \{-1/8, -1/4, \underline{3/4}, -1/4, -1/8\}.
\end{aligned} \tag{6.24}$$

Here G_0 was as for the wavelet of piecewise linear functions since we use the same scaling function. G_1 was changed, however. Let us use Theorem 6.17 and the remark above to compute the two remaining filters H_0 and H_1. These filters

are also symmetric, since G_0, G_1 were. From the simple computation above we get that

$$\alpha = \sum_n (-1)^n (G_0)_n (G_1)_n = \frac{1}{2}\left(-\frac{1}{2}\left(-\frac{1}{4}\right) + 1 \cdot \frac{3}{4} - \frac{1}{2}\left(-\frac{1}{4}\right)\right) = \frac{1}{2}.$$

Theorem 6.17 now gives

$$\begin{aligned}(H_0)_n &= \alpha^{-1}(-1)^n (G_1)_n = 2(-1)^n (G_1)_n \\ (H_1)_n &= \alpha^{-1}(-1)^n (G_0)_n = 2(-1)^n (G_0)_n,\end{aligned} \tag{6.25}$$

so that

$$\begin{aligned}H_0 &= \sqrt{2}\{-1/8, 1/4, \underline{3/4}, 1/4, -1/8\} \\ H_1 &= \sqrt{2}\{-1/2, \underline{1}, -1/2\}.\end{aligned} \tag{6.26}$$

We now have that

$$\begin{aligned}\lambda_{G_1}(\omega) &= -1/(8\sqrt{2})e^{2i\omega} - 1/(4\sqrt{2})e^{i\omega} + 3/(4\sqrt{2}) - 1/(4\sqrt{2})e^{-i\omega} - 1/(8\sqrt{2})e^{-2i\omega} \\ &= -\frac{1}{4\sqrt{2}}\cos(2\omega) - \frac{1}{2\sqrt{2}}\cos\omega + \frac{3}{4\sqrt{2}}.\end{aligned}$$

The magnitude of $\lambda_{G_1}(\omega)$ is plotted in Figure 6.6. Clearly, G_1 now has highpass characteristics, while the lowpass characteristic of G_0 has been preserved.

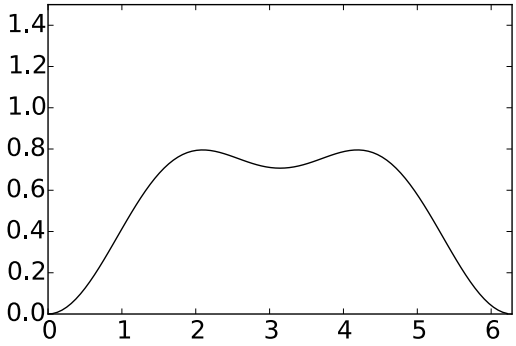

Figure 6.6: The frequency response $\lambda_{G_1}(\omega)$ for the alternative wavelet for piecewise linear functions.

The filters G_0, G_1, H_0, H_1 are particularly important in applications: Apart from the scaling factors $1/\sqrt{2}$, $\sqrt{2}$ in front, we see that the filter coefficients are all dyadic fractions, i.e. they are on the form $\beta/2^j$. Arithmetic operations with dyadic fractions can be carried out exactly on a computer, due to representations as binary numbers in computers. These filters are thus important in applications, since they can be used as transformations for lossless coding. The same argument can be made for the Haar wavelet, but this wavelet had one less vanishing moment.

In the literature, two particular cases of filter banks have been important. They are both refered to as *Quadrature Mirror Filter banks*, or QMF filter banks, and some confusion exist between the two. Let us therefore make precise definitions of the two.

Definition 6.20. *Classical QMF filter banks.*

In the classical definition of a QMF filter banks it is required that $G_0 = H_0$ and $G_1 = H_1$ (i.e. the filters in the forward and reverse transforms are equal), and that

$$\lambda_{H_1}(\omega) = \lambda_{H_0}(\omega + \pi). \tag{6.27}$$

It is straightforward to check that, for a classical QMF filter bank, the forward and reverse transforms are equal (i.e. $G = H$). It is easily checked that conditions (6.18) and (6.19) are satisfied with $\alpha = 1, d = 0$ for a classical QMF filter bank. In particular, the alias cancellation condition is satisfied. The perfect recontruction condition can be written as

$$2 = \lambda_{H_0}(\omega)\lambda_{G_0}(\omega) + \lambda_{H_1}(\omega)\lambda_{G_1}(\omega) = \lambda_{H_0}(\omega)^2 + \lambda_{H_0}(\omega + \pi)^2. \tag{6.28}$$

Unfortunately, it is impossible to find non-trivial FIR-filters which satisfy this quadrature formula (Exercise 6.13). Therefore, classical QMF filter banks which give perfect reconstruction do not exist. Nevertheless, one can construct such filter banks which give close to perfect reconstruction [19], and this together with the fulfillment of the alias cancellation condition still make them useful. In fact, we will see in Section 8.3 that the MP3 standard take use of such filters, and this explains our previous observation that the MP3 standard does not give perfect reconstruction. Note, however, that if the filters in a classical QMF filter bank are symmetric (so that $\lambda_{H_0}(\omega)$ is real), we have no phase distortion.

