Wavelets for Feature Detection; Theoretical background

M. van Berkel

CST 2010.009

Literature study

Coach: Ir. G. Witvoet
Supervisors: Dr. ir. P.W.J.M. Nuij
Prof. dr. ir. M. Steinbuch

Eindhoven University of Technology
Department of Mechanical Engineering
Control Systems Technology Group

Eindhoven, March 2010

Contents

Chapter 1

Introduction

Wavelet theory is a relatively new tool for signal analysis. Although the first wavelet was derived by Haar in 1909, the real breakthrough came in 1988 when Daubechies derived her famous wavelet design. Since then a lot of wavelets have been designed for many applications. A wavelet is a function that goes to zero at the bounds and between these bounds it behaves like a wave. The word wavelet originates from a combination of wave and the French word for small wave, ondelette.
This small wave is convoluted with a signal. This expresses the amount of the overlap of the wavelet as it is shifted over the signal. In other words, where the signal resembles the wavelet, the resulting function will have high magnitude and where it has a totally different shape it will have low magnitude. How well the wavelet resembles the signal locally can be calculated by shifting the small wave over the entire signal. By not only comparing the signal with shifted wavelets but also comparing wavelets that are differently dilated, something can be said about the scale (frequency) content of the signal.
There are many different wavelets with a verity of shapes and properties. Therefore it is a much broader tool for signal analysis compared to the Fourier Transformation where the signal is only compared to sinusoids. However, Fourier analysis plays an important role in wavelet analysis and is still one of the most important tool for signal analysis. Therefore in Chapter 3 a short introduction is given about signal analysis in general and about the decomposition of signals. The Fourier Transformation and the Short Time Fourier Transformation are introduced as two possible analyzing methods. Thereafter, in Chapter 4, the Continuous Wavelet Transformation is introduced and two examples are presented. The Continuous Wavelet Transformation is in general considered redundant because it uses continuous signals and therefore needs to be made discrete before it can be used in an application. This makes the Continuous Wavelet Transform inefficient. This can be overcome by using the Discrete Wavelet Transform. It is very efficient if it is applied through a filter bank, which is an important part of the Discrete Wavelet Transform. The Discrete Wavelet Transform is discussed in Chapter 5. In Chapter 6 its most important properties are explained. In addition a number of issues related with the DWT are discussed. Finally the most important applications are explained in Chapter 7, whereafter in Chapter 8 some conclusions are presented.

Chapter 2

Literature overview

In this chapter a small overview is given about the most important wavelet literature available. Although the subject is relatively new, already a lot of literature is available in the form of books and articles. Especially after the important breakthrough in 1988 when Daubechies derived her famous wavelet design, a lot of new books have been published. However most books have been written by mathematicians, for mathematicians. Therefore most books are difficult to read, especially if one is new to this subject. If one wants to get a quick overview on wavelets and their applications, it is recommended to read [26] and [3]. The first gives a quick overview about wavelets and their applications, the second is easy to read because of the many illustrations and relatively easy mathematical background but still gives an extended overview.
When one wishes to get a full understanding of wavelet transforms and to be able to apply wavelets on an actual problem this literature will probably be insufficient. Before starting to read any more extensive literature it is important to make a distinction between the Continuous Wavelet Transform (CWT) and the Discrete Wavelet Transform (DWT). Although both transforms are described in most books, most of them focus on the DWT. If you need to analyze, change and reconstruct your signal or need a fast and efficient transform, one should use the DWT. In case one only needs to analyze a signal and computation time is of less importance probably the CWT (in discretized form) will do the job. This is of course only a guideline but combining these two methods of transforms is a subject on its own [12]. The CWT is relatively easy to understand and is often described as an introduction to the DWT in most books. As already mentioned a good overview on CWT is given by [3]. A good overview of Continuous Wavelets is also presented in [27], which also describes the DWT. The DWT is the subject of most modern books. A good and easy to read introduction is given by [37] and the first chapter of [28]. They describe the basics of the DWT and are easy to read without any difficult mathematics. However this is only an introduction and gives hardly any background information. Two books that give a full description and that have understandable mathematics are [8] and [32]. Where [8] focusses on the actual DWT and theoretical background, [32] focuses on the application of the DWT through filter banks. Together they offer more than enough background to bring the DWT in practice. They describe how a DWT works and discuss the most important properties of wavelets. The most extensive description of the general properties of discrete wavelets used in practice are found in [8]. On the other hand [32] extensively describes how to apply the DWT, what their properties and problems are related to the DWT.
If one wishes to get a full understanding of wavelets and their properties a number of good books are available, although very mathematical. In this case the book of Daubechies [13] is an absolute must. It describes in full extend the most important classes of orthogonal wavelets, their properties and considerations. Also [11] offers an extensive mathematical description of wavelets. Although both books give good background information the book of [24] is probably one of the best books to use as a reference. It offers an extensive overview of signal analysis and describes almost everything in great detail and precision. However its main focus is the Continuous Wavelet Transform. If one is looking for a good digital reference, [38, 27] offer a quick overview of the most important properties of wavelets. One important last remark is

that the main application of wavelets is image compression. Therefore most papers and books focus on this application.

Chapter 3

Decomposition of signals and the Fourier Transform

The analysis of signals is based on the decomposition of a signal in sub-signals; this holds for both the Fourier Transform and Wavelet Transforms. Therefore in this chapter the decomposition of signals in general is explained. The most widely known signal analysis tool is the Fourier Transformation and for that reason it is also discussed here. Finally the Short Time Fourier Transform (STFT) is explained, which is commonly seen as the first step towards wavelets.

3.1 Decomposition of signals

The decomposition of signals in sub-signals is the fundament of almost all signal analysis techniques. The goal is to express a signal as a linear combination of other functions. These other functions are often shifted and dilated functions of some (mother) function. Often only the signal itself and the functions one wants to express this signal in are known, i.e. the signal $g(t)$ needs to be expressed as function of ψ_n. In this report the signal $g\,(t)$ is considered real, although the functions ψ_n can be complex. It is possible with the help of (expansion) coefficients a_n, to express $g(t)$ as function of ψ_n. The coefficients a_n represent the presence of ψ_n in the signal $g(t)$,

$$g(t) = \sum_n a_n \, \psi_n(t) \tag{3.1}$$

This kind of decomposition is also called a linear expansion. The number of functions ψ_n depend on the signal and type of functions used. For instance the most common used functions are sinusoids (Fourier). If the signal is a smooth function they can often be expressed as a finite number of functions ψ_n (n is finite). On the other hand when the function has a discontinuity, e.g. a sawtooth, an infinite number of functions are needed to represent it exactly. However in practice only a finite number of points of a function (sampling) and a finite number of functions ψ_n are used. If the expansion (3.1) is unique, the set of functions ψ_n is called a *basis* for the class of functions that can be expressed this way [8]. Normally the number of functions are much less than the number of sample points especially in the case of periodic functions. Therefore the signal built up from these functions is often only an approximation of the original signal. Finding the best approximation is an important part of signal analysis and will be partly discussed in the next section. In this section it is assumed that the number of functions equals the number of sample points and the signal can be perfectly represented by these functions. Although this seems not realistic from a Fourier analysis point of view, from a wavelet point of view this is not too far from the truth. This kind of signals and functions will be discussed in this section. Let us consider a signal $g(t)$ in its discrete form $g(k)$ as a linear expansion of ψ_n:

$$g(k) = a_0\psi_0(k) + a_1\psi_1(k) + ... + a_n\psi_n(k) \tag{3.2}$$

Now we are interested in the expansion coefficients a_n, which give information about the content of ψ_n in $g(t)$. Suppose we have three sample points of our signal $g(t)$ and three discrete functions ψ_n. Than the linear expansion can be written out in matrix form as:

$$\begin{bmatrix} g(0) \\ g(1) \\ g(2) \end{bmatrix} = \begin{bmatrix} \psi_0(0) & \psi_1(0) & \psi_2(0) \\ \psi_0(1) & \psi_1(1) & \psi_2(1) \\ \psi_0(2) & \psi_1(2) & \psi_2(2) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \tag{3.3}$$

Both the sample points and the expansion functions ψ_n are known. So by taking the inverse it is possible to find the expansion coefficients a_n. Important to notice is that the inverse is independent of the signal itself. It only depends on ψ_n.

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \psi_0(0) & \psi_1(0) & \psi_2(0) \\ \psi_0(1) & \psi_1(1) & \psi_2(1) \\ \psi_0(2) & \psi_1(2) & \psi_2(2) \end{bmatrix}^{-1} \begin{bmatrix} g(0) \\ g(1) \\ g(2) \end{bmatrix} \tag{3.4}$$

Finding a well conditioned inverse of this matrix consisting of ψ_n makes it possible to calculate a_n directly. This is of course only possible if the inverse exists. When choosing a proper basis this is the most weak condition which needs to be fulfilled. Now let us consider a second set of functions $\tilde{\psi}_n$ called a dual-basis that is defined as follows:

$$\begin{bmatrix} \psi_0(0) & \psi_1(0) & \psi_2(0) \\ \psi_0(1) & \psi_1(1) & \psi_2(1) \\ \psi_0(2) & \psi_1(2) & \psi_2(2) \end{bmatrix}^{-1} = \begin{bmatrix} \tilde{\psi}_0(0) & \tilde{\psi}_0(1) & \tilde{\psi}_0(2) \\ \tilde{\psi}_1(0) & \tilde{\psi}_1(1) & \tilde{\psi}_1(2) \\ \tilde{\psi}_2(0) & \tilde{\psi}_2(1) & \tilde{\psi}_2(2) \end{bmatrix} \tag{3.5}$$

If this set of functions respectively vectors $\tilde{\psi}_n$ exists the expansion coefficients a_n can be directly be calculated. Of course this new set should also be a bases i.e. linear independent.

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \tilde{\psi}_0(0) & \tilde{\psi}_0(1) & \tilde{\psi}_0(2) \\ \tilde{\psi}_1(0) & \tilde{\psi}_1(1) & \tilde{\psi}_1(2) \\ \tilde{\psi}_2(0) & \tilde{\psi}_2(1) & \tilde{\psi}_2(2) \end{bmatrix} \begin{bmatrix} g(0) \\ g(1) \\ g(2) \end{bmatrix} \tag{3.6}$$

This can be rewritten to (3.7) that is nothing else than an inner-product,

$$a_n = g(0)\tilde{\psi}_n(0) + g(1)\tilde{\psi}_n(1) + ... + g(k)\tilde{\psi}_n(k) = \sum_k g(k)\tilde{\psi}_n(k) \tag{3.7}$$

Now by finding a set of vectors $\tilde{\psi}_n$ that is the inverse of ψ_n it is possible to decompose the signal.
This derivation is done for discrete functions (vectors) but it can be extended to continuous (complex) functions. It is more common to write down the relationships between these functions in Hilbert space, when working with functions instead of vectors . The inner-product of two functions in (Hilbert) function space is defined as:

$$\langle x(t), y(t) \rangle = \int x(t)y^*(t)dt, \tag{3.8}$$

where $*$ denotes the complex conjugate of a function. Rewriting (3.1) and (3.7) for continuous functions gives:

$$g(t) = \sum_n \langle g(t), \tilde{\psi}_n(t) \rangle \psi_n \tag{3.9}$$

The expansion coefficients are then calculated with:

$$a_n = \langle g(t), \tilde{\psi}_n(t) \rangle = \int g(t)\tilde{\psi}_n^*(t)dt \tag{3.10}$$

The sum in (3.7) has become an integral because we are dealing with continuous functions. The decomposition described in (3.9) uses a dual-basis and is called a biorthogonal system. This is because $\tilde{\psi}_n$ and ψ_n are not orthogonal to each other but to the expansion set itself, i.e.

for a set of vectors it is the inverse of this set. Consequently the product between them is the unity matrix, which is denoted by the Kronecker delta.

$$\langle \psi_n(t), \tilde{\psi}_l(t) \rangle = \delta\left(n-l\right) \tag{3.11}$$

The main goal is to find a set of functions $\tilde{\psi}_n(t)$ that is the inverse of $\psi_n(t)$ and fulfills (3.11). There is one trivial set of functions that will fulfill this condition: an orthogonal set of functions $\psi_n(t)$. Such a set of functions is called an orthogonal basis (vectors are perpendicular to each other), which means that the reconstruction wavelet is equal to the decomposition wavelet when normalized; orthogonality is generally denoted as,

$$\langle \psi_n(t), \tilde{\psi}_l(t) \rangle = C\delta_{n,l} \qquad \text{with} \qquad C = \text{Constant} \tag{3.12}$$

So far it is explained how to calculate the expansion coefficients from a signal through an example. However this example was to ideal because the number of samples equaled the number of expansion coefficients. Often the number of sample points is much bigger than the number of calculated expansion coefficients. In a discrete sense the number of vectors need to be extended to make it possible to calculate the expansion coefficients i.e. the matrices (3.6) become non-square matrices. Such an extended basis is called a frame and is an over-complete version of a basis. One can imagine that when calculating the dual-frame (pseudo-inverse), it is no longer unique. For a frame (also Riesz basis) following condition must hold:

$$A||g||^2 \leq |\langle \psi_n, g \rangle|^2 \leq B||g||^2 \qquad \text{with} \qquad ||g|| = \sqrt{\langle g, g \rangle} \tag{3.13}$$

Where A and B are two constants so that $0 < A \leq B < \infty$. If $A \neq B$ we have again a biorthogonal system, however, if $A = B$ we have an orthogonal system (frame). Then (3.13) reduces to Parseval's theorem:

$$g\left(t\right) = A^{-1} \sum_n \langle \psi_n\left(t\right), g\left(t\right) \rangle \psi_n\left(t\right) \tag{3.14}$$

i.e. when our frame is orthogonal,

$$\tilde{\psi}_n(t) = A^{-1}\psi_n(t) \qquad \text{or} \qquad \left(\psi_n(k)\right)^{-1} = \left(\psi_n(k)\right)^T \tag{3.15}$$

Frames that are orthogonal are called tight-frames, the complex vector of the Fourier transform is such a tight-frame. Despite this remark concerning frames and bases the continuous expansions (Hilbert spaces) apply for both bases and frames. Note that it is assumed that $g\left(t\right)$ is perfectly approximated, which is in real applications rarely true. Therefore this will be the subject of the next section.

3.2 Approximation of signals

In the last section it is assumed that our continuous signal $g(t)$ can be approximated perfectly by a_n and ψ_n. In reality this is often not true. In this section is described how to calculate the best approximation of a signal. Let us consider a real signal $g(t)$ and its approximation $\hat{g}(t)$ which can be perfectly approximated by a_n and ψ_n . The approximation error is defined as the difference between $g(t)$ and $\hat{g}(t)$. This results in an error function, which is used as a measure for the overall error. Therefore the sum of the remaining squared inner products is calculated (2-norm) and is used as a measure for the error (ϵ),

$$\epsilon[M] = ||g(t) - \hat{g}(t)||^2 = \sum_{n=M}^{+\infty} |\langle g(t), \psi_n(t) \rangle|^2 \qquad \text{with} \qquad \hat{g}(t) = \sum_{n=0}^{M-1} \langle g(t), \psi_n(t) \rangle \psi_n \tag{3.16}$$

Note that here indeed the 2-norm is used to define the error. It is of course possible to use other norms e.g. the 1-norm or ∞-norm. However it depends on the problem which norm is best suited. Here the 2-norm is chosen because it simplifies the derivation significantly and is

most commonly used. If the number of terms M is increased the error becomes smaller i.e. when M goes to $+\infty$ then the approximation error goes to zero. The rate that determines how fast $\epsilon[M]$ goes to zero with increasing M is called the decay rate and says something about how well a certain frame can approximate a signal. In practice it is rarely possible to approximate a signal perfectly. In that case the coefficients that minimize the error ϵ need to be found. This can be done by calculating the optimal problem as explained in [21]:

$$J = \int \left[g(t) - \sum_{n=0}^{N} a_n \, \psi_n(t) \right] dt \tag{3.17}$$

J should be minimized to find the best values for a_n,

$$\frac{\delta}{\delta a_l} J(a_0, a_1, ..., a_N) = 0; \; l = 0, 1, ..., N \tag{3.18}$$

$$\frac{\delta}{\delta a_l} J(a_0, a_1, ..., a_N) = -2 \int [g(t) - \sum_{n=0}^{N} a_n \psi_n(t)] \psi_l(t) dt \tag{3.19}$$

The minimum of the error ϵ is found when this equation is equal to zero.

$$0 = \int [g(t) - \sum_{n=0}^{N} a_n \psi_n(t)] \psi_l(t) dt \tag{3.20}$$

If we would use vectors instead of functions, this is exactly the least square method, which also finds the values that minimizes the error. This result is nice but the proof given here only holds when using the 2-norm and using an orthogonal basis. In [24] the same derivation is done for biorthogonal systems, where it is proven that for an uniform sampling grid the orthogonal projection (i.e. approximation) is still the best approximation.

$$g(t) \approx \sum_{n} \langle g(t), \tilde{\psi}_k(t) \rangle \psi_n \tag{3.21}$$

This means that the expansion a_n can be calculated by taking the inner-product between the dual basis/frame and the signal. As already mentioned the approximation greatly depends on the functions ψ_n. The decay rate of the error can often be directly related to the decay rate of the functions ψ_n (read wavelets). Therefore it is very beneficial to choose functions with a high decay rate to have a good approximation of the signal.
In these sections the basic theory and ideas behind bases, frames and approximation of signals are given, however it is only a short introduction. In case one wants to know much more about this subject in relation to wavelets [24] offers almost a full description.

3.3 The Fourier Transform

The Fourier Transform is based on the use of sinusoidal basis functions for signal approximation. There are different types of Fourier Transforms based on the type of time signal and the calculated frequency. They are listed in Table 3.1:

Table 3.1: Fourier Transforms

Fourier transforms	Discrete time (DT)	Continuous time (CT)
Discrete Frequency (DF)	Discrete Fourier Transform (DFT)	Fourier Series (FS)
Continuous Frequency (CF)	Discrete Time Fourier Transform (DTFT)	Continuous Fourier Transform (CFT)

The DTFT and CFT are not often used and have no relevance for wavelets at all. Therefore they are not discussed in this report. More information about these transforms can be found in [29, 39]. The Fourier Transform used in practice is the DFT, simply because computers can only work with discrete variables. However most ideas and background ideas are derived

through the Fourier Series (FS). This is useful as long as the discrete time signal has a high enough sample rate so that a discrete time signal is a good representation of the considered continuous time signal. In the previous sections the general form of the Fourier Series is given. Only the choice of the functions ψ_n needs to be specified. In the case of the Fourier transform $\psi_n = e^{in\frac{2\pi}{T}t}$, which is related to the harmonic functions: $e^{it} = \cos(t) + i\sin(t)$. The signal $g(t)$ can than be approximated as:

$$g(t) \approx \sum_{n=-\infty}^{n=\infty} a_n e^{in\frac{2\pi}{T}t} \tag{3.22}$$

This is called the synthesis of the function or inverse Fourier Series. Although this function $g(t)$ is defined as an infinite sum here, often it suffices to take a finite number of coefficients. To find the Fourier Series the dual-basis needs to be found. The basis $\psi_n = e^{-ik\frac{2\pi}{T}t}$ is orthogonal so by taking the conjugate transpose of this basis the analysis basis is found ($\tilde{\psi}_n = \frac{1}{T}e^{-in\frac{2\pi}{T}t}$) i.e. the coefficients in the Fourier Series are defined as:

$$a_n = \frac{1}{T}\int_{t_0}^{t_0+T} g(t)e^{-in\frac{2\pi}{T}t}dt \tag{3.23}$$

Often the coefficients a_n are denoted as $X_{FS}(k)$ in the context of Fourier transforms. The vector $e^{in\frac{2\pi}{T}t}$ is periodic and extends from minus to plus infinity. In this context it is assumed that the signal to be analyzed $g(t)$ is also periodic and extends from minus to plus infinity. Therefore it is only necessary to calculate the transform over one period T. The practical application of the Fourier Transform is the DFT and is defined as,

$$a_n = \frac{1}{N}\sum_{n=0}^{N-1} g(k)e^{-in\frac{2\pi}{N}k}dt \quad n = 0, ..., N-1, \tag{3.24}$$

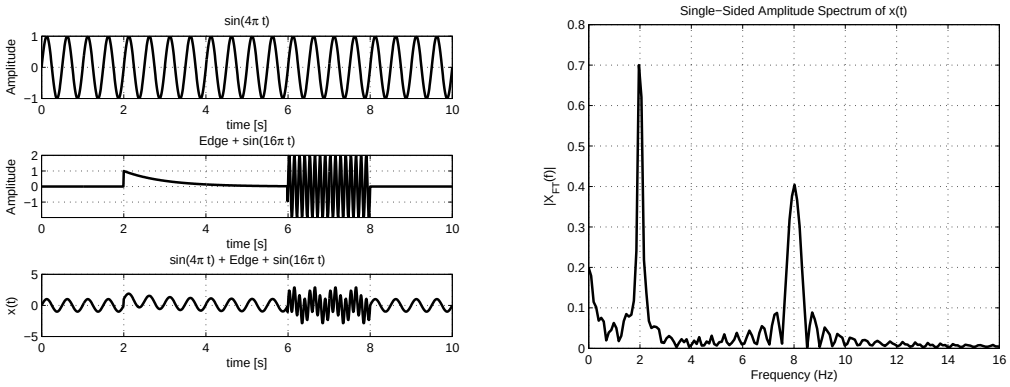
and the inverse transform as:

$$g(k) \approx \sum_{n=0}^{N-1} a_n e^{in\frac{2\pi}{N}k} \qquad k = 0, ..., N-1 \tag{3.25}$$

Both the Fourier Transform and inverse Fourier transform are now in a discrete form and periodic with period length N. Even when the discrete approximation of the signal $g(k)$ is finite and not periodic. It is possible to treat the signal as a periodic signal by applying a window. This window ensures smooth transition from one period to the other. This is necessary to prevent sudden transitions which will introduce frequencies that are not present in the signal. This effect is called signal leakage. As an example of a DFT, consider a signal consisting of one sinus (2 Hz) that extends over the entire signal length, a sinus (8 Hz) only present between 6 and 8 seconds and a one sided edge (Fig. 3.1(a)). The resulting Magnitude of the Discrete Fourier Transform is presented in Fig. 3.1(b).
In the resulting magnitude plot (Fig. 3.1(b)) the frequencies of the two sinusoids are clearly visible. Nevertheless it is difficult to distinguish the edge (0 Hz) and the time when the local sinusoid occurs. The Fourier Transform only gives information about the frequency content of a signal. Consequently one needs other methods to distinguish the edge and the location of the other sinus. One of these methods could be the STFT which is described in the next section.

3.4 The Short Time Fourier Transform

The Short Time Fourier Transform (STFT) is the natural extension to the Fourier Transform. The Fourier Transform only returns information of the frequency content of a signal. However when one is also interested in a more (time) localized frequency content an extension needs to be made, this is the STFT. It is nothing else than the Fourier transform combined with a window $w(t-\tau)$ that moves over the signal.



(a) (b)

Figure 3.1: Signal and its Magnitude

$$X_{STFT}(\tau, f) = \int_{\infty}^{\infty} g(t)\, w^*(t-\tau)\, e^{-j\frac{2\pi}{T}t} dt \tag{3.26}$$

The STFT gives only frequency information about the part of the signal that is visible through the window. Therefore the choice of the size of the window is important. A small window gives a good time resulution but a not so good frequency resolution. Conversely a big window gives good frequency information but less information about the location. In Fig. 3.2 an example is given to illustrate the STFT. The signal used is presented in Fig. 3.1(a).

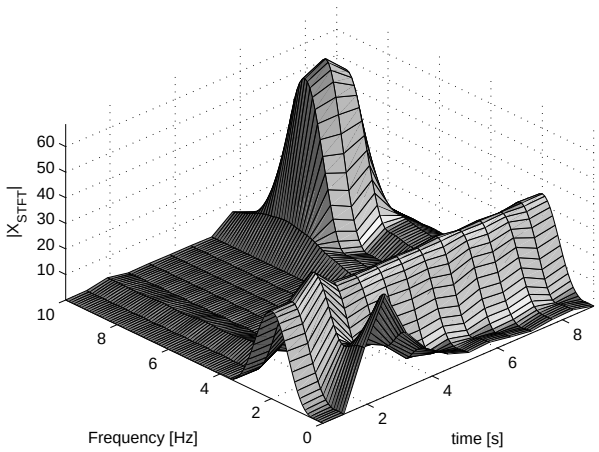


Figure 3.2: The Short Time Fourier Transform of the signal presented in Fig. 3.1(a)

The sinusoids can be clearly distinguished in Fig. 3.2, including the time where the edge is present, although it is difficult to distinguish it as an edge. In Fig. 3.2 the window has already been optimized to give a good trade-off between frequency and time. This trade-off is defined through the Heisenberg Uncertainty Principle [24]. It defines what the ratio is between the time and frequency resolution. The term uncertainty comes from the fact that if you are sure on which point in time an event occurs, you are unsure about the exact frequency of the same event. This means that it is not possible to achieve maximum resolution in both time and frequency simultaneously. The Fourier transform has excellent localization in the frequency domain, but it extends over all time giving no information about when which frequency occurs.

Even with the help of the STFT and even wavelets it is still impossible to determine the exact time or the exact frequency an event occurs. It has been proven that there is a constant lower bound on the ratio between the time and frequency resolution [32]. This is calculated by using the standard deviations Δt and Δf i.e. the r.m.s. duration and r.m.s. bandwidth,

$$\Delta t = \sqrt{\frac{\int t^2 \left|g(t)\right|^2 dt}{\int \left|g(t)\right|^2 dt}}, \qquad \Delta f = \sqrt{\frac{\int t^2 \left|G(f)\right|^2 dt}{\int \left|G(f)\right|^2 dt}} \tag{3.27}$$

The lower bound on the product is than:

$$\Delta f \Delta t \geq \frac{1}{4\pi} \tag{3.28}$$

This is only a lower bound on the resolution and for a lot of wavelets this product is much worse. In [31] a list of wavelets is given with their time resolution and frequency resolution. The Heisenberg principle is often not of practical use but the trade-off between resolutions is always present.

