Filtering of Periodic Noise Using the Complex Wavelet Transform

Claus Benjaminsen

Kongens Lyngby 2007

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Summary

Engines, compressors and other machinery performing cyclic processes produce a special kind of noise, which can be called periodic noise. This very common phenomenon - often loud - can create great difficulties, when trying to communicate verbally with another person. With the signal processing possibilities in cell phones and other telecommunication devices, this disturbance can be removed.

In this report a periodic noise filtering scheme is presented based on nearly analytic complex wavelet packets with good shift invariant properties. The shift invariance comes from the Dual-Tree Complex Wavelet Transform, which the nearly analytic complex wavelet packets are built on. But in order to fully maintain the good shift invariant properties of the Dual-Tree Complex Wavelet Transform, the extension to wavelet packets can not be done straight forwardly. It turns out that a special ordering of the wavelet packet filters is needed, and that specific ordering giving nearly analytic complex wavelet packets is developed and presented in this report.

The developed periodic noise filtering scheme gives promising results compared to a spectral subtraction scheme in both a measure of the signal to noise ratio and in a subjective listening test. The scheme calls for some further improvements and tests, but has a potential of making its way into tomorrows telecommunication devices.

Resumé

Motorer, kompressorer og andre maskiner der udfører cykliske processer producere en speciel type støj, som kan kaldes periodisk støj. Denne type støj er et hyppigt fænomen, ofte højt, og kan skabe store problemer, når man prøver at kommunikere verbalt med en anden person. Med de signalbehandlingsmuligheder, som findes i mobiltelefoner og andre telekommunikationsudstyr, kan denne forstyrrende støj blive fjernet.

I denne rapport bliver et periodisk støjfilteringssystem præsenteret baseret på næsten analytiske komplekse wavelet pakker med gode shift invariante egenskaber. Disse komplekse wavelet pakker bygger på en Dual-Tree Complex Wavelet Transformation, men for fuldt ud at beholde de gode shift invariante egenskaber af denne transformation, er udvidelsen til komplekse wavelet pakker ikke lige frem. Det viser sig, at wavelet pakke filtrene skal være i en speciel orden, og denne orden, som giver næsten analytiske komplekse wavelet pakker bliver udviklet og præsenteret i denne rapport.

Det udviklede periodiske støjfiltreringssystem giver lovende resultater sammenlignet med en spectral subtraction metode både hvad angår signal til støj niveau og i en subjektiv lyttetest. Det periodiske støjfiltreringssystem kræver nogle yderligere forbedringer og test, men har et potentiale til at finde vej til morgendagens telekommunikationsudstyr.

Preface

This master's thesis was carried out in collaboration with Informatics and Mathematical Modelling at the Technical University of Denmark, and advised there by associate professor Jan Larsen. The actual project work was done at the Institut für Industrielle Informationstechnik, University of Karlsruhe, Germany, in cooperation with MSc Thomas Weickert. The thesis is the fulfillment of the final step in the electrical engineering master's degree at the Technical University of Denmark. The project was started on January 8th 2007 and was handed in approximately 7 months later on the 15th of August 2007.

The main topic of this thesis is speech signal processing. In this broad area an especially interesting problem has been chosen, namely how to remove periodic noise corrupting a speech signal. Until now not a lot of research has been put into dealing with periodic noise, because the capacity of electronics has not allowed space for algorithms dealing with more specialized problems. With advances in signal processing tools such as complex wavelets, and continued improvements in the processing power of electronics, new possibilities for developing and implementing more powerful algorithms have arisen. The motivation for this project lies in these new opportunities to deal with specialized, but common and hence important problems like periodic noise.

Lyngby, August 2007

Claus Benjaminsen

Acknowledgements

Writing this thesis was a good and interesting process, and I would like to thank my very encouraging and helpful German advisor Thomas Weickert, for being ready to discuss my work and to come up with valuable comments and ideas at any time. I would also like to thank my Danish advisor Jan Larsen for his time, valuable observations and guidelines to help me complete this report. Further I would like to give a special thanks to my sweet girlfriend Melanie, who was always there to back me up, when things were not going as well as I wanted. Also of course a special thanks to my family for always being supportive, and a thanks to all other people who helped and contributed to my work on this project.

Contents

Summary i

Resumé iii

Preface v

Acknowledgements vii

1 Introduction 1

1.1 Overview of A Complete Periodic Noise Filtering System 2

1.2 Chapter Overview 3

2 Basic Theory of Wavelet Filtering 5

2.1 The Wavelet Transform 6

2.2 Wavelet Packets 15

3 Periodic Noise and The Period Wavelet Packet Transform 25

3.1 Periodic Noise 25
3.2 Period Wavelet Packet (PWP) Transform 26

4 Shift Invariance and Complex Wavelet Packets 39

4.1 Shift Invariant Real Wavelet Transforms 39
4.2 The Dual Tree Complex Wavelet Transform 41
4.3 Expanding the DTCWT to Complex Wavelet Packets 48

5 Implementation 57

5.1 Implementation of the Noise Period Analyzer and the Noise Filter 57
5.2 A Spectral Subtraction Scheme 60
5.3 Matlab Implementation 60

6 Evaluation 63

6.1 Evaluating the Periodic Noise Filtering Scheme Using SNR's . . 63
6.2 Evaluation Using Listening Test 73

7 Conclusion 79

7.1 The Achievements 79
7.2 Outlook 80

A Mathematical Derivation of Wavelet Transform Equations 83

A.1 The Forward Calculation 84
A.2 The Inverse Calculation 84

B Complex Wavelet Packet Transform Filter Coefficients 87

Chapter 1

Introduction

Telecommunication is everywhere in modern society, and the ability to talk to another person through an electronic device is a natural thing. Everybody has a cell phone and many people also use hand free headsets, so they can talk to people anytime, anywhere, while doing any kind of activity. Having only the voice transferred through such devices, the users rely heavily on good sound quality with very little noise. This can normally be achieved using todays technology, but that is not always good enough. There are many environments in which background noise is unavoidable, and that can in many situations be very annoying for the users and make their communication slow, difficult, faulty or even impossible. Everybody knows the annoying situation where surrounding noise corrupts the phone conversation, and you either have to yell into the phone or find a quieter place to continue. This is currently an unsolved problem, but with the right advances in electronics and signal processing, the situation could be greatly improved.

This project is a step in the direction of developing tools to deal with such noise problems. The focus has been put on a special, but common kind of background noise called periodic noise. This kind of noise or sound is produced by machinery performing cyclic processes such as engines, conveyor belts and compressors, but is also produced in ordinary households by things such as vacuum cleaners, hand mixers and blenders. This noise is nonstationary, because it changes with time, but it changes in a special way, which can be exploited. The noise at time t can

not be used to say anything about the noise at any time $t + x$ into the future, but for the specific time $t + T$, where T is the period of the noise, it can give useful information.

A tool which can use this information is the wavelet transform. The wavelet transform can trade time information for frequency information in a good controllable way, and hence it is well suited for working with periodic noise, where the time information is important. This project therefore includes a lot of wavelet theory, the extension to wavelet packets and the extension to complex wavelets, plus the powerful development of the combination of the two. Further it involves a period wavelet packet scheme, which basically tries to match the wavelet packets to the given length of the noise periods. All of these things are then put together to form a periodic noise filtering scheme with good noise removal abilities. The overall goal is to preserve the speech signal, while suppressing the noise, so that easier understanding of the spoken words is achieved.

1.1 Overview of A Complete Periodic Noise Filtering System

A filtering system is often more than just a filter, typically other components are also needed in order to effectively process the desired signal(s). A complete system for filtering periodic noise is shown in figure 1.1. It consists of 4 components, which in corporation do the filtering task.

This project will not cover the whole filtering system, but focus on the two blocks shown in gray, the Noise Period Analyzer and the Noise Filter. The Noise Period Analyzer is processing the noise period for period. In order to do that it needs information about when the speech isn't present in the signal, and how long the periods of the noise are. These informations are provided by the Speech Pause Detector and the Period Length Estimator respectively, and the development of these components are projects of themselves. In this project the information from these two components are assumed available for the Noise Period Analyzer.

The Noise Period Analyzer will construct a thresholding function, which is supplied to the Noise Filter. In the Noise Filter the noisy speech signal is filtered using the thresholding function, and the resulting signal is the output of the system. Both the Noise Period Analyzer and the Noise Filter will be implemented with complex wavelet packets, which will be developed in this project.

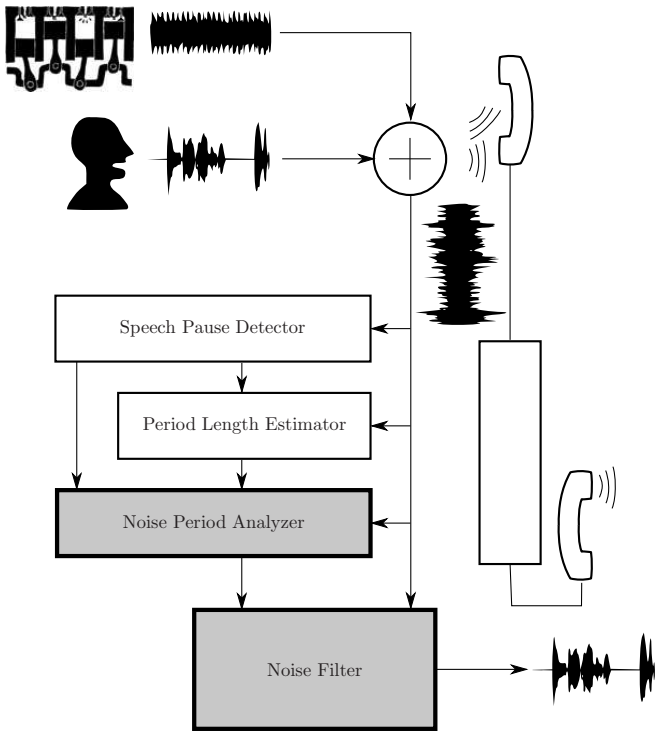


Figure 1.1: A complete periodic noise filtering system.

1.2 Chapter Overview

This report is mainly dealing with wavelets and wavelet theory, but it doesn't require any prior knowledge in this area. Anybody with a basic knowledge of signal processing can read this report, as it includes all the necessary theory to understand the more advanced wavelet developments made in the later chapters. The more advanced reader can therefore skip over most of the general theory presented in chapter 2, which includes wavelet packets and denoising using wavelets, and proceed to chapter 3. When specific theory from chapter 2 is used, it is normally referenced, which makes it easy to jump back and read through that specific section of chapter 2, when needed. In chapter 3 some insights into periodic noise are given, and thereafter the period wavelet packet transform is presented, and modifications to the transform are discussed. Chapter 4 starts with a discussion of shift invariance and shift invariant wavelet transforms, and proceeds with an introduction of the Dual-Tree Complex Wavelet Transform. From this transform the extension to complex wavelet packets is made, and a

special ordering of the wavelet packet filters to achieve maximal shift invariance is developed. The theory from all of these chapters is put together in chapter 5, where the Noise Period Analyzer and the Noise Filter are more thoroughly described. Finally the periodic noise filtering scheme is tested in chapter 6, and the report is ended with a conclusion and an outlook in chapter 7.

Chapter 2

Basic Theory of Wavelet Filtering

Filtering is normally associated with the Fourier transform. Maybe the filtering is not done in the frequency (Fourier) domain by transforming the signal, but the filter used is normally designed to have specific frequency characteristics. This standard filtering approach is effective in many situations, because time-overlapping signals with different frequency contents can be separated in the frequency domain. The biggest drawback of the Fourier Transform is that it doesn't give any time-information. It will show that certain frequencies are contained in a signal, but not when they were present.

Time-information can be very important especially for time varying signals like speech, and therefore other transforms have been developed, which try to give both time- and frequency-information at the same time. Such transforms are for instance the Short Time Fourier Transform (STFT) and the wavelet transform. The STFT is calculated over a certain time-frame, the longer the frame the higher the frequency resolution over the entire frequency range, this is therefore a time-frequency resolution trade-off.

The Wavelet Transform is different in the aspect that the frequency resolution is not uniform over the entire frequency range, but different for different frequency bands. For the high frequencies the resolution is low, but the time resolution

is high, and for the lower frequencies that gradually changes toward higher frequency resolution and lower time resolution. This predefined time-frequency resolution structure is even relaxed with the extension to wavelet packets, which makes it possible to choose the time-frequency resolution trade-off over the entire frequency range. Such non-uniform time-frequency resolution can very effectively be adapted to the processed signal, and this is in many cases an advantage compared to the STFT.

In the following sections the wavelet transform will be introduced and the extension to wavelet packets will be presented in section 2.2.

2.1 The Wavelet Transform

2.1.1 Projection on Basis Functions

The wavelet transform is in principle the projection of a signal onto wavelet basis functions. These are called scaling and wavelet functions and are normally denoted by $\varphi_{j,k}(t)$ and $\psi_{j,k}(t)$ respectively.

2.1.1.1 The Scaling Function

The scaling functions are functions of two parameters j and k, which are called the scaling coefficient and the shifting coefficient respectively [1]. This is a result of how the scaling functions are defined, as scaled and shifted versions of a "mother" scaling function:

$$\varphi_{j,k}(t) = 2^{j/2}\varphi(2^j t - k) \tag{2.1}$$

Scaling functions with the same scale parameter j will all be shifted versions of the same function, where the shift is controlled by the parameter k. The $j+1$ scaling functions will be compressed versions of the scaling functions at level j by a factor of 2, and the level $j-1$ scaling functions will be expanded versions also by a factor of 2.

An example of scaling functions at different levels is shown in figure 2.1. It is clear how increasing j compress the scaling function, and hence increase the time resolution. This comes as an expense in frequency resolution though, and in that way j controls the time-frequency resolution trade-off.

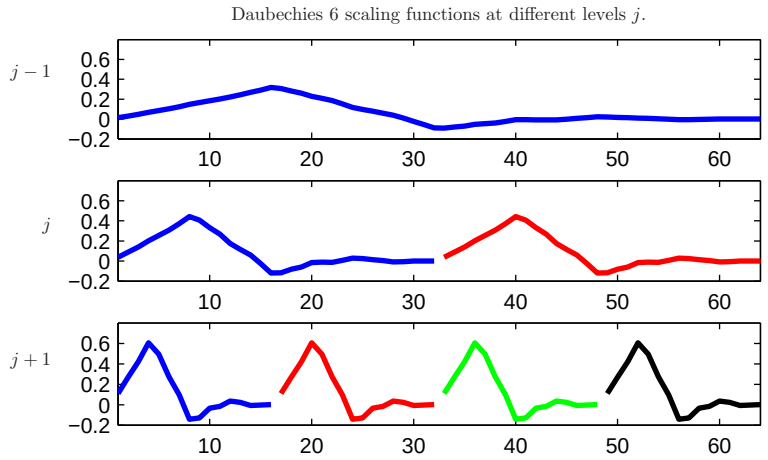


Figure 2.1: Daubechies 6 scaling functions at three different levels j.

At all levels the scaling functions with the same parameter j are orthogonal and span a space $\mathcal{V}_j$

$$\operatorname*{Span}_k\{\varphi_{j,k}(t)\} = \mathcal{V}_j \tag{2.2}$$

which includes the spaces spanned by scaling functions at all lower levels (lower values of j) [2]. This is illustrated in figure 2.2.

2.1.1.2 The Wavelet Function

The wavelet functions are in the same way as the scaling functions characterized by the two parameters j and k:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k), \qquad \operatorname*{Span}_k\{\psi_{j,k}(t)\} = \mathcal{W}_j \tag{2.3}$$

Also all the wavelet functions at a certain level are orthogonal and span a space $\mathcal{W}_j$, and these wavelet function spaces are orthogonal to each other. The space $\mathcal{W}_j$ is also orthogonal to the space $\mathcal{V}_j$ and together they span the space $\mathcal{V}_{j+1}$. Mathematically this can be written as

$$\mathcal{W}_j \perp \mathcal{V}_j, \qquad \mathcal{W}_j \oplus \mathcal{V}_j = \mathcal{V}_{j+1} \tag{2.4}$$

and is illustrated in figure 2.2.

Since a scaling function at level j is included in the space spanned by the scaling functions at level $j+1$, it can be written as a linear combination of the level

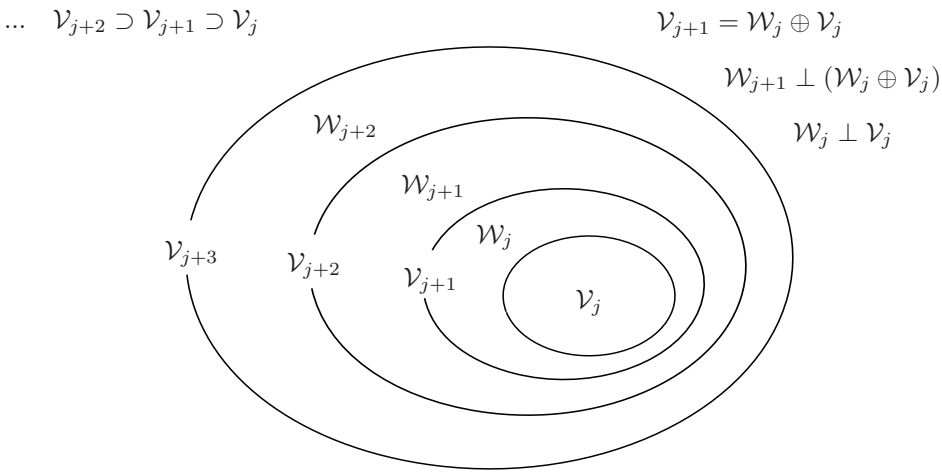


Figure 2.2: Relation between the spaces spanned by scaling and wavelet functions at different levels j.

$j+1$ scaling functions

$$\varphi_{j,0}(t) = \sum_n g_0(n)\varphi_{j+1,n}(t) = \sum_n g_0(n)\sqrt{2}\varphi_{j,n}(2t) \tag{2.5}$$

or

$$\varphi(t) = \sum_n g_0(n)\sqrt{2}\varphi(2t-n). \tag{2.6}$$

For the wavelet functions we have $\mathcal{W}_{j-1} \subset \mathcal{V}_j$ and therefore, in the same way as for the scaling functions, it is possible to write

$$\psi_{j,0}(t) = \sum_n g_1(n)\sqrt{2}\varphi_{j,n}(2t) \tag{2.7}$$

and for $\mathcal{W}_j \perp \mathcal{V}_j$ to be true one can show [2] that

$$g_1(n) = (-1)^k g_0(1-n) \tag{2.8}$$

The g_0 coefficients completely define the scaling function, and since they also give the g_1 coefficients, they are sufficient to describe a complete wavelet system of scaling and wavelet functions. As will be apparent in section 2.1.2, the g_0 and g_1 coefficients are also what is used in practical calculations of the wavelet transform.

2.1.2 Practical Calculation Using Filter Banks

2.1.2.1 Forward Wavelet Transform

Let us assume that the signal $f(t) \in \mathcal{V}_{j_1+1}$, then one possible basis in which the signal can be fully represented is the collection of scaling functions at level j_1+1. Another possible basis could be $\{\mathcal{W}_{j_1}, \mathcal{V}_{j_1}\}$ and yet another one could be $\{\mathcal{W}_{j_1}, \mathcal{W}_{j_1-1}, \mathcal{V}_{j_1-1}\}$. In that way it is possible to choose many different bases in which the signal can be expanded, because the space spanned by the scaling functions at level j, can always be spanned by wavelet functions and scaling functions at a level below $(j-1)$. The signal $f(t)$ can then be written as

$$f(t) = \sum_k c_{j_0}(k)\varphi_{j_0,k}(t) + \sum_{j=j_0}^{j_1} \sum_k d_j(k)\psi_{j,k}(t) \tag{2.9}$$

where $c_{j_0}(k)$ are the scaling function coefficients at level j_0, and $d_j(k)$ are the wavelet function coefficients at the levels from j_0 to j_1.

Instead of first choosing a basis for the wavelet transform, and then projecting the input signal onto these basis functions by calculating the inner products, it turns out that there is a more convenient way of calculating the wavelet transform coefficients (c and d) namely by conjugate mirror filter banks [2]. As shown in appendix A, there exists a simple relation between the scaling and wavelet function coefficients at level j and the scaling function coefficients at level $j+1$

$$c_j(k) = \sum_m g_0(m-2k)c_{j+1}(m) \tag{2.10}$$

$$d_j(k) = \sum_m g_1(m-2k)c_{j+1}(m) \tag{2.11}$$

where g_0 and g_1 are the same as in equations (2.6) and (2.7).

These equations actually corresponds to a filtering operation of c_{j+1} by $g(-n) = h(n)$ followed by down-sampling by a factor 2 as shown in figure 2.3.

The coefficients from the highpass filter are the wavelet coefficients corresponding to a projection onto the wavelet functions at level j, and the coefficients from the lowpass filter are the projections onto scaling functions at level j. As a good approximation, samples of an input signal can be used as the highest level scaling function coefficients [3]. If more filter bank stages are applied to the scaling function coefficients, the result is a filter bank, which give an easy way of calculating the wavelet transform of an input signal as shown in figure 2.4.

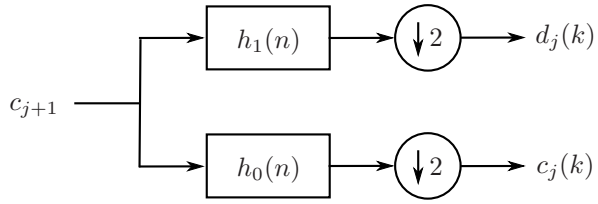


Figure 2.3: A single wavelet decomposition stage.

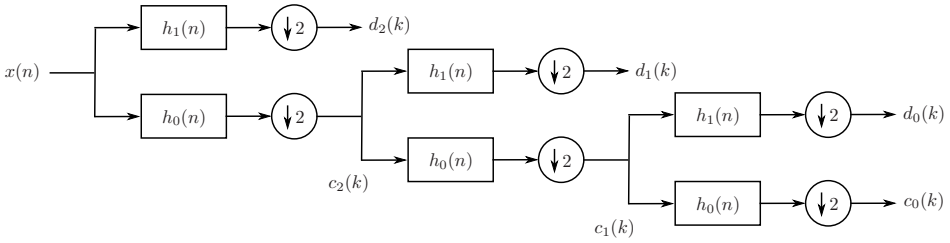


Figure 2.4: Filter bank used to calculate the wavelet transform of an input signal x.