The second type of QMF filter bank is defined as follows.

Definition 6.21. *Alternative QMF filter banks.*

In the alternative definition of a QMF filter bank it is required that $G_0 = (H_0)^T$ and $G_1 = (H_1)^T$ (i.e. the filter coefficients in the forward and reverse transforms are reverse of oneanother), and that

$$\lambda_{H_1}(\omega) = \overline{\lambda_{H_0}(\omega + \pi)}. \tag{6.29}$$

The perfect reconstruction condition for an alternative QMF filter bank can be written as

$$2 = \lambda_{H_0}(\omega)\lambda_{G_0}(\omega) + \lambda_{H_1}(\omega)\lambda_{G_1}(\omega) = \lambda_{H_0}(\omega)\overline{\lambda_{H_0}(\omega)} + \overline{\lambda_{H_0}(\omega+\pi)}\lambda_{H_0}(\omega+\pi)$$
$$= |\lambda_{H_0}(\omega)|^2 + |\lambda_{H_0}(\omega+\pi)|^2.$$

We see that the perfect reconstruction property of the two definitions of QMF filter banks only differ in that the latter take absolute values. It turns out that the latter also has many interesting solutions, as we will see in Chapter 7. If we in in condition (6.18) substitute $G_0 = (H_0)^T$ we get

$$\lambda_{H_1}(\omega) = \alpha^{-1}e^{-2id\omega}\lambda_{G_0}(\omega+\pi) = \alpha^{-1}e^{-2id\omega}\overline{\lambda_{H_0}(\omega+\pi)}.$$

If we set $\alpha = 1, d = 0$, we get equality here. A similar computation follows for Condition (6.19). In other words, also alternative QMF filter banks satisfy the alias cancellation condition. In the literature, a wavelet is called *orthonormal* if $G_0 = (H_0)^T, G_1 = (H_1)^T$. From our little computation it follows that alternative QMF filter banks with perfect reconstruction are examples of orthonormal wavelets, and correpond to orthonormal wavelets which satisfy $\alpha = 1, d = 0$.

For the Haar wavelet it is easily checked that $G_0 = (H_0)^T, G_1 = (H_1)^T$, but it does not satisfy the relation $\lambda_{H_1}(\omega) = \overline{\lambda_{H_0}(\omega+\pi)}$. Instead it satsifies the relation $\lambda_{H_1}(\omega) = -\overline{\lambda_{H_0}(\omega+\pi)}$. In other words, the Haar wavelet is not an alternative QMF filter bankthe way we have defined them. The difference lies only in a sign, however. This is the reason why the Haar wavelet is still listed as an alternative QMF filter bank in the literature. The additional sign leads to orthonormnal wavelets which satisfy $\alpha = -1, d = 0$ instead.

The following is clear for orthonormal wavelets.

Theorem 6.22. *Orthogonality og the DWT matrix.*

A DWT matrix is orthogonal (i.e. the IDWT equals the transpose of the DWT) if and only if the filters satisfy $G_0 = (H_0)^T, G_1 = (H_1)^T$, i.e. if and only if the MRA equals the dual MRA.

This can be proved simply by observing that, if we transpose the DWT-matrix, Theorem 6.25 says that we get an IDWT matrix with filters $(H_0)^T, (H_1)^T$, and this is equal to the IDWT if and only if $G_0 = (H_0)^T, G_1 = (H_1)^T$. It follows that QMF filter banks with perfect reconstruction give rise to orthonormal wavelets.

Exercise 6.13: Finding FIR filters

Show that it is impossible to find a non-trivial FIR-filter which satisfies Equation (6.28) in the compendium.

Exercise 6.14: The Haar wavelet as an alternative QMF filter bank

Show that the Haar wavelet satisfies $\lambda_{H_1}(\omega) = -\overline{\lambda_{H_0}(\omega + \pi)}$, and $G_0 = (H_0)^T$, $G_1 = (H_1)^T$. The Haar wavelet can thus be considered as an alternative QMF filter bank.

6.3 A generalization of the filter representation, and its use in audio coding

It turns out that the filter representation, which we now have used for an alternative representation of a wavelet transformation, can be generalized in such a way that it also is useful for audio coding. In this section we will first define this generalization. We will then state how the MP3 standard encodes and decodes audio, and see how our generalization is connected to this. Much literature fails to elaborate on this connection. We will call our generalizations *filter bank transforms*, or simply *filter banks*. Just as for wavelets, filters are applied differently for the forward and reverse transforms. The code for this section can be found in a module called `mp3funcs`.

We start by defining the forward filter bank transform and its filters.