3.5 Conclusions and considerations

In this chapter the idea behind signal analysis is explained and the general problems related to this. The Fourier Transform is the most important tool in signal analysis. The STFT combines the Fourier Transform with a window, to give information about frequency content of a signal in time. From here it is only a small step towards the Morlet wavelet. The Morlet wavelet simply combines a window with the complex vector used in the Fourier Transform. Combining a window and the analyzing function in one function is the main idea behind wavelets. Note that a signal is usually split up in sub-functions. The information acquired depends on the choice of sub-functions. The sub-functions not only decide what information is extracted but also how well a signal can be approximated using a certain sub-function. The set of sub-functions is called a basis or frame depending on the transformation, which is discussed in Section 3.1. In discrete sense a basis (also a frame) is related to a square matrix and a frame is related to a non-square matrix. For all transforms it holds in general that the inner-product is the best approximation although there are some exceptions for biorthogonal systems. Frames are the starting point to every signal analysis transform. In this chapter also the most used frame has been introduced, the Fourier Analysis. In addition the example described the limits of the Fourier Transform clearly. It only gives frequency information and no information when certain events or frequencies are present. Therefore an extended Fourier Transform is needed in the form of the STFT, which combines the Fourier Transform with a window. Now information about time and frequency can be measured, although it is still quite limiting. The next extension is the wavelet transform that often gives a better time and frequency-resolution. Wavelets in general and the Continuous Wavelet Transform will be subject of the next chapter.

Chapter 4

Introduction to Wavelets

In the preceding chapter the idea behind signal analysis in general is described and the most important techniques are explained. The Short Time Fourier Transform already gave a first idea about how to combine time-resolution and frequency-resolution. Wavelets are the expansion of this; where at first they are still quite close to the idea of Fourier Transforms and windowing (Morlet). The group wavelets is expanded to a whole new group of functions that give different time localized information. In this chapter a small introduction is given on the Continuous Wavelet Transform (CWT). The basic concept and necessary conditions of wavelets are explained. Finally two examples are presented.

4.1 Types of Wavelet Transforms

There exists a number of different wavelet transforms, which correspond to the different types of Fourier Transforms. The main difference with Fourier transforms is that all wavelet transforms are aperiodic functions and consequently their outputs are also aperiodic. This is a logical consequence of their localization in time. Although there also exists a periodic wavelet transform, it is an extension on wavelets and not part of wavelet transforms itself.

Table 4.1: Different types of Wavelet Transforms

Wavelet Transforms	Discrete Time (DT)	Continuous Time (CT)
Discrete Wavelet (DW)	Discrete Time Wavelet Transform (DTWT)	Discrete Wavelet Transform (DWT)
Continuous Wavelet (CW)	Discrete Time Continuous WT (DTCWT)	Continuous Wavelet Transform (CWT)

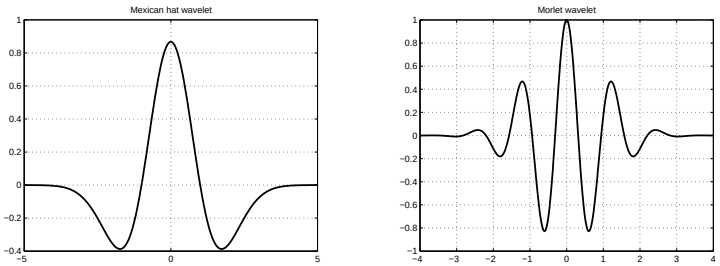
Calculations are often done in a computer setting. Therefore the only transform that can be used in practice is the DTWT, where both time and the wavelet coefficients, which is the output of the wavelet transform are discrete. Strange enough this type of transform is rarely described. This is because the DWT is used as a theoretical basis for deriving the wavelets and their properties. The DWT differs only in two properties from the Fourier Series: it is not periodic and the basis functions used are different. They both deal with continuous time and the analyzing and synthesis functions are both continuous. In addition the resulting expansion coefficients are discrete. Using the DWT instead of the DTWT is fine as long as the discrete time signal has a high enough sample rate so that it is a good representation of the considered continuous time signal. There are many properties, which are defined in the DWT that have no meaning in the context of DTWT, e.g. vanishing moments (6.7). The connection between the DTWT and DWT is:

- Very fine scale; expansion coefficients of the DWT are (almost) the same for the DTWT
- Large number of scales; the DTWT basis vectors converge to the DWT

So the DWT is simply discretized to arrive at the DTWT. However in the case of CWT this is much more complex and the use of for instance approximation methods is necessary to calculate its transform e.g. by using a Riemann sum. Despite this fact the CWT is still a valuable tool and is generally used as an introduction for the DWT. Therefore the next section will explain the working of a wavelet transformation specifically based on the CWT.

4.2 The Continuous Wavelet Transform

The name wavelet (ψ) means small wave and this is indeed what a wavelet is. This "small wave" will be compared to a signal to see if parts of it matches the shape of the wavelet and thereby extracting relevant information about the signal. This is not only done for different translations but also for dilations of the wavelet. This results in information about the signal for different frequencies (scales) and temporal information. This is very powerful from a signal analysis point of view because it gives information about events in time and frequency. This important property of wavelets is called multiresolution. Two examples of continuous wavelets are given in Fig. 4.1.



(a) (b)

Figure 4.1: Two mother wavelets (a) Mexican Hat wavelet (b) Morlet wavelet

These two wavelets have totally different purposes, the Mexican Hat wavelet is used for edge detection whereas the Morlet wavelet is used to locally determine the frequency. Although most wavelets are quite different, they are required to fulfill two criteria to be called a wavelet. For instance a wavelet should have zero average,

$$\int_{-\infty}^{\infty} \psi(t)\, dt = 0 \tag{4.1}$$

The initial wavelet is called the mother wavelet and this mother wavelet is dilated with a scale parameter s and is translated with parameter τ,

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{|s|}} \psi\left(\frac{t-\tau}{s}\right) \tag{4.2}$$

Dilating wavelets makes it necessary to normalize the energy for different scales. This normalization is done by dividing the wavelet coefficients by $1/\sqrt{|s|}$. The goal is to compare the signal to a wavelet and calculate how well they match each other. The wavelet is translated (τ) and compared to the signal to differentiate between location (time). Also it is dilated (s) so it can differentiate between scales or frequency. In wavelets the term scale is preferred because not all wavelets are related to frequency in the same sense as the Fourier transform of a signal. However there are a number of wavelets that are related: in that case high scale means low frequency and small scale means high frequency. Scales can be transformed back to the frequency using the center frequency of the wavelet. The center frequency is the position of the maximum modulus of the wavelet in the frequency domain. However, often the meaning of frequency is lost because we are searching for different features, unrelated to frequency. The

analysis at different scales and locations is done by convolving your signal with the wavelet. Convolution expresses the amount of overlap of one function (wavelet) as it is shifted over another function (signal), which process is described in the previous chapter as analysis of a function ($\tilde{\psi}$). Therefore the continuous wavelet transform is defined as:

$$X_{WT}(s,\tau) = \langle x(t), {}^{1}\tilde{\psi}_{s,\tau}(t)\rangle = \int_{-\infty}^{\infty} x(t)\,\frac{1}{\sqrt{|s|}}\tilde{\psi}^{*}\left(\frac{t-\tau}{s}\right)dt \tag{4.3}$$

The result of a wavelet transform for an one-dimensional signal results in a two-dimensional function depending on the location τ and scale s. In Fig. 4.2 the working principle of a wavelet transform is explained.

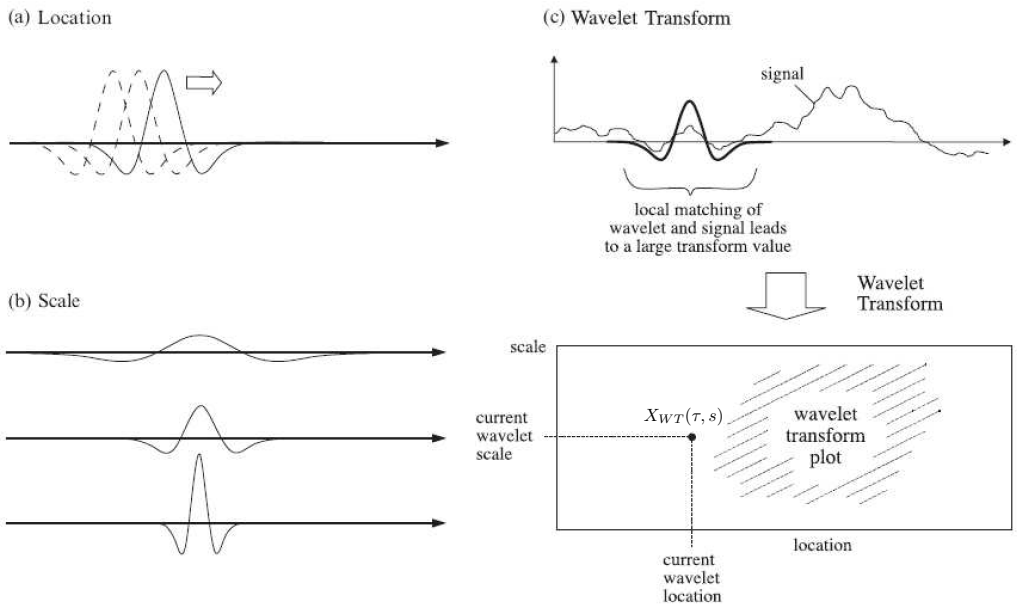


Figure 4.2: Representation of the (continuous) wavelet transform [3]

A wavelet $\psi(t)$ is a small wave that oscillates between defined boundaries. It is zero outside these boundaries and therefore the wavelet acts as a window and analyzing function at once. There is great freedom in choosing the analyzing functions (wavelets).
There are only two mathematical criteria that a function $\psi(t)$ respectively $\tilde{\psi}(t)$ has to fulfill to be called a wavelet:

1. A wavelet must have finite energy:

$$E = \int_{-\infty}^{\infty} |\psi(t)|^{2}\,dt < \infty \tag{4.4}$$

2. The admissibility condition:

$$C_{\psi} = \int_{0}^{\infty} \frac{|\Psi(f)|^{2}}{|f|}df < \infty \tag{4.5}$$

[1] In this report the analysis function is denoted by $\tilde{\psi}$ because the reasoning behind wavelets is explained from the viewpoint of decomposing a signal in sub-signals ψ (synthesis of a function), which is more comprehensive from a DWT perspective. However when is reasoned from a CWT point of view, the synthesis part or decomposition of signals in sub-signals has no meaning. Therefore the notation is reversed in many books, there the analysis function is denoted by ψ and the synthesis by $\tilde{\psi}$ i.e. instead of denoting it as in (3.1), it is denoted as $g(t) = \sum_n a_n\,\tilde{\psi}_n(t)$

The finite energy makes sure that the wavelet is square integrable, this is necessary to be sure that the inner-product (wavelet transform) between the signal and wavelet exist.
The admissibility condition on the other hand guarantees that the original function can be reconstructed again after the decomposition. The derivation of this proof is described in [13]. In (4.5) $\Psi(f)$ is the Fourier Transform of $\psi(t)$. The admissibility condition implies that $\psi(t)$ has no zero frequency component, $\Psi(0) = 0$ i.e. the mean of $\psi(t)$ is zero as was already stated earlier. For Complex wavelets $\Psi(f)$ should not only vanish for real frequencies but also for negative frequencies.
The inverse wavelet transform is defined as:

$$x(t) = \frac{1}{C_{\tilde{\psi}}{}^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} X_{WT}(\tau, s) \frac{1}{s^2} \tilde{\psi}\left(\frac{t-\tau}{s}\right) d\tau ds \tag{4.6}$$

Here also the admissibility condition reappears indeed since if $C_{\tilde{\psi}} = \infty$ the resulting signal $x(t)$ would be zero.
Comparing the CWT carefully to the method of decomposition explained in Section 3.1, it becomes clear that in the strict sense the CWT is not really a decomposition of a signal. The purpose of the CWT is to extract information by convolution and not to decompose the signal into sub-signals. Therefore the reconstruction frame is of less importance. The reconstruction frame is often numerically problematic (non-orthogonal) because of the inverse. Although there are certain scale bounds where the reconstruction frame is still well defined. The problem of reconstruction is also implied by (4.6) where a double infinite sum is necessary to calculate the inverse transform. Considering all this it is easy to understand why the inverse transform is rarely calculated.

4.3 Morlet example of a Continuous Wavelet Transform

In the previous section the CWT is introduced. In this section two examples of a CWT are presented. For this purpose the signal shown in Fig. 3.1(a) is used again. This signal consisted of a local sinus at 8 Hz, an edge (at $t = 2$ s) and one global sinus at 4 Hz. The wavelet transform will be done with the Morlet wavelet Fig. 4.1(b). The Morlet is the most basic wavelet, it combines an harmonic oscillation with a gaussian window. It is ideally suited to detect local frequencies in signals. Using the definition in (4.3) the wavelet transform is calculated. Although the signal is discrete and consequently the resulting transform is DTWT, calculating it at the sample points can be considered as a CWT. The resulting coefficients $X_{WT}(s, \tau)$ are presented in Fig. 4.3.

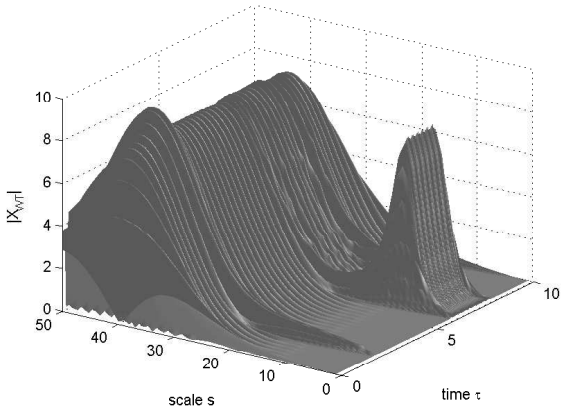


Figure 4.3: Coefficients $X_{WT}(s, \tau)$ of the Morlet Wavelet and Signal

In the three dimensional presentation of the wavelet coefficients it clearly can be seen that there are two frequencies that can be separated. One frequency has local a peak at scale 10

and one globally at about scale 40. Also around two seconds there is a small local disturbance. In general these CWT plots are presented as a contour plot (height map). The contour plot is shown in Fig. 4.4.

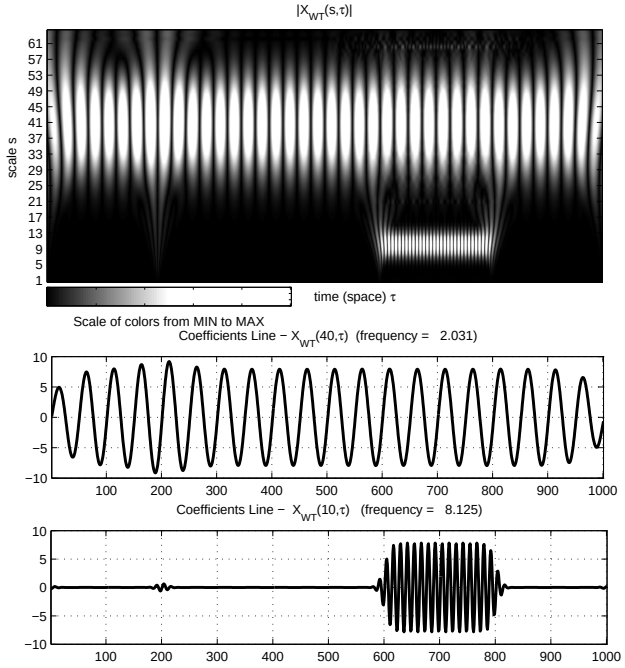


Figure 4.4: Contour plot of the wavelet coefficients $X_{WT}(s, \tau)$ and at two local scales

The contour plot contains exactly the same information as Fig. 4.3. The two different regimes (different sinusoids) can be easily distinguished, as some disturbance where the edge is located. It is possible to gain more information by plotting the transform at a certain scale. It was already concluded that important information is present at scales 10 and 40, which are also shown in Fig. 4.4. These scales s can be transformed to frequencies f using the center frequency f_c and sample time T_{st}:

$$f = \frac{f_c}{T_{st} s} \tag{4.7}$$

In our case $f_c = 0.8125$ and $T_{st} = 0.01$ s it follows that the corresponding frequencies are approximately 2 Hz and 8 Hz. The edge can also be seen, however, from the contour plot it is difficult to distinguish it as an edge. Furthermore the amplitudes could also be calculated using the scale factor in front of the wavelet transform. When analyzing the wavelet transform around the presumed frequencies the wavelet coefficients are still high. This is because around the real sinusoids the wavelet still fits quite well on the signal.

4.4 Mexican hat example of a Continuous Wavelet Transform

Now let us consider the case where we are not interested in frequencies or amplitudes but in finding the edge itself. In this case a wavelet should be used that is fitted for this task. A Mexican Hat is such a continuous wavelet used for edge detection. The Mexican Hat wavelet is the negative normalized second derivative of a Gaussian function. The CWT of the signal with the Mexican Hat is shown in Fig. 4.5.
On the slope side of the edge ($t < 2$ s), the negative part of the wavelet will match relatively well, leading to big negative wavelet coefficients. When the wavelet shifts further the positive

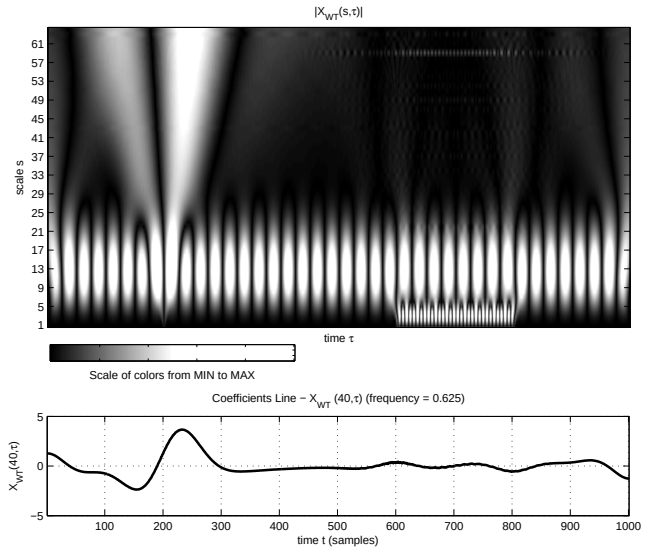


Figure 4.5: Mexican Hat contour plot and coefficients at scale 40

part of the Mexican Hat will match well leading to big positive values. In between there is a singularity, this is the exact position of the edge. The wavelet transform is done on different scales (higher scales, wider wavelets) leading to a singularity line with broadened negative peaks at the left side of the edge and positive at the right side. These become bigger with higher scales. In Fig. 4.5 the edge location is clearly visible. At lower scales a lot of false edges can be identified. This is because a sinus also acts as an edge. The relationship between scale and frequency is different from Fig. 4.4 to Fig. 4.5 because the used wavelets have different localization in time and frequency i.e. different central frequencies ($f_c = 0.25$). Especially in the case of the Mexican Hat it is difficult to speak of frequency.
In general the CWT is considered redundant. There is a much more efficient method to compute wavelet transforms, which is the Discrete Wavelet Transform (DWT). It is not only more efficient, it is well defined in the sense of reconstructing signals as well. The DWT comes with its own classes of wavelets which have other special properties compared to the CWT. However, continuous wavelets are still often used but they need to be transformed into a discrete setting. In the next chapter the discrete wavelet transform is introduced.

4.5 Conclusions and considerations

In this chapter it is explained what a wavelet transform is. Although this chapter focuses on the CWT, reading and understanding the results will hold for all wavelet transforms. In case of the DWT, which was not discussed here, the results will be similarly but more rough. Continuous wavelets have the most broad applications compared to DWT. They are still used but applied in a more discrete setting. This makes them not very efficient but for non real-time applications still very useful. As we have learned the result of a CWT gives information as function of scale and location. When interpreting these results, first ask yourself what kind of information is generated by the WT. In case one wants to determine the localized frequencies it is very clear what to search for (notice the similarity with the STFT). In case one looks for features as for instance an edge it needs to be present in a number of scales. The number of scales this feature is present in, gives information about how big this edge is and the singularity gives information about its location. It is important to remember that different wavelets show different features and their WT need to be interpreted differently. There are for instance different edge detectors (wavelets) that present edges in different ways.

Chapter 5

The Discrete Wavelet Transform and Filter Banks

A Discrete Wavelet Transform (DWT) is very different from the CWT especially in how it is applied. In practice the DWT is almost always applied in the form of a filter bank. A filter bank consists of a great number of filters either FIR-filters or IIR-filters (although the latter is rarely used). A FIR-filter is a transfer function with all poles in the origin whereas a IIR-filter is a full transfer function. The FIR-filters or IIR-filters are combined with each other in a smart way using something that is called downsampling to delete redundant information. Filter banks are very efficient in calculating the wavelet coefficients of the DWT especially when the signal also needs to be reconstructed.

The analysis on different scales and locations (multiresolution) is key to wavelet transforms and this holds of course also for the DWT. Therefore in the first section, multiresolution for the DWT is shortly introduced. Then the discrete wavelet transform is explained through the scaling and wavelet function using an example. The DWT cannot be used in practice because it is only defined for continuous time signals and continuous wavelets. Therefore in practice the Discrete Time Wavelet Transform (DTWT) is needed. This is done by making the time discrete in the DWT. This is equivalent to using a wavelet based filter bank. A special operation called downsampling is needed in discrete time to calculate the DWT in discrete time. Filter banks as wavelets come in many different forms thus a number of different filter banks are explained. The most important property of DWT is perfect reconstruction. This is elaborated afterwards and finally the filter bank that is used in practice is explained.

5.1 Multiresolution

The idea of multiresolution is to decompose a single event on different scales and to study the event on these different scales. In the case of the DWT this means: first study the event in detail (small scale or high frequency) and by moving through every step of the DWT, study it on a less detailed level. The DWT starts with comparing the signal with small wavelets, which means that scale s is small. Consequently because the wavelet is small, only high frequent behavior can be studied i.e. small scale means high frequency. The formal definition of multiresolution analysis (MRA) is described in almost every book [32, 11, 8]. These definitions of MRA slightly differ but their content remains almost the same. They have a highly mathematical background and are necessary to get a full understanding of DWT and multiresolution. Therefore a more detailed explanation of MRA of the DWT is explained in Chapter 6.1. The DWT (but also a CWT) is applied on a grid of time and scale (frequency) and this is called multiresolution.

First we want to give an idea about the DWT. As was explained in Section 3.1 a signal can be decomposed and written down as a linear expansion of functions and coefficients. However, instead of only having one expansion over all scales as with the Fourier Transform, we have an expansion in two directions, the scale m and translation n, i.e.

$$x(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} d_m(n)\,\psi(2^m t - n) = \sum_{m,n} d_{m,n}\psi_{m,n}(t) \tag{5.1}$$

where $d(m,n)$ are called the wavelet coefficients, ψ denotes the wavelet and $x(t)$ is the signal to be analyzed. The signal $x(t)$ is still exactly approximated because the sum is still infinite. The coefficients that describe the location (translation) are dependent on n. The scales are regulated by m using a dyadic grid 2^m. The numbering of the scales is somewhat arbitrary and inconsistent in the literature: where most classic books use 2^{-m} to number their scales, most modern books use the counting used in this report. This is probably because the classic books use the CWT as a reference, where most new books see the DWT and filter banks as something separately and more related to the frequency domain. The scale numbering used in this report is presented in Fig. 5.1.

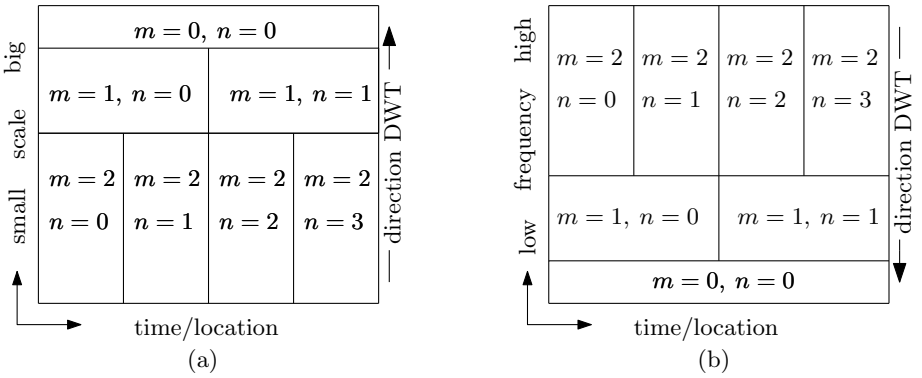


Figure 5.1: Numbering of the scales and translations (a) in scales (b) in frequency

Using (5.1) as a way to decompose a signal is unpractical because the whole signal needs to be decomposed from the smallest scale to the highest scale, in continuous time this means from $m = -\infty$ to $m = \infty$. Instead we analyze only a limited number of scales and the rest is stored by using another function called the scaling function. In principle a wavelet ψ always comes with a scaling function ϕ. Exactly like the wavelet, a scaling function is translated and dilated over the signal. The coefficients belonging to this ϕ are called the expansion coefficients $a_{m,n}$ ($a_m(n)$) and are together with the wavelet coefficients $d_{m,n}$ ($d_m(n)$) defined in a linear signal decomposition as:

$$x(t) = \sum_{n=-\infty}^{\infty} a_0(n)\,\phi(t-n) + \sum_{n=-\infty}^{\infty} \sum_{m=0}^{\infty} d_m(n)\,\psi(2^m t - n) \tag{5.2}$$

The scale at which $m = 0$ is the scale where the wavelet has the biggest scale or in continuous sense where the wavelet is stretched out maximally (not covering the entire signal length!). All wavelets that are smaller (higher frequency) or have the same size are defined from $m = 0$ till $m = \infty$. All wavelet coefficients that are normally calculated for $m < 0$ are now replaced by ϕ and $a_0(n)$. This is not a revolution yet because it is now still necessary to calculate all wavelet dilations and coefficients for $m > 0$. Although the DWT holds for continuous signals one can imagine when taking the step to discrete time the lowest scale (highest frequency) that can be analyzed is limited by the sampling frequency. Therefore using the scaling function ϕ in a discrete setting it is possible to decompose a discrete signal with only a limited number of expansion coefficients and wavelet coefficients.

