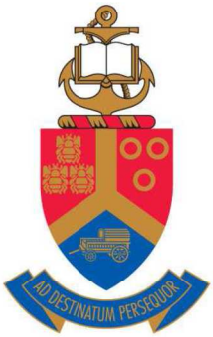


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

NON-STATIONARY SIGNAL CLASSIFICATION FOR RADAR TRANSMITTER IDENTIFICATION

MC DU PLESSIS

2010

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

NON-STATIONARY SIGNAL CLASSIFICATION FOR RADAR TRANSMITTER IDENTIFICATION

By

Marthinus Christoffel du Plessis

Study leader: Prof. J.C. Olivier

Submitted in partial fulfilment of the requirements for the degree

Master of Engineering (Computer)

in the

Department of Electrical, Electronic & Computer Engineering

of the

Faculty of Engineering, Built Environment & Information Technology

UNIVERSITY OF PRETORIA

January 2010

SUMMARY

NON-STATIONARY SIGNAL CLASSIFICATION FOR RADAR TRANSMITTER IDENTIFICATION

by

Marthinus Christoffel du Plessis

Study leader: Prof. J.C. Olivier

Department of Electrical, Electronic & Computer Engineering

Master of Engineering (Computer)

The radar transmitter identification problem involves the identification of a specific radar transmitter based on a received pulse. The radar transmitters are of identical make and model. This makes the problem challenging since the differences between radars of identical make and model will be solely due to component tolerances and variation.

Radar pulses also vary in time and frequency which means that the problem is non-stationary. Because of this fact, time-frequency representations such as shift-invariant quadratic time-frequency representations (Cohen's class) and wavelets were used. A model for a radar transmitter was developed. This consisted of an analytical solution to a pulse-forming network and a linear model of an oscillator.

Three signal classification algorithms were developed. A signal classifier was developed that used a radially Gaussian Cohen's class transform. This time-frequency representation was refined to increase the classification accuracy. The classification was performed with a support vector machine classifier.

The second signal classifier used a wavelet packet transform to calculate the feature values. The classification was performed using a support vector machine. The third signal classifier also used the wavelet packet transform to calculate the feature values but used a Universum type classifier for classification. This classifier uses signals from the same domain

to increase the classification accuracy.

The classifiers were compared against each other on a cubic and exponential chirp test problem and the radar transmitter model. The classifier based on the Cohen's class transform achieved the best classification accuracy. The classifier based on the wavelet packet transform achieved excellent results on an Electroencephalography (EEG) test dataset. The complexity of the wavelet packet classifier is significantly lower than the Cohen's class classifier.

Keywords:
Non-stationary signal classification, quadratic time-frequency representation, discrete wavelet transform, multiresolution analysis, wavelet packet transform, support vector machine, Wigner-Ville transform, Battle-Lemarié wavelet.

OPSOMMING

NON-STATIONARY SIGNAL CLASSIFICATION FOR RADAR TRANSMITTER IDENTIFICATION

deur

Marthinus Christoffel du Plessis

Studie leiers: Prof. J.C. Olivier

Departement Elektriese-,Elektroniese- & Rekenaar Ingenieurswese

Meester in Ingenieurswese (Rekenaar)

Die radar sender identifikasie probleem behels die identifikasie van ’n spesifieke radar gebaseer op ’n ontvangde puls. In die probleem word aanvaar dat die radar senders dieselfde maak en model is. Dit maak die probleem moeiliker aangesien die verskille tussen radars van identiese maak en model slegs veroorsaak word deur komponent toleransies.

Radar pulse verander in beide tyd en frekwensie wat beteken dat die probleem nie-stasionêr is. As gevolg hiervan moet tyd en frekwensie voorstellings soos skuif-invariante kwadratiese tyd-frekwensie voorstellings (Cohen se klas) en golfies gebruik word. ’n Model vir ’n radar sender was ontwikkel. Dit bestaan uit ’n analitiese oplossing van ’n stel differensiaal vergelykings wat ’n pulsvormingsnetwerk beskryf en ’n lineêre model vir ’n ossillator.

Drie sein klassifikasie algoritmes is ontwikkel. ’n Sein klassifiseerder was ontwikkel wat die radiaal Gaussiese Cohen klas transformasie gebruik. Die tyd en frekwensie voorstelling was verstel om die klassifikasie akkuraatheid te maksimeer. Die klassifikasie is met ’n ondersteunings vektor klassifiseerder uitgevoer.

’n Tweede klassifiseerder wat gebruik maak van die golf-pakkie transformasie om die kenmerke van die sein uit te werk. Die klassifikasie was gedoen met behulp van ’n ondersteuningsvektor klassifiseerder. ’n Derde sein klassifiseerder is ontwikkel wat ’n “Universum”-klassifiseerder gebruik het. Die klassifiseerder gebruik seine van dieselfde

domein om klassifikasie akkuraatheid te maksimeer.

Die klassifiseerders was teen mekaar vergelyk met ’n kubiese tjirp en eksponensiële tjirp probleem asook op die radar sender model. Die klassifiseerder gebaseer op die Cohen klas transformasie het die beste klassifikasie akkuraatheid behaal. Die klassifiseerder gebaseer op die golflet pakkie transformasie het uitstekende resultate getoon op die EEG datastel. Die kompleksiteit van die golflet pakkie klassifiseerder is beduidend laer as die Cohen klas klassifiseerder .

Sleutelwoorde:
Nie-stasionêre sein klassifikasie, kwadratiese tyd en verkwensie voorstellings, diskrete golflet transform, golflet pakkie transform, vektor ondersteunings masjien, Wigner-Ville transform, Battle-Lemarié golf.

To my family

who supported me through my years of study.

ACKNOWLEDGEMENTS

My heartfelt thanks to the following people and institutions.

- First and foremost, my appreciation and thanks go to my parents for their enduring love, support and encouragement.
- I am grateful to my supervisor, prof. J.C. Olivier for his support and guidance during my study.
- I would like to thank Phillip Botha whose comments on the draft of this dissertation helped to significantly improve the quality.
- My thanks to Hans Grobler for answering several pattern recognition questions and help with the Linux Cluster.
- I would like to thank Ferdi Potgieter for help in the development of the transmitter model.
- Credit goes to the Council for Scientific and Industrial Research (CSIR) for the financial sponsorship of my master's degree.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations

ANN	Artificial Neural Network
AWGN	Additive White Gaussian Noise
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
EEG	Electroencephalogram
FMCW	Frequency Modulated Continuous Wave
HPF	High Pass Filter
LPF	Low Pass Filter
PFN	Pulse Forming Network
PSO	Particle Swarm Optimisation
RDWT	Redundant Discrete Wavelet Transform
RGK	Radially Gaussian Kernel
SSL	Semi-Supervised Learning
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine
TFR	Time frequency representation
WPT	Wavelet Packet Transform
WVD	Wigner-Ville distribution

CHAPTER ONE

INTRODUCTION

The main focus of this dissertation is on the development and evaluation of non-stationary signal classification methods for use in radar transmitter identification. The goal of radar transmitter identification is to identify the particular radar transmitter from which a specific pulse originated. The problem is challenging since the system should be able to distinguish between radars of the same make and model (but with differing serial numbers). Transmitters of the same make and model therefore differ due to factors such as manufacturing tolerances, tolerances on components and damage to components. Other effects (such as heat) also cause pulses from the same transmitter to differ from each other.

These differences in the transmitters cause differences in the modulation of the radar pulses. The signal will therefore be non-stationary (i.e. the frequency content of the signal will vary with time). Due to the non-stationary nature of the signal, frequency-based methods such as the Fourier transform cannot be used and methods based on time-frequency representations (TFR) should be developed.

To ensure that the problem is realistic, it is assumed that the classifier does not have a model of the source of the pulses. The transmitter classification problem is viewed as a special case of the broader non-stationary signal classification problem and the classifier is developed to be problem independent. This allows the classifier to be applied to a variety of non-stationary signal classification tasks, an example of which is electroencephalogram (EEG) classification problems [3].

1.1 BRIEF BACKGROUND

The radar transmitter identification problem has not yet been widely studied. The most significant research found are available in [5] and [6] and are based on the application of a general non-stationary signal classifier developed in [7] and [8].

Non-stationary signal classification is performed in most research studies by mapping the signals to a time-frequency representation from which a small amount of features are extracted. These features are then used in combination with a classification algorithm such as an artificial neural network (ANN). The most prominent time-frequency representations are Cohen's class distributions (also known as shift-invariant quadratic distributions) and discrete wavelet transforms.

Cohen's class distributions are quadratic representations of the signal energy over time and frequency. This class of distributions have the shared property that they are invariant with respect to time and frequency shifts. The disadvantages of these distributions are that they may produce interference terms (also called cross-terms) and that they may produce negative values. Because of these properties, they are sometimes referred to as quasi-distributions. The most prominent member of the Cohen's class of distributions is the Wigner-Ville transform which has important applications in signal processing and physics. The radially Gaussian distribution (described in [9]) is an attempt to reduce the effect of cross-terms by changing the distribution according to specific criteria such as classification accuracy.

The wavelet transform correlates the signal with a function that is both limited in time and frequency (the so-called wavelet). The advantage of using a wavelet is that it can be scaled to have a different spread in time and frequency, thereby allowing varying resolutions at different frequencies. The continuous wavelet transform scales the wavelet to have good time resolution at high frequencies and good frequency resolution at low frequencies (this is often advantageous for the processing of transient signals). A discrete counterpart of this transform, called the discrete wavelet transform (DWT), can be constructed as a multiresolution signal decomposition. The advantage of using a multiresolution decomposition is that it allows for an efficient implementation with filter banks. By changing the filterbank construction slightly, a signal can also be represented with both low and high frequency resolution at all frequencies.

This is known as the wavelet packet transform (WPT).

1.1.1 Non-stationary signal classifiers

A Cohen's class TFR was used with a radially Gaussian kernel function in a series of papers to perform non-stationary signal classification [10–13]. In [10] a classification system using the radially Gaussian kernel was presented. The radially Gaussian kernel is optimised according to a criterion function that is similar to the Fisher criterion (where the difference between class means is maximised while the variance within each class is minimised). A Kolmogorov distance metric was used to determine the distances between class means. This work was extended to several different distance measure cases and optimisation with respect to the probability of misclassification in [11]. The work was further extended to using a support vector machine (SVM) classifier in [12] (this paper did not explain how the radially Gaussian kernel was optimised). An optimisation method for the radially Gaussian distribution, using a support vector machine classifier, was presented in [13].

Classification methods using the DWT are less systematic and tend to be problem specific. Most methods treat the feature extraction and classification as two separate problems. This is mostly due to the limitations of the classifier used (such as ANNs) in classifying high dimensional spaces. In pattern recognition this limitation is known as the "curse of dimensionality" [14]. A method to compute the features for classification was described in [15]. In that paper, the DWT is computed and the energy values calculated from the DWT. These values are used as the input to a clustering-based feature extractor. These features are then subsequently used to perform the classification.

This approach was also followed by [16]. In that paper features, such as the average power of wavelet coefficients in each subband, were used for classification. An ANN was used as the classifier and the system was tested on a lung sound classification problem. A similar classifier was developed in [17] to analyse EEG data. In that paper simple features such as the minimum and maximum of wavelet coefficients in each subband were extracted and used as the input to an ANN.

An interesting classification method based on the wavelet packet transform (WPT) was

presented in [18]. This study used a wavelet packet transform to calculate an energy map for each bin in the wavelet packet tree. This energy map was then used to calculate a feature set using singular value decomposition.

1.2 OBJECTIVES OF THE DISSERTATION

The objectives of the dissertation are to:

- Develop a realistic radar transmitter model that can serve as a test bed for the evaluation of radar transmitter classification algorithms.
- Develop non-stationary signal classifiers that can perform accurate classification without requiring an explicit model of the source.
- Compare the newly developed signal classifiers to classifiers in literature on well-defined test problems.
- Simulate the classifiers on a realistic radar transmitter model in realistic channel conditions.

1.3 AUTHOR'S CONTRIBUTIONS AND PUBLICATIONS

1.3.1 Novel contributions

Some novel contributions developed in this dissertation are:

- The development of a radar transmitter model. This includes the modelling of a pulse-forming network (PFN) with a system of linear differential equations and the modelling of an oscillator. The channel is modelled as an additive white Gaussian noise (AWGN) channel.
- Significant improvements are made to the classifier presented in [13]. A new spread function using a Bernstein expansion is developed and this is optimised with a particle swarm optimisation (PSO) algorithm. A proof is presented to show that the constraints on the Bernstein expansion can be added to the PSO algorithm without any additional overhead if the initial particles are initialised in a specific manner. An alternative

optimisation criterion is derived that makes provision for multiple classes. A method is also developed to allow the criterion to be calculated with lower complexity (when compared to the method in [13]).

- A classifier based on the wavelet packet transform using a support vector machine is developed. The classifier has much lower complexity than the classifier based on the Cohen's class TFR. This classifier is also capable of using multiple signals per pattern.
- A classifier is developed that makes provision for the incorporation of a priori information. This information is in the form of similar signals that are not from the set of classes to be classified, but originates from the same domain.
- The filter coefficients for high order Battle-Lemarié wavelets are derived. Experimental results show that the choice of wavelet has a significant effect on the performance of the classifier. High order Battle-Lemarié wavelets performed well on certain problems. This was noteworthy since most research available in literature use Daubechies wavelets.
- The wavelet packet classifier is used to perform classification on a standard EEG dataset. The classifier showed a significant improvement when compared to other classifiers in literature.
- The simulation of the classifiers on a cubic and exponential chirp problem and on the radar transmitter model (in an AWGN channel).

1.3.2 Publications

In the course of this study the author co-wrote and presented the following conference paper:

- M.C. du Plessis, J.C. Olivier, "Radar transmitter classification using a non-stationary signal classifier," in *Proceedings of IEEE ICWAPR 2009*, Baoding, China, 11-15 July 2009, CDROM.

The author co-wrote and submitted the following journal paper:

- A paper entitled, "Non-stationary signal classifier for radar transmitter identification", authored by M.C. du Plessis and J.C. Olivier was submitted for publication in the Frontiers of Computer Sciences (China) journal in November 2009.

1.4 OUTLINE OF THE DISSERTATION

The dissertation consists of eleven chapters and four appendices. In Chapter 2 a radar transmitter model is developed. Chapter 3 discusses the particle swarm optimisation algorithm and Chapter 4 gives an overview of support vector machines. A reader who is familiar with these topics may pass over these sections. Chapter 5 discusses Cohen's class of quadratic TFRs. Wavelet transforms are discussed in Chapter 6. In that chapter, higher order Battle-Lemarié wavelets are also developed. This is done because the filter coefficients for higher order Battle-Lemarié wavelets are not readily available. These filter coefficients are given in Appendix C. Only the aspects of wavelet analysis that are directly applicable to methods developed in the dissertation are discussed – for a broad overview the reader is referred to [19–21].

Chapter 7 describes the classifier that uses the radially Gaussian TFR to represent the signal and uses an SVM to perform the classification. The performance of this classifier is compared with other classifiers in Chapter 10. A classifier using the wavelet packet transform to represent the signal and an SVM to perform the classification is developed in Chapter 8. The classification accuracy of this classifier (shown in section 10.3.1) exceeds that of several other classifiers based on EEG data. Chapter 9 shows how a priori information of the source can be implemented by using signals in the same domain. The conclusion is given in Chapter 11.

CHAPTER TWO

RADAR TRANSMITTER MODELLING

A radar transmitter model is developed in this section. The primary purpose of the transmitter model is to aid with the comparison of classification techniques. Knowledge of the model will not be used directly for the classification task – it is assumed that the model is unknown to the classifier. The motivation for this is threefold:

1. The classifiers will be used on several radar transmitters of differing make and models. To develop a model for each of these would be infeasible.

2. Schematics such as circuit diagrams are not readily available for most radar transmitters.

3. Data sheets for components such as oscillators do not in general give an exact model for deviations. This makes creating a stochastic model for a specific transmitter impossible for practical purposes.

Since each type of transmitter has certain unique characteristics, using a stochastic model can significantly aid in the classification process. Alternative methods can however be used to allow the incorporation of this prior information without the need to construct an explicit stochastic model of the source. One such a method is presented in Chapter 9.

A simple model with an analytic solution was preferred over a complex model since it simplifies simulation. By using such a model, there is no need to resort to numerical methods when solving the system of differential equations that describes the transmitter. By considering

only the components responsible for the greatest variation in the pulse, a simple model such as this can be constructed without decreasing the quality of the results obtained on the model.

2.1 STRUCTURE OF THE RADAR TRANSMITTER

For the dissertation the focus is on pulsed radar systems. The classification techniques that are developed in this dissertation are also applicable to other types of radar systems such as frequency modulated continuous-wave (FMCW) transmitters – although the pre-processing of the signal would be more complex. The motivation for modelling a pulsed radar transmitter is that pulse-forming networks (PFN) often used in pulsed radar transmitters are simpler to model than solid state components used in other transmitter types.

A pulse-forming network is a type of modulator. The purpose of the modulator (also called a pulser) is to provide a short pulse to a microwave oscillator. Magnetron oscillators are commonly used for this purpose. An analytic model for a PFN is developed in the next section followed by the development of an oscillator model.

2.2 PULSE-FORMING NETWORK (PFN)

Line-type modulators are often used as pulse-forming networks. A line-type modulator's behaviour is similar to that of an open-ended transmission line discharging through its characteristic impedance [22]. Line-type modulators are usually constructed as a network of identically valued capacitors and inductors. One such a network type – called a Guillemin E-type network – is given in figure 2.1.

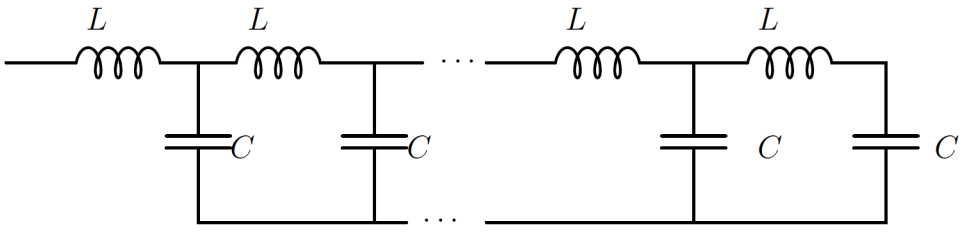


FIGURE 2.1: Structure of a Guillemin E-type pulse-forming network.

The network is charged to a voltage V_S and then discharges a pulse of length τ and an amplitude of $\frac{V_S}{2}$. The network consists of capacitors and inductors of identical size. The size of the components in a PFN can be calculated from [22, 23]

$$Z_0 = \sqrt{\frac{L_n}{C_n}} \tag{2.1}$$

$$\tau = \sqrt{L_n C_n} \tag{2.2}$$

$$C_n = \frac{\tau}{2Z_0} \tag{2.3}$$

$$L_n = \frac{\tau Z_0}{2}, \tag{2.4}$$

where Z_0 is the characteristic impedance (chosen to match the load), τ is the pulse length and L_n and C_n are the total network inductance and capacitance. The induction and capacitance per section can be calculated as

$$C = \frac{C_n}{n} \tag{2.5}$$

$$L = \frac{L_n}{n}, \tag{2.6}$$

where n is the number of sections in the PFN. The more sections a PFN has the higher the rise time of the pulse will be (the resulting pulse will therefore be more rectangular). The analytic solution for the output of a PFN for an ideal biased diode is developed in the next section.

2.2.1 Analytical model of PFN

Figure 2.2 gives the case where the PFN discharges through a load consisting of a voltage source and resistor. This situation occurs in the linear model for the oscillator when the output voltage of the PFN is larger than the"switch on" voltage (also called the Hartree voltage) for the oscillator. The linear model for the oscillator is developed in the next section. The PFN circuit is modelled as a system of linear differential equations. The equations for the current through the oscillator and voltage over the first capacitor is

$$i_1(t) = -\frac{v\,(t)}{R_\ell} \tag{2.7}$$

$$v_1\,(t) = (v\,(t) + V_H) - L_1 \frac{di_1\,(t)}{dt}. \tag{2.8}$$

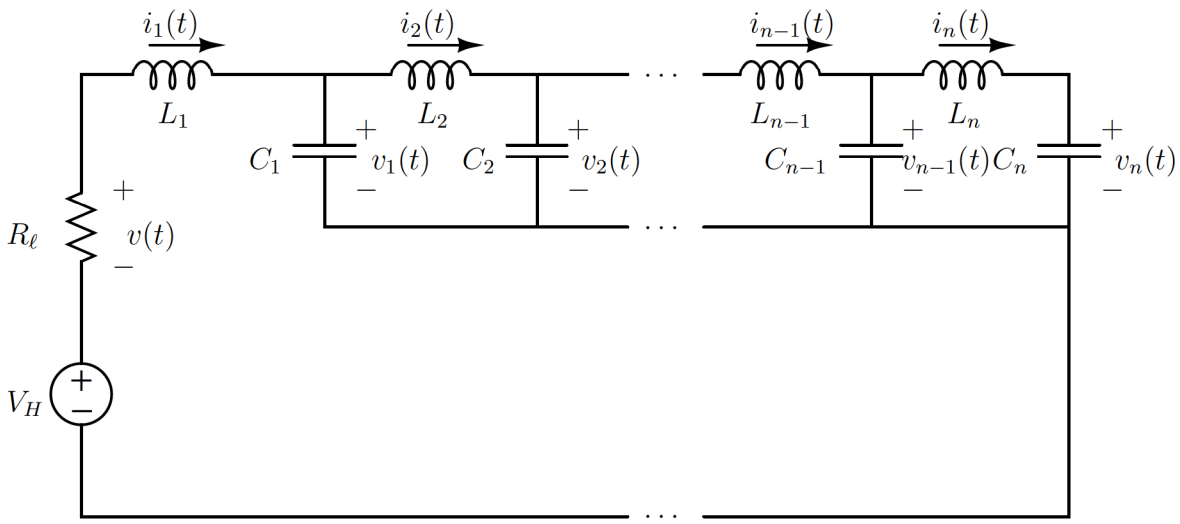


FIGURE 2.2: The discharging circuit for the pulse-forming network with a ideal biased diode model of section 2.3. In this model the output voltage is higher than V_H. All the currents and voltages are indicated.

For the remaining sections the system can be calculated as

$$i_2(t) = i_1(t) - C_1\frac{dv_1(t)}{dt} \tag{2.9}$$

$$v_2(t) = v_1(t) - L_2\frac{di_2(t)}{dt} \tag{2.10}$$

$$\vdots$$

$$i_n(t) = i_{n-1}(t) - C_{n-1}\frac{dv_{n-1}(t)}{dt} \tag{2.11}$$

$$v_n(t) = v_{n-1}(t) - L_n\frac{di_n(t)}{dt}. \tag{2.12}$$

The current through the last capacitor in the system can also be calculated in terms of $v_n(t)$ as

$$i_n(t) = C_n\frac{dv_n(t)}{dt}. \tag{2.13}$$

Substituting equation 2.7 into equation 2.8 results in

$$\frac{di_1(t)}{dt} = -\frac{R_\ell}{L_1}i_1(t) - \frac{1}{L_1}v_1(t) + \frac{V_H}{L_1}. \tag{2.14}$$

The above equations can be rewritten in a system format given below.

$$\frac{di_1(t)}{dt} = -\frac{R_\ell}{L_1}i_1(t) - \frac{1}{L_1}v_1(t) + \frac{V_H}{L_1} \tag{2.15}$$

$$\frac{dv_1(t)}{dt} = \frac{1}{C_1}i_1(t) - \frac{1}{C_1}i_2(t) \tag{2.16}$$

$$\frac{di_2(t)}{dt} = \frac{1}{L_2}v_1(t) - \frac{1}{L_2}v_2(t) \tag{2.17}$$

$$\vdots \quad \vdots \tag{2.18}$$

$$\frac{dv_{n-1}(t)}{dt} = \frac{1}{C_{n-1}}i_{n-1}(t) - \frac{1}{C_{n-1}}i_n(t) \tag{2.19}$$

$$\frac{di_n(t)}{dt} = \frac{1}{L_n}v_{n-1}(t) - \frac{1}{L_n}v_n(t) \tag{2.20}$$

$$\frac{dv_n(t)}{dt} = \frac{1}{C_n}i_n(t) \tag{2.21}$$

This gives rise to a matrix notation. The currents and voltages in the system can be expressed as a vector

$$\mathbf{x}(t) = \begin{bmatrix} i_1(t) \\ v_1(t) \\ i_2(t) \\ v_2(t) \\ \vdots \\ i_n(t) \\ v_n(t) \end{bmatrix}. \tag{2.22}$$

Using the above definition the system can be written as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b}. \tag{2.23}$$

The coefficient matrix of this system is

$$\mathbf{A} = \begin{bmatrix} -\frac{R_\ell}{L_1} & -\frac{1}{L_1} & 0 & 0 & 0 & \dots & 0 & 0 \\ \frac{1}{C_1} & 0 & -\frac{1}{C_1} & 0 & 0 & \dots & 0 & 0 \\ 0 & \frac{1}{L_2} & 0 & -\frac{1}{L_2} & 0 & \dots & 0 & 0 \\ 0 & 0 & \frac{1}{C_2} & 0 & -\frac{1}{C_2} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & -\frac{1}{C_{n-1}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & -\frac{1}{L_n} \\ 0 & 0 & 0 & 0 & 0 & \dots & \frac{1}{C_n} & 0 \end{bmatrix} \tag{2.24}$$

and the vector $\mathbf{b}$ is

$$\mathbf{b} = \begin{bmatrix} \frac{V_H}{L_1} \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}. \tag{2.25}$$

The solution of the above system is the sum of the characteristic and particular solution (the characteristic solution is the solution to the homogeneous equation $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t)$). This can be expressed as

$$\mathbf{x}\left(t\right) = \mathbf{x_c}\left(t\right) + \mathbf{x_p}\left(t\right). \tag{2.26}$$

Since the non-homogeneous term is constant, a constant term will be used as a particular solution. Differentiating the constant term $\mathbf{x}_p(t) = \left[a_1\ a_2\ \dots\ a_{2n-1}\ a_{2n}\right]^T$ gives $\dot{\mathbf{x}}_\mathbf{p}(t) = \mathbf{0}$. This constant term can therefore be calculated as

$$\dot{\mathbf{x}}_\mathbf{p}(t) = \mathbf{A}\mathbf{x}_\mathbf{p}(t) + \mathbf{b} \tag{2.27}$$

$$\mathbf{0} = \mathbf{A}\mathbf{x}_\mathbf{p}(t) + \mathbf{b} \tag{2.28}$$

$$\mathbf{x}_\mathbf{p}(t) = -\mathbf{A}^{-1}\mathbf{b} \tag{2.29}$$

The characteristic solution will be the general solutions to the homogeneous system

$$\mathbf{x}_c(t) = c_1\mathbf{x}_1(t) + c_2\mathbf{x}_2(t) + \ldots + c_{2n-1}\mathbf{x}_{2n-1}(t) + c_{2n}\mathbf{x}_{2n}(t). \tag{2.30}$$

The above can be calculated as

$$\mathbf{x}_c(t) = c_1 e^{\lambda_1 t}\mathbf{u}_1 + c_2 e^{\lambda_2 t}\mathbf{u}_2 + \ldots + c_{2n-1}e^{\lambda_{2n-1} t}\mathbf{u}_{2n-1} + c_{2n}e^{\lambda_{2n} t}\mathbf{u}_{2n} \tag{2.31}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_{2n}$ are the eigenvalues of $\mathbf{A}$ and $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{2n}$ are the eigenvectors of $\mathbf{A}$. The constants can be calculated from the initial values as

$$\mathbf{c} = \mathbf{U}^{-1}\left(\mathbf{x}\left(0^-\right) - \mathbf{x_p}\right), \tag{2.32}$$

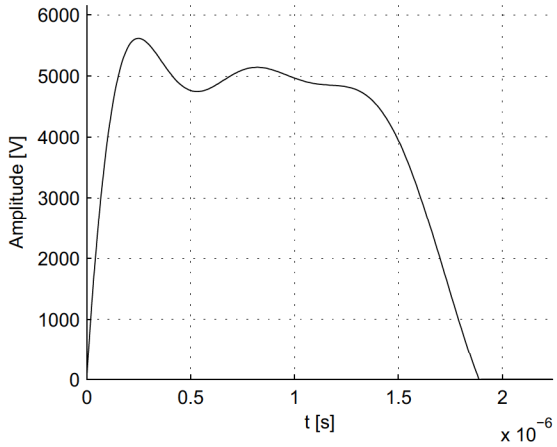
where $\mathbf{c}$ is the coefficient vector, $\mathbf{U}$ is a matrix containing the eigenvectors and $\mathbf{x}(0^-)$ is the initial values of a fully charged network. The initial values are expressed as

$$\mathbf{x}\left(0^-\right) = \begin{bmatrix} 0 \\ V_S \\ 0 \\ V_S \\ \vdots \\ 0 \\ V_S \end{bmatrix}. \tag{2.33}$$

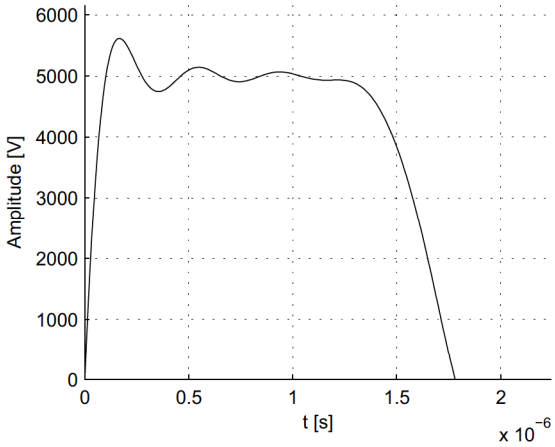
The pulse will end when $v(t) = -i_1(t)/R_\ell < 0$. Since this equation is the sum of exponentials it is best to determine the end condition numerically.

2.2.1.1 Example of pulses

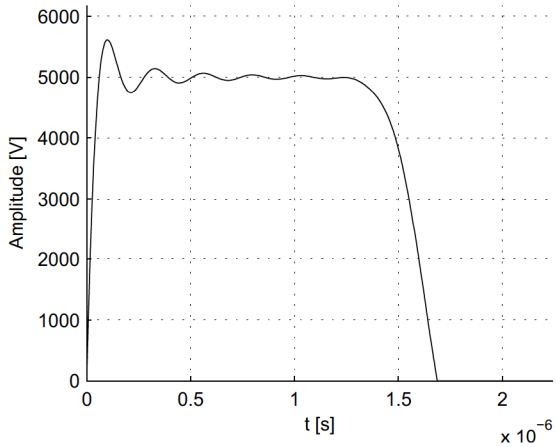
An example of a few pulses with a purely resistive load is given in figure 2.3. The design parameters for the pulses were: $V_S = 10kV, \tau = 1.5\mu s, Zo = 1250\Omega$ and $V_H = 0V$. These pulses only differ in the number of sections used for each pulse. As can be seen from the figures, the more sections the PFN contains the more rectangular the pulse will be (corresponding to a higher rise time).



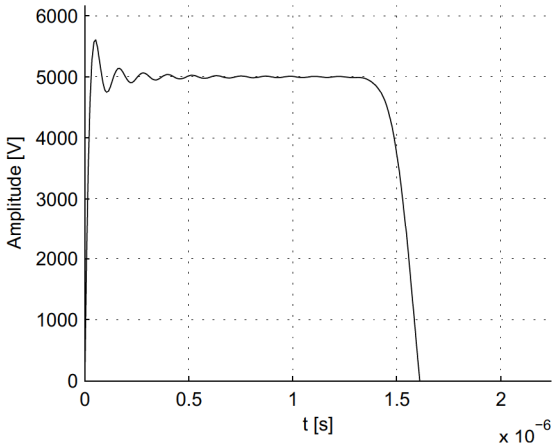
(a) Pulse from a 4 section PFN.



(b) Pulse from a 6 section PFN.



(c) Pulse from a 10 section PFN.



(d) Pulse from a 20 section PFN.

FIGURE 2.3: Example of 1.5μs long 5kV pulses from pulse-forming networks of different length.

2.2.2 Simulink model

A Simulink[1] model was developed for a two section pulse forming network in order to validate the above model. The Simulink model is given in figure 2.4.