Definition 6.23. *Forward filter bank transform.*

Let $H_0, H_1, \ldots, H_{M-1}$ be $N \times N$-filters. A *forward filter bank transform* H produces output $\boldsymbol{z} \in \mathbb{R}^N$ from the input $\boldsymbol{x} \in \mathbb{R}^N$ in the following way:

- $z_{iM} = (H_0\boldsymbol{x})_{iM}$ for any i so that $0 \leq iM < N$.
- $z_{iM+1} = (H_1\boldsymbol{x})_{iM+1}$ for any i so that $0 \leq iM + 1 < N$.
- ...
- $z_{iM+(M-1)} = (H_{M-1}\boldsymbol{x})_{iM+(M-1)}$ for any i so that $0 \leq iM + (M-1) < N$.

In other words, the output of a forward filter bank transform is computed by applying filters $H_0, H_1, \ldots, H_{M-1}$ to the input, and by downsampling and assembling these so that we obtain the same number of output samples as input samples (also in this more general setting this is called critical sampling). $H_0, H_1, \ldots, H_{M-1}$ are also called *analysis filter components*, the output of filter H_i is called channel i *channel*, and M is called the number of channels. The output samples z_{iM+k} are also called the *subband samples* of channel k.

Clearly this definition generalizes the DWT and its analysis filters, since these can be obtained by setting $M = 2$. The DWT is thus a 2-channel forward filter bank transform. While the DWT produces the output $\begin{pmatrix} \boldsymbol{c}_{m-1} \\ \boldsymbol{w}_{m-1} \end{pmatrix}$ from the input $\boldsymbol{c}_m$, an M-channel forward filter bank transform splits the output into M components, instead of 2. Clearly, in the matrix of a forward filter bank

transform the rows repeat cyclically with period M, similarly to MRA-matrices. In practice, the filters in a forward filter bank transform are chosen so that they concentrate on specific frequency ranges. This parallels what we saw for the filters of a wavelet, where one concentrated on high frequencies, one on low frequencies. Using a filter bank to split a signal into frequency components is also called *subband coding*. But the filters in a filter bank are usually not ideal bandpass filters. There exist a variety of different filter banks, for many different purposes [37, 30]. In Chapter 7 we will say more on how one can construct filter banks which can be used for subband coding.

Let us now turn to reverse filter bank transforms.

Definition 6.24. *Reverse filter bank transforms.*

Let $G_0, G_1, \ldots, G_{M-1}$ be $N \times N$-filters. An *reverse filter bank transform* G produces $\boldsymbol{x} \in \mathbb{R}^N$ from $\boldsymbol{z} \in \mathbb{R}^N$ in the following way:

- Define $\boldsymbol{z}_k\mathbb{R}^N$ as the vector where $(\boldsymbol{z}_k)_{iM+k} = \boldsymbol{z}_{iM+k}$ for all i so that $0 \leq iM + k < N$, and $(\boldsymbol{z}_k)_s = 0$ for all other s.

$$\boldsymbol{x} = G_0\boldsymbol{z}_0 + G_1\boldsymbol{z}_1 + \ldots + G_{M-1}\boldsymbol{z}_{M-1}. \tag{6.30}$$

$G_0, G_1, \ldots, G_{M-1}$ are also called *synthesis filter components*.

Again, this generalizes the IDWT and its synthesis filters, and the IDWT can be seen as a 2-channel reverse filter bank transform. Also, in the matrix of a reverse filter bank transform, the columns repeat cyclically with period M, similarly to MRA-matrices. Also in this more general setting the filters G_i are in general different from the filters H_i. But we will see that, just as we saw for the Haar wavelet, there are important special cases where the analysis and synthesis filters are equal, and where their frequency responses are simply shifts of oneanother. It is clear that definitions 6.23 and 6.24 give the diagram for computing forward and reverse filter bank transforms shown in Figure 6.7:

Here $\downarrow_M$ and $\uparrow_M$ means that we extract every M'th element in the vector, and add $M-1$ zeros between the elements, respectively, similarly to how we previously defined $\downarrow_2$ and $\uparrow_2$. Comparing Figure 6.3 with Figure 6.7 makes the similarities between wavelet transformations and the transformation used in the MP3 standard very visible: Although the filters used are different, they are subject to the same kind of processing, and can therefore be subject to the same implementations.

In general it may be that the synthesis filters do not invert exactly the analysis filters. If the synthesis system exactly inverts the analysis system, we say that we have a *perfect reconstruction filter bank*. Since the analysis system introduces undesired frequencies in the different channels, these have to cancel in the inverse transform, in order to reconstruct the input exactly.

We will have use for the following simple connection between forward and reverse filter bank transforms, which follows imemdiately from the definitions.