5.2 Scaling function ϕ

The scaling function is the complement of the wavelet and for that reason it is often called the father wavelet. In the last section it was already explained that the scaling function ϕ helps to

limit the number of scales that need to be calculated, to prevent loss of information. This can be very useful when the signal needs to be reconstructed. The scaling function has a second purpose in the DWT: it handles the transfer between scales creating a less detailed signal that can than again be used for wavelet analysis.
In the previous section the full decomposition of the entire signal has been shown by expansion (at the highest scale $m = 0$). It is also possible to express a signal solely as a linear combination of scaling functions. Normally it is only necessary to do this at the lowest scale ($m = 2$ in Fig. 5.1) to initialize the DWT. However it is possible to do this also for every desired scale.

$$x(t) \approx \sum_{n=0}^{N} a_m(n)\,\phi(2^m t - n) \tag{5.3}$$

The number of coefficients N needed to represent the signal depends on the length of the signal (in practice number of sample points) and dilation of ϕ. This is nothing else than a linear decomposition using only the scaling functions. The relation between scales of the scaling function is determined by the dilation equation (synthesis and analysis).

$$\phi(t) = \sqrt{2}\sum_{n=0}^{N_1} l(n)\,\phi(2t - n) \qquad \text{and} \qquad \tilde{\phi}(t) = \sqrt{2}\sum_{n=0}^{N_1} l'(n)\,\tilde{\phi}(2t - n) \tag{5.4}$$

It states that a certain scale can be expressed in translated scaling functions of the scale below. Both the synthesis function and analysis function of the dilation equation are presented above. However, because these relationships are the same with the only difference in the filters $l(n)$ and $l'(n)$. Only the synthesis relationships will be given from here on. Relations (5.4) also called the two-scale relation. In the dilation equation $l(n)$ are the coefficients that handle the scaling function between scales (N_1 depends on the length of $l(n)$). Adjusting one of the coefficients leads to a different scaling function, hence $l(n)$ defines the scaling function. The coefficients $l(n)$ can also be seen as the filter coefficients of a discrete filter. This can be derived by filling (5.4) into (5.3). Therefore it is also necessary to apply the z-transform. A scaling function is similar to a wavelet also dilated and as a consequence it is also a dilated version of itself. Therefore if one scaling function (father wavelet) is known the rest can be calculated using:

$$\phi_{m,n}(t) = 2^{\frac{m}{2}}\,\phi(2^m t - n) \tag{5.5}$$

Although up till now all wavelets and thus also the scaling functions are known, this is slightly different in the DWT. Often only the coefficients $l(n)$ are known and they generate the scaling function by using (5.4). Nevertheless there are a number of predefined scaling functions, which corresponding wavelets are predefined.
Two examples of such scaling functions are the box scaling function and the triangle scaling function. The box scaling function is the scaling function belonging to the Haar Wavelet. The triangle scaling function looks like a triangle hence its name. Here the analysis scaling function $\tilde{\phi}$ is introduced and later on the analysis wavelet $\tilde{\psi}$ will be presented. Here we are explicitly dealing with the analysis of a function. Therefore generally all analysis functions should have a tilde. However the Haar scaling and wavelet function are orthogonal to their synthesis functions and are therefore the same. This is in contrast with the Triangle scaling and wavelet function although orthogonal to each other they are not to themselves. Therefore it is necessary to use the notation $\tilde{\phi}$ respectively $\tilde{\psi}$. The scaling functions of the Triangle and Haar wavelet are both presented in Fig. 5.2 with their translates of the scale above.
The original scaling function is chosen such that it has an area equal to one $\int_{-\infty}^{\infty}\phi(t)\,dt = 1$. Summing up the area of all scaled versions of the scaling function at one level should result in the area of the original scaling function again. The second normalization is that the scaling filter coefficients are chosen such that the sum is equal to $\sqrt{2}$.

$$\sum_{n} l(n) = \sqrt{2} \tag{5.6}$$

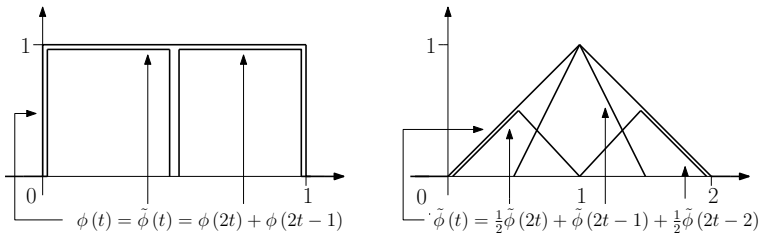


Figure 5.2: Two examples of scaling functions: Haar scaling function and triangle scaling function

A factor $\sqrt{2}$ stands in front of (5.4) so that the areas of the sum of all scaled functions are indeed one. This normalization of $\sqrt{2}$ is unnecessary when analyzing the coefficients of every scale separately. Only when comparing different scales with each other this factor needs to be taken into account. This normalization is generally applied for orthogonal scaling functions because the 2-norm will remain the same when a vector is multiplied with an orthonormal matrix, maintaining the vector length. In other wavelet systems, where this condition does not hold, other normalizations could be necessary to make it more easy to compare different scales, for example a different normalization factor for every scale.
Now with the knowledge gathered in this section it is possible to calculate the filter coefficients of the scaling functions presented in Fig. 5.2. The Haar scaling function has coefficients $l(n) = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$ and the triangle scaling filter coefficients are $l(n) = \left[\frac{1}{2\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{2\sqrt{2}}\right]$, which are derived from Fig. 5.2(below). These coefficients can also be seen as filter coefficients thus it is possible to calculate what the filter responses are using $l(n)$ (5.9).
All information of scales $m < 0$ is stored in $a_m(n)$, this becomes clear when looking at Fig. 5.1(b) because only low frequent information is present at $m < 0$. In other words the coefficients $l(n)$ are always the coefficients of a low-pass filter. Low-pass filtering is of course also some kind of smoothing operation. This becomes more clear when $l(n)$ is considered, in case of the Haar scaling function simply the mean of two samples is calculated (Section 5.4). Therefore $l(n)$ is also a smoothing operation. In summary the scaling function has several purposes:

- Its filter $l(n)$ is a low-pass filter
- Its filter $l(n)$ acts like an averaging filter
- It stores the information on the scales above the highest wavelet scale calculated
- It is a fingerprint for the corresponding wavelet

The relationship between wavelets and the scaling function has not been discussed yet. Therefore the next section will explain the relationship between the scaling function and the wavelet. We will for instance see that the scaling function is used to construct the wavelet.

5.3 Wavelets in the Discrete Wavelet Transform

In the continuous wavelet transform the wavelet is often a predefined function. Whereas in the discrete wavelet transform as we have seen, it is not. It is constructed during every step of the discrete wavelet transform from the scaling function. In general a scaling function is first constructed and than the wavelet that belongs to this scaling function is constructed. This sounds strange and it contradicts the natural feeling that we have from the CWT. The scaling function is the complement of the wavelet and when certain properties are given to the scaling function, it can be predicted what the properties of the wavelet will be. A wavelet is constructed from the scaling function as follows:

$$\psi(t) = \sqrt{2} \sum_{n=0}^{N} h(n)\, \phi(2t - n) \tag{5.7}$$

This equation is very similar to (5.4), but now the coefficients $h(n)$ instead of $l(n)$ are used. The result is not the scaling function ϕ but the wavelet ψ expressed in terms of ϕ at one scale below. The wavelets are also scaled and translated versions of themselves, therefore all wavelets can be constructed by using:

$$\psi_{m,n}(t) = 2^{\frac{m}{2}} \psi(2^m t - n) \tag{5.8}$$

The wavelets can be determined from the scaling functions presented in Fig. 5.2. However from (5.4) it is also known that the scaling function can be derived only knowing the coefficients of $l(n)$. When the scaling function is known either the wavelet needs to be known or its filter coefficients $h(n)$. In general there exist two basic methods for the construction of DWT wavelets: the first is based on known continuous functions that are adapted to fit into the DWT e.g. by approximating them with spline functions. The second method is based on the direct design of filters e.g. orthogonal filter banks. In Section 5.7 this method is explained in detail, more about the design from continuous functions can be found in [24]. In addition in Appendix A the design method for the most famous class of filter based wavelets is explained. Both the triangle and Haar wavelet are chosen such that their filters are quadrature mirror filters. Therefore they can be calculated from $l(n)$ with $h(n) = (-1)^{(}1-n)l(1-n)$ (6.11). The filter coefficients belonging to these wavelets are $h(n) = \left[\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right]$ and respectively $h(n) = \left[-\frac{1}{2\sqrt{2}}, \frac{1}{\sqrt{2}}, -\frac{1}{2\sqrt{2}}\right]$. The wavelets belonging to the scaling functions from Fig. 5.2 are shown in Fig. 5.3.

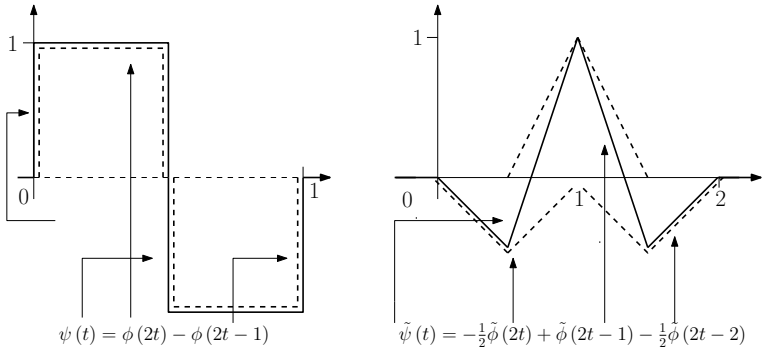


Figure 5.3: Haar and triangle wavelet

Wavelets are generated from the scaling function, this means that a wavelet is always a sum of shifted versions (multiplied by a factor) of a scaled ϕ. Therefore it is understandable that a wavelet generated from a scaling function does not have to be unique, one scaling function can generate many different wavelets and only by imposing extra conditions can a wavelet be related uniquely to its scaling function. One trivial example is the Haar wavelet with filter coefficients $h(n) = \left[-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$, this is the negative solution of (6.11). There are different properties that can be imposed on wavelets limiting the freedom of design e.g. orthogonality and minimum phase filters. As a consequence the degree of freedom is limited further often leaving only a straightforward choice of wavelet coefficients. Important to remember is that the wavelet is the useful part of the transform whereas the scaling function makes the transform efficient. The wavelet is compared to the signal, this leads to a number of wavelet coefficients that give insight into the analyzed signal. The exact meaning of the wavelet coefficients depends on the chosen wavelet or better the chosen coefficients $h(n)$ and indirectly on $l(n)$. We know already that the scaling filters are low-pass filters and wavelets have a complementary character. It is only logical to expect that the wavelet filters $h(n)$ are high-pass filters. The filter responses can be calculated using:

$$L(z) = \sum_{n=0}^{N} l(-n) z^{-n} \qquad H(z) = \sum_{n=0}^{N} h(-n) z^{-n} \tag{5.9}$$

This is expressed in the z-domain where $z = e^{j\omega}$. The minus sign in $l(-n)$ and $h(-n)$ is a result of the definition of convolution in discrete time: $y(n) = \sum_{k=-\infty}^{k=\infty} x(k)h(n-k)$ ($n = \tau$ translation). However the wavelet transform (4.3) in discrete time for scale $s = 1$ is defined as: $X_{WT}(\tau) = \sum_{k=-\infty}^{k=\infty} x(k)\psi(k-\tau)$ introducing a minus sign gives $\psi(-(\tau-k))$. The amplitudes of the response are represented in Fig. 5.3.

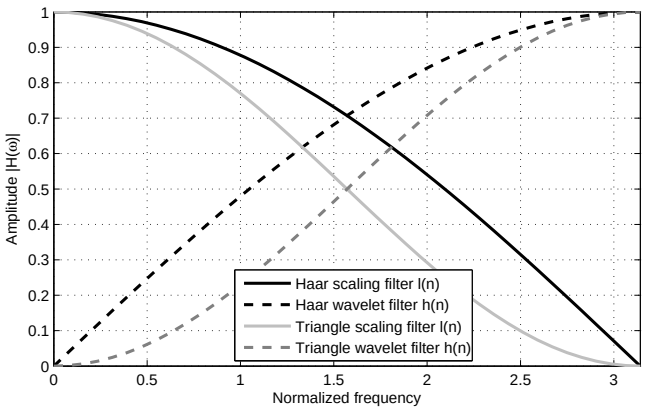


Figure 5.4: Amplitude of the Haar and triangle scaling and wavelet filters

The filters $l(n)$ and $h(n)$ are indeed the low-pass and high-pass filters. The attentive reader noticed that the scaling function is orthogonal to the wavelet function. The consequence is that when $L(z)$ is shifted by π it is equal to $H(z)$. Therefore it is directly clear which filter responses belong together. However this relationship only holds for wavelets that are orthogonal to their scaling functions. The general relationships can be found in Section 5.7. In this chapter all the tools used in the DWT are explained although we did not apply a DWT. Therefore to make it totally clear the Haar wavelet is used to perform a discrete wavelet transform, but of course any other wavelet could be used.

5.4 DWT applied on a Haar wavelet

The Haar- wavelet is the most basic wavelet and therefore very suited to explain the idea of a discrete wavelet transform. In Fig. 5.2 the scaling function was already introduced. The rest of the dilated or translated scaling filters can be found using (5.4) or (5.5). These dilations and translations are shown in Fig. 5.5 for a limited number of scales and translations.

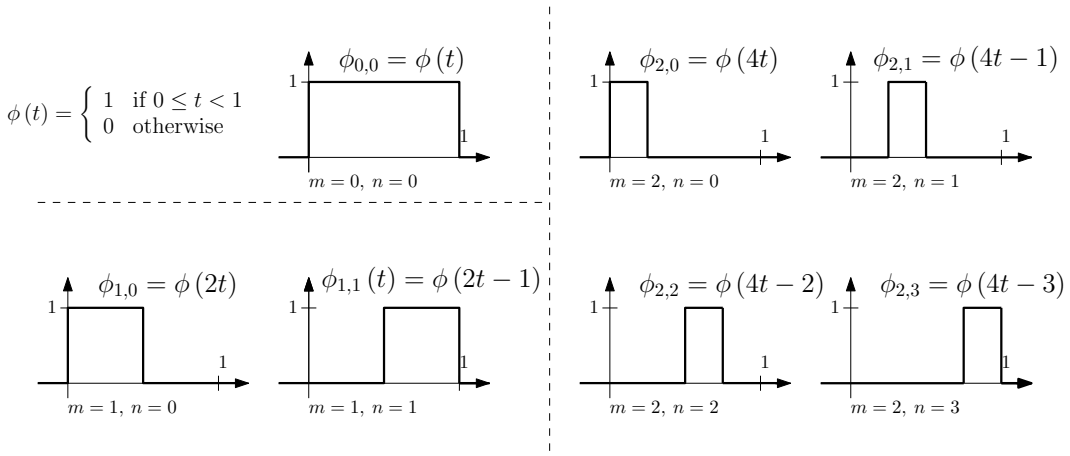


Figure 5.5: Box function (Haar's scaling functions)

The scale with the biggest scaling function is called here $m = 0$. Than it is easy to define $\phi_{0,0}$ as function of the smaller scales, see Fig. 5.5 or by using:

$$\phi_{0,0} = \sqrt{2}\left(\frac{1}{\sqrt{2}}\phi_{1,0} + \frac{1}{\sqrt{2}}\phi_{1,1}\right) \tag{5.10}$$

From these scaling functions it is also possible to calculate the wavelets as has been explained in Section 5.3. The different dilations and translations of the mother Haar wavelet $\psi_{0,0}$ are presented in Fig. 5.6.

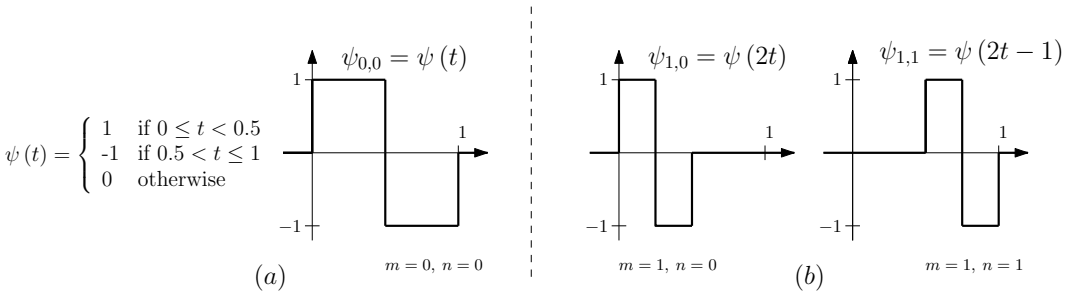


Figure 5.6: Haar wavelets

The relationship between scaling function and wavelet function can also be verified by analyzing Fig. 5.5 and Fig. 5.6. Here the wavelet dilation equation is elaborated for $\psi_{0,0}$.

$$\psi_{0,0} = \sqrt{2}\left(\frac{1}{\sqrt{2}}\phi_{1,0} - \frac{1}{\sqrt{2}}\phi_{1,1}\right) \tag{5.11}$$

In the discrete wavelet transform we want to calculate the wavelet coefficients from the smallest scale to the biggest scale (m=0). Therefore it is unnecessary to derive the so-called inverse relationships, nevertheless it gives good inside in the DWT. Combining (5.10) and (5.11) yields the inverse relationships that expresses ϕ (scaling function) as function of ϕ and ψ at a bigger scale.

$$\begin{aligned} \phi_{1,0} &= \tfrac{1}{2}(\phi_{0,0} + \psi_{0,0}) \qquad ((5.10)+(5.11)) \\ \phi_{1,1} &= \tfrac{1}{2}(\phi_{0,0} - \psi_{0,0}) \qquad ((5.10)-(5.11)) \end{aligned} \tag{5.12}$$

Now consider a function or signal $x(t)$, that can be expressed as a linear expansion (5.3). In this example an expansion of four basis functions $\phi_{2,n}$ is considered. These functions are already represented in Fig. 5.5 and can be also written down as:

$$x(t) = \sum_n a_2(n)\,\phi_{2,n} = a_2(0)\,\phi_{2,0} + a_2(1)\,\phi_{2,1} + a_2(2)\,\phi_{2,2} + a_2(3)\,\phi_{2,3} \tag{5.13}$$

Now the inverse relationships (5.12) are used to express this signal in terms of wavelets and scaling functions instead of only using scaling functions. These used scaling functions are of a different scale.

$$x(t) = \underbrace{\frac{a_2(0) + a_2(1)}{2}}_{\text{Mean}}\phi_{1,0} + \frac{a_2(2) + a_2(3)}{2}\phi_{1,1} + \underbrace{\frac{a_2(0) - a_2(1)}{2}}_{\text{Difference}}\psi_{1,0} + \frac{a_2(2) - a_2(3)}{2}\psi_{1,1} \tag{5.14}$$

In the case of the Haar wavelet something remarkable can be seen. The coefficients in front of the scaling coefficients $a_{1,n}$ at scale $m = 1$ are the average of the coefficients $a_{2,n}$ on scale $m = 2$. The wavelet coefficients $d_{1,n}$ are the difference of $a_{2,n}$ (an approximation of the first derivative). (5.14) is a linear expansion again but on a higher scale. The number of scaling functions and wavelets used in this new linear expansion results from (5.14). This new linear

expansion consists of two scaling functions and two wavelet functions. This is logical because a dyadic grid is used. Now (5.14) is rewritten in its new form:

$$x(t) = \sum_n a_1(n)\,\phi_1(n) + \sum_n d_1(n)\,\psi_1(n) = a_1(0)\,\phi_{1,0} + a_1(1)\,\phi_{1,1} + d_1(0)\,\psi_{1,0} + d_1(1)\,\psi_{1,1} \tag{5.15}$$

The coefficients $a_1(n)$ represent the average of two coefficients next to each other on a lower scale ($m = 2$) and $d_1(n)$ the difference. Now let us consider the part of the original signal $\tilde{x}(t)$ created by the scaling functions and coefficients.

$$\tilde{x}(t) = a_1(0)\,\phi_{1,0} + a_1(1)\,\phi_{1,1} \tag{5.16}$$

This function $\tilde{x}(t)$ is a rougher representation of the original function $x(t)$ because it was the average of the expansion coefficients $a_m(n)$ of one scale below. The coefficients $d_1(n)$ belong to the wavelet and contain the desired information of the signal. In this case it is the difference between two coefficients at one scale below.
In general the filter coefficients that generate ϕ are low-pass filters. As a consequence the linear expansion coefficients of ϕ always represent the signal depending on the scale. The approximation can be a very coarse (high scales) or very smooth (low scales). Therefore it is possible to use this coarser version of the linear expansion of our signal again. Taking this coarser version is more than acceptable because it is not only more efficient than again using an original expansion again, the missing part (high frequent information) is already contained in the wavelet coefficients. It is possible to move up one level again to a higher scale using the same method and the coarser representation of the signal $\tilde{x}(t)$.

$$\tilde{x}(t) = a_0(0)\,\phi_{0,0} + d_0(0)\,\psi_{0,0}, \qquad a_0(0) = \frac{a_1(0) + a_1(1)}{2}, d_0(0) = \frac{d_1(0) - d_1(1)}{2} \tag{5.17}$$

Now lets combine (5.16) and (5.17) and substitute into (5.14). Than we arrive at following equation that is nothing else than (5.2):

$$x(t) = a_0(0)\,\phi_{0,0} + d_0(0)\,\psi_{0,0} + (d_1(0)\,\psi_{1,0} + d_1(1)\,\psi_{1,1}) = \sum_n a_{0,n}\phi_{0,n} + \sum_n \sum_m d_{m,n}\psi_{m,n} \tag{5.18}$$

Here the discrete wavelet transform for the Haar wavelet was derived. Important to note is that it is possible to express the approximation of a signal on a higher scale with exactly the same accuracy by combining the wavelet coefficients and scaling coefficients of smaller scales. This procedure works for all discrete wavelet transforms exactly the same. Only the filter coefficients and length can be different; consequently the meaning of $d_{m,n}$ can be different. Also the correct expansion coefficients should be used as input to the filter bank although this often is neglected. Now the step needs to be made from the DWT to the DTWT through filter banks.

5.5 Filter Banks

So far only the CWT and the DWT were considered. In the DWT we have continuous time signals and discrete wavelet coefficients. However using computers requires discrete time signals and discrete outputs. Consequently the DWT needs to be discretized (DTWT). Although DTWT is the correct term in practice, all properties are derived for the DWT. The same steps are taken to find the wavelet coefficients $d_{m,n}$ but now using discrete time. Therefore the Haar wavelet is taken as an example again.
At the lowest scale, here $m = 2$, the decomposition coefficients $a_2(n)$ need to be found. In the continuous case this is the box function (Fig. 5.2). The sample frequency is now defined as the width of these scaling functions $\phi_2(n)$. Then the box function in discrete time is nothing else than the dirac delta function. Therefore the expansion coefficients $a_2(n)$ for the Haar scaling

function are exactly the sample points of our discrete time signal. The number of coefficients equals the number of samples of the original sampled signal (with length N).

$$x(n) = a_2(n) = [a_2(0), a_2(1), a_2(2), a_2(3), ..., a_2(N)] \tag{5.19}$$

This is only true when using the Haar scaling function; in case other scaling functions are used, these coefficients need to be calculated as is explained in Section 3.1. For filter banks a method of finding the initial expansion coefficients is explained in Section 6.6. For now only the Haar scaling function is considered.
If the scaling function at scale $\phi_2(n)$ is the dirac function, than that of $\phi_1(n)$ is a discrete function of 2 coefficients ($[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$) and the wavelet coefficients at this scale are ($[\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}]$). These coefficients are nothing else than the filter coefficients $l(n)$ and $h(n)$. Now the first wavelet coefficients can be calculated by taking the product with the filter coefficients $h(n)$. The filters used here take the inner-product with the expansion coefficients. This product is here generated using a FIR-filter, which means here that the wavelet has compact support; the wavelet is only non-zero in a limited interval. Taking this product as if the coefficients are filtered gives us the following result:

$$d_1' = \frac{1}{\sqrt{2}} [a_2(0) - a_2(1), a_2(1) - a_2(2), a_2(2) - a_2(3), a_2(3) - a_2(4), ...] \tag{5.20}$$

Compare this to the result of our DWT in (5.14); clearly this is a very similar result with the main difference that the number of coefficients that are calculated are doubled thus every uneven coefficient is a redundant one. Consequently these coefficients need to be removed. Deleting the uneven samples is called downsampling ($\downarrow 2$). Deleting every uneven sample leads to the same result as in (5.14), except for a factor of $\frac{1}{\sqrt{2}}$. This factor is typical for filter banks where downsampling removes half the samples: it changes the energy (2-norm). Therefore the 2-norm of the downsampled vector needs to be compensated. It should have length $\sqrt{2}$ instead of 1. This means when comparing different scales, this factor should be taken into account i.e.

$$d_1(n) = \frac{1}{2} [a_2(0) - a_2(1), a_2(2) - a_2(3), ...] \tag{5.21}$$

The coefficients $d_1(n)$ can be used to analyze the signal on this scale. Exactly the same method is used for every step of the expansion coefficients ($a_m(n)$) using the filter coefficients $l(n)$ instead of $h(n)$. In this example this means finding the expansion coefficients $a_1(n)$. If $a_1(n)$ are known, than the wavelet coefficients $d_0(n)$ can be calculated using the same procedure. This means that for every new level a new set of filters should be added to the existing filters. Such a combination of filters is called a filter bank. The number of steps are associated with the same number of wavelet scales calculated. Such a filter bank is represented in Fig. 5.7:

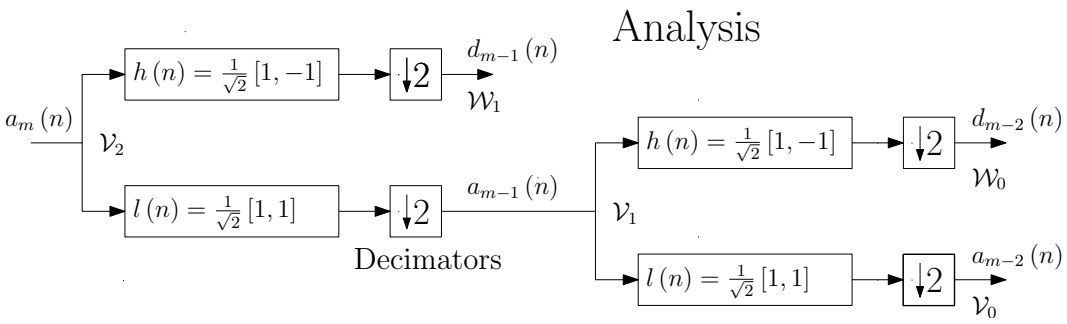


Figure 5.7: Filter bank (Haar)

The space spanned by the translated scaling functions $\phi_{m,n}$ is called $\mathcal{V}_m$ and for the translated wavelets $\psi_{m,n}$ is called $\mathcal{W}_m$. This type of filter bank does not only hold for the Haar wavelet, but in principle for every DWT when it has a scaling function or scaling filter. The only difference will be the filter coefficients inside the filter bank. In Fig. 5.3 the amplitudes of the

scaling and wavelet filter are given. One level of a filter bank splits up the spectrum in two parts: a low-pass spectrum and high-pass spectrum. This is represented in Fig. 5.8.