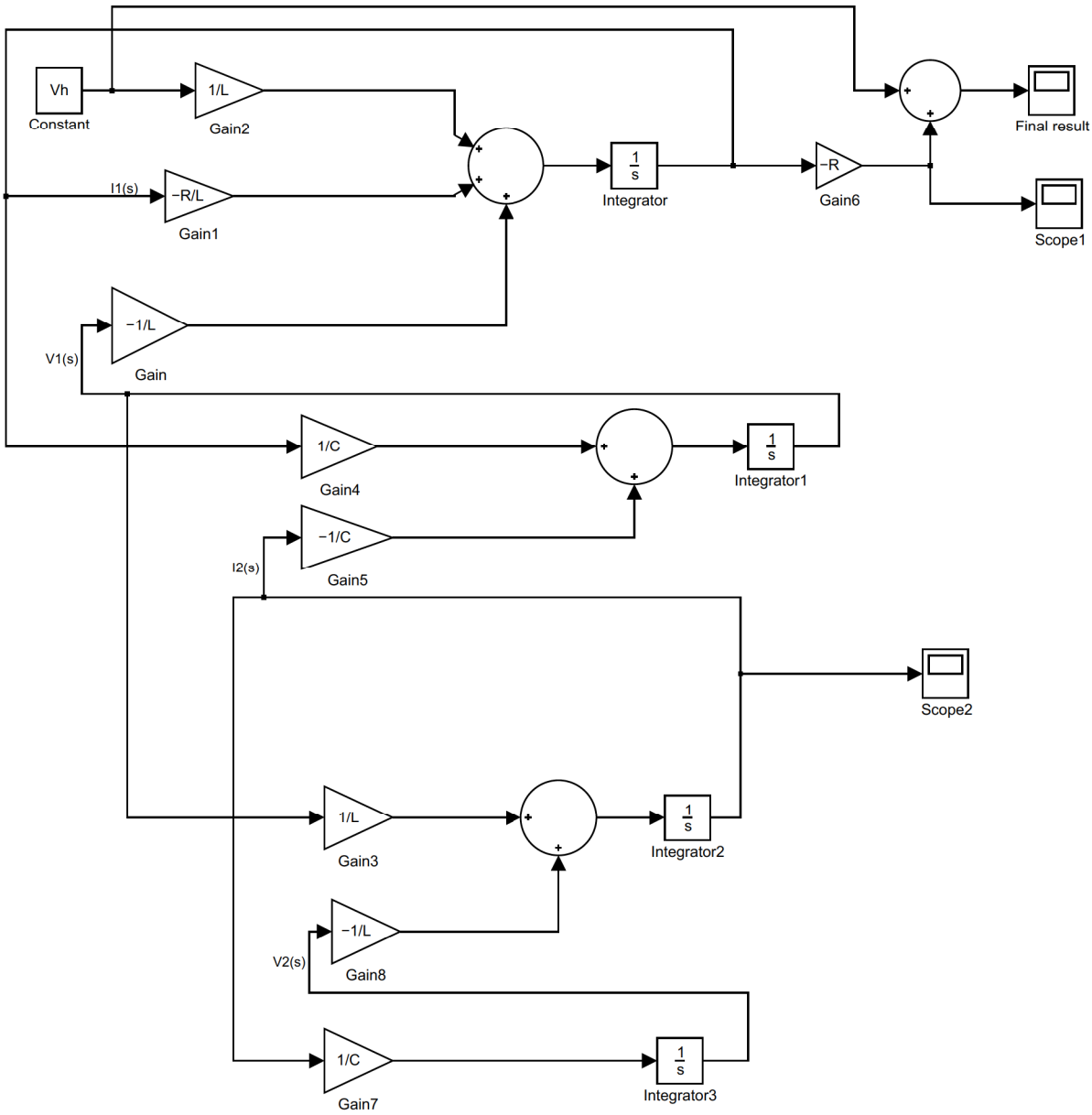


FIGURE 2.4: Simulink model of a two section pulse-forming network.

[1] Mathworks Simulink is a program for modelling dynamic systems. The result of the simulation is obtained by numerical methods.

2.3 OSCILLATOR MODEL AND MODEL VARIATION

Magnetrons are commonly used as oscillators in pulsed radar systems. In this section a simple model for magnetrons are developed. The input current and voltage relationship in magnetrons are highly non-linear. Only a small amount of current is drawn in a magnetron until a certain voltage – called the Hartree voltage – is reached [22]. This is similar to a reverse-biased Zener-diode with a certain "switch on" voltage. This behaviour is illustrated in figure 2.6(a). A linear model was created to approximate this (figure 2.6(b)). The primary motivation for the linear model is to ensure that the system differential equations can be solved analytically. The value for R_ℓ is calculated as

$$R_\ell = R_f - \frac{V_H}{I_O}. \tag{2.34}$$

The circuit model for this is a voltage source in series with a resistor if the applied voltage across the oscillator is higher than the Hartree voltage.

The oscillator output was modelled as a cosine signal of random phase (since most magnetrons are non-coherent) with a small amount of frequency pushing. Frequency pushing occurs when the oscillating frequency in some oscillators (such as magnetrons) is dependent on the input current [24]. The pushing figure and shape differs significantly from oscillator model to oscillator model. The frequency pushing was modelled as the input current multiplied by the pushing figure ($\mathrm{MHz/A}$) over a small range of currents around the operating current I_O. The power dissipated by the oscillator is directly proportional to the power dissipated by the linear model.

The actual value for capacitors are assumed to be uniformly spread within a range of $x\%$ of the specified values. This is a standard model with differences due to manufacturing tolerances in the components. The reason for using a uniform distribution is threefold:

1. Datasheets only specify the tolerance range within a certain percentage of the nominal value (and not the distribution) which the actual value of the capacitor is guaranteed to be in (e.g. the International Electrotechnical Commission (IEC) 60063 standard).

2. Different types of capacitors have different underlying distributions. The specification of some types of capacitors (such as electrolytic capacitors) can even be asymmetric.

3. The capacitors are often measured beforehand and any capicitors that are not within the specified tolerance (as percentage of the nominal value) are rejected. If the initial distribution of the capacitors are Gaussian, the resulting distribution will not be. This is illustrated in figure 2.5.

It is also assumed that the capacitors will exhibit tolerances from signal to signal (due to such factors as heat). This is modelled as a Gaussian distribution with the standard deviation specified as a percentage of the nominal component value.

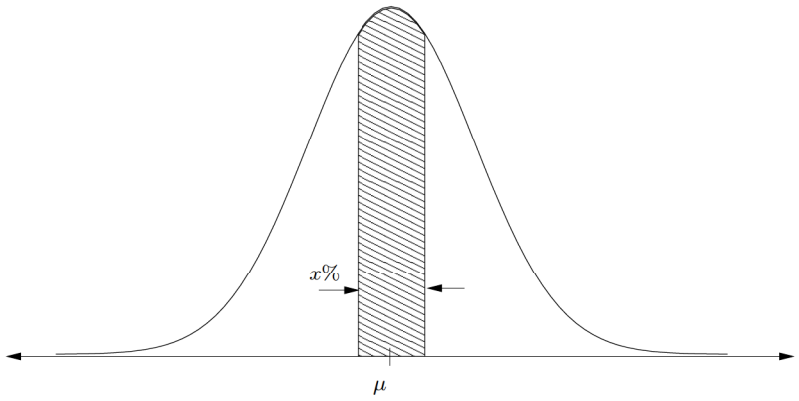
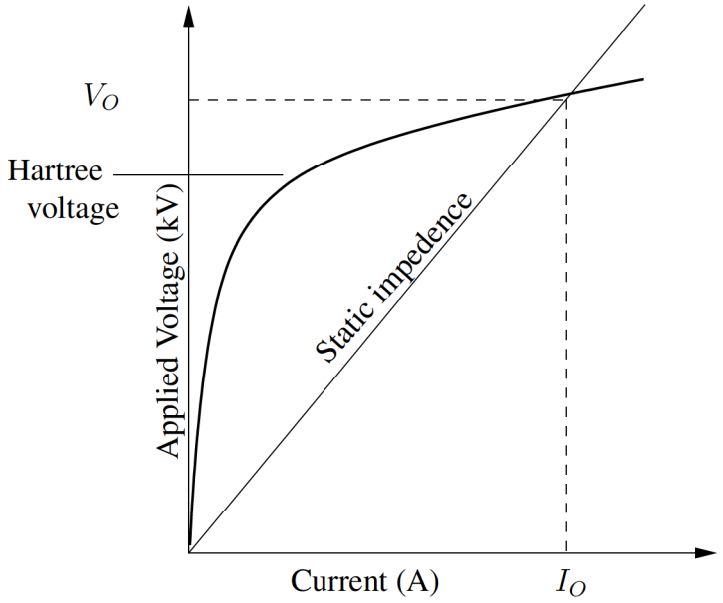
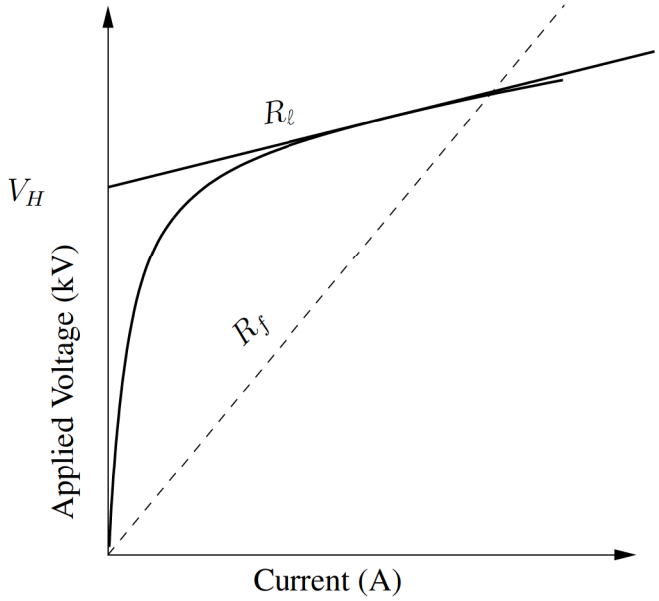


FIGURE 2.5: Selection of capacitors from a batch that is Normally distributed.



(a) Voltage current relationship of a magnetron (adapted from [22])



(b) Linear model developed from the true non-linear model

FIGURE 2.6: Model of a magnetron's current and voltage relationships and the linear model.

CHAPTER THREE

PARTICLE SWARM OPTIMISATION

Particle swarm optimisation (PSO) is a stochastic optimisation technique which is based on the social behaviour of birds in a flock [25]. The individual particles (each particle represents a solution) in the swarm are flown through the search space. The individual particles emulate the success of other individuals in the swarm [25]. If the vector $\mathbf{x_i}(t)$ represents a particle (which is a possible solution for the problem), the updated particle can be calculated as

$$\mathbf{x_i}(t+1) = \mathbf{x_i}(t) + \mathbf{v_i}(t+1), \tag{3.1}$$

where $\mathbf{v_i}(t)$ is a particle dependent velocity vector. The velocity of the particle update should reflect both the experimental knowledge of the particle and the socially exchanged information from the particle's neighbourhood [26]. The velocity update is divided into two components [26]:

1. The *cognitive component* which is proportional to the distance of the particle to its own best position.

2. The *social component* which is calculated from the socially exchanged information.

3.1 GLOBAL BEST PSO

The global best PSO algorithm uses the whole swarm as its neighbourhood. The personal best for particle i at time step t is denoted as $\mathbf{y_i}(t)$ and the global best by $\hat{\mathbf{y}}(t)$. The update for the

velocity of particle i is [26]

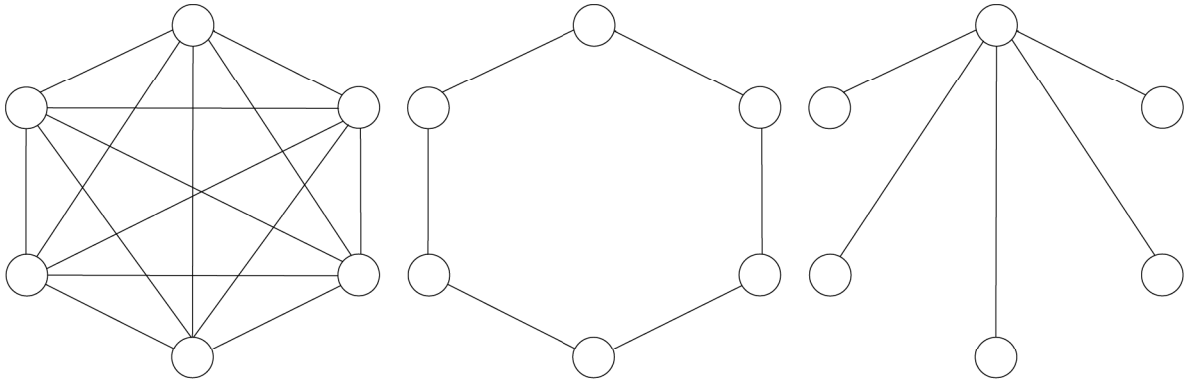
$$\mathbf{v_i}(t+1) = \phi \mathbf{v_i}(t) + \rho_1 (\mathbf{y_i}(t) - \mathbf{x_i}(t)) + \rho_2 (\hat{\mathbf{y}}(\mathbf{t}) - \mathbf{x_i}(t)), \tag{3.2}$$

where ϕ is the inertia constant and $\rho_1 = c_1 r_1$ and $\rho_2 = c_2 r_2$. The values r_1 and r_2 are chosen as uniformly distributed random variables in the range $[0, 1]$. These random values introduces a stochastic element in the algorithm. The constants c_1 and c_2 is used to scale the contribution of the personal best and the global best. The contribution due to the personal best is referred to as the cognitive component and the global best as the social component. The optimal choice of these values is problem dependent. In this update rule, the random values will only create a difference in the composition of the final update value and scale the individual contributions. The resulting velocity is therefore still a linear combination of $\mathbf{y}_i$, $\hat{\mathbf{y}}$ and $\mathbf{x}_i$.

The algorithmic description of the global best algorithm is given in algorithm 1 (the same notation as [26] is used). In the algorithm, the function $f(\mathbf{x})$ will be minimized.

Algorithm 1 Algorithmic description of the global best PSO that minimizes the function $f(\mathbf{x})$

repeat
  for $i = 1, \ldots, n_s$ **do**
    // set the personal best
    if $f(\mathbf{x_i}) < f(\mathbf{y_i})$ **then**
      $\mathbf{y_i} \leftarrow \mathbf{x_i}$
    end if
    // Set the global best position
    if $f(\mathbf{x_i}) < f(\hat{\mathbf{y}})$ **then**
      $\hat{\mathbf{y}} \leftarrow \mathbf{x_i}$
    end if
  end for
  for $i = 1, \ldots, n_s$ **do**
    // Update the velocity of particle i
    $r_1 \leftarrow U(0, 1)$
    $r_2 \leftarrow U(0, 1)$
    $\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 r_1(t) [\mathbf{y}_i(t) - \mathbf{x}_i(t)] + c_2 r_2(t) [\hat{\mathbf{y}}_j(t) - \mathbf{x}_i(t)]$
    // Update the position of particle i
    $\mathbf{x_i}(t+1) = \mathbf{x_i}(t) + \mathbf{v_i}(t+1)$
  end for
until Stopping condition is met



(a) The star neighbourhood structure. (b) The ring neighbourhood structure (c) The wheel neighbourhood structure

FIGURE 3.1: Different particle swarm optimisation neighbourhood topologies [26].

3.2 PARTICLE SWARM OPTIMISATION TOPOLOGIES

Several different types of topologies that are used for the update of the social component of the velocity are presented in [25]. Some basic topologies are given in figures 3.1(a), 3.1(b) and 3.1(c).

The topology of the particle swarm determines how the velocity update of each particle is calculated. The three topologies illustrated in figure 3.2 differ in their communication structure. In the star topology all the particles are connected and will therefore follow the global best particle [25]. The global best PSO in the previous section uses the star topology. The ring topology on the other hand, differs in that each particle only communicates with its immediate neighbours and will therefore attempt to move closer to the individual best of its neighbours. The advantage of the ring topology is that due to its slower convergence, a larger area of the search space is searched, though at the cost of slower convergence. A compromise in this is the wheel topology. All particles in the wheel topology are connected to the focal particle. Only the focal particle includes the global best in its velocity calculation (therefore following the global best). All the other particles will update from the focal particle. The focal particle therefore communicates any improvements to the rest of the particles.

3.3 STOPPING CONDITIONS

Several stopping criteria are presented in [26] and [25]. The simplest criteria is to terminate after a specified number of iterations or functional evaluations. This method has two drawbacks: the particle swarm may deliver a suboptimal solution, or the particle swarm will execute even though a global minimum has been found. This criterion is used in the implementation as a *worst case* bound. Some other stopping conditions are listed below.

- *Stop the algorithm when an acceptable solution is found.* In other words, the algorithm is stopped when $f(\mathbf{x_i}) < f(\mathbf{x}^*) + \epsilon$ where $\mathbf{x}^*$ is the global minimum. The drawback of this approach is that the global optimum has to be known. As an example of how this approach can be used, consider $\mathbf{x}$ as the parameters of a classifier (such as a neural network) and $f(\mathbf{x})$ the classification accuracy for the parameters $\mathbf{x}$ on a dataset. The algorithm may then be stopped if a sufficiently good classification accuracy is achieved using the parameters.

- *Stop the algorithm when no improvements are observed for a number of iterations.* For drawbacks of the stopping criterion of this algorithm see [26].

- *Stop when the normalised swarm radius is small.* The rationale for using an approach such as this is that the swarm will have little probability of improvement if all the particles lie together, since both the social and cognitive contribution will be extremely small.

3.4 INITIALISATION OF STARTING VALUES

The initial particles are set to uniformly distributed values in the range of the search space [25]

$$\mathbf{x_i}(0) \sim U(\mathbf{x_{min}}, \mathbf{x_{max}}). \tag{3.3}$$

The initial particles should also be linearly independent from each other to ensure that the whole subspace created by the initial vectors is searched (this aspect is addressed in section 7.3.2). The optimum cognitive and social values tend to be problem dependent, and should be set empirically. The sum of the cognitive and social values should however be smaller than 4 to ensure stability (see [25]). To ensure convergence the following relation should hold (proved in

Van Den Bergh [27])

$$\phi > \frac{1}{2}(c_1 + c_2) - 1 \tag{3.4}$$

where ϕ (the inertia constant in equation 3.2) is less than 1.

CHAPTER FOUR

SUPPORT VECTOR MACHINES

4.1 INTRODUCTION

The binary classification problem is the simplest model for classification. The task of the classifier is to associate a class label $\{-1, 1\}$ to any unknown vector $\mathbf{x}_i \in \mathcal{H}$ (where $\mathcal{H}$ is a vector space endowed with an inner product and a norm [28]). The training data will therefore be $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \ldots, (\mathbf{x}_\ell, y_\ell)\} \in \mathcal{H} \times \{-1, 1\}$. Classification in this setting can be performed with a hyperplane defined by

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0, \tag{4.1}$$

where $\langle ., . \rangle$ is the inner product. This leads to the decision function

$$f(\mathbf{x}) = sgn\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right). \tag{4.2}$$

Of all the possible hyperplanes that can separate the training patterns, the hyperplane that separates the training patterns with the maximum margin will lead to the best generalisation performance (see the discussion in the next section for the motivation). To simplify the calculation of the margin, the hyperplane in equation 4.1 can be scaled by a constant without changing its position. The hyperplane can therefore be scaled so that

$$\min_{i=1\ldots,\ell} |\langle \mathbf{w}, \mathbf{x_i} \rangle + b| = 1. \tag{4.3}$$

This form of a hyperplane is called a canonical hyperplane [2]. Assuming that the patterns are linearly separable, a separable hyperplane for $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\ell$ can be expressed as

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 \quad \textit{if } y_i = 1 \tag{4.4}$$

$$\langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq -1 \quad \textit{if } y_i = -1 \tag{4.5}$$

or more concisely as

$$y_i \left(\langle \mathbf{w}, \mathbf{x_i} \rangle + b \right) \geq 1. \tag{4.6}$$

The distance of a vector $\mathbf{x}$ from the hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ can be calculated as

$$r = \frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\|\mathbf{w}\|} + \frac{b}{\|\mathbf{w}\|}. \tag{4.7}$$

The vectors nearest to the hyperplane (called the "support vectors") will lie on either $\langle \mathbf{w}, \mathbf{x} \rangle + b = 1$ or $\langle \mathbf{w}, \mathbf{x} \rangle + b = -1$ (because of equation 4.3). By substituting this into the above equation, the margin will therefore be

$$\Delta = \frac{1}{\|\mathbf{w}\|} \tag{4.8}$$

(this is illustrated in figure 4.1).

4.1.1 Motivation for the maximum margin hyperplane

The motivation behind selecting the maximum margin separating hyperplane stems from statistical learning theory (SLT) considerations. The main aim of SLT is to guarantee good generalisation. Generalisation is the ability of a classifier to produce good classification accuracy on patterns not in the training set (unseen data) [29]. The generalisation performance of a support vector machine classifier is linked to its Vapnik-Chervonenkis (VC) dimension. The VC dimension is a measure of the "richness " of a classifier. An upper bound for the VC dimension for a hyperplane in n dimensional space is [30]

$$h \leq \min \left(\left[\frac{R^2}{\Delta} \right], n \right) + 1 \tag{4.9}$$

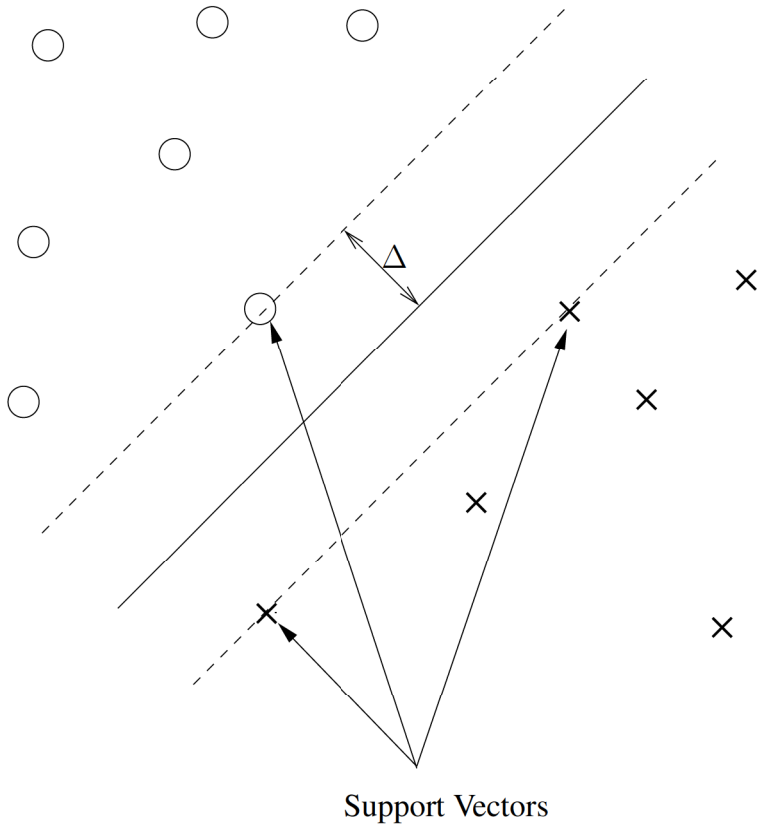


FIGURE 4.1: Optimal hyperplane and margin

where R is the radius of a sphere in which all the patterns are contained. Thus from the above equation it is clear that the larger the margin of separation between the two classes are, the lower the VC-dimension of the classifier will be. The following theorem from [30] relates the generalisation performance to the VC-dimension.

Theorem 1 ***Vapnik [30]*** *With probability* $1 - \eta$ *the probability that test examples will not be correctly separated by* Δ*-margin hyperplanes is bounded by*

$$P_{err} \leq \frac{m}{\ell} + \frac{\mathcal{E}}{2}\left(1 + \sqrt{1 + \frac{4m}{\ell\mathcal{E}}}\right) \tag{4.10}$$

where

$$\mathcal{E} = 4\frac{h\left(\ln\frac{2\ell}{h} + 1\right) - \ln\eta/4}{\ell}, \tag{4.11}$$

m *is the number of training examples that are not separated by the hyperplane and* h *is the bound of the VC-dimension.*

The first term in the above inequality is a measure for the training error (the number of samples that could not be separated divided by the total number of samples – i.e. m/ℓ). The second term is dependent on the VC dimension. Thus from equation 4.10 and 4.11, smaller the the VC dimension h is, the smaller $\mathcal{E}$ will be and consequently the error bound (P_{err}) will be smaller. Thus a maximum margin separating hyperplane provides the lowest error bound.

4.2 CALCULATION OF THE OPTIMAL HYPERPLANE

For the optimal hyperplane the margin $\Delta = \frac{1}{\|\mathbf{w}\|}$ is maximised. Therefore the problem can be expressed as the constrained quadratic program

$$\begin{aligned} &\textit{minimise} \quad \phi(\mathbf{w}) = \tfrac{1}{2}\|\mathbf{w}\|^2 \\ &\textit{subject to} \quad y_i\left[\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right] \geq 1 \quad i = 1, 2, \ldots, l. \end{aligned} \tag{4.12}$$

This is called the primal problem. The problem can also be calculated in its dual form with the aid of Lagrange multipliers (see [31–33] or [34] for a detailed discussion of Lagrange multipliers). The Lagrangian of the above problem is

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[\left(y_i\left[\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right]\right) - 1\right]. \tag{4.13}$$

Because of the inequality constraint the following constraints apply to the Lagrange multipliers

$$\alpha_i \geq 0, \ \ \forall i = 1 \ldots, \ell. \tag{4.14}$$

The primal problem is both convex and differentiable [2, 34]. This means that the Karush-Kuhn-Tucker (KKT) conditions are both necessary and sufficient for the solution [34]. The KKT conditions imply that [2]

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{0} \tag{4.15}$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0. \tag{4.16}$$

The above results in

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i \tag{4.17}$$

and

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0. \tag{4.18}$$

Equation 4.13 can be expanded as

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle - \sum_{i=1}^{l} \alpha_i y_i b + \sum_{i=1}^{l} \alpha_i. \tag{4.19}$$

The third term is zero when equation 4.18 is substituted. Substituting equation 4.17 into the first term of the above gives

$$\begin{aligned} \frac{1}{2}\|\mathbf{w}\|^2 &= \frac{1}{2}\langle \mathbf{w}, \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x} \rangle & (4.20) \\ &= \frac{1}{2} \sum_{i=1}^{\ell} \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle. & (4.21) \end{aligned}$$

Subtracting the second term from the first leaves

$$-\frac{1}{2} \sum_{i=1}^{\ell} \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle. \tag{4.22}$$

Substituting equation 4.17 again in the above results in

$$\begin{aligned} -\frac{1}{2} \sum_{i=1}^{\ell} \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle &= -\frac{1}{2} \sum_{i=1}^{\ell} \alpha_i y_i \langle \left(\sum_{j=1}^{\ell} \alpha_j y_j \mathbf{x}_j \right), \mathbf{x}_i \rangle & (4.23) \\ &= -\frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle. & (4.24) \end{aligned}$$

Finally, the Lagrangian can be completely written in terms of the multipliers as

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle. \tag{4.25}$$

The Dual problem can be expressed as [2] (keeping in mind the second condition for optimality, equation 4.18)

$$\begin{array}{ll} \textit{maximise} & W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \textit{subject to} & \alpha_i \geq 0, i = 1, 2, \ldots, l \\ \textit{and} & \sum_{i=1}^{l} \alpha_i y_i = 0. \end{array} \tag{4.26}$$

From the above it can be seen that the final dual problem depends only on inner products of the training data.

4.2.1 Discriminant function

The discriminant function for the SVM is therefore (by taking equation 4.17 into account)

$$g(\mathbf{x}) = sgn\left(\sum_{i=1}^{\ell} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b\right). \tag{4.27}$$

Since the Lagrange multipliers will be zero for non-support vectors, the above summation only needs to be performed over the support vector set. By using a positive support vector the b value for the hyperplane can be calculated as [35]

$$b = 1 - \langle \mathbf{w}, \mathbf{x}^s \rangle. \tag{4.28}$$

4.3 SUPPORT VECTOR MACHINES FOR THE NON-SEPARABLE CASE

In section 4.2, only the case where the patterns are linearly separable was treated. In most practical problems the two classes cannot be separated by a hyperplane. SVM formulations that deal with this problem are often called "soft margin" support vector machines. In this dissertation two approaches to the soft-margin SVM were used: C-SVM and ν-SVM.

4.3.1 C-SVM formulation

For the non-separable case, some patterns will violate the $y_i(\langle \mathbf{w}, \mathbf{x} \rangle + b) \geq 1$ constraint. This can be remedied by introducing a slack variable, ξ, into the constraint [36]. The constraints then

become [36]

$$y_i\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right) \geq 1 - \xi_i, \;\; i = 1, 2, \ldots, \ell. \tag{4.29}$$

By increasing the size of ξ_i for each pattern $\mathbf{x}_i$, the constraints can always be met. Since the size of the slack variables should be kept as small as possible, the problem can be written as [30]

$$\begin{array}{ll} \textit{minimise} & \phi\left(\mathbf{w}, \xi\right) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{\ell}\sum_{i=1}^{\ell}\xi_i \\ \textit{subject to} & y_i\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right) \geq 1 - \xi_i \\ \textit{and} & \xi_i \geq 0. \end{array} \tag{4.30}$$

where the parameter C determines the effect of the slack variables. The term $\frac{C}{\ell}\sum_{i=1}^{\ell}\xi_i$ is included to ensure that the slack variables are kept small. It is suggested in [29] that the $C > 0$ should be selected based on the performance of the classifier on a validation set or through using cross validation. This primal problem can be written in the standard form as

$$\begin{array}{ll} \textit{minimise} & \phi\left(\mathbf{w}, \xi\right) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{\ell}\sum_{i=1}^{\ell}\xi_i \\ \textit{subject to} & -y_i\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right) + 1 - \xi_i \leq 0 \\ \textit{and} & -\xi_i \leq 0. \end{array} \tag{4.31}$$

The Lagrangian for the above problem can then be written as

$$L\left(\mathbf{w}, b, \alpha\right) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{\ell}\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\alpha_i\left[\left(y_i\left[\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right]\right) - 1\right] - \sum_{i=1}^{\ell}\alpha_i\xi_i - \sum_{i=1}^{\ell}\lambda_i\xi_i \tag{4.32}$$

The KKT conditions for the non-separable case (eq. 4.17, 4.18) stay exactly the same. The new conditions for the slack variables are

$$\frac{\partial L\left(\mathbf{w}, \boldsymbol{\xi}\right)}{\partial \xi_i} = 0, \;\; i = 1, 2, \ldots, \ell. \tag{4.33}$$

This gives the expression

$$\frac{\partial L\left(\mathbf{w}, \boldsymbol{\xi}\right)}{\partial \xi_i} = \frac{C}{\ell} - \alpha_i - \lambda_i = 0 \tag{4.34}$$

$$\lambda_i = \frac{C}{\ell} - \alpha_i. \tag{4.35}$$

Using the above results in

$$\frac{C}{\ell}\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\alpha_i\xi_i - \sum_{i=1}^{\ell}\lambda_i\xi_i = \sum_{i=1}^{\ell}\xi_i\left(\frac{C}{\ell} - \alpha_i - \lambda_i\right) \tag{4.36}$$

$$= 0. \tag{4.37}$$

The dual for this problem is therefore exactly the same as for the separable case (equation 4.26) except for the additional constraint of $\lambda_i \geq 0$. This constraint can be rewritten (by substituting equation 4.35) as

$$\frac{C}{\ell} - \alpha_i \geq 0 \tag{4.38}$$

$$\alpha_i \leq \frac{C}{\ell}. \tag{4.39}$$

The dual problem can therefore be written as

$$\begin{aligned} &\textit{maximise} && W(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \langle \mathbf{x_i}, \mathbf{x_j}\rangle \\ &\textit{subject to} && 0 \leq \alpha_i \leq \tfrac{C}{\ell}, \quad i = 1, 2, \ldots, l \\ &\textit{and} && \sum_{i=1}^{l}\alpha_i y_i = 0. \end{aligned} \tag{4.40}$$

The only difference between this problem and the separable problem (equation 4.26) is the upper constraint $\frac{C}{\ell}$ on α_i.

4.3.2 ν-SVM formulation

The SVM can also be parameterised as follows

$$\begin{aligned} &\textit{minimise} && \tau(\mathbf{w}, \boldsymbol{\xi}, \rho) = \tfrac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \tfrac{1}{\ell}\sum_{i=1}^{\ell}\xi_i \\ &\textit{subject to} && y_i\left(\langle \mathbf{w}, \mathbf{x_i}\rangle + b\right) \geq \rho - \xi_i \\ & && \xi_i \geq 0, \rho \geq 0 \end{aligned} \tag{4.41}$$

with the ν parameter selected beforehand [37]. The ν parameter controls the number of margin errors. If the classes are linearly separable the margin of the hyperplane will be $\rho/\|\mathbf{w}\|$. The

Lagrangian can be written as

$$L\left(\mathbf{w}, \boldsymbol{\xi}, \rho, b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \delta\right) = \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{\ell}\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\alpha_i\left(y_i(\langle\mathbf{w}, \mathbf{x_i}\rangle + b) - \rho + \xi_i\right) - \sum_{i=1}^{\ell}\beta_i\xi_i - \delta\rho. \tag{4.42}$$

Applying the KKT conditions to this problem results in

$$\mathbf{w} = \sum_{i=1}^{\ell}\alpha_i y_i \mathbf{x}_i \tag{4.43}$$

$$\sum_{i=1}^{\ell}\alpha_i y_i = 0 \tag{4.44}$$

$$\nu = \sum_{i=1}^{\ell}\alpha_i - \delta \tag{4.45}$$

$$\alpha_i + \beta_i = \frac{1}{\ell}. \tag{4.46}$$

When these values are substituted into the Lagrangian it results in

$$L\left(\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta\right) = -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j y_i y_j \langle\mathbf{x}_i, \mathbf{x}_j\rangle. \tag{4.47}$$

The Lagrangian is only dependent on the dual variables (and only uses inner products of the pattern vectors). The inequality constraints in the primal problem result in the following constraints in the dual problem

$$\alpha_i \geq 0 \tag{4.48}$$

$$\beta_i \geq 0 \tag{4.49}$$

$$\delta \geq 0. \tag{4.50}$$

Substituting 4.45 and 4.46 in the above results in the following constraints

$$0 \leq \alpha_i \leq \frac{1}{\ell} \tag{4.51}$$

$$\sum_{i=1}^{\ell}\alpha_i \geq \nu. \tag{4.52}$$

Writing the dual problem, taking into account the above constraints (and 4.44) gives

$$\begin{aligned} &\textit{maximise} && -\tfrac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i}, \mathbf{x_j} \rangle \\ &\textit{subject to} && 0 \leq \alpha_i \leq \tfrac{1}{\ell} \\ & && \textstyle\sum_i \alpha_i \geq \nu \\ & && \textstyle\sum_i \alpha_i y_i = 0 \end{aligned} \tag{4.53}$$

The advantage of the ν-SVM problem over C-SVM is that it is more intuitive to select the ν parameter. A theorem is proved in [37] that connects the ν parameter to the amount of margin errors and number of support vectors. It is suggested in [2] that the ν parameter should be set to the expected error rate. There is also a connection between the ν-SVM and C-SVM in that should ν-SVM training lead to $\rho > 0$, then the C-SVM training with $C = 1/\rho$ will lead to the same decision function (proved in [37]).