Theorem 6.25. *Connection between forward and reverse filter bank transforms.*

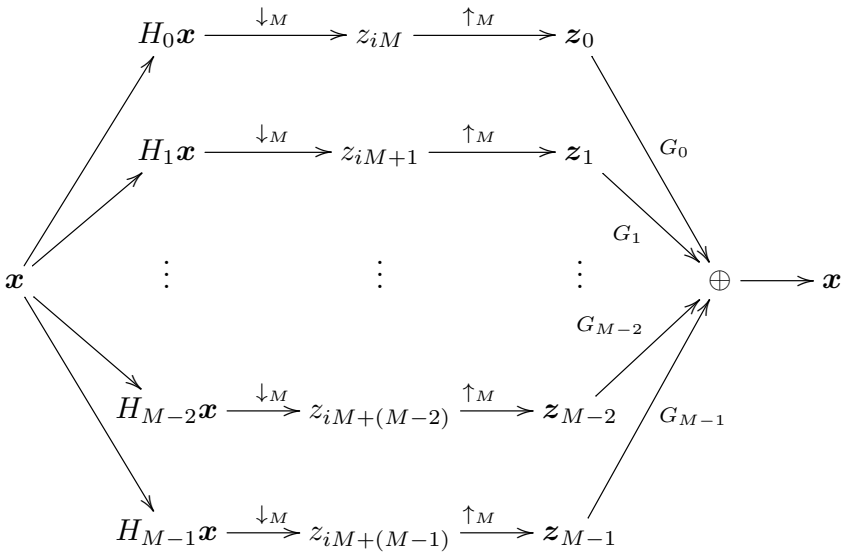

Figure 6.7: Illustration of forward and reverse filter bank transforms.

Assume that H is a forward filter bank transform with filters $H_0, \ldots, H_{N-1}$. Then H^T is a reverse filter bank transform with filters $G_0 = (H_0)^T, \ldots, G_{N-1} = (H_{N-1})^T$.

6.3.1 Forward filter bank transform used for encoding in the MP3 standard

Now, let us turn to the MP3 standard. The MP3 standard document states that it applies a filter bank, and explains the following procedure for applying this filter bank, see p. 67 of the standard document (the procedure is slightly modified with mathematical terminology adapted to this book):

- Input 32 audio samples at a time.
- Build an input sample vector $X \in \mathbb{R}^{512}$, where the 32 new samples are placed first, all other samples are delayed with 32 elements. In particular the 32 last samples are taken out.
- Multiply X componentwise with a vector C (this vector is defined through a table in the standard), to obtain a vector $Z \in \mathbb{R}^{512}$. The standard calls this *windowing*.
- Compute the vector $Y \in \mathbb{R}^{64}$ where $Y_i = \sum_{j=0}^{7} Z_{i+64j}$. The standard calls this a *partical calculation*.

- Calculate $S = MY \in \mathbb{R}^{32}$, where M is the 32×64- matrix where $M_{ik} = \cos((2i+1)(k-16)\pi/64)$. S is called the vector of output samples, or output subband samples. The standard calls this *matrixing*.

The standard does not motivate these steps, and does not put them into the filter bank transform framework which we have established. Also, the standard does not explain how the values in the vector C have been constructed.

Let us start by proving that the steps above really corresponds to applying a forward filter bank transform, and let us state the corresponding filters of this transform. The procedure computes 32 outputs in each iteration, and each of them is associated with a subband. Therefore, from the standard we would guess that we have $M = 32$ channels, and we would like to find the corresponding 32 filters $H_0, H_1, \ldots, H_{31}$.

It may seem strange to use the name matrixing here, for something which obviously is matrix multiplication. The reason for this name must be that the at the origin of the procedure come from outside a linear algebra framework. The name windowing is a bit strange, too. This really does not correspond to applying a window to the sound samples as we explained in Section 3.3.1. We will see that it rather corresponds to applying a filter coefficient to a sound sample. A third and final thing which seems a bit strange is that the order of the input samples is reversed, since we are used to having the first sound samples in time with lowest index. This is perhaps more usual to do in an engineering context, and not so usual in a mathematical context. FIFO.

Clearly, the procedure above defines a linear transformation, and we need to show that this linear transformation coincides with the procedure we defined for a forward filter bank transform, for a set of 32 filters. The input to the transformation are the audio samples, which we will denote by a vector $\boldsymbol{x}$. At iteration s of the procedure above the input audio samples are $x_{32s-512}, x_{32s-510}, \ldots, x_{32s-1}$, and $X_i = x_{32s-i-1}$ due to the reversal of the input samples. The output to the transformation at iteration s of the procedure are the $S_0, \ldots, S_{31}$. We assemble these into a vector $\boldsymbol{z}$, so that the output at iteration s are $z_{32(s-1)} = S_0$, $z_{32(s-1)+1} = S_1, \ldots, z_{32(s-1)+31} = S_{31}$.