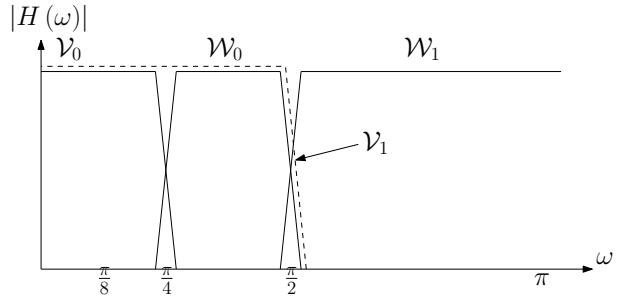


Figure 5.8: Spectrum created by a filter bank

This spectrum belongs to the most common class of filter banks, the Quadrature Mirror Filter banks (QMFB). They are called QMFB because the scaling filter and wavelet filters spectra are mirrored and the spectrum is split by powers of two. Although there are other ways to split up the spectrum this is the most common one. Different names for the filter bank are the pyramidal algorithm or Mallat's algorithm after its inventor.

5.6 Downsampling and Upsampling

In the last sections the DWT and filter banks are discussed. It already became clear from comparing the DWT and DTWT that downsampling is necessary. This downsampling ($\downarrow 2$) removes all uneven samples. Downsampling also makes a filter bank efficient because it reduces the number of data for every level by two. This operation seems to be destroying a lot of useful data. This is not the case as we already saw in the previous section. In this section this is made even clearer. Let us consider a signal $x(n)$ that needs to be downsampled:

$$(\downarrow 2)\begin{bmatrix} . \\ x(-1) \\ x(0) \\ x(1) \\ x(2) \\ . \end{bmatrix} = \begin{bmatrix} . \\ x(-2) \\ x(0) \\ x(2) \\ x(4) \\ . \end{bmatrix} \tag{5.22}$$

The inverse operation of down- is upsampling ($\uparrow M$) it adds a zero(s) between sample(s) on the place where it is removed by downsampling.

$$u(n) = (\uparrow 2)\begin{bmatrix} . \\ x(-2) \\ x(0) \\ x(2) \\ x(4) \\ . \end{bmatrix} = \begin{bmatrix} . \\ x(-2) \\ 0 \\ x(0) \\ 0 \\ x(2) \\ . \end{bmatrix} \tag{5.23}$$

The operations of first down- and then upsampling results in a new discrete signal where all uneven signal coefficients are zero. This can also be expressed using a so-called alternating sign:

$$u(n) = (\downarrow 2)(\uparrow 2)x(n) = \frac{1}{2}\left[x(n) + (-1)^n x(n)\right] \tag{5.24}$$

If n is uneven the signal coefficients are subtracted giving zero for its coefficients. Conversely when n is even they are summed up together and need to be divided by two again. This

equation can be used to study the effect of down- and upsampling in the discrete frequency domain (z-transform (5.9)).

$$U(z) = \frac{1}{2}\sum_{n}\left[x(n) + (-1)^{n} x(n)\right] z^{-n} \tag{5.25}$$

The summation in (5.25) is expressed in two separate sums. The alternating sign function will remain the same if replaced by $(-1)^{-n}$:

$$U(z) = \frac{1}{2}\left[\sum_{n} x(n) z^{-n} + \sum_{n} x(n) (-1)^{-n} z^{-n}\right] \tag{5.26}$$

Now the z-transform of the down- and upsampled signal is nothing else than:

$$U(z) = \frac{1}{2}\left[X(z) + X(-z)\right] \tag{5.27}$$

In the continuous frequency domain the term $-z$ is exactly the same as $\omega + \pi$, which is also common in literature. The extra term $X(-z)$ is an aliasing term that is generated as a consequence of changing the nyquist frequency two times during the operation of up- and downsampling. This term can be partly canceled out. More about aliasing can be found in Section 6.8.
It is still possible to reconstruct the original signal using down- and upsampling. The most simple form of a filter bank with no loss of information (perfect reconstruction) is the lazy filter bank. In essence it does nothing else than down- and upsample the signal. The lazy filter bank is shown in Fig. 5.9.

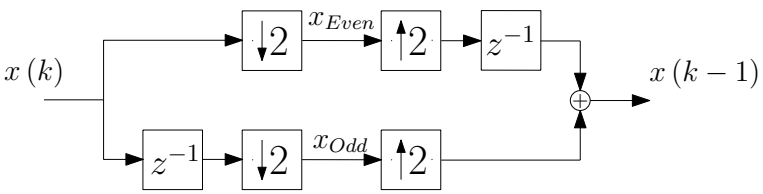


Figure 5.9: Lazy filter bank

This can be expressed as:

$$x(k)\, z^{-1}(\downarrow 2)(\uparrow 2) + x(k)\,(\downarrow 2)(\uparrow 2) z^{-1} = \begin{bmatrix} \cdot \\ 0 \\ x(0) \\ 0 \\ x(2) \\ \cdot \end{bmatrix} + \begin{bmatrix} \cdot \\ x(-1) \\ 0 \\ x(1) \\ 0 \\ \cdot \end{bmatrix} = x(k-1) \tag{5.28}$$

It becomes clear that even after down- and upsampling; it is possible to have perfect reconstruction. Even if it seems we are throwing away a lot of valuable data by downsampling. This whole operation only results in a delayed reconstructed signal compared to the original signal. The most common case is downsampling with two samples but there are many other filter banks that use other downsampling operations.
It is highly inefficient to first calculate all data and then downsample. Therefore we want to interchange the filter operation with each other. In the z-domain one often uses the noble relationships to calculate this interchange,

$$G(z)(\downarrow M) = (\downarrow M) G\left(z^{M}\right) \tag{5.29}$$

$$(\uparrow M) G(z) = G\left(z^{M}\right)(\uparrow M) \tag{5.30}$$

This operation of first downsampling and then filtering will be discussed more thoroughly in Section 5.9. For now we consider the classic case with first filtering and downsampling.

5.7 Synthesis and Perfect reconstruction

In the previous sections the filter bank is discussed. This explanation is incomplete because only the analysis part is explained. The analysis part decomposes the signal but often we also want to reconstruct the signal again after changing a few wavelet coefficients. This is actually one of the strengths of the filter bank and the DWT. In Fig. 5.9 the lazy filter bank is introduced to give an idea about this synthesis step. In Section 3.1 the synthesis of a function is explained. Here exactly the same holds with the only difference, that now it is formed in a number of steps. The synthesis of the function is of course nothing else than (5.2). Here we try embed this (5.2) in a synthesis filter bank. If the expansion coefficients of the original signal can be reconstructed from all the wavelet coefficients and the highest scale expansion coefficients, then such a filter bank is called a **Perfect Reconstruction Filter Bank** (PR-bank). The general shape of such a PR-bank is presented in Fig. 5.7.

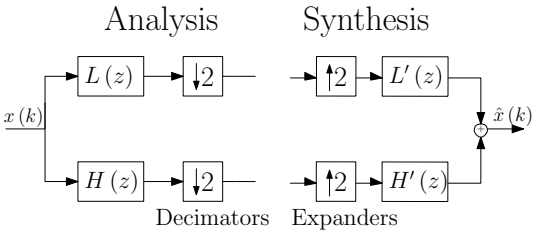


Figure 5.10: Filter bank, 1 level

The filters are represented in the z-domain, in this case a one level filter bank is used. However adding more levels to the filter bank will not change its properties. This report considers almost only real-time wavelet analysis (causality). Therefore the reconstructed signal will be always delayed compared to its original signal.

$$\hat{x}(k) = x(k - N) \tag{5.31}$$

In Section 5.5 we already saw that the expansion coefficients need to be put into the filter bank. Perfect Reconstruction is one of the most important properties of the DWT if not the most important property. Perfect reconstruction is normally stated in two relationships. These can be derived by calculating the discrete transfer function from input to output of the filter bank. Therefore every channel of the filter bank (Fig. 5.7) needs to be elaborated in the frequency domain with the help of (5.27). The low-pass and high-pass channel have following relationships in the frequency domain, where the z-transform of the signal is denoted as $X(z)$.

$$\text{low-pass channel output} = \frac{1}{2}L'(z)\left[L(z)X(z) + L(-z)X(-z)\right] \tag{5.32}$$

$$\text{high-pass channel output} = \frac{1}{2}H'(z)\left[H(z)X(z) + H(-z)X(-z)\right] \tag{5.33}$$

$$\tag{5.34}$$

The input is first filtered by the analysis filter, this gives output $H(z)X(z)$. Then it is down- and upsampled resulting in the alias term $H(-z)X(-z)$ and finally filtered again with $H'(z)$. These channels need to be added together to form the original delayed signal again.

$$z^{-N}X(z) = \frac{1}{2}\left[H'(z)H(z) + L'(z)L(z)\right] + \frac{1}{2}\left[H'(z)H(-z) + L'(z)L(-z)\right] \tag{5.35}$$

The terms with $-z$ are related to aliasing and need to be canceled out. This can only be done by canceling out the second term of (5.35) i.e. the second term should be equal to zero, hence the name aliasing cancelation.

$$H'(z)\,H(-z) + L'(z)\,L(-z) = 0 \tag{5.36}$$

This condition is often further simplified because there is one straightforward choice of filters that fulfills the aliasing condition:

$$\begin{aligned} H'(z) &= -L(-z) \\ L'(z) &= H(-z) \end{aligned} \tag{5.37}$$

In theory there are other possibilities nevertheless this is the combination generally used in practice. The second condition for perfect reconstruction will be achieved if there is also no distortion. This condition is called perfect reconstruction is equal to (5.35) without the second term related to aliasing.

$$H'(z)\,H(z) + L'(z)\,L(z) = 2z^{-N} \tag{5.38}$$

Often these two PR relationships are combined in a matrix relationship.

$$F_m(z)\,H_m(z) = \begin{bmatrix} L'(z) & H'(z) \\ L'(-z) & H'(-z) \end{bmatrix} \begin{bmatrix} L(z) & L(-z) \\ H(z) & H(-z) \end{bmatrix} = \begin{bmatrix} 2z^{-N} & 0 \\ 0 & 2(-z)^{-N} \end{bmatrix} \tag{5.39}$$

The matrix H_m is called the modulation matrix and can be used for instance to check orthogonality.

5.8 Types of filter banks

In Section 5.5 the filter bank related to wavelets is discussed. However, one can imagine that filter banks could be build in many different configurations. As a consequence sometimes they are not really related to a wavelet anymore. Three major types of filter banks can be distinguished. First of all the one that was discussed in the previous sections, the Haar filter bank. One can imagine that it is also possible to develop every root of the filter bank, which is called a symmetric filter bank or full wavelet packet decomposition. The third major type is the M-band Wavelet system which uses several filters on one channel to split up the signal, thus downsampling greater than two is needed. All of them can be also used to reconstruct the original signal when all necessary conditions are fulfilled. All three types are represented in Fig. 5.11.

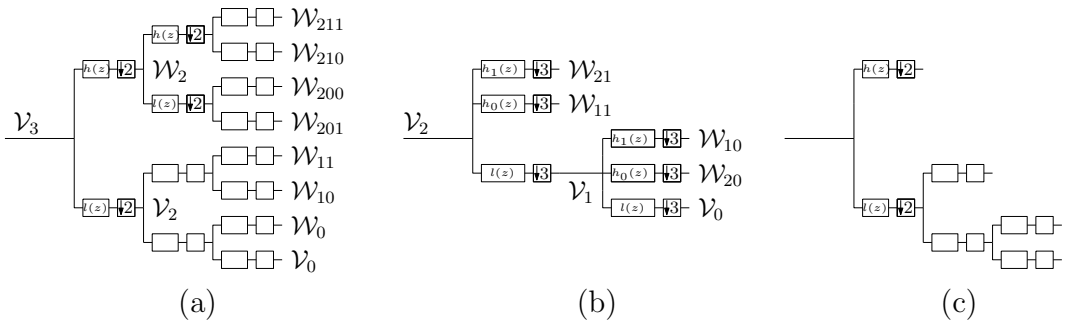


Figure 5.11: Three different types of filter banks; (a) Symmetric (b) M-band (c) Haar

Already we explained for the Haar filter bank how this relates to frequency domain. The same can be done for the frequency bands that are belonging to these other two filter banks. They are represented in Fig. 5.12.
The symmetric filter bank splits up the signal in even frequency bands. On the other hand the M-band filter bank splits up a signal in three frequency bands and splits the low-frequency

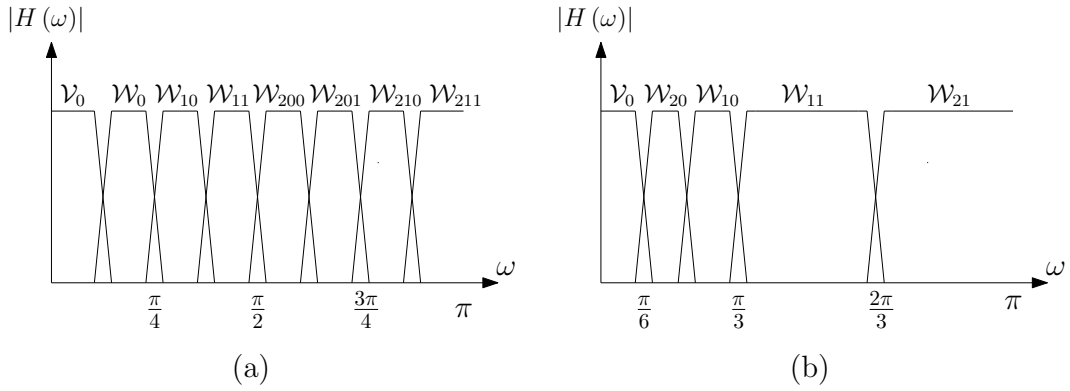


Figure 5.12: Frequency resolution (a) Symmetric Filter Bank (b) M-band Filter Bank

band again in three smaller frequency bands. It becomes clear that filter banks can also be used to separate every desired frequency band using the right sequence of downsampling and filters. When using general wavelet theory in principle the Haar filter bank (c) should always be used. If we are only interested in a certain frequency band or a limited number of frequency bands the M-band is the preferred choice. If one simply wants to split up a signal into a number of frequency bands and perhaps do some operations the full-wavelet package should be used. However this type of filter bank is not very efficient, since calculation of the great number of channels can be time consuming.

5.9 Efficient filter banks; Polyphase domain

A slightly different filter bank is created when instead of doubling the signal, it is split in an even and odd part. As a consequence the signal is no longer doubled and put through both the scaling and wavelet filter and then downsampled. Instead the signal is split in two, the even and odd samples, this is called polyphase. The difference of implementation is presented in Fig. 5.13.

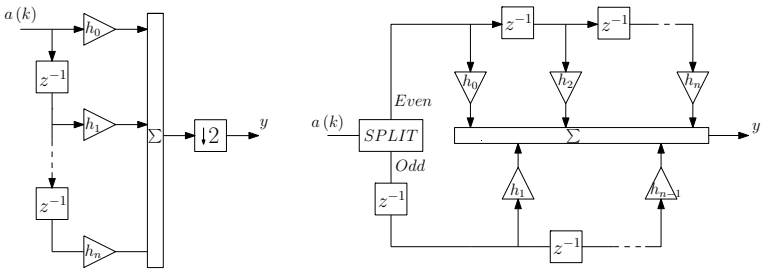


Figure 5.13: Normal FIR filter with downsampling (left) and Polyphase filter (right)

The polyphase implementation is possible because the downsampling after normal filtering removes all the outputs where even samples are multiplied with odd filter coefficients and vice versa. This leads to a reduction of calculation time without any loss of information. The DWT for a basic two channel polyphase filter bank will have following form:

$$\begin{pmatrix} a_{m-1}(z) \\ d(z) \end{pmatrix} = H_p \begin{pmatrix} a_m^{even}(z) \\ z^{-1} a_m^{odd}(z) \end{pmatrix} \quad \text{with} \quad H_p = \begin{pmatrix} L_{even}(z) & L_{odd}(z) \\ H_{even}(z) & H_{odd}(z) \end{pmatrix}, \tag{5.40}$$

Where H_p is called the polyphase matrix. The delay comes from the fact that odd-samples are always delayed with one sample. In the case of perfect reconstruction filter banks also have a synthesis step. The PR-filter bank is represented in Fig. 5.14.
In this case perfect reconstruction is achieved when:

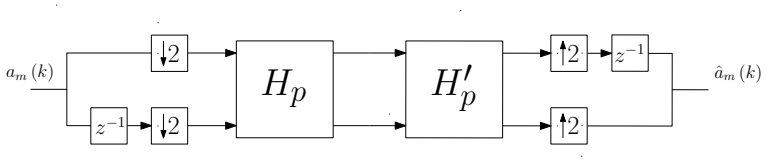


Figure 5.14: Polyphase filter bank

$$H_p\left(z^{-1}\right) H_p(z)' = \mathbf{I} \text{ Perfect reconstruction} \tag{5.41}$$

This in essence means that when the polyphase matrix needs to be inverted (and thus H_p needs to be invertible) to find the synthesis part. In the case of a rectangular H_p, the pseudo-inverse needs to be calculated. In theory this representation is often not used or only mentioned. In practice it should always be applied and often it already is, for instance in Matlab Simulink. Not only does this representation reduce the number of computations, it also makes the design much easier. The only requirement is that the product of the matrix $H_p\left(z^{-1}\right) H_p(z)'$ is the unity matrix. Here a non-causal design is applied if the filter bank is causal the right-hand-side is delayed. One should remember there are many ways to derive this and often literature gives one or more of these sub-requirements. For instance in case of a square matrix H_p, one needs to find a H_p of which the determinant is equal to one.
The polyphase matrix gives us a more efficient implementation but also simplifies important relationships between analysis and synthesis. It also makes the transition to lifted DWT easier as will be seen later on. Also IIR-polyphase filter banks can be implemented.

5.10 Conclusions and considerations

In this chapter the DWT is introduced through the example of the Haar wavelet. Similar steps can be taken for every different discrete wavelet. In Section 5.2 and Section 5.3 where the scaling function and wavelet are predefined: in practice the filters are often designed. Those filters can then directly be applied in filter banks. The Haar example showed that the filter bank is an efficient way of applying a discretized DWT. This discretization will lead to some approximation errors that will be discussed later.
The requirements for perfect reconstruction can easily be written down for filters, making it easier to design them. Filter banks not only offer the chance of applying efficient wavelet transforms but also can be extended to separate every desired frequency band. Separating frequency bands is very useful when one needs to detect or change certain frequencies inside a signal. The filter bank is already quite efficient, however, it has the problem that the channels are split in two; doubling the information temporarily. This last disadvantage is taken away by the polyphase design which first downsamples the signal. This comes without any loss of information or additional problems and therefore should always be used in practice.

Chapter 6

Properties of the discrete wavelet transforms

There are a great number of properties that are related to discrete wavelets. Together they determine the performance of a wavelet. The most important property is multiresolution, which was also indirectly subject of the previous chapter. In the preceding chapter it is explained through an example; in this chapter Multi Resolution Analysis (MRA) will be explained theoretically in Section 6.1. Then the different types of multiresolution are explained i.e. orthogonal and biorthogonal wavelets. In Section 6.5 the relationship between filters and wavelets is derived. Thereafter two concepts related to problems originating from filter banks are discussed: initialization and aliasing. Finally the most important types of wavelets will be shortly discussed including multiwavelets.

6.1 Theoretical Aspects of the DWT Multiresolution

In the previous chapter multiresolution was introduced. Although the introduction was done through an example, it was already quite complete. However most mathematical books use a more formal definition for multiresolution. Before a good multiresolution can be introduced, some mathematical definitions need to be given. The signal $g(t)$ is defined in $L^2(\mathbb{R})$, this means that the function is square integrable i.e. $\int_{-\infty}^{\infty} |g(t)|^2 dt$ is finite. All translations on one scale $\phi_{0,n}$ span the space $\mathcal{V}_0$ and all translations of $\psi_{1,n}$ span the space $\mathcal{W}_1$ or more formal:

$$\mathcal{V}_m = \overline{Span_n\{\phi_{m,n}(t)\}}, \qquad \mathcal{W}_m = \overline{Span_n\{\psi_{m,n}(t)\}} \tag{6.1}$$

The mathematical definition of a genuine multiresolution is as follows:

$$\{0\} = \mathcal{V}_{-\infty} \subset ... \subset \mathcal{V}_{-1} \subset \mathcal{V}_0 \subset \mathcal{V}_1 \subset ... \subset \mathcal{V}_{\infty} = L^2(\mathbb{R}) \quad \text{Completeness} \tag{6.2}$$

$$g(t) \in \mathcal{V}_m \Longleftrightarrow g(t-n) \in \mathcal{V}_m \quad \text{Translation} \tag{6.3}$$

$$g(t) \in \mathcal{V}_m \Longleftrightarrow g(2t) \in \mathcal{V}_{m+1} \quad \text{Dilation} \tag{6.4}$$

$$\mathcal{V}_{m+1} = \mathcal{V}_m \dot{+} \mathcal{W}_m \quad \text{Orthogonality} \tag{6.5}$$

In (6.2) it is stated that a bigger subspace contains all smaller subspaces. The biggest subspace (very small width of scaling functions) should contain all information of the signal and an infinitely small subspace no information at all. $\mathcal{V}_{\infty}$ is theocratically the first step in the filter bank and it contains all information of the signal. After every downsample step the subspace looses information, to the wavelet subspace, until it contains no information anymore ($\mathcal{V}_{-\infty}$). In (6.3) is stated that translations of the signal should be part of the same space . The dilation property (6.4) is written down for a dyadic multiresolution i.e. a signal that is twice as fast in time is part of the space above. The fourth definition is important because it explains why the DWT is efficient, this is also explained in Section 5.2. Here a $\dot{+}$ is used to denote the sums

between the subspaces. This is because it depends on the relationship between the scaling function and wavelet. This will be the subject of the next section.