4.4 KERNELS

In the preceding discussion the classification was performed by a hyperplane. Many practical problems are not linearly separable. These patterns can be mapped to a high-dimensional space where they are linearly separable. This can be performed by a mapping $\boldsymbol{\phi} : \mathbf{x} \mapsto \mathbf{z}$ [38]. Since the discriminant function (equation 4.27) and optimisation problems (eqs 4.40 and 4.53) are all in terms of inner products, only $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle$ needs to be calculated. Under certain conditions the inner product can be calculated in a simple kernel function

$$k(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle. \tag{4.54}$$

This is known as the "kernel trick" (introduced in [39]). For a function $k(\mathbf{x}, \mathbf{x}')$ to be a kernel function it should fulfil certain conditions known as Mercer's conditions (see [2], [29] or [38]).

The above can be made clear by using the classic XOR classification problem (introduced in [40]). This problem is illustrated in figure 4.2. The classification problem on the left is not separable by a plane. When the mapping

$$\mathbf{z} = \boldsymbol{\phi}(\mathbf{x}) \tag{4.55}$$

$$= \left([\mathbf{x}]_1^2, \sqrt{2}[\mathbf{x}]_1[\mathbf{x}]_2, [\mathbf{x}]_2^2 \right) \tag{4.56}$$

is used the problem becomes separable. The dot product of $\mathbf{z}$ and $\mathbf{z}'$ is calculated below

$$\begin{aligned}(\mathbf{z}\cdot\mathbf{z}') &= \boldsymbol{\phi}(\mathbf{x},)\cdot\boldsymbol{\phi}(\mathbf{x}') && (4.57)\\ &= [\mathbf{x}]_1^2[\mathbf{x}']_1^2 + 2[\mathbf{x}]_1[\mathbf{x}]_2[\mathbf{x}']_1[\mathbf{x}']_2 + [\mathbf{x}]_2^2[\mathbf{x}']_2^2 && (4.58)\\ &= \left([\mathbf{x}]_1[\mathbf{x}']_1 + [\mathbf{x}]_2[\mathbf{x}']_2\right)^2 && (4.59)\\ &= (\mathbf{x}\cdot\mathbf{x}')^2. && (4.60)\end{aligned}$$

The result is a simple kernel function $k(\mathbf{x},\mathbf{x}') = \langle\mathbf{x},\mathbf{x}'\rangle^2$. As can be seen from figure 4.2, the problem has become linearly separable. The problem can be implicitly mapped by using the kernel function (there is no need to calculate the higher dimension representation). Several types of these kernel functions are discussed in the next section.

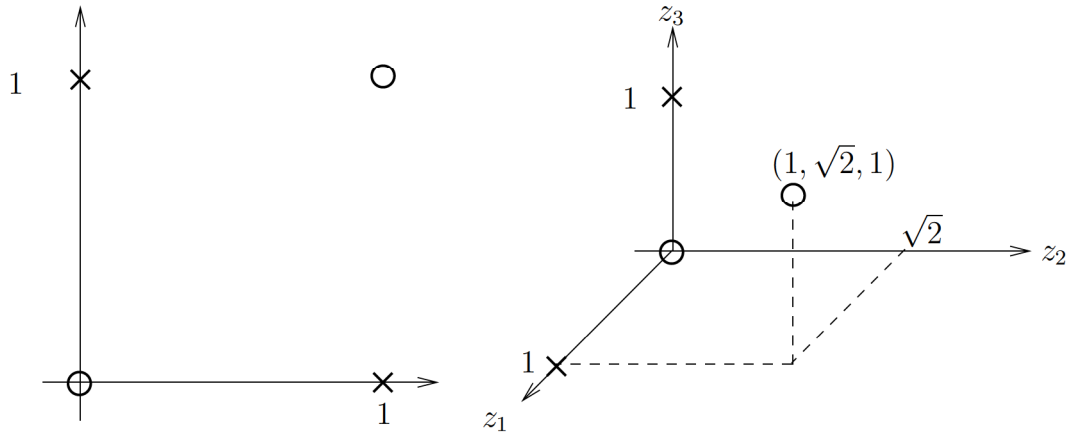


FIGURE 4.2: A simple binary classification problem on the left with a mapping to a space where the problem is separable (using equation 4.56).

4.4.1 Kernel types and rules for constructing new kernels

The following positive definite kernels are amongst those commonly used and included in most SVM toolkits (e.g. SVMlight [41], libSVM [42], Universvm [1], Shogun [43]) [2]:

1. *Polynomial kernel* $k(\mathbf{x},\mathbf{x}') = \langle\mathbf{x},\mathbf{x}'\rangle^d$ (d is the order of the polynomial).
2. *Inhomogeneous polynomial kernel* $k(\mathbf{x},\mathbf{x}') = (1+\langle\mathbf{x},\mathbf{x}'\rangle)^d$.
3. *Gaussian radial basis function (RBF) kernel* $k(\mathbf{x},\mathbf{x}') = \exp\left(\frac{-\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$.

The Gaussian kernel is sometimes specified as

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right), \tag{4.61}$$

where $\gamma = 1/2\sigma^2$. The width of the Gaussian needs to be specified beforehand and is often problem specific. When the Gaussian kernel is used (in section 7.3, for example) σ is calculated by minimising the error using the training set and a strategy similar to cross-validation.

4.4.1.1 Creating new kernels

New kernels can be constructed from other existing kernels. If k_1 and k_2 are kernels, then the following are also kernels [2, 29]:

1. *Sums of kernels* $k_3(\mathbf{x}, \mathbf{x}') = \alpha_1 k_1(\mathbf{x}, \mathbf{x}') + \alpha_2 k_2(\mathbf{x}, \mathbf{x}'), \alpha_1, \alpha_2 > 0$
2. *Pointwise products* $k_3(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')\, k_2(\mathbf{x}, \mathbf{x}')$.

A larger set of rules can be found in [2] and [29]. Kernels represent prior information about the problem, therefore the choice of a kernel is important. Another approach to incorporate prior information in the classifier, is to explicitly map the pattern to a feature space (e.g. using the wavelet packet transform) – as opposed to the implicit mapping by the kernel function. The Universum also presents a new approach in which prior information can be incorporated by adding patterns of related classes (this is discussed in chapter 9).

4.5 TRAINING SUPPORT VECTOR MACHINES

The SVM problems in equations 4.40 and 4.53 are constrained convex optimisation problems. Several algorithms were investigated to perform the optimisation problems. Osuna et. al. [44], introduced a strategy in which the C-SVM problem is decomposed into smaller subproblems which are then solved iteratively. The sequentional minimal optimisation (SMO) algorithm used the extreme case where the size of the subproblems were only two parameters [45]. The advantage of two parameter subproblems are that they can be solved analytically (i.e. without resorting to numerical methods). The optimisation for the C-SVM case is discussed in the next section.

4.5.1 Calculation of optimal solution for the subproblem

The two parameter subproblem for the α_a and α_b multipliers can be written as

$$\begin{aligned} W(\alpha_a, \alpha_b) =& \alpha_a + \alpha_b - \frac{1}{2}\alpha_a^2 K_{aa} - \alpha_a \alpha_b K_{ab} y_a y_b \\ & - \frac{1}{2}\alpha_b^2 K_{bb} - \alpha_b y_b v_b - \alpha_b y_b v_b + W_{const} \end{aligned} \tag{4.62}$$

where the W_{const} value is constant with respect to α_a and α_b. v_i is defined as

$$v_a = \sum_{j=1, j\neq a, j\neq b}^{\ell} \alpha_j y_j K_{aj}. \tag{4.63}$$

The values α_a^{old} and α_b^{old} refers to the unoptimised values where α_a and α_b refers to the new values. Using the equality constraint $\sum_{i=1}^{\ell} \alpha_i y_i = 0$, α_a can be expressed as

$$\alpha_a = \alpha_a^{old} + y_a y_b \left(\alpha_b^{old} - \alpha_b\right). \tag{4.64}$$

The subproblem can be minimised by setting the derivative to zero. By substituting the above into this, the minimum of the subproblem is [46]

$$\alpha_b = \alpha_b^{old} + \frac{y_b (E_a - E_b)}{K_{aa} - 2K_{ab} + K_{bb}}. \tag{4.65}$$

This equation is written in terms of the error E_i defined as

$$E_i = g(\mathbf{x}_i) - y_i \tag{4.66}$$

$$= \sum_{j=1}^{\ell} \alpha_j^{old} y_j K_{ij} + b_{old} - y_i. \tag{4.67}$$

The advantage of this approach is that the error E_i can be updated without re-evaluating the whole expression when the two parameters are changed. This can be expressed as

$$E_i^{new} = E_i^{old} - \alpha_a^{old} y_a^{old} K_{ia} - \alpha_b^{old} y_b^{old} K_{ib} - b_{old} + \alpha_a y_a K_{ia} + \alpha_b y_b K_{ib} + b. \tag{4.68}$$

The constraints on the multipliers should still be handled. The constraints are $0 \leq \alpha_i \leq \frac{C}{\ell}$ (see equation 4.64). The two cases where $y_a \neq y_b$ and $y_a = y_b$ are dealt with separately.

Case 1: $y_a \neq y_b$

For this case equation 4.64 reduces to

$$\alpha_a = \alpha_a^{old} - \alpha_b^{old} + \alpha_b. \tag{4.69}$$

The box constraints on α_1 therefore become

$$0 \leq \alpha_a \leq \frac{C}{\ell} \tag{4.70}$$

$$0 \leq \alpha_a^{old} - \alpha_b^{old} + \alpha_b \leq \frac{C}{\ell} \tag{4.71}$$

$$\alpha_b^{old} - \alpha_a^{old} \leq \alpha_b \leq \frac{C}{\ell} + \alpha_b^{old} - \alpha_a^{old}. \tag{4.72}$$

If α_b is constrained to

$$L \leq \alpha_b \leq H \tag{4.73}$$

then $L = \max\left(0, \alpha_b^{old} - \alpha_a^{old}\right)$ and $H = \min\left(\frac{C}{\ell}, \frac{C}{\ell} + \alpha_b^{old} - \alpha_a^{old}\right)$.

Case 1: $y_a = y_b$

In the same manner as above (with $\alpha_a = \alpha_a^{old} + \alpha_b^{old} - \alpha_b$) the constraints become

$$\alpha_a^{old} + \alpha_b^{old} - \frac{C}{\ell} \leq \alpha_b \leq \alpha_a^{old} + \alpha_b^{old}. \tag{4.74}$$

Using the same strategy as above, $L = \max\left(0, \alpha_a^{old} + \alpha_b^{old} - \frac{C}{\ell}\right)$ and $H = \min\left(\frac{C}{\ell}, \alpha_a^{old} + \alpha_b^{old}\right)$.

The selection of the working set (i.e. the two parameters to be optimised) can dramatically increase the training speed. A good working set selection algorithm, for which the convergence was proven, is given in [47]. The algorithm explained above can also be developed for the ν-SVM case. This was done in [48] and [47].

4.5.2 Implementation notes

The above algorithm was implemented with the original SMO working set selection algorithm for C-SVM. For the ν-SVM case, the libSVM library was used (described in [42]). This library

implements the SMO algorithm, discussed above, for the ν-SVM case. The Shogun library was also used for some SVM and multiple kernel learning experiments [43].

CHAPTER FIVE

SHIFT INVARIANT QUADRATIC TIME-FREQUENCY REPRESENTATIONS (COHEN'S CLASS)

5.1 INTRODUCTION

Two representations of a signal often used in signal processing is the instantaneous energy and energy-density spectrum. The instantaneous energy for a signal $x(t)$ is $|x(t)|^2$ and the energy-density spectrum is $\frac{1}{2\pi}|\hat{x}(\omega)|^2$. The total signal energy can be calculated as

$$E_x = \int_{-\infty}^{\infty} |x(t)|^2 \, dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |\hat{x}(\omega)|^2 \, d\omega. \tag{5.1}$$

This representation is not useful in all instances since the frequency content of many practical signals vary with time. To illustrate the inadequacy of these methods, consider a simple linear chirp (see section 10.1.2 for a definition). The signal's frequency changes with time – so both the instantaneous energy of a signal and the energy density spectrum is of limited utility. This gives rise to the idea of an energy density that is a joint function of the time and frequency [49]. This joint time and frequency function is referred to as the time-frequency representation (TFR) of the signal. This is denoted as $Pf(u, \xi)$ where u is the time and ξ is the frequency.

5.2 PROPERTIES OF DISTRIBUTIONS

Several properties of a time-frequency representation may be required so that it corresponds with the properties that are expected of an energy distribution. This section discusses those properties to ensure that only quadratic time-frequency representations that fulfil these properties (at least in part) are used. The properties are listed below [19, 49, 50].

1. *Invariance to phase shifts.* The distribution should be invariant to phase shifts.

$$f(t) = e^{j\theta}g(t) \Leftrightarrow Pf(u,\xi) = Pg(u,\xi) \tag{5.2}$$

2. *Time-shift invariant.* The distribution should be invariant to shifts in time. If the input signal is $f(t)$ then the following should hold:

$$f(t) = g(t-u_0) \Leftrightarrow Pf(u,\xi) = Pg(u-u_0,\xi). \tag{5.3}$$

3. *Frequency-shift invariant.* The distribution should be invariant to shifts in input frequency:

$$f(t) = g(t)e^{-j\xi_0 t} \Leftrightarrow Pf(u,\xi) = Pg(u,\xi-\xi_0). \tag{5.4}$$

4. *Time and frequency marginal properties.* The most important property for a TFR is that the time and frequency marginal properties should hold:

$$\int_{-\infty}^{\infty} Pf(u,\xi)du = \frac{1}{2\pi}\left|\hat{f}(\xi)\right|^2 \tag{5.5}$$

and

$$\int_{-\infty}^{\infty} Pf(u,\xi)d\xi = |f(u)|^2. \tag{5.6}$$

5. *Real valued energy density spectrum.* The energy density spectrum should be real valued. In other words $Pf(u,\xi) = [Pf(u,\xi)]^*$ where $[]^*$ is the complex conjugate.

6. *Scaling in time.* A scaling in time should also cause an appropriate scaling in frequency

parameters (as the short-time Fourier transform (STFT) would), i.e.

$$f(t) = \frac{1}{\sqrt{s}} g\left(\frac{t}{s}\right) \Leftrightarrow Pvf(u,\xi) = Pg\left(\frac{u}{s}, s\xi\right) \tag{5.7}$$

7. *Positive valued.* The distribution should preferably be positive:

$$Pf(u,\xi) \geq 0. \tag{5.8}$$

Wigner proved that there is no quadratic time-frequency distribution that satisfies the marginal conditions (properties 4) and is positive valued (property 7) [19].

5.3 WIGNER-VILLE DISTRIBUTION

The Wigner-Ville distribution (WVD) is the most well known of the Cohen's class TFRs (discussed in the next section). The Wigner-Ville distribution is defined as[1]

$$P_V(u,\xi) = \int_{-\infty}^{\infty} x(u+\frac{\tau}{2}) x^*(u-\frac{\tau}{2}) e^{-j\tau\xi} d\tau \tag{5.9}$$

where u is the time and ξ is the angular frequency. The WVD is both time and frequency shift invariant (shown in next section). It also satisfies the time and frequency marginal properties [49] and is real valued. The latter can be proved by

$$[P_V(u,\xi)]^* = \left[\int_{-\infty}^{\infty} x(u+\frac{\tau}{2}) x^*(u-\frac{\tau}{2}) e^{-j\tau\xi} d\tau\right]^* \tag{5.10}$$

$$= \int_{-\infty}^{\infty} x^*(u+\frac{\tau}{2}) x(u-\frac{\tau}{2}) e^{j\tau\xi} d\tau. \tag{5.11}$$

Making the substitution $\nu = -\tau$ yields

$$[P_V(u,\xi)]^* = \int_{-\infty}^{\infty} x^*\left(u-\frac{v}{2}\right) x\left(u+\frac{\nu}{2}\right) e^{-j\nu\xi} d\nu \tag{5.12}$$

$$= P_V(u,\xi). \tag{5.13}$$

The WVD is however not positive valued (property 7). Because of this, it is sometimes referred to as a quasi-distribution.

[1] The notation of the WVD and the Cohen's class TFR differs from the standard notation in order to keep it consistent with the next chapter.

Another interesting property of the WVD is that it does not spread the time or frequency support of diracs or sines [19]. This can be shown by

$$f(t) = \delta(u - u_0) \quad \Leftrightarrow P_V f(u, \xi) = \delta(u - u_0) \tag{5.14}$$

$$f(t) = exp(i\xi_0 t) \quad \Leftrightarrow P_V f(u, \xi) = \frac{1}{2\pi}\delta(\xi - \xi_0). \tag{5.15}$$

5.3.1 Cross components

A disadvantage of the Wigner-Ville distribution (and quadratic TFRs in general) is that it creates cross-components. As an example, consider the signal

$$x(t) = x_a(t) + x_b(t). \tag{5.16}$$

The WVD of this is

$$\begin{aligned} P_V x(u,\xi) = &\int_{-\infty}^{\infty} x_a(u + \frac{\tau}{2}) x_a^*(u - \frac{\tau}{2}) e^{-j\tau\xi} d\tau \\ &+ \int_{-\infty}^{\infty} \left[x_a(u + \frac{\tau}{2}) x_b^*(u - \frac{\tau}{2}) + x_b(u + \frac{\tau}{2}) x_a^*(u - \frac{\tau}{2}) \right] e^{-j\tau\xi} d\tau \\ &+ \int_{-\infty}^{\infty} x_b(u + \frac{\tau}{2}) x_b^*(u - \frac{\tau}{2}) e^{-j\tau\xi} d\tau. \end{aligned} \tag{5.17}$$

Collecting the terms above results in

$$\begin{aligned} P_V x(u,\xi) = &\underbrace{P_V x_a(u,\xi) + P_V x_b(u,\xi)}_{Auto-components} \\ &+ \underbrace{\int_{-\infty}^{\infty} \left[x_a(u + \frac{\tau}{2}) x_b^*(u - \frac{\tau}{2}) + x_b(u + \frac{\tau}{2}) x_a^*(u - \frac{\tau}{2}) \right] e^{-j\tau\xi} d\tau}_{Cross-components} \end{aligned} \tag{5.18}$$

The auto-components are the normal WVD of the different components of the signal. The cross-components are interference terms caused by the quadratic nature of the WVD.

As an example, consider a signal composed of two linear chirps both and 0Hz and ending at 0.3 and 0.4Hz respectively. The WVD of the chirps are given in figure 5.1. As can be seen from figure 5.1 (c), the sum of the two chirps creates cross components. Only the cross components ($P_V x(u,\xi) - P_V x_a(u,\xi) - P_V x_b(u,\xi)$) are given in figure 5.1 (d).

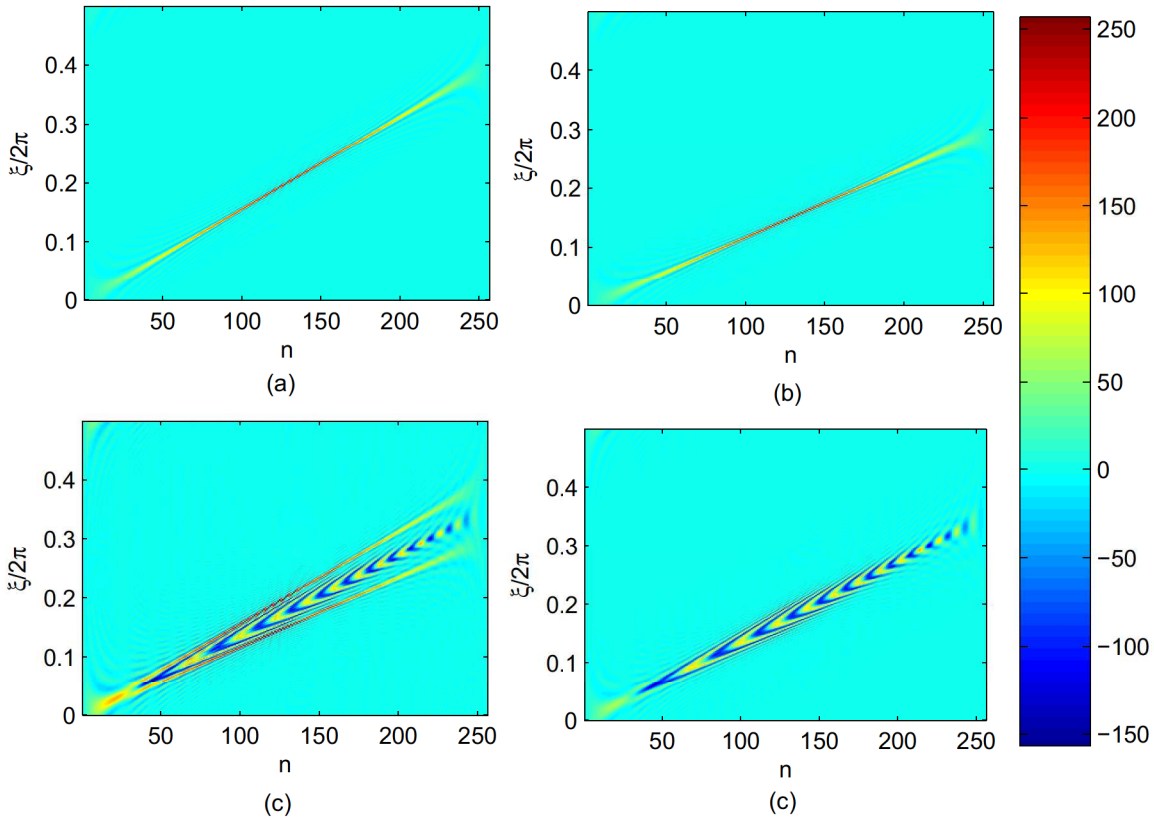


FIGURE 5.1: Wigner-Ville distribution of the sum of two linear chirps. (a), (b) Two linear chirps starting at 0 Hz and ending at 0.3 and 0.4 Hz. (c) Wigner-Ville transform of the sum. (d) Cross-terms that appear in (c)

5.4 COHEN'S CLASS DISTRIBUTIONS

The Cohen's class of quadratic time-frequency distributions can be expressed as [51]

$$P_C^{\phi}(u,\xi) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \phi_{d-D}(\tau,\eta)\, \mathcal{A}_x(\tau,\eta) e^{j(\eta u - \xi\tau)} d\eta d\tau. \tag{5.19}$$

The $\phi_{d-D}(\tau,\eta)$ function is the kernel function that determines the properties of the distribution. $\mathcal{A}_x(\tau,\eta)$ is the symmetric ambiguity function and is defined as

$$\mathcal{A}_x(\tau,\eta) = \int_{-\infty}^{\infty} x\left(t+\frac{\tau}{2}\right) x^*\left(t-\frac{\tau}{2}\right) e^{-j\eta t} dt. \tag{5.20}$$

An equivalent representation for the Cohen's class TFR is as a smoothing of the Wigner-Ville distribution. This can be expressed as [51]

$$P_{CE}^{\Psi}(u,\xi) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \phi_{u-\xi}(u-u',\xi-\xi')\, P_V(u',\xi') du' d\xi'. \tag{5.21}$$

The function $\phi_{u-\xi}$ can be obtained from ϕ_{d-D} by a Fourier transform and Fourier inversion (see for instance [51]). Cohen's class of distributions are also known as shift-invariant distributions because they are invariant to shifts in time and frequency (properties 2 and 3 in section 5.2). The other properties in section 5.2 are fulfilled for certain kernel functions. For the TFR to be real valued (property 5), the kernel function must fulfil [50]

$$\phi_{d-D}(\tau,\eta) = \phi_{d-D}^*(-\tau,-\eta). \tag{5.22}$$

For the time and frequency marginal properties (property 4 in section 5.2), the kernel function should fulfil

$$\phi_{d-D}(0,\eta) = 1 \tag{5.23}$$

and

$$\phi_{d-D}(\tau,0) = 1 \tag{5.24}$$

respectively.

5.4.1 Specific kernel functions

The Wigner-Ville distribution is a Cohen's class distribution with a kernel function $\phi_{d-D}(\tau, \eta) = 1$.

5.4.1.1 Radially Gaussian Kernel

Consider the definition of the a Cohen's class TFR in equation 5.19. The kernel function completely determines the properties of the TFR. The cross and auto-components occur in the ambiguity function (since it is a quadratic function). As an example of this, see figure 5.2 which is the ambiguity function for the sum of two linear chirps (from 0.2 to 0 Hz and 0.4 to 0.3 Hz). The kernel function determines which of the cross components are passed to the TFR.

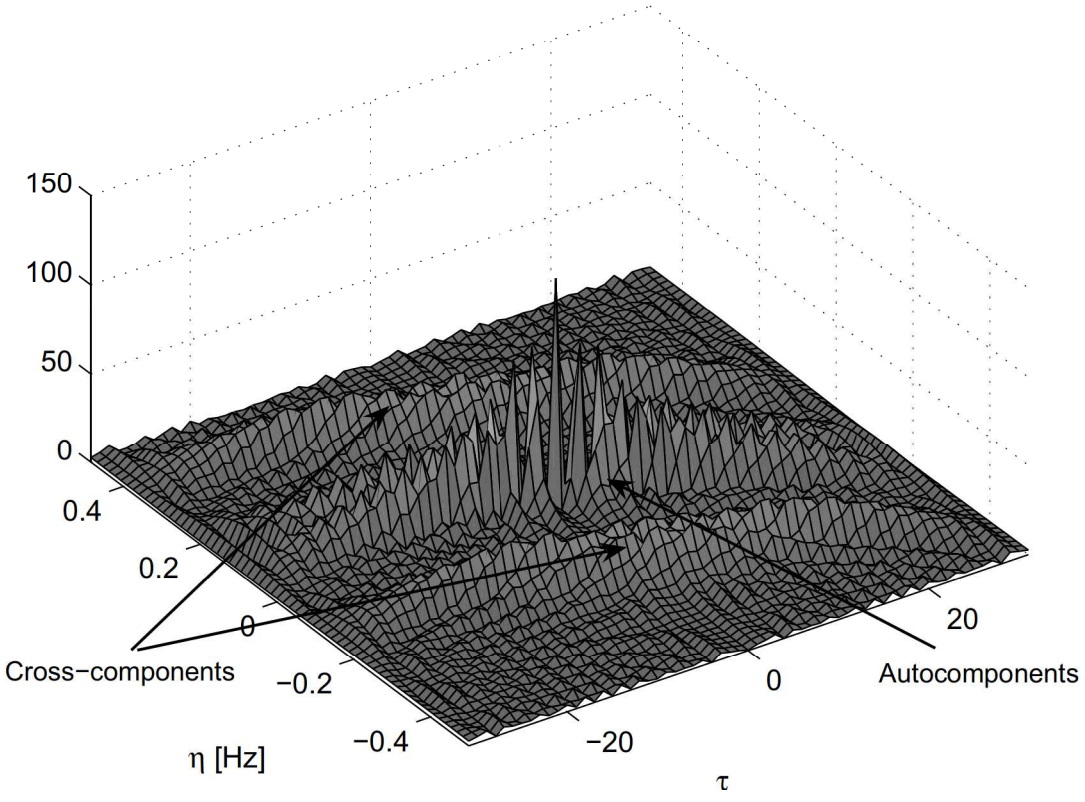


FIGURE 5.2: The ambiguity function of the sum of two linear chirps. The auto- and cross-components are indicated.

Since the auto-components occur at the origin, the kernel function can be viewed as a two dimensional low pass filter.

The goal of a Cohen's class TFR is to achieve good cross-component suppression and auto-component concentration [9]. It is not possible to find a general kernel that will achieve this because there will always be signals with either significant auto-components in the stop band or cross-components in the pass band [9]. It is suggested by Baraniuk and Jones [9] that a Gaussian be used to suppress the cross-components. The radially Gaussian kernel is defined as [9]

$$\phi(\tau,\eta) = e^{\frac{-\left(\tau^2+\eta^2\right)}{2\sigma(\theta)^2}} \tag{5.25}$$

where $\theta = \arctan\left(\frac{\eta}{\tau}\right)$. The function $\sigma(\theta)$ is the spread function (or contour function) that completely defines the radially Gaussian kernel. As stated above, the optimal kernel function is signal specific. Since the kernel function for the radially Gaussian kernel only depends on the spread function, the search for the optimal kernel is equivalent to the search for the optimal spread function. To ensure that the resulting distribution is real, the kernel function must satisfy $\phi(u,\xi) = [\phi(-u,-\xi)]^*$. This implies that the spread function should be π-periodic. Another desirable property for the spread function is to be smooth (continuous and differentiable) [9]. In [10] it is suggested that a truncated Fourier series should be used as a spread function (since this function is both π-periodic and continuous). As an alternative to this, a truncated Bernstein series is developed in Chapter 7. This function has interesting properties that makes optimisation more efficient. Criteria for optimising the spread function is also discussed in that chapter.

CHAPTER SIX

WAVELET TRANSFORMS

This chapter presents a brief overview of wavelet transforms. Since the field is large, only the areas that are directly applicable to the classifiers that were developed in chapter 8 and chapter 9 are discussed. A large part of the theoretical considerations are omitted – several other sources such as the book and paper by Daubechies [20, 52] or the book and paper by Mallat [19, 53] have a thorough discussion of these and other aspects.

The first section discusses the continuous wavelet transform. The discrete wavelet transform (DWT) is developed as a multi-resolution approximation of this in section 6.2. The wavelet packet transform that is derived as an extension of the DWT is discussed in section 6.3. Section 6.4 discusses the properties of wavelets and section 6.5 discusses several wavelets. The filters for the Battle-Lemarié wavelets are calculated in this section.

6.1 CONTINUOUS WAVELET TRANSFORM

The wavelet transform (like the short-time Fourier transform) correlates a signal with a dictionary of waveforms that are concentrated in both time and frequency. The continuous wavelet transform can be defined as

$$Wf(u,s) = \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{s}} \psi^* \left(\frac{t-u}{s} \right) dt, \tag{6.1}$$

where the function $\psi(t)$ is known as the mother wavelet. The mother wavelet has a mean of zero

$$\int_{-\infty}^{\infty} \psi(t)\, dt = 0, \tag{6.2}$$

is centred around $t = 0$ and normalized[1] $\|\psi\| = 1$ [20]. This can be written more concisely as an inner product

$$Wf(u,s) = \langle f, \psi_{u,s} \rangle, \tag{6.3}$$

where the inner product in $L^2(\mathbb{R})$ (square integrable functions) is defined as

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(t)\, g^*(t)\, dt \tag{6.4}$$

and $\psi_{u,s}(t)$ is defined as

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-u}{s}\right). \tag{6.5}$$

6.1.1 Effect of the scaling parameter

The effect of the scaling parameter is to dilate the mother wavelet in time. It is therefore intuitively clear that it will decrease the frequency of the wavelet. If we let

$$\psi_s(t) = \psi\left(\frac{t}{s}\right) \tag{6.6}$$

then its Fourier transform is

$$\hat{\psi}_s(\omega) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{s}} \psi\left(\frac{t}{s}\right) e^{-j\omega t} dt. \tag{6.7}$$

Making the substitution $v = \frac{t}{s}$ results in

$$\hat{\psi}_s(\omega) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{s}} \psi(v)\, e^{-j\omega v s} s.dv \tag{6.8}$$

$$= \sqrt{s}\hat{\psi}(s\omega). \tag{6.9}$$

[1] The $L^2(\mathbb{R})$ norm is defined as $\|f\|^2 = \langle f, f \rangle = \int_{-\infty}^{\infty} |f(\iota)|^2\, d\iota$

An increase in the scaling parameter therefore decreases the frequency. A real wavelet transform maintains energy conservation if the mother wavelet satisfies an admissibility condition [19–21]

$$0 < \int_0^\infty \frac{\left|\hat{\psi}(\omega)\right|^2}{\omega} d\omega < \infty. \tag{6.10}$$

The admissibility condition ensures that the signal can be reconstructed from its continuous wavelet transform.