We will have use for the following cosine-properties, which are easily verified:

$$\cos\left(2\pi(n+1/2)(k+2Nr)/(2N)\right) = (-1)^r \cos\left(2\pi(n+1/2)k/(2N)\right) \tag{6.31}$$

$$\cos\left(2\pi(n+1/2)(2N-k)/(2N)\right) = -\cos\left(2\pi(n+1/2)k/(2N)\right). \tag{6.32}$$

With the terminology above and using Property (6.31) the transformation can be written as

$$\begin{aligned}
z_{32(s-1)+n} &= \sum_{k=0}^{63} \cos((2n+1)(k-16)\pi/64) Y_k = \sum_{k=0}^{63} \cos((2n+1)(k-16)\pi/64) \sum_{j=0}^{7} Z_{k+64j} \\
&= \sum_{k=0}^{63} \sum_{j=0}^{7} (-1)^j \cos((2n+1)(k+64j-16)\pi/64) Z_{k+64j} \\
&= \sum_{k=0}^{63} \sum_{j=0}^{7} \cos((2n+1)(k+64j-16)\pi/64)(-1)^j C_{k+64j} X_{k+64j} \\
&= \sum_{k=0}^{63} \sum_{j=0}^{7} \cos((2n+1)(k+64j-16)\pi/64)(-1)^j C_{k+64j} x_{32s-(k+64j)-1}.
\end{aligned}$$

Now, if we define $\{h_r\}_{r=0}^{511}$ by $h_{k+64j} = (-1)^j C_{k+64j}$, $0 \leq j < 8, 0 \leq k < 64$, and $h^{(n)}$ as the filter with coefficients $\{\cos((2n+1)(k-16)\pi/64) h_k\}_{k=0}^{511}$, the above can be simplified as

$$\begin{aligned}
z_{32(s-1)+n} &= \sum_{k=0}^{511} \cos((2n+1)(k-16)\pi/64) h_k x_{32s-k-1} = \sum_{k=0}^{511} (h^{(n)})_k x_{32s-k-1} \\
&= (h^{(n)} \boldsymbol{x})_{32s-1} = (E_{n-31} h^{(n)} \boldsymbol{x})_{32(s-1)+n}.
\end{aligned}$$

This means that the output of the procedure stated in the MP3 standard can be computed as a forward filter bank transform, and that we can choose the analysis filters as $H_n = E_{n-31} h^{(n)}$.

Theorem 6.26. *Forward filter bank transform for the MP3 standard.*

Define $\{h_r\}_{r=0}^{511}$ by $h_{k+64j} = (-1)^j C_{k+64j}$, $0 \leq j < 8, 0 \leq k < 64$, and $h^{(n)}$ as the filter with coefficients $\{\cos((2n+1)(k-16)\pi/64) h_k\}_{k=0}^{511}$. If we define $H_n = E_{n-31} h^{(n)}$, the procedure stated in the MP3 standard corresponds to applying the corresponding forward filter bank transform.

The filters H_n were shown in Example 3.37 as examples of filters which concentrate on specific frequency ranges. The h_k are the filter coefficients of what is called a prototype filter. This kind of filter bank is also called a *cosine-modulated filter*. The multiplication with $\cos\left(2\pi(n+1/2)(k-16)/(2N)\right) h_k$, modulated the filter coefficients so that the new filter has a frequency response which is simply shifted in frequency in a symmetric manner: In Exercise 3.30, we saw that, by multiplying with a cosine, we could contruct new filters with real filter coefficients, which also corresponded to shifting a prototype filter in frequency. Of course, multiplication with a complex exponential would also shift the frequency response (such filter banks are called *DFT-modulated filter banks*), but the problem with this is that the new filter has complex coefficients: It will turn out that cosine-modulated filter banks can also be constructed so that they are invertible, and that one can find such filter banks where the inverse is easily found.

The effect of the delay in the definition of H_n is that, for each n, the multiplications with the vector $\boldsymbol{x}$ are "aligned", so that we can save a lot of multiplications by performing this multiplication first, and summing these. We actually save even more multiplications in the sum where j goes from 0 to 7, since we here multiply with the same cosines. The steps defined in the MP3 standard are clearly motivated by the desire to reduce the number of multiplications due to these facts. A simple arithmetic count illutrates these savings: For every 32 output samples, we have the following number of multiplications:

- The first step computes 512 multiplications.
- The second step computes 64 sums of 8 elements each, i.e. a total of $7 \times 64 = 448$ additions (note that $q = 512/64 = 8$).

The standard says nothing about how the matrix multiplication in the third step can be implemented. A direct multiplication would yield $32 \times 64 = 2048$ multiplications, leaving a total number of multiplications at 2560. In a direct implementation of the forward filter bank transform, the computation of 32 samples would need $32 \times 512 = 16384$ multiplications, so that the procedure sketched in the standard gives a big reduction.

The standard does not mention all possibilities for saving multiplications, however: We can reduce the number of multiplications even further, since clearly a DCT-type implementation can be used for the matrixing operation. We already have an efficient implementation for multiplication with a 32×32 type-III cosine matrix (this is simply the IDCT). We have seen that this implementation can be chosen to reduce the number of multiplications to $N \log_2 N/2 = 80$, so that the total number of multiplications is $512 + 80 = 592$. Clearly then, when we use the DCT, the first step is the computationally most intensive part.