6.2 Orthogonality

In this section orthogonality is explained. Two other types of wavelets are the semi-orthogonal and biorthogonal wavelets. Orthogonality is the most specific property (orthogonal wavelet) and biorthogonality the most general property. Orthogonality makes sure that the filter bank has perfect numerical conditions. In Section 3.1 frames in general and orthogonal frames are discussed. There it is explained that orthogonal basis functions give orthogonal matrices in discrete time. When calculating the synthesis step the matrices only need to be transposed instead of numerically problematic calculations of the inverse. In the DWT we have a scaling function and wavelet that span the signal (5.2) together. Also in filter banks there is an analyzing step where the dual-frames of the scaling and wavelet function are used. So in total there are four functions that can be related to each other. The relationships are orthogonality, semi-orthogonality and biorthogonality. Here only perfect reconstruction is considered, which is embedded in the multiresolution analysis. Orthogonality in the sense of wavelets means that not only the frame and dual-frame are orthogonal but also the scaling function and wavelet are orthogonal to each other:

$$\int_{-\infty}^{\infty} \phi\left(t-n\right)\phi\left(t-l\right)dt = \delta\left(n-l\right) \tag{6.6}$$

$$\int_{-\infty}^{\infty} \phi\left(t-n\right)\psi\left(t-l\right)dt = 0 \tag{6.7}$$

and all wavelets over all scales should be orthogonal (not necessary for ϕ),

$$\int_{-\infty}^{\infty} \psi_{m,n}\left(t\right)\psi_{m',n'}\left(t-l\right)dt = \delta\left(m-m'\right)\delta\left(n-n'\right) \tag{6.8}$$

This can be summarized as:

$$\mathcal{V}_m \perp \mathcal{V}_m,\ \mathcal{V}_m \perp \mathcal{W}_m,\ \mathcal{W}_1 \perp \mathcal{W}_2 \perp \mathcal{W}_{m-1}, \tag{6.9}$$

Already it is known that (6.6) is numerically conditioned well when transforming from analysis to synthesis. In addition the energy is partitioned equally when the scaling function is orthogonal to the wavelet and all wavelet spaces to each other . Therefore the energy of every scale is the same when normalized. This is important because comparing different scales is then much easier. This is the most important property of orthogonal wavelets.
There are a lot of different representations of orthogonality. In general, however, design of the filter coefficients is the practical way to design wavelets. To fulfill this kind of orthogonality the following condition on the filter coefficients must hold [13]:

$$\delta\left(k\right) = \sum l\left(n\right)l\left(n-2k\right) \qquad k\,\epsilon\,\mathbb{Z} \tag{6.10}$$

This equation is often used together with the relationship for conjugate mirror filters (6.11), which together form a orthogonal system.

$$h\left(n\right) = \left(-1\right)^{\left(1-n\right)}l\left(1-n\right) \tag{6.11}$$

The orthogonality of the scaling functions is denoted by (6.10) and the relationship between the scaling function and wavelet is denoted by (6.11). Important to remember is that the filter coefficients are related to wavelets by (5.4). It is well known that a convolution, here (6.6), in the z-domain will become a product. This explains why it is a product of filters. The second filter has a term $n-2k$, which is because of double-shift orthogonality. In DTWT always a downsampling (or in DWT $2t$) step is applied, thus the coefficients do not need to be orthogonal to every uneven sample because they are deleted anyway. Now with the help

of (5.7), (6.7) and (6.8), the relationship between scaling and wavelet filters can be derived (6.11), this is the famous Daubechies design.
In Section 5.9 it is explained that it is more efficient to first downsample and then filter. Here it is explained that the orthogonal filter design using the polyphase matrix H_p is more straightforward. The complex version of an orthogonal matrix is the unitary matrix. A matrix is unitary if the complex transposed is equal to the inverse (or pseudo-inverse).
In the polyphase domain this means that the following relationship must hold for H_p to be orthogonal.

$$\left(H_p^*\right)^T (z) H_p (z) = H_p^T \left(z^{-1}\right) H_p (z) = \mathbf{I} \tag{6.12}$$

Unitary matrices are known to preserve the vector length ($\|Ux\|_2 = \|x\|_2$). This is also exactly what the Parseval's theorem says. If one wants to find an orthogonal set of filters: H_p needs to fulfill (6.12). The main disadvantage of orthogonal wavelets is that orthogonality greatly limits the design freedom of the filters. Filters have to be the same length (square matrix), also orthogonality prevents symmetry, which will be discussed in the next section.

6.3 Symmetry of wavelets

Symmetry simply means that the wavelet and scaling function are symmetric. Leading to also symmetric filters, which generate the wavelet and scaling function. Symmetry is equivalent to the notion of linear phase of filters. Linear phase means that the phase is a linear function of the frequency and consequently the filter is symmetric. Causal filters have linear phase when their filter coefficients are (anti-)symmetric around a central coefficient.

$$\begin{array}{ll} h(n) = h(N-n) & \text{Symmetric} \\ h(n) = -h(N-n) & \text{Anti-symmetric} \end{array} \tag{6.13}$$

Symmetry has from a numerical point of view no purpose, however, in image processing symmetry plays an important role. This is because the human eye reacts much better on symmetric errors than on asymmetric errors. In images more symmetry will lead to better compressibility because the perceptual error will allow more symmetrical errors compared to asymmetrical errors. In addition image edges, which are often symmetric are more difficult to handle without symmetry.Symmetry and orthogonality are mutual exclusive except for one exception: the Haar wavelet. In [13, 32] it is shown that it is only possible to have symmetry and orthogonality when the filter has only two non-zero elements. As a consequent of this mutual exclusiveness other non-orthogonal wavelets are necessary, which will be discussen in the next section.

6.4 Semi-orthogonality and Biorthogonality

In this section semi-orthogonality and biorthogonality is explained, which are continuation of the discussion on orthogonality. If one wants symmetry and more freedom, the orthogonality conditions can be slightly loosened i.e. no longer requiring wavelets to be orthogonal to itself. However still requiring the scaling function to be orthogonal to the wavelet:

$$\mathcal{V}_m \perp \mathcal{W}_m \tag{6.14}$$

These wavelets are called semi-orthogonal. They can have desirable properties in relation to approximation of signals, symmetry and filter lengths. These properties, however, are not a consequence of semi-orthogonality but it is a consequence of the filter choices. Semi-orthogonality itself has no relevance at all.
Biorthogonal wavelets and biorthogonal filter banks are another extension. They can be symmetric and offer more degrees of freedom. They use two frames instead of one orthogonal frame . However they still need to fulfil the minimum requirement of perfect reconstruction (5.38). In (5.38) this relationship was derived using the z-transform. However it is possible to derive this also in the continuous time domain (DWT) [32]. This results in the following relationships:

$$\mathcal{V}_m \perp \tilde{\mathcal{W}}_m,\ \mathcal{W}_m \perp \tilde{\mathcal{V}}_m \tag{6.15}$$

Now it becomes even more clear why these wavelets are called biorthogonal. The analyzing scaling filter is orthogonal to the synthesis wavelet filter (and vice versa), to be complete double shift orthogonal because downsampling removes every uneven sample, which makes it unnecessary to be orthogonal to every shift;

$$\sum_n h(n)\, l'(n-2k) = 0, \qquad \sum_n l(n)\, h'(n-2k) = 0 \qquad k \,\epsilon\, \mathbb{Z} \tag{6.16}$$

The great advantage of biorthogonal wavelets is that they give a lot of design freedom. Filter lengths for reconstructing and decomposition often differ and the filters can be symmetric. Biorthogonal filters are the most popular filter type to use, however, it has some disadvantages. This is the result of the fact that the Parseval's theorem no longer holds for biorthogonal wavelets. In other words $A \neq B$ in (3.13), which are the 2-norm bounds of the inner-product between the wavelet and function. This makes the calculations numerically more problematic, the problem is exactly the same as numerical problems related to calculating the inverse matrix (condition number). There is no longer (energy) correlation of the decomposition coefficients (wavelet coefficients) at the different scales. This makes it much harder to compare different scales with each other. Therefore in most of the biorthogonal filter designs $A \approx B$. The other reason is simply because of computational issues. This is understood properly if one considers the polyphase domain. Here we remember that we have the product of $H_p\left(z^{-1}\right) H_p(z)^{-1} = \mathbf{I}$. When using biorthogonal filters we are no longer computing the complex conjugate transpose but the inverse. The bigger the difference between A and B, the worse the condition number is. Biorthogonal wavelets have proven to be a very good alternative to orthogonal wavelets, if one keeps $A \approx B$ biorthogonal wavelets are the preferred choice. In this case they are nearly orthogonal: still benefiting from the advantages of orthogonality, but also have the benefit of biorthogonality such as symmetry. In terms of scaling functions and wavelets, instead of having the same wavelet and scaling function for decomposition and reconstructing, we have now two different functions for reconstruction and decomposition. An example of such a biorthogonal wavelet system is the Bior2.2 which is presented in Fig. 6.1.

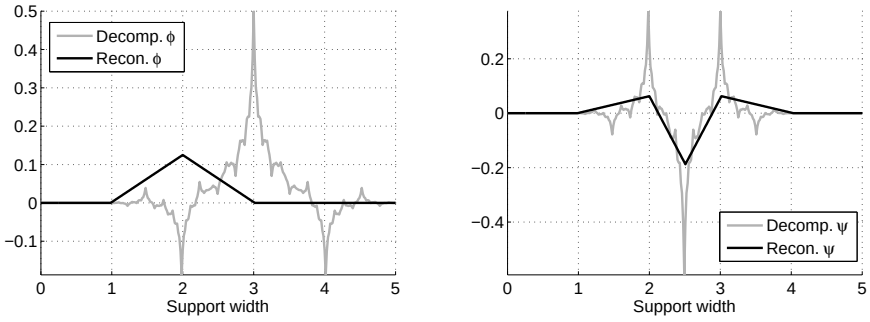


Figure 6.1: The scaling and wavelet functions of the Bior2.2 wavelet system

Clearly the decomposition and reconstruction wavelets are very different. For the Bior-wavelets the reconstruction wavelet has been designed and the decomposition has been calculated using the perfect construction condition (for the reversed biorthogonal wavelets(Rbio) the decompostion wavelet is designed). Consequently the decomposition filters generate the decomposition wavelet, which shows generally fractal behavior because it is based on recursion. The relationship between filters and wavelets is explained in the next section for another wavelet: the DB-wavelet that also shows fractal behavior.

6.5 Relationship filter bank and wavelet

In Chapter 5 filter banks have been introduced through existing wavelets. However most discrete wavelets are defined through the filter coefficients. In this case the relationship between filter banks and actual wavelets is still unclear. In this section it is explained how wavelets are related to these filter banks. It is possible to derive the wavelet that belongs to the filter bank by recursion of the two scale relationship (5.4).
It is already known that wavelets generated from FIR-filter banks have compact support. The compact support of the scaling function (Father wavelet) and the (Mother) wavelet are $[N_1, N_2]$, which means that $l(n)$ has a finite impulse response from N_1 to N_2, and the wavelet a support of $[\frac{N_1-N_2+1}{2}, \frac{N_2-N_1+1}{2}]$ [24]. The support width of the scaling filter depends on the number of filter coefficients, which is logical for FIR-filters. For instance in the case of filter length 10 the support width of the scaling function is $[0, 9]$. The wavelet support width follows by definition from the scaling function. In general the first recursion step for the scaling function (5.4) can be written out as (6.17). Important to remember is that outside the support width the scaling function equals zero.

$$T\vec{\phi} = \vec{\phi} \text{ with } T = \begin{pmatrix} l(0) & 0 & 0 & \cdot & & & & & \\ l(2) & l(1) & l(0) & 0 & \cdot & & & & \\ l(4) & l(3) & l(2) & l(1) & \cdot & & & & \\ \cdot & l(5) & l(4) & l(3) & \cdot & & & & \\ & & & & \cdot & 0 & l(n) & l(n-1) & l(n-2) \\ & & & & & & \cdot & 0 & l(n) \end{pmatrix}, \vec{\phi} = \begin{pmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \\ \cdot \\ \phi(n-1) \\ \phi(n) \end{pmatrix} \tag{6.17}$$

The coefficients of T are the scaling filter coefficients. It is clear that $\phi(0) = \phi(n) = 0$ by considering the first and last row of T. This is because the coefficients $l(0)$ and $l(n)$ are non-zero elements. This kind of problem, (6.17), is called an eigenvalue one problem and by solving it, the values of $\vec{\phi}$ can be found. However the solution $\vec{\phi}$ is still a vector. Therefore the unique solution to this problem needs to be determined. This is done by normalization ($\sum_k \phi(k) = 1$) of $\vec{\phi}$ defining a unique ϕ.
The construction method is made more clear by considering a DB_2 scaling function and wavelet. The scaling filter has a support width of $[0, 3]$ i.e. the coefficients are defined as:

$$l(n) = [l(0), l(1), l(2), l(3)]^T \tag{6.18}$$

Already it is known that $\phi(0) = \phi(n = 3) = 0$ results in following eigenvalue one problem,

$$\begin{pmatrix} l(1) & l(0) \\ l(3) & l(2) \end{pmatrix} \begin{pmatrix} \phi(1) \\ \phi(2) \end{pmatrix} = \begin{pmatrix} \phi(1) \\ \phi(2) \end{pmatrix} \tag{6.19}$$

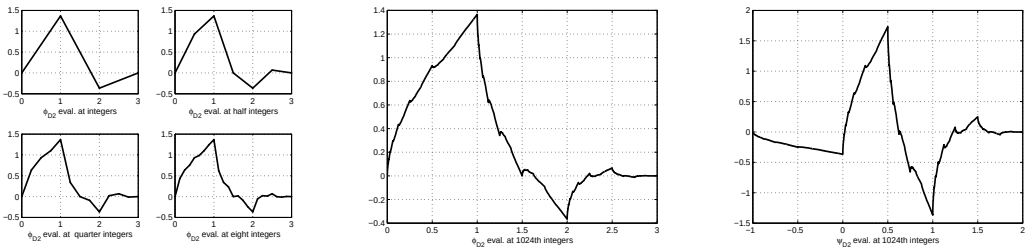
The eigenvector $\vec{\phi}$ can now be found by solving this system and by using the normalization ϕ is found i.e.

$$\begin{pmatrix} l(1)-1 & l(0) \\ l(3) & l(2)-1 \end{pmatrix} \begin{pmatrix} \phi(1) \\ \phi(2) \end{pmatrix} = \vec{0} \text{ and } \phi(1) + \phi(2) = 1 \tag{6.20}$$

Now by using (5.4) and the calculated initial values of ϕ it is possible to find all the values of the scaling function ϕ. For instance $\phi(0.5)$ can be calculated as follows,

$$\phi(0.5) = l(0)\phi(1) + l(1)\phi(0) + l(2)\phi(-1) + l(3)\phi(-2) = l(0)\phi(1) \tag{6.21}$$

because it is only a function of integers. The next recursion step for instance $\phi(0.25)$ is than only a function of half- and full-integers. In Fig. 6.2 a number of iterations for construction of the scaling function are presented.
In Fig. 6.2(a) the first recursion steps are shown and in Fig. 6.2(b) the scaling function after a great number of recursions. After the calculation of the scaling function the mother wavelet (Fig. 6.2(c)) can be calculated using (5.7), ϕ and its known support width.



(a) (b) (c)

Figure 6.2: DB_2 (a) scaling function after a few iterations (b) relatively smooth scaling function and (c) smooth wavelet

The generation of these wavelets have no real purpose other than to give a visual representation of the scaling functions and wavelets belonging to the filter banks. However the study of T, that generates these scaling functions (and wavelets), gives inside into the convergence and smoothness of the filter bank. There are a number of issues related with T, these are summarized below.

- *Existence*: ϕ exists if T has an eigenfunction ϕ for eigenvalue 1 (Does the eigenvalue problem (6.17) have a solution)
- *Uniqueness*: There is only one eigenfunction (scaling function) ϕ when the Kernel$(T-I)$ has dimension 1 i.e. $rank(T) = n-1$
- *Smoothness*: Every time (5.4) is iterated $\vec{\phi}$ becomes larger and so does T, the smaller the eigenvalues of T the smoother ϕ is (excluding $\lambda = 1$). For instance (6.20) has eigenvalues $\lambda = 1$ and $\lambda = 0.5$ (not very smooth)
- *Convergence*: if T has eigenvalues bigger than 1: the recursion will explode. Multiple eigenvalues $|\lambda| = 1$ will lead to weak convergence. Preferably T must have only one $\lambda = 1$ and the rest of the eigenvalues $|\lambda| < 1$.

Convergence of the filter bank is not always easy to verify and calculation of a smooth $\vec{\phi}$ can be time consuming. Therefore ϕ is often calculated in the frequency domain. This is because in the frequency domain (6.2) can be written down as (Appendix C):

$$\hat{\phi}(\omega) = \prod_{j=1}^{\infty} L\left(\frac{\omega}{2^j}\right) \text{ with } \hat{\phi}(0) = 1 \tag{6.22}$$

Instead of doing time consuming recursion calculations, ϕ can be calculated by a product series in the frequency domain. This also makes the study of convergence and smoothness much easier. In the next section the initialization of a filter bank algorithm is explained. Therefore it is also necessary to calculate the initial values of $\tilde{\phi}$ or better $\tilde{\psi}$.

6.6 Initialization

In Section 3.1 the principle of how to decompose signals is explained. In Section 5.5 it was explained how a filter bank works and which initial coefficients (the expansion coefficients) are needed as input to the filter bank. In the example however, the box function is used, consequently the expansion coefficients are equal to the sample points. In general this is not true at all, from Section 3.1 we know that $x(t)$ in discrete form is defined as:

$$x(k) = \sum_{n} a_m(n)\,\phi_m(k-n) \tag{6.23}$$

The coefficients $a_m(n)$, of the highest level denoted by m, need to be found for every sample point. This is nothing else than (3.7):

$$a_m(n) = \sum_k x(k)\tilde{\phi}_m(k-n) \tag{6.24}$$

For every sample point n an expansion coefficient $a_m(n)$ is filled in at level m (Fig. 5.7). Therefore the assumption that the number of equations is equal to the number of sample points is indeed valid (3.3). The question is now how to convert the sample points to the expansion coefficients. This is done through pre-filtering i.e. a filter needs to be put in front of the filter bank. The first idea, without thinking about the background, is to simply take the scaling filter as the pre-filter. This is wrong, because it is the scaling filter that transforms expansion coefficients from scale m to $m-1$ and not from sample points $x(k)$ to $a_m(n)$. The correct implementation is already described in (6.24); one needs to take the inner-product between the scaling function and the sample points. From Section 6.5 we know that the entire scaling function respectively wavelet is generated by recursion. Nevertheless we are only interested in the values at the sample points at level m i.e. the initial values of the scaling function. This is shown for two scaling functions in Fig. 6.3:

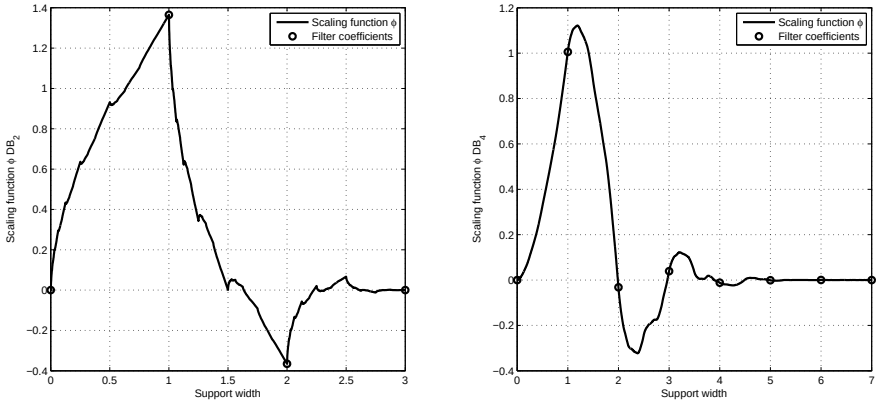


Figure 6.3: Scaling function ϕ of DB_2 and DB_4

In Section 6.5 it was already explained that by solving the eigenvalue problem the initial values of the scaling function can be found. Now let us consider an example using the DB_2 scaling function of which the values can be determined using the method described in Section 6.5. The function ϕ_n is orthonormal for the DB_2 scaling function. Therefore $\tilde{\phi}_n$ can be replaced by ϕ_n.

$$\phi = [\phi(0), \phi(1), \phi(2), \phi(3)] \tag{6.25}$$

It is now easily verified using (6.24) that the pre-filter is defined as:

$$H_{PF}(z) = \phi(0)z^{-3} + \phi(1)z^{-2} + \phi(2)z^{-1} + \phi(3) \tag{6.26}$$

One should remember that in real-time applications the information must always be causal. Consequently pre-filtering will cause a delay, the amount of delay becomes more clear by considering an example. The expansion coefficients for an arbitrary scaling function are calculated for a non-causal and causal signal, here the dirac function.
From (6.6) it is clear that pre-filtering will introduce a delay that depends on the filter length of the scaling filter: to be exact the signal is delayed by $\frac{N}{2}+1$ samples. Also one should remember that (6.24) is an approximation of the continuous integral. These two factors plus the ignorance of decompositions makes a lot of filter bank designers decide to simply put the sample points into the filter bank. This will have undesired effects, but can be acceptable especially when the signal is highly oversampled.

Table 6.1: Approximation of expansion coefficients of $\delta(t)$, DB_2

Correct implementation (Non-Causal)			Filter implementation (Causal)		
$a(-2)$	=	$x(0)\phi(2)$	$a(-2)$	=	0
$a(-1)$	=	$x(0)\phi(1)$	$a(-1)$	=	0
$a(0)$	=	0	$a(0)$	=	0
$a(1)$	=	0	$a(1)$	=	$x(0)\phi(2)$
$a(2)$	=	0	$a(2)$	=	$x(0)\phi(1)$

Some of the consequences of this incorrect implementation are described in [2]. For example if the filter bank is not initialized correctly the location of an event cannot be exactly determined. It will be delayed with unknown delay and the shape will be distorted. Secondly consider a signal that has exactly the shape of the wavelet that is used to analyze the signal. The wavelet coefficients should then only appear at the correct scale (same dilation) and the coefficient should be one. However when using the sample points, the wavelet coefficients will not detect this i.e. the algorithm will not detect what you are searching for. Therefore it is highly recommended to use the correct initialization although there are exceptions related to delay. Also other approximations for the expansion coefficients exists depending heavily on the application (multi-wavelet, optimal time-shift etc.). Some of them are described in [22, 20, 40, 17, 41].

6.7 Vanishing moments, regularity and approximation of signals

The most important property of wavelets are the vanishing moments. The number of vanishing moments is related to the approximation order of functions, the decay rate and smoothness of wavelets. There is a difference between vanishing moments for scaling filters and wavelet filters. Let us consider only the vanishing wavelet moments. A wavelet ψ has p vanishing moments when:

$$\int t^k \psi(t)\, dt = 0 \text{ for } 0 \leq k < p \tag{6.27}$$

From the definition the number of vanishing moments says something about the differentiability of ψ and it is generally known that differentiability is a measure of the smoothness of functions. Next to this the number of vanishing moments are related to the approximation order. A wavelet can approximate polynomials of up to order $p-1$ exactly. The result is that the wavelet coefficients (resulting coefficients after transform) will be equal to zero. For these polynomials often wavelets are derived from filters, this also holds for vanishing moments. In the z-domain vanishing wavelet moments are defined as the number of zeros p in $\omega = \pi$ of the scaling function ϕ,

$$L(z) = \left(\frac{1+z^{-1}}{2}\right)^p \mathcal{Q}(z), \tag{6.28}$$

where $\mathcal{Q}(z)$ is a polynomial with no zeros or poles at $z = -1$. When the orthogonal wavelet is calculated from the scaling filter the zeros shift from $z = -1$ for the scaling filter to $z = 1$ of the wavelet filter. This implies that the wavelet filter has p zeros at $z = 1$, in the z-domain this is expressed as,

$$H(z) = (1-z^{-1})^p \mathcal{P}(z),\ p = \text{number of vanishing moments} \tag{6.29}$$

From here it is easy to show that the vanishing moments indeed are related to the number of differentiators. Let us consider a wavelet with 2 vanishing moments, we know that polynomials of order one $(p-1)$ can exactly be approximated, thus turning the wavelet coefficients zero.

$$H(z) = (1 - z^{-1})^2 \Rightarrow h(k) = u(k-2) - 2u(k-1) + u(k) \tag{6.30}$$

This is true because $h(n)$ is nothing else than the centered difference formula of the second derivative. Consequently taking the second derivative of a first order polynomial will result in zero. If the polynomial is of a slightly higher order than $p-1$, the wavelet coefficients will remain small i.e. smooth signals give small wavelet coefficients whereas non-smooth functions give big wavelet coefficients. However more vanishing moments makes the filter longer which will require more computation and will introduce more delay in real-time.
Regularity is also determined by the number of vanishing moments. The notion of regularity in wavelet theory refers to the order of smoothness of functions [33] in both time and frequency domain [37]. However it remains a difficult concept and it is hard to explain what regularity is exactly. It is only important to remember as the number of vanishing moments is a measure of regularity.
It is also possible to give the scaling function vanishing moments called vanishing scaling moments [8]. The vanishing scaling moments have the same influence on ϕ that the vanishing wavelet moments have on ψ. Vanishing moments not only have an effect on the approximation order but on the approximation of signals in general. The effect of vanishing moments on the signal approximation can be made more clear by analyzing the errors (2-norm) created by the approximation. The approximation error is defined by the projections of $g(t)$ on our space $\mathcal{V}_m$. The projection P^m is defined as:

$$P^m\{g(t)\} = \sum_n \langle g(t), \tilde{\phi}_{m,n}(t)\rangle \phi_{m,n} \tag{6.31}$$

Here $P^m\{g(t)\}$ is the best approximation of $g(t)$ that minimizes the 2-norm. This is different from (3.1) where is assumed that $g(t)$ can be exactly approximated by the functions $\phi_{m,n}(t)$. In (6.31), $\phi_n(t)$ is still continuous in time (DWT); this is not true when applying the DWT through the DTWT. In that case the approximation defined in (6.24) is used. Then $S^m\{g(t)\}$ is the approximation of our signal $g(t)$.

$$S^m\{g(t)\} = \sqrt{2}\sum_n 2^{-m} g\left(2^{-m}n\right)\phi_{m,n}(t) \tag{6.32}$$

Both functions depend on the scale m which is logical because when there are many dilated small functions $\phi_{m,n}$, a signal is easier to approximate than when there are only a few big functions available. The projections with the errors ϵ are presented in Fig. 6.4.

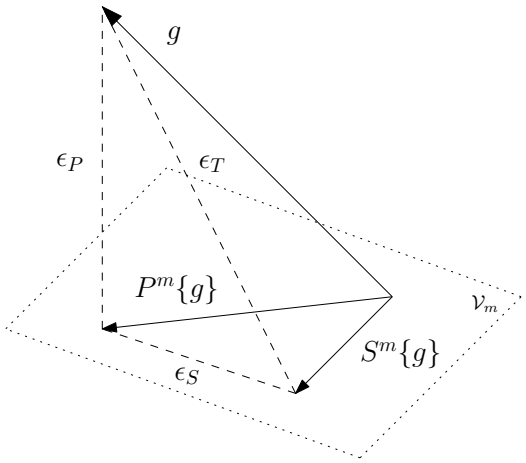


Figure 6.4: Projections of $g(t)$ and their errors ϵ

The error ϵ_P is the error made by approximating the signal by the functions $\phi_{m,n}(t)$. Approximating the continuous $P^m\{g(t)\}$ to the discrete approximation $S^m\{g(t)\}$ introduces another

error ϵ_S. Consequently the total error ϵ_T between $g(t)$ and the practical signal $S^m\{g(t)\}$ is than defined as:

$$\epsilon_T = ||\epsilon_P + \epsilon_S||_2 \tag{6.33}$$

So far we did not deal with vanishing moments. Now it can be proven [36] that the number of vanishing wavelet moments, that are indirectly related to the scaling functions, influence the error as follows:

$$\epsilon_P = ||g(t) - P^m\{g(t)\}||_2 \leq C_1 2^{-m(P+1)} \tag{6.34}$$

and in [34] it is derived that the scaling vanishing moments will influence ϵ_S as follows:

$$\epsilon_S = ||S^m\{g(t)\} - S^m\{g(t)\}||_2 \leq C_2 2^{-m(L+1)} \tag{6.35}$$

The constants C_1 and C_2 only depend on the signal and the mother wavelet. It is independent of scale and the number of vanishing moments. It becomes now clear why giving wavelets extra vanishing moments is so beneficial to the approximation error. The number of vanishing wavelet moments reduces the approximation error for the DWT and the number of vanishing scaling moments reduces the error from DWT to DTWT, which is desirable.