6.1.1.1 Wavelet example

The above is illustrated with the Mexican hat wavelet. The Mexican hat wavelet is defined as [20]

$$\psi(t) = \frac{2}{\sqrt{3}\sqrt[4]{\pi}}(1-t^2)e^{-t^2/2}. \tag{6.11}$$

The Fourier transform for this is calculated as (source given in appendix B.2)

$$\hat{\psi}(\omega) = \frac{2(1-\omega^2)e^{-\omega^2/2}}{(\sqrt{3}\pi^{1/4})}. \tag{6.12}$$

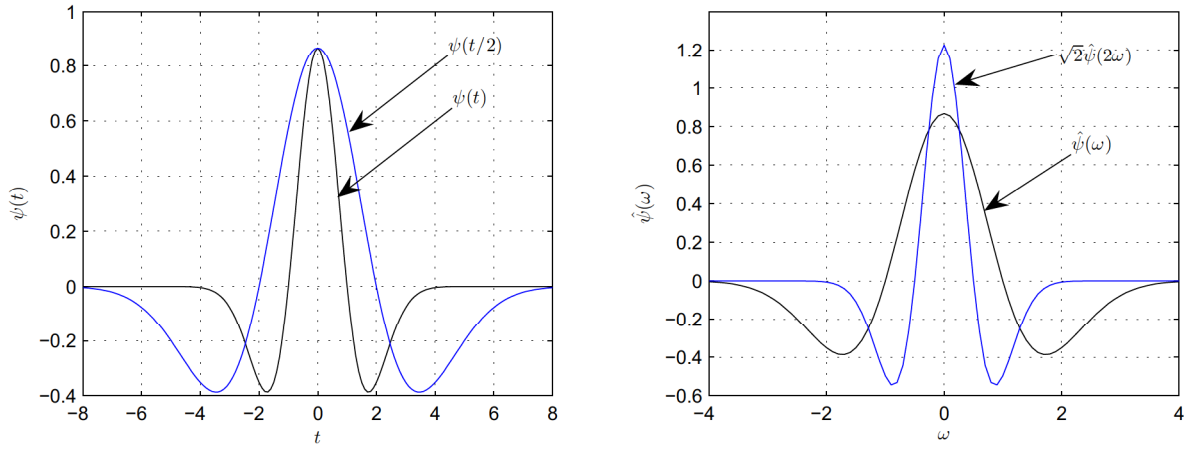
The above function's Fourier transform is the same as the function itself. The example of a Mexican hat function and a Mexican hat function dilated by 2 is given in figure 6.1. As can be seen from the figure, the dilated function is narrower in frequency.

6.1.2 Time-frequency resolution

The time and frequency localisation of a wavelet is limited by the Heisenberg uncertainty theorem [19]. Suppose that σ^2_{t-m} is the spread in time for the mother wavelet and $\sigma^2_{\omega-m}$ is the spread in frequency for the mother wavelet (The time and frequency spread is defined as the variance of the wavelet in time and frequency [19]). The spread for $\psi_{u,s}$ in time and frequency is then

$$\sigma_t^2 = s^2\sigma^2_{t-m} \tag{6.13}$$

$$\sigma_\omega^2 = \frac{\sigma^2_{\omega-m}}{s^2}. \tag{6.14}$$



(a) The Mexican hat function and a dilated Mexican hat function.

(b) The Fourier transform of the Mexican hat function and the dilated function.

FIGURE 6.1: A Mexican hat function and a dilated Mexican hat function.

For an increasing value of s the spread of $\psi_{u,s}$ will be narrower in time and wider in frequency. The Heisenberg uncertainty theorem states that the area of the Heisenberg box is [19]

$$\sigma_t \sigma_\omega \geq \frac{1}{2}. \tag{6.15}$$

This limits the time and frequency resolution of the signal. The better the signal is resolved in time, the worse it is in frequency (and vice versa). The area of the Heisenberg box for $\psi_{u,s}$ is constant $\sigma_t \sigma_\omega$ but the frequency and time resolution differs. The box is longer in time and narrower in frequency at low frequencies, which means that the frequency resolution is good, but the time resolution is bad. At high frequencies it is the exact opposite – the box is narrow in time and wide in frequency (this means that the time-resolution will be good at high frequencies).

This is advantageous in situations where transient signals are analysed (transient signals generally contain high frequency components of a short duration). The short-time Fourier transform has constant-sized boxes across all frequencies.

6.1.3 Scaling function

An important property of the continuous wavelet transform is that the function $f(t)$ can be reconstructed from $Wf(u,s)$. Since the focus is to find a basis for which the signal is separable

(for classification purposes), reconstruction is not discussed in depth – for details see [19] and [21].

If $Wf(u,s)$ is known only for $s < s_0$ the signal can be reconstructed with the aid of a scaling function that is an aggregation of scales larger than unity (i.e. low frequencies). The frequency response for the scaling function is defined as [19]

$$\left|\hat{\phi}(\omega)\right|^2 = \int_1^\infty \left|\hat{\psi}(s\omega)\right|^2 \frac{ds}{s} \tag{6.16}$$

with an arbitrary complex phase. The low frequency approximation is therefore [19]

$$Lf(u,s) = \left\langle f(t), \frac{1}{\sqrt{s}}\phi\left(\frac{t-u}{s}\right)\right\rangle. \tag{6.17}$$

6.1.4 Energy density of the continuous wavelet transform

The centre frequency of the mother wavelet is [19][2]

$$\eta = \frac{1}{2\pi}\int_0^\infty \omega \left|\hat{\psi}(\omega)\right|^2 d\omega. \tag{6.18}$$

The Fourier transform of $\psi_{u,s}(t)$ in terms of the mother wavelet is

$$\hat{\psi}_{u,s}(\omega) = \sqrt{s}\hat{\psi}(s\omega)\exp(-j\omega u). \tag{6.19}$$

The centre frequency of this is therefore

$$\xi = \frac{1}{2\pi}\int_0^\infty \omega \left|\sqrt{s}\right|^2 \left|\hat{\psi}(s\omega)\right|^2 (1) d\omega. \tag{6.20}$$

Substituting $v = s\omega$ gives

$$\xi = \frac{1}{2\pi}\int_0^\infty \frac{1}{s} vs \left|\hat{\psi}(v)\right|^2 \frac{1}{s} dv \tag{6.21}$$

$$= \frac{\eta}{s}. \tag{6.22}$$

[2] The $\frac{1}{2\pi}$ scaling factor is caused by the energy density spectrum (Parseval's theorem). The energy density spectrum of a signal $x(t)$ is $|X(f)|^2$ or $\frac{1}{2\pi}|\hat{x}(\omega)|^2$ in angular frequency.

The local time-frequency energy density (scalogram) at $\left(u, \xi = \frac{\eta}{s}\right)$ is therefore [19]

$$P_W(u,\xi) = \left|Wf\left(u, \frac{\eta}{s}\right)\right|^2. \tag{6.23}$$

6.1.5 Example

The time and frequency spread of the Mexican hat wavelet function (defined in equation 6.11) will be calculated in this section. The function is centred around $t = 0$ in time. The time spread will therefore be (using the definition for the time-spread in [19])

$$\sigma_t^2 = \int_{-\infty}^{\infty} t\,|\psi(t)|^2\,dt \tag{6.24}$$

$$= \int_{-\infty}^{\infty} (t-0)^2 \frac{4\left(1-t^2\right)^2 e^{-t^2}}{3\sqrt{\pi}} dt \tag{6.25}$$

$$= \lim_{b\to\infty} \left. \frac{4\left(-2\left(\frac{3\sqrt{\pi}\,erf(t)}{8} - \frac{\left(2t^3+3t\right)e^{-t^2}}{4}\right) + \frac{19\sqrt{\pi}\,erf(t)}{16} - \frac{\left(4t^5+10t^3+15t\right)e^{-t^2}}{8} - \frac{te^{-t^2}}{2}\right)}{3\sqrt{\pi}} \right|_{-b}^{b} \tag{6.26}$$

$$= \frac{7}{6}. \tag{6.27}$$

In the last two steps Maxima was used (the source for this is given in appendix B.2). The centre frequency can be calculated (using 6.18) as

$$\eta = \frac{1}{2\pi}\int_0^{\infty} \omega \left|\hat{\psi}(\omega)\right|^2 d\omega \tag{6.28}$$

$$= \frac{1}{2\pi}\int_0^{\infty} \frac{\omega\left(\sqrt{2}\sqrt{\pi}\left(2\omega^2+2\right) - 2\sqrt{2}\sqrt{\pi}\right)^2 e^{-\omega^2}}{3\sqrt{\pi}} d\omega \tag{6.29}$$

$$= \lim_{b\to\infty} \left. \frac{1}{2\pi}\frac{4\sqrt{\pi}\left(-\omega^4 - 2\omega^2 - 2\right)e^{-\omega^2}}{3} \right|_0^b \tag{6.30}$$

$$= \frac{4}{3\sqrt{\pi}}. \tag{6.31}$$

The spread in frequency can be calculated as (see appendix B.2 again)

$$\sigma_\omega^2 = \frac{15\pi - 32}{12\pi}. \tag{6.32}$$

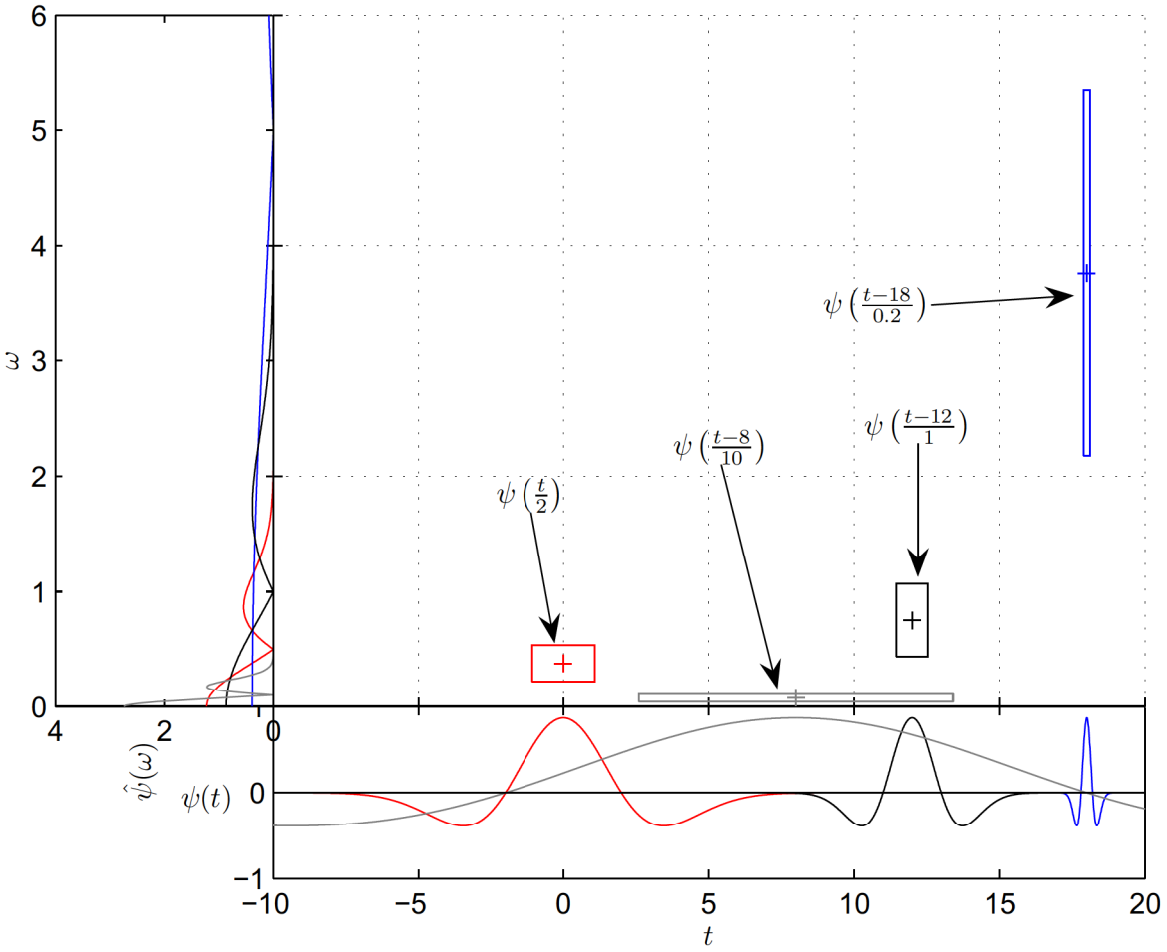


FIGURE 6.2: Example of the time and frequency spread of the Mexican hat wavelet

Therefore

$$\sigma_t \sigma_\omega = \sqrt{\frac{7}{6}} \sqrt{\frac{15\,\pi - 32}{12\,\pi}} \approx 0.6841 > \frac{1}{2} \tag{6.33}$$

An example of the above is given in figure 6.2. In this figure several Mexican hat wavelets are drawn against time and frequency. The centre frequency and Heisenberg boxes are indicated on the time-frequency axis. This is drawn in a similar manner as the figures in [19] and [54]

6.2 MULTIRESOLUTION SIGNAL DECOMPOSITION

Multiresolution signal decomposition using wavelets was introduced by S. Mallat in 1989 in the paper "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation" [53]. In this the link between wavelet analysis and subband decomposition was established.

In multiresolution analysis a discrete signal $x[n]$ is represented as two signals, $a[n]$ and $d[n]$ that are both half the length of the original signal. The first signal $a[n]$ is an approximation of the original and the second signal, $d[n]$ contains the fluctuations (detail components). The approximation is the result of filtering the signal with a low-pass filter and down-sampling it by a factor of two. The detail signal is the result of a high-pass filter and down-sampling. The approximation signal can again be further subdivided (each offering a lower resolution representation of the original signal). Such a setup is illustrated with the Haar transform below. The sections that follow this systematically develops the multiresolution representation from wavelet functions.

6.2.1 Haar transform example

For a discrete signal $x[n]$ (length N) the Haar approximation coefficients can be calculated by

$$a[n] = \frac{1}{\sqrt{2}}x[2n] + \frac{1}{\sqrt{2}}x[2n+1], \tag{6.34}$$

and the detail coefficients can be calculated as

$$d[n] = \frac{1}{\sqrt{2}}x[2n] - \frac{1}{\sqrt{2}}x[2n+1]. \tag{6.35}$$

As can be seen from the above, the $2n$ index introduces downsampling. As an example, consider the signal

$$x = \begin{bmatrix} 1 & 2 & 2 & 1 & 0 & 1 & 2 & 1 \end{bmatrix} \tag{6.36}$$

The approximation coefficients is calculated using the above as

$$a[n] = \frac{1}{\sqrt{2}} \begin{bmatrix} 3 & 3 & 1 & 3 \end{bmatrix} \tag{6.37}$$

and the detailed coefficients as

$$d[n] = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix}. \tag{6.38}$$

It can easily be seen that the original signal can be reconstructed from $a[n]$ and $d[n]$ by

$$x\left[2n\right] = \frac{\sqrt{2}}{2}a[n] + \frac{\sqrt{2}}{2}d[n] \tag{6.39}$$

and

$$x\left[2n+1\right] = \frac{\sqrt{2}}{2}a[n] - \frac{\sqrt{2}}{2}d[n]. \tag{6.40}$$

6.2.2 Preliminaries

This section will cover several preliminaries needed to develop multiresolution analysis.

Definition 1 (Riesz basis [19]) *The sequence $\{\phi_n\}_{n\in\Gamma}$ is a frame of $\mathcal{H}$ if there exists two constants $B \geq A > 0$ such that*

$$\forall f \in H, \quad A\left|f\right|^2 \leq \sum_{n\in\Gamma} \left|\langle f, \phi_n\rangle\right|^2 \leq B\left|f\right|^2. \tag{6.41}$$

If $A = B$ and $\{\phi_n\}_{n\in\Gamma}$ is linearly independent then the frame is a Riesz basis.

A Riesz basis therefore spans the space (i.e. any $f \in \mathcal{H}$ can be represented as a linear combination of $\phi_n(t)$ functions). In linear algebra an analogous concept is a linearly independent (but not orthogonal) basis. The orthogonalisation trick to orthogonalise a Riesz basis is given in section 6.2.3.1. This is again analogous to the Gram-Schmidt orthogonalisation in linear algebra.

6.2.3 Multiresolution representation requirements

A function f is approximated at a scale 2^j. The approximation of a function at a scale 2^j is the orthogonal projection on a space $V_j \subset L^2(\mathbb{R})$. The orthogonal projection is the function $f_j \in V_j$ that minimises $|f - f_j|$ [19].

The idea behind multiresolution analysis is to describe a signal from a coarse to a high resolution. A signal is represented in a sequence of approximation spaces V_j. The closed subspaces satisfy [20, 53]

$$\ldots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \ldots \tag{6.42}$$

with

$$\bigcup_{j\in\mathbb{Z}} V_j = L^2(\mathbb{R}) \tag{6.43}$$

and

$$\bigcap_{j\in\mathbb{Z}} V_j = \{0\}\,. \tag{6.44}$$

The above ensures that $\lim_{j\to-\infty} P_j f = f$ (where P_j is an orthogonal projection of f on V_j) [19]. The multiresolution aspect is created by the requirement that [20]

$$\forall j \in \mathbb{Z}, \quad f(t) \in V_j \Leftrightarrow f\left(\frac{t}{2}\right) \in V_{j+1}. \tag{6.45}$$

In other words, a dilation of every function is included in the coarser space. The full requirements for a multiresolution approximation are given in [19].

6.2.3.1 Orthogonalisation of the scaling function

The construction of an orthonormal basis often starts with a Riesz basis. Conditions are given in [20] to ensure that $\{\theta_{j,n}\}_{n\in\mathbb{Z}}$ is a Riesz basis for every V_j (conditions 5.2.4 and 5.2.5 in [20]). Orthogonalising the basis can be performed by the following theorem from [19].

Theorem 2 *Let $\{V_j\}_{j\in\mathbb{Z}}$ be a multiresolution approximation and ϕ be the scaling function having a Fourier transform*

$$\hat{\phi}(\omega) = \frac{\hat{\theta}(\omega)}{\left(\sum_{k=-\infty}^{\infty} \left|\hat{\theta}(\omega+2k\pi)\right|^2\right)^{\frac{1}{2}}}. \tag{6.46}$$

Then

$$\phi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\phi\left(\frac{t-2^j n}{2^j}\right) \tag{6.47}$$

forms an orthonormal basis of V_j for all $j \in \mathbb{Z}$.

The above theorem is easily proven by noting that $\langle\phi(t-n), \phi(t-p)\rangle = \delta_{n-p}$. The theorem is also known as the orthogonalisation trick in [20] (The Riesz basis is denoted in this text as ϕ

and the orthogonalised basis is denoted as $\phi^{\#}$). The orthogonal projection

$$P_{V_j} f = \sum_{n=-\infty}^{\infty} \langle f, \phi_{j,n} \rangle \, \phi_{j,n} \tag{6.48}$$

is obtained if f is expanded over V_j. The inner products $a_j[n] = \langle f, \phi_{j,n} \rangle$ provides a discrete approximation at scale 2^j [19].

6.2.3.2 Calculation of multiresolution filters

Since $V_1 \subset V_0$, any scaling function in V_1 can be written in terms of a sum in V_0:

$$\frac{1}{\sqrt{2}}\phi\left(\frac{t}{2}\right) = \sum_{n=-\infty}^{\infty} \left\langle \frac{1}{\sqrt{2}}\phi\left(\frac{t}{2}\right), \phi(t-n) \right\rangle \phi(t-n) \tag{6.49}$$

which can be more concisely written as

$$h[n] = \left\langle \frac{1}{\sqrt{2}}\phi\left(\frac{t}{2}\right), \phi(t-n) \right\rangle \tag{6.50}$$

and

$$\frac{1}{\sqrt{2}}\phi\left(\frac{t}{2}\right) = \sum_{n=-\infty}^{\infty} h[n]\phi(t-n). \tag{6.51}$$

$h[n]$ above is clearly a discrete filter. Taking the Fourier transform of both sides of the above equation and using the identities in equations A.3 and A.7 gives

$$\sqrt{2}\hat{\phi}(2\omega) = \sum_{n=-\infty}^{\infty} h[n]\hat{\phi}(\omega)e^{-j\omega n} \tag{6.52}$$

$$\sqrt{2}\hat{\phi}(2\omega) = \hat{\phi}(\omega) \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n}. \tag{6.53}$$

The right hand side of the above is recognized as the discrete-time Fourier transform of $h[n]$ (see equation A.18). The $\hat{h}(\omega)$ can therefore be calculated as

$$\hat{h}(\omega) = \sqrt{2}\frac{\hat{\phi}(2\omega)}{\hat{\phi}(\omega)}. \tag{6.54}$$

6.2.3.3 Orthogonal wavelets

Since $V_j \subset V_{j-1}$, another space can be defined that is orthogonal to V_j such that

$$V_{j-1} = V_j + W_j. \tag{6.55}$$

The projection of V_{j-1} can therefore be decomposed as [19]

$$P_{V_{j-1}} f = P_{V_j} f + P_{W_j} f. \tag{6.56}$$

The wavelet ψ that is the basis for W_j can be constructed from the scaling function as the following theorem from [19] states:

Theorem 3 (Orthonormal basis [19]) *Let ϕ be a scaling function and h the corresponding filter. Let ψ be a function with a Fourier transform*

$$\hat{\psi}(\omega) = \frac{1}{\sqrt{2}} \hat{g}(\omega) \hat{\phi}\left(\frac{\omega}{2}\right) \tag{6.57}$$

with

$$\hat{g}(\omega) = e^{-j\omega} \hat{h}^*(\omega + \pi). \tag{6.58}$$

Then $\psi_{j,n} = \frac{1}{\sqrt{2^j}} \psi\left(\frac{t-2^j n}{2^j}\right)$ is a basis for any scale at 2^j.

$g[n]$ can be calculated from

$$g[n] = \frac{1}{\sqrt{2}} \left\langle \psi\left(\frac{t}{2}\right), \phi(t-n) \right\rangle. \tag{6.59}$$

For real filters this can be calculated as

$$g[n] = (-1)^{1-n} h[1-n]. \tag{6.60}$$

The proof that the above stems from $\hat{g}(\omega) = e^{-j\omega}\hat{h}^*(\omega + \pi)$ is given below.

$$\sum_{n=-\infty}^{\infty} g[n]e^{-j\omega n} = e^{-j\omega} \sum_{n=-\infty}^{\infty} (h[n])^* e^{jn(\omega+\pi)} \tag{6.61}$$

$$= \sum_{n=-\infty}^{\infty} (h[n])^* e^{jn\omega}(-1)^n e^{-j\omega} \tag{6.62}$$

$$= \sum_{n=-\infty}^{\infty} (h[n])^* e^{j\omega(n-1)}(-1)^n \tag{6.63}$$

Making the substitution $m = 1 - n$ gives

$$\sum_{n=-\infty}^{\infty} g[n]e^{-j\omega n} = \sum_{m=-\infty}^{\infty} \left[h^*[1-m](-1)^{1-m}\right] e^{-j\omega m}. \tag{6.64}$$

If $h[n]$ is real, equating the two discrete-time Fourier transforms proves 6.60.

6.2.3.4 Multiresolution calculation with a filter bank

To make the multiresolution approximation clear, consider a function $f(t)$ that is approximated at a scale V_0 with $a_0[n]$. This approximation can be expressed as

$$P_{V_0} f = \sum_{n=-\infty}^{\infty} \langle f, \phi(t-n)\rangle \phi(t-n) \tag{6.65}$$

$$= \sum_{n=-\infty}^{\infty} a_0[n]\phi(t-n) \tag{6.66}$$

where

$$a_0[n] = \langle f, \phi(t-n)\rangle . \tag{6.67}$$

The calculation of $P_{V_1} f$ can be done in a similar manner as

$$P_{V_1} f = \sum_{n=-\infty}^{\infty} \left\langle f, \frac{1}{\sqrt{2}}\phi\left(\frac{t-2n}{2}\right)\right\rangle \frac{1}{\sqrt{2}}\phi\left(\frac{t-2n}{2}\right) \tag{6.68}$$

$$= \sum_{n=-\infty}^{\infty} a_1[n]\frac{1}{\sqrt{2}}\phi\left(\frac{t-2n}{2}\right) \tag{6.69}$$

where

$$a_1[n] = \left\langle f, \frac{1}{\sqrt{2}}\phi\left(\frac{t-2n}{2}\right)\right\rangle. \tag{6.70}$$

To help with the above simplification, the inner product of $\phi_{1,p}$ and $\phi_{0,n}$ is calculated below.

$$\left\langle \frac{1}{\sqrt{2}}\phi\left(\frac{t-2p}{2}\right), \phi(t-n)\right\rangle = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2}}\phi\left(\frac{t-2p}{2}\right)\phi^*(t-n)\,dt. \tag{6.71}$$

Making the substitution $\tau = t - 2p$ gives

$$\left\langle \frac{1}{\sqrt{2}}\phi\left(\frac{t-2p}{2}\right), \phi(t-n)\right\rangle = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2}}\phi\left(\frac{\tau}{2}\right), \phi^*(\tau+2p-n)\,dt \tag{6.72}$$

$$= h[n-2p] \tag{6.73}$$

where $h[n]$ is defined in equation 6.50. Using the above, $\frac{1}{\sqrt{2}}\phi\left(\frac{t-2n}{2}\right)$ can be written as (analogous to 6.51)

$$\frac{1}{\sqrt{2}}\phi\left(\frac{t-2n}{2}\right) = \sum_{k=-\infty}^{\infty} h[k-2n]\phi(t-k). \tag{6.74}$$

Substituting the above into the expression for $a_1[n]$ (equation 6.70) gives

$$a_1[n] = \left\langle f, \sum_{k=-\infty}^{\infty} h[k-2n]\phi(t-k)\right\rangle \tag{6.75}$$

$$= \sum_{k=-\infty}^{\infty} h[k-2n]\,\langle f, \phi(t-k)\rangle. \tag{6.76}$$

The inner product above is the same as 6.67. Substituting this in turn results in

$$a_1[n] = \sum_{k=-\infty}^{\infty} h[k-2n]a_0[k]. \tag{6.77}$$

The above is a convolution sum of $h[-n]$ with a_0 which is then downsampled. To make it clearer, substitute $k - 2n = -p$ and define $\bar{h}[n] = h[-n]$. This gives

$$a_1[n] = \sum_{p=-\infty}^{\infty} h[-p]a_0[2n-p]. \tag{6.78}$$

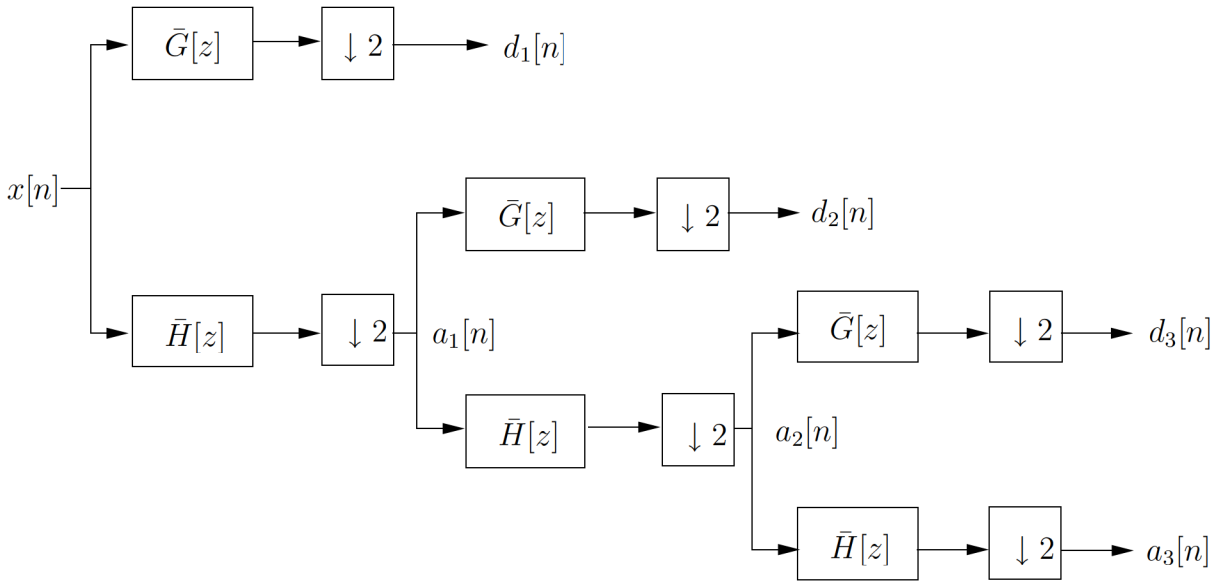


FIGURE 6.3: Wavelet decomposition tree for the discrete wavelet transform.

This is the convolution sum with $h[-n]$ (where $2n$ causes the downsampling). Writing in terms of $\bar{h}[n] = h[-n]$ gives

$$a_1[n] = \sum_{p=-\infty}^{\infty} \bar{h}[p] a_0[2n-p]. \tag{6.79}$$

Therefore the multiresolution approximation can be done by only using discrete filters. As can be seen from above, the coarser approximation is calculated by applying the filter $\bar{h}(n)$ and then downsampling the result by a factor of two. A filter $\bar{g}[n]$ can also be derived along the same lines for $g[n]$.

These filters are conjugate mirror filters (also known as quadrature mirror filters). Conjugate filters are traditionally used in subband coding schemes. Another set of filters can be calculated to reconstruct the signal (these filters are called the synthesis filters). The proof that the filters are indeed conjugate filters is in [53]. A discussion of quadrature mirror filters in the traditional digital signal processing setting can be found in [55] and an extensive discussion of subband decomposition is done in [56].

6.2.3.5 Discrete wavelet transform

The decomposition in figure 6.3 is often described as the discrete wavelet transform (DWT) (even though the DWT is sometimes defined by discretising equation 6.1 instead of using

multiresolution filters – cf. [19]). The decomposition described in the previous section can be represented as a tree (see figure 6.3). A similar tree can be created for the reconstruction case.

6.2.3.6 Haar wavelet example

The above principles is illustrated for the Haar wavelet. The scaling function for the Haar wavelet is [52]

$$\phi(t) = \begin{cases} 1 & 0 \leq t \leq 1 \\ 0 & otherwise \end{cases} \tag{6.80}$$

and the Haar wavelet is

$$\psi(t) = \begin{cases} 1 & 0 \leq t \leq \frac{1}{2} \\ -1 & \frac{1}{2} < t \leq 1 \\ 0 & otherwise. \end{cases} \tag{6.81}$$

These functions are illustrated in figure 6.4. The Fourier transform of the above functions can be calculated using the rectangular function (defined in equation A.12). This gives

$$\phi(t) = \Pi\left(t - \frac{1}{2}\right) \tag{6.82}$$

$$\psi(t) = \Pi\left(\frac{t \quad \frac{1}{4}}{2}\right) - \Pi\left(\frac{t \quad \frac{3}{4}}{2}\right). \tag{6.83}$$

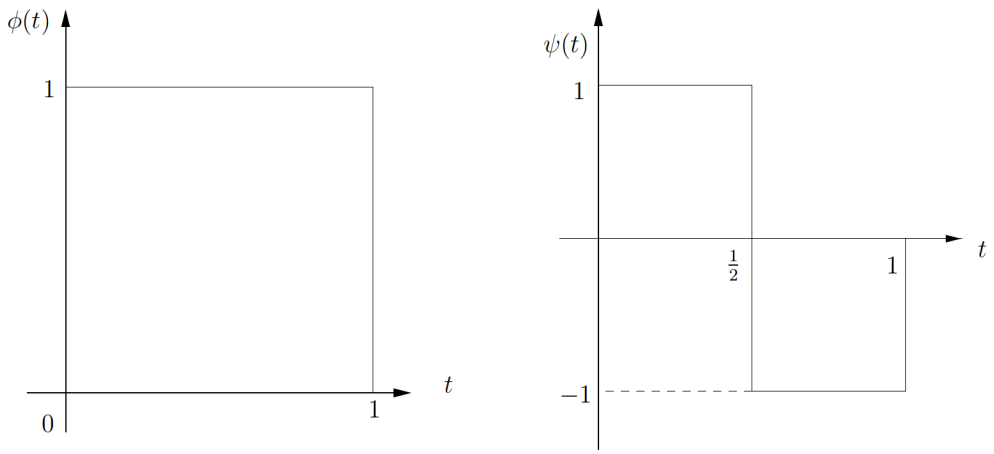


FIGURE 6.4: The scaling function (left) and wavelet (right) for the Haar wavelet.