6.3.2 Reverse filter bank transform used for decoding in the MP3 standard

Let us now turn to how decoding is specified in the MP3 standard, and see that we can associate this with a reverse filter bank transform. The MP3 standard also states the following procedure for decoding:

- Input 32 new subband samples as the vector S.
- Change vector $V \in \mathbb{R}^{512}$, so that all elements are delayed with 64 elements. In particular the 64 last elements are taken out.
- Set the first 64 elements of V as $NS \in \mathbb{R}^{64}$, where N is the 64×32-matrix where $N_{ik} = \cos((16 + i)(2k + 1)\pi/64)$. The standard also calls this matrixing.
- Build the vector $U \in \mathbb{R}^{512}$ from V from the formulas $U_{64i+j} = V_{128i+j}$, $U_{64i+32+j} = V_{128i+96+j}$ for $0 \leq i \leq 7$ and $0 \leq j \leq 31$, i.e. U is the vector where V is first split into segments of length 132, and U is constructed by assembling the first and last 32 elements of each of these segments.

- Multiply U componentwise with a vector D (this vector is defined in the standard), to obtain a vector $W \in \mathbb{R}^{512}$. The standard also calls this windowing.
- Compute the 32 next sound samples as $\sum_{i=0}^{15} W_{32i+j}$.

To interpret this also in terms of filters, rewrite first steps 4 to 6 as

$$\begin{aligned} x_{32(s-1)+j} &= \sum_{i=0}^{15} W_{32i+j} = \sum_{i=0}^{15} D_{32i+j} U_{32i+j} \\ &= \sum_{i=0}^{7} D_{64i+j} U_{64i+j} + \sum_{i=0}^{7} D_{64i+32+j} U_{64i+32+j} \\ &= \sum_{i=0}^{7} D_{64i+j} V_{128i+j} + \sum_{i=0}^{7} D_{64i+32+j} V_{128i+96+j}. \end{aligned} \tag{6.33}$$

The elements in V are obtained by "matrixing" different segments of the vector $\boldsymbol{z}$. More precisely, at iteration s we have that

$$\begin{pmatrix} V_{64r} \\ V_{64r+1} \\ \vdots \\ V_{64r+63} \end{pmatrix} = N \begin{pmatrix} z_{32(s-r-1)} \\ z_{32(s-r-1)+1} \\ \vdots \\ z_{32(s-r-1)+31} \end{pmatrix},$$

so that

$$V_{64r+j} = \sum_{k=0}^{31} \cos((16+j)(2k+1)\pi/64) z_{32(s-r-1)+k}$$

for $0 \leq j \leq 63$. Since also

$$V_{128i+j} = V_{64(2i)+j} \qquad V_{128i+96+j} = V_{64(2i+1)+j+32},$$

we can rewrite Equation (6.33) as

$$\begin{aligned} &\sum_{i=0}^{7} D_{64i+j} \sum_{k=0}^{31} \cos((16+j)(2k+1)\pi/64) z_{32(s-2i-1)+k} \\ &+ \sum_{i=0}^{7} D_{64i+32+j} \sum_{k=0}^{31} \cos((16+j+32)(2k+1)\pi/64) z_{32(s-2i-2))+k}. \end{aligned}$$

Again using Relation (6.31), this can be written as

$$\sum_{k=0}^{31}\sum_{i=0}^{7}(-1)^i D_{64i+j}\cos((16+64i+j)(2k+1)\pi/64)z_{32(s-2i-1)+k}$$
$$+\sum_{k=0}^{31}\sum_{i=0}^{7}(-1)^i D_{64i+32+j}\cos((16+64i+j+32)(2k+1)\pi/64)z_{32(s-2i-2)+k}.$$

Now, if we define $\{g_r\}_{r=0}^{511}$ by $g_{64i+s}=(-1)^i C_{64i+s}$, $0\le i<8, 0\le s<64$, and $g^{(k)}$ as the filter with coefficients $\{\cos((r+16)(2k+1)\pi/64)g_r\}_{r=0}^{511}$, the above can be simplified as

$$\begin{aligned}&\sum_{k=0}^{31}\sum_{i=0}^{7}(g^{(k)})_{64i+j}z_{32(s-2i-1)+k}+\sum_{k=0}^{31}\sum_{i=0}^{7}(g^{(k)})_{64i+j+32}z_{32(s-2i-2)+k}\\ =&\sum_{k=0}^{31}\left(\sum_{i=0}^{7}(g^{(k)})_{32(2i)+j}z_{32(s-2i-1)+k}+\sum_{i=0}^{7}(g^{(k)})_{32(2i+1)+j}z_{32(s-2i-2)+k}\right)\\ =&\sum_{k=0}^{31}\sum_{r=0}^{15}(g^{(k)})_{32r+j}z_{32(s-r-1)+k},\end{aligned}$$