By convention, the coefficients at the lowest level is denoted by 0, and the coefficients at higher levels are then numbered accordingly. It should be noted, that when the transform is used the first coefficients one obtains (after the first filtering stage) have the highest number, which depends on the depth of the transform. It can therefore be rather confusing at times, how the coefficients are numbered and ordered, so care must be taken in order to avoid mistakes.

Since each stage in the filter bank reduces the number of scaling function coefficients by a factor 2, it is only possible to continue to extend the filter bank as long as the number of scaling function coefficients are dividable by two. Therefore the length of the input signal actually determines the highest possible number of sections in the filter bank and can be found by evaluating the following expression:

$$\operatorname{rem}\left\{N, 2^D\right\} = 0. \tag{2.12}$$

Here N is the length of the input signal, D is the number of filter stages and $\operatorname{rem}\{\}$ is the remainder of the division of N by 2^D. Often the length of the input signal is required to be dyadic, that means it can be written in the form $N = 2^L$, where L is an integer, even though that is not necessary as long as the above equation (2.12) is satisfied.

2.1.2.2 Inverse Wavelet Transform

The inverse transform is described by the equation

$$c_{j_0+1}(m) = \sum_k c_{j_0}(k) g_0(m-2k) + \sum_k d_{j_0}(k) g_1(m-2k) \tag{2.13}$$

which is derived in appendix A.

This is equivalent to first up-sampling and then filtering of the scaling function and wavelet function coefficients. The corresponding inverse filter bank is shown in figure 2.5. In the figure the filters are denoted by g_0 and g_1, and they are the reverse of h_0 and h_1 respectively, which were used in the forward transform.

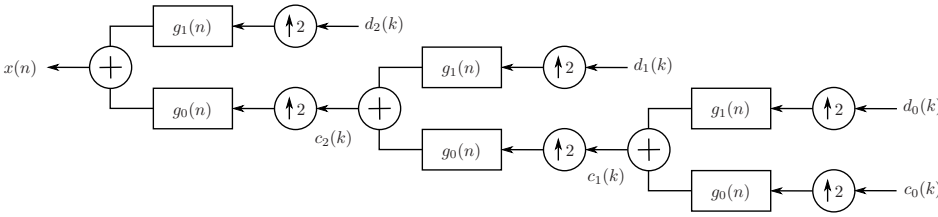


Figure 2.5: The inverse filter bank structure.

At each stage the scaling function coefficients are recombined with the wavelet coefficients at the same level to reconstruct the scaling function coefficients at the level above.

This structure can also be used to find the basis functions of the wavelet transform. As can be seen from equation (2.9), each of the c and d coefficients are a weight of a scaling or a wavelet function. Therefore if all coefficients are set to 0, and only the $d_{j_0}(k_0)$ coefficient is set to 1, then $f(t) = \psi_{j_0,k_0}(t)$ and the inverse transform will reconstruct that particular wavelet function.

As seen above the wavelet filters are all that is needed to calculated the wavelet transform. This also means that the design of wavelet systems is normally done by designing the wavelet filters. These filters have to fulfill certain requirements, which can be found in both [1] and [2] and most other wavelet literature. Since wavelet filter design is beyond the scope of this project, it will not be discussed here. Instead it is useful to note that the forward and inverse transforms form a perfect reconstruction (PR) filter bank, which means that whatever is feed to the forward transform can be exactly recovered by feeding the wavelet coefficients to the inverse transform. Also the wavelet filters can be finite length FIR filters, and that very short filters have been designed with good properties. This makes

it possible to implement the wavelet transform with low computation costs, and since it can run on a sample by sample basis, it is well suited for real-time applications.

2.1.2.3 The Filtering Operation

As shown above the wavelet transform is conveniently calculated using filtering operations, which are based on convolutions. This is straight forward when the sequences are infinitely long, but with finite length sequences, the edges of the input signal need to be considered and circular convolution is then used. The circular convolution is normally calculated as a normal convolution with the input signal circularly extended as shown in figure 2.6. The extension is done with $N_f - 1$ samples, where N_f is the number of coefficients in the filter. After the convolution, only the convolution coefficients obtained, when the filter and signal fully overlap, are kept.

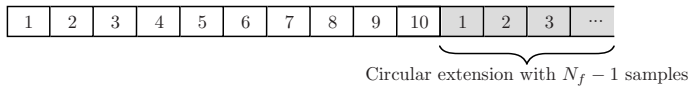


Figure 2.6: Circular convolution is calculated as a normal convolution by extending the input signal with $N_f - 1$ samples. Then only the convolution coefficients achieved, when filter and signal fully overlap, are kept.

The convolution operation (also the circular) is distributive meaning that

$$f * (s + n) = f * s + f * n. \tag{2.14}$$

Therefore the wavelet transform is also distributive. An interesting result of this is that the wavelet coefficients of a noisy signal are equal to the sum of the wavelet coefficients of the signal and the wavelet coefficients of the noise.

As will be described in the following section, each wavelet coefficient represents the transformed signal in a certain time period. When looking at the wavelet coefficients it is therefore important that they are aligned well with the input signal, so that they can be interpreted correctly. When doing the convolution, N_f signal samples are combined in every convolution coefficient (N_f is the number of filter coefficients), so which signal sample should the convolution coefficient be aligned with? It is not possible to give a simple answer to that question, and there is in principle no correct answer. The convolution is a weighted sum, so depending on the distribution of the weights, some samples will have a bigger effect on the convolution coefficient than others. The alignment should therefore in general depend on the filter coefficients, but a simple and in general

good approach is to align the convolution coefficient with a sample in the middle of the filter impulse response. This alignment can be achieved by shifting the convolution coefficients after the whole convolution is done, or when using circular convolution by extending the input sequence both in front and in the back, before doing the convolution, as shown in figure 2.7.

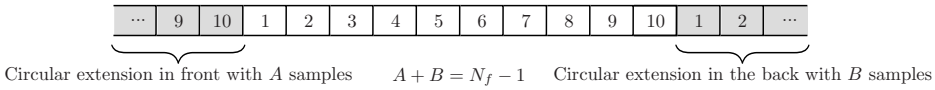


Figure 2.7: The circular extension can also be done in front or both in front and in the back, the results are the same just shifted.

2.1.3 Time-Frequency Interpretation

2.1.3.1 Parseval's Theorem

The scaling and wavelet functions, which from here on will be referred to as wavelet basis functions, all have the same energy independent of the level j. This can be verified by examining equation (2.1) and (2.3), where the factor of $2^{j/2}$ ensures that the energy remains the same at different levels. The wavelet basis functions are normally designed to fulfill

$$\int_{-\infty}^{\infty} \varphi_{j,k}(t)dt = \int_{-\infty}^{\infty} \psi_{j,k}(t)dt = 1 \tag{2.15}$$

which, along with the fact that the wavelet basis functions are orthogonal, means that they form an orthonormal basis, and further that the energy of the wavelet coefficients is equal to the energy of the original signal. This relation is for the Fourier transform known as Parseval's theorem and can be written as [1]

$$\sum_n |f(n)|^2 = \sum_k |c_{j_0}(k)|^2 + \sum_{j=j_0}^{j_1} \sum_k |d_j(k)|^2. \tag{2.16}$$

The energy conservation in the wavelet domain is very useful for signal analysis, as it makes it easier to interpret the wavelet coefficients.

2.1.3.2 Time-Frequency Planes

The filters h_0 and h_1 in figure 2.4 are low- and highpass filters respectively. That means by each stage in the wavelet transform, the $c_j(k)$ coefficients are

split in a highpass part ($d_{j-1}(k)$) and a lowpass part ($c_{j-1}(k)$). In this way the spectrum of the input signal is repeatedly divided [2] as illustrated in figure 2.8.

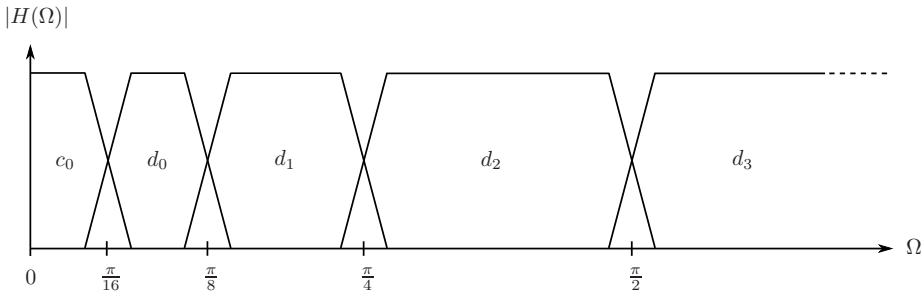


Figure 2.8: The wavelet transform splits a signal into smaller frequency bands. $\Omega = \frac{2\pi f}{f_s}$ is the normalized angular frequency, f is the actual frequency in Hz and f_s is the sampling frequency in Hz.

The energy of the input signal, which falls into a specific frequency band, is represented by the corresponding set of wavelet or scaling function coefficients. These coefficients are time dependent, and therefore carry information about the input signal in both the time and the frequency domain.

If we first look at a discrete time signal, each sample will represent the energy of the signal over all frequencies within the bandwidth of the signal determined by the sampling rate. This bandwidth is given by the Nyquist sampling theorem

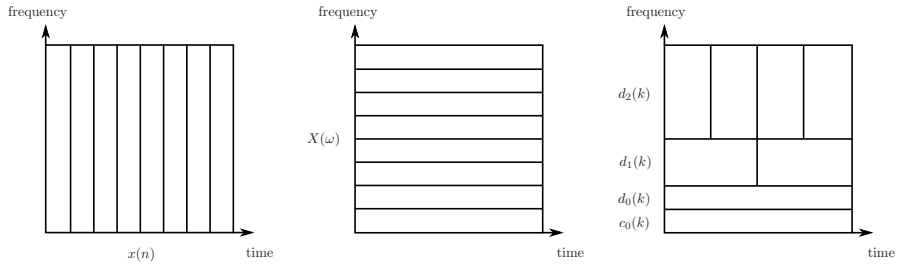
$$B = \frac{f_s}{2} \tag{2.17}$$

where f_s is the sampling frequency. Therefore each sample will represent the signal in a time period of $T = \frac{1}{f_s}$ and a frequency band of $B = \frac{f_s}{2}$. In a time-frequency plane this gives a rectangle with an area of

$$A = TB = \frac{1}{f_s}\frac{f_s}{2} = \frac{1}{2} \tag{2.18}$$

and this is the highest possible resolution according to the Heisenberg Uncertainty Principle [1]. For a discrete time signal each sample will therefore correspond to a square in the time-frequency plane in figure 2.9(a).

The same time-frequency plane can be drawn for a Fourier transformed signal. In that case each Fourier coefficient corresponds to a certain frequency band and represents the energy in that frequency band during the entire time length of the signal. This is shown in figure 2.9(b).



(a) Time samples (b) Fourier coefficients (c) Wavelet coefficients

Figure 2.9: Time-frequency planes for a signal in different domains.

Finally comparing with a wavelet transformed signal, it is found to be in between the discrete time signal and the Fourier transformed signal, because the wavelet coefficients carry both time and frequency information. Each filtering stage in the wavelet transform splits the signal up in two, one sequence carrying the upper half of the frequencies in the signal (the d coefficients) and the other carrying the lower half (the c coefficients). In that way the new coefficients represents half as wide frequency bands, but since the sequences are at the same time down-sampled, the time period is also doubled. The result is a time-frequency plane like the one shown in figure 2.9(c).

It should be noted here that no practical filters have a vertical transition between the passband and the stopband, therefore a small part of the energy from the lower frequencies will always be present in the d coefficients representing the high frequencies and vice versa. The horizontal lines between the squares in figure 2.9(c) are therefore only approximate, and in reality no exact line can be drawn, because energy is leaking between the squares.

2.2 Wavelet Packets

The filters h_0 and h_1 in figure 2.4 together with g_0 and g_1 in figure 2.5 are a perfect reconstruction filter set, which means that when used as in the wavelet transform, it will always be able to reconstruct the original signal. It is therefore straight forward to extend the wavelet transform, so that both the scaling function coefficients and the wavelet function coefficients are repeatedly filtered and down-sampled. This extension is called the wavelet packet transform and is shown in the top of figure 2.12. Note that two filter pairs are shown dotted to illustrate, that it is possible to choose many filter structures for the wavelet packet transform.

The structure is often called a tree structure or a basis tree, and such a basis tree for the above example is given in figure 2.10. Here the high and lowpass filters are labeled with h and ℓ, and the numbers label what is called the nodes. A node is a junction in the graph of the tree structure or can be considered as the collection of the low- and highpass filters and the down-samplers following the junction, see figure 2.12.

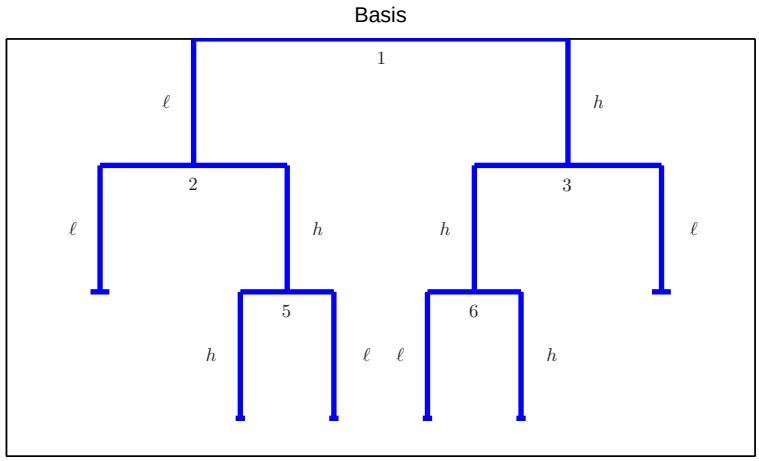


Figure 2.10: The basis tree for the wavelet packet transform shown in figure 2.12.

It might seem strange how the low- and highpass filters are mixed in figure 2.10, instead of all the lowpass filters in the left branches and the highpass filters in the right branches. The special ordering is done to sort the outputs according to frequency content of the input signal, so that the outputs containing coefficients coming from the lowest frequencies in the input signal are on the far left, and going to the right in the tree means increasing frequencies. Why this is not achieved, when all the left branches contain lowpass filters, is a result of down-sampling the outputs of the highpass filters. Note that it is in the nodes after the highpass filters, in figure 2.10 node 3, 5 and 6, where the filters are switched around compared to the previous node.

To illustrate what is going on, the magnitude spectrum of the output of a highpass filter is shown in the top of figure 2.11.

As the output signal is discrete the spectrum is repeated at $\Omega = \pm\pi$. After the highpass filter the signal is down-sampled resulting in a sampling frequency, which is half the previous one. This results in the spectrum in the bottom of figure 2.11. Note how the spectrum in the range from $-\pi$ to π has been turned

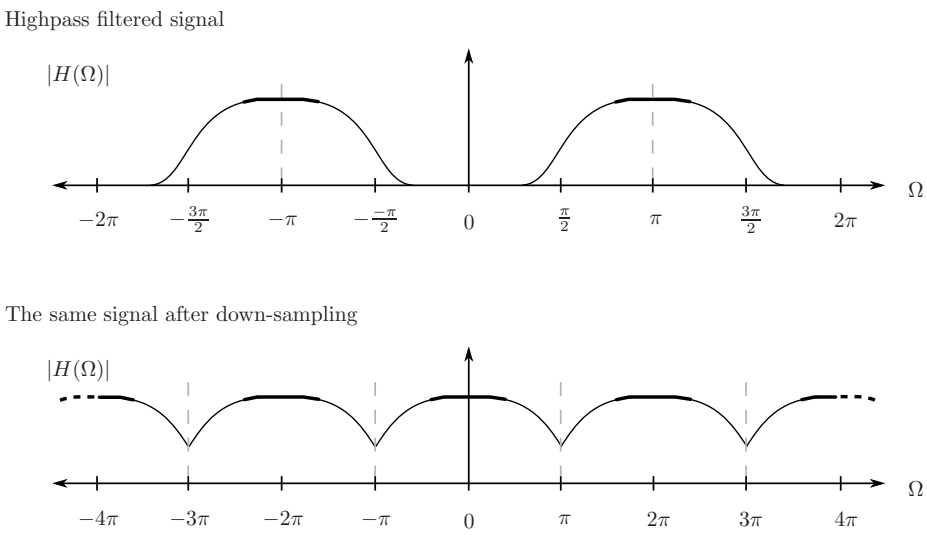


Figure 2.11: The top graph shows the magnitude spectrum of a highpass filtered signal. The bottom graph shows the magnitude spectrum of the same signal after down-sampling.

around, so that what was the high frequencies before the down-sampling (shown with a thicker line) is now the low frequencies. That means that when the next filter is a lowpass filter, it will actually pick out what was originally the high frequencies of the input signal, and hence it will be in the right branch and the highpass filter in the left.

What can also be seen in figure 2.11 is that the down-sampling also causes some aliasing. This is not a problem in the sense, that the original signal can still be perfectly reconstructed, but when the output coefficients are interpreted as coming from different frequency bands, the aliasing has to be kept in mind.

Along with the structure of the filter bank in figure 2.12, an input vector of eight elements is given, and the values of these eight samples are shown going through each stage of the transform. Notice how the samples are labeled as $c_{d,b}$ at the different nodes in the filter bank. The d gives the depth in the filter bank, and the b the specific node at that depth. At depth d there are 2^d nodes labeled from 0 to $b = 2^d - 1$. The number of coefficients n_d from a given node is determined by the depth and the number of input samples N as

$$n_d = \frac{N}{2^d} \tag{2.19}$$

The nodes are also often numbered with just a single number as shown in figure

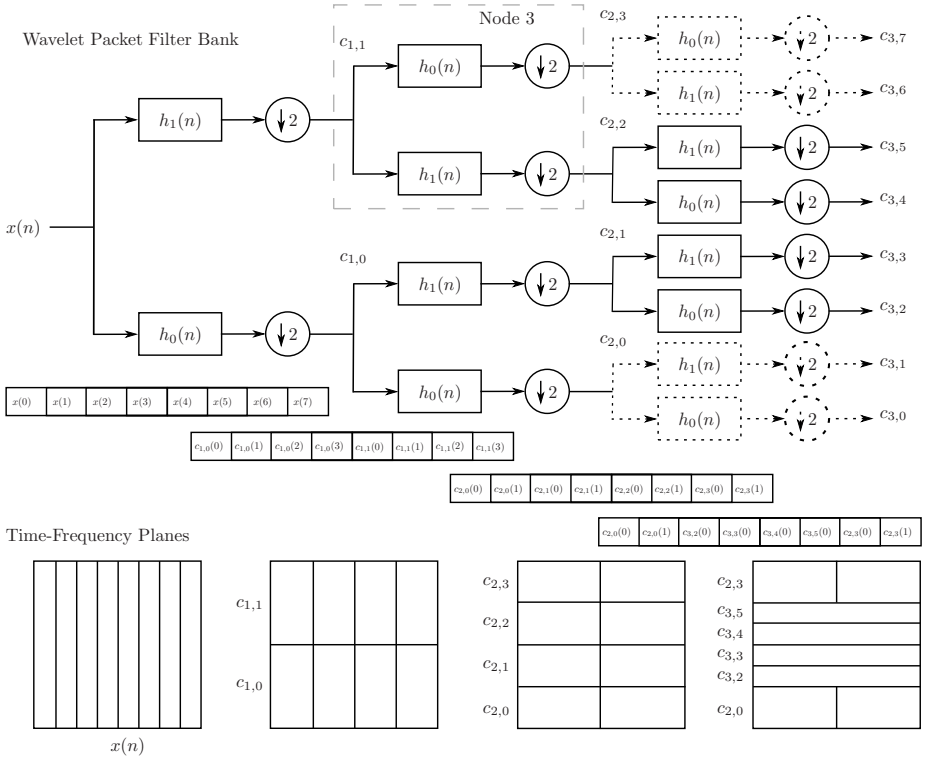


Figure 2.12: The wavelet packet transform.

2.10. The relation between the node number and the d and b parameters can be written as

$$node = 2^d + b \tag{2.20}$$

Different basis tree structures results in different time-frequency tilings as shown in the bottom of figure 2.12. Therefore knowing the input signal, it is possible to find a basis tree, which matches the time-frequency content of the input signal, and hence give a very compact representation of the signal. This is important, because a compact representation, where the signal is represented using only a few coefficients, is desirable for both compression and denoising problems.

2.2.1 Finding the Best Wavelet Packet Basis Tree

The basis tree which matches a given input signal the best, in the sense that most of the signal energy is represented by fewest possible coefficients, can be defined as follows [1]:

If the wavelet packet coefficients are sorted in descending order so that $c(m) > c(m+1)$, then the best basis tree a will be the one for which

$$\sum_{m=0}^{M} |c^a(m)|^2 \geq \sum_{m=0}^{M} |c^b(m)|^2, \qquad 0 \leq M \leq N-1 \tag{2.21}$$

over all other structures b, where N is the total number of wavelet packet coefficients. To find the best basis tree using the above relation requires a lot of calculations, and therefore another equation has been constructed, which can be used instead. It uses what is called a concave function and is written as

$$\sum_{m=1}^{N} \Phi\left(\frac{|c^a(m)|^2}{\|f\|^2}\right) \leq \sum_{m=1}^{N} \Phi\left(\frac{|c^b(m)|^2}{\|f\|^2}\right) \tag{2.22}$$

where Φ is the concave function, and $\|f\|^2$ is the total energy of the input signal. An example of a concave function is the entropy function defined as

$$\Phi(x) = -x\ln(x), \qquad x > 0 \tag{2.23}$$

which in this project is used to find the best basis tree.