The Fourier transform is therefore

$$\hat{\phi}(\omega) = \frac{2\sin\left(\frac{\omega}{2}\right)}{\omega} e^{-\frac{1}{2}j\omega} \tag{6.84}$$

and

$$\hat{\psi}(\omega) = \frac{1}{2}\hat{\Pi}\left(\frac{\omega}{2}\right) e^{-j\omega\frac{1}{4}} - \frac{1}{2}\hat{\Pi}\left(\frac{\omega}{2}\right) e^{-j\omega\frac{3}{4}} \tag{6.85}$$

$$= \frac{2\sin\left(\frac{\omega}{4}\right)}{\omega} e^{-\frac{1}{4}\omega j} - \frac{2\sin\left(\frac{\omega}{4}\right)}{\omega} e^{-\frac{3}{4}\omega j} \tag{6.86}$$

It is easy to show that these two functions are orthogonal. The scaling filter can be calculated using equation 6.50 as

$$h[n] = \left\langle \frac{1}{\sqrt{2}}\phi\left(\frac{t}{2}\right), \phi(t-n) \right\rangle. \tag{6.87}$$

This calculation is illustrated in figure 6.5. The first value of the discrete filter will therefore be

$$h[0] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2}}\phi\left(\frac{t}{2}\right)\phi(t)dt \tag{6.88}$$

$$= \int_{0}^{1} \frac{1}{\sqrt{2}}(1)dt = \frac{1}{\sqrt{2}} \tag{6.89}$$

For the $h[1]$ component this is

$$h[1] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2}}\phi\left(\frac{t}{2}\right)\phi(t-1)dt \tag{6.90}$$

$$= \int_{1}^{2} \frac{1}{\sqrt{2}}(1)dt = \frac{1}{\sqrt{2}} \tag{6.91}$$

From the illustration it can be seen that all other components of h will be zero. The filter for the wavelet can be calculated in almost the same way using equation 6.59

$$g[n] = \left\langle \frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right), \phi(t-n) \right\rangle. \tag{6.92}$$

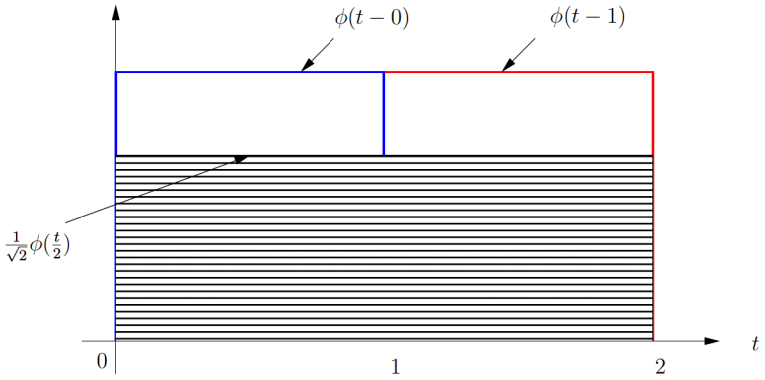


FIGURE 6.5: Calculation of scaling filter.

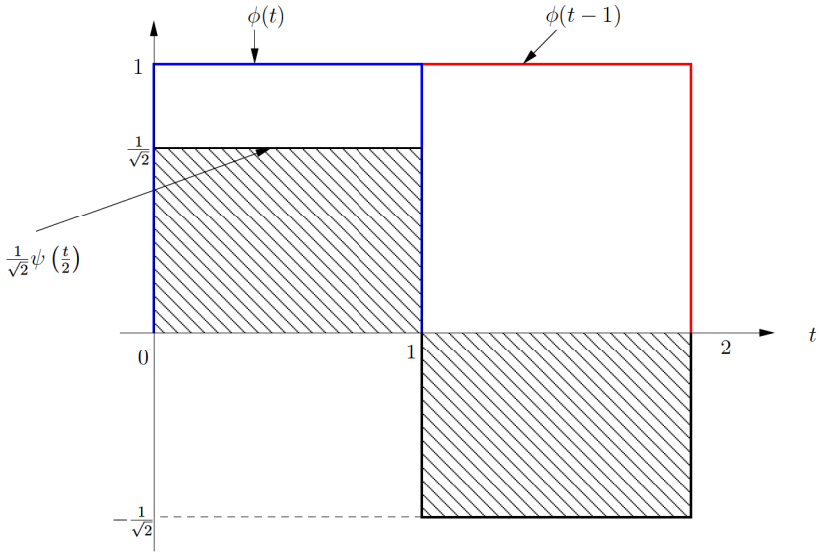


FIGURE 6.6: Calculation of Haar wavelet filter

This is illustrated in figure 6.6. The filters can therefore be calculated as

$$g[0] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right)\phi(t)dt \tag{6.93}$$

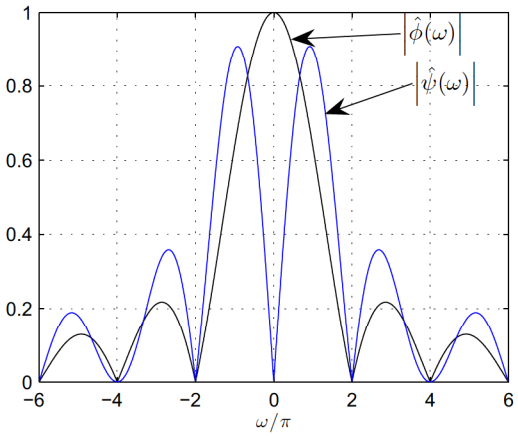
$$= \int_{0}^{1} \frac{1}{\sqrt{2}}dt = \frac{1}{\sqrt{2}} \tag{6.94}$$

and

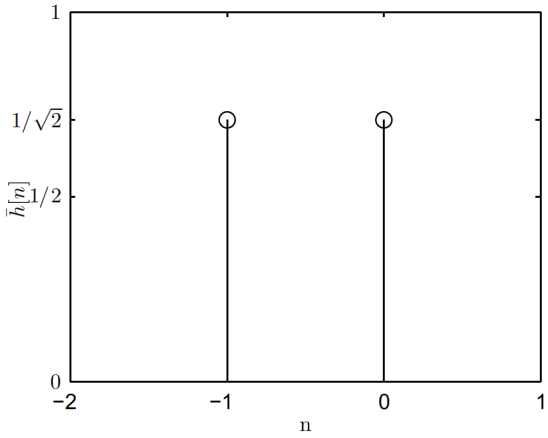
$$g[1] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right)\phi(t-1)dt \tag{6.95}$$

$$= \int_{1}^{2} \frac{-1}{\sqrt{2}}dt = -\frac{1}{\sqrt{2}}. \tag{6.96}$$

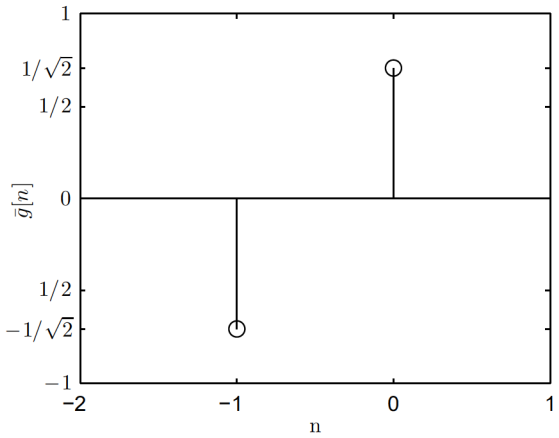
The filters to be used for the multiresolution approximation are therefore $\bar{h}[n] = h[-n]$. These



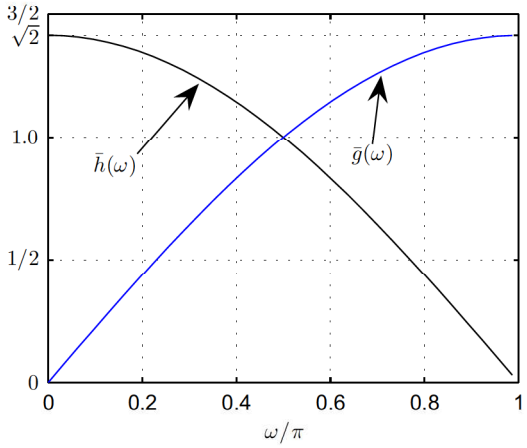
(a) Fourier transform of the Haar scaling function and wavelet function



(b) Filter coefficients of $\bar{h}$



(c) Filter coefficients of $\bar{g}$



(d) Frequency response of the $\bar{h}(\omega)$ and $\bar{g}(\omega)$ filters.

FIGURE 6.7: Filter coefficients and frequency response for the Haar wavelet.

are given as

$$\bar{h}[n] = \begin{cases} \frac{1}{\sqrt{2}} & n = -1, 0 \\ 0 & \textit{otherwise} \end{cases} \tag{6.97}$$

and

$$\bar{g}[n] = \begin{cases} \frac{1}{\sqrt{2}} & n = 0 \\ -\frac{1}{\sqrt{2}} & n = -1 \\ 0 & \textit{otherwise}. \end{cases} \tag{6.98}$$

The filter impulse response and discrete-time Fourier transforms are plotted in figure 6.7.

6.2.4 Summary

A summary for the calculation of the multiresolution filters from the Riesz basis is given in figure 6.2.4. This was used to calculate the Battle-Lemarié quadrature mirror filters in section 6.5.1.

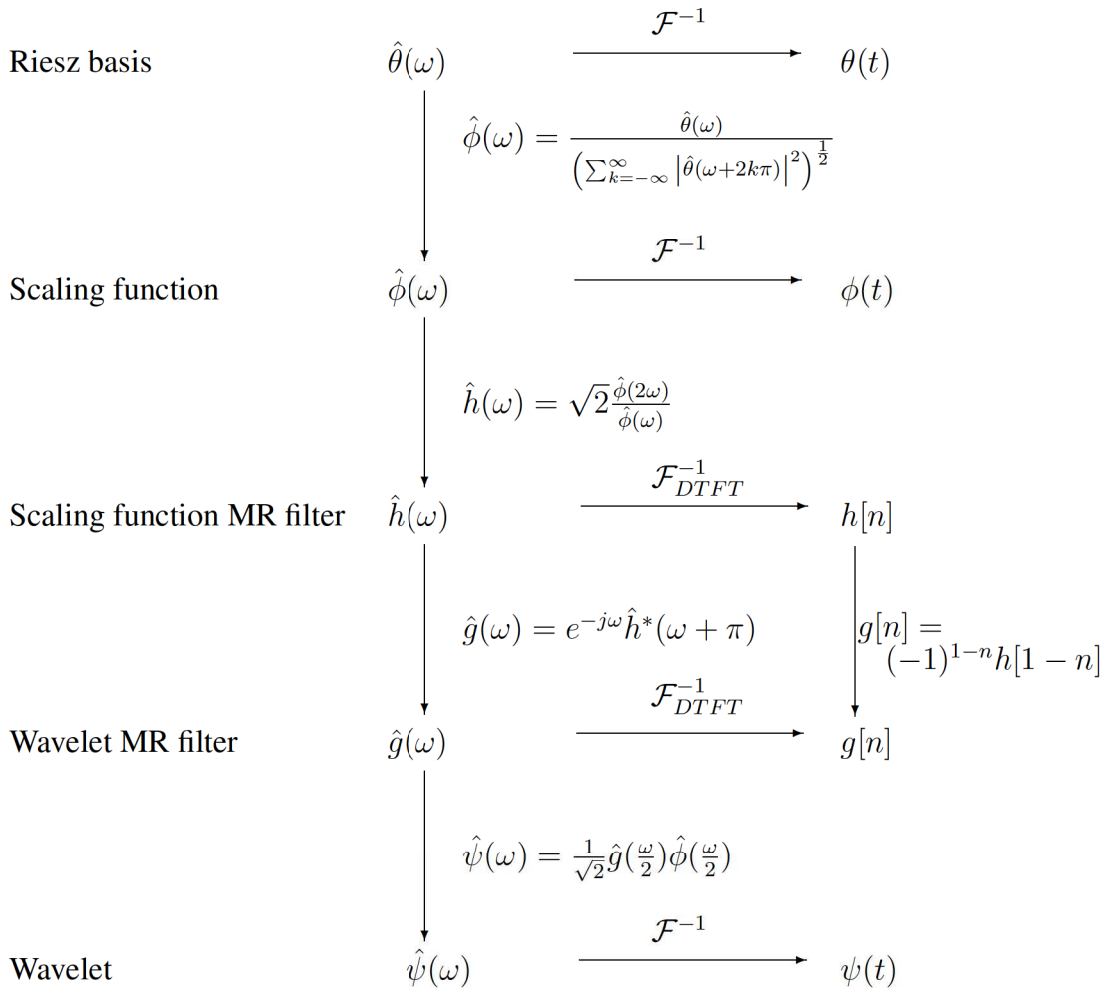


FIGURE 6.8: Calculation of scaling function, wavelet and multiresolution filters from a Riesz basis. The inverse Fourier transform ($\mathcal{F}^{-1}$) and the inverse discrete-time Fourier transform ($\mathcal{F}^{-1}_{DTFT}$) are defined in equations A.2 and A.19 respectively.

6.3 WAVELET PACKET TRANSFORMS

The multiresolution approach repeatedly divides the approximation space (V_j) into two separate spaces (as shown in figure 6.9). This can be generalised by also subdividing the detail space. This is called wavelet packet analysis and it was introduced by Coifman, Meyer and Wickerhauser in [57].

This subdivision is shown in figure 6.10. A new notation is used – the subscript indicates the level in the tree and the superscript indicates the position. A left branch indicates the approximation space (low-pass) and the right branch indicates the detailed space.

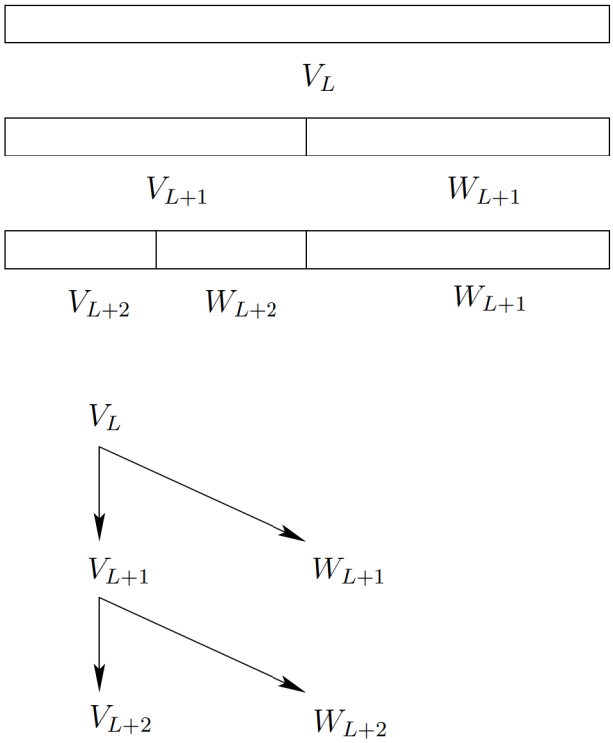


FIGURE 6.9: Subdivision of the signal in the normal multiresolution case.

Several possible subdivisions are possible. A tree is an admissible tree if each node has either zero or one child (i.e. the original signal can be reconstructed from the WPT). It is easy to see that the original signal can be reconstructed from the leaf nodes of any admissible tree. Each different admissible tree has a different covering of the time-frequency spectrum.

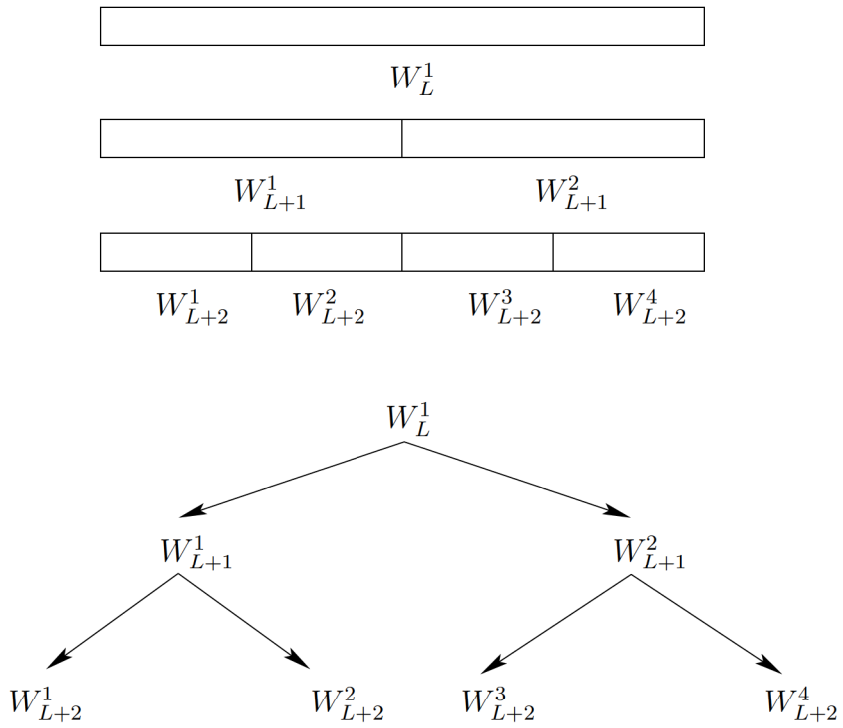


FIGURE 6.10: Subdivision of signal in the wavelet packet case.

6.4 WAVELET PROPERTIES

6.4.1 Vanishing moments

The function ψ has p vanishing moments if [19]

$$\int_{-\infty}^{\infty} t^k \psi(t)\, dt = 0 \quad \forall\, 0 \leq k < p. \tag{6.99}$$

This means that $\psi(t)$ is orthogonal to any polynomial up to degree $p-1$. If a signal is orthogonal to the wavelet, it would produce zero values. This is often desirable in compression applications – if the underlying signal is highly regular only a few non-zero values would be produced.

6.4.2 Size of support

A function $f(t)$ is supported on $[a, b]$ if the function is zero outside that interval. A signal can be represented in W_j as

$$P_{W_j} f = \sum_{n=-\infty}^{\infty} \langle f, \psi_{j,n} \rangle \psi_{j,n}. \tag{6.100}$$

The size of the wavelet support determines the effect that an isolated singularity will have [19]. Suppose that the signal is zero except for a single singularity. The detail coefficients of the signal is $d[n] = \langle f, \psi_{j,n} \rangle$. It is clear that the larger the support of $\psi_{j,n}$ is, the more non-zero $d[n]$ values there will be.

There is an interplay between the length of the support and the number of vanishing coefficients. If ψ has p vanishing moments, then it has a support of at least $2p \quad 1$ [20]. Daubechies wavelets have the minimum support size for the number of vanishing moments.

The amount of $d[n]$ that is small or non-zero is usually motivated by compression considerations. As an example, the JPEG2000 standard uses a wavelets with a small support similar to the Daubechies wavelet. Since classification is the focus, little attention has been paid to the size of the wavelet support in the classifiers that was developed.

6.5 WAVELET FUNCTIONS

Several different orthogonal wavelets are discussed in this section. The Battle-Lemarié wavelet is discussed in depth in this section. The motivation for this is twofold. Firstly, the filter coefficients for high-order Battle-Lemarié filters are not available and secondly these wavelets were the best performing wavelets in certain classification problems.

A concise overview is given for Daubechies wavelets, Symmlets and Coiflets. Since the design considerations for these wavelets are not applicable for classification and their filter coefficients are readily available, they are only discussed briefly.

6.5.1 Battle-Lemarié wavelets

This section shows the calculation of Battle-Lemarié filters. This was done to calculate high order Battle-Lemarié filters as the filter coefficients for these are not readily available. The derivation of the scaling function is done the same way as in [20]. The Battle-Lemarié wavelet is defined as

$$\beta^{(n)}(t) = \underbrace{\Pi(t) * \Pi(t) * \ldots * \Pi(t)}_{n+1} \tag{6.101}$$

where Π is the rectangular pulse function defined as

$$\Pi(t) = \begin{cases} 1 & -\frac{1}{2} \leq t \leq \frac{1}{2}. \\ 0 & otherwise. \end{cases} \tag{6.102}$$

The convention is that $\beta^{(n)}(t)$ is centred around $t = 0$ for odd n and centred around $t = \frac{1}{2}$ for even n. The result for $\beta^{(n)}(t)$ for $n = 1$ and $n = 2$ is calculated in the next sections. It can be proven that equation 6.101 is a Riesz basis (cf. [20] p.g. 146).

6.5.1.1 Calculation for $n = 1$

$$\begin{aligned} \beta^{(1)}(t) &= \Pi(t) * \Pi(t) && (6.103) \\ &= \int_{-\infty}^{\infty} \Pi(\tau)\Pi(t-\tau)dt && (6.104) \end{aligned}$$

$\Pi(t - \tau)$ is calculated as

$$\begin{aligned} \Pi(t-\tau) &= \begin{cases} 1 & -\frac{1}{2} \leq t - \tau \leq \frac{1}{2} \\ 0 & otherwise \end{cases} && (6.105) \\ &= \begin{cases} 1 & -\frac{1}{2} + t \leq \tau \leq \frac{1}{2} + t. \\ 0 & otherwise. \end{cases} && (6.106) \end{aligned}$$

The two cases where the convolution is nonzero is illustrated in figure 6.11. The overlapping area is shaded. For $-1 < t < 0$ the value is

$$\begin{aligned} \beta^{(1)}(t) &= \int_{-\frac{1}{2}}^{\frac{1}{2}+t} (1)d\tau && (6.107) \\ &= t + 1. && (6.108) \end{aligned}$$

For $0 < t < 1$ this is calculated as

$$\begin{aligned} \beta^{(1)}(t) &= \int_{-\frac{1}{2}+t}^{\frac{1}{2}} 1 d\tau && (6.109) \\ &= 1 - t. && (6.110) \end{aligned}$$

The result is also shown in figure 6.11.

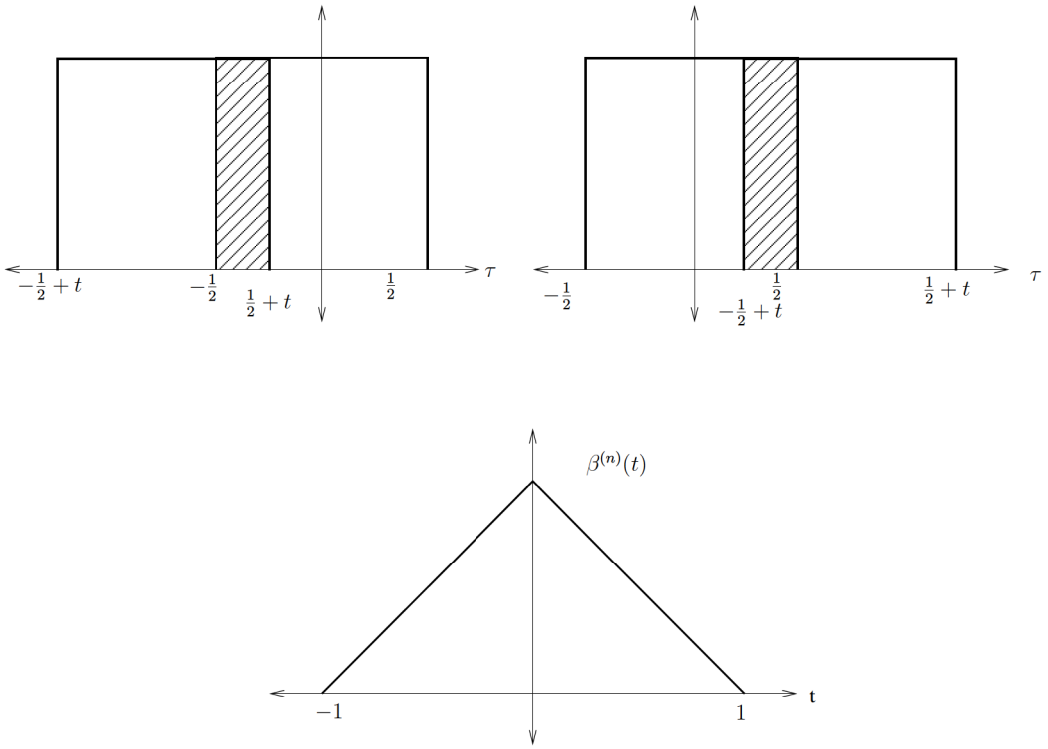


FIGURE 6.11: The result of convolution for $\beta^{(1)}(t)$.

6.5.1.2 Calculation for $n = 2$

The three cases for which the convolution is nonzero are shown in figure 6.12. For $-\frac{3}{2} < t < -\frac{1}{2}$ this is

$$\beta^{(2)}(t) = \int_{-1}^{\frac{1}{2}+t} \tau + 1 d\tau \tag{6.111}$$

$$= \frac{1}{2}t^2 + \frac{3}{2}t + \frac{9}{8}. \tag{6.112}$$

For $-\frac{1}{2} \leq t \leq \frac{1}{2}$ this is

$$\beta^{(2)}(t) = \int_{-\frac{1}{2}+t}^{0} \tau + 1 d\tau + \int_{0}^{\frac{1}{2}+t} \tau - 1 d\tau \tag{6.113}$$

$$= \frac{3}{4} - t^2. \tag{6.114}$$

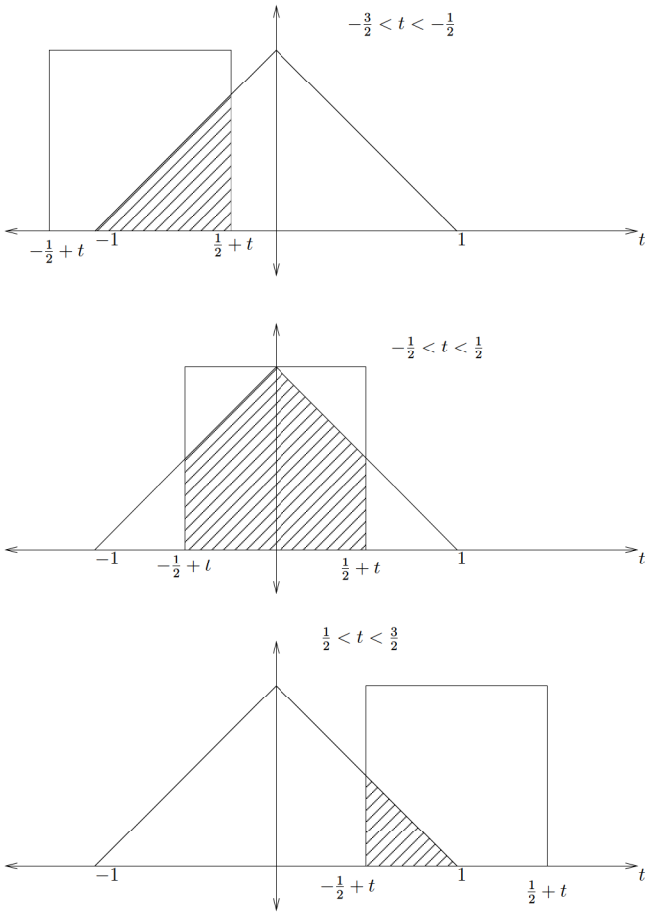


FIGURE 6.12: The convolution for $\beta^{(2)}(t)$.

For $\frac{1}{2} < t < \frac{3}{2}$ this is

$$\beta^{(2)}(t) = \int_{-\frac{1}{2}+t}^{1} \tau - 1 d\tau \tag{6.115}$$

$$= \frac{1}{2}t^2 - \frac{3}{2}t + \frac{9}{8}. \tag{6.116}$$

The result is plotted in figure 6.13.

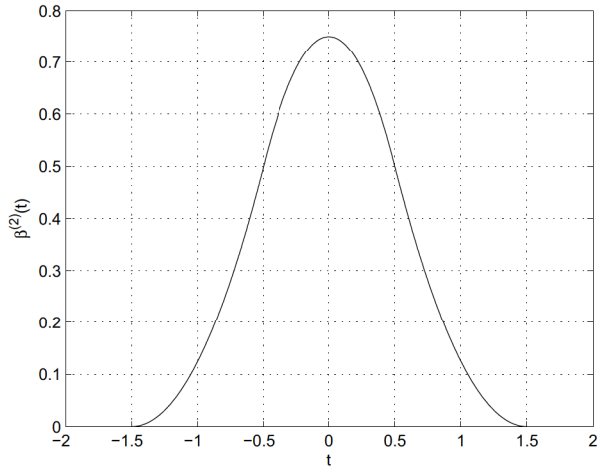


FIGURE 6.13: Result of Battle-Lemarié for $n = 2$

6.5.1.3 Frequency representation

The Fourier transform of the scaling function for the Battle-Lemarié wavelet is easier to calculate since the explicit calculation of the convolution can be avoided. The Fourier transform of the block pulse is

$$\hat{\Pi}(\omega) = \int_{-\infty}^{\infty} \Pi(t) e^{-j\omega t} dt \tag{6.117}$$

$$= \int_{-\frac{1}{2}}^{-\frac{1}{2}} e^{-j\omega t} dt \tag{6.118}$$

$$= \frac{1}{-j\omega} e^{-j\omega t} \Big|_{-\frac{1}{2}}^{\frac{1}{2}} \tag{6.119}$$

$$= \frac{1}{-j\omega} \left[e^{-j\frac{1}{2}\omega} - e^{\frac{1}{2}j\omega} \right]. \tag{6.120}$$

Using the identity $2j \sin x = e^{-jx} - e^{-jx}$ gives

$$\hat{\Pi}(\omega) = \frac{1}{-j\omega}(-2j) \sin \frac{1}{2}\omega \tag{6.121}$$

$$= \frac{\sin \frac{1}{2}\omega}{\frac{1}{2}\omega} \tag{6.122}$$

$$= sinc\left(\frac{\omega}{2\pi}\right) \tag{6.123}$$

with $sinc(x) = \frac{\sin \pi x}{\pi x}$. The standard for Battle-Lemarié wavelets is for the odd wavelets to be centred around 0 and the even wavelets to be centred around $\frac{1}{2}$. Thus the odd wavelets are

defined by:

$$\beta^{(n)}(t) = \underbrace{\Pi(t) * \Pi(t) * \ldots * \Pi(t)}_{n+1}. \tag{6.124}$$

Since convolution is just multiplication in the frequency domain, the Fourier transform of the above can be expressed as

$$\hat{\beta}^{(n)}(\omega) = \left[sinc\left(\frac{\omega}{2\pi}\right)\right]^{n+1}. \tag{6.125}$$

The case for where n is even is defined as

$$z(t) = \beta^{(n)}(t - t_0), \tag{6.126}$$

where $t_0 = \frac{1}{2}$ and $t_0 = 0$ for the odd case (i.e. unchanged). Using the time shift statement proven in (A.11), the Fourier transform of this can be expressed as

$$\hat{z}(\omega) = \hat{\beta}^{(n)}(\omega)e^{-j\omega t_0} \tag{6.127}$$

$$= \left[sinc\left(\frac{\omega}{2\pi}\right)\right]^{n+1} e^{-j\omega t_0}, \tag{6.128}$$

where $t_0 = \frac{1}{2}$ for the even case and $t_0 = 0$ for the odd case.

6.5.1.4 Orthogonalising the basis

The Riesz basis can be orthogonalised with equation 6.46. For the Battle-Lemarié wavelet the Fourier transform of the Riesz basis for odd n is

$$\hat{\theta}(\omega) = \left[\frac{sin\left(\frac{\omega}{2}\right)}{\frac{\omega}{2}}\right]^{n+1}. \tag{6.129}$$

Therefore

$$\left|\hat{\theta}(\omega + 2\pi k)\right|^2 = \left[\frac{sin\left(\frac{\omega}{2} + \pi k\right)}{\frac{\omega}{2} + \pi k}\right]^{2(n+1)} \tag{6.130}$$

(where $k \in \mathbb{Z}$). A simple identity can be defined to simplify the sin factor (when $k \in \mathbb{Z}$) :

$$\begin{aligned} \sin(x + k\pi) &= \sin x \cos k\pi + \sin k\pi \cos x & (6.131) \\ &= (-1)^k \sin x. & (6.132) \end{aligned}$$

Simplifying the denominator of 6.46 gives

$$\begin{aligned} \sum_{k=-\infty}^{\infty} \left|\hat{\theta}(\omega + 2k)\right|^2 &= \left|\sin\left(\frac{\omega}{2}\right)\right|^{2(n+1)} \sum_{k=-\infty}^{\infty} \frac{1}{\left(\frac{\omega}{2} + \pi k\right)^{2(n+1)}} & (6.133) \\ \sqrt{\sum_{k=-\infty}^{\infty} \left|\hat{\theta}(\omega + 2k)\right|^2} &= \left|\sin\left(\frac{\omega}{2}\right)\right|^{(n+1)} \sqrt{\sum_{k=-\infty}^{\infty} \frac{1}{\left(\frac{\omega}{2} + \pi k\right)^{2(n+1)}}}. & (6.134) \end{aligned}$$

The above can be written in a simpler form if the following definition is made

$$z_p(\omega) = \textstyle\sum_{k=-\infty}^{\infty} \frac{1}{\left(\frac{\omega}{2}+\pi k\right)^p} \quad (6.135)$$

where $p - 2(n + 1)$. The above is then

$$\sqrt{\sum_{k=-\infty}^{\infty} \left|\hat{\theta}(\omega + 2k)\right|^2} = \left|\sin\left(\frac{\omega}{2}\right)\right|^{(n+1)} \sqrt{z_{2(n+1)}(\omega)}. \quad (6.136)$$

There exists an identity that relates the series expression in terms of a sine function (see [58] for a list of identities)

$$\frac{1}{\sin^2 x} = \sum_{k=-\infty}^{\infty} \frac{1}{(k\pi + x)^2} \quad \forall \frac{x}{\pi} \notin \mathbb{Z}. \quad (6.137)$$

The summation can be written as

$$z_p(\omega) = \sum_{k=-\infty}^{\infty} \frac{1}{\left(\frac{\omega}{2} + \pi k\right)^p}. \quad (6.138)$$

The derivative of this is

$$\frac{d}{d\omega}z_p(\omega) = \frac{d}{d\omega}\sum_{k=-\infty}^{\infty}\frac{1}{\left(\frac{\omega}{2}+\pi k\right)^p} \tag{6.139}$$

$$= \frac{1}{2}(-p)\sum_{k=-\infty}^{\infty}\frac{1}{\left(\frac{\omega}{2}+\pi k\right)^{p+1}} \tag{6.140}$$

$$= \frac{-p}{2}z_{p+1}(\omega). \tag{6.141}$$

Thus $z_{p+1}(\omega)$ can be expressed as

$$z_{p+1}(\omega) = \frac{-2}{p}\frac{d}{d\omega}z_p(\omega). \tag{6.142}$$

A general formula can be derived from the above in terms of $z_2(\omega)$ by recursively expanding $z_p(\omega)$ as

$$z_p(\omega) = \frac{-2}{p-1}\frac{d}{d\omega}\left(\frac{-2}{p-2}\frac{d}{d\omega}\left(\ldots\frac{-2}{2}\frac{d}{d\omega}z_2(\omega)\right)\right) \tag{6.143}$$

$$= (-2)^{p-2}\frac{1}{(p-1)!}\frac{d^{p-2}}{d\omega^{p-2}}z_2(\omega) \tag{6.144}$$

$$= (-1)^{p-2}2^{p-2}\frac{1}{(p-1)!}\frac{d^{p-2}}{d\omega^{p-2}}z_2(\omega). \tag{6.145}$$

The values for each of the $z_p(\omega)$ functions can thus be calculated if the value of $z_2(\omega)$ is known (It can easily be seen that the series does not converge for $z_1(\omega)$). $z_2(\omega)$ is written with the aid of the sine identity as

$$z_2(\omega) = \frac{1}{\sin^2\frac{\omega}{2}}. \tag{6.146}$$

$\hat{\phi}(\omega)$ can be calculated for the odd case as (from 6.46)

$$\hat{\phi}(\omega) = \frac{\hat{\theta}(\omega)}{\left(\sum_{k=-\infty}^{\infty}\left|\hat{\theta}(\omega+2k\pi)\right|^2\right)^{\frac{1}{2}}}. \tag{6.147}$$

Substituting the values above gives (odd n)

$$\hat{\phi}(\omega) = \left(\frac{\left[\sin\frac{\omega}{2}\right]^{n+1}}{\left(\frac{\omega}{2}\right)^{n+1}} \right) / \left(\left[\sin\frac{\omega}{2}\right]^{n+1} \sqrt{z_{2(n+1)}(\omega)} \right) \tag{6.148}$$

$$= \left(\frac{1}{\left(\frac{\omega}{2}\right)^{n+1}} \right) / \left(\sqrt{z_{2(n+1)}(\omega)} \right) \tag{6.149}$$

$$= \frac{1}{\left(\frac{\omega}{2}\right)^{n+1} \sqrt{z_{2(n+1)}(\omega)}}. \tag{6.150}$$

The calculation of the $z_{2(n+1)}$ functions is given in table 6.1. The calculation of $\hat{\phi}(\omega)$ is trivial once z_p is known. The calculation of the filter coefficients is discussed in the next section.