6.8 Aliasing

Aliasing is a well known phenomenon in signal analysis. It also occurs in wavelet analysis and particular in DWT. The operation of downsampling and upsampling introduces another form of aliasing next to the well known aliasing. The well known aliasing occurs when the highest frequency in the analyzed signal is higher than half the sampling frequency. One can imagine that if a signal is sampled with a too low sample frequency other low frequent signals can also be sampled in the same way causing aliasing. This effect is described in almost every signal analysis book. This chapter focuses on the aliasing that is occurring from down- and upsampling in relation to filter banks. If a signal is downsampled the nyquist frequency related to aliasing changes. This operation will cause the spectrum to be stretched out depending on the downsample level. As already known it is very common to downsample by two samples although different downsamplings can be applied. Therefore it is explained in detail for the two sample case, where the original signal is already low-passed filtered to prevent regular aliasing i.e. the nyquist frequency $= \pi$.
In a filter bank signals are split up in a high-pass and low-pass channel. In an ideal case both channels are band-limited. In terms of nyquist frequency this means for the low-pass channel $\omega < \frac{\pi}{2}$ and for the high-pass channel $\frac{\pi}{2} < \omega < \pi$. Now if it is downsampled, the spectrum is stretched and the nyquist frequency is halved. This stretching works in two ways which causes not only the spectrum of interest to be stretched but also the spectrum bigger than $|\pi|$ to be stretched. Now if the spectrum is upsampled again (when doing perfect reconstruction) this aliased stretched spectrum is also compressed again leading to what is called imaging. This is represented in Fig. 6.5.

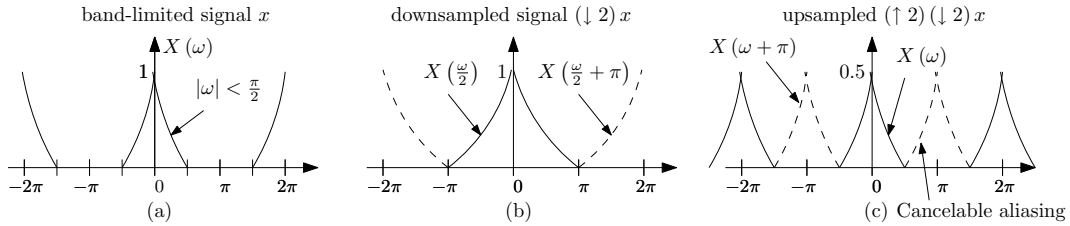


Figure 6.5: Cancelable aliasing for a strictly low-pass signal

This kind of aliasing is called cancelable aliasing because it can be canceled out; if the reconstruction filter is chosen such that is fulfills the aliasing cancelation condition. This effect is

also described in (5.27) where $X(-z)$ is the cancelable aliasing. This kind of aliasing does not cause real problems because it can be canceled out in case of PR-bank. The real low-pass and high-pass filter indeed create a very similar situation as is presented in Fig. 6.5. However it becomes problematic when the signals are not exactly band-limited i.e. the low- and high-pass are not brick wall filters. This means that the low-pas filtered signal still contains frequency content above $\frac{\pi}{2}$ and vice-versa. Although the cause of aliasing is exactly the same, the effect is more severe. The stretched out spectra start to overlap each other; leading to a summation of spectra. This makes it impossible to distinguish the original spectrum. This is represented in Fig. 6.6.

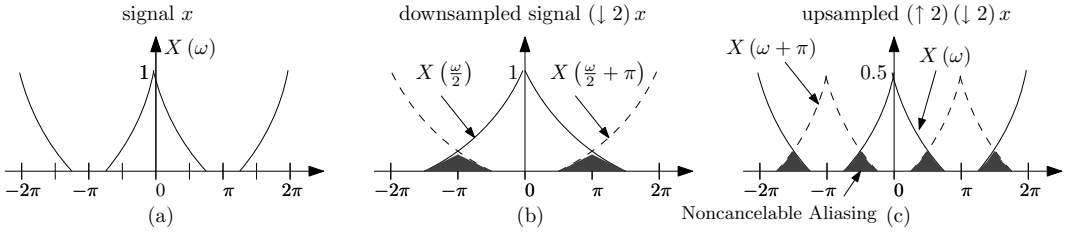


Figure 6.6: Non-cancelable aliasing for a overlapping low-pass signal

This kind of aliasing is not cancelable and is very hard to avoid because it is almost always present. The high-pass and low-pass filters do not perfectly separate the signal in two band limited frequencies. The severity of this effect depends on the amount of overlap and the frequency content on the overlap. There is only one real cure to try to limit this effect and that is by choosing the filters such that they give a better band-limited signal. It is also possible to change the position of the overlap to a region where the overlapping has a very low amplitude compared to the rest of the spectrum and or by changing the amount of downsampling. Simply taking a higher sampling frequency as is recommended in normal aliasing does not solve anything. Actually it makes it worse, higher sampling frequency's often means taking more filter bank levels to separate the same frequencies.
This effect occurs when the spectra overlap each other i.e. the halfband filters, which split the frequency spectrum in half will overlap each other around the nyquist frequency. The wavelet filter is the high-pass filter and therefore automatically aliasing will occur at the frequencies in the low-pass part. This is logical and therefore these aliased frequencies are discarded. In theory problematic non-cancelable aliasing can only originate at the overlapping frequency bands. This is shown in Fig. 6.7.

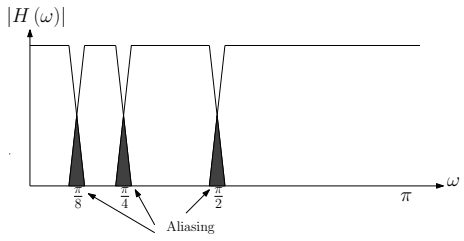


Figure 6.7: Frequency bands where aliasing can occur in theory

This theory gives an idea about one level of the filter bank, however it is also important to understand what will happen when the created aliasing is introduced into the next channel of the filter bank. This can be quit problematic because aliasing is also introduced at lower frequencies where it is not obvious. To get a better idea an example is elaborated for a two channel filter bank with four decomposition (Appendix D). The result is shown in Fig. 6.8.
In Fig. 6.8 a DB_4 is used and a signal with two frequencies is analyzed and synthesizes. The resulting error is in the right figure. One can see that aliasing can be devastating to the signal because every new step introduces more aliasing.

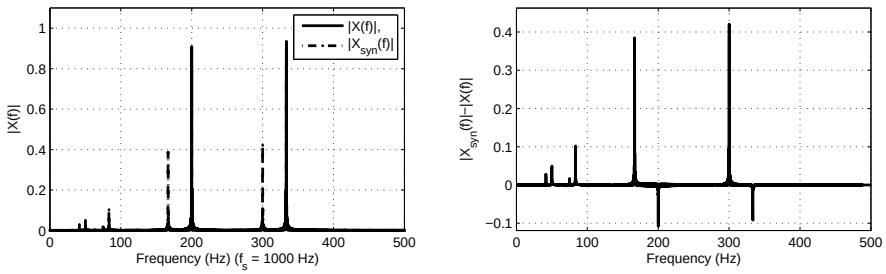


Figure 6.8: Original spectrum and synthesized spectrum with aliasing from DB_4-filters

Summarizing, there are two kinds of aliasing: the classic aliasing and the aliasing caused by down- and upsampling. The classic aliasing can be simply resolved by taking a higher sampling frequency. The aliasing caused by down- and upsampling can be separated in two parts: cancelable and non-cancelable. The cancelable part does not cause any problems. The non-cancelable aliasing can only be solved by taking more brick like filters or sometimes changing the area effected, but this is often difficult. Also the effect of aliasing transferred to the next level of the filter bank should be taken into account. The problem of aliasing cannot really be solved when downsampling is involved, but it is good to be aware of it.

6.9 Overview of the different types of wavelets

It is difficult to give a full overview of all available wavelets. Every book focuses on a certain part of the wavelet family. However in the first place one wants to know which wavelets are available. A first idea about how to categorize wavelets is given in [1].

- **Orthogonal wavelets with FIR filters**
- **Biorthogonal wavelets with FIR filters**
- **Orthogonal wavelets without FIR filter, but with scale function**
- **Wavelets without FIR filter and without scale function**
- **Complex wavelets without FIR filter and without scale function**

The first three wavelet types can be used without any real issues in the filter bank i.e. they are discrete wavelets. The orthogonal wavelets with FIR-filters include: the Haar, Daubechies, Symlets and Coiflets. A lot can be found about these wavelets and they are considered to be the most important types. More about the design and properties can be found in Appendix A. The biorthogonal wavelets are also discussed in this report. They give more freedom but still fit well in perfect reconstruction. The most important type of these filters are the Bior-splines e.g. the Triangle wavelet introduced in Section 5.3. They are all symmetric but often still use similar design methods as the orthogonal design made by Daubechies. The orthogonal wavelets without FIR-filters lack compact support. This does not have to be problematic because it is also possible to use IIR-filters. However if the filter is not designed in terms of filter coefficients it will be difficult to apply in PR-filterbank. IIR-filters lack compact support (their response extends from $-\infty$ to ∞), therefore in practice only offer near perfect reconstruction. The last two wavelet transforms belong to the class of continuous wavelets making it difficult to apply them in a DWT. It is possible to analyze them by discretization but perfect reconstruction is almost never possible. These last two groups include the Morlet, Mexican Hat wavelets and complex discrete wavelets. An overview of the most important wavelets is given in Fig. 6.9. This overview also includes multiwavelets that are a natural extension to scalar wavelets. Instead of using one scaling function ϕ and one wavelet function ψ, multiple scaling and wavelet functions are used, hence the name multiwavelets. More about them can be found in Appendix B.

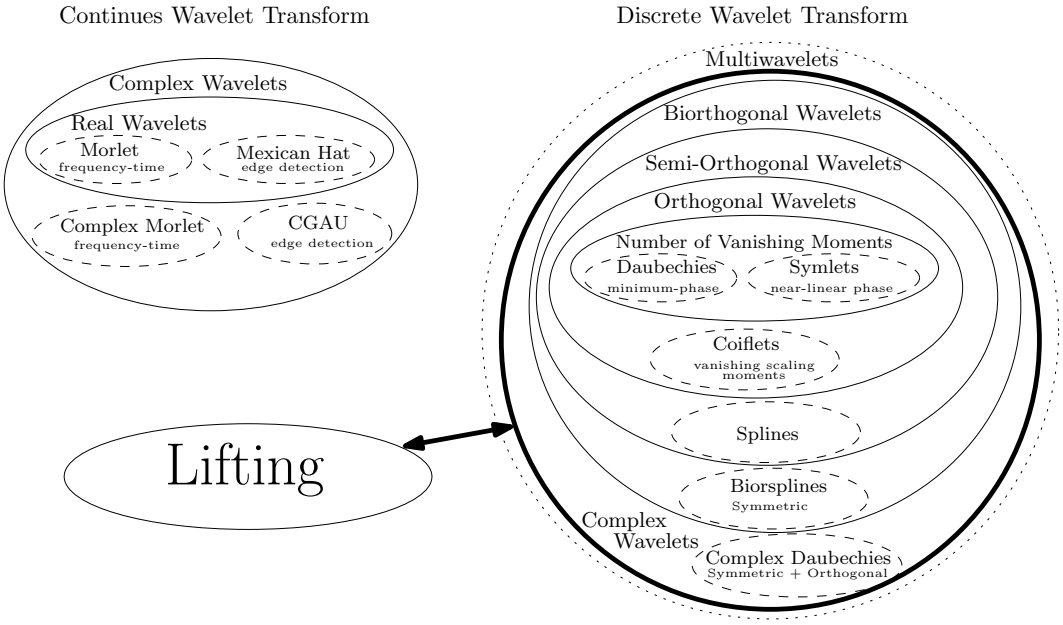


Figure 6.9: Overview of the most important wavelets available

The overview separates the wavelets in two main groups the CWT and DWT. Also a step called lifting is included. Lifting is very often used but unfortunately is not very useful in real-time applications. Lifting couples the two output channels with a filter, making it possible to design the wavelet filter with much more freedom because it can be corrected with this filter.

Chapter 7

Main Applications of Wavelets

In the previous chapters the main properties and considerations belonging to the DWT have been discussed. In this chapter the most important applications are discussed. The two most important applications are compression and denoising of signals. Another application of wavelets is edge detection and it is therefore also discussed in more detail. Another field of wavelets that is not very progressed yet is the pattern persuit, however, it can be an important application of wavelets inside the control community. In the first section we discuss compression because it is the most common application and does not require any manipulation of the resulting wavelet coefficients.

7.1 Compression

The most important application of wavelets is without any doubt the compression of images. In this section we try to give insight into the principles whereupon compression is based. The main idea of compression with wavelets, and also Fourier based methods, is that the number of non-zero, wavelet and expansion coefficients necessary to store it, is reduced compared to the original signal (sample points). This means for wavelets that if the (reconstruction) wavelet resembles part of the signal, this part can be stored in one coefficient. If the signal is stored in wavelet coefficients, it is very important that the signal can be reconstructed from these coefficients i.e. the operation should be invertible. We already know that a signal can be reconstructed if the conditions for Perfect Reconstruction are fulfilled (5.31).
Currently the most important compression technique is the Discrete Cosine Transform (DCT). It expresses the signal (or image) in terms of sinusoid's and it is the method whereupon JPEG is based. Moreover the DCT can also be seen as a method that is related to wavelets because a finite discrete transform always involves windowing to handle the bounds of the signal properly. Expressing a signal in terms of sinusoid's is a good choice because of the orthogonality, which gives a perfectly conditioned transform. A good compression is only achieved when the signal is relatively smooth or for a transform with loss that the signal can be described well. Wavelets can be seen as an extension of the DCT as is described in Section 3.3.
Compared to the DCT, wavelets offer almost an infinite number of sub-functions for compression. Wavelets for good compression should fulfill a number of properties that are already discussed in this report. Generally the important contents of signals and images are smooth, think of images where often there are areas with a nearly uniform color or small gradient. Therefore it is natural to assume that the signal contains many smooth parts. The properties of the most common wavelets are related to smoothness and approximation of smooth signals. The most important properties that make wavelets describe smooth signals well are the vanishing moments and regularity described in Section 6.5. As we know the vanishing moments describe the approximation order of functions. For example a wavelet with one vanishing moment can describe a signal that is locally smooth of order zero; so if we for instance have an image that has a grey-plane the corresponding wavelet coefficients will be zero i.e. if a signal is locally smooth the coefficients become zero and do not have to be stored.

Up till now we discussed the loss-less compression, however, often most wavelet coefficients are nearly zero. However, often these small coefficients do not contribute to important information and by removing them the difference is negligible. Therefore it can be concluded that the main idea of compression is to find a set of sub-functions (here wavelets) that can describe the relevant information in the original signal with a minimum number of non-zero coefficients. In the literature the term sparse is generally used to describe a representation of the signal with a minimum number of non-zero coefficients. Generally the next step is to apply a number of operations on the coefficients to further compress the signal, for instance quantization to further compress it (this will result in loss of information). This will not be further discussed here.
There are also some properties indirectly related to the previous discussion. One of them is the compression and decompressing time. This is related to the filter lengths of the filter bank and the number of decomposition levels. Short filters take less time but their degree of freedom is reduced e.g. the number of vanishing moments are limited. Longer filters can much better approximate signals but take more calculation time. In addition, they have another advantage compared to the short filters: their cut-off frequency. As we know not cancelable aliasing occurs if the signal is down-sampled and the severity depends on the cut-off frequency. Next to this is the conditioning of the transform that is ideal for orthogonal wavelets, however, for vision we know symmetry is important (biorthogonality).
We have explained the concept of compression, however, it remains unclear what kind of wavelet fits best to our signal. There is no wavelet that always gives the optimal sparse representation of the signal. There is always a number of wavelets that are ideal for a certain application, such a set of wavelets is called a dictionary. Here we gave the example of smooth functions that fit well to images. However one could think of other signals that are full with discontinuities, which other than smooth wavelets (dictionary) can compress this signal much better.
Generally there are two possibilities for selecting wavelets for compression: selecting a predefined dictionary to compress the signal or search for filters that give the near-optimal sparse representation of your signal. Already many dictionaries have been developed that describe a great number applications well. One of the most famous compression algorithms is the JPEG2000 [10]. It uses two biorthogonal wavelets, one with loss of information and one loss-less, to be precise two variants of the CDF (Cohen-Daubechies-Feauveau) wavelet. Another application is the FBI fingerprint bank, where it is important to capture the important feature of a fingerprint and not every detail of the picture unrelated to the fingerprint recognition. Therefore by designing a specific DWT and only storing the important coefficients it is possible to achieve very high compression ratios (20:1) [7]. In many compression applications a predefined transform or dictionary is already chosen. However it depends heavily on the signal itself which wavelet compresses the signal best. Therefore it is also possible to find the optimal wavelet respectively filter coefficients for an individual application.
Finding the optimal wavelet respectively filter coefficients has become a popular research subject whereby the wavelets are chosen which minimize the error. The criteria that define the optimal problem are very broad they range from finding a spline that minimize the number of non-zero coefficients of an individual problem, to difficult genetic algorithms that find the minimum for a whole set of signals [18].

7.2 Denoising (Wavelet shrinkage)

The second most important application of wavelets is the denoising of signals. Denoising is classically done with the help of low-pass filters, unfortunately, they also remove the non-smooth features of a signal consequently leading to rounding-off edges and the removal of high-frequent information. With wavelets it is possible to retain such features and still remove much of the noise. We already learned that it is possible to decompose and reconstruct signals. Consequently if we can identify the wavelet coefficients that belong to the noise terms, we can remove them and reconstruct the signal, yielding a denoised signal. Therefore it is necessary to determine which wavelet coefficients represent the noise in a signal. This depends on the chosen wavelet, the method of removal and the classification of the different coefficients.

First of all a suitable wavelet needs to be chosen, which depends on the properties of the signal. This means when for instance a sawtooth signal is used, take a wavelet that can resemble a sawtooth well, so that the non-zero coefficients only represent the noise, thus by removing them, the signal is denoised. It is important to note here, that although wavelets are linear operations, the denoising process is not.
Reducing the coefficients works well when one knows important properties of the signal, however, these are often unknown. Therefore often the wavelets that have been designed based on smoothness properties are chosen. We already know that locally very smooth signals give wavelet coefficients zero for most wavelets. Conversely non-smooth signals give coefficients very different from zero, that has a bigger amplitude then the coefficients generated by noise. Thresholding is the mostly used denoising technique, which removes all coefficients below a certain threshold value λ, which is set just above the noise level so that all important information is retained. In wavelet analysis this is done at many different scales whereby from every level the coefficients are retained above the local noise level. This has a big advantage compared to low-pass filtering, where all high-frequent information is removed. Instead high frequent information that is above the noise can be retained and therefore remains in the reconstructed signal.
In general there are two types of thresholding, i.e. the hard-thresholding

$$d_{m,hard}(n) = \begin{cases} d_m(n), & \text{if} \quad |d_m(n)| \geq \lambda \\ 0, & \text{if} \quad |d_m(n)| < \lambda \end{cases}$$

and the soft-thresholding operation,

$$d_{m,soft}(n) = \begin{cases} d_m(n) - \lambda & \text{if} \quad d_m(n) \geq \lambda \\ d_m(n) + \lambda & \text{if} \quad d_m(n) \leq -\lambda \\ 0 & \text{if} \quad |d_m(n)| \leq \lambda \end{cases}$$

The main difference between hard- and soft-thresholding is that after hard-thresholding the values above the threshold are unchanged, whereas the original coefficients are reduced with the threshold value when soft thresholding. In most cases λ is chosen just above the noise level so that all noise is removed. Nevertheless it happens regularly that some of the noisy coefficients are retained, especially in the smaller scales because of a bad signal-to-noise ratio; that is often smaller in these scales because the deterministic signal is normally a low-frequent signal. When a hard-threshold is applied some noise peaks are present in the reconstructed signal. Although originally part of the noisy signal, it becomes a local noise component in the signal whereas around it no noise is present anymore. The introduced noise could even become bigger than it originally was. This is possible because a noise component can be built out of different wavelet coefficients (over different scales), some of them are partly counteracting each other. So by deleting some and keeping others it could happen that the noise component becomes locally bigger then it originally was. This effect of introducing sudden noise peaks can be reduced by taking a soft-threshold. The soft-threshold reduces all coefficients with the original threshold. It is expected that the noisy coefficients are close to this threshold and are therefore reduced in amplitude. Nevertheless this means also that all coefficients are reduced and the signal is slightly changed, resulting in the signal becoming somewhat smoother. In addition, the difference between taking a hard- and soft-thresholding only makes a big difference in the smaller scales because in the bigger scales the λ that is chosen just above the noise level is relatively small compared to the wavelet coefficients of the deterministic signal (which is low-frequent) and therefore the reduction of amplitude off all coefficients has a much smaller effect. This is the reason why the soft-threshold is the most popular in multiscale wavelet denoising. The effect of selecting different wavelets and threshold methods is also presented in the following figure.
On the left of Fig. 7.1 a signal has been denoised with two different wavelets using exactly the same threshold method and criteria. The advantage of wavelet denoising becomes immediately clear. The original signal has almost entirely been denoised and the edge is in the case of the Bior2.2 (Fig. 6.1) nicely been preserved. When we look at the signal denoised with a

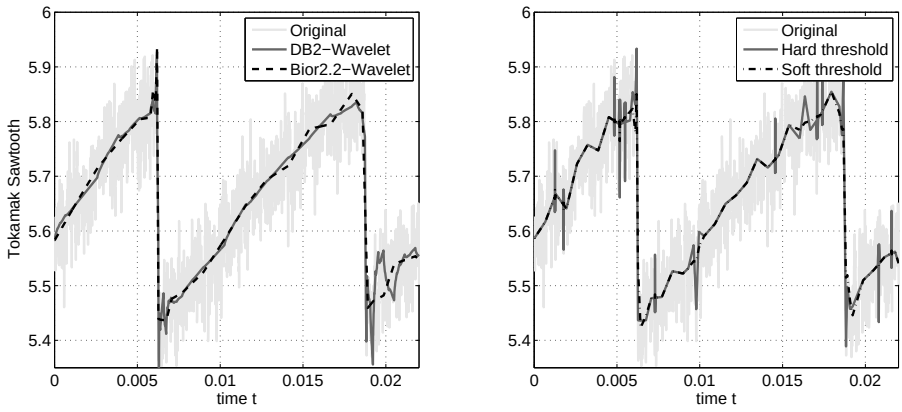


Figure 7.1: Denoising of a sawtooth signal selecting two different wavelets (left) and different thresholds for the Bior2.2-Wavelet

DB_2-wavelet (Fig. 6.2), we see around the edges some strange shapes. However, if we look more carefully we recognize the shape of the DB_2-wavelet, especially after the second crash. This is a consequence of the very different shape of the DB_2-wavelet compared to the signal: the signal is decomposed in many small wavelet coefficients that eventually are removed leaving one or two big coefficients in the signal. This problem does not occur when a Bior2.2 wavelet is selected: this (reconstruction) wavelet is build from triangles, which can present this signal much easier.

On the right of Fig. 7.1 the difference between hard- and soft-thresholding is shown. There is hardly any difference, except the sharp spikes that are produced by the noise above λ.

More important than the method of thresholding is the choice of the threshold value λ. This depends on many different properties and the desired level of denoising. In general the λ is chosen such that the threshold is just above the noise level or in terms of threshold design, the risk of retaining noise coefficients should be minimum. These design methods make use of estimator based methods. It is outside the scope of this report to discuss them in detail here, so we discuss only the most important threshold selection methods. The most famous threshold value is derived by Donoho, which is known as the fixed-threshold [14] and is defined as:

$$\lambda = \sigma\sqrt{2log_e N}$$

where σ denotes the standard deviation of the noise and N denotes the number of sample points. The threshold is dependent on N because when Gaussian noise is considered the change of a high noise amplitude becomes bigger when we have more samples. This λ gives an upper-bound and is therefore often considered to be conservative. Therefore other popular selection methods exist, for instance the Stein's unbiased risk estimate (SURE), which needs to be minimized [15].

$$\text{SURE}(\lambda) = N + a(\lambda)(\lambda^2 - 2) + \sum_{k=a(\lambda)+1}^{N} d_k^2$$

where $a(\lambda)$ are the number of coefficients less than or equal to the threshold λ, and d_k are the wavelets coefficients that have been rearranged into an increasing series. This threshold is less conservative and works generally much better in environments where the noise is relatively small compared to the signal. However, when the noise is relatively high the fixed threshold works better. Another popular threshold is the minimax threshold, where a number of functions are compared to the signal and the minimum of the maximum mean square error obtained for the worst function in a given set is used to choose the threshold.

The threshold values can also be chosen to be different for local regions, for instance when noise is much less in a certain scale; when important features are present that need to be preserved; or when noise levels change in time. An example of the necessity of a feature that needs to be preserved is an edge in a signal or picture. We already know that when it is necessary to preserve edges, it is possible to detect them in a different scale and lower the threshold in the other scales in the same neighborhood. This ensures that around the edge it is better sustained, however, the noise also remains. In Fig. 7.2 a number of hard-threshold choices are compared with each other.