Equation (2.22) still requires one summation of all the wavelet coefficients for all possible different basis trees. A fast implementation first calculates all possible wavelet packet coefficients using a full basis tree, where all nodes are included. Then it calculates the summation in equation (2.22) for all nodes, and from the bottom of the basis tree it starts comparing the summations for the different nodes. If in figure 2.12 the summation of the coefficients $c_{2,3}$ is smaller than the total summation of the coefficients $c_{3,6}$ and $c_{3,7}$, then $node = 2^2 + 3 = 7$ is pruned away as shown by the dotted lines in figure 2.12. In that way the best basis tree structure can be found efficiently, and such an algorithm is used in this project to find the best basis tree for a given input signal.

The above described method assumes that the input signal can be used for finding the best basis tree, but that might not always be the case. In a real-time implementation it is not possible to wait for the complete input signal, before starting to process it, because that would make the delay too large. This problem will not be discussed further here, it will just be noted, that for a real-time implementation another method for finding the best basis tree, without using the input signal, needs to be found.

2.2.2 Wavelet Denoising Using Thresholding

2.2.2.1 White Noise

White noise is characterized by having its energy spread equally over all frequencies at all times. That means all the time samples, all the Fourier coefficients and all the wavelet and wavelet packet coefficients of a white noise signal will have the same expected amount of noise energy. White noise is therefore equally well (or equally bad) represented in the different domains as shown in figure 2.13, but since speech signals can be compactly represented in the wavelet domain, the wavelet packet transform can be used to effectively remove white noise from speech signals as described in the next section.

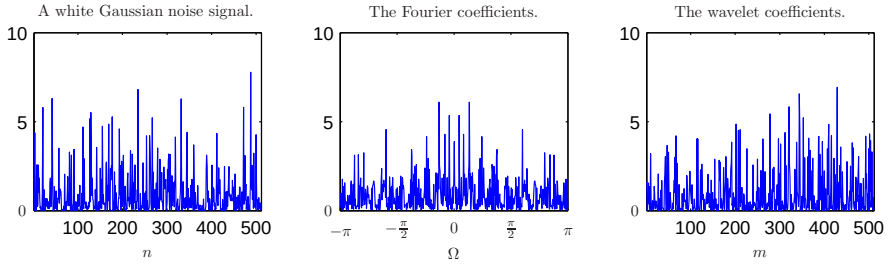


Figure 2.13: The absolute value of 512 samples of white Gaussian noise in time domain (left), Fourier coefficients (middle) and Daubechies 6 wavelet coefficients (right).

2.2.2.2 Denoising

Denoising can also be considered as a separation problem. Usually there will be a desired signal, which is corrupted by other signals considered as the noise. In order to retrieve the desired signal, the noise needs to be decreased or preferably completely removed. To do that you need to separate the desired signal from the noise, so that they can be processed differently. When the noise is white, it will be present in all wavelet packet coefficients with the same amount of energy. It is therefore impossible to completely separate the desired signal from the noise using the wavelet packet transform. But if the wavelet packet coefficients are divided into two groups, one containing all the coefficients with signal energy (the signal coefficients group), and the other containing coefficients with only noise energy (the noise coefficients group), the best possible separation of the

signal and the noise has been achieved. And clearly the fewer coefficients used to represent the signal, the less noise energy is included.

The problem is then how to determine, which coefficients contain signal energy, and which contain only noise. If the noise is white and the energy is known, its average impact on every coefficient is also know. Therefore a thresholding value (T_n) is normally calculated or estimated, and all coefficients with absolute values lower than the thresholding value are considered to mostly consist of noise, and all values above to mostly consist of signal. An example is shown in figure 2.14. All coefficients with values above the threshold are in the signal coefficients group, and all coefficients with values below the threshold are in the noise coefficients group.

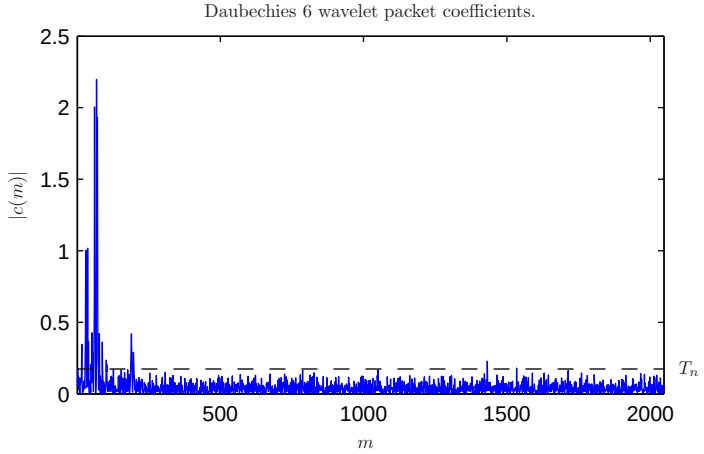


Figure 2.14: The absolute value of Daubechies 6 wavelet packet coefficients from a noisy speech signal. The black dotted line shows the thresholding value.

After the separation different thresholding methods can be used to process the two groups of coefficients, before the inverse wavelet packet transform is applied. Three of those thresholding methods are described here.

2.2.2.3 Hard Thresholding

The hard thresholding method is the easiest and most intuitive way of processing the wavelet packet coefficients. It simply sets all the noise coefficients to zero and leaves all the signal coefficients unchanged. Mathematically this can be

written as

$$f_H(x) = \begin{cases} 0 & |x| \leq T_n \\ x & |x| > T_n \end{cases} \tag{2.24}$$

2.2.2.4 Soft Thresholding

In the soft thresholding method the noise coefficients are also set to zero, but the signal coefficients are not left unchanged. If the noise is white, there will be some noise in the signal coefficients, and the thresholding value is therefore subtracted from these in order to reduce this noise contribution. The mathematical representation is

$$f_S(x) = \begin{cases} 0 & |x| \leq T_n \\ \text{sign}(x)(|x| - T_n) & |x| > T_n \end{cases} \tag{2.25}$$

The advantage of this method is that the thresholding value can normally be decreased a little compared to the hard thresholding. The reason is that if a coefficient containing only noise is just above the threshold value, it will be decrease a lot, and therefore it isn't as important, if it was just above the threshold or not. This method decreases the signal group coefficients, which normally has the effect that it smooths the output a little. If the thresholding value is set too high, the output will be smoothed too much, which of course is a drawback of the method.

2.2.2.5 Garrote Thresholding

Another interesting thresholding method is called Garrote [4]. This method is also different in the way it processes the signal coefficients and the mathematical representation is

$$f(x) = \begin{cases} 0 & |x| \leq T_n \\ x - \frac{T_n^2}{x} & |x| > T_n \end{cases} \tag{2.26}$$

In a way it is a compromise between hard and soft thresholding. When the coefficients are just above the thresholding value, it works like soft thresholding, subtracting the thresholding value from the coefficients. For the larger coefficients the amount subtracted is decreasing. Thereby it achieves the good properties of the soft thresholding method, but without smoothening the filtered signal too much. The garrote thresholding function is used for all filtering tasks in this project.

2.2.2.6 Colored Noise

When the energy of the noise signal is not evenly distributed over all frequencies, but stationary, that is the statistics of the noise are not changing with time, the noise is said to be colored. This has an implication on the threshold value, because a given value might be good around some frequencies with low noise energy, but at other frequencies, where the noise energy is bigger, it might be poor. Since the wavelet packet coefficients represent different frequency bands of the input signal, all coefficients belonging to the same frequency band, that is coming from the same output filter, can be assumed to include the same amount of noise. Hence an individual threshold value can be used for each wavelet filter output, each adapted to the average noise energy at that particular frequency band [5]. This can be viewed as a 1D thresholding function, because the thresholding value is a function of one parameter, namely the frequency.

Chapter 3

Periodic Noise and The Period Wavelet Packet Transform

In the previous sections the wavelet packet transform has been described, and how to filter stationary noise has been shortly mentioned. Before the method for filtering periodic noise is presented in section 3.2, the next section will introduce periodic noise and its characteristics.

3.1 Periodic Noise

The noise considered in this project is noise created by machinery, engines and other types of cyclic processes. The noise will, to some extend, sound like continued repetitions of the same short sound signal, and is therefore in this project denoted periodic noise. Since sounds are best described by their frequency content over time, the periodic noise can be described in the same way. The power density spectrum of periodic noise will therefore to some extend be repeated in time, and hence the repetition can be seen in time-frequency planes.

Another important aspect is the stationarity of the periodic noise. Being peri-

odic the noise can not really be said to be stationary, and only knowing that the power density spectrum of the noise is periodic with time, it doesn't necessarily make it fall under the category of cyclostationary signals. On the other hand it might be valid to say, that the periods of the noise can be stationary. If the underlying process generating the noise periods is not changing with time, the noise will be called periodically stationary. For periodically stationary noise the n'th noise period will be just as good at describing the (n+1)'th noise period as it will be at describing the (n+100)'th noise period. If that is not the case, the noise will be denoted periodically nonstationary.

In the top of figure 3.1 a part of a periodically stationary noise signal is shown in the time domain. The noise is recorded from a running car engine with a sampling frequency of $f_s = 44.1kHz$. In the plot about 6 periods of noise are shown, the period length N_T has been estimated to $N_T = 2731$ samples, and the vertical lines split the periods of the noise signal according to N_T. It can be seen that the noise signal looks somewhat periodic on such a large scale, but when zooming in the periodicity is weakened. In the bottom plot of figure 3.1 the same noise signal is shown in a time-frequency plane. The time-frequency plot is constructed using Symmlet 4 wavelets, and here the periodicity of the power spectrum is seen. The periodicity is not as clear as could be expected, which can be explained by several factors.

First the signal is a noise signal and include a certain amount of randomness. Second the wavelet coefficients might not match the period of the noise signal, more about that in the next sections. Third the period length of the periodic noise is not perfectly stable, which makes the periods appear, as if they where slightly shifted versions of each other.

3.2 Period Wavelet Packet (PWP) Transform

The periodicity of the power spectrum of periodic noise is information, which we would like to exploit, when trying to remove the noise. In cases where the noise is stationary and known to have a certain color, this information can be used to make individual threshold values for each frequency band, as described in section 2.2.2.6. This is in principle a 1D thresholding function, which only depends on the frequency. When the noise is periodic, the thresholding function also needs to be periodic with time. The suggestion is therefore, as proposed in [6], to have a specific thresholding value not only for each frequency band, but for each wavelet packet coefficient within a period. The resulting thresholding function is a 2D function, which is dependent on both time and frequency.

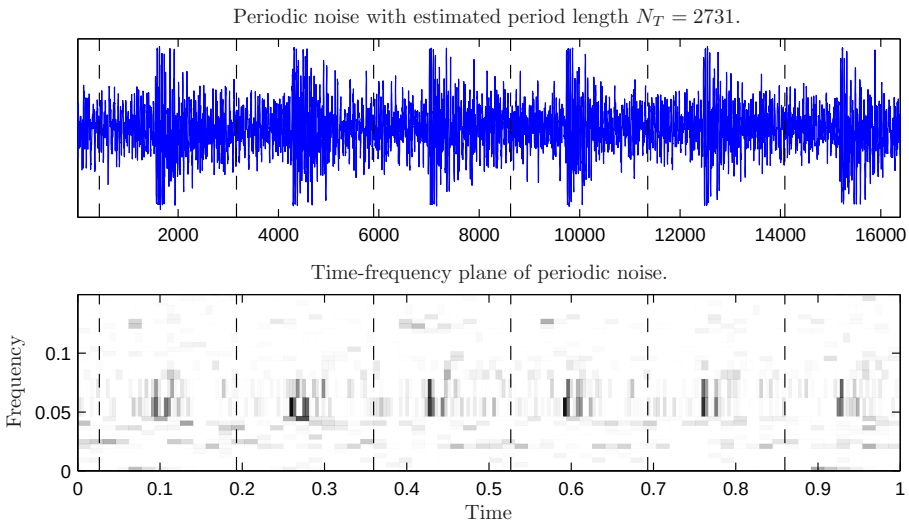


Figure 3.1: The top plot shows a part of a periodic noise signal recorded from a running car engine in the time domain. The bottom plot shows the same signal in a time-frequency plane.

The idea can easily be illustrated with an example. In figure 3.2 a speech signal (the top plot) is contaminated by a repeated chirp signal considered as a periodic noise signal (in the bottom plot).

During the first period of the noise, there is no speech, and this is therefore considered as a speech pause. In the last periods of the noise the speech is present. One can now imagine, that if the wavelet packet coefficients, obtained during the first period of the noise, are subtracted from the coefficients during the following periods, the noise will be removed. This is shown in figure 3.3.

This seems very straight forward, but as stated in [6], doing the wavelet transform of only one period of noise is not a straight forward task.

3.2.1 The Periodicity of the Wavelet Packet Coefficients

The wavelet packet transform has a limited resolution in time, and in fact as more stages are added to the filter bank, this resolution is decreasing; refer to the squares in the time-frequency plane in figure 2.12. If a whole number of squares, placed horizontally next to each other, don't match the period of the noise signal, then the wavelet packet coefficients won't be periodic. If the

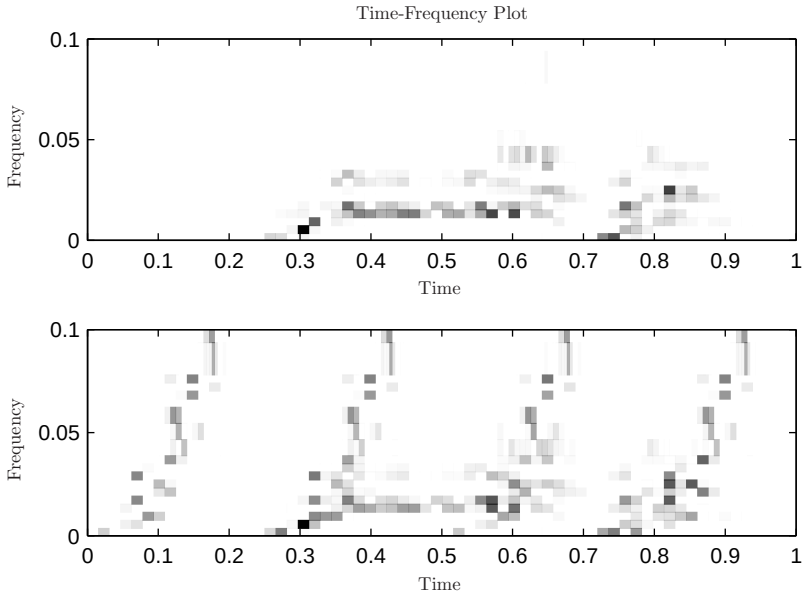


Figure 3.2: Top plot is a clean speech signal. The bottom plot is the same speech signal contaminated by a periodic chirp signal.

coefficients of the first period are then subtracted from the coefficients in the next period, the result won't be good.

The problem is illustrated in figure 3.4, where the squares in the bottom of the plot correspond to wavelet packet coefficients after 8 filter stages, and the squares in the top part to only 7 filter stages.

Here it can be seen how the top part is perfectly periodic with every chirp (period $T = 0.2422s$), while the bottom part is only periodic over two chirps (period $2T$). This is even one of the better cases, since the wavelet packet coefficients show the right periodicity through 7 filter stages. If the noise period is equal to an odd number of signal samples, the periodicity of the wavelet packet coefficients is increased to $2T$ already after the first stage.

It is important to note that the periodicity in time is not the same as the periodicity of the wavelet packet coefficients. A time period of T will correspond to $N = Tf_S$ number of signal samples, where f_s is the sampling frequency. That also means that after one filter stage in the wavelet packet transform, the time period T corresponds to $N_1 = \frac{Tf_s}{2}$ wavelet packet coefficients at the first level of the transform. If N is an odd number, then N_1 is not going to be an integer, and

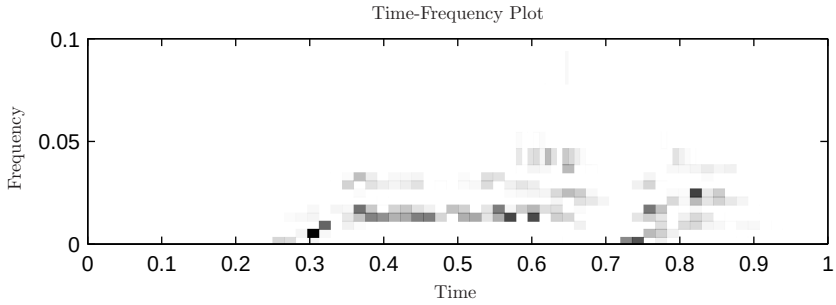


Figure 3.3: The speech signal after the noise was removed.

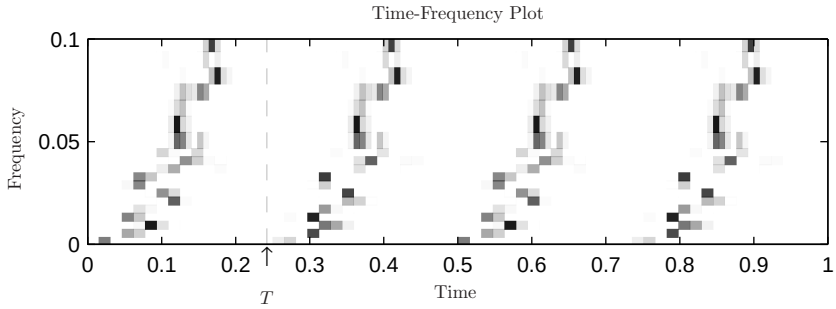


Figure 3.4: Wavelet transform of chirp signal with non-dyadic period length.

hence the periodicity of these level one coefficients will be $2N_1$ corresponding to a time period of $2T$.

Even if the noise period corresponds to an odd number of signal samples, it is still possible to use the principle of subtracting the wavelet packet coefficients from each other to remove the noise. Enough periods without speech are then needed, so that at all levels there are at least one period of wavelet packet coefficients. If, as in the worst case, the period T of the noise corresponds to an odd number of signal samples, then after 5 filter stages the wavelet packet coefficients would be periodic with a period of 2^5T. One could therefore assume that the speech pause is long enough to give sufficient periods of the noise, which might be possible. Normally the periodic noise will not be perfectly periodic though, but each period will be slightly different from each other, therefore it is desirable to extract as much information out of each period as possible. What could be done is to repeat every period enough times, so that all the wavelet packet coefficients get periodic; this would increase the number of computations drastically, but would be a solution to the problem.

3.2.2 Sorting Wavelet Packet Coefficients Instead of Down-sampling

The approach taken in [6] is in a way similar to that. Instead of repeating the noise periods, before applying the wavelet packet transform, it does the wavelet packet transform without down-sampling, and does a special kind of sorting instead. If the down-sampling is not done at each stage, it is possible to get all the information out of just one period of noise exactly as if the period was repeated.

To see how the sorting works let's assume that the periodic noise has a period of $N_T = 10$. In figure 3.5 two periods of the noise are shown in the first row. The noise is fed into a wavelet packet transform.

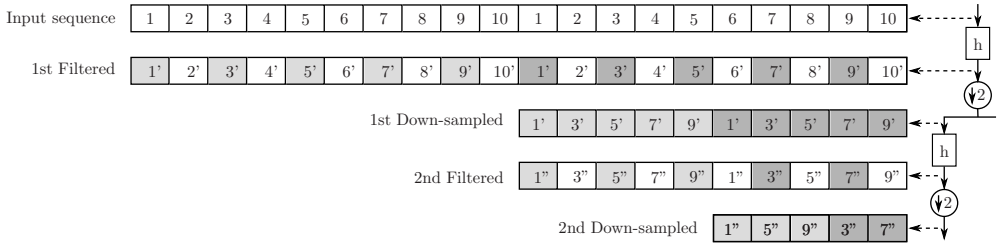


Figure 3.5: The wavelet packet transform of a periodic sequence.

After the sequence has been filtered (circular convolution) at the first stage, the sequence is still periodic with $N_T = 10$. The down-sampling results in the sequence in the third row of figure 3.5. The period of the sequence is now $N_T = \frac{10}{2} = 5$. Going through another filter stage and down-sampling, the samples in row five are obtained and $N_T = 5$. If this is continued, the period will remain $N_T = 5$ at all lower stages. Now during the analysis of one noise period, the samples should be arranged in the same way as in figure 3.5. How that is done is shown in figure 3.6.

In the first row one period of noise is shown ($N_T = 10$). After the first filtering stage, instead of down-sampling the samples are reordered, so that only the odd numbered samples are taken, and then repeated twice to maintain the same number of samples at each stage. The result is shown in the third row. The period is now $N_T = 5$, which is odd, but since there are two periods, the signal can be considered as having an even period of $N_T = 10$, and so after the next filtering stages, the reordering can be repeated and the sequence in the fifth row is obtained. One can see that the sequences after the reordering (row three and five) are matching the ones in figure 3.5.

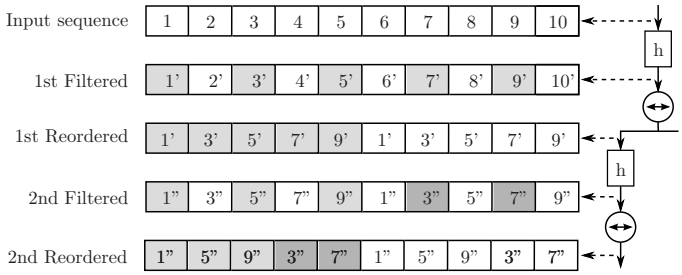


Figure 3.6: The PWP transform of one period of noise.

If there weren't two periods in the fourth row (only the samples 1", 3", 5", 7" and 9"), it would still be possible to obtain the sequence in the fifth row by first taking the odd samples and then the even samples of just one period in row four. Therefore if the length of the input noise sequence is odd, first the samples at the odd places are taken, and then the samples at the even places. That way the sequence continues to have the same length and the period also remains the same.

Now it can be summarized how the sorting is done. If the period of the noise is even then odd samples are taken and repeated. If the period is odd, first the odd samples are taken followed by the even samples. The wavelet packet transform, when using this reordering instead of normal down-sampling is called the period wavelet packet (PWP) transform, and as seen it can be applied to sequences of any length.