6.5.1.5 Calculation of Battle-Lemarié filter coefficients

The scaling function for the Batttle-Lemarié wavelets can be calculated by using equation 6.46. Because of $z_p(\omega)$ this expression tends to be extremely large even for moderate degrees of the filter. As an example, consider the values for $z_p(\omega)$ in table 6.1. For $n = 7$, $z_{16}(\omega)$ will be used. The expression for this is given in the last row of the table. Odd Battle-Lemarié filters up to the order of 31, were calculated. For this reason, CAS (Computer Algebra System) software was used. The scaling MR filter was calculated as (using equation 6.54)

$$\hat{h}(\omega) = \sqrt{2}\frac{\hat{\phi}(2\omega)}{\hat{\phi}(\omega)}. \tag{6.151}$$

The filter coefficients were calculated from the above using a numerical inverse discrete Fourier transform (DFT). The source code for to calculate $z_p(\omega)$ and the Fourier transform of the multiresolution filters is given in appendix B[3].

The scaling function, wavelet function and multiresolution filters for $n = 1, 5$ are plotted in figures 6.14 and 6.15. The time domain values were calculated using a numerical inverse Fourier transform. The multiresolution filters are plotted for $n = 1, 7, 31$ in figure 6.16 to show the effect of increasing the order of the polynomial.

[3] This was written in Maxima, an open source CAS package.

TABLE 6.1: Calculation for $z_p(\omega)$

Function	Value
$z_2(\omega)$	$\frac{1}{\sin^2\left(\frac{w}{2}\right)}$
$z_3(\omega)$	$\frac{\cos\left(\frac{w}{2}\right)}{\sin^3\left(\frac{w}{2}\right)}$
$z_4(\omega)$	$\frac{2\cos^2\left(\frac{w}{2}\right)+1}{3\sin^4\left(\frac{w}{2}\right)}$
$z_5(\omega)$	$\frac{\cos^3\left(\frac{w}{2}\right)+2\cos\left(\frac{w}{2}\right)}{3\sin^5\left(\frac{w}{2}\right)}$
$z_6(\omega)$	$\frac{2\cos^4\left(\frac{w}{2}\right)+11\cos^2\left(\frac{w}{2}\right)+2}{15\sin^6\left(\frac{w}{2}\right)}$
$z_7(\omega)$	$\frac{2\cos^5\left(\frac{w}{2}\right)+26\cos^3\left(\frac{w}{2}\right)+17\cos\left(\frac{w}{2}\right)}{45\sin^7\left(\frac{w}{2}\right)}$
$z_8(\omega)$	$\frac{4\cos^6\left(\frac{w}{2}\right)+114\cos^4\left(\frac{w}{2}\right)+180\cos^2\left(\frac{w}{2}\right)+17}{315\sin^8\left(\frac{w}{2}\right)}$
$z_9(\omega)$	$\frac{\cos^7\left(\frac{w}{2}\right)+60\cos^5\left(\frac{w}{2}\right)+192\cos^3\left(\frac{w}{2}\right)+62\cos\left(\frac{w}{2}\right)}{315\sin^9\left(\frac{w}{2}\right)}$
$z_{10}(\omega)$	$\frac{2\cos^8\left(\frac{w}{2}\right)+247\cos^6\left(\frac{w}{2}\right)+1452\cos^4\left(\frac{w}{2}\right)+1072\cos^2\left(\frac{w}{2}\right)+62}{2835\sin^{10}\left(\frac{w}{2}\right)}$
$z_{11}(\omega)$	$\frac{2\cos^9\left(\frac{w}{2}\right)+502\cos^7\left(\frac{w}{2}\right)+5097\cos^5\left(\frac{w}{2}\right)+7192\cos^3\left(\frac{w}{2}\right)+1382\cos\left(\frac{w}{2}\right)}{14175\sin^{11}\left(\frac{w}{2}\right)}$
$z_{12}(\omega)$	$\frac{4\cos^{10}\left(\frac{w}{2}\right)+2026\cos^8\left(\frac{w}{2}\right)+34096\cos^6\left(\frac{w}{2}\right)+83021\cos^4\left(\frac{w}{2}\right)+35396\cos^2\left(\frac{w}{2}\right)+1382}{155925\sin^{12}\left(\frac{w}{2}\right)}$
$z_{13}(\omega)$	$\frac{2\cos^{11}\left(\frac{w}{2}\right)+2036\cos^9\left(\frac{w}{2}\right)+55196\cos^7\left(\frac{w}{2}\right)+217186\cos^5\left(\frac{w}{2}\right)+171511\cos^3\left(\frac{w}{2}\right)+21844\cos\left(\frac{w}{2}\right)}{467775\sin^{13}\left(\frac{w}{2}\right)}$
$z_{14}(\omega)$	$\frac{4\cos^{12}\left(\frac{w}{2}\right)+8166\cos^{10}\left(\frac{w}{2}\right)+349500\cos^8\left(\frac{w}{2}\right)+2123860\cos^6\left(\frac{w}{2}\right)+2801040\cos^4\left(\frac{w}{2}\right)+776661\cos^2\left(\frac{w}{2}\right)+21844}{6081075\sin^{14}\left(\frac{w}{2}\right)}$
$z_{15}(\omega)$	$\frac{4\cos^{13}\left(\frac{w}{2}\right)+16356\cos^{11}\left(\frac{w}{2}\right)+1089330\cos^9\left(\frac{w}{2}\right)+9893440\cos^7\left(\frac{w}{2}\right)+20376780\cos^5\left(\frac{w}{2}\right)+10262046\cos^3\left(\frac{w}{2}\right)+929569\cos\left(\frac{w}{2}\right)}{42567525\sin^{15}\left(\frac{w}{2}\right)}$
$z_{16}(\omega)$	$\frac{8\cos^{14}\left(\frac{w}{2}\right)+65476\cos^{12}\left(\frac{w}{2}\right)+6715896\cos^{10}\left(\frac{w}{2}\right)+88951490\cos^8\left(\frac{w}{2}\right)+273021880\cos^6\left(\frac{w}{2}\right)+225028452\cos^4\left(\frac{w}{2}\right)+43800104\cos^2\left(\frac{w}{2}\right)+929569}{638512875\sin^{16}\left(\frac{w}{2}\right)}$

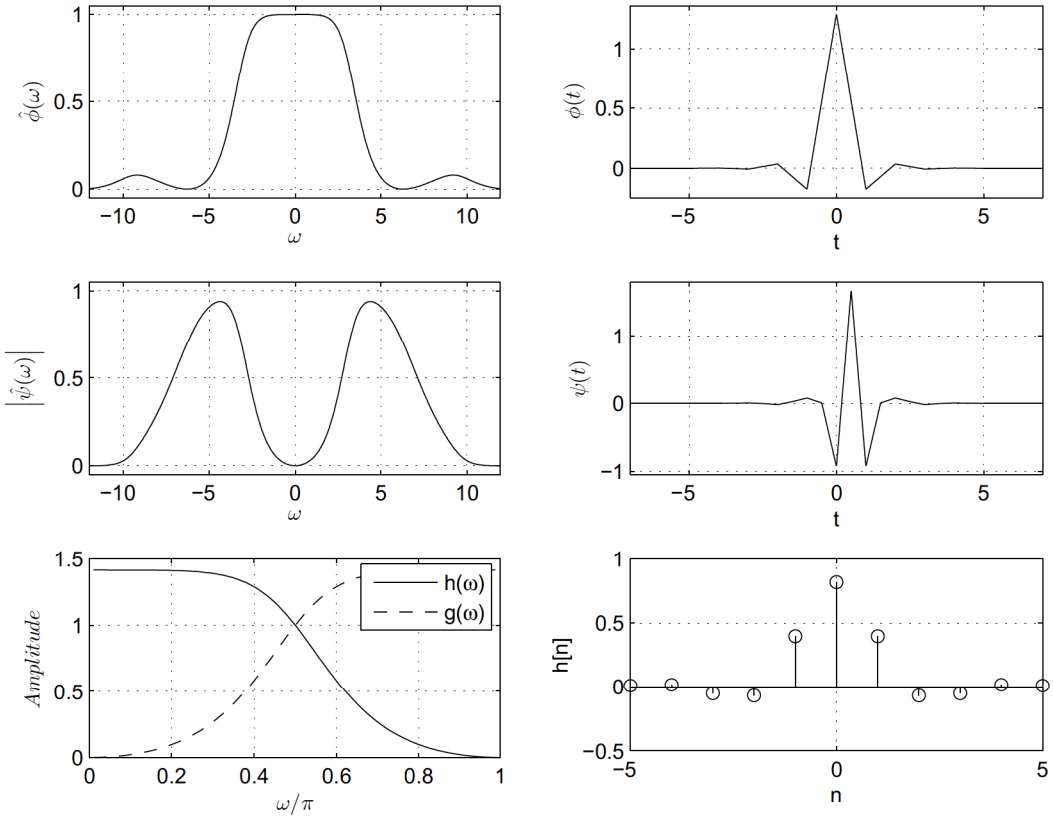


FIGURE 6.14: Plots for the linear Battle-Lemarié wavelet ($n = 1$). From top to bottom, left to right are the Fourier transform of the scaling function, scaling function, Fourier transform of the wavelet, the wavelet, the multiresolution filters (conjugate mirror filters) and the impulse response for the $h[n]$ filter.

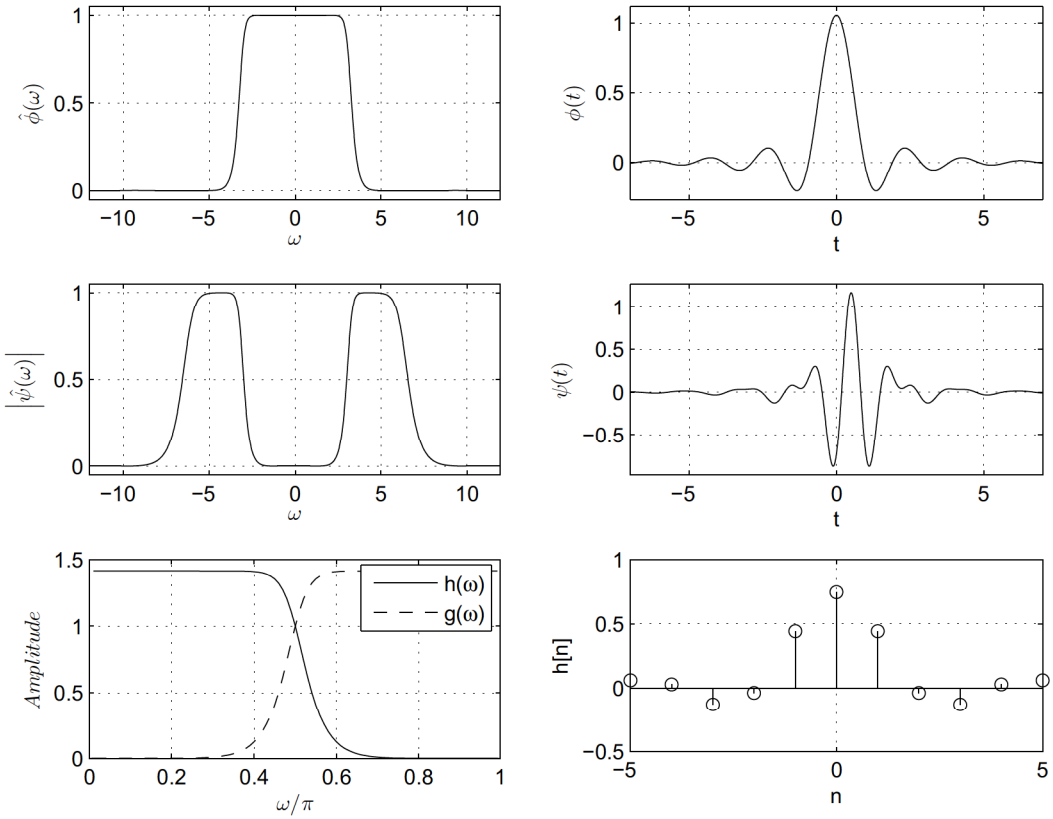


FIGURE 6.15: Plots for the linear Battle-Lemarié wavelet ($n = 5$). From top to bottom, left to right are the Fourier transform of the scaling function, scaling function, Fourier transform of the wavelet, the wavelet, the multiresolution filters (conjugate mirror filters) and the impulse response for the $h[n]$ filter.

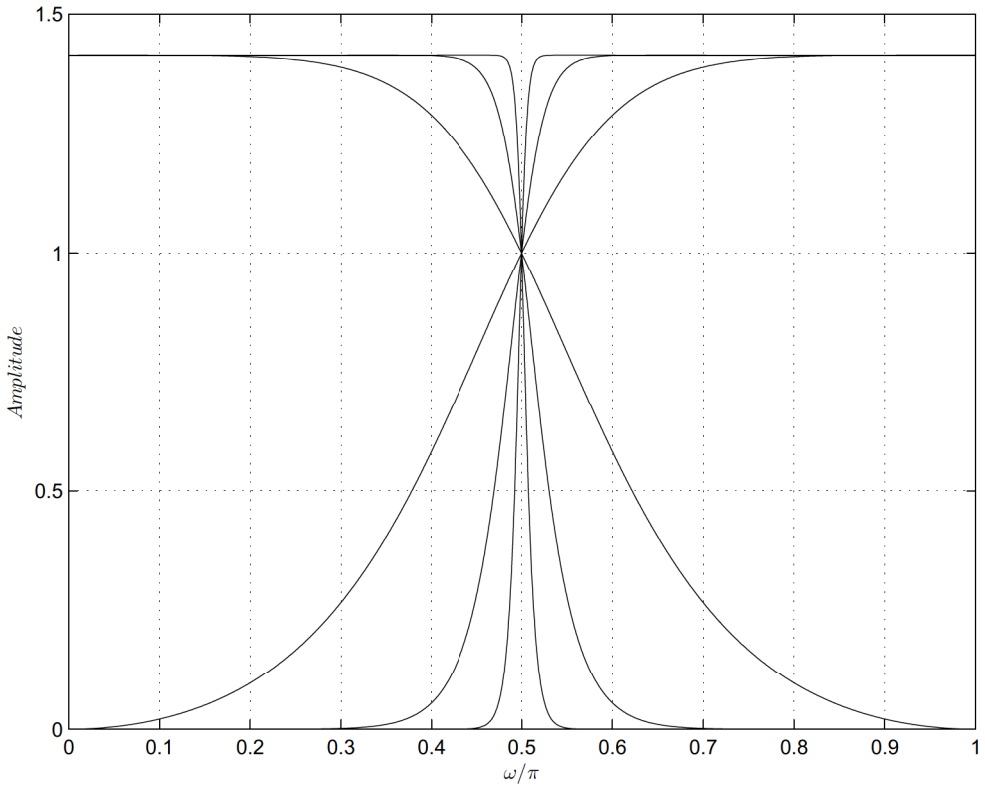


FIGURE 6.16: The multiresolution filters ($\hat{g}(\omega)$ and $\hat{h}(\omega)$) are plotted for the $n = 1$ (linear), $n = 7$ and the extreme case of $n = 31$. The filters can be identified by the sharpness of the cut-off – the higher the order of the filter, the sharper the cut-off.

6.5.2 Daubechies wavelets

As stated earlier, Daubechies wavelets are constructed to have the smallest support for the number of vanishing moments. If p is the number of vanishing moments then the size of a Daubechies wavelet is $2p - 1$ [20]. In particular the wavelet support is $[0, 2p - 1]$ and the scaling function support is $[-p + 1, p]$. The scaling function and wavelet for the Daubechies 2, Daubechies 5 and Daubechies 7 wavelet are plotted in figure D.2. The corresponding multiresolution filters are plotted in figure D.1. The filter coefficients for different Daubechies wavelets can be found in [20] p.g. 159.

6.5.3 Symmlets

Symmlets are designed according to the same criteria as the Daubechies filters but they are selected to be more symmetric [19]. Daubechies wavelets have their energy concentrated at the start of their support. Symmlets also have a minimum support, but are constructed to be more symmetric around the centre of its support. The Symmlets 2 and 5 scaling function and wavelet are plotted in figure D.3.

CHAPTER SEVEN

COHEN'S CLASS CLASSIFIER

The classifier presented in this section builds on the series of classifiers developed in [11–13]. The classifier calculates a Cohen's class time-frequency representation (TFR) with a radially Gaussian kernel for each signal. The classification is performed on this TFR with a ν-SVM classifier. The TFR is furthermore refined to increase the classification accuracy.

7.1 TIME-FREQUENCY REPRESENTATION

The Cohen class TFR can be calculated as (see equation 5.19)

$$P_C^{\phi}(u,\xi) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\phi(\tau,\eta)\,\mathcal{A}_x(\tau,\eta)e^{j(\eta u-\xi\tau)}d\eta d\tau \tag{7.1}$$

where $\phi(\tau,\eta)$ is the kernel function. The above is normalized before classification as

$$NP_C^{\phi}(u,\xi) = \frac{|P_C^{\phi}(u,\xi)|}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}|P_C^{\phi}(u,\xi)|dud\xi}. \tag{7.2}$$

The Radially Gaussian kernel is defined as

$$\phi(\tau,\eta) = \exp\left(\frac{-(\tau^2+\eta^2)}{2\sigma(\theta)^2}\right), \tag{7.3}$$

where $\theta = \arctan\left(\frac{\eta}{\tau}\right)$. The spread function $\sigma(\theta)$ completely defines the radially Gaussian kernel. The search for a kernel that minimises a suitable criterion is therefore the search for the optimal spread function. For the resulting TFR to be real, the spread function should be

π-periodic to ensure that $\phi(\tau,\eta) = \phi^*(-\tau,-\eta)$ (see section 5.4). The kernel should also preferably be smooth (continuous and differentiable) in order to prevent ringing [9]. Since the function is π-periodic it implies that the function should be differentiable from π to 0.

A method is proposed in [10] that uses a truncated Fourier series be used as a spread function. This can be defined as

$$\sigma(\theta) = a_0 + \sum_{i=1}^{p_{max}} [a_i \cos(2i\theta) + b_i \sin(2i\theta)]. \tag{7.4}$$

The motivation for this is that the function will be continuous and π-periodic. Since it is used as a divisor, it is required that

$$\sigma(\theta) \geq a_0 > 0 \tag{7.5}$$

An improvement presented in the next section is to use a Bernstein expansion as a spread function. This provides several advantages with respect to the optimisation (as discussed in section 7.3.2).

7.2 BERNSTEIN BASIS FUNCTIONS

The Bernstein expansion is defined as [59]

$$B_n(t) = \sum_{i=0}^{n} \beta_i B_{i,n}(t) \tag{7.6}$$

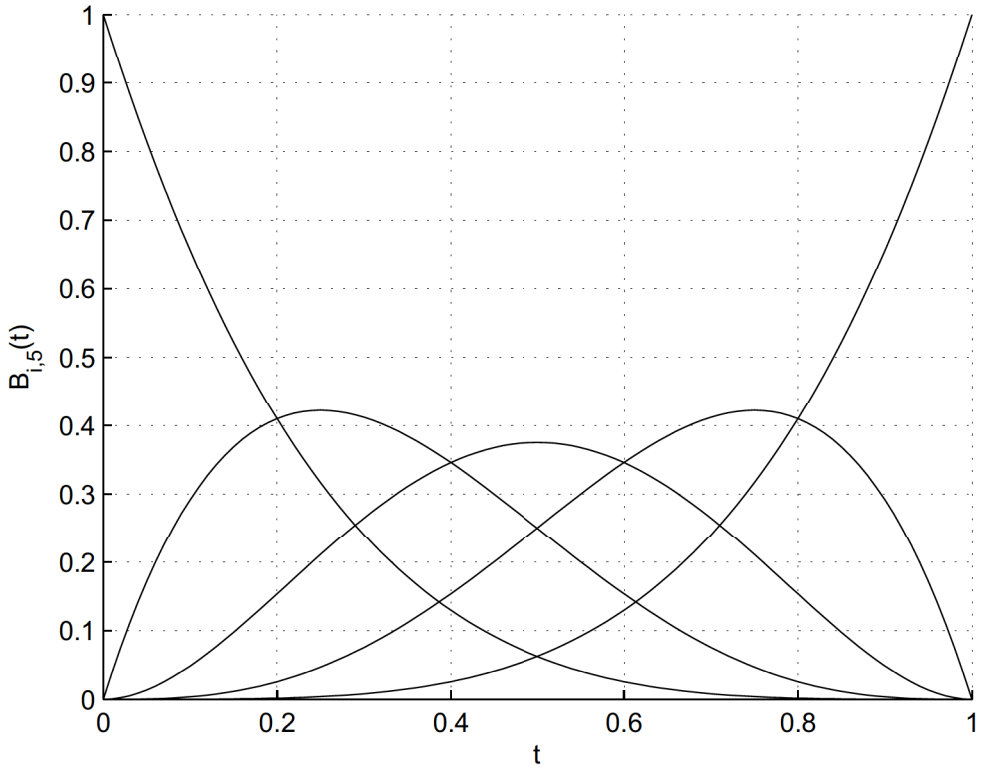
where β_i is the coefficient and $B_{i,n}(t)$ is the Bernstein basis polynomial. The number of Bernstein polynomials used in the above is $n+1$. A Bernstein polynomial can be defined as

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \tag{7.7}$$

where the binomial coefficient is calculated as

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}. \tag{7.8}$$

The highest order of a polynomial in an expansion will therefore be n. Bernstein polynomials were initially used to prove the Weierstrass approximation theorem but found applications in many other fields (e.g. [60] and [61]). A function $f(x)$ can be approximated by setting $\beta_i = f\left(\frac{i}{n}\right)$ [59]. The Bernstein polynomials for $B_{i,4}(t)$ are plotted in figure 7.1.



FIGUR E 7.1: Plot of Bernstein polynomials for $n = 4$. The Bernstein polynomials is from left to right, $B_{0,4}, B_{1,4}, \ldots, B_{4,4}$.

To ensure that the spread function is periodic it will be calculated as

$$\sigma(\theta) = \begin{cases} B_n(\theta/\pi) & 0 \leq \theta < \pi. \\ B_n(\theta/\pi - 1) & \pi < \theta \leq 2\pi. \end{cases} \tag{7.9}$$

The function must be continuous at $\sigma(\pi)$ and differentiable. To ensure differentiability the derivative of the Bernstein expansion must be calculated. The derivative of a Bernstein

polynomial can be can be calculated as

$$\frac{d}{dt}B_{i,n}(t) = \frac{d}{dt}\binom{n}{i}t^i\,(1-t)^{n-i} \tag{7.10}$$

$$= i\frac{n!}{(n-i)!i!}t^{i-1}\,(1-t)^{n-i} + (n-i)\,\frac{n!}{(n-i)!i!}t^i\,(1-t)^{n-i-1}\,(-1) \tag{7.11}$$

$$= i\frac{n(n-1)!}{([n-1]-[i-1])!i(i-1)!}t^{i-1}(1-t)^{[n-1]-[i-1]} - \frac{n(n-1)!}{((n-1)-i)!i!}t^i(1-t)^{(n-1)-i} \tag{7.12}$$

$$= n\binom{n-1}{i-1}t^{i-1}(1-t)^{[n-1]-[i-1]} - n\binom{n-1}{i}t^i(1-t)^{(n-1)-i} \tag{7.13}$$

$$= nB_{i-1,n-1}(t) - nB_{i,n-1}(t). \tag{7.14}$$

The derivative of the Bernstein expansion is therefore

$$\frac{d}{dt}B_n(t) = \sum_{i=0}^{n}\beta_i n\left[B_{i-1,n-1}(t) - B_{i,n-1}(t)\right]. \tag{7.15}$$

The values for the Bernstein polynomial at $t = 0$ is

$$B_{i,n}(0) = \binom{n}{i}0^i(1)^{n-i} \tag{7.16}$$

$$= \begin{cases} 1 & i = 0 \\ 0 & otherwise \end{cases} \tag{7.17}$$

and at $t = 1$ is

$$B_{i,n}(1) = \binom{n}{i}1^i(0)^{n-i} \tag{7.18}$$

$$= \begin{cases} 1 & i = n \\ 0 & otherwise \end{cases} \tag{7.19}$$

To ensure continuity from 1 to 0, $B_{i,n}(0) = B_{i,n}(1)$, the following must hold

$$\beta_0 = \beta_n. \tag{7.20}$$

To ensure differentiability the following must be true

$$\frac{d}{dt}B_n(0) = \frac{d}{dt}B_n(1) \tag{7.21}$$

$$-n\beta_0 + n\beta_1 = n\beta_n - n\beta_{n-1} \tag{7.22}$$

$$\beta_1 - \beta_0 = \beta_n - \beta_{n-1}. \tag{7.23}$$

Both constraints therefore are

$$\beta_0 = \beta_n \tag{7.24}$$

$$\beta_1 - \beta_0 = \beta_n - \beta_{n-1}. \tag{7.25}$$

It is interesting to note that both of these constraints are linear. The spread function can then be written as

$$\sigma(\theta) = \begin{cases} B_n(\theta/\pi) & 0 \leq \theta < \pi. \\ B_n(\theta/\pi - 1) & \pi < \theta \leq 2\pi. \end{cases} \tag{7.26}$$

7.3 CRITERION FUNCTION

The criterion function should approximate the probability of misclassification. The approximation of the probability of misclassification is then minimised by changing the spread function. The parameter vector that is changed in order to minimise the criterion function is the coefficients of the Bernstein approximation (that is used as a spread function)

$$\boldsymbol{\theta} = \begin{bmatrix} \beta_0 & \beta_0 & \dots & \beta_n \end{bmatrix} \tag{7.27}$$

when a linear SVM is used. When a Gaussian kernel is used, the parameters to be optimised is

$$\boldsymbol{\theta} = \begin{bmatrix} \beta_0 & \beta_0 & \dots & \beta_n & \sigma \end{bmatrix}, \tag{7.28}$$

where σ is the width of the Gaussian kernel. The criterion function is similar to the one presented in [10] (in that the normal assumption is used for the distribution of the classes).

The discriminant function that is used for classification is (see equation 4.27)

$$g(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b, \tag{7.29}$$

where the classification result is $sgn(g(\mathbf{x}))$.

The criterion function is calculated as the probability of misclassification using the training set. The training dataset $\mathcal{O}_N$ is divided into two parts, $\mathcal{O}_L$ and $\mathcal{O}_T$ of approximately equal size which contain elements of both classes. The criterion function $K(\boldsymbol{\theta}|\mathcal{O}_N)$ is obtained in the following manner:

1. $\mathcal{O}_L$ is used to train the ν-SVM classifier (the current parameter set $\boldsymbol{\theta}$ is used for the TFR kernel). The set of parameters $\boldsymbol{\alpha}$ and b will be obtained. This can be done with standard SVM training algorithms (such as SMO).

2. The decision function $g(\mathbf{x})$ is evaluated for each signal in the test set ($\mathcal{O}_T$). The number of positive patterns ($y_i = 1$) in the test set $\mathcal{O}_T$ is denoted as T^+ and the number of negative patterns in this set is denoted as T^-.

3. The empirical mean and variance is calculated for each of the two classes

$$\mu_+ = \frac{1}{T_{+1}} \sum_{i=1}^{T_{+1}} g_{+1}^i \tag{7.30}$$

and

$$\sigma_+^2 = \frac{1}{T_{+1}} \sum_{i=1}^{T_{+1}} \left(g_{+1}^i - \mu_{+1}\right)^2. \tag{7.31}$$

The mean and variance for the negative case is calculated in an similar manner.

4. The criterion function for these specific subsets is calculated as (derived in the next section)

$$\mathcal{K}(\boldsymbol{\theta}|\mathcal{O}_T, \mathcal{O}_L) = \frac{1}{2} + \frac{1}{4}\operatorname{erfc}\left(\frac{-\mu_-}{\sqrt{2}\sigma_-}\right) - \frac{1}{4}\operatorname{erfc}\left(\frac{-\mu_+}{\sqrt{2}\sigma_+}\right) \tag{7.32}$$

The assumption that the two classes are normally distributed is motivated in [10]. This

procedure is repeated R times for different subsets $\mathcal{O}_T$ and $\mathcal{O}_L$ of $\mathcal{O}_N$. The final criterion is therefore

$$\mathcal{K}_R\left(\boldsymbol{\theta}|\mathcal{O}_N\right) = \frac{1}{R}\sum_{i=1}^{R}\mathcal{K}\left(\boldsymbol{\theta}|\mathcal{O}_T,\mathcal{O}_L\right) \tag{7.33}$$

where $\mathcal{O}_T$ and $\mathcal{O}_L$ are different random subdivisions of $\mathcal{O}_T$.

Since the above selection of subsets is random, the criterion to be optimised will be a stochastic function. An alternative that was implemented in this dissertation is loosely based on k-fold cross-validation (see [62] for an explanation of cross-validation). For each calculation the dataset is subdivided into n sets (each set containing an equal number of patterns from both classes). The first set is then used to test while the rest is used for training. An example of this for 5 folds is given in figure 7.3. This strategy offers three advantages:

1. Each pattern appears at least once in the test set.
2. The training dataset is much larger than in the previous case (where the complete dataset is subdivided into a training and test set). This is especially advantageous if the dataset is small.
3. The subdivision is not stochastic which means that optimisation methods will work better.