where we observed that $2i$ and $2i+1$ together run through the values from 0 to 15 when i runs from 0 to 7. Since $\boldsymbol{z}$ has the same values as $\boldsymbol{z}_k$ on the indices $32(s-r-1)+k$, this can be written as

$$\begin{aligned}&=\sum_{k=0}^{31}\sum_{r=0}^{15}(g^{(k)})_{32r+j}(\boldsymbol{z}_k)_{32(s-r-1)+k}\\ &=\sum_{k=0}^{31}(g^{(k)}\boldsymbol{z}_k)_{32(s-1)+j+k}=\sum_{k=0}^{31}((E_{-k}g^{(k)})\boldsymbol{z}_k)_{32(s-1)+j}.\end{aligned}$$

By substituting a general s and j we see that $\boldsymbol{x}=\sum_{k=0}^{31}(E_{-k}g^{(k)})\boldsymbol{z}_k$. We have thus proved the following.

Theorem 6.27. *Reverse filter bank transform for the MP3 standard.*

Define $\{g_r\}_{r=0}^{511}$ by $g_{64i+s}=(-1)^i C_{64i+s}$, $0\le i<8, 0\le s<64$, and $g^{(k)}$ as the filter with coefficients $\{\cos((r+16)(2k+1)\pi/64)g_r\}_{r=0}^{511}$. If we define $G_k=E_{-k}g^{(k)}$, the procedure stated in the MP3 standard corresponds to applying the corresponding reverse filter bank transform.

In other words, both procedures for encoding and decoding stated in the MP3 standard both correspond to filter banks: A forward filter bank transform for the encoding, and a reverse filter bank transform for the decoding. Moreover, both filter banks can be constructed by cosine-modulating prototype filters, and the coefficients of these prototype filters are stated in the MP3 standard (up to

multiplication with an alternating sign). Note, however, that the two prototype filters may be different. When we compare the two tables for these coefficients in the standard they do indeed seem to be different. At closer inspection, however, one sees a connection: If you multiply the values in the D-table with 32, and reverse them, you get the values in the C-table. This indicates that the analysis and synthesis prototype filters are the same, up to multiplication with a scalar. This connection will be explained in Section 8.3.

While the steps defined in the MP3 standard for decoding seem a bit more complex than the steps for encoding, they are clearly also motivated by the desire to reduce the number of multiplications. In both cases (encoding and decoding), the window tables (C and D) are in direct connection with the filter coefficients of the prototype filter: one simply adds a sign which alternates for every 64 elements. The standard document does not mention this connection, and it is perhaps not so simple to find this connection in the literature (but see [26]).

The forward and reverse filter bank transforms are clearly very related. The following result clarifies this.

Theorem 6.28. *Connection between the forward and reverse filter bank transforms in the MP3 standard.*

Assume that a forward filter bank transform has filters on the form $H_i = E_{i-31}h^{(i)}$ for a prototype filter h. Then $G = E_{481}H^T$ is a reverse filter bank transform with filters on the form $G_k = E_{-k}g^{(k)}$, where g is a prototype filter where the elements equal the reverse of those in h. Vice versa, $H = E_{481}G^T$.

Proof. From Theorem 6.25 we know that H^T is a reverse filter bank transform with filters

$$(H_i)^T = (E_{i-31}h^{(i)})^T = E_{31-i}(h^{(i)})^T.$$

$(h^{(i)})^T$ has filter coefficients $\cos((2i+1)(-k-16)\pi/64))h_{-k}$. If we delay all $(H_i)^T$ with $481 = 512 - 31$ elements as in the theorem, we get a total delay of $512 - 31 + 31 - i = 512 - i$ elements, so that we get the filter

$$\begin{aligned}
&E_{512-i}\{\cos((2i+1)(-k-16)\pi/64))h_{-k}\}_k \\
&= E_{-i}\{\cos((2i+1)(-(k-512)-16)\pi/64))h_{-(k-512)}\}_k \\
&= E_{-i}\{\cos((2i+1)(k+16)\pi/64))h_{-(k-512)}\}_k.
\end{aligned}$$

Now, we define the prototype filter g with elements $g_k = h_{-(k-512)}$. This has, just as h, its support on $[1, 511]$, and consists of the elements from h in reverse order. If we define $g^{(i)}$ as the filter with coefficients $\cos((2i+1)(k+16)\pi/64))g_k$, we see that $E_{481}H^T$ is a reverse filter bank transform with filters $E_{-i}g^{(i)}$. Since $g^{(k)}$ now has been defined as for the MP3 standard, and its elements are the reverse of those in h, the result follows. □

We will have use for this result in Section 8.3, when we find conditions on the prototype filter in order for the reverse transform to invert the forward

transform. Preferably, the reverse filter bank transform inverts exactly the forward filter bank transform. In Exercise 6.16 we construct examples which show that this is not the case. In the same exercise we also find many examples where the reverse transform does what we would expect. These examples will also be explained in Section 8.3, where we also will see how one can get around this so that we obtain a system with perfect reconstruction. It may seem strange that the MP3 standard does not do this.