It can here be noted, that the above described scheme, which is given in [6], can be speeded up a little. If the noise period is even, there is no reason to repeat the down-sampled sequence, since that is in principle just causing more computation in the following filtering stages. Instead a normal down-sampling can be done, and the period of the down-sampled sequence needs to be remembered. When the period then becomes odd, the scheme should be switched, and the following stages should continue as if the noise period was odd. That is by first taking the odd samples and then even samples. By changing the scheme periodic noise with an even period N_T requires almost only half the number of computations, when N_T is dividable by four only a little more than one fourth of the computations, and so on. In a time critical implementation, this will therefore be an important improvement.

3.2.3 Obtaining the Thresholding Packet

When the samples from one period have been obtained, they are combined with samples from the following periods until the speech signal starts. In [6] an averaging formula with a forgetting factor λ is suggested

$$\begin{aligned} F_1(d,b,n) &= P_1(d,b,n) \\ F_k(d,b,n) &= \frac{\sum_{i=1}^{k} \lambda^{k-i} P_i(d,b,n)}{\sum_{i=1}^{k} \lambda^i} \end{aligned} \tag{3.1}$$

where $F_k(d,b,n)$ is the averaged noise energy distribution after k periods, $P_i(d,b,n)$ is the analyzed noise energy distribution of period i, that is the PWP coefficients found as described above, and λ is a forgetting factor. λ is a chosen value between zero and one, where one means that no periods are forgotten, and smaller values gives the PWP coefficients of old periods a smaller weight in the average compared to new coefficients. This is relevant for periodically nonstationary noise, where consecutive noise periods will be more alike than periods further apart.

The equations can be combined to a recursive equation

$$F_{k+1}(d,b,n) = \frac{P_k(d,b,n) + F_k(d,b,n) \sum_{i=1}^{k} \lambda^i}{\sum_{i=1}^{k} \lambda^i}. \tag{3.2}$$

The thresholding coefficients can be obtained and continuously updated during speech pauses using the above equation and the PWP transform. The variable $P_k(d,b,n)$ contains one of each of the PWP coefficients in the period k, and hence $F_k(d,b,n)$ contains the same number of coefficients just averaged over the last periods. The function in equation (3.2) will be called the average thresholding packet.

Averaging the PWP coefficients over several periods seems like a good approach for estimating the noise level at a given coefficient. There will of course be a lot of noise coefficients above the average, so to use the average values for thresholding something needs to be added or multiplied to the average values. But without knowing the distributions of the coefficients, the variance might be very different for different coefficients, and therefore a good value to add or multiply a given coefficient with might be too small for other coefficients.

To avoid that problem a new updating function is constructed, which instead of averaging the PWP coefficients take the max of the coefficients. In that way the likelihood that noise, when the speech is present, is going to be above the threshold is very low, and it will therefore also be less needed to multiply or

add anything to the thresholding coefficients. The max thresholding packet is obtain using the following equation

$$F_{k+1}(d,b,n) = \max\left\{P_k(d,b,n), F_k(d,b,n)\lambda\right\}. \tag{3.3}$$

There is a chance that very big PWP coefficients are going to drive the thresholding values too high, and therefore it will be more important to use a forgetting factor λ, which is smaller than one, when periodically nonstationary noise is processed.

3.2.4 Problem With Finite Length Sequences

As stated in section 2.1.2.3, filtering finite length sequences is done by circular convolution. This actually causes a problem when the wavelet coefficients of the noisy speech signal are thresholded using the thresholding coefficients. The length of the input sequence to the standard wavelet packet transform is normally required to be $N = 2^L$, or at least a length as given by equation (2.12). This means that, when the input signal is periodic, the length can be written as done in [6]

$$N = kT + \Delta T, \qquad 0 \leq \Delta T < T \tag{3.4}$$

where k is an integer, T is the period of the signal and ΔT is the length of the last unfinished period. The last period of the signal is therefore in general not complete as shown in figure 3.7.

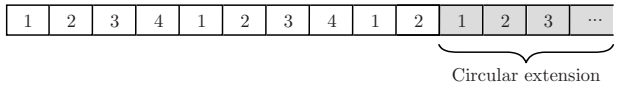


Figure 3.7: Circular extension of periodic signal. The last period is incorrectly extended.

When the circular convolution is done, it is necessary to extend the signal, but that actually destroys the periodicity of the signal in the last period. The result is that some of the wavelet packet coefficients at the edge of the signal will not be periodic as all the other coefficients. This is in principle not a problem, since it is still possible to do the inverse calculation and reconstruct them again. When the thresholding is done, using the coefficients from the PWP transform, the coefficients at the edge of the signal will not match any coefficients in the thresholding packet. This might seem like a minor problem when the input sequence is very long, but the number of edge coefficients of each filter output can be shown to remain constant after a few filter stages. Therefore in very deep filter banks the edge coefficients might end up being a substantial part of the low level coefficients.

3.2.4.1 One Approach Using Periodic Extension

The problem was already realized in [6], and the suggested solution was to change the circular convolution in the standard wavelet packet transform. Instead of doing the normal extension, one could do a periodic extension as shown in figure 3.8.

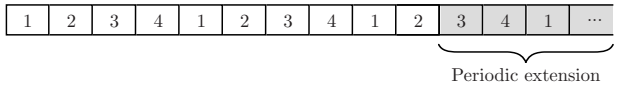


Figure 3.8: Periodic extension of periodic signal. The last period is correctly extended.

This solves the problem with the special coefficients at the edge of the input signal, but causes another problem. When the signal is down-sampled at each stage in the wavelet packet transform, at some level there will not be enough samples to represent a whole period. When that happens the periodic extension can't be done anymore, since the samples needed for the periodic extension are not available. Therefore the standard wavelet packet transform is only done down to a critical dept, after which the filter coefficients at the lower levels are calculated as by the PWP transform, which maintains the same number of samples at each level.

The principle of doing the periodic extension works, when the signal transformed is perfectly periodic. The input signal we want to transform is a periodic noise signal, which is normally not perfectly periodic, plus a speech or sound signal, which means that the total input signal is actually not really periodic. When that is the case, one will see that doing the periodic extension instead of the normal circular extension, makes it impossible to perfectly reconstruct the edge coefficients at each filter stage.

If the circular extension is kept, the solution would be to extend the signal in both ends. This would leave enough information in the filtered and down-sampled sequence to reconstruct all the original samples again, but instead of decreasing the number of samples at each filter stage by a factor two, there will be $n = \frac{N+N_f}{2}$ number of samples after each stage. This also results in a change of the inverse wavelet packet transform, since there is no longer any need for doing any extensions for the circular convolution and a standard convolution can be used instead.

In a real-time implementation of the filtering scheme, the input sequence can be considered infinite in length, and the circular convolution is replaced by a

standard convolution. Therefore the above changes will be irrelevant for such an implementation and will just complicate a direct conversion of the scheme from the off line version to the real-time implementation.

3.2.5 Calculating Thresholding Coefficients for the Edge Coefficients

It is possible to deal with the described problem in another way, where the periodic extension is dropped and the normal circular extension is used instead. This has the benefit, that a standard wavelet packet transform can be used, and the only problem needed to be solved is the mismatch between the edge coefficients and the thresholding coefficients obtained using the PWP transform. Also when the scheme is converted to a real-time implementation, the problem with the edge coefficients can just be left out, since the signals can be considered as infinite in length, and the circular convolutions are exchanged with normal convolutions.

All the samples in the thresholding packet are needed for thresholding the periodic (inner) sections of the wavelet packet coefficient sequences of the noisy speech. But new thresholding coefficients can be calculated from each period of pure noise to use at the edges. Since the edge coefficients appear, because of the uncorrectly extended last period of the noisy speech signal, the same extension needs to be done to the periods of pure noise, which are analyzed with the PWP transform during speech pauses. In figure 3.9 a periodic sequence is filtered using a wavelet packet transform. Period $N_T = 10$ and $N_f = 4$.

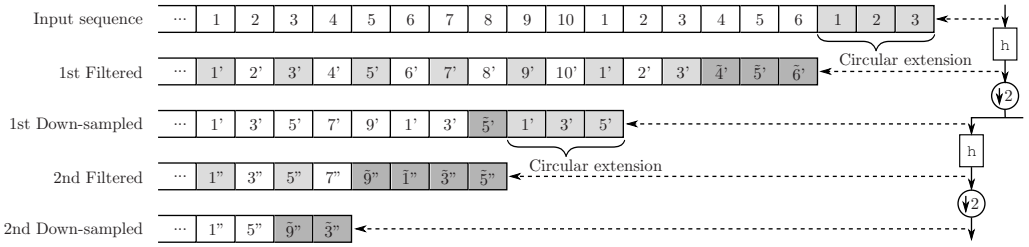


Figure 3.9: A periodic sequence filtered by a wavelet packet transform with length $N_f = 4$ filter. The dark grey samples also marked with a $\sim$ are edge samples.

The sequence is circularly extended causing the last period to be erroneous, and the last three convolution coefficients to be non-periodic. These non-periodic

samples are what until now have been called the edge coefficients. As seen in the figure these coefficients after down-sampling travel on into the next filter stage, where they along with the circular extension cause even more samples to be non-periodic. The number of edge coefficients at each stage depends on the number of filter coefficients (N_f) in the wavelet packet transform filters, and the number of edge coefficients from the previous stage. Luckily the number of edge coefficients doesn't continue to grow, but becomes constant after a few filter stages and is maximally $nEdge = N_f - 1$. The edge coefficients are calculated using the same wavelet packet transform as the noisy speech.

First step is building the first input sequence of $2(N_f - 1)$ pure noise samples taken to match the samples in figure 3.9. That would for the above example be noise sample number 4, 5, 6, 1, 2 and 3 (the same numbers as in the end of the first row). Then the filtering and down-sampling is done, and at the following filter stages new sequences are built of the edge coefficients calculated at the previous stage and PWP transform coefficients from the corresponding stage already calculated using the PWP transform.

3.2.6 Conclusion of the PWP Transform Filtering Method

To conclude, the filtering method consists of calculating thresholding coefficients for each pure noise period using the PWP transform. After these coefficients have been obtained the edge coefficients, which are also used as thresholding coefficients, can be calculated as described above. As long as there is no speech this is continued on each pure noise period, and the thresholding packet is updated as described by equation (3.2) or (3.3). When the speech is present it is filtered by a normal wavelet packet transform, then thresholded period by period using the coefficients from the average or max thresholding packet. Finally an inverse wavelet packet transform is used on the thresholded coefficients, which results in the cleaned speech signal.

The scheme was already tested in [6] using a wavelet packet transform with periodic extension instead of circular extension, as described in section 3.2.4.1, with good results. Using circular extension and edge coefficients plus the max instead of the average thresholding packet, the results should already be improved, but there is another important area, which could also be improved. Referring to section 3.1 it was mentioned how consecutive periods of periodic noise might look like slightly shifted versions of each other. That is a big problem, when using the wavelet packet transform, because it is very shift variant. Therefore shifted versions of the same input signal might result in very different wavelet packet coefficients. When that is the case, then the thresholding values obtained during one period of pure noise will not match the noise in the next period, where the

speech is present. An important improvement would therefore be to make the wavelet packet transform shift invariant, or maybe exchange it with a similar but shift-invariant transform.

Chapter 4

Shift Invariance and Complex Wavelet Packets

As stated in the previous section an improvement to the PWP transform method would be to incorporate shift invariance in the wavelet packet transform. Shift invariance is in many problems a very desirable property, and there have been several attempts to construct shift invariant wavelet transforms.

4.1 Shift Invariant Real Wavelet Transforms

The most known and straight forward approach is the undecimated wavelet transform also called the Algorithme à Trous [1]. This transform uses a filter bank as the one shown in figure 2.4, but without all the down-samplers. This algorithm largely increases the number of computations, and results in a large redundancy, since the number of coefficients is doubled with each stage. Further it should be noted that, when shifted inputs are transformed using the algorithm, the outputs will also be shifted versions of each other. This is not a form of shift invariance, which is easily used in the processing scheme described in the last chapter. The shifts in the PWP coefficients would need to be tracked and shifted all the time to be correctly combined to a usable thresholding packet. Also when the speech signal is present, the shift in the noise should be estimated

in order to align the coefficients with the thresholding packet before performing the thresholding.

Another interesting approach is called the shift invariant discrete wavelet transform (SIDWT) [7]. This transform basically uses the choice of wavelet packet basis to obtain shift invariance. For that purpose the set of wavelet packet bases is expanded, so that shifted versions of all the bases are included in the set. When finding the best basis all the shifted versions are searched and the one matching the signal best is chosen, call that basis A. If the signal is shifted, the best basis search will result in a shifted version of basis A, hence the wavelet packet coefficients will exactly match the previous ones. The shift invariant result of the method is very good, but the way it is achieved is problematic. When doing the filtering, a new basis would need to be found for each period of the noisy speech signal. The bases should of course not be selected from all bases, but only from shifted versions of an initially chosen basis. The choice of the basis would still be difficult, since it should preferably only be determined according to the periodic noise in the noisy speech signal. This, along with the fact that the inverse transform applied after thresholding should use the same bases as the forward transform, would greatly complicate the method, and make it hard to convert the implementation into a real-time scheme.

4.1.1 Getting Help From Complex Representation

Shift invariance is a very well known property of the Fourier transform, where any shift of the input signal only results in a phase change of the Fourier coefficients. This form of shift invariance is obtained through complex coefficients, which the Fourier transform naturally produces by having complex basis functions. These basis functions consist of a cosine and a sine, which are identical functions offset by a 90 degree phase shift, and thereby forming a Hilbert transform pair. The Hilbert transform ($\mathcal{H}$) is easiest described in the frequency domain, where the frequency response is [8]

$$\mathcal{H}(\omega) = \begin{cases} i & \omega < 0 \\ 0 & \omega = 0 \\ -i & \omega > 0 \end{cases} \tag{4.1}$$

where $i = \sqrt{-1}$ is the imaginary unit.
When a signal is added to its Hilbert transform times i as in the Fourier transform

$$e^{i\phi} = \cos(\phi) + i\sin(\phi) \tag{4.2}$$

the resulting signal is called an analytic signal.
Analytic signals are characterized by having only positive frequencies that is

$$F(\omega) = 0, \qquad \omega < 0 \tag{4.3}$$

which is a direct result of the Hilbert transform and the multiplication by i.

An approach to copy the good shift invariant property of the Fourier transform would be to make the wavelet basis functions analytic. Unfortunately a time limited signal can not be limited in frequency, and in more general can not be zero on a finite frequency interval. Therefore analytic signals must be infinite in time, and as a result the time limited basis functions of the wavelet transform can not be perfectly analytic. Knowing this, research has focused on developing time limited approximately analytic wavelet basis functions, and successful achievements have resulted in the Dual Tree Complex Wavelet Transform [9] described in the next section.

4.2 The Dual Tree Complex Wavelet Transform

The Dual Tree Complex Wavelet Transform (DTCWT) has been developed to incorporate the good properties of the Fourier transform in the wavelet transform. As the name implies two wavelet trees are used, one generating the real part of the complex wavelet coefficients tree $\mathfrak{Re}$ and the other generating the imaginary part tree $\mathfrak{Im}$ [9]. The structure is illustrated in figure 4.1.

It should be noted that there are no links between the two trees, which makes it easy to implement them in parallel. Also the filters in the two trees are different, and the filters in the first stage of each tree are different from the filters in all the later stages. Why that is necessary will be described in section 4.2.2.2. Further there is no complex arithmetic involved in any of the trees. The complex coefficients are simply obtained as

$$d_j^{\mathbb{C}}(k) = d_j^{\mathfrak{Re}}(k) + i d_j^{\mathfrak{Im}}(k) \tag{4.4}$$

and the complex wavelet basis functions are given by

$$\psi_{j,k}^{\mathbb{C}}(n) = \psi_{j,k}^{\mathfrak{Re}}(n) + i\psi_{j,k}^{\mathfrak{Im}}(n) \tag{4.5}$$

The inverse DTCWT is calculated as two normal inverse wavelet transforms, one corresponding to each tree, and the results of each of the two inverse transforms are then averaged to give the reconstructed signal. Again, there is no complex arithmetic needed, since the $d_j^{\mathbb{C}}(k)$ coefficients are split up into $d_j^{\mathfrak{Re}}(k)$ and $d_j^{\mathfrak{Im}}(k)$, before they are used in the corresponding inverse transforms.

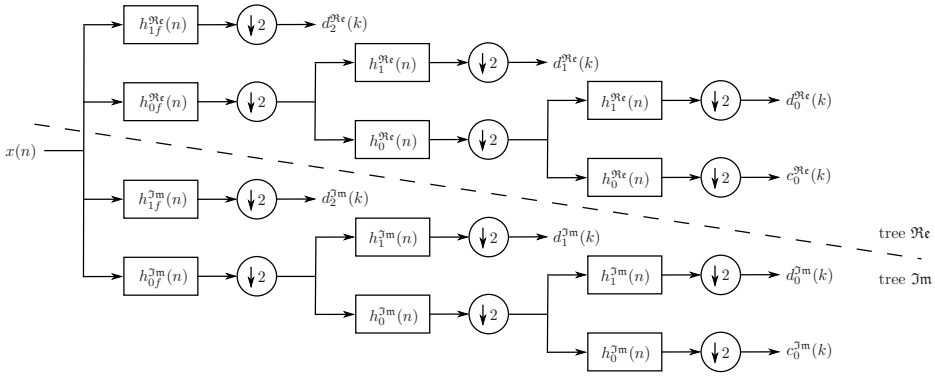


Figure 4.1: Filter bank for the dual tree complex wavelet transform.

4.2.1 Filter Requirements

As was discussed in section 4.1.1 complex coefficients can be obtained by projection onto a Hilbert transform pair together constituting an analytic signal. Therefore the wavelet basis functions of tree $\mathfrak{Im}$ have to be the Hilbert transform of the basis functions of tree $\mathfrak{Re}$. Since the basis functions are determined by the wavelet filters through equations (2.6) and (2.7), the design of wavelet basis functions is normally turned into a filter design problem by translating the design criteria into filter criteria. This has also been done for the DTCWT, where the relation between the impulse responses of the scaling function filters in the two trees can be written as [10]

$$h_0^{\mathfrak{Im}}(n) = h_0^{\mathfrak{Re}}\left(n - \tfrac{1}{2}\right) \tag{4.6}$$

This makes the filters satisfy the requirement of the tree $\mathfrak{Im}$ wavelet functions being the Hilbert Transform of the tree $\mathfrak{Re}$ wavelet functions. In the frequency domain the equation can be translated into the following relations

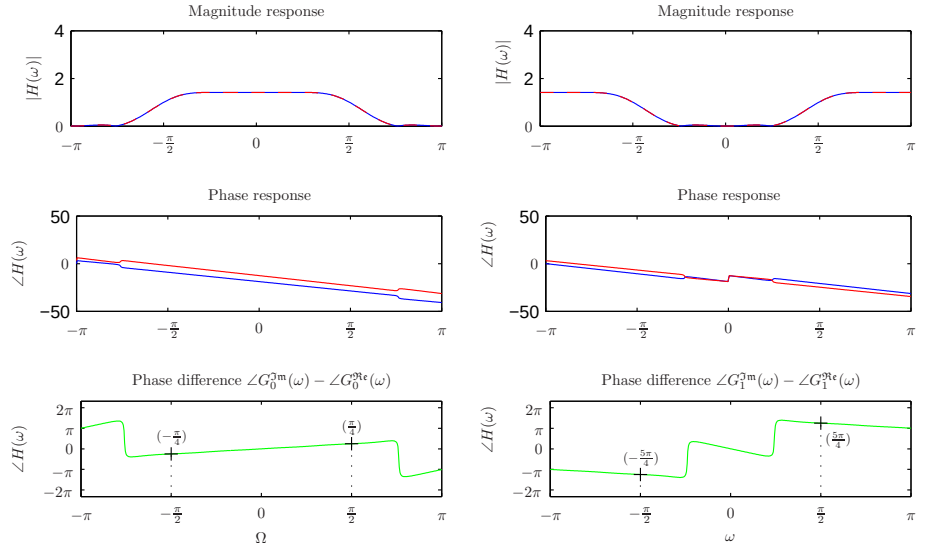
$$|H_0^{\mathfrak{Im}}(\omega)| = |H_0^{\mathfrak{Re}}(\omega)| \tag{4.7}$$

$$\angle H_0^{\mathfrak{Im}}(\omega) = \angle H_0^{\mathfrak{Re}}(\omega) - \tfrac{1}{2}\omega \tag{4.8}$$

Unfortunately these equations can not be perfectly satisfied simultaneously by finite length FIR filters, which is equivalent to the fact stated in section 4.1.1, that wavelet functions forming an analytic signal can not have a finite length.

As a result different filter design methods have been developed to design wavelet filters of different lengths approximating (4.7) and (4.8). One of these methods generates what is called q-shift filters and is described in [11]. The q-shift filters perfectly fulfill (4.7), but only approximate (4.8). Since the basis functions are

important, and these are constructed using the inverse DTCWT, the frequency responses of length 14 q-shift filters used in the inverse DTCWT are shown in figure 4.2. Also the phase difference between the filters in the two trees is plotted, and it is seen that in the lowpass filter passbands the phase difference approximates $\frac{1}{2}\omega$. This is not a negative slope as stated by equation (4.8), which comes from the fact that the plots show the filters ($g_0^{\mathfrak{Re}}(n)$ and $g_0^{\mathfrak{Im}}(n)$) in the inverse DTCWT, which are reversed versions of the filters in the forward transform. The filter coefficients for the q-shift filters for both the forward and inverse transforms are given in table B.1 and B.2 respectively in appendix B.



(a) Lowpass filter responses of length 14 q-shift filters. (b) Highpass filter responses of length 14 q-shift filters.

Figure 4.2: Transfer functions of length 14 q-shift filters used in the inverse DTCWT. Tree $\mathfrak{Re}$ is red, tree $\mathfrak{Im}$ is blue and the phase difference ($\mathfrak{Im}$-$\mathfrak{Re}$) is green.