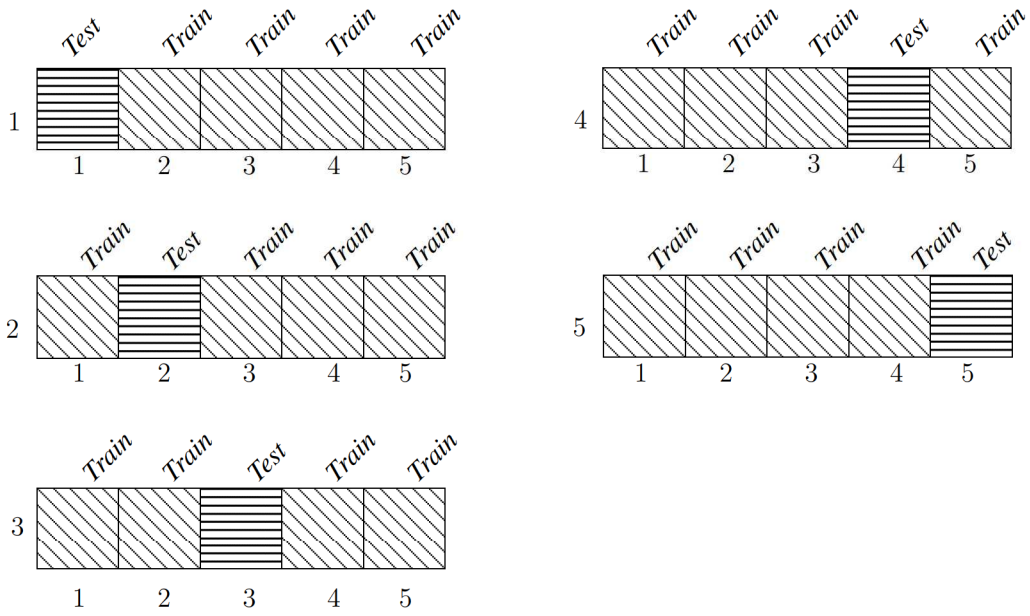


FIGURE 7.2: Cross-validation strategy to calculate the probability of misclassification

7.3.1 Derivation of the criterion function

It is assumed that the pattern $\mathbf{x}$ will be classified as 1 if $g(\mathbf{x}) > 0$ and -1 if $g(\mathbf{x}) < 0$ and that $g(\mathbf{x}_i|y = 1)$ and $g(\mathbf{x}_i|y = -1)$ are normally distributed (see [10] for a motivation). The probability that a pattern will be misclassified is

$$P(misclassification) = \\ P(\hat{\omega} = -1|\omega = 1)\, P(\omega = 1) + P(\hat{\omega} = 1|\omega = -1)\, P(\omega = -1) \tag{7.34}$$

where ω is the true class of the vector and $\hat{\omega}$ is the classified class of the vector. Assume that $p_+(x)$ probability density function of $g(\mathbf{x}|y = 1)$ and $p_-(x)$ is the probability density function of $g(\mathbf{x}|y = -1)$. The error function and complementary error function for the Gaussian distribution is used in the derivation and is defined here as

$$\text{erf}(t) = \frac{2}{\pi}\int_0^t e^{-t^2} dt \tag{7.35}$$

and

$$\begin{aligned} \text{erfc}(t) &= \frac{2}{\pi}\int_t^\infty e^{-t^2} dt & (7.36)\\ &= 1 - \text{erf}(t). & (7.37) \end{aligned}$$

The cumulative distribution function of a Gaussian distribution $N(\mu, \sigma)$ can be written in terms of the error function as

$$F(t) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{t-\mu}{\sqrt{2}\sigma}\right)\right]. \tag{7.38}$$

Using the normal assumption the derivation below simplifies equation 7.34. The probability of misclassification for a pattern of class $\omega = -1$ is

$$\begin{aligned} P(\hat{\omega} = 1|\omega = -1) &= \int_0^\infty p_-(x)\, dx & (7.39)\\ &= 1 - F(0) & (7.40)\\ &= 1 - \frac{1}{2}\left[1 + \text{erf}\left(\frac{-\mu_-}{\sqrt{2}\sigma_-}\right)\right] & (7.41)\\ &= \frac{1}{2}\,\text{erfc}\left(\frac{-\mu_-}{\sqrt{2}\sigma_-}\right). & (7.42) \end{aligned}$$

The probability of misclassification for a pattern of class $\omega = 1$ is

$$\begin{aligned} P(\hat{\omega} = -1|\omega = 1) &= \int_{-\infty}^{0} p_+(x)\,dx && (7.43)\\ &= F(0) && (7.44)\\ &= \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{-\mu_+}{\sqrt{2}\sigma_+}\right)\right] && (7.45)\\ &= \frac{1}{2}\left(2 - \operatorname{erfc}\left(\frac{-\mu_+}{\sqrt{2}\sigma_+}\right)\right) && (7.46)\\ &= 1 - \frac{1}{2}\operatorname{erfc}\left(\frac{-\mu_+}{\sqrt{2}\sigma_+}\right) && (7.47) \end{aligned}$$

The total probability for misclassification is therefore

$$\begin{aligned} P(\text{misclassification}) = P(\omega = -1)&\left[\frac{1}{2}\operatorname{erfc}\left(\frac{-\mu_-}{\sqrt{2}\sigma_-}\right)\right]\\ + P(\omega = 1)&\left[1 - \frac{1}{2}\operatorname{erfc}\left(\frac{-\mu_+}{\sqrt{2}\sigma_+}\right)\right]. \end{aligned} \tag{7.48}$$

The $P(\omega = 1)$ and $P(\omega = -1)$ can be set as $\frac{1}{2}$ if both classes are equally likely. It is important though to use the above formula when the strategy of one-versus-all is used to change the SVM into a multi-class classifier. The case where the two classes are equally likely simplifies the above to

$$P(misclassification) = \frac{1}{4}\operatorname{erfc}\left(\frac{-\mu_-}{\sqrt{2}\sigma_-}\right) + \frac{1}{2} - \frac{1}{4}\operatorname{erfc}\left(\frac{-\mu_+}{\sqrt{2}\sigma}\right). \tag{7.49}$$

7.3.2 Optimisation

The optimisation was performed with the global best PSO algorithm (this is explained in Chapter 3). The optimization should be performed while the linear constraints are met (equations 7.25 and 7.25). A general set of linear equations can be expressed as

$$A\mathbf{x} = \mathbf{b} \tag{7.50}$$

where $\mathbf{x}$ is the vector to be optimized and A and b are the linear constraints. Consider the calculation for the velocity update (equation 3.2)

$$\mathbf{v_i}(t+1) = \mathbf{v_i}(t) + \rho_1(\mathbf{y_i}(t) - \mathbf{x_i}(t)) + \rho_2(\hat{\mathbf{y}}(\mathbf{t}) - \mathbf{x_i}(t)). \tag{7.51}$$

If the initial particle positions satisfy the constraints and the initial velocity is zero then multiplying the velocity update with A gives

$$\begin{aligned} A\mathbf{v_i}(t+1) &= A\mathbf{v_i}(t) + \rho_1 (A\mathbf{y_i}(t) - A\mathbf{x_i}(t)) + \rho_2 (A\hat{\mathbf{y}}(\mathbf{t}) - A\mathbf{x_i}(t)) && (7.52) \\ &= A\mathbf{v}_i + \rho_1 (b-b) + \rho_2 (b-b) && (7.53) \\ &= 0 && (7.54) \end{aligned}$$

With an induction argument it can therefore be proven that if the initial conditions satisfy the constraints and the initial velocity is zero, all subsequent solutions will be zero. This means that as long as the initial particles satisfy the constraints, all subsequent solutions will satisfy the constraints. This is significant since it allows the constraints on the Bernstein coefficients to be added without any additional overhead.

What is also interesting to note is that any intermediate result is therefore simply a linear combination of all the initial starting vectors. This implies that just a subspace of the full vector space is searched. Therefore, to find the optimal solution for an n parameter problem, at least n linearly independent particles must be used. The performance of the classifier is discussed in chapter 10.

CHAPTER EIGHT

WAVELET PACKET CLASSIFIER

8.1 INTRODUCTION

As previously stated, the radar transmitter signal is non-stationary (the frequency content of the signal varies with time). This motivates the use of representations that represent the signal in both time and frequency. As from the discussion in section 1.1, methods for non-stationary signal classification can roughly be divided into three groups:

1. Parametric signal dependent classifiers using a stochastic model of the source.

2. Ad hoc signal dependent models utilising a TFR and a problem dependent feature extraction method. The feature extraction and classifier are usually constructed based on trial and error.

3. Signal independent methods for automatic classification. These also incorporate a feature extraction method but is not tuned to a specific model (an example of this method is given in [18]).

The feature extraction step is usually motivated by the so called "curse of dimensionality" problem many classifiers face. In short, to approximate an unknown probability distribution with a given accuracy, the number of patterns needed increases exponentially with the dimensionality of the feature space [63]. This problem is not just limited to neural networks but also occurs in Bayesian analysis approaches.

Support vector machines do, however, not suffer from this problem. If patterns are inseparable

a kernel function is used to implicitly map a feature vector to a higher dimensional space. The reason for the good performance of SVMs in high dimensional spaces is that methods such as Bayesian analysis attempt to approximate the probability density of the feature space. This, however, is not necessary and often disadvantageous. The central philosophy behind SVMs as stated by V. Vapnik is [64] :

> "When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one"

The following advice is also given in the same text

> "Do not estimate a density if you need to estimate a function. Do not use the classical statistics paradigm for prediction in a high dimensional world: Do not use generative models for prediction."

It is this philosophy that enables SVMs to work well in high dimensional spaces (by avoiding the calculation of a probability density function).

8.1.1 Wavelet Packet Transform

This section will show, with the aid of examples, why the wavelet packet transform is an apt method to extract the time and frequency information. For the normal multiresolution case (figure 6.10) the approximation can be expressed as

$$P_{V_j} = \sum_{n=-\infty}^{\infty} a_j[n] \frac{1}{\sqrt{2^j}} \phi\left(\frac{t-2^j n}{2^j}\right). \tag{8.1}$$

where $a_j[n]$ is the inner product defined as

$$a_j[n] = \left\langle f, \frac{1}{\sqrt{2^j}} \phi\left(\frac{t-2^j n}{2^j}\right) \right\rangle. \tag{8.2}$$

f is approximated at V_0 as the sampled discrete signal.

The local time-frequency energy at $(t = 2^j n, \xi = \frac{\eta}{2^j})$ can be calculated as the energy contributed by the scaling function (some scaling functions are not centred at $t = 0$ – this

however does not change the argument). The energy contributed by the scaling function is therefore (assuming a normalized wavelet)

$$P^E f(t = 2^j n, \frac{\eta}{2^j}) = \int_{-\infty}^{\infty} \left| a_j[n] \frac{1}{\sqrt{2^j}} \phi\left(\frac{t - 2^j n}{2^j}\right) \right|^2 dt \quad (8.3)$$

$$= |a_j[n]|^2 (1) . \quad (8.4)$$

The same approach can be followed for the wavelet function (and the approach can be extended to the wavelet packet transform case). The above function $\frac{1}{\sqrt{2^j}} \phi\left(\frac{t-2^j n}{2^j}\right)$ has a higher time resolution for small j and a higher frequency resolution for large j. The above expression can therefore be used to calculate the energy content of the signal (similar to a spectrogram or scalogram). The wavelet packet tree is given in figure 8.1 to illustrate the time and frequency solution for the WPT.

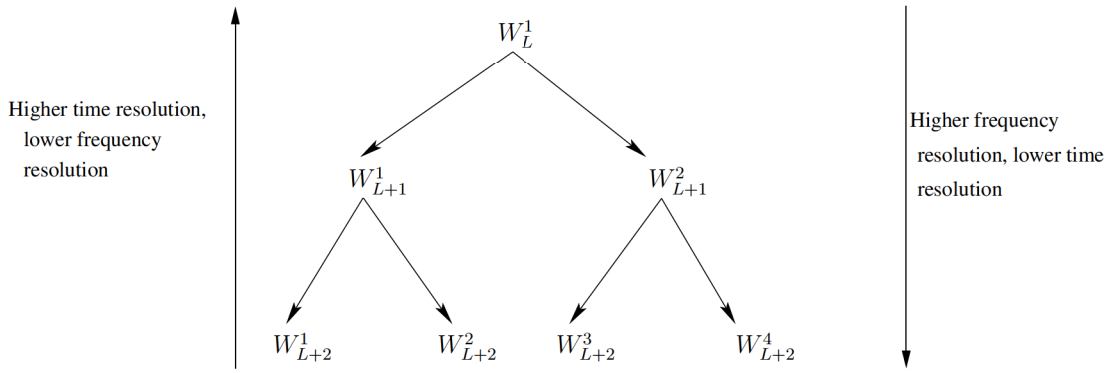


FIGURE 8.1: Time and frequency resolution in the wavelet packet tree.

Due to space considerations the wavelet packet tree will be represented as a matrix. This notation is illustrated in figure 8.2. It is assumed that the signal starts at time T_{start} and ends at T_{end}. The matrix containing the intensities of the wavelet packet tree for a signal $\mathbf{x}$ is denoted as $TFR_W(i, j; \mathbf{x})$. It is assumed that the signal length is extended to be a power of two. Each wavelet packet will therefore have $\frac{Ns}{2^j}$ samples and there will be $\log_2 N_s + 1$ levels. The intensity for $t = 2^j n$ for packet p (using the notation in figure 8.2) can be indexed as $TFR_W\left(j + 1, \frac{Ns}{2^j}(p) + 1 + n\right)$.

The WPT of a cosine signal of 512 samples is given in figure 8.3(a). This example

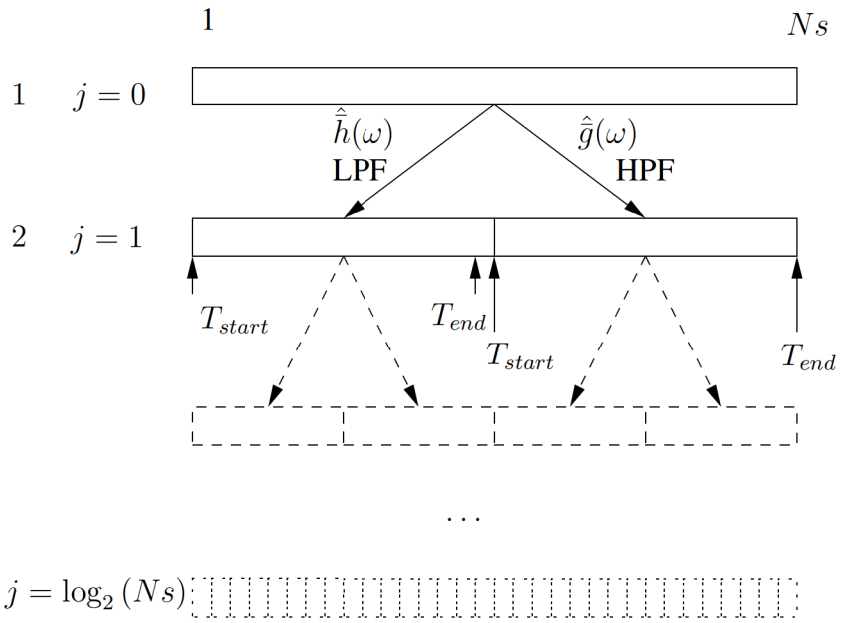
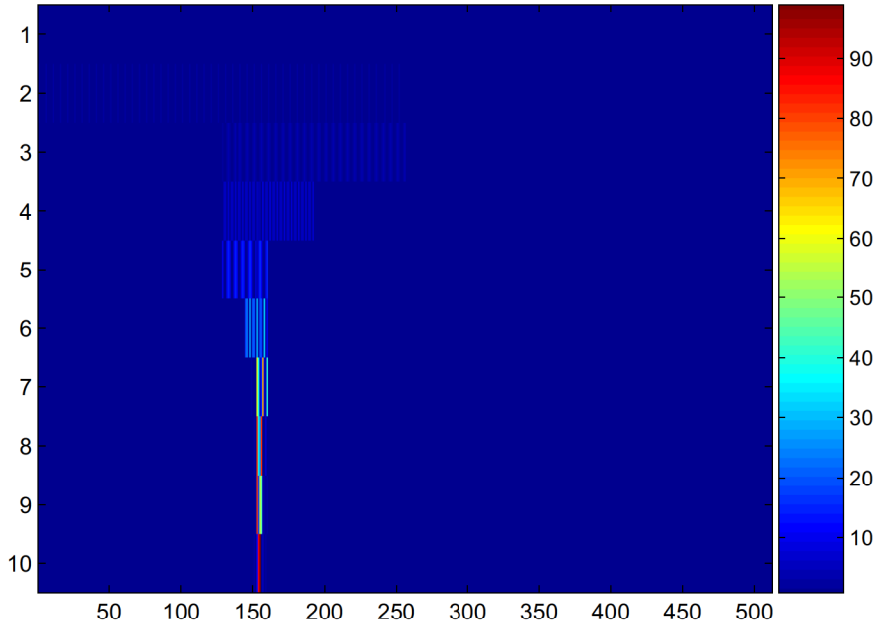


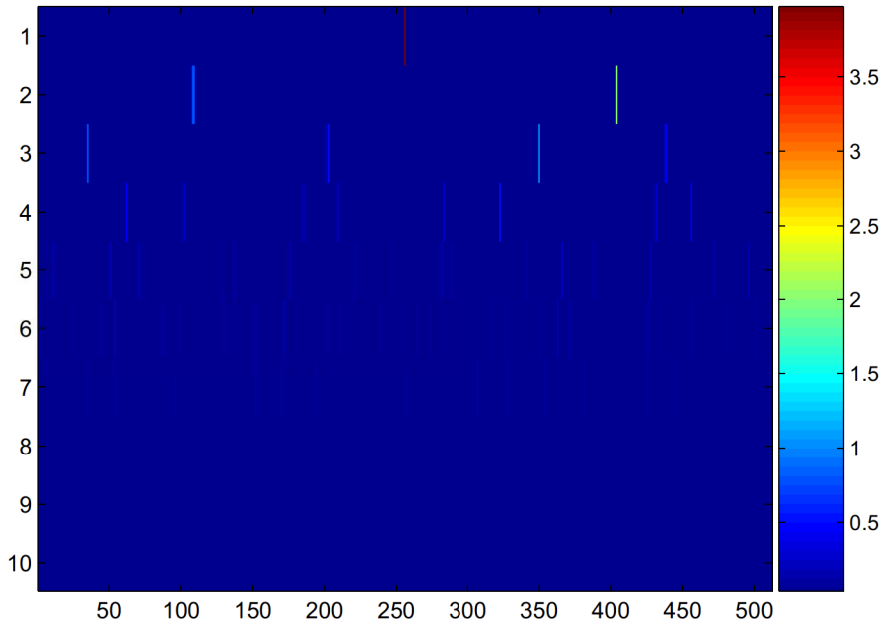
FIGURE 8.2: Description of a wavelet packet tree in a matrix format. j is the row of the matrix and the columns range from $1, 2 \ldots, Ns$.

shows that the signal is concentrated in frequency. Since the lowest levels of the WPT is completely resolved in frequency (since each packet only contains a single sample), the packet with the highest intensity in should therefore be around $\frac{0.15}{0.5} \times 512 = 153.6$. It is clear from this example that two pure cosine signals are separated best at the packets with the highest frequency resolution.

A signal with opposite characteristics is the Kronecker delta. An example of such a signal with a Kronecker delta at $t = 256$ is given in figure 8.3(b). As can be seen this figure the signal is spread over the frequencies lower in the wavelet packet tree. Therefore if two Kronecker delta signals are to be separated the first level would offer the best resolution. These two extreme examples illustrate the advantage of the WPT – the WPT can "focus" to the correct resolution.



(a)



(b)

FIGURE 8.3: Wavelet packet transform of two common signals concentrated in frequency and time respectively. (a) WPT of a cosine signal of 0.15Hz. (b) WPT of a Kronecker delta at $t = 256$. The row and column indices is explained above. Both figures used the Battle-Lemarié 31 filter.

8.1.2 Description of the classifier

The wavelet packet transform is calculated for each signal . An SVM is used with the WPT representation for each signal to be classified. A linear kernel for an SVM can be defined as

$$k_{linear}(\mathbf{x},\mathbf{x}^{'}) = \sum_{i=1}^{\log_2 Ns+1}\sum_{j=1}^{Ns} TFR_{WV}(i,j;\mathbf{x})TFR_{WV}(i,j;\mathbf{x}^{'}). \tag{8.5}$$

The discriminant function for the SVM can then be defined as

$$g(\mathbf{x}) = \sum_{i=1}^{\ell}\alpha_i y_i k_{linear}(\mathbf{x},\mathbf{x}_i)+b. \tag{8.6}$$

The discriminant function can either be calculated with the C-SVM or ν-SVM algorithm.

8.1.2.1 Other kernel functions

A kernel function was also defined that combined second order polynomial kernels for each packet. The kernel was therefore defined as

$$k\left(\mathbf{x},\mathbf{x}^{'}\right) = \sum_{i=1}^{p}\left(\left\langle \mathbf{x}_i,\mathbf{x}_i^{'}\right\rangle +1\right)^2 \tag{8.7}$$

where $\mathbf{x}_i$ is the ith wavelet packet and p is the total number of wavelet packets (it is easy to show, by using the rules of 4.4.1.1, that the above is a proper kernel). This construction of a kernel was done in an attempt to exploit correlations in the energy of each position. This however did not result in an improvement in the classification accuracy.

Another kernel was defined along the lines of the locality improved kernel in [2, 38, 65]. The calculation of the resulting kernel is illustrated in figure 8.4. The point-wise product is taken between $\mathbf{x}$ and $\mathbf{x}^{'}$. This is then sampled with a pyramidal receptive field centered at each location [2]. It can be expressed as

$$z_i = \sum_j w\left(|i-j|\right)\left(\mathbf{x}.*\mathbf{x}^{'}\right)_j \tag{8.8}$$

It is suggested in [2] to use a weighting function as $w(n) = \max(p-n,0)$ where p is the width of the receptive field. It is further suggested that z_i can be raised to a power d_1 and the sum of

z_i can be raised to a power d_1. The locality improved kernel was implemented for each wavelet

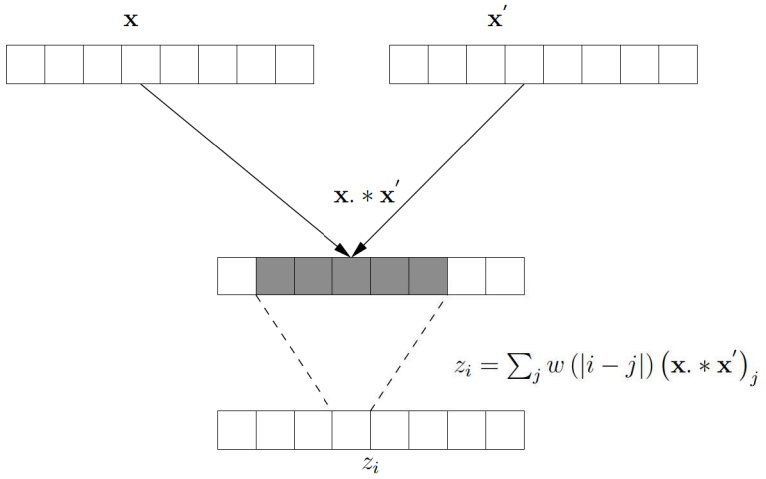


FIGURE 8.4: Locality improved kernel with a pyramidal receptive field.

packet and the result summed (as in the previous section). This however did not result in an increase in the classification accuracy.

8.1.2.2 Combining multiple signals

Several non-stationary signal classification problems involve multiple signals for each pattern. A good example of such a class of problems are several electroencephalography (EEG) classification problems. Each EEG pattern contains multiple non-stationary signals – each of these signals represent different physical positions of electrodes and are recorded at the same time. A recently developed SVM algorithm called multiple kernel learning (MKL) allows for the combination of multiple kernels [66]. The resulting kernel can be expressed as [43]

$$k_M(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{K_c} \beta_k k_k(\mathbf{x}, \mathbf{x}') \tag{8.9}$$

with $\beta_k \geq 0$ and $\sum_{k=1}^{K_c} \beta_k = 1$. In the above each kernel k_k uses a distinct set of features. In the multiple signal problem with k signals we can define each $k_{linear}(\mathbf{x}, \mathbf{x}')$ as

$$k_k(\mathbf{x}, \mathbf{x}') = k_{linear}(\mathbf{x}_{sig_k}, \mathbf{x}'_{sig_k}) \tag{8.10}$$

where $\mathbf{x}_{sig_k}$ is the kth signal of the pattern $\mathbf{x}$ and k is defined in equation 8.6. The resulting discriminant function would therefore be

$$g(\mathbf{x}) = \sum_{i=1}^{\ell} k_M(\mathbf{x}, \mathbf{x}_i) + b. \tag{8.11}$$

Several efficient algorithms have been developed to calculate the weight vector $\boldsymbol{\beta}$ and the $(\boldsymbol{\alpha}, b)$ values (see for instance [43]). The Shogun toolbox includes an implementation of multiple kernel learning [43].

8.1.2.3 Incorporating prior information

Certain properties of the signal may be known beforehand. For some types of signals (e.g. radar pulses) it may be a requirement for the classifier to be invariant to small translations of input signals. A well known method to incorporate this is to generate a virtual training set. This training set contains the original training data and translations of the training set. This method unfortunately creates an extremely large training sets.

A different technique known as the virtual SV method can overcome this problem [38]. The classifier is trained with the training set. After the training is complete, the translation is applied to only the support vectors. The SVM is then retrained on the generated dataset and the training dataset. The motivation for this is that the SV set generally only encompasses a fraction of the training set. These vectors also support the hyperplane therefore translations of these vectors are more likely to change the classification boundary.

Another method to incorporate prior information is to use data that is generated from a similar source to the two sources that are to be classified. Such a method is discussed in the next chapter.

CHAPTER NINE

WAVELET PACKET-UNIVERSUM CLASSIFIER

9.1 INTRODUCTION

In the classical binary classification setup, the training of a classifier is performed on a dataset $((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell))$ where $\mathbf{x}_i$ is the pattern vector and $y_i \in \{-1, 1\}$ is the class label. One method used to improve the classification performance is semi-supervised learning (SSL) [67]. In this framework a second dataset is used. This dataset originates from the same two classes but is unlabelled. The unlabelled training dataset is usually a lot larger than the training dataset. The motivation for this is that the cost of labelling a pattern for training may be high or the labels are simply not available. The performance improvement in this framework is due to the clustering assumption (patterns in the same clusters are likely to be of the same class [67]). An an example of this, consider the general problem of e-mail spam classification. A relatively small dataset of labelled e-mails is available (since human effort is required to perform the labelling). The unlabelled dataset is much larger since it does not require any human effort to create.

A new classification framework was introduced in [1]. In this framework the second dataset (known as the Universum) is in the same domain as the classes of interest but not from the two classes. To make this clear, consider the a classification task between two faces (from person A and person B). The size of this dataset is very small. Another dataset consisting of images of faces of people not in those two classes is then used to improve the classification (i.e. person C,

D, ...).

9.2 DEFINITION OF THE UNIVERSUM PROBLEM

The Universum classifier in [1] is developed along the lines of the classic C-SVM classifier (described in section 4.3.1). The C-SVM problem can be rewritten with the aid of a hinge loss function as [1]

$$\min_{\mathbf{w},b} \frac{1}{2} |\mathbf{w}|^2 + C \sum_{i=1}^{\ell} H_1 \left[y_i g_{\mathbf{w},b}(\mathbf{x}_i) \right] \tag{9.1}$$

where $H_\theta[t] = \max(0, \theta - t)$ (note that the C parameter is not divided by ℓ in this definition). From theoretical reasoning (discussed in [1]) the hyperplane with the maximum number of contradictions is preferable. If $f(\mathbf{x}_i)$ is close to zero a small change in f will cause a contradiction. This leads to adding a term $C_{\mathfrak{U}} \sum_{i=1}^{m} U_\varepsilon[f(\mathbf{x}_i)]$ to the above expression. $U_\varepsilon[t]$ is called the ε-insensitive loss and is defined as $U_\varepsilon[t] = H_{-\varepsilon}[t] + H_{-\varepsilon}[-t]$ [1]. The problem can therefore be written as

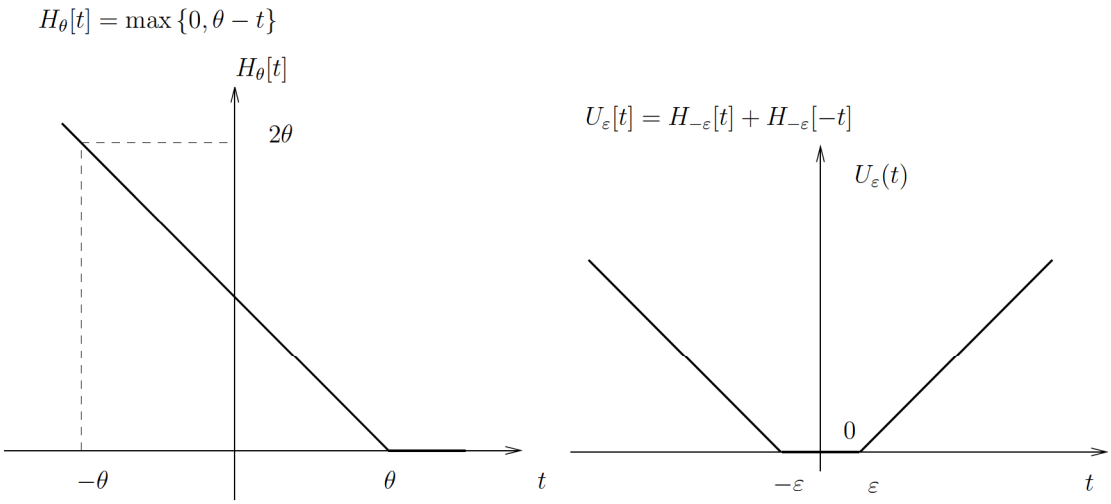
$$\min_{\mathbf{w},b} \frac{1}{2} |\mathbf{w}|^2 + C \sum_{i=1}^{\ell} H_1 \left[y_i g_{\mathbf{w},b}(\mathbf{x}_i) \right] + C_{\mathfrak{U}} \sum_{i=1}^{\ell_{\mathfrak{U}}} U_1 \left[g_{\mathbf{w},b}(\mathbf{x}_i^*) \right] \tag{9.2}$$

where $\mathbf{x}^*$ are the Universum patterns. The Hinge and the ε-insensitive loss function is illustrated in figure 9.1. This problem is a convex optimization problem that can be solved in its dual. Details regarding this can be found in [1]. Selection of the Universum samples is discussed in [68] and [69].

9.2.1 Example problem

An example is used in this section to illustrate a scenario where the Universum can be useful. Consider the problem where each class is specified by a radius that is generated from a uniform distribution $r \sim U(0, 10)$. For each pattern generated by the class, an angle is calculated as $\theta \sim U(0, 2\pi)$. The pattern is then generated as $\mathbf{x} \sim N(\boldsymbol{\mu}, \Sigma)$ where

$$\boldsymbol{\mu} = r \begin{bmatrix} sin(\theta) \\ cos(\theta) \end{bmatrix}, \tag{9.3}$$



(a) Hinge Loss (b) ε-insensitive loss

FIGURE 9.1: Hinge loss and ε-insensitive loss functions [1, 2]

and

$$\Sigma = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}. \tag{9.4}$$

Figure 9.2 shows 20 patterns of a class with $r = 3.0$. The advantage of using the Universum will be the greatest for problems with a few training vectors. An example of this is figures 9.3 and 9.4. In this problem two classes were generated as described above with 4 patterns per class. The decision boundary is given in figure 9.3. The same problem with a decision boundary generated using the Universum is given in figure 9.4. For these two figures $\frac{C}{\ell} = 10$ and $\frac{C_{\mathfrak{U}}}{\ell_{\mathfrak{U}}} = 1$ (the Universum examples will therefore only effect the final solution in a small way). It can be seen that the first figure will have a high misclassification rate – since the patterns will be generated as a Gaussian with a mean on the circle and the classifier classifies the bottom half as a single class.

The improvement in classification accuracy for various values of $C_{\mathfrak{U}}/\ell_{\mathfrak{U}}$ and training sizes for the problem is given in figure 9.5 (calculated by subtracting the classification accuracy without using the Universum from the classification accuracy using the Universum). From the figure it is clear that the increase in classification accuracy increases for extremely small values of $C_{\mathfrak{U}}/\ell_{\mathfrak{U}}$. It is also clear from this that the Universum values should have a very low weight (i.e. $C_{\mathcal{U}}/\ell$ should have a small value). The Universum also seems to be most beneficial

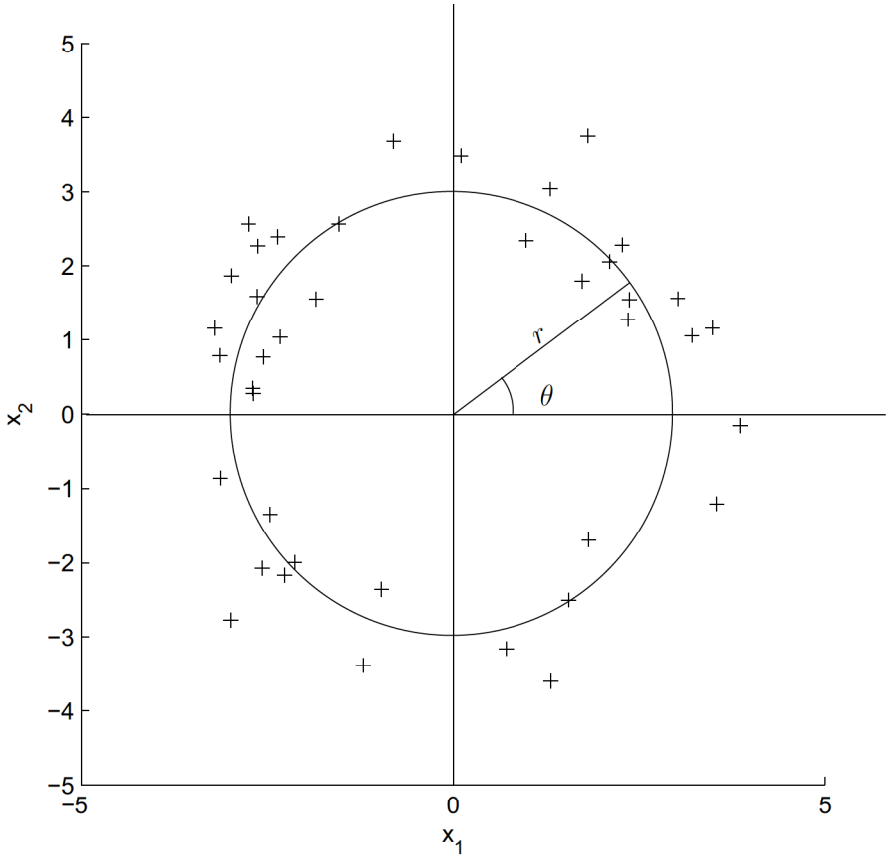


FIGURE 9.2: An example of a patterns of a test class.