In the MP3 standard, the output from the forward filter bank transform is processed further, before the result is compressed using a lossless compression method.

Exercise 6.15: Plotting frequency responses

The values C_q, D_q can be found by calling the functions `mp3ctable`, `mp3dtable` which can be found on the book's webpage.

a) Use your computer to verify the connection we stated between the tables C and D, i.e. that $D_i = 32C_i$ for all i.

b) Plot the frequency responses of the corresponding prototype filters, and verify that they both are lowpass filters. Use the connection from Theorem (6.26) to find the prototype filter coefficients from the C_q.

Exercise 6.16: Implementing forward and reverse filter bank transforms

It is not too difficult to make implementations of the forward and reverse steps as explained in the MP3 standard. In this exercise we will experiment with this. In your code you can for simplicity assume that the input and output vectors to your methods all have lengths which are multiples of 32. Also, use the functions `mp3ctable`, `mp3dtable` mentioned in the previous exercise.

a) Write a function `mp3forwardfbt` which implements the steps in the forward direction of the MP3 standard.

b) Write also a function `mp3reversefbt` which implements the steps in the reverse direction.

6.4 Summary

We started this chapter by noting that, by reordering the target base of the DWT, the change of coordinate matrix took a particular form. From this form we understood that the DWT could be realized in terms of two filters H_0 and H_1, and that the IDWT could be realized in a similar way in terms of two filters G_0 and G_1. This gave rise to what we called the filter representation of wavelets. The filter representation gives an entirely different view on wavelets: instead of constructing function spaces with certain properties and deducing corresponding

filters from these, we can instead construct filters with certain properties (such as alias cancellation and perfect reconstruction), and attempt to construct corresponding mother wavelets, scaling functions, and function spaces. This strategy, which replaces problems from function theory with discrete problems, will be the subject of the next chapter. In practice this is what is done.

We stated what is required for filter bank matrices to invert each other: The frequency responses of the lowpass filters needed to satisfy a certain equation, and once this is satsified the highpass filters can easily be obtained in the same way we previously obtained highpass filters from lowpass filters. We will return to this equation in the next chapter.

A useful consequence of the filter representation was that we could reuse existing implementations of filters to implement the DWT and the IDWT, and reuse existing theory, such as symmetric extensions. For wavelets, symmetric extensions are applied in a slightly different way, when compared to the developments which lead to the DCT. We looked at the frequency responses of the filters for the wavelets we have encountered upto now. From these we saw that G_0, H_0 were lowpass filters, and that G_1, H_1 were highpass filters, and we argued why this is typically the case for other wavelets as well. The filter reprersentation was also easily generalized from 2 to $M > 2$ filters, and such transformations had a similar interpretation in terms of splitting the input into a uniform set of frequencies. Such transforms were generally called filter bank transforms, and we saw that the processing performed by the MP3 standard could be interpreted as a certain filter bank transform, called a cosine-modulated filter bank. This is just one of many possible filter banks. In fact, the filter bank of the MP3 standard is largely outdated, since it is too simple, and as we will see it does not even give perfect reconstruction (only alias cancellation and no phase distortion). It is merely chosen here since it is the simplest to present theoretically, and since it is perhaps the best known standard for compression of sound. Other filters banks with better properties have been constructed, and they are used in more recent standards. In many of these filter banks, the filters do not partition frequencies uniformly, and have been adapted to the way the human auditory system handles the different frequencies. Different contruction methods are used to construct such filter banks. The motivation behind filter bank transforms is that their output is more suitable for further processing, such as compression, or playback in an audio system, and that they have efficient implementations.

We mentioned that the MP3 standard does not say how the prototype filters were chosen. We will have more to say on what dictates their choice in Section 8.3.

There are several differences between the use of wavelet transformations in wavelet theory, and the use of filter bank transforms in signal processing theory One is that wavelet transforms are typically applied in stages, while filter bank transforms often are not. Nevertheless, such use of filter banks also has theoretical importance, and this gives rise to what is called *tree-structured filter banks* [37]. Another difference lies in the use of the term perfect reconstruction system. In wavelet theory this is a direct consequence of the wavelet construction, since the DWT and the IDWT correspond to change of coordinates to and from the same bases. The alternative QMF filter bank was used as an example

of a filter bank which stems from signal processing, and which also shows up in wavelet transformation. In signal processing theory, one has a wider perspective, since one can design many useful systems with fast implementations when one replaces the perfect reconstruction requirement with a near perfect reconstruction requirement. One instead requires that the reverse transform gives alias cancellation. The classical QMF filter banks were an example of this. The original definition of classical QMF filter banks are from [7], and differ only in a sign from how they are defined here.

All filters we encounter in wavelets and filter banks in this book are FIR. This is just done to limit the exposition. Much useful theory has been developed using IIR-filters.