4.2.2 Constructing Analytic Basis Functions Using the Inverse DTCWT

From equation (4.1) the necessary relationship between the basis functions in the two trees can be written as

$$|\Psi^{\mathfrak{Im}}(\omega)| = |\Psi^{\mathfrak{Re}}(\omega)| \tag{4.9}$$

$$\angle\Psi^{\mathfrak{Im}}(\omega) - \angle\Psi^{\mathfrak{Re}}(\omega) = \begin{cases} \frac{1}{2}\pi + (2\pi)m & \omega < 0 \\ 0 & \omega = 0 \\ \frac{3}{2}\pi + (2\pi)m & \omega > 0 \end{cases} \qquad m = ...,\ -2,\ -1,\ 0,\ 1,\ 2,\ ... \tag{4.10}$$

These equations state that the magnitude spectrums of the basis functions in the two trees have to be equal, and that the difference of the phases has to be a kind of step function equal to for instance $-\frac{3}{2}\pi$ for the negative and $\frac{3}{2}\pi$ for the positive frequencies.

As mentioned in section 2.1.2.2 a wavelet basis function can be calculated using the inverse wavelet transform. Therefore to investigate how the basis functions in tree $\mathfrak{Im}$ is related to the basis functions in tree $\mathfrak{Re}$, the calculation of a basis function is performed in the $\mathfrak{Im}$ and $\mathfrak{Re}$ inverse wavelet transforms simultaneously, and the results are shown in the frequency domain step by step.

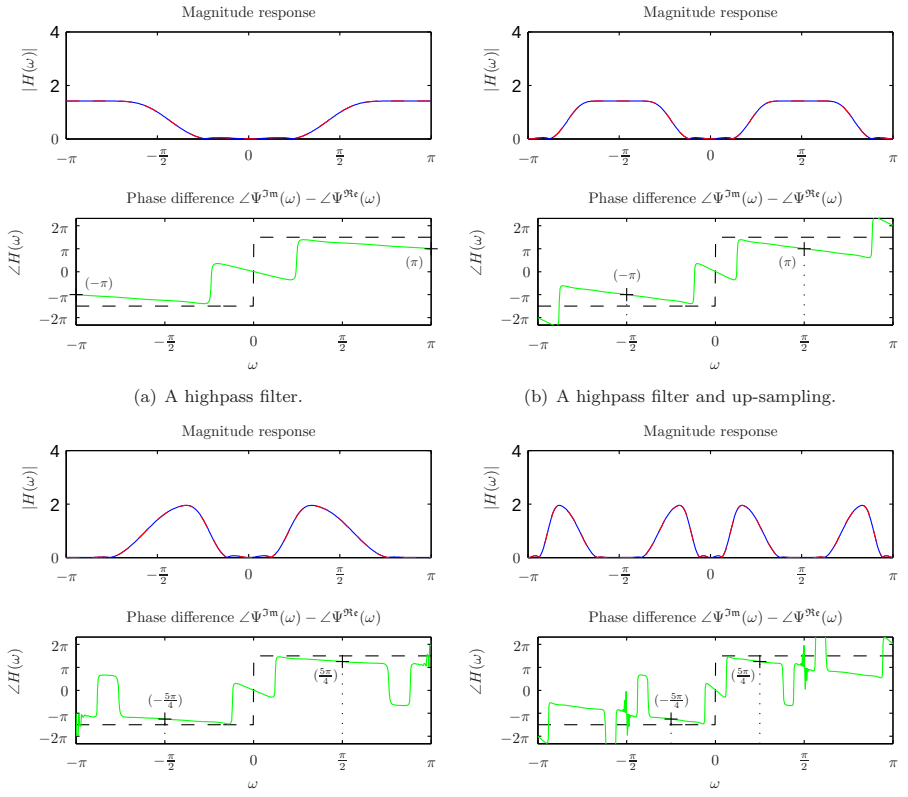
Setting the coefficient $d_0^{\mathbb{C}}(1) = 1 + i1$ and setting all other coefficients to zero will - using the inverse DTCWT - construct the basis function $\psi_{0,1}^{\mathbb{C}}(n)$. Now refer to the inverse wavelet transform filter bank, which was illustrated in figure 2.5 to keep track of the components encountered in the inverse DTCWT. Remember that there are two parallel filter banks, and in both all coefficients are set to zero except the $d_0(1) = 1$.

4.2.2.1 Stepping Through The Inverse DTCWT

Now the first component encountered in the inverse DTCWT is an up-sampler. This up-sampler is only going to add zeros between the existing coefficients, which won't have any important influence here.

The next component is a highpass filter. The single non-zero coefficient will result in the impulse response of the highpass filters, which in the frequency domain is the filter transfer function. This is shown in figure 4.3(a), which is the same as given in figure 4.2(b). Note that the phase plot is the phase difference between the two inverse filter banks and not the actual phase of the

filters, and that the black dotted line illustrates the Hilbert transform criteria in equation (4.10).



(a) A highpass filter.

(b) A highpass filter and up-sampling.

(c) A highpass filter, up-sampling and a lowpass filter.

(d) A highpass filter, up-sampling, a lowpass filter and up-sampling.

Figure 4.3: Frequency domain relation between tree $\mathfrak{Im}$ and tree $\mathfrak{Re}$ in the inverse DTCWT after different operations.

The basis functions go unchanged through the adder, because apart from the basis functions there is nothing but zeros in the inverse DTCWT.

At the next stage the basis functions are first up-sampled, which results in a compression or a scaling of the basis function spectrum as illustrated in figure 4.3(b). This up-sampling doubles the slope of the phase difference from $-\frac{1}{2}\omega$ to -1ω. Further it moved the center points of the passband regions from $\Omega = \pm\pi$ to $\Omega = \pm\frac{1}{2}\pi$, which are $\frac{1}{2}\pi$ away from the black line.

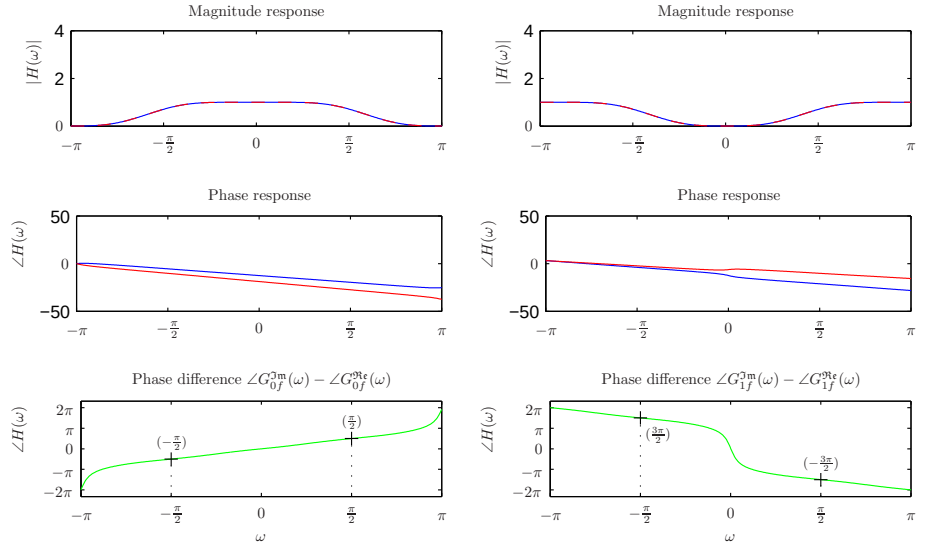
Following the up-sampler is a lowpass filter, and its transfer function is shown in figure 4.2(a). This lowpass filter can be applied by multiplying the magnitude response with the one in figure 4.3(b) and adding the phase difference plots, the result is shown in figure 4.3(c). The positive phase difference slope of the lowpass filters changes the slope from -1ω back to $-\frac{1}{2}\omega$. Also the passband center points are moved $\frac{1}{4}\pi$ closer to the black dotted line, that is half of the previous distance.

The lowpass filter did half the job of making the phase difference fulfill the Hilbert transform criteria given by the black dotted line. If the lowpass filter is applied again the phase difference criteria will be fulfilled. This is not the case though, because in the next stage through the inverse DTCWT the first component will be an up-sampler. This will as shown in figure 4.3(d) again double the slope of the phase difference, and move the passband center points to $\Omega = \pm\frac{1}{4}\pi$, half the way inward toward $\Omega = 0$. A following lowpass filter will therefore again only do half the job of getting to the black dotted line. It will decrease the slope steepness again and move the center points by $\frac{1}{8}\pi$. In that way no matter how many stages the inverse DTCWT has, the basis function phase difference will still have a slope changing from $-\pi$ to $-\frac{1}{2}\pi$ and back in the up-sampler and the lowpass filter. Also the center points in the passband regions will be moved further and further inward toward $\Omega = 0$ resulting in less changes in these points by the lowpass filters, because the phase difference of the lowpass filters decreases toward $\Omega = 0$.

4.2.2.2 The First Stage Filters

To get all the way to the black line also for only a few stages in the inverse filter bank, the lowpass filters in the first stage are different from the other filters. By having a phase difference slope of 1ω, the phase difference of the basis functions will be made flat. Additionally the center points in the passband regions will be moved double the distance compared with applying the lowpass filters in the other stages, and hence all the way to the black dotted line. A usable filter set for the first stages filters has been downloaded from [12], and their frequency responses are given in figure 4.4. The filter coefficients for both the forward and the inverse transforms are given in table B.3 and B.4 respectively in appendix B.

When continuing the construction of the basis functions through the inverse DTCWT and applying the first stage filters to the frequency response given in figure 4.3(d), the result is the frequency response in figure 4.5(a). It can be observed that the Hilbert transform criteria is approximately fulfilled in the passband regions, and when using equation (4.5) the resulting complex basis



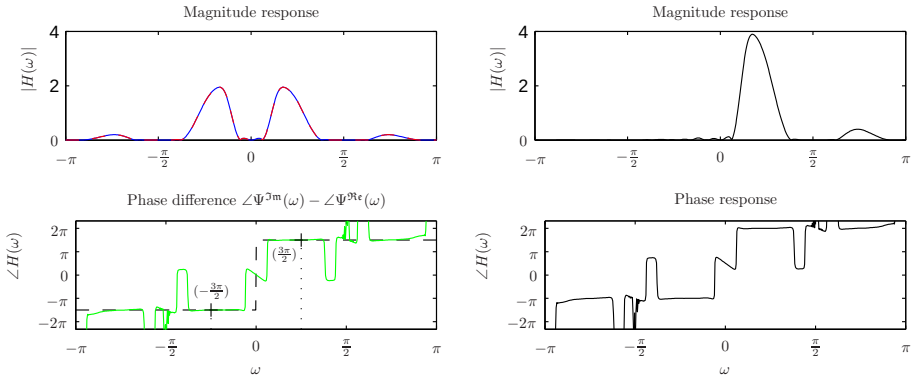
(a) Lowpass filter responses of length 10 first stage filters.

(b) Highpass filter responses of length 10 first stage filters.

Figure 4.4: Transfer functions of length 10 first stage filters used in the inverse DTCWT. Tree $\mathfrak{Re}$ is red, tree $\mathfrak{Im}$ is blue and the phase difference ($\mathfrak{Im}$-$\mathfrak{Re}$) is green.

function will be nearly analytic with a frequency response as shown in figure 4.5(b).

Note that not every basis function is constructed by going through first a highpass filter and then one or more lowpass filters in the inverse DTCWT. The scaling function is constructed by going through only lowpass filters, and the highest frequency wavelet function is constructed by going through only the first stage highpass filter. These two basis functions will therefore not be nearly analytic in the sense of having only positive frequencies, but the rest of the basis functions will, as shown in figure 4.6. In this and the following illustrations, only one basis tree will be shown, since only the structure of the tree is important. In the implementation two trees with the given structure are used to calculate the real and imaginary parts of the complex wavelet coefficients.



(a) A highpass filter, up-sampling, a lowpass filter, up-sampling and a first stage lowpass filter.

(b) Nearly analytic basis function.

Figure 4.5: Frequency domain relation between tree $\mathfrak{Im}$ and tree $\mathfrak{Re}$ in the inverse DTCWT after a series of operations, and the spectrum of the resulting nearly analytic basis function.

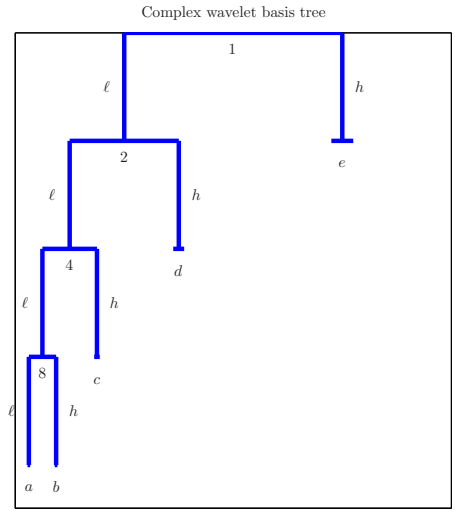
4.3 Expanding the DTCWT to Complex Wavelet Packets

The normal (real) wavelet transform is easily extended to wavelet packets, and the structure of the DTCWT doesn't impose any apparent difficulties either. Just apply filter stages to the outputs of the highpass filters in both trees, and the DTCWT is extended to wavelet packets. This has also been done in [13] with an earlier type of DTCWT filters described in [14]. Unfortunately (not considered in [13]) the new complex wavelet packet basis functions are not all analytic like the DTCWT basis functions are, and when that is desired the extension is not as straight forward.

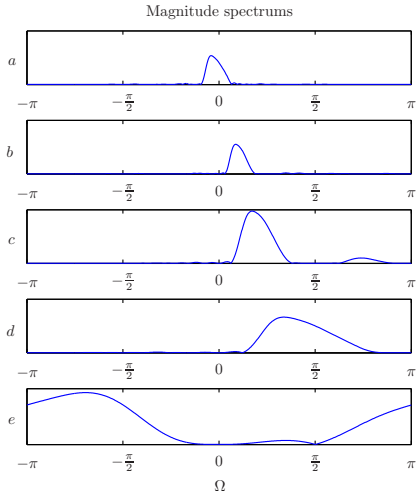
4.3.1 Problems With Straight Forward Expansion

In figure 4.7 the basis tree of a wavelet packet configuration is plotted to the left, and to the right the magnitude spectrum of four of the resulting basis functions. It is shown there how none of the basis functions are nearly analytic.

Notice how the low- and highpass filters in figure 4.7(a) are switched in the branches after the highpass filter. This is done to keep the filter outputs ordered according to frequency content, that is the left most output (a) gives the



(a) The standard wavelet basis tree, ℓ marks the lowpass filters and h the highpass filters.



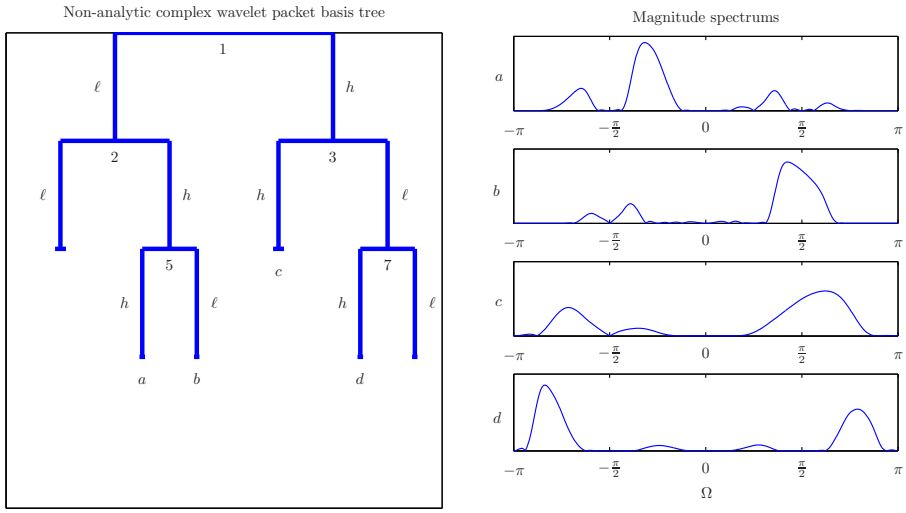
(b) Magnitude spectrums of the basis functions.

Figure 4.6: Standard basis tree and magnitude spectrums of the corresponding basis functions.

lowest frequencies, and the right most output (d) gives the highest frequencies. The reason for the switching comes from down-sampling of the outputs of the highpass filters and is explained in section 2.2.

To figure out why the wavelet packet basis functions aren't analytic, it is useful to consider exactly how the DTCWT basis functions get analytic through the inverse DTCWT. In section 4.2.2 it was shown that to get an analytic spectrum, a complicated sequence of up-sampling and filtering is used. It could seem like only a path through the inverse DTCWT going through first a highpass filter and then a sequence of lowpass filters ending with the first stage lowpass filter will ensure an analytic basis function. The condition for the final phase difference to be flat can be formulated a little more general though.

The first phase difference introduced by a ($\mathfrak{Re}$, $\mathfrak{Im}$) filter set in the inverse DTCWT has to have a slope with a given steepness α. Further all the following ($\mathfrak{Re}$, $\mathfrak{Im}$) filter sets must have a phase difference with a slope of $-\alpha$, and the last ($\mathfrak{Re}$, $\mathfrak{Im}$) filter set (first stage filter) must have a phase difference slope of -2α. This ensures that the phase difference will end up being flat. This is, as given by equation (4.8), of course not enough for the basis functions to be nearly analytic, but is an important start.



(a) A wavelet packet basis tree, ℓ marks the lowpass filters and h the highpass filters.

(b) Magnitude spectrums of the basis functions.

Figure 4.7: A wavelet packet basis tree and magnitude spectrums of the corresponding basis functions.

That not a lot of the wavelet packet basis functions are nearly analytic comes from the fact, that most of the paths through the inverse complex wavelet packet transform don't fulfill the above described phase difference requirement. The problem with these non-analytic complex wavelet packet basis functions is not trivial, and it might seem like new filters need to be designed in order for the basis functions to be nearly analytic, but fortunately that is not necessary. The key is to remember, that it is the phase difference of the filters in the two trees that is important, not the actual phase of the filters. In a given node it is possible to switch the filters between tree $\mathfrak{Im}$ and $\mathfrak{Re}$, which will mirror the phase difference around $\Omega = 0$, so that the phase difference for the positive frequencies is mirrored on to the negative frequencies and opposite. This will thereby also change the slope of the phase difference from positive to negative or opposite. Also the same filters can be used in both trees, which would make the phase difference zero, and hence leave the phase difference unchanged. These observations make it possible to change the way the basis functions are created in the inverse complex wavelet packet transform, and this is enough to achieve nearly analytic basis functions, as will be described in the next section.

4.3.2 Achieving Nearly Analytic Wavelet Packet Basis Functions

It is helpful to divide the wavelet packet basis tree in two sections. One being the left side of the tree ending with the lowpass filter in the first stage and the other being the right side ending with the highpass filter in the first stage as done in figure 4.8.

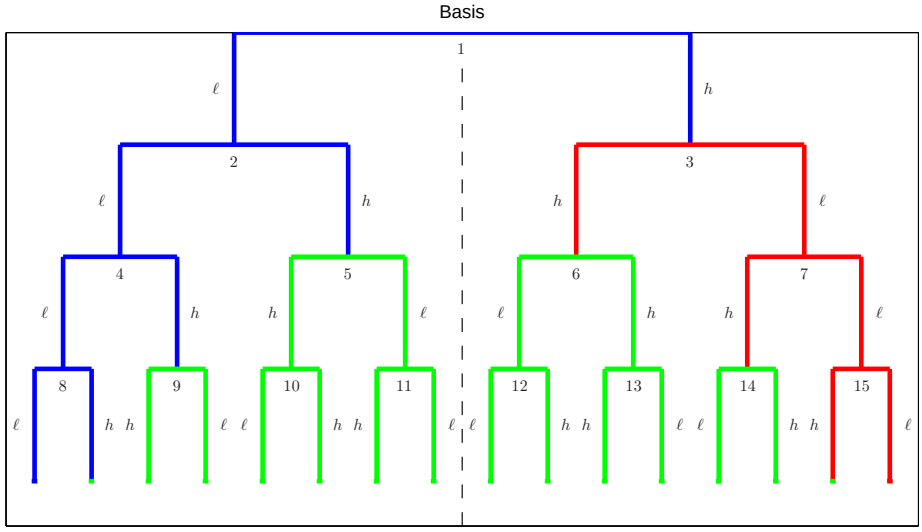


Figure 4.8: Full wavelet packet tree arranged to achieve analytic basis functions. Blue nodes are the same as in the DTCWT. In the red nodes the filters are switched between trees $\mathfrak{Im}$ and $\mathfrak{Re}$. The green nodes have the same filters in both trees.

The left side includes the DTCWT (the blue nodes), which can be exploited when making the complex wavelet packet basis functions analytic. By using the same filters in tree $\mathfrak{Im}$ and $\mathfrak{Re}$ for all the green nodes, the phase difference between the two trees, before getting to one of the DTCWT highpass filters, will be zero. The rest of the way through the inverse complex wavelet packet transform runs as the inverse DTCWT creating the necessary phase difference between tree $\mathfrak{Im}$ and $\mathfrak{Re}$.