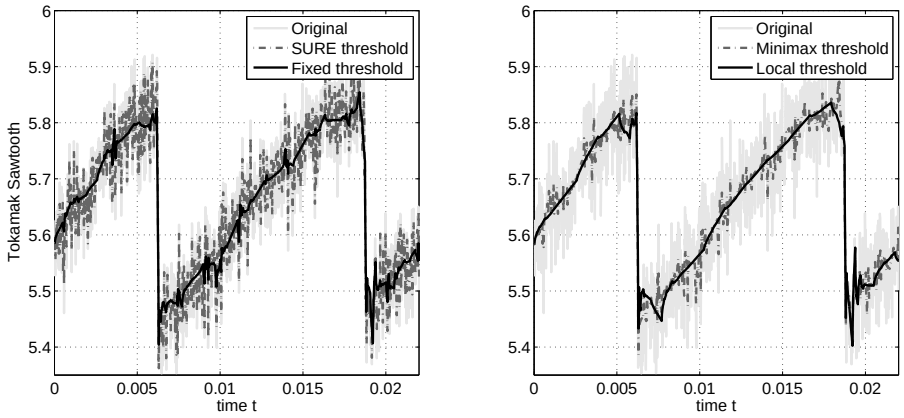


Figure 7.2: Signal denoised with a DB2-wavelet with a different choice of thresholds

Fig. 7.2 presents the denoised signal where only the threshold λ has been varied, which is a constant over one scale except for the local-threshold where it is locally reduced around the crash. The fixed-threshold is the most conservative threshold although most noise is removed, generally the crash will also be smoothed more. The SURE threshold leaves a fair amount of noise because its threshold is generally lower. The minimax performs much better but it takes more effort to calculate and does not always perform better. The local-threshold, here tuned manually, cannot really be compared (because it is not automatically generated) to the rest but it shows that it is possible to have a sharp crash and still have a good denoising away from the crash.
There also exists non-linear filtering techniques that can be used for denoising, for instance median-filtering: the median is calculated from a number of neighboring wavelet coefficients and all these coefficients are then replaced by the median. It is very effective against speckle noise (lasers, images), often preserving edges in images much better than linear techniques [5, 16].

7.3 Edge detection

Edge detection in images is a classic problem. Therefore many edge detectors have been designed for signals and images. Apart from some special methods, they are based on derivative methods. Generally these methods are limited to the first and second derivative methods. An edge is generally a non-smooth part in a relatively smooth neighborhood. Consequently its derivative gives a higher response than the surrounding derivative. For the first-derivative this is a maximum and for the second derivative a zero-crossing surrounded by a maximum and a minimum. Famous gradient operators for edge detection are the Roberts, Prewitt, Sobel and Fri-Chen edge detectors. The two most famous designs are the Canny edge detector [9] and the Marr-Hildreth [25, 19]. They are both based on smoothing with a Gaussian-kernel, where Canny takes the first derivative and Marr-Hildreth the second derivative. These methods are so popular because it has been proven that based on the criteria of localization, signal-to-noise ratio (SNR) (response on edge) and a unique response on one edge, these edge detectors are

optimal for step-edges. However, the criteria of localization and SNR cannot simultaneously be optimized due to the Heisenberg uncertainty principle. Therefore always a choice between good localization and a good response on an edge needs to be made. Here wavelet theory and especially multiresolution offers a solution. Instead of only analyzing with a translated edge detector, it is also dilated. It can be done on multiple scales changing the uncertainty between localization and response just like time and frequency with the morlet wavelet. In this way it is possible to detect edges with more certainty where the edge occurred and still get a high response. This is done through the so called maxima-chaining and zero-crossing chaining. In principle almost every classic edge detector can be used in the wavelet setting, also wavelets with a few vanishing moments are popular to choose (Haar, DB_2 and even DB_3). The in general best working edge detectors are based on the Canny and Marr-Hildreth edge detectors. In CWT these edge detectors can be directly applied because they are based on continuous functions, however, for the DWT they need to be designed. This is done using B-splines, which can approximate these hermite functions quite well using scaling and wavelet filters. Generally a redundant DWT filter bank is used, which means that downsampling is omitted. This gives a higher resolution but also the number of operations increase. In [23] it is stated and derived that by chaining the maxima we can find back the edge when it is unique in its neighborhood (singularity). An example of such a such an algorithm can be found in [24], it is also shown in Fig. 7.3.

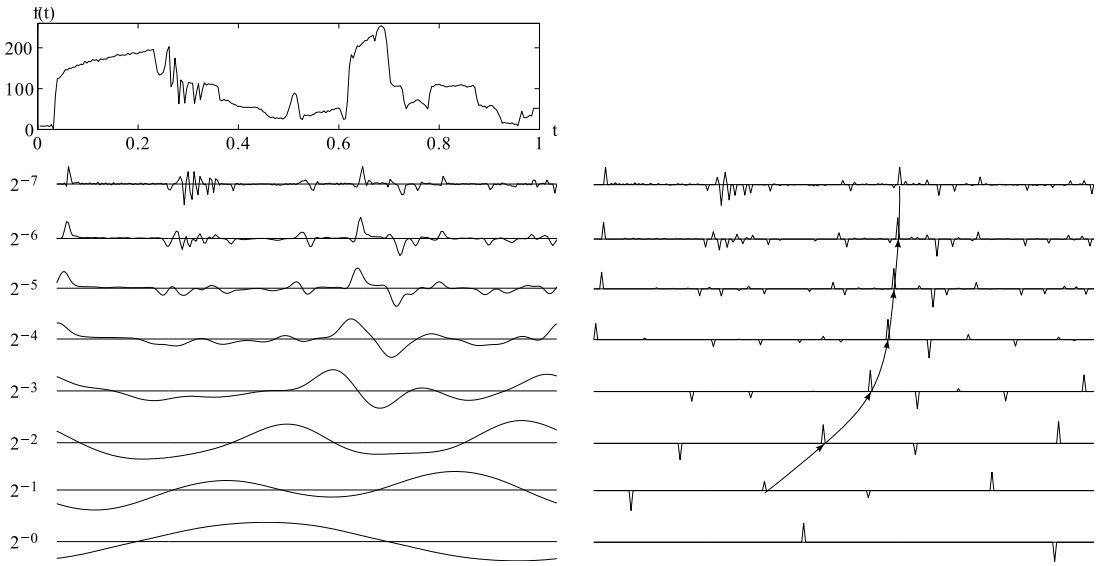


Figure 7.3: Dyadic wavelet transform computed at several scales with the quadratic spline wavelet (left) Modulus maxima of the dyadic wavelet transform (right) [24].

In the scale, 2^{-7} there are still many maxima so its hard to determine which belongs to the bigger edges. In many applications this is even more difficult because of the noise. However the maxima become less if the scales move up and on a certain moment only the big edges remain. However these maxima have become very uncertain and can drift far away from the real edge. Therefore by chaining them back we can find the location of the original edge. Here it has been shown for the first hermite function. The uncertainty of every scale is directly related to the width C of the wavelet (detector). In a formal way this chaining back to a singularity can be described with Lipschitz as also can be seen in Fig. 7.4.

Here we explained the maxima chaining method for the first hermite function. In Section 4.3 we actually showed it for the second hermite function, which is also known as the mexican hat wavelet. Although it is more difficult to chain the zero-crossings, there are actually three lines that can be chained, namely the maxima, the minima and zero-crossings. This idea of chaining is the very core of the wavelet analysis of edges. However the method of chaining varies, for instance they can be chained using fuzzy-logic whereby some maxima are given more

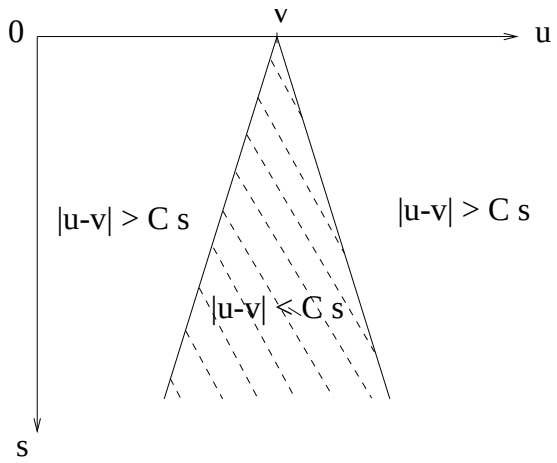


Figure 7.4: Cone of uncertainty, with s scale and v location of singularity or edge [24]

importance than others [4, 30]. In addition complex wavelets are used, where the modulus maxima need to be chained [35]. The method of chaining is so popular because it can also be extended to two dimensions, although in 2D the maxima chaining that is presented here [23] under-preforms compared to other methods, mainly because of the directionality of edges in images, which is beyond the scope of this report.

7.4 Pattern recognition (matching pursuit)

In Chapter 3 we already learned that wavelets describe a given signal in terms of its sub-functions, that are translated and dilated over the signal. Therefore if we can find a sub-function that is very similar to the searched pattern we can detect it, because it will give a bigger wavelet coefficient compared to its surrounding coefficients. This comparable to the Fourier coefficients of a noisy sinusoidal: the frequency corresponding to the sinusoidal will have a big Fourier coefficient and its surrounding Fourier coefficients will be very small because they are only generated from the noise. It is important to remember that it is not the analyzing function that should resemble the pattern but its reconstruction function ψ i.e.
The coefficients $d_{m,n}$ give a representation of $\psi_{m,n}$ in $x(t)$, which are dilations and translations of one function ψ. So ψ should be chosen such that it resembles the pattern, thus its biorthogonal function $\tilde{\psi}$ should be used to find the coefficients. There are a number of methods whereby this $\tilde{\psi}$ could be found. The most straightforward method would be by using the biorthogonality property (6.16). In discrete sense it means taking the inverse function. This is a difficult process and often results in numerical problematic wavelet transforms. Therefore an alternative could be orthogonalizing ψ with for instance Gram-Schmidt [24]. Wavelets that are orthogonal to each other have a number of advantages next to becoming numerically better conditioned. The most important improvement is that the reconstruction function and analyzing function are the same. If the function is also orthogonalized over its scales than the coefficients turn to zero on the other scales if it resembles the pattern in one scale perfectly e.g. if our wavelet is equal to the function on $s = 1$, the other scales except $s = 1$ have wavelet coefficients zero. This is presented in Fig. 7.5, where also another problem is encountered; that of the DWT of this wavelet not being shift-invariant.
In Fig. 7.5 the Haar wavelet is used to detect a function that is similar. As we know the Haar wavelet is an orthogonal wavelet. The first pattern of $x(t)$ is without any problem detected because $d_{1,0}$ is the only coefficient that is turned non-zero. In addition all the other functions are orthogonal compared to the searched pattern. Nevertheless the second pattern is more problematic this because it is shifted by a half translation compared to the wavelet at its corresponding scale. Although this Haar-pattern is detected it is now spread over different

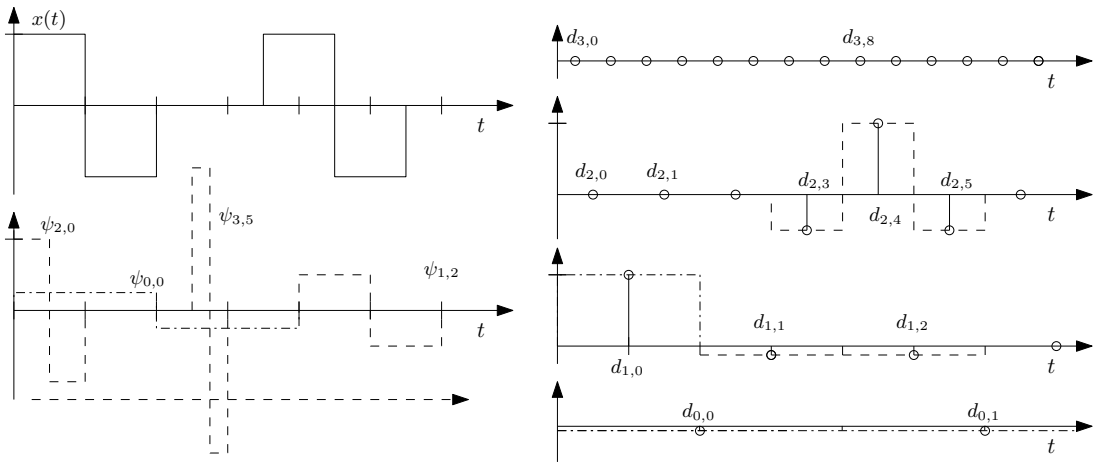


Figure 7.5: Haar wavelet analysis and the problem of not being shift-invariant

scales. Although it is clear that it belongs to the scales $m = 1$ or $m = 2$, the exact shape and scale are not easily interpret anymore. This problem can be solved by making either the wavelet shift-invariant [6] or by applying a redundant DWT. Finding a $\tilde{\psi}$ that is shift-invariant can solve this problem but again the wavelet needs to be adjusted, which could lead to errors. A redundant DWT can be seen as removing the down-sampling in the filter bank, and adjusting the filters correspondingly. Making the DWT redundant is a much easier process, the wavelet transform now also includes convolutions of smaller translations. However, the consequence is that now also the coefficients of other scales can respond on this pattern, which should be taken into account. This will lead to much more calculations and actually is comparable to a discredited CWT.
It is important to note that pattern recognition is mainly developed for compression, where it is used to find patterns in signals, which can be expressed by only a few non-zero coefficients. Also the pattern recognition does not only involve wavelets or multiresolution. It could be a great number of functions. Most often the patterns are found by minimizing some residue that is left after the transform for an orthogonal sub-function this has following form. The vector Ψ_{p0} of a possible set of vectors Γ, where Ψ_p are the selectable wavelets, that approximates the signal best:

$$\Psi_{p0} = \arg\max_{p \epsilon \Gamma} |\langle x(t), \Psi_p \rangle|$$

$$Rx(t) = x(t) - \langle x(t), \Psi_p \rangle \Psi_p$$

This leads then to following form:

$$||x(t)||^2 = |\langle x(t), \Psi_p \rangle|^2 + ||Rx(t)||^2$$

This is because for an orthogonal system the 2-norm remains the same so by minimizing the term $||Rx(t)||^2$ we are maximizing $|\langle x(t), \Psi_p \rangle|^2$ [24]. It becomes clear that pattern recognition is a complex field especially when handling biorthogonal functions and examples of wavelets designed for pattern recognition are limited. Only in an orthogonal case there is a clear optimization problem. Good examples are hard to find, some examples of pattern recognition are found in [3].

7.5 Conclusive remarks

In the beginning of this chapter, the principles on which compression is based are explained. It becomes clear why wavelets are designed in the way described throughout this report. A signal or image should be described with as little as possible non-zero coefficients. Also the

methods described in the last section, about matching pursuits, are actually mainly used for this purpose. Denoising is another important application of wavelets. However in real-time it depends on which scales the noise is present. It is possible to remove high-frequent noise but low-frequent noise would probably induce too much delay. Edge detection is also an important application of wavelets. As we have described the great advantage of wavelets is that they clearly show that multiresolution offers a chance to help overcome the uncertainty problem between localization and response in regular edge detectors. The selection of the right wavelet is often a difficult process. In the selection procedure play the purpose and main properties discussed in this report an important role. Therefore is in Fig. 7.6 a short overview given of all wavelets discussed in this report (and some more), together with their applications and properties.

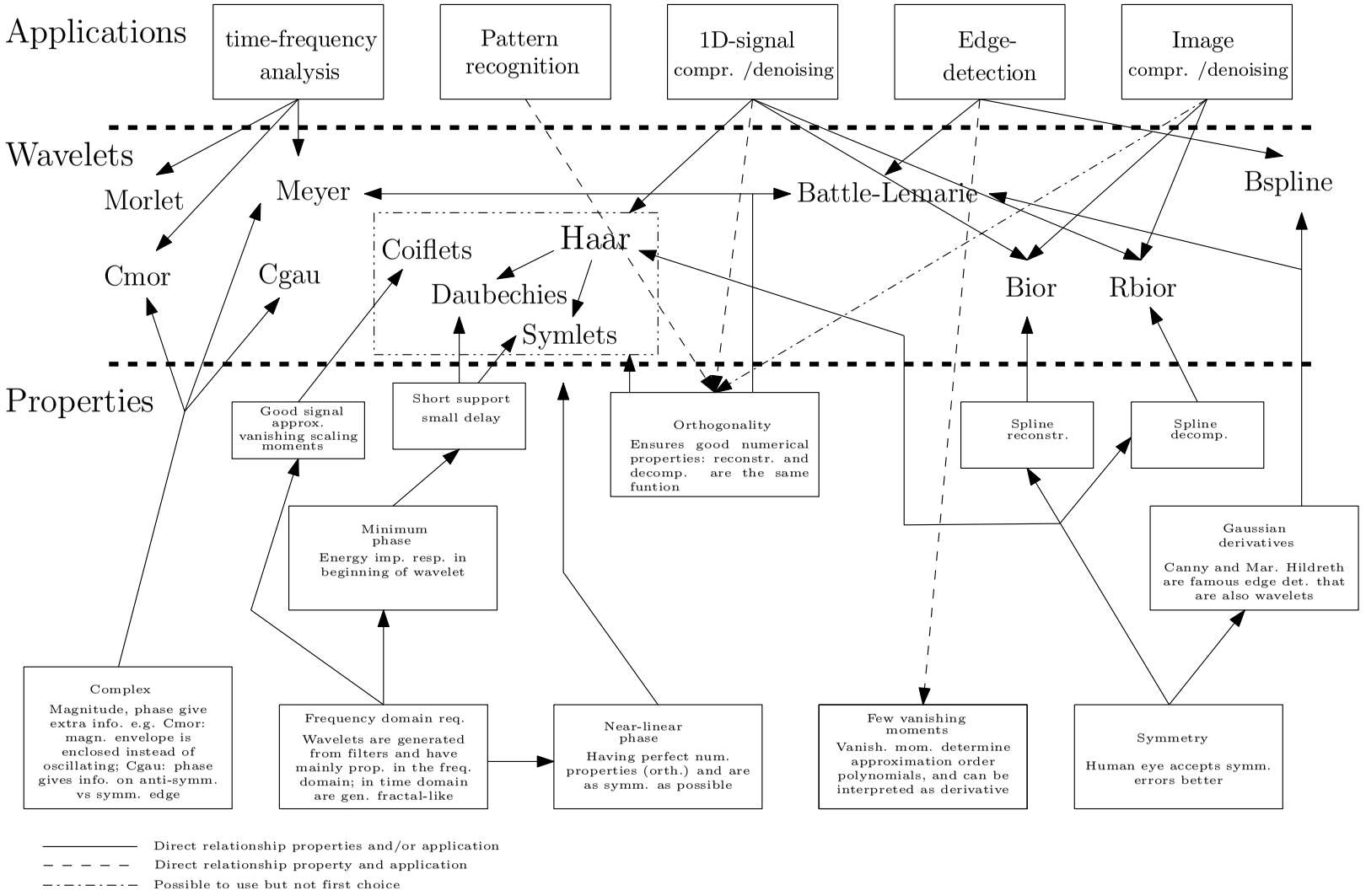


Figure 7.6: Wavelet selection diagram

Chapter 8

Conclusions and Recommendations

In this report a short overview is given about wavelets and its properties. It has become clear that a wavelet is a type of signal analysis tool, that in essence is not so different from Fourier analysis. It still obeys the standard theories on signal decompositions and approximation of signals. Nevertheless the comparison between Fourier analysis and wavelet analysis revealed that there are still some major differences. Wavelet analysis offers multiresolution which is the biggest advantage of wavelets. Depending on the wavelet it either offers time-frequency information (Morlet) or the opportunity to study the same effect on multiple scales (edge detection). The Continuous Wavelet Transform makes it possible to calculate all the wavelet coefficients which represent the desired information on multiple scales. However wavelets cannot only be used for analysis of signals but also for the manipulation of signals i.e. filtering. Although it should theoretically be possible to reconstruct a signal, in practice it is often difficult and inefficient. Therefore another wavelet transform was introduced, the Discrete Wavelet Transform, which is much more than only a discrete form of the Continuous Wavelet Transform. In practice continuous wavelets often do not even fit in this setting. Therefore a whole new set of wavelets have been designed that are good in decomposing, changing and reconstructing a signal. All these wavelets come together with a scaling function, which helps to make the calculations efficient. Despite the fact that the Discrete Wavelet Transform is considered the main application of wavelet transforms, in practice it is the Discrete Time Wavelet Transform that is applied. This transformation from Discrete Wavelet Transform to Discrete Time Wavelet Transform turns continuous time into discrete time, transforming the DWT into a filter bank. This has again consequences for correct approximations because the expansion coefficients need to be approximated again. Nevertheless it is possible to calculate wavelets by simply coupling a set of filters together (filter bank); this also can include a pre-filter. Filter banks need downsampling to reduce the number of calculations and downsampling is used to impose multiresolution naturally. Therefore downsampling is a very good step towards efficiency but introduces aliasing into your signal, which can have serious consequences.

Although there exist a great number of wavelets, this report focusses on the discrete wavelets. In general when selecting a wavelet the first question is what kind of features to extract. Continuous wavelets range from very simple wavelets as for instance the Morlet and Mexican Hat wavelet, to more complex ones in the true sense of the word, Complex Morlet and Cgau (complex gaussian). The discrete wavelets focus on noise reduction and compression of images. Therefore the wavelets are designed to separate frequency bands that can be thresholded to reduce noise. Discrete wavelets are also good in separating smooth signal parts from non-smooth parts. This is a consequence of the vanishing moments that determine the number of derivatives taken on the analyzed piece of signal. Therefore edge detection is another possibility with these kind of wavelets, because edges are generally non-smooth features that will result in big wavelet coefficients. There is such a great variety in discrete wavelets that it is hard to get insight in which one to select. Therefore it is recommended to make a list of requirements

and select your wavelet based on that. The main requirements that can be thought of are what kind of non-smoothness one needs to detect (vanishing wavelets moments). The approximation order of the signal, if it is good enough or needs improvement (vanishing scaling moments). Which frequency bands need to be separated (downsampling, type of filter bank). The filter length also plays an important role; long filters give the chance of high number of vanishing moments and good cut-off frequency that helps reducing aliasing. On the other hand it will cause more delay. A delay is also caused by pre-filtering, both can be problematic in real-time applications. In this report the relationship between the different filters (orthogonality versus biorthogonality) is also explained. From a computational point of view orthogonality is good but if symmetry is desired, for instance in images, biorthogonality is used. The applications presented in the last chapter give an idea about how the wavelet properties that we learned are related to its applications. The two most important applications are clearly compression and denoising. Edge detection also plays an important role in wavelet theory. Nevertheless the application of pattern detection that would be very useful for control is not progressed enough and the literature is very limited on this subject. Although there are more properties and applications this report gives a general idea of what the important properties of wavelets are and how to select the proper wavelet for your application.

Bibliography

[1] *Matlab documentation, 1984-2008, The MathWorks, Inc.*

[2] P. Abry and P. Flandrin. On the initialization of the discrete wavelet transform algorithm. *IEEE Signal processing letters*, 1:32 – 34, 1994.

[3] Paul S. Addison. *The illustrated wavelet transform handbook : introductory theory and applications in science, engineering, medicine and finance.* Institute of Physics, Bristol, 2002.

[4] A.S. Akbari and JJ Soraghan. Fuzzy-based multiscale edge detection. *Electronics letters*, 39:30, 2003.

[5] J. Astola, P. Haavisto, and Y. Neuvo. Vector median filters. *Proceedings of the IEEE*, 78(4):678–689, 1990.

[6] A.P. Bradley. Shift-invariance in the discrete wavelet transform. In *Digital Image Computing: Techniques and Applications*, volume 4. Citeseer, 2003.

[7] J.N. Bradley and C.M. Brislawn. The wavelet/scalar quantization compression standard for digital fingerprint images. *Circuits and Systems, 1994. ISCAS '94., 1994 IEEE International Symposium on*, 3:205 – 208, 1994.

[8] C S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms, A primer.* Prentice Hall, Upper Saddle River, New Jersey 07458, 1998.

[9] J. Canny. A computational approach to edge detection. *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*, pages 184–203.

[10] C. Christopoulos, A. Skodras, and T. Ebrahimi. The jpeg2000 still image coding: An overview. *IEEE Transactions on Consumer Electronics*, 46:1103–1127, 2000.

[11] C.K. Chui. *An Introduction to Wavelets.* Academic press, San Diego, CA, 1992.

[12] C.H.A. Criens. LPV control of an active vibration isolation system. Master's thesis, Eindhoven University of Technology, 2008. DCT 2008.082.

[13] I. Daubechies. *Ten Lectures on Wavelets*, volume 61 CMBMS-NSF Series in Applied Mathematics. Siam publications, Philadelphia, 1992.

[14] D.L. Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995.

[15] D.L. Donoho and I.M. Johnstone. Adapting to Unknown Smoothness Via Wavelet Shrinkage. *Journal of the american statistical association*, 90(432), 1995.

[16] D.L. Donoho and T.P.Y. Yu. Nonlinear pyramid transforms based on median-interpolation. *SIAM Journal on Mathematical Analysis*, 31(5):1030–1061, 2000.

[17] S. Ericsson and N. Grip. Efficient wavelet prefilters with optimal time-shifts. *IEEE Transactions on Signal Processing*, 53:2451– 2461, 2005.

[18] U. Grasemann and R. Miikkulainen. Effective image compression using evolved wavelets. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, page 1968. ACM, 2005.

[19] R.M. Haralick. Digital step edges from zero crossing of second directional derivatives. *Readings in computer vision: issues, problems, principles, and paradigms*, page 216, 1987.

[20] K . Kashima, Y. Yamamoto, and M. Nagahara. Optimal wavelet expansion via sampled-data control theory. *Signal Processing Letters, IEEE*, 11:79 – 82, 2004.

[21] J.J. Kok and M.J.G. van de Molengraft. *Signaalanalyse*. Eindhoven : Technische Universiteit Eindhoven, 2006.