In the other half the last filter is a highpass filter, which has a negative phase difference slope (opposite the lowpass filter). That means that the first phase difference slope introduced by a ($\mathfrak{Re}$, $\mathfrak{Im}$) filter set on a path in the inverse complex wavelet packet transform, which ends at the highpass filter, has to be

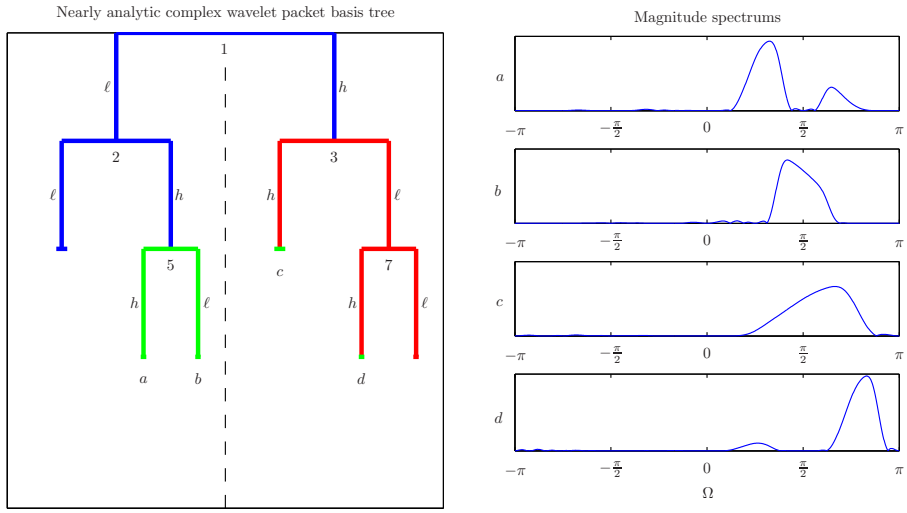
positive. All the ($\mathfrak{Re}$, $\mathfrak{Im}$) filters sets in between have to have negative phase difference slopes.

As can be seen in figure 4.8 the right side is a mirror image of the left side (except for the first highpass filter), therefore it seems to be a good approach to try and mirror the operation in the left side. That means that the filters need to be switched between tree $\mathfrak{Im}$ and $\mathfrak{Re}$ in all the red nodes, in order to make the phase difference slope of the lowpass filters be negative like the first stage highpass filter. Also in all the green nodes, the filters are the same in both trees. That way the phase difference is kept at zero until the highpass filters in the red nodes.

Now the operations on the right side are mirroring the operations on the left side at least until the first stage filters in the end of the inverse complex wavelet packet transform. The mirroring accomplishes that all the magnitude plots in figure 4.3 (illustrating the operation of the blue nodes) will be the same for the operation in the red nodes in figure 4.8, and all the phase difference plots will be mirrored around $\Omega = 0$. That means that the value at the positive frequencies will be switched with the value at the negative frequencies. This could lead one to think, that the right side of the basis tree in figure 4.8 will create basis functions, which are inverse analytic (only consist of negative frequencies instead of positive).

This is not the case, because the last filter on the right side is a highpass filter and not a lowpass filter as on the left side. The result of the highpass filter is that the frequencies around $\omega = \pm\pi$ are preserved instead of the frequencies around $\omega = 0$, so it is the phase difference level in the center of these passband regions which is important. As described in section 4.2.2 the points marked in figure 4.3 will, with each lowpass filter stage, move closer and close to $\Omega = 0$, and their values will approach $\pm\frac{3}{2}\pi$. Similarly the center points in the frequency regions preserved by the highpass filter will move closer and closer to $\Omega = \pm\pi$ and their values will approach $\frac{1}{2}\pi$ for the negative frequencies and $-\frac{1}{2}\pi$ for the positive frequencies. These are the correct values according to equation (4.8), and by examining the phase difference plot in figure 4.4(b), it can be seen that the first stage highpass filter will move the phase difference plots by the last small amount, just as the first stage lowpass filter does it as illustrated in figure 4.5(a).

With this new reordering of the filters between the $\mathfrak{Im}$ and $\mathfrak{Re}$ trees (see again figure 4.8), a nearly analytic dual tree complex wavelet packet transform has been constructed. In figure 4.9 the magnitude spectrums of the same four basis functions as in figure 4.7 are shown, and it is seen how the reordering of the filters correct the basis functions in such a way, that they become nearly analytic.



(a) A wavelet packet basis tree, ℓ marks the lowpass filters and h the highpass filters.

(b) Magnitude spectrums of the basis functions.

Figure 4.9: The corrected wavelet packet basis tree and magnitude spectrums of the corresponding nearly analytic basis functions.

It should be noted that there are still two non-analytic basis functions. The first one is the one coming from the row of only lowpass filters on the left side, and the second one is its mirror function on the right going through only lowpass filters and the first stage highpass filter. These two basis functions are in principle the same as the non-analytic basis functions in the DTCWT, and hence with the DTCWT filters the best possible wavelet packet filter ordering has been created.

4.3.3 Shift Invariance of Complex Wavelet Packet Coefficients

The above described corrections to the complex wavelet packet transform was done to make the basis functions nearly analytic. But the analyticity is of course not a goal in itself, the goal is a more shift invariant transform. Therefore the shift invariance will be investigated further by returning to the time domain, and the time domain representation of the basis functions.

The shift invariance provided by the Fourier transform is complete in the sense, that any shift in an input signal is just encoded as a phase shift in the complex Fourier coefficients; the absolute value will be unchanged. This kind of shift

invariance can only be achieved with infinitely long basis functions, and since the wavelet basis functions are time limited, the shift invariance will be limited too.

A wavelet coefficient can, in the same way as a Fourier coefficient, be described as the inner product between the input signal and a basis function. This can be written as

$$c_{j,k} = \langle x(n), \psi^{\mathbb{C}}_{j,k}(n) \rangle = \sum_n x(n) \left(\psi^{\mathfrak{Re}}_{j,k}(n) + i\psi^{\mathfrak{Im}}_{j,k}(n) \right) \tag{4.11}$$

Remember, when investigating the shift invariance, the absolute value of the coefficient is the interesting part. When the input signal $x(n)$ is an impulse at $n = n_1$ the absolute value of $c_{j,k}$ is

$$|c_{j,k}| = \sqrt{\left(\psi^{\mathfrak{Re}}_{j,k}(n_1) \right)^2 + \left(\psi^{\mathfrak{Im}}_{j,k}(n_1) \right)^2} \tag{4.12}$$

If $|c_{j,k}|$ is shift invariant, the calculation should be independent of n_1, that is

$$\left(\psi^{\mathfrak{Re}}_{j,k}(n) \right)^2 + \left(\psi^{\mathfrak{Im}}_{j,k}(n) \right)^2 = k \tag{4.13}$$

where k is a constant.

This is not the only requirement. If $x(n)$ is exchanged with two impulses at n_1 and n_2, the calculation changes to

$$|c_{j,k}| = \sqrt{\left(\psi^{\mathfrak{Re}}_{j,k}(n_1) \right)^2 + \left(\psi^{\mathfrak{Im}}_{j,k}(n_1) \right)^2 + \left(\psi^{\mathfrak{Re}}_{j,k}(n_2) \right)^2 + \left(\psi^{\mathfrak{Im}}_{j,k}(n_2) \right)^2 + 2\psi^{\mathfrak{Re}}_{j,k}(n_1)\psi^{\mathfrak{Re}}_{j,k}(n_2) + 2\psi^{\mathfrak{Im}}_{j,k}(n_1)\psi^{\mathfrak{Im}}_{j,k}(n_2)} \tag{4.14}$$

From this equation it can be seen that there is another criteria for the shift invariance of $|c_{j,k}|$, which can be written as

$$\psi^{\mathfrak{Re}}_{j,k}(n_1)\psi^{\mathfrak{Re}}_{j,k}(n_2) + \psi^{\mathfrak{Im}}_{j,k}(n_1)\psi^{\mathfrak{Im}}_{j,k}(n_2) = f(n_1 - n_2) \tag{4.15}$$

for some function $f(n)$. The function $f(n)$ is not important, the important thing is that it is only a function of the difference between n_1 and n_2.

The Fourier basis functions fulfill both requirements, because

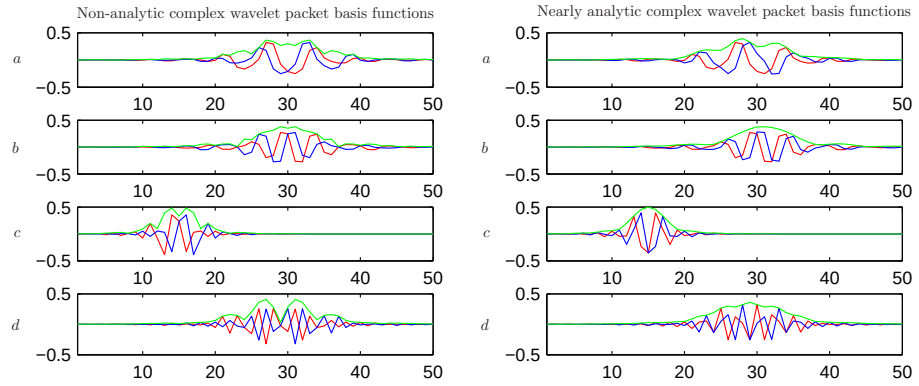
$$\cos^2(n) + \sin^2(n) = 1 \tag{4.16}$$

and

$$\cos(n_1)\cos(n_2) + \sin(n_1)\sin(n_2) = \cos(n_1 - n_2) \tag{4.17}$$

but the complex wavelet packet basis functions only approximate these equations. In figure 4.10 four basis functions of the complex wavelet packet transform, before and after the analyticity reordering of the wavelet packet filters, are plotted. The magnitude spectrums of these basis functions have already been plotted in figure 4.7 and 4.9, and here the time domain representations are given. The green line shows the absolute value of the complex basis functions, and according to equation (4.13) this should be a constant. It can be seen, that this can not be fulfilled by time limited functions, but for longer basis functions the approximation improves. That means that the deeper the complex wavelet packet filter bank is, the more shift invariant the transform will be.

When comparing the nearly analytic basis functions with the non-analytic basis functions, it is seen that the analyticity ensures a more smooth absolute value curve compared to the non-analytic basis functions. This is the first sign of an improved shift invariance. The second requirement in equation (4.15) is harder to illustrate, and it doesn't seem like the analytic basis functions fulfill it better than the non-analytic ones.

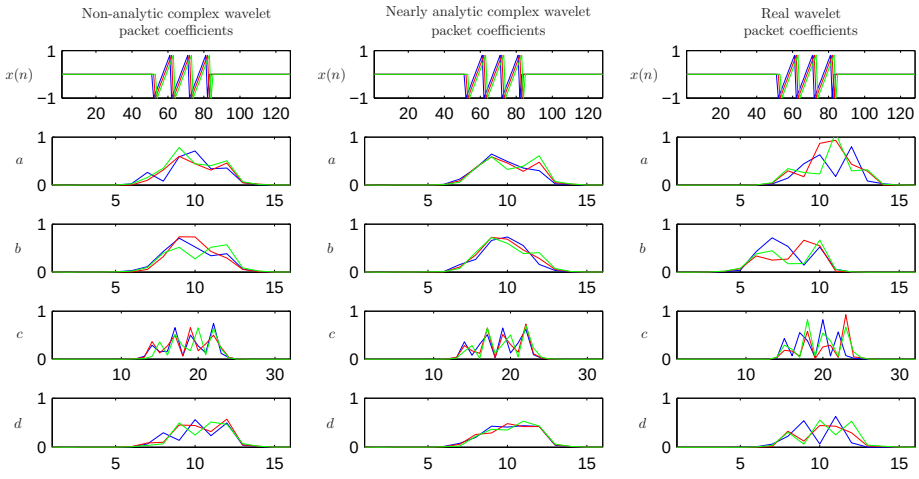


(a) Non-analytic complex wavelet packet. (b) Near analytic complex wavelet packet.

Figure 4.10: Complex wavelet packet and nearly analytic complex wavelet packet basis functions. $\psi^{\mathfrak{Re}}$ is red, $\psi^{\mathfrak{Im}}$ is blue and $|\psi^{\mathbb{C}}|$ is green.

Finally the shift invariance is tested with an example, and the complex-, the nearly analytic complex- and the real wavelet packet coefficients are compared. The wavelet packet basis is still the same as illustrated in figure 4.7(a) and 4.9(a), and the input signal - a sawtooth - is shifted twice by one sample. The results are shown in figure 4.11, where the top plots show the input signal, and the following plots show the wavelet packet coefficients. Here the superior shift invariance of the near analytic complex wavelet transform can be seen, and a big improvement has been achieved especially compared with the real wavelet

transform.



(a) Non-analytic complex wavelet packet.

(b) Nearly analytic complex wavelet packet.

(c) Real wavelet packet.

Figure 4.11: Absolute value of non-analytic complex, nearly analytic complex and real wavelet packet coefficients of shifted version of sawtooth input signal $x(n)$.

The correction from the non-analytic complex- to the nearly analytic complex wavelet packet transform is done only by reordering the wavelet packet filters used in the transform. This reordering can be done before implementing the transform, which means that the improved shift invariance is achieved with no extra computation costs.

Chapter 5

Implementation

In the previous chapters theory and tools have been described and developed, and it is now possible to put it all together to a periodic noise filtering scheme. The complete system was already shortly introduced in section 1.1, and with that as a platform the implementation of the Noise Period Analyzer and the Noise Filter will be explained. In contrary to a real-time implementation with sample by sample processing, the implementations done in this project work on whole signals with finite length. This is normally easier and faster than a real-time implementation, but it creates some differences, which are discussed.

5.1 Implementation of the Noise Period Analyzer and the Noise Filter

5.1.1 The Noise Period Analyzer

The goal of the Noise Period Analyzer is to gather information about the periodic noise in speech pauses, so it can be used to remove the periodic noise, when speech is present. The information consists of wavelet packet coefficients resulting from transforming each period of noise. These coefficients are then

combined to a thresholding packet, which can be used to remove the periodic noise when the speech is present.

The Noise Period Analyzer only works on the periodic noise, when there isn't any speech in the input signal. This information can be given by a speech pause detector, which is also a topic of a lot of research, see for instance [15]. Also the length of the periods is needed, which is another research topic beyond the scope of this project, but simple autocorrelation has been tested and can in some cases be used to get a decent estimate.

In this project the Noise Period Analyzer will be given information about how many noise periods are available, before the speech signal starts, and the length of the periods. The Noise Period Analyzer then takes out the available noise periods and divides them into chunks of one period each. These periods are then wavelet packet transformed one by one, using the PWP transform described in section 3.2. This PWP transform can be implemented with any kind of wavelet system, Daubechies, Symmlets and complex wavelets. The implementation was already available with Symmlet 4 wavelet packets (real implementation), but has in this project been extended to also be able to use the complex wavelet packets, using length 14 Q-shift filters and length 10 first stage filters. Both an implementation using the straight forwardly extended non analytic complex wavelet packets (complex implementation) and the correction to nearly analytic complex wavelet packets (analytic implementation) have been made. The basis tree structure for the wavelet packets is in the current implementation found by the Noise Filter, and therefore unknown to the Noise Period Analyzer. As a result all the coefficients, in a full basis tree down to a specified level, are calculated. This gives a lot of calculations, of which a lot are not going to be used, and this should be avoided in a real-time implementation. Hence the Noise Period analyzer should be informed of which basis tree to use for the wavelet packets.

The wavelet packet coefficients of each period of noise can be combined into a thresholding packet using two approaches - average and max - presented in section 3.2.3. All the above mentioned implementations of the PWP transform can use both thresholding functions. Finally the solution to the problems with the edge coefficients described in 3.2.4 can also be applied to the different implementations.

5.1.2 The Noise Filter

The Noise Filter is where the periodic noise is removed from the speech signal. It gets the information (the thresholding packet) from the Noise Period

Analyzer and assumes that the noise periods, when the speech is present, are well described by the thresholding packet coefficients. In the Noise Filter the coefficients are used as individual thresholding values for the noisy speech signal period by period, and the thresholded signal is the final output signal.

Only the part of the input signal, where speech is present, is processed by the Noise Filter, so the Noise Period Analyzer passes that part of the input signal on to the Noise Filter. This has been done, because only that part of the signal is interesting, when evaluating the Noise Filter, but in a real-time implementation, the Noise Filter would be continuously running removing also the noise in speech pauses. The noisy speech signal is wavelet packet transformed in a full wavelet packet basis tree, and the best basis is then found from these coefficients as described in section 2.2.1. This gives a very good basis choice for the given signal, but it is not feasible to do the same in a real-time implementation. Here the basis tree has to be chosen, before the signal is available, and how to do that is a problem,which would need to be addressed. The type of wavelet packets used has to be the same as in the Noise Period Analyzer, and implementations have been done using the same wavelet systems as described above.

After finding the best basis, the wavelet packet coefficients are thresholded using the thresholding packet coefficients. This is done by periodically extending the thresholding packet until there are as many thresholding packet coefficients as wavelet packet coefficients of the noisy speech signal. In the implementation where the edge effects are corrected, the edges of the extended thresholding packet are exchanged with the specifically calculated edge coefficients. Then all the noisy speech wavelet coefficients are thresholded using the individual thresholding values in the extended thresholding packet, which is done with the Garrote thresholding function described in section 2.2.2.5. In the case of complex coefficients the thresholding is done in a little more advanced way. The absolute value or length of both the complex signal coefficients and the complex thresholding coefficients is used in the same way as the real coefficients. The thresholded complex signal coefficients are then just shortened, while keeping the same vectorial direction. This can be done as

$$c_t = \left(|c| - \frac{|t|}{|c|} \right) \left[\cos\left(\angle c\right) + j \sin\left(\angle c\right) \right] \tag{5.1}$$

where c_t is the thresholded complex coefficient, c is the complex signal coefficient, t is the complex thresholding coefficient, and $\angle c = \tan^{-1}\left(\frac{\mathfrak{Im}\{c\}}{\mathfrak{Re}\{c\}}\right)$, where $\mathfrak{Re}$ and $\mathfrak{Im}$ give the real and the imaginary part respectively.

The thresholded complex coefficients are then inverse wavelet packet transformed, and the clean output signal is thereby obtained.

With the possibility of using different wavelet packet systems (real, complex

and nearly analytic complex), along with the two different thresholding packets (average and max) and the extra calculation of the edge coefficients, many different periodic noise filtering setups can be made. This will be used in chapter 6, where the performance of the filtering scheme is evaluated, to give an overview of the influence of the different improvements.

5.2 A Spectral Subtraction Scheme

In order to have something to compare the results of the periodic noise filtering scheme with, another method should be used to do the same filtering tasks. The method chosen is a spectral subtraction scheme, because it is relatively simple and works in a similar way as the periodic noise filtering scheme developed here. That is it uses sequences, where only noise is present, to analyze the noise by estimating the spectrum using STFT. A single sequence is split up into several smaller overlapping segments, and each segment is windowed, and thereafter the Fast Fourier Transform (FFT) is calculated. All the FFTs are then averaged to give the estimate of the noise spectrum. When the speech is present, it is also split into overlapping segments, windowed and FFTed. Then the noise spectrum estimate is subtracted, and the inverse FFT is calculated of all the segments. Finally the segments are combined to give the cleaned signal, by adding the segments where they overlap. A well working implementation of this, using non linear magnitude spectral subtraction including the mathematical theory behind it, can be found in [16], and that implementation has been used in this project.

5.3 Matlab Implementation

All the implementations in this project have been done in Matlab, and a special free wavelet toolbox developed at Stanford called WAVELAB850 has been used [17]. The WAVELAB toolbox includes a lot of m-files of which the ones used in this project were m-files to calculate real wavelet transforms, real wavelet packet transforms and best basis algorithms. Further some smaller functions were used, especially functions to plot basis trees and time-frequency planes. The PWP transform in a real version was also already programmed, before the start of this project [6].

All the functions needed to calculate the complex wavelet and complex wavelet packet transforms have been programmed during this project. This has been done in order to be able to control the shift of the circular convolution, and get a full insight in the complex wavelet packets. Only the best basis algorithm of

the WAVELAB850 toolbox is used; other than that the developed m-files work in Matlab without the need of other tools or functions.

With the Matlab implementation of the periodic noise filtering scheme and of the spectral subtraction scheme, filtering tests can easily be constructed, which is the topic of the next chapter.

Chapter 6

Evaluation

In the previous chapters the periodic noise filtering scheme has been described, and the tools it uses have been developed. It is now interesting to test the performance of the scheme, when used to remove/suppress periodic noise.

Here two different approaches will be taken to evaluate the performance of the periodic noise filtering scheme. The first method is mathematical and will use the signal to noise ratio (SNR) of the filtered test signals to evaluate and compare different filtering results. The second method is subjective and will consist of a test, where people listen to the filtered signals and evaluate their sound quality.

6.1 Evaluating the Periodic Noise Filtering Scheme Using SNR's

A commonly used measure of the quality of speech signals is the signal to noise ratio (SNR). The SNR is usually given in dB and is calculated as

$$\mathrm{SNR} = 10 \log_{10} \left\{ \frac{\sum_{n=1}^{N} (\hat{x}(n))^2}{\sum_{n=1}^{N} (\hat{x}(n) - x(n))^2} \right\} \tag{6.1}$$

where $x(n)$ is the clean speech signal, $\hat{x}(n)$ is the filtered signal, and N is the length of the speech signal.

In addition to testing the general performance, the influence of the following improvements will be evaluated

a. The max thresholding packet and the edge effects

b. The complex wavelets

c. The nearly analytic complex wavelets

The filtering scheme furthermore includes some parameters, which can be varied, and these can also influence the performance. The parameters are described here:

$N_{analysis}$
In each test a noisy speech signal (test signal) is created by overlapping a clear speech signal with periodic noise. The start of the test signal will consist of only periodic noise, which can be used to obtain the thresholding packet. The number of noise periods without speech can be varied, and the number is given by the parameter $N_{analysis}$. The influence of this parameter will also be evaluated.

thscale
Another important element is the scaling of the thresholding packet coefficients. As stated in section 3.2.3, it might be beneficial to scale the thresholding packet coefficients by some amount, given by the parameter thscale, in order to achieve a better SNR. thscale is a multiplicative constant, which all the thresholding packet coefficients are multiplied with, before they are used for thresholding. A thscale value of one means no scaling of the coefficients. This parameters influence will be investigated too.

λ
Finally the parameter λ, which is the forgetting factor in the average and the max thresholding packet, can be varied. This only serves to test the performance, when many noise periods are used for obtaining the thresholding packet, and the noise is periodically nonstationary. This parameter will therefore be very dependent on the specific periodic noise, and since only periodically stationary noise signals will be considered here, λ will be set to 1 in all tests.

There are further parameters related to the wavelet packet transform. These parameters will not be varied through the tests, only the different types of wavelets (Real, Complex, Nearly analytic complex) will be tested. The other

parameters are the depth of the transforms, which will be set to 8 filtering stages in all tests. The specific wavelets used in the different setups are for the real Symmlet 4, and for the complex schemes the length 14-qshift wavelets, with the length 10 first stage filters. The wavelet packet basis will be found individually in each test using the best basis algorithm described in section 2.2.1, and the specific noisy input signal used in the test. The thresholding function used is the Garrote described in section 2.2.2.5.

Three different periodic noise signals are used in creating the test signals for the filtering scheme. The three noise signals are

Chirp - Repeated chirps with some periodic variations
Asma - a sequence of engine noise
Alfa - a different sequence of engine noise

The **Chirp** signal consists of periods with the length of $N_T = 6202$ samples, and in each period half the samples are a chirp, and the remaining samples are zeros. The chirps are placed in the middle of each period and then moved by a random number taken from a Gaussian distribution with zero mean and variance $0.05N_T$. In that way the Chirp noise signal is not perfectly periodic, because all the periods are shifted versions of each other, but it is definitely periodically stationary.

About five periods of the chirp noise are plotted in time-frequency planes in figure 6.1. The same basis tree is used for both the real and the nearly analytic complex wavelet packets, and the benefit of the complex wavelet packets is well illustrated by the plots. First the energy of the chirps is much better represented by the nearly analytic complex wavelet packets, and second the improve in shift invariance makes the chirps look almost identical. The time shifts of the chirps are still seen though. The time between the 2nd and the 3rd chirp is smaller than the time between the 3rd and the 4th. This will cause problems when trying to remove the chirps, since the chirps gathered in the thresholding packet might be located at different times within a period than the chirps corrupting the speech signal. The nearly analytic complex wavelet packets will therefore need several noise periods for obtaining the thresholding packet in order to remove chirps with different shifts.

The **Asma** signal has periods of $N_T = 2731$ samples and is approximately periodically stationary. The energy of the noise is widely spread out in both time and frequency as can be seen in the top plot in figure 6.2.

Finally the **Alfa** signal, shown in the bottom plot in figure 6.2 is periodic with period lengths of $N_T = 888$ samples. It is concentrated at low frequencies, and its total energy is lower than the energy of the Asma noise.

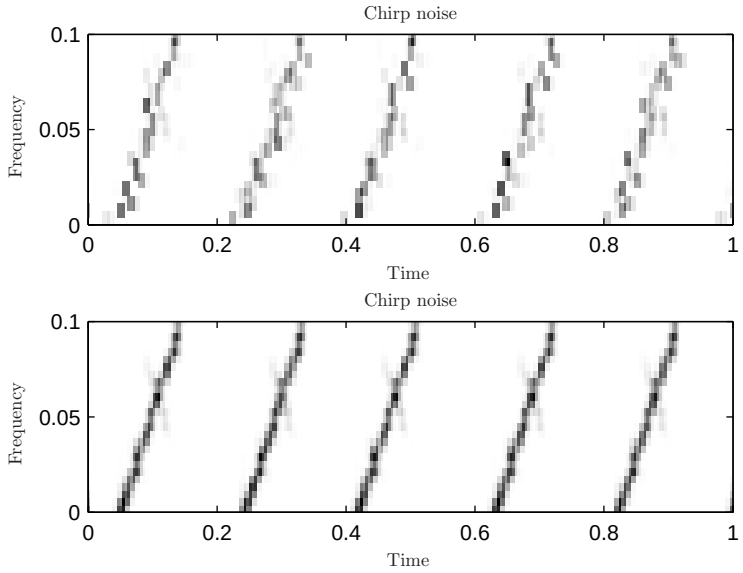


Figure 6.1: Real (top plot) and nearly analytic complex (bottom plot) wavelet packet representation of chirp noise in time-frequency planes.

Along with the periodic noise signals four different speech signals are used. The speech signals are denoted by t1, t2, t3 and t4 and are a male voice, a female voice and two other different male voices respectively. The speech sequences are relatively short - between 1 and 3 seconds - and are also used in the listening test, which will be presented in section 6.2.

The speech signals overlapped with the periodic noise signals make up 12 different test signals for the evaluation tests described in the following sections. In each of the evaluation tests, the exact same test signals will be filtered using different methods and/or with different parameters, hence the initial SNR will be the same for the different methods. The evaluation test will calculate the SNR after filtering, which can then be directly compared.

6.1.1 Comparing the Different Improvements

In the first test the different improvements given by a, b and c above will be compared. Further a spectral subtraction scheme is included to compare the periodic filtering method with another type of filtering approach. That gives

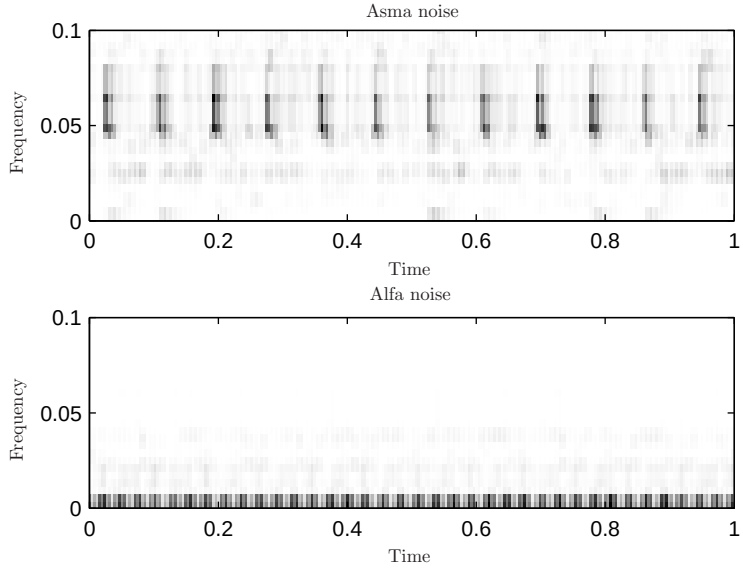


Figure 6.2: The top plot shows a nearly analytic complex wavelet packet representation of the Asma noise. The bottom plot shows the Alfa noise signal.

the following different setups

SpecSub	The spectral subtractions scheme.
Real	The filtering scheme using real wavelets (Symmlet 4) and the average thresholding packet
Complex	The filtering scheme using complex wavelets and the average thresholding packet
Analytic	The filtering scheme using the nearly analytic complex wavelets and the average thresholding packet
Analytic_Max_Edge	The filtering scheme using the nearly analytic complex wavelets, the max thresholding packet, and correcting the edge effects.

The max thresholding packet and the edge effects tested together using the

Analytic_Max_Edge setup. This was done, because it turned out during the following experiments, that the correction of the edge effects didn't have a very big influence. This can come from the fact, that the wavelet packet filter bank depth is set to be only 8, which means that the percentage of edge coefficients is not very high. This was - because of lack of time - not investigated further though.

6.1.1.1 Testing With a thscale Value of One

The test is done with $N_{analysis} = 10$ noise periods used to obtain the thresholding packet; for the spectral subtraction scheme, these periods are used to estimate the spectrum of the noise. Also the thresholding coefficients will not be scaled (thscale=1), and finally $\lambda = 1$. The test evaluates the SNR ratio after filtering, and 12 test signals are created using the three different noise signals and the four different speech signals.

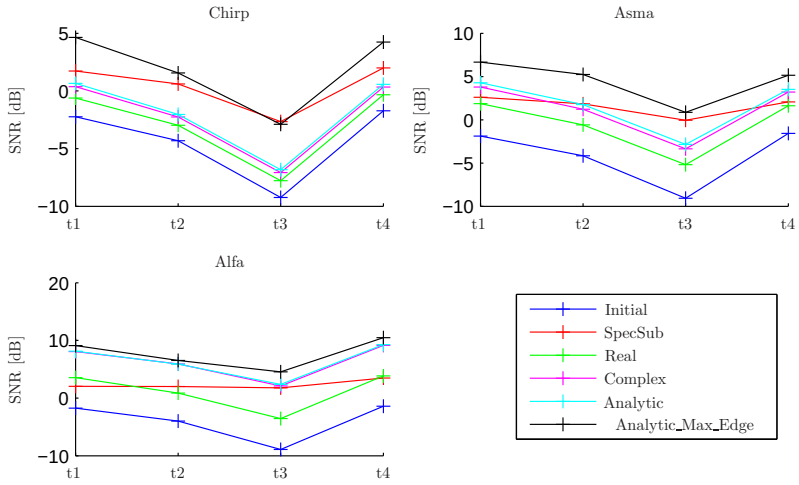


Figure 6.3: SNR results of filtering the four different speech signals (on the x-axes) corrupted by the three kinds of periodic noise, thscale=1. Chirp top left, Asma top right and Alfa bottom left.

In figure 6.3 the SNRs for the different setups are plotted. Each plot shows the results for one type of noise, and the four different speech signals are given along the x-axes. The plots show that the **Analytic_Max_Edge** generally has the best performance, which comes from the use of the max thresholding

packet. The **SpecSub** does well on the chirp signal, because there the periods are shifted versions of each other, and it is the only setup, which is fully time shift invariant. For the Asma noise signal, it depends on the specific speech signal, which of the setups - **Analytic**, **Complex** or **SpecSub** - have the best performance. But clearly for the Alfa noise the **Analytic** and the **Complex** are the superior methods. The improvements using the nearly analytic complex wavelet packets in comparison to the non analytic complex wavelet packets are shown for the chirp and the Asma noises.

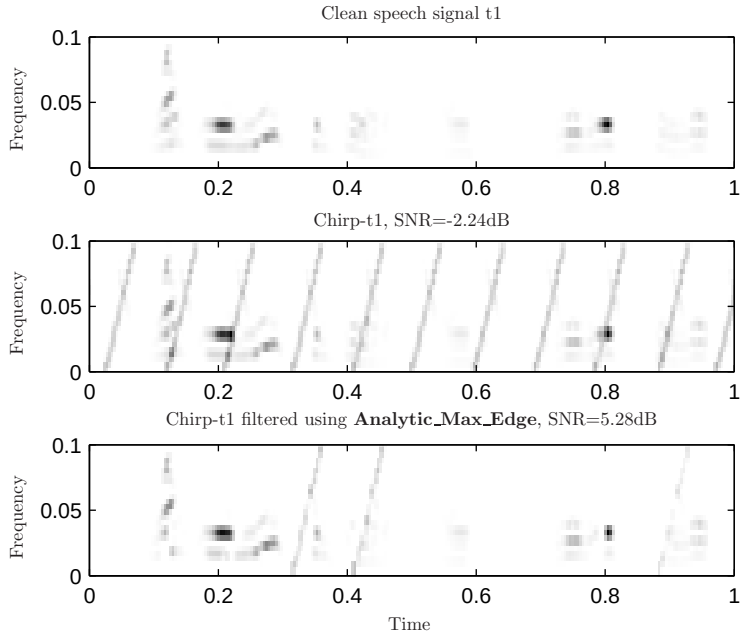


Figure 6.4: Time-frequency planes illustrating the filtering of the chirp-t1 test signal using the **Analytic_Max_Edge** setup.

The filtering of the chirp-t1 test signal using the **Analytic_Max_Edge** setup is illustrated in figure 6.4 using time-frequency planes. The plot in the top of the figure shows the clean t1 speech signal, while the plot in the middle of the figure shows the speech signal corrupted by the chirp noise, and the bottom plot shows the signal after the filtering was performed. It can be seen, that only elements from three out of 10 noise chirps are still left in the signal, while most of the speech is preserved. This visualizes the filtering achievements, which can be obtained using the **Analytic_Max_Edge** setup, and how the SNR is improved

from an initial value of -2.24dB to an SNR after filtering of 5.28dB.

6.1.1.2 Testing With Individual thscale Values

The performance of the different setups can be improved by letting the thscale value be different than one. Especially the setups using the average thresholding packet require a thscale value bigger than one to give good performance. In the following tests the thscale value, which gives the highest SNR is found for each setup, using a simple search algorithm. Finding the thscale value is easy, when the filtering is not done in a real time setup. Then the filtering can simply be done using different thscale values, and the aforementioned simple search algorithm can be used to speed up the search for the value giving the best SNR. When the filtering is done in real time, finding a good thscale value can be a really challenging task. This is not considered further here, but should be investigated for a real-time implementation.

In this test the thscale value has been limited to the interval between 0 and 8, which has been done in order to avoid that it increases to very high values removing both the noise and the signal. This can happen since the noise energies are high compared to the speech signal energies (initial SNRs less than 0), and hence removing both the signal and the noise will result in SNRs of 0, which is an improvement compared with the initial SNR. Because of the search algorithm used the maximum thscale value was 7.94.

As above the SNR for the three different noise signals and the four different speech signals are plotted in figure 6.5. It can there be seen how the performance of all the periodic filtering setups improve, and all of them are now equal to or better than the **SpecSub**, which is not changed and has the same SNR values as in figure 6.3. It is interesting that the **Analytic_Max_Edge** now doesn't have a better performance than the **Analytic** and **Complex** setups. But since it generally uses smaller thscale values, it makes it easier to estimate a good thscale value especially important in a real time application.

The tests show that the nearly analytic complex wavelet packets have successfully improved the periodic noise filtering scheme in comparison with the real wavelet packets. Also when the average thresholding packet is used, the setups depend heavily on the thscale value, but with the max thresholding packet the thscale value given the highest SNR will in most cases be close to 1. It is therefore not very important to find a good thscale value, because good results are already achieved, when it is kept at one.

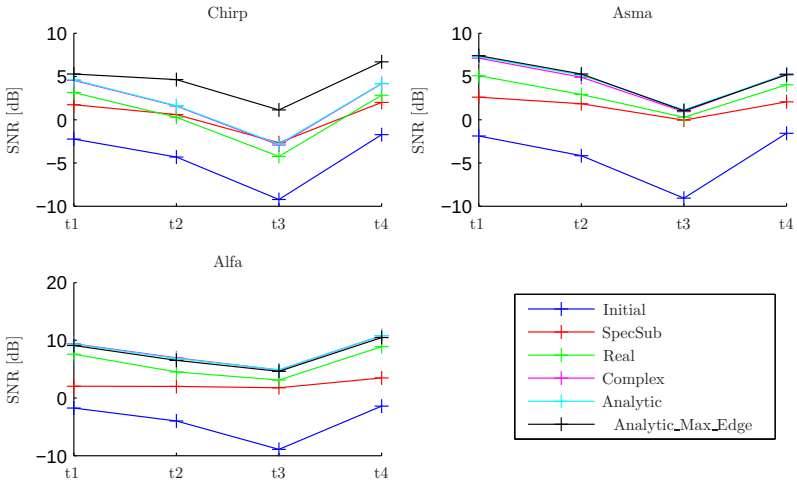


Figure 6.5: SNR results of filtering the four different speech signals (on the x-axes) corrupted by the three kinds of periodic noise, thscale is set individually for each setup to achieve maximal SNR. Chirp top left, Asma top right and Alfa bottom left.

6.1.2 Investigating the Results of Changing the N_analysis Parameter

The above tests were all made with N_analysis=10 noise periods used to obtain the thresholding packet. But when the periodic noise filtering scheme is implemented in a complete system to remove periodic noise, the number of noise periods available for obtaining the thresholding packet might vary a lot. For instance before speech starts there might be a lot of available noise periods, but if the noise is changing, it might be desirable to update the thresholding packet or even completely renew it in speech pauses. It is therefore very relevant to investigate, what influence a change in N_analysis will have.

In the following tests the **SpecSub**, the **Real**, the **Analytic** and the **Analytic_Max_Edge** setups are compared. The tests are both done for thscale=1, and thscale values set individually for the different setups.

With the 12 different test signals and different number of N_analysis for each, there are a lot of test combinations. All of them have been tested, but the results, when using a specific noise signal, and different speech signals, look alike, and therefore only the results using the t2 signal corrupted by periodic

noise will be presented here. The t2 signal has been chosen, because it gives an initial SNR, which lies between the other test signals, see figure 6.3 and 6.5.

For the chirp noise the SNR results, when filtering the chirp-t2 signal with both thscale=1 and individual thscale values, are shown in figure 6.6. First considering the test where thscale=1, shown in the left plot, it is interesting that only the **Analytic_Max_Edge** setup really improves with increasing N_analysis. This further illustrates that with the max thresholding packet, it is much less important to be able to find a good thscale value. Also the **Analytic** setup with the nearly analytic complex wavelet packets gives an almost constant improvement in comparison with the **Real** setup, which was also seen in the previous tests. When the thscale is set individually as done in the right plot, the **Analytic** and **Real** setups start to improve with increasing $N_{analysis}$ values, with the **Analytic** setup still being better by an almost constant amount. The **SpecSub** achieves good SNRs compared with the other schemes for low $N_{analysis}$ values, but when the $N_{analysis}$ is increased, the **SpecSub** doesn't improve and the **Analytic_Max_Edge** then achieves the best SNR.

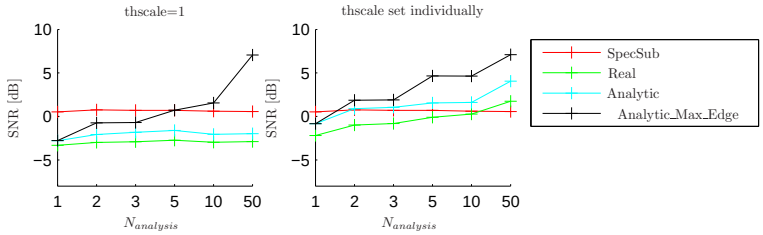


Figure 6.6: SNR results of filtering the t2 speech signal corrupted by the chirp noise with different values of $N_{analysis}$. In the left figure thscale=1, and in the right figure thscale is set individually for each setup to achieve maximal SNR.

The SNR results, when filtering the asma-t2 signal, are shown in figure 6.7. The results are very similar to the results obtained when filtering the chirp-t2 signal, though the periodic filtering schemes generally achieve better SNR values, when compared to the **SpecSub** setup.

The last periodic noise signal is the alfa noise. This noise has the lowest energy, which can be seen on the SNR levels in figure 6.8. The interesting about the results here is that the average and the max thresholding packets seem to perform equally good, (the **Analytic_Max_Edge** and the **Analytic** setups). Also these schemes don't improve with increasing $N_{analysis}$, which indicates that the alfa noise periods are very alike in the nearly analytic complex wavelet representation. Another reason is of course that the SNR values are high, and that makes further improvements difficult.

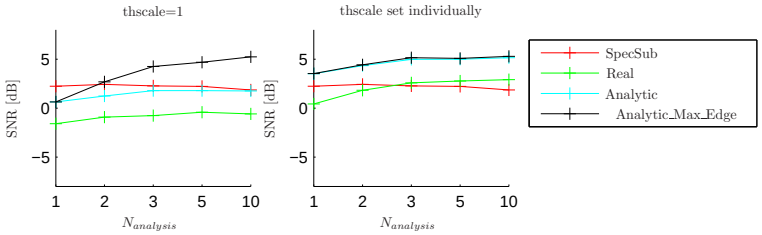


Figure 6.7: SNR results of filtering the t2 speech signal corrupted by the asma noise with different values of $N_{analysis}$. In the left figure thscale=1, and in the right figure thscale is set individually for each setup to achieve maximal SNR.

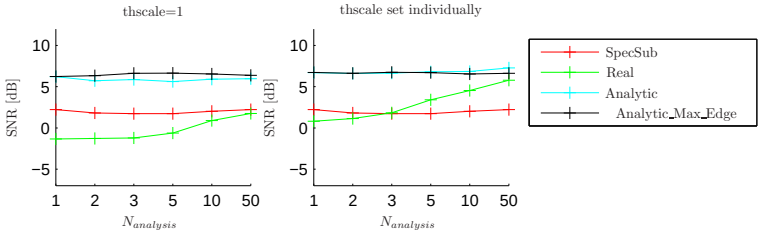


Figure 6.8: SNR results of filtering the t2 speech signal corrupted by the alfa noise with different values of $N_{analysis}$. In the left figure thscale=1, and in the right figure thscale is set individually for each setup to achieve maximal SNR.

The SNR tests have shown that the **Analytic_Max_Edge** scheme achieves the best results. It gives the highest SNRs in nearly all the tests, and is clearly outperforming the **SpecSub** scheme.

6.2 Evaluation Using Listening Test

The SNR is a standard mathematical way of evaluating the quality of a speech signal, but it doesn't always reflect how the sound is perceived by the human ear. Therefore when evaluating speech signals it is very relevant to also do a subjective listening test. For that purpose a website was created, where test persons could listen to test signals and give subjective feedback. On the website the following information was given about the test:

> *The test consists of 10 different blocks. The first 7 blocks are preference tests, where two signals are compared, and the test person is asked to choose the one he/she prefers or no preference. The last*

3 blocks are point or score tests, where 7 signals are compared; one is a clean speech signal and the others are noisy and filtered signals. The test person gives points from 1 to 10 to all the signals, where 10 is the highest score and should be given to the clean signal. The preference tests have been arranged in random order, as have the point tests. Also the sound signals in all the tests have been placed randomly.

Further the following instructions were given on the website, regarding how to complete the test.

The test consists of seven preference tests and three point or score tests. The preference tests consist of two sound signals A and B, and the options of preference for A, B or no preference. The point or score tests consists of seven signals, and each signal should be given points between 1 and 10, where 10 is the best score. In the point test one of the seven signals will be clean and should be given 10 points, and this then serves as a reference when grading the other six signals.

When listening and comparing the sounds they should be evaluated according to first how easy it is to understand the spoken words, and secondly how your personal impression of the sound is. Please take the test from the top down, and feel free to listen to the sounds as many times as needed.

Unfortunately the website came up really late, and when it finally came up, there were some problems with the server, it was installed on. Therefore the number of test persons who took the test is very limited.