[22] Chongchun Liu, Yaohui Liu, Zhengding Qiu, and Xiyu Du. On the initialization of the discrete wavelet transform algorithm. *IEEE International Conference on Intelligent Processing Systems*, Volume 2, Issue , 28-31:1220 – 1222, 1997.

[23] S. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on pattern analysis and machine intelligence*, 14(7):710–732, 1992.

[24] Stephane Mallat. *A wavelet tour of signal processing; The sparse way*. Academic Press, London, 3th edition, 2009.

[25] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, pages 187–217, 1980.

[26] R.J.E. Merry. Wavelet theory and applications, a literature study. Eindhoven University of Technology, June 2005. DCT nr. 2005.53.

[27] M. Misiti, Y. Misiti, G. Oppenheim, and J. Poggi. *Wavelet Toolbox™ User's Guide*, march 2009.

[28] Y. Nievergelt. *Wavelets Made Easy*. Birkhuser, 1999.

[29] M.G.E. Schneiders. Wavelets in control engineering. Master's thesis, Eindhoven University of Technology, August 2001. DCT nr. 2001.38.

[30] SK Setarehdan and JJ Soraghan. Fuzzy multiscale edge detection (FMED) applied to automatic leftventricle boundary extraction. 2, 1999.

[31] L.R. Soares, H.M. de Oliveira, R.J.S. Cintra, and R.M. Campello de Souza. Fourier eigenfunctions, uncertainty gabor principle and isoresolution wavelets. *SIMPSIO BRASILEIRO DE TELECOMUNICAES-SBT*, xx:3–4, 2003.

[32] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Welleseley MA USA, 1996.

[33] H.G. ter Morsche. Lecture slides; wavelets and applications 2wn06, 2008.

[34] Jun Tian. *The Mathematical Theory and Applications of Biorthogonal Coifman Wavelet Systems*. PhD thesis, Rice University, 1996.

[35] C.L. Tu, W.L. Hwang, and J. Ho. Analysis of singularities from modulus maxima of complex wavelets. *IEEE Transactions on Information Theory*, 51(3):1049–1062, 2005.

[36] M. Unser. Approximation power of biorthogonal wavelet expansions. *IEEE Transactions on Signal Processing,*, 44(3):519–527, 1996.

[37] C. Valens. A really friendly guide to wavelets. "http://pagesperso-orange.fr/polyvalens/clemens/wavelets/wavelets.html", 1999-2004.

[38] C. Valens. Mathematica wavelet explorer. "http://documents.wolfram.com/applications/wavelet/index.html", 2009.

[39] Imke D. Maassen van den Brink. Real-time detection algorithm for sawtooth crashes in nuclear fusion plasmas. Master's thesis, Eindhoven University of Technology, August 2009. DCT nr. 2009.76.

[40] X. Xia, J.S. Geronimo, D.P. Hardin, and B.W. Suter. Design of prefilters for discrete multiwavelet transforms. *IEEE transactions on signal processing*, 44:25–35, 1996.

[41] X. Xia, C.J. Kuo, and Z. Zhang. Design of optimal fir prefilters for wavelet coefficient computation. *IEEE International Symposium on Circuits and Systems*, Volume , Issue , 3-6:523 –526, 1993.

Appendix A

Orthogonal Discrete Wavelets

In this appendix a summary is given of the most important design properties of orthogonal wavelets. All have been designed by Daubechies but have different properties. In this appendix is explained what the properties are and how to design them.

A.1 Daubechies

The Daubechies wavelets/filters are named after the designer of the filters. The main properties are: the minimum-phase design; maximum number of vanishing moments compared to its support width. All the properties of Daubechies filters are summarized below,

1. Daubechies wavelets are orthogonal
2. The frequency responses have maximum flatness at $\omega = 0$ and $\omega = \pi$ (Maximum number of vanishing moments and zero moments for the wavelet filter)
3. Compactly supported
4. Minimum-phase

The following properties follow automatically from the derivation by Daubechies.

$$L(\omega) = \sum_n l(n) e^{-in\omega} \qquad \text{Compact support} \tag{A.1}$$

$$|L(\omega)| + |L(\omega + \pi)| = 2 \qquad \text{Orthogonality} \tag{A.2}$$

$$L(\omega) = \left(\frac{1+e^{-i\omega}}{2}\right)^P \mathcal{L}(\omega) \qquad \text{Regularity} \tag{A.3}$$

Now the derivation of the scaling and wavelet filters is explained. First the square of the magnitude is calculated with $\mathrm{L}(\omega) = |\mathcal{L}(\omega)|^2$.

$$|L(\omega)|^2 = \left(\cos^2 \frac{\omega}{2}\right)^P \mathrm{L}(\omega) \tag{A.4}$$

$\mathrm{L}(\omega)$ is also a polynomial in $\cos(\omega)$. However it can also be rewritten to $\mathrm{L}(\omega) = P\left(sin^2 \frac{\omega}{2}\right)$. This is a mathematical trick for later derivation. Our goal is to find out how this $P\left(sin^2 \frac{\omega}{2}\right)$ looks. For simplicity $y = \sin^2 \frac{\omega}{2}$. Therefore (A.4) can be rewritten as:

$$|H(\omega)|^2 = (1-y)^K P(y) \tag{A.5}$$

This equation is filled into the orthogonality condition (A.3).

$$(1-y)^K P(y) + y^K P(1-y) = 2 \tag{A.6}$$

Bezout's theorem states: if we have polynomial $p_1(x) q_1(x) + p_2(x) q_2(x) =$ Constant, then p_1 and p_2 are polynomials of degree n_1, n_2. There exist unique polynomials q_1, q_2 of degree

$n_1 - 1, n_2 - 1$. Uniqueness in our case implies that $q_2(y) = q_1(1-y)$ and if we set $q_1(y) = P(y)$. Then we get following equality.

$$P(y) = (1-y)^N \left(2 - y^N P(1-y)\right) \tag{A.7}$$

$P(y)$ can be calculated with the help of a Taylor expansion. The solution has following form:

$$P(y) = \sum_{k=0}^{K} \left(\begin{array}{c} K+k-1 \\ k \end{array} \right) y^k \tag{A.8}$$

Now we can also work out a simple example. Let us consider a filter with length $N = 4$. The maximum number of vanishing moments is $\frac{N}{2} = 2$. With this we use (A.8) to calculate also $P(y)$.

$$P(y) = 1 + 2y = 1 + 2\sin^2\frac{\omega}{2} \tag{A.9}$$

Now all relationships can be filled back in and it is than possible to calculate the magnitude. From the above relationships it becomes clear that a maximum number of vanishing moments design only defines the amplitude of the FIR-filter and leaves the choice of phase open. Therefore Daubechies chose to make the filters minimum-phase filters i.e. the remaining freedom is chosen such that all remaining zeros are inside the unit circle. The corresponding zeros are constructed with the use of spectral factorization. More vanishing moments will lead to more flatness at 0 and π and also for a better separation between the frequency bands (brick-wall). For three different Daubechies wavelets this is shown in Fig. A.1.

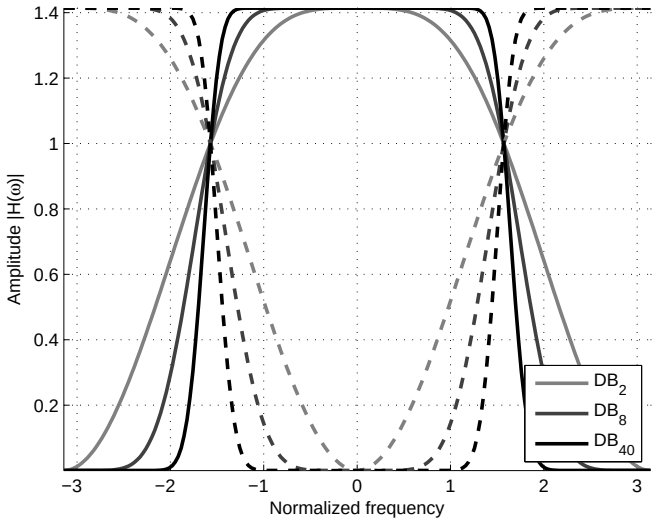


Figure A.1: Daubechies scaling (striped) and wavelet filter (full)

The filter coefficients for the DB_2 that is used in most examples are given in Table A.1:

Table A.1: Daubechies 2 wavelet with support width $N = 4$

n	0	1	2	3
$l(n)$	$\frac{1+\sqrt{3}}{4\sqrt{2}}$	$\frac{3+\sqrt{3}}{4\sqrt{2}}$	$\frac{1-\sqrt{3}}{4\sqrt{2}}$	$-\frac{1-\sqrt{3}}{4\sqrt{2}}$
$h(n)$	$\frac{1-\sqrt{3}}{4\sqrt{2}}$	$\frac{3-\sqrt{3}}{4\sqrt{2}}$	$-\frac{3+\sqrt{3}}{4\sqrt{2}}$	$\frac{1+\sqrt{3}}{4\sqrt{2}}$

A.2 Symlets

Symlets are also designed to have a maximum number of vanishing moments for their support length. However the remaining freedom of the filter phase is used to make it as linear phase as possible. Symlets are closer to linear phase filters and therefore are more symmetric. In Fig. A.2 the Daubechies filter/wavelet is compared to the Symlet filter in the frequency domain.

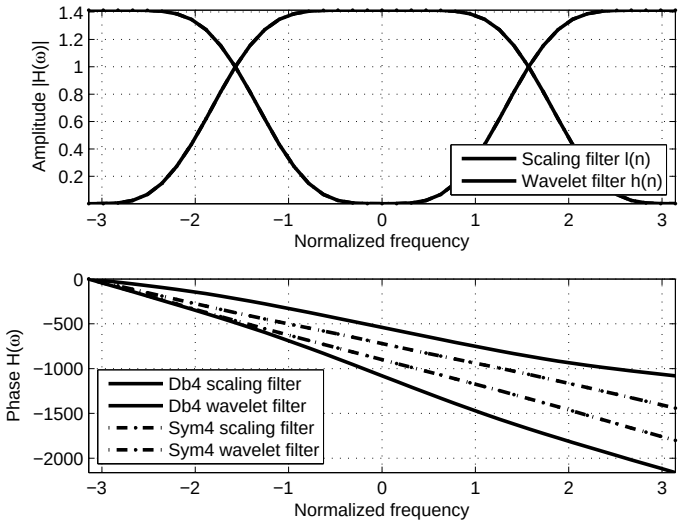


Figure A.2: Comparison between Symlet filter and Daubechies filter

A.3 Coiflets

If it is not desired to have filters with maximum vanishing moments but also zero scaling moments (Coiflets) or better brick-wall behavior or other favorable behavior; this can be also included in the design. This means that the length of the filter will be longer than twice the number of vanishing moments $N > 2K$. $P(y)$ is than calculated with (A.10),

$$P(y) = \sum_{k=0}^{K} \binom{K+k-1}{k} y^k + y^K R\left(\frac{1}{2} - y\right) \tag{A.10}$$

Appendix B

Multiwavelets and multifilters

Multiwavelets are a natural extension to scalar wavelets. Instead of using one scaling function ϕ and one wavelet function ψ, multiple scaling and wavelet functions are used, hence the name multiwavelets. Multiwavelets are expressed in matrix form. The vector scaling function is given by:

$$\Phi(t) = [\phi_1(t), \phi_2(t), .., \phi_N(t)]^T \tag{B.1}$$

Then the dilation and wavelet equation are given by:

$$\Phi(t) = \sqrt{2}\sum_n L(n)\,\Phi(2t-n), \quad \mathcal{V}_0 \in \mathcal{V}_1 \tag{B.2}$$

$$\Psi(t) = \sqrt{2}\sum_n H(n)\,\Phi(2t-n), \quad \mathcal{W}_0 \in \mathcal{V}_1 \tag{B.3}$$

The scalar case has been expanded to the matrix case. This has consequences for a number of relationships. However the Perfect Reconstruction conditions stated in (5.36) and (5.38) remain the same but in a matrix form.

$$\mathbf{H}'(z)\,\mathbf{H}(-z) + \mathbf{L}'(z)\,\mathbf{L}(-z) = 0 \tag{B.4}$$

$$\mathbf{H}'(z)\,\mathbf{H}(z) + \mathbf{L}'(z)\,\mathbf{L}(z) = 2z^{-N}\mathbf{I} \tag{B.5}$$

Also as in the scalar case the multiwavelet transform is orthogonal if (delay is discarded):

$$\mathbf{H_m}(z)\,\mathbf{H_m}(z)^\dagger = 2\mathbf{I} \tag{B.6}$$

These matrix relations greatly enhances freedom of design compared to the scalar case because there are more combinations that will fulfill both perfect reconstruction, orthogonality, linear phase.
The wavelet and scaling coefficients are now also more dimensional and are defined as:

$$C_j(k) = [c_{1,j}(k), c_{2,j}(k), .., c_{N,j}(k)]^T \quad \text{with} \quad c_{i,j} = \langle f(t), \phi_{i,j,k}(t)\rangle \tag{B.7}$$

$$D_j(k) = [d_{1,j}(k), d_{2,j}(k), .., d_{N,j}(k)]^T \quad \text{with} \quad d_{i,j} = \langle f(t), \psi_{i,j,k}(t)\rangle \tag{B.8}$$

The most important advantages and disadvantages have been summarized below:

- Can be symmetric and orthogonal at once impossible with scalar wavelets
- Much more freedom in the design
- More difficult to construct, there are no straightforward designing methods available

- It is no longer straightforward how $\mathbf{H}(\mathbf{z}), \mathbf{L}(\mathbf{z})$ have to be chosen because in the matrix case they often do not commute as is the case with scalar filters (more difficult, but more freedom)
- Initializing the multifilters (multiwavelets) is important (Pre-filtering)
- Higher order of approximation (polynomials) is possible with lower length of filter coefficients

Although there are many different multiwavelets possible, but the design is difficult. A good example of a multiwavelet is the Geronimo-Hardin-Massopust (ϕ) and Strang-Strela (ψ) multiwavelet it is presented in Fig. B.1. It is orthogonal, symmetric (linear phase) and it has short support width.

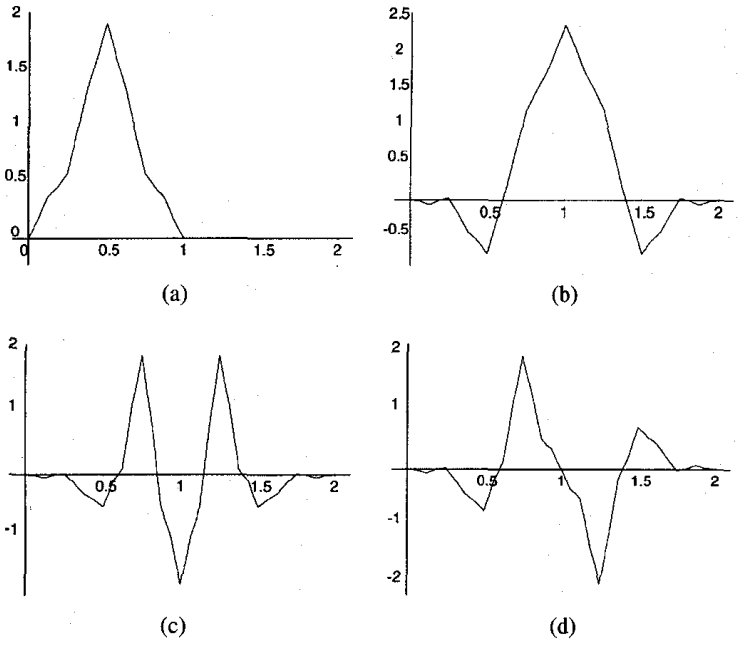


Figure B.1: Multiwavelets generated by Geronimo, Hardin and Massopust (a) and (b) scaling functions respectively; (c) and (d) are wavelet functions generated by Strang and Strela [41]

Appendix C

Wavelets in Frequency Domain

In this appendix is explained how one calculates the scaling function and wavelet in the frequency domain from only the filter coefficients.

C.1 Fourier Transform of the recursion equation

The continuous Fourier Transform is defined as:

$$\hat{\phi}(\omega) = \int_{-\infty}^{\infty} \phi(t) e^{-i \cdot t \cdot \omega} dt \tag{C.1}$$

and in discrete time as:

$$L(\omega) = \sum_{n=0}^{N} l(n) e^{-i \cdot n \cdot \omega} \tag{C.2}$$

The recursion equation (dilation equation) has following form:

$$\phi(t) = \sum_{n=0}^{N} l(n) \phi(2t - n) \tag{C.3}$$

Here we show how in the dilation equation is defined in the Fourier domain the entire scaling function can be calculated and most importantly its convergence. Therefore we first apply the Fourier Transform on the entire dilation equation (C.3),

$$\hat{\phi}(\omega) = \int_{-\infty}^{\infty} \sum_{n=0}^{N} l(n) \phi(2t - n) e^{-i \cdot t \cdot \omega} dt$$

and by substitution we replace $\tau = 2t - n$ and $d\tau = 2dt$;

$$\hat{\phi}(\omega) = \int_{-\infty}^{\infty} \sum_{n=0}^{N} \frac{1}{2} l(n) \phi(\tau) e^{-i \cdot \omega \cdot \frac{\tau + n}{2}} d\tau$$

$$\hat{\phi}(\omega) = \sum_{n=0}^{N} \frac{1}{2} l(n) e^{-i \cdot \frac{\omega}{2} \cdot n} \int_{-\infty}^{\infty} \phi(\tau) e^{-i \cdot \frac{\omega}{2} \cdot \tau} d\tau$$

The integral is exactly the continuous Fourier Transform (C.1) but for the variable $\frac{\omega}{2}$ replacing this equation leads to:

$$\hat{\phi}(\omega) = \left[\sum_{n=0}^{N} \frac{1}{2} l(n) e^{-i \cdot \frac{\omega}{2} \cdot n} \right] \hat{\phi}(\frac{\omega}{2}) \tag{C.4}$$

Between brackets appears the Fourier Transform again but in this case the discrete Fourier Transform. Rewriting and taking the factor 0.5 as part of the filter coefficients $l(n)$ the dilation equation in the Fourier domain is defined as:

$$\hat{\phi}(\omega) = L(\frac{\omega}{2})\hat{\phi}(\frac{\omega}{2}) \tag{C.5}$$

This equation implies; if we take $\frac{\omega}{4}$ instead of $\frac{\omega}{2}$ that than,

$$\hat{\phi}(\frac{\omega}{2}) = L(\frac{\omega}{4})\hat{\phi}(\frac{\omega}{4})$$

Therefore we can rewrite this in a more general form,

$$\hat{\phi}(\frac{\omega}{2^{p-1}}) = L(\frac{\omega}{2^p})\hat{\phi}(\frac{\omega}{2^p})$$

Substituting the recursion equation into itself gives a product, our goal is to find the continuous ω. All values of ω have been calculated when $\omega = 0$. ϕ has converged when $\lim_{p\to+\infty} \hat{\phi}(\frac{\omega}{2^p}) = \hat{\phi}(0)$.

$$\hat{\phi}(\omega) = \prod_{p=1}^{\infty} L(\frac{\omega}{2^p})\hat{\phi}(0)$$

The scaling function ϕ has been normalized with $\sum_k \phi(k) = 1$, in other words $\hat{\phi}(0) = 1$.

$$\hat{\phi}(\omega) = \prod_{p=1}^{\infty} L(\frac{\omega}{2^p})$$

In this way it is possible to check if the scaling function converges in other words exists. We can also consider equation (C.4) in the z-domain. With the relation $z = e^{j\omega}$it can be shown that this results in:

$$\hat{\phi}(z) = L(z^{\frac{1}{2}})\hat{\phi}(z^{\frac{1}{2}}) \rightarrow \hat{\phi}(z^2) = L(z)\hat{\phi}(z)$$

Appendix D

Aliasing in filter banks

Section 6.8 gives an idea about one level of the filter bank. However, it is also interesting to know what happens when aliasing is introduced into the consecutive channels of the filter bank. Here an real example is worked out for a two channel filter bank with four decomposition steps. For the first two downsampling steps the frequency spectrum (FFT) is shown. Also the last two reconstructing steps are shown. Combined with wavelet filtered information will result in the original signal.

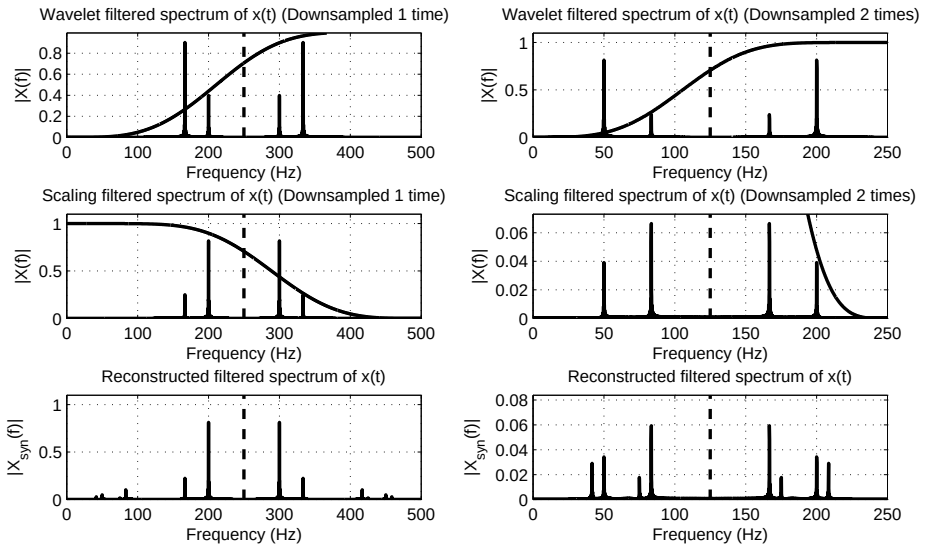


Figure D.1: Filter bank decomposition with filter response (and the striped line is the nyquist frequency)

The original signal consists of two sinusoids one with frequency 200 Hz and another one with frequency 333 Hz. They are both below the nyquist frequency of 500 Hz. The high-pass and low-pass filter decompose the signal into two parts. To show the effect of aliasing we have chosen for DB_4 filters. The signal is split, filtered and downsampled. Downsampling will half the nyquist frequency, in this case to 250 Hz. After the first downsample step in the first Fig. D.1 (wavelet decomposition). The peak at 333 Hz is part of the original signal. The peak at 200 Hz should disappear totally if one would have a brick-wall filter. However because it is not brick-wall part of the peak of 200 Hz has been aliased into the high frequency domain (300 Hz) and has become part of the high frequency spectrum. The same happens for the scaling filter (low-pass filter) but in this case the peak at 333 Hz has been aliased to 167 Hz.
The signal is filtered and downsampled again (downsampled 2 times in total). The wavelet

filtered signal is in principle perfect because there are no original frequencies that can be aliased into the high frequency domain. However in the low frequency domain, that has been filtered by the scaling filter the same effect occurs. Not only the peak at 200 Hz is aliased but also the peak at 167 Hz has now again been aliased. The peak has been aliased two times: the first time from 333 Hz to 167 Hz and now again from 167 Hz to 83 Hz.
The next two decomposition steps are not shown. No original frequencies are present in these domains, but aliased signals are further aliased into the lower frequency ranges. The two figures at the bottom of Fig. D.1 show the reconstructed parts, they should be exactly the same as the above ones of the low-pass filtered signal. They are not because of aliasing from the last two downsampling steps (not represented here). If frequency domains of the original signal and the synthesized signal are compared it immediately becomes clear where aliasing has occurred. Also the difference is plot between the original spectrum and synthesized spectrum.

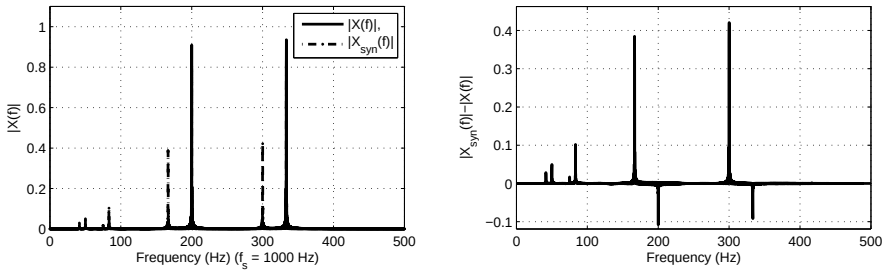


Figure D.2: Original spectrum and synthesized spectrum with aliasing from DB_4-filters

The effect of aliasing is very big and can be very dangerous in the case of frequencies that are relatively near to the center where filters overlap (in the case of halfband filters $(\frac{1}{2})^n f_s$). Also the aliasing is copied again to every next channel causing indirect aliasing on parts where it is not expected. A filter bank that has much better cut-off frequency, for instance the DB_{40} filter bank, reduces the effect of aliasing significantly. However filters will get often much longer which causes delays in real-time applications. The original and synthesized spectra are shown in Fig. D.3. Aliasing is a much smaller problem here.

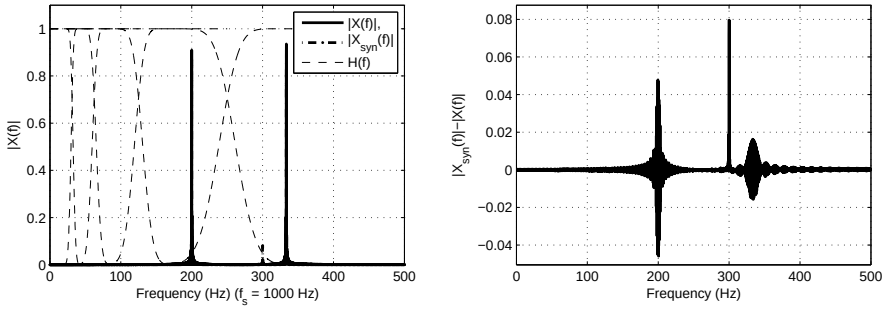


Figure D.3: Original spectrum and synthesized spectrum with aliasing from DB_4-filters (dashed line are the filter magnitudes)