6.2.1 The Preference Tests

The test signals used in the 7 preference tests are listed in table 6.1, where also the accumulated results of the different tests are given. Only the scores of 19 test persons were recorded, after a few were removed, because they didn't give answers to all the questions. Even though the data set is small, a statistical treatment of the data can still be carried out. For the preference tests it is desired to show that one sound for instance $A1$ is preferred over sound $B1$. That can be done by showing that the chance that a test person prefers sound $A1$, is greater than 50%. To show this a hypothesis test is set up, in which the null hypothesis is the opposite of what needs to be shown, namely that

Test	Test signal	$N_{analysis}$	thscale	Max_Edge	Score	α
	chirp-t1					
1	Noisy				8	
	Analytic	50	1	+	10	
	No preference				1	
	alfa-t3					
2	Clean				17	
	Analytic	10	1	+	1	
	No preference				1	
	alfa-t1					
3	SpecSub	10			3	
	Real	10	Opt.		14	0.0268
	No preference				2	
	alfa-t3					
4	SpecSub	10			3	
	Analytic	10	1	+	12	0.1332
	No preference				4	
	alfa-t4					
5	Real	10	1		2	
	Analytic	10	1		3	$-t$
	No preference				14	
	asma-t4					
6	Analytic	10	Opt.	+	6	$-t$
	Real	10	Opt.		2	
	No preference				11	
	chirp-t2					
7	SpecSub	10			0	
	Analytic	10	Opt.		15	0.0106
	No preference				4	

Table 6.1: An overview of the test signals, and the results of the 7 preference tests.

the chance a test person will prefer sound $B1$ or have no preference is greater or equal to 50%. If it can be shown that the null hypothesis is wrong with a significance level of 95%, the alternative hypothesis that sound $A1$ is prefered with a chance of more than 50% is assumed [18].

The hypotheses (one for each preference tests) are evaluated using a one sample t-test. The test statistic is

$$t = \frac{p_0 - p}{\sqrt{\frac{p(1-p_0)}{n}}} \tag{6.2}$$

where p is the proportion of the test persons choosing $B1$ or no preference,

p_0 is 0.50 and $n = 19$ is the total number of test persons. A t-distribution with $n - 1 = 18$ degrees of freedom is then used to give the probability α of the calculated t value. If the resulting probability is smaller than 0.05 the null hypothesis is rejected, and it is shown that sound $A1$ is preferred with a chance of more than 50% at a significance level of 95%.

The first two preference tests were only intended to introduce the listener to the listening test, and give the listener an idea of what types of sound he/she would encounter. These are therefore not so interesting to make hypothesis tests on, but the α value calculated for all the other tests are given in the last column in table 6.1. The α value is placed in the row of the sound signal in the alternative hypothesis of each test.

It can be seen that the null hypothesis can be rejected in test 3 and 7, meaning that the **Real** and **Analytic** setup are preferred over the **SpecSub**. In test 4 the percentage α isn't high enough to reject the null hypothesis, even though a large percentage of the test persons preferred the **Analytic** setup. In test 5 and 6 the value of $-t$ indicates a negative test statistics, which also means that the null hypothesis can not be rejected. The negative values comes from the fact that many of the test persons didn't prefer one of the sounds over the other, giving a lot of no preference answers.

6.2.2 The Point Tests

The results from the 3 point tests were also collected, and the accumulated points along with the test signals are given in table 6.2. All three tests included a clean signal, a noisy signal and five filtered signals using different setups or different $N_{analysis}$ values. From the accumulated scores it can be seen, that the test persons were able to pick out the clean signal, but had a really hard time distinguishing the other signals. In test 8 the improvement in filtering performance, when $N_{analysis}$ is increased, was tested. The results point in the direction of increased performance with increasing $N_{analysis}$, but are not significant enough to make solid conclusions. In test 9 and 10 the different filtering setups were compared, and the **Analytic_Max_Edge** gets the most points in both tests (after the clean signals). The accumulated points for the different setups are not very far from each other though, and it appears as if the tests were too difficult for the test persons. It is therefore hard to draw any significant conclusions about the relative performance differences between the different setups without further tests.

Test	Test signal	$N_{analysis}$	thscale	Max_Edge	Score
	alfa-t2				
	Analytic	50	1	+	120
	Noisy				79
8	Analytic	3	1	+	93
	Analytic	10	1	+	100
	Clean				174
	Analytic	1	1	+	95
	Analytic	5	1	+	99
	chirp-t1				
	Noisy				76
	Clean				170
9	Real	10	Opt.		58
	Complex	10	Opt.		72
	SpecSub	10	Opt.		60
	Analytic	10	Opt.	+	83
	Analytic	10	Opt.		83
	asma-t2				
	Complex	10	1		40
	Analytic	10	1		44
10	Real	10	1		41
	SpecSub	10	1		36
	Analytic	10	1	+	55
	Clean				181
	Noisy				52

Table 6.2: An overview of the test signals, and the results of the 3 point tests.

Chapter 7

Conclusion

7.1 The Achievements

In this thesis a periodic noise filtering scheme was presented. The introduced filtering scheme consists of four components of which the two central ones, the Noise Period Analyzer and the Noise Filter were described and implemented. A non-complex wavelet packet version of the scheme, using what is called the Period Wavelet Packet transform, was already presented in [6]. In section 3.2 of this thesis a few problems with this transform were discovered, and improvements were made. This gave rise to a better performance, and especially the development of the max thresholding packet improved the results in the tests. Another noteworthy correction made in section 3.2 was the change of the filter convolution from periodic extension to circular extension, and the calculation of the edge coefficients, to which that lead.

In chapter 4 the lack of shift invariance in the real wavelet transform was identified as another place for improvements. The choice to exchange the real wavelet packets with complex wavelet packets was made, and the starting point was the Dual-Tree Complex Wavelet Packet Transform. The extension of this transform to complex wavelet packets was found to give non-analytic complex wavelet basis functions, when done straight forwardly just like the extension from real wavelets to real wavelet packets. This non-analyticity is undesirable, since it

makes the complex wavelet packet transform less shift invariant compared to a transform with analytic basis functions. The problem with the straight forward extension was discovered and solved by a reordering of the complex wavelet packet filters. This reordering described in section 4.3.2 is one of the most interesting results of this thesis. The reordering gives nearly analytic complex wavelet basis functions, which result in a more shift invariant transform.

The periodic noise filtering scheme was tested in chapter 6. Here the scheme was tested with both real, complex and the nearly analytic complex wavelet packets, and also an average thresholding packet and a max thresholding packet were tested. The SNR results, using the different types of wavelet packets and thresholding packets, and using a spectral subtraction scheme, were evaluated and compared. The conclusion was that the nearly analytic complex wavelet packets using the max thresholding packet gave the best SNRs in the periodic noise filtering scheme, and was also evidently better than the spectral subtraction scheme.

A listening test was created, that had test persons subjectively judge the sound quality of the filtered signals. Some test signals were picked out, and the listeners were asked to choose the sound they preferred according to how understandable the spoken words were, and secondly from the personal impression of the sound. The results of the listening test were not as clear as the ones obtained by calculating and comparing the SNRs. This can partially be explained by the relatively few test persons taking the test (because of time and server problems), but also because the specific sound signals in the test weren't well enough selected. The listening test tried to compare too many different improvements, which lead to unclear results and only a few distinct conclusions.

7.2 Outlook

The not fully successful listening test is a good place to start, when considering the future work which could be done in the domain of this periodic noise filtering scheme. A similar test should be constructed, but different test signals should be chosen, a bigger group of test persons should be used, and only the nearly analytic complex wavelet packet setup with the max thresholding packet and the spectral subtraction scheme should be compared. That is the important comparison, which can fully prove, that the periodic noise filtering scheme is also superior to the ears of listeners.

There are of course also other elements of the periodic noise filtering scheme, which should be tested. Especially the effect of a poorly estimated noise period,

non periodically stationary noise, and the implications of setting the forgetting factor λ to values less than one, when obtaining the thresholding packet, need to be examined. Further the depth of the wavelet packet filter bank could be increased, and the importance or lack hereof correcting the edge coefficients in the thresholding packet could be investigated. And of course more thorough tests using other noise and other speech signals should be performed.

Another area, which should be probed, is the choice of basis tree for the wavelet packet transform. The basis tree, which is currently used, is found using the best basis algorithm working on the noisy speech signal. This algorithm tries to find a basis tree giving large wavelet packet coefficients when transforming the input signal, which means both large speech signal coefficients and large noise coefficients. This might not be the optimal basis for the filtering scheme presented here, and it would be interesting to investigate other possibilities. Also in a real time implementation one would not have the input signal before selecting the basis tree, and therefore one would probably need to find a generalized way of classifying the speech signals expected by the filtering scheme, and from that derive how to choose the basis tree. A learning algorithm could also be developed, in which the filtering scheme tries to learn from its basis tree choices, and that way determines what is a good basis tree.

Finally, the main goal of the periodic noise filtering scheme is, that it should be implemented in a real time application. This requires a speech pause detector and a period length estimator, which are also needed in the scheme. A lot of work is already being done on developing good speech pause detectors, but this problem should of course be addressed in further research papers, as well as the development of a period length estimator. The Periodic Noise Analyzer and the Noise Filter are both based on the nearly analytic complex wavelet packet transform, which is relatively straight forward to implement in real time. The Periodic Noise Analyzer requires a rather large amount of computations, because the input sequence is not down-sampled at each level in the filter bank; but the computations can easily be parallelized, so one can trade size for speed. Additionally both components can work on a sample by sample basis, which keeps the processing delay at a very low level. All these factors make the implementation in a real time application, like a cell phone or a headset, realistic, and a possibility for the future.

Appendix A

Mathematical Derivation of Wavelet Transform Equations

A scaling function at level j is included in the space spanned by the scaling functions at level $j+1$, and therefore it can be written as a linear combination of the level $j+1$ scaling functions [2]. Starting with the scaling function for which $k=0$

$$\varphi_{j,0}(t) = \sum_n g_0(n)\varphi_{j+1,n}(t) = \sum_n g_0(n)\sqrt{2}\varphi_{j,n}(2t) \tag{A.1}$$

or

$$\varphi(t) = \sum_n g_0(n)\sqrt{2}\varphi(2t-n) \tag{A.2}$$

For a shifted scaling function ($k \neq 0$)

$$\begin{aligned}\varphi_{j,k}(t) = 2^{j/2}\varphi(2^j t-k) &= 2^{j/2}\sum_n g_0(n)\sqrt{2}\varphi(2(2^j t-k)-n) \\ &= \sum_n g_0(n)2^{(j+1)/2}\varphi(2^{(j+1)}t-2k-n)\end{aligned} \tag{A.3}$$

and making a change of variable $m = 2k+n$

$$\varphi_{j,k}(t) = \sum_m g_0(m-2k)\varphi_{j+1,m}(t) \tag{A.4}$$

A.1 The Forward Calculation

The coefficients c and d are found by projecting the function $f(t)$ on the scaling and wavelet functions; this corresponds to taking the inner product

$$c_j(k) = \langle f(t), \varphi_{j,k}(t) \rangle \,, \qquad d_j(k) = \langle f(t), \psi_{j,k}(t) \rangle \tag{A.5}$$

For continuous time functions the inner product is an integral, and if we further use the recursive relation obtained in equation (A.4), it is possible to obtain a recursive relation between scaling function coefficients at different levels.

$$\begin{aligned} c_j(k) = \int_{t=-\infty}^{\infty} f(t)\varphi_{j,k}(t)dt &= \int_{t=-\infty}^{\infty} f(t) \sum_m g_0(m-2k)\varphi_{j+1,m}(t)dt \\ &= \sum_m g_0(m-2k) \int_{t=-\infty}^{\infty} f(t)\varphi_{j+1,m}(t)dt \\ &= \sum_m g_0(m-2k)c_{j+1}(m) \end{aligned} \tag{A.6}$$

In the same way a relation between the wavelet function coefficients and the scaling function coefficients at a higher level can be found.

$$d_j(k) = \sum_m g_1(m-2k)c_{j+1}(m) \tag{A.7}$$

A.2 The Inverse Calculation

If $f(t) \in V_{j_0+1}$, $f(t)$ can be written as a sum of scaling functions at level $j_0 + 1$.

$$f(t) = \sum_k c_{j_0+1}(k) 2^{(j_0+1)/2} \varphi(2^{(j_0+1)}t - k) \tag{A.8}$$

Or as a sum of scaling functions and wavelet functions at level j_0

$$\begin{aligned} f(t) &= \sum_k c_{j_0}(k)\varphi_{j_0,k}(t) + \sum_k d_{j_0}(k)\psi_{j_0,k}(t) \\ &= \sum_k c_{j_0}(k) \sum_n g_0(n) 2^{(j+1)/2} \varphi(2^{(j+1)}t - 2k - n) \\ &\quad + \sum_k d_{j_0}(k) \sum_n g_1(n) 2^{(j+1)/2} \varphi(2^{(j+1)}t - 2k - n) \end{aligned} \tag{A.9}$$

where equation (A.3) was used.
Now setting the two above equations equal to each other, multiplying by $\varphi(2^{(j_0+1)}t-$

m) and taking the integral gives

$$\begin{aligned}
&2^{(j_0+1)/2} \sum_k c_{j_0+1}(k) \int \varphi(2^{(j_0+1)}t - m)\varphi(2^{(j_0+1)}t - k)dt \\
&= 2^{(j_0+1)/2} \sum_k c_{j_0}(k) \sum_n g_0(n) \int \varphi(2^{(j_0+1)}t - m)\varphi(2^{(j_0+1)}t - 2k - n)dt \\
&+ 2^{(j_0+1)/2} \sum_k d_{j_0}(k) \sum_n g_1(n) \int \varphi(2^{(j_0+1)}t - m)\varphi(2^{(j_0+1)}t - 2k - n)dt
\end{aligned} \tag{A.10}$$

Remembering that the scaling functions are orthogonal, the integral on the left side is non-zero only for $m = k$, and the integrals on the right side are only non-zero for $m = 2k+n$. We then finally get the following equation to calculate the inverse wavelet transform.

$$c_{j_0+1}(m) = \sum_k c_{j_0}(k)g_0(m - 2k) + \sum_k d_{j_0}(k)g_1(m - 2k) \tag{A.11}$$

Appendix B

Complex Wavelet Packet Transform Filter Coefficients

	$h_0^{\mathfrak{Re}}$	$h_1^{\mathfrak{Re}}$	$h_0^{\mathfrak{Im}}$	$h_1^{\mathfrak{Im}}$
$h(1)$	0,00325314	-0,00455690	-0,00455690	-0,00325314
$h(2)$	-0,00388321	0,00543948	-0,00543948	-0,00388321
$h(3)$	0,03466035	0,01702522	0,01702522	-0,03466035
$h(4)$	-0,03887280	-0,02382538	0,02382538	-0,03887280
$h(5)$	-0,11720389	-0,10671180	-0,10671180	0,11720389
$h(6)$	0,27529538	-0,01186609	0,01186609	0,27529538
$h(7)$	0,75614564	0,56881042	0,56881042	-0,75614564
$h(8)$	0,56881042	-0,75614564	0,75614564	0,56881042
$h(9)$	0,01186609	0,27529538	0,27529538	-0,01186609
$h(10)$	-0,10671180	0,11720389	-0,11720389	-0,10671180
$h(11)$	0,02382538	-0,03887280	-0,03887280	-0,02382538
$h(12)$	0,01702522	-0,03466035	0,03466035	0,01702522
$h(13)$	-0,00543948	-0,00388321	-0,00388321	0,00543948
$h(14)$	-0,00455690	-0,00325314	0,00325314	-0,00455690

Table B.1: Coefficients of the length 14 q-shift filters for the forward complex wavelet packet transform

	$g_0^{\mathfrak{Re}}$	$g_1^{\mathfrak{Re}}$	$g_0^{\mathfrak{Im}}$	$g_1^{\mathfrak{Im}}$
$h(1)$	-0,00455690	-0,00325314	0,00325314	-0,00455690
$h(2)$	-0,00543948	-0,00388321	-0,00388321	0,00543948
$h(3)$	0,01702522	-0,03466035	0,03466035	0,01702522
$h(4)$	0,02382538	-0,03887280	-0,03887280	-0,02382538
$h(5)$	-0,10671180	0,11720389	-0,11720389	-0,10671180
$h(6)$	0,01186609	0,27529538	0,27529538	-0,01186609
$h(7)$	0,56881042	-0,75614564	0,75614564	0,56881042
$h(8)$	0,75614564	0,56881042	0,56881042	-0,75614564
$h(9)$	0,27529538	-0,01186609	0,01186609	0,27529538
$h(10)$	-0,11720389	-0,10671180	-0,10671180	0,11720389
$h(11)$	-0,03887280	-0,02382538	0,02382538	-0,03887280
$h(12)$	0,03466035	0,01702522	0,01702522	-0,03466035
$h(13)$	-0,00388321	0,00543948	-0,00543948	-0,00388321
$h(14)$	0,00325314	-0,00455690	-0,00455690	-0,00325314

Table B.2: Coefficients of the length 14 q-shift filters for the inverse complex wavelet packet transform

	$h_{0f}^{\mathfrak{Re}}$	$h_{1f}^{\mathfrak{Re}}$	$h_{0f}^{\mathfrak{Im}}$	$h_{1f}^{\mathfrak{Im}}$
$h(1)$	0,00000000	0,00000000	0,00793854	0,00000000
$h(2)$	-0,06250000	-0,00793854	0,00793854	0,00000000
$h(3)$	0,06250000	0,00793854	-0,06250000	-0,06250000
$h(4)$	0,49206146	0,06250000	0,06250000	-0,06250000
$h(5)$	0,49206146	0,06250000	0,49206146	0,49206146
$h(6)$	0,06250000	-0,49206146	0,49206146	-0,49206146
$h(7)$	-0,06250000	0,49206146	0,06250000	0,06250000
$h(8)$	0,00793854	-0,06250000	-0,06250000	0,06250000
$h(9)$	0,00793854	-0,06250000	0,00000000	0,00793854
$h(10)$	0,00000000	0,00000000	0,00000000	-0,00793854

Table B.3: Coefficients of the length 10 first stage filters for the forward complex wavelet packet transform

	$g_{0f}^{\mathfrak{Re}}$	$g_{1f}^{\mathfrak{Re}}$	$g_{0f}^{\mathfrak{Im}}$	$h_{1f}^{\mathfrak{Im}}$
$h(1)$	0,00000000	0,00000000	0,00000000	-0,00793854
$h(2)$	0,00793854	-0,06250000	0,00000000	0,00793854
$h(3)$	0,00793854	-0,06250000	-0,06250000	0,06250000
$h(4)$	-0,06250000	0,49206146	0,06250000	0,06250000
$h(5)$	0,06250000	-0,49206146	0,49206146	-0,49206146
$h(6)$	0,49206146	0,06250000	0,49206146	0,49206146
$h(7)$	0,49206146	0,06250000	0,06250000	-0,06250000
$h(8)$	0,06250000	0,00793854	-0,06250000	-0,06250000
$h(9)$	-0,06250000	-0,00793854	0,00793854	0,00000000
$h(10)$	0,00000000	0,00000000	0,00793854	0,00000000

Table B.4: Coefficients of the length 10 first stage filters for the inverse complex wavelet packet transform

Bibliography

[1] S. Mallat. *A wavelet tour of signal processing*. Academic Press, 2. edition, 1999.

[2] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms*. Prentice Hall, 1998.

[3] J. E. Odegard, R. A. Gopinath, and C. S. Burrus. Optimal wavelets for signal decomposition and the existence of scale limited signals. In *IEEE Proc. Int. Conf. Acoust., Speech, Signal Processing*, volume 4, pages IV 597–600, San Francisco, CA, 1992.

[4] S. Ayat, M. T. Manzuri, and R. Dianat. Wavelet based speech enhancement using a new thresholding algorithm. In *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing*, October 2004.

[5] I. M. Johnstone and B. W. Silverman. Wavelet threshold estimators for data with correlated noise. *Journal of the Royal Statistical Society B*, 59(2):319–351, 1997.

[6] T. Weickert and U. Kiencke. Adaptive estimation of periodic noise energy distributions for speech enhancement. In *Proceedings of 9th IFAC Workshop ALCOSP'07*, 2007.

[7] I. Cohen, S. Raz, and D. Malah. Shift invariant wavelet packet bases. In *IEEE Proc. Int. Conf. Acoust., Speech, Signal Processing*, volume 4, pages 1080–1084, Detroit, MI, 1995.

[8] J. O. Smith. Mathematics of the discrete fourier transform (dft). Website. Stanford University, California. http://ccrma.stanford.edu/~jos/mdft/.

[9] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury. The dual-tree complex wavelet transform. *IEEE Signal Processing Magazine*, 22(6):123–151, November 2005.

[10] Ivan W. Selesnick. The design of approximate hilbert transform pairs of wavelet bases. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 50(5):1144–1152, May 2002.

[11] N. G. Kingsbury. Design of q-shift complex wavelets for image processing using frequency domain energy minimisation. In *IEEE Proc. Conf. on Image Processing*, Barcelona, 2003.

[12] I. W. Selesnick, S. Cai, and K. Li. DTCWT first stage filter. Website. Polytechnic Institute, New York. http://taco.poly.edu/WaveletSoftware/.

[13] A. Jalobeanu, L. Blanc-Féraud, and J. Zerubia. Satellite image deblurring using complex wavelet packets. *IJCV*, 51(3):205–217, 2003.

[14] N. G. Kingsbury. A dual-tree complex wavelet transform with improved orthogonality and symmetry properties. In *Proc. International Conference on Image Processing, 2000*, volume 2, pages 375–378, 2000.

[15] B. McKinley and G. Whipple. Model based speech pause detection. In *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)-Volume 2*, page 1179, Washington, DC, USA, 1997. IEEE Computer Society.

[16] E. Zavarehei and S. Vaseghi. Spectral subtraction. Website. Brunel University, London. http://dea.brunel.ac.uk/cmsp/Home_Esfandiar/.

[17] D. Donoho, A. Maleki, and M. Shahram. WAVELAB850 comprehensive wavelet toolbox for Matlab. Website. Stanford University, California. http://www-stat.stanford.edu/~wavelab/.

[18] R. A. Johnson. *Miller and Freund's Probability and Statistics for Engineers*. Prentice Hall, 2000.