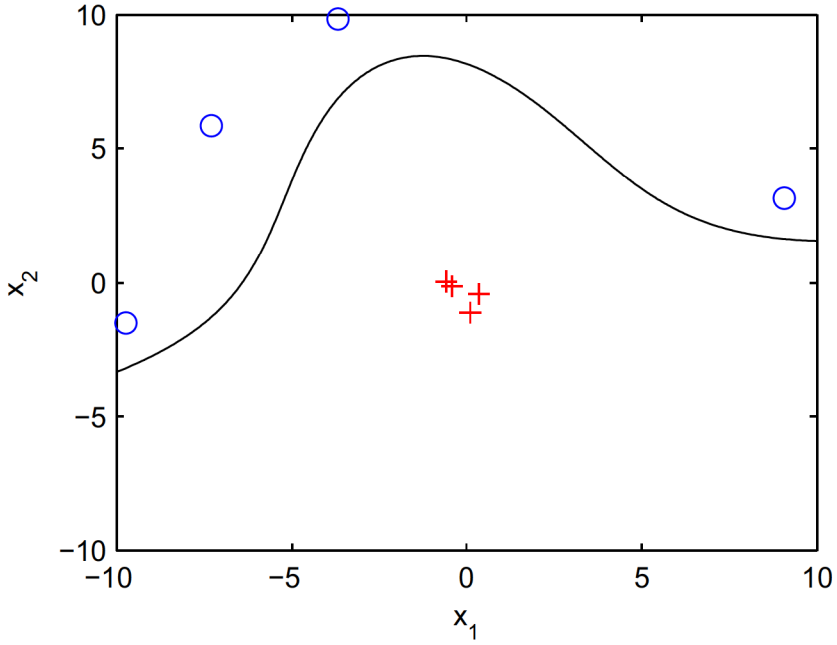


FIGURE 9.3: Classification boundary without the Universum.

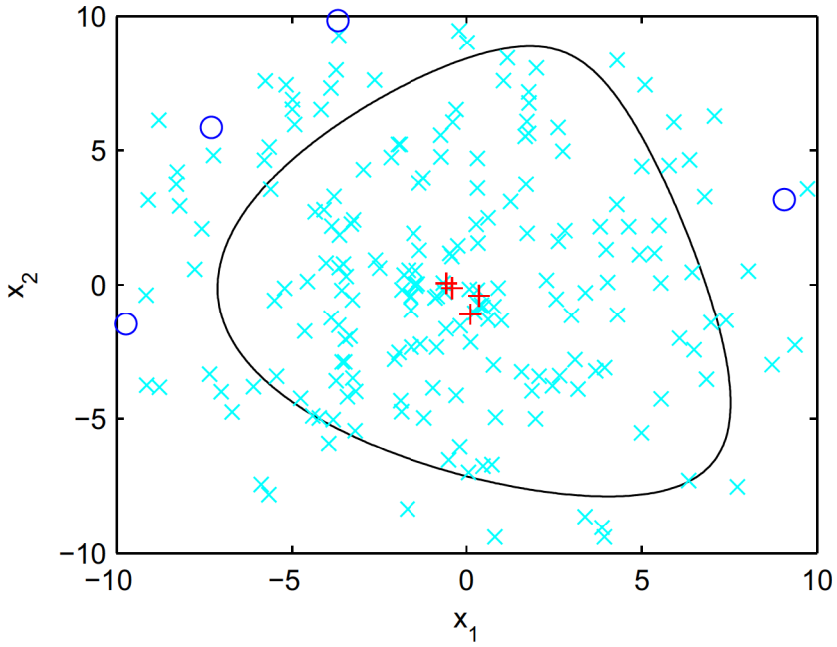


FIGURE 9.4: Classification boundry for the problem with the Universum. Universum examples are given in cyan.

when the training set is small. If the training set becomes larger, the weight of $C_{\mathfrak{U}}/\ell_{\mathfrak{U}}$ should be reduced.

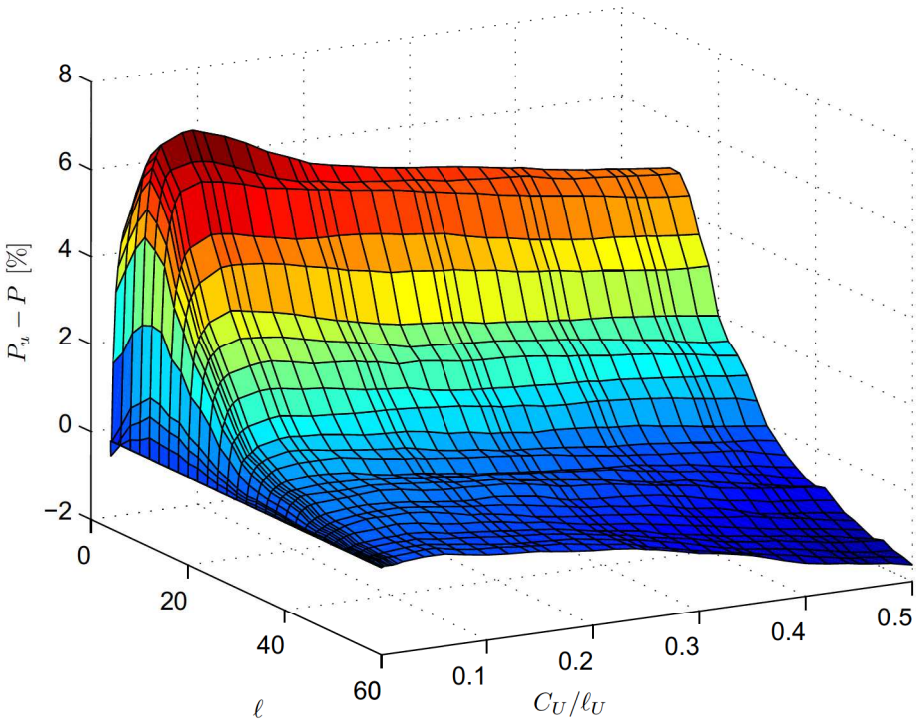


FIGURE 9.5: Improvement of classification accuracy for various training sizes (ℓ) and values of $\frac{C_{\mathfrak{U}}}{\ell_{\mathfrak{U}}}$. The size of the Universum dataset is kept constant at 200 samples.

9.3 DESCRIPTION OF THE WPT-UNIVERSUM CLASSIFIER

The WPT-Universum classifier is the same as the WPT-SVM classifier described in the previous section except for the fact that the Universum classifier is used instead of the ν-SVM classifier. For most signals it was found that the Gaussian kernel achieved the best performance. This is interesting since experiments in [1] (digit, character and document classification) also show better performance when Gaussian kernels are used. The width of the Gaussian kernel was calculated using a strategy similar to cross validation. The training dataset is divided into n datasets. Each of these sub-datasets is used as a test dataset while the rest are used for training the classifier (see figure 5.2). The average classification accuracy using this strategy is calculated and the Gaussian's width optimised according to this (using a line search). The process of searching for a kernel width is however computationally costly.

For the radar transmitter problem, the two transmitters that will be classified are treated as the two classes. Other transmitters of the same make and model are treated as the Universum. This is an apt model since in practical situations the models will not be known. The results for this is presented in the next chapter.

CHAPTER TEN

SIMULATION RESULTS

10.1 TEST PROBLEMS

This section describes three test problems against which the classifiers were evaluated. The first is an EEG classification problem. This is a non-stationary signal that has a wide spectrum. The second problem is the classification of a model that generates cubic and exponential chirps. This is a model that generates models (i.e. each source is created according to a probability distribution). The chirps are also multi-component. The third problem is the radar transmitter classification problem using the model described in Chapter 2.

10.1.1 EEG data

An EEG dataset described in [70] was selected to test the classifier. The motivation for this was twofold:

1. The dataset is large and well studied which facilitates a reasonable comparison between the classifiers developed here with other classifiers in literature.

2. EEG data is non-stationary and the generating model is unknown.

The dataset contains five categories of a 100 segments each (from a single channel). Two of these categories were used in the classification experiment. The A category contained EEG measurement of healthy volunteers with eyes open. Categories C, D and E was from a presurgical diagnosis for epilepsy. The dataset E contained data from an epileptic patient in seizure. For the problem, classification was performed between category A and E. The dataset

was recorded at 173.61Hz and each segment is 23.6 seconds long (giving a total of 4097 samples which were truncated to 4096). The testing procedure for different studies unfortunately differ so the results between studies are not directly comparable. The testing procedure for each study was exactly reproduced to compare it with the classifiers developed in this dissertation.

10.1.2 Cubic-Exponential chirp problem

10.1.2.1 Chirp definitions

The equations for a linear chirp, cubic chirp and exponential chirp is developed in this section. The instantaneous frequency for a linear chirp can be expressed as

$$\omega_i(t) = \omega_0 + \beta t. \tag{10.1}$$

If the start and end time and frequencies are specified by $(0, \omega_0)$ and (t_f, ω_f) the β parameter can be calculated as

$$\beta = \frac{\omega_f - \omega_0}{t_f}. \tag{10.2}$$

The chirp is then calculated as

$$\phi(t) = A\cos\left(\omega_0 t \frac{1}{2}\beta t^2 + \psi_s\right), \tag{10.3}$$

where ψ_s is the starting phase of the cosine. The instantaneous frequency of a cubic chirp is defined as

$$\omega(t) = a\left[2\left(\frac{t}{t_f} - \frac{1}{2}\right)\right]^3 + \omega_c, \tag{10.4}$$

where t is the time step and t_f is the final time. The start and end values for the instantaneous frequency will therefore be

$$\omega(0) = -a + \omega_c \tag{10.5}$$

$$\omega(t_f) = a + \omega_c \tag{10.6}$$

$$\omega(\frac{1}{2}t_f) = \omega_c. \tag{10.7}$$

The chirps can therefore be specified by the center frequency ω_c and the final frequency ω_f. The a value can be calculated as $a = \omega_f - \omega_c$. The signal can then be generated as

$$\phi(t) = A\cos(2a\left(t/t_f - \frac{1}{2}\right)^4 t_f + \omega_c t + \psi_s) \tag{10.8}$$

Alternatively the cubic chirp can be specified by specifying the start and end values. ω_c and a are then calculated as

$$\omega_c = \frac{w_s + w_f}{2} \tag{10.9}$$

$$a = \frac{w_f - w_s}{2}. \tag{10.10}$$

The instantaneous frequency of an exponential chirp can be defined as (where t is normalized as above)

$$\omega_i(t) = a\exp\left(bt/t_f\right). \tag{10.11}$$

When the chirp is specified by its start and end frequencies it the parameters can be calculated as

$$a = \omega_i(0) \tag{10.12}$$

$$b = \log_e \frac{\omega_i(t_f)}{a}. \tag{10.13}$$

The final chirp can be expressed as

$$\phi\left(t\right) = A\cos\left(\frac{at_f}{b}\exp(bt/t_f) + \psi_s\right). \tag{10.14}$$

10.1.2.2 Cubic-exponential chirp model definition

The cubic-exponential chirp problem was created to be a simple test alternative for the transmitter model. The problem is similar to the radar transmitter model in that a model generates different models.

10.1.2.3 Model definition

The parameters of each model is generated as random variables. This model will then generate signals randomly according to the generating parameters. Each model will therefore differ from other models and each signal generated by a specific model will differ from other signals generated by the same model. The model is a multi-component non-stationary signal consisting of a cubic chirp and an exponential chirp.

The means for the start and endpoint for the cubic chirp in the model is generated as

$$\mu_{cs} \sim U(0.05, 0.25) \tag{10.15}$$
$$\mu_{cf} \sim U(0.35, 0.45). \tag{10.16}$$

For the exponential component the start and end values means are generated as

$$\mu_{es} \sim U(0.1, 0.3) \tag{10.17}$$
$$\mu_{ef} \sim U(0.3, 0.45) \tag{10.18}$$

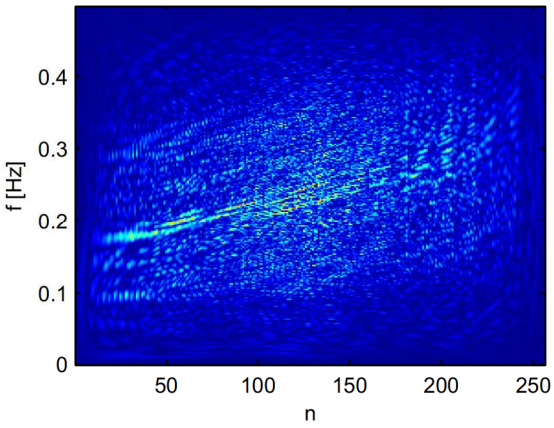
For each signal, the start and end values for the cubic chirp are generated as

$$\omega_s \sim N(\mu_{cs}, \sigma_{cs}) \tag{10.19}$$
$$\omega_f \sim N(\mu_{cf}, \sigma_{cf}) \tag{10.20}$$

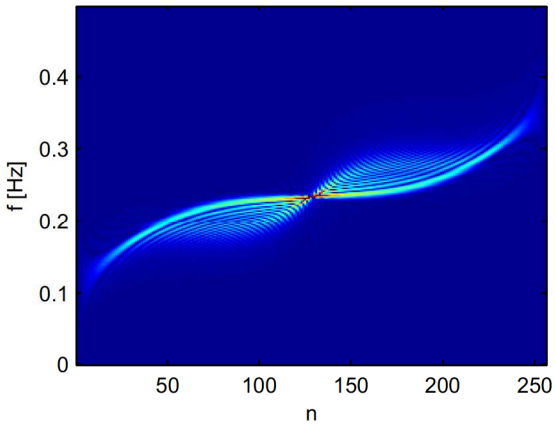
and for the exponential chirp this is generated as

$$a \sim N(\mu_{es}, \sigma_{es}) \tag{10.21}$$
$$\omega_f \sim N(\mu_{ef}, \sigma_{ef}). \tag{10.22}$$

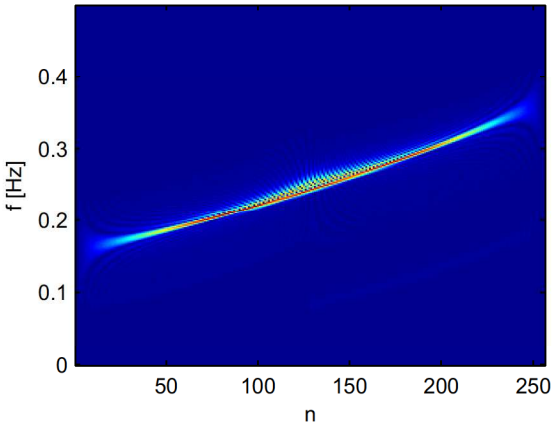
The σ values are set to 0.025. All the start and end instantaneous frequencies are clamped to between 0 and 0.5 if they are out of this range. Gaussian noise is added with $\sigma_n^2 = 1$. An example of the WVD of a signal generated by such a model is given in figure 10.1(a). The cubic, exponential and noise components are separately given in figures 10.1(b), 10.1(c) and 10.1(d). Each training set consists two models generated in the manner explained above. 80 signals in total are generated in this way (equally divided as 40 from each class).



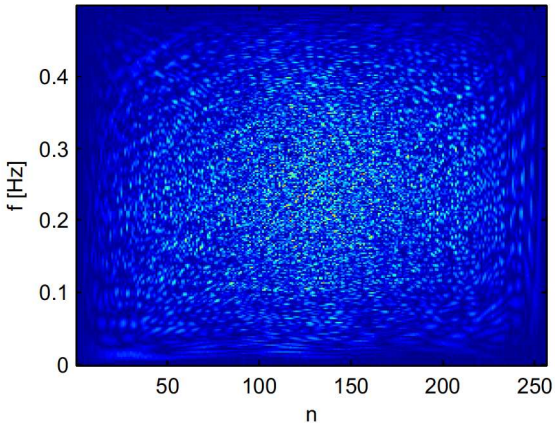
(a) WVD of a generated signal. Sum of an exponential chirp, cubic chirp and noisc.



(b) WVD of the cubic chirp



(c) WVD of the exponential chirp.



(d) WVD of the AWGN noise.

FIGURE 10.1: Plot of the abosolute WVD values of the components of the Cubic-Exponential chirp problem.

10.1.3 Radar transmitter problem

Radar pulses were generated according to the model in Chapter 2. The number of sections used in the pulse-forming network was 7. The pulse width were $1.5\,\mu s$. The pulse height was $5.8\,kV$. The tolerances on the capacitors and inductors were 5% from model-to-model. R_ℓ was set to $1450\,\Omega$. The center frequency was set to $900\,Mhz$ and the frequency pulling was set to $0.6\,Mhz$ in the range of $899.4\,Mhz$ and $900.6\,Mhz$. Each pulse was sampled to produce a length of 128 samples.

10.2 COHEN'S CLASS CLASSIFIER RESULTS

10.2.1 Cubic-exponential chirp problem

The Cohen's class classifier achieved an accuracy of 79.16% on the Cubic-exponential chirp problem.

10.2.2 Radar transmitter model

The classification accuracy for the Cohen's class classifier on the radar transmitter problem is given in figure 10.2.

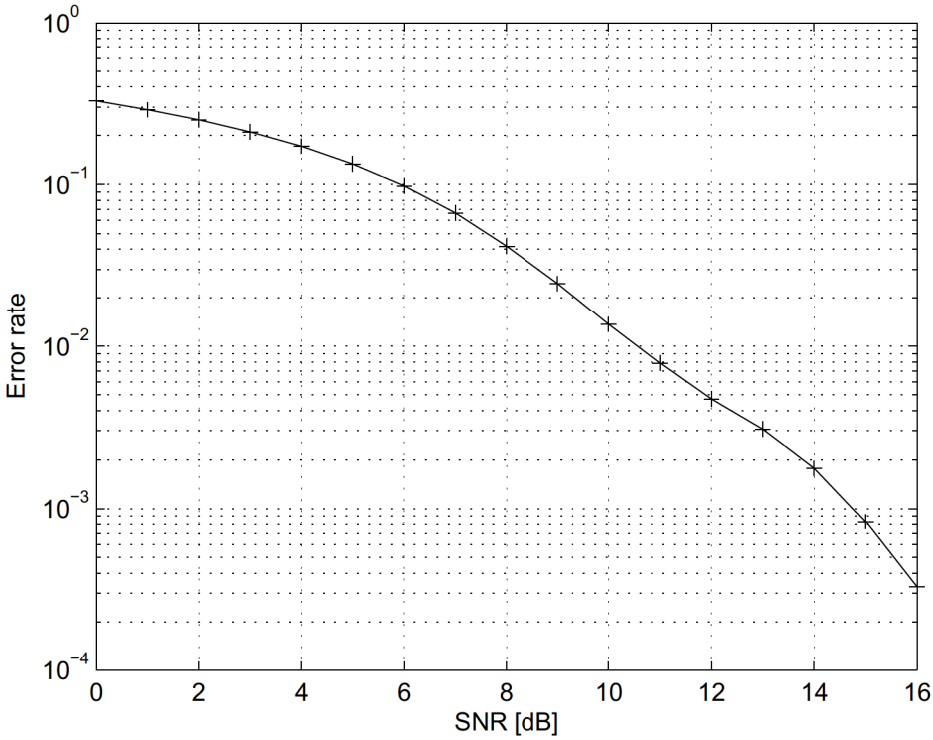


FIGURE 10.2: Classification accuracy for the radar transmitter model

10.3 WPT-SVM CLASSIFIER

10.3.1 EEG data

10.3.1.1 Comparison one

The study in [3] used the discrete wavelet transform to calculate the relative wavelet energy (calculated from the wavelet coefficients). An ANN was used to perform the classification of the EEG signals. The classification accuracy was determined by randomly subdividing the dataset of 200 segments into a testing and training dataset of a 100 segments each. This was performed repeatedly and the average classification accuracy was calculated for this. The performance of the WPT-SVM classifier that was developed is given in figure 10.3. The performance of the best and worst wavelet in the WPT-SVM classifier is given in table 10.1.

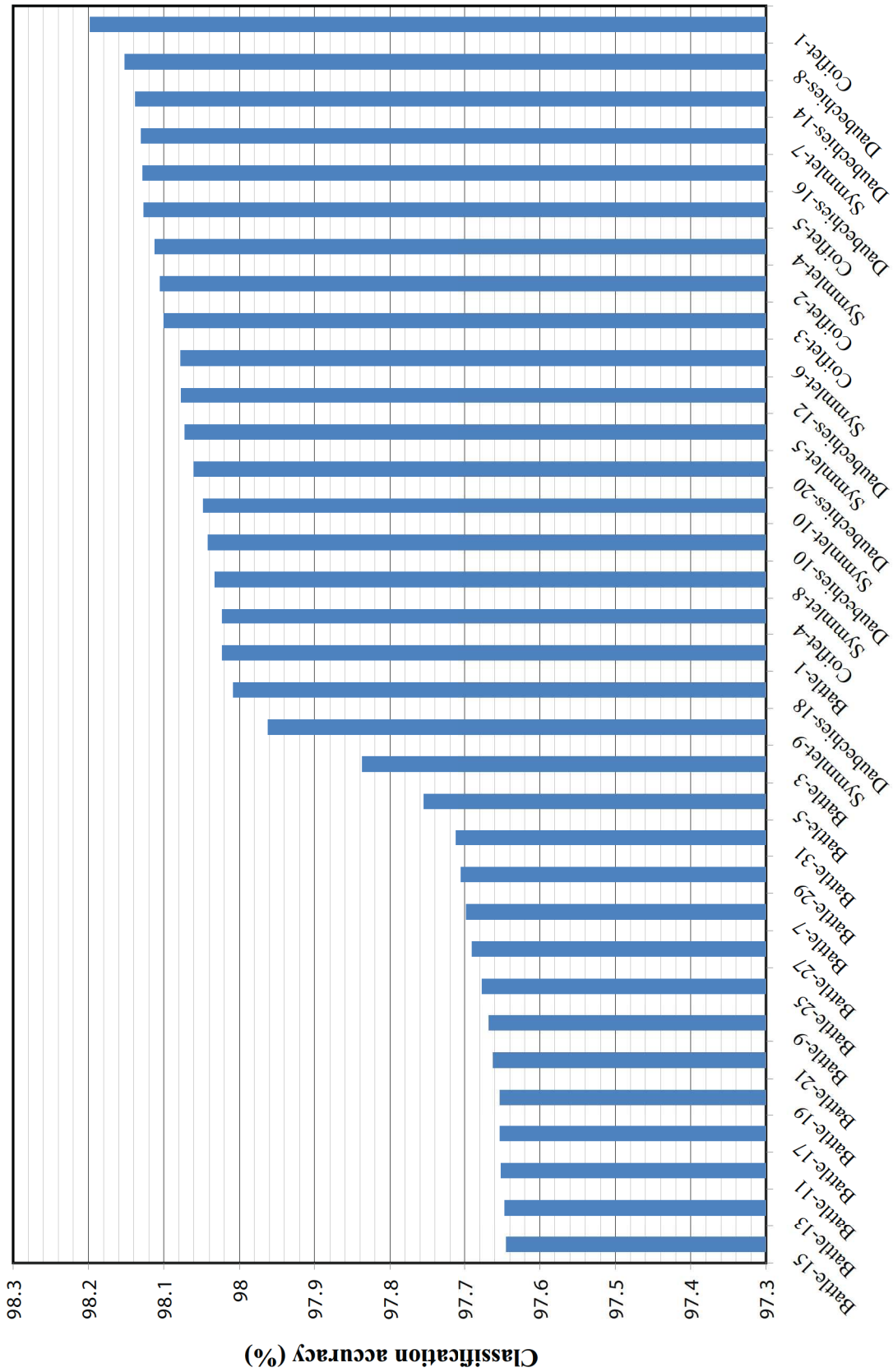


FIGURE 10.3: Classification accuracy for the WPT-SVM classifier on the EEG problem using the testing procedure of [3].

TABLE 10.1: Performance of the WPT-SVM classifier compared to the classifier in [3].

Classifier	**Classification Accuracy (%)**
Guo, Rivero, et. al. [3]	95.20
WPT-SVM Battle-15 wavelet (*Worst*)	97.65
WPT-SVM Coiflet-1 wavelet (*Best*)	98.20

10.3.1.2 Comparison two

The study [4] used a different testing methodology. Each EEG section of 4096 is split up into sixteen 256 sample length signals. These are then in turn treated as independent samples. 5 and 10 fold cross-validation was then used to calculate the performance (an overview of k-fold cross validation can be found in [62]). The use of cross validation ensures that the training set would be larger (when compared to the 50%-50% training and test set division in the previous section). This will therefore increase the classification accuracy. The classification accuracy for the WPT-SVM classifier for the 5-fold cross validation is given in figure 10.4 and for the 10-fold cross validation in figure 10.5. The performance of the best and worst case classification of these and [4] is given in table 10.2.

TABLE 10.2: Performance of the WPT-SVM classifier compared to the classifier in [4].

5-fold cross-validation	
Classifier	*Classification Accuracy (%)*
Polat & Günes [4]	98.68
WPT-SVM Battle-15 wavelet (*Worst*)	99.63
WPT-SVM Coiflet-1 wavelet (*Best*)	99.75
10-fold cross-validation	
Classifier	*Classification Accuracy (%)*
Polat & Günes [4]	98.72
WPT-SVM Battle-13 wavelet (*Worst*)	99.63
WPT-SVM Symmlet-10 (*Best*)	99.78

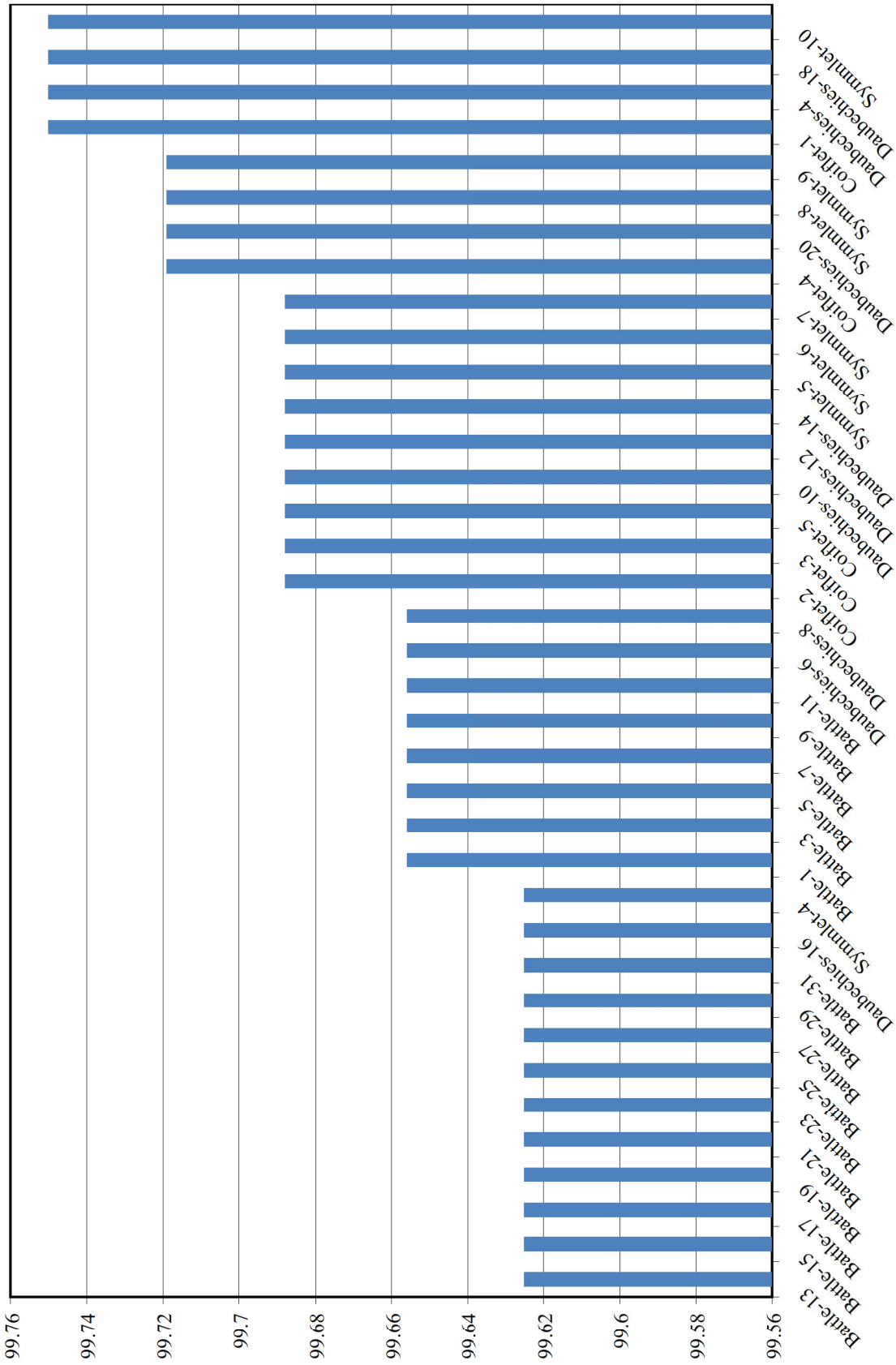


FIGURE 10.4: Classification accuracy for the WPT-SVM classifier on the EEG problem using the 5-fold cross-validation testing procedure of [4].

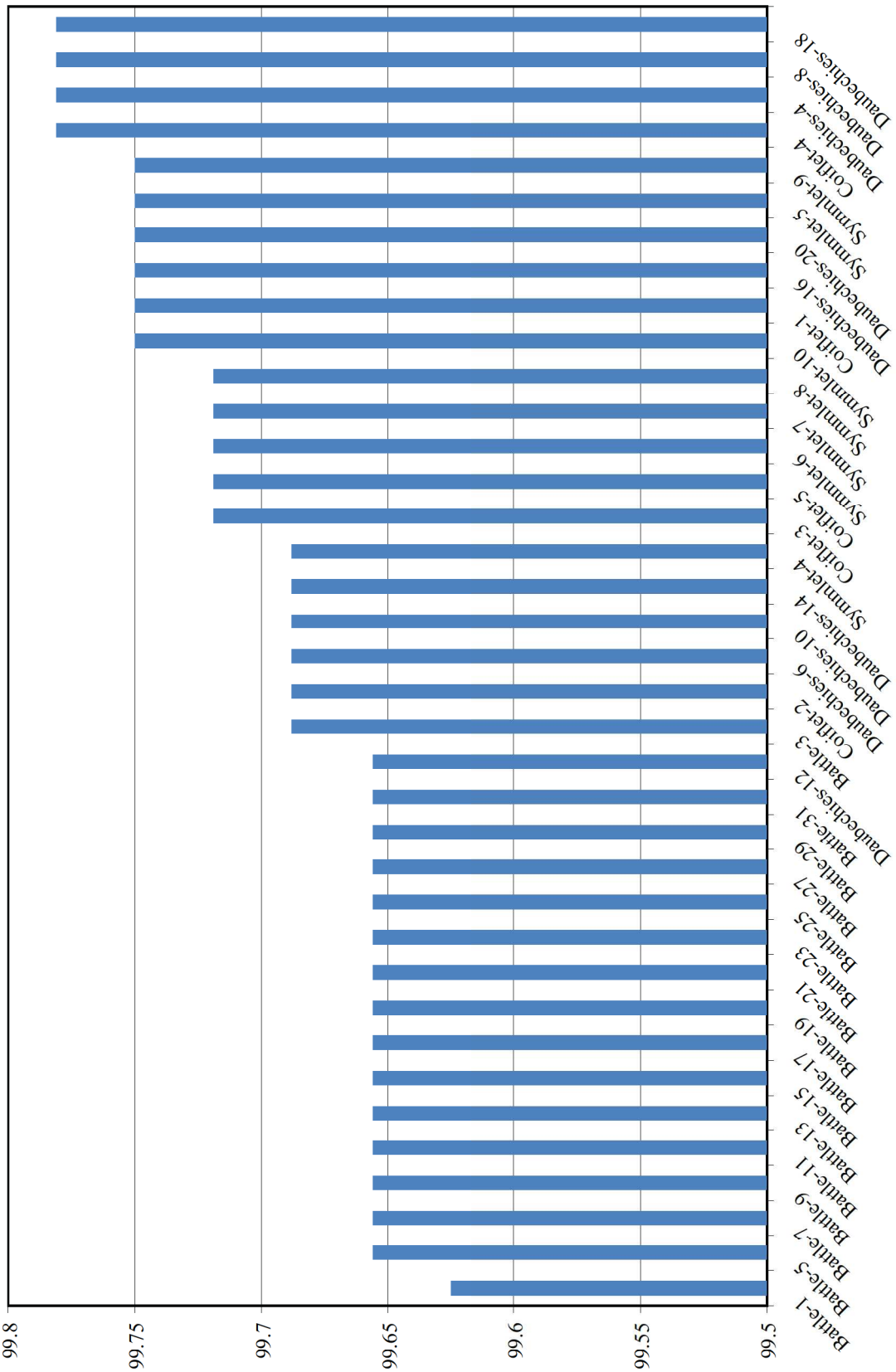


FIGURE 10.5: Classification accuracy for the WPT-SVM classifier on the EEG problem using the 10-fold cross-validation testing procedure of [4].

10.3.2 Cubic-exponential chirp problem

The classification accuracy of the WPT-SVM classifier on the cubic-exponential chirp problem (described in section 10.1.2.3is given in figure 10.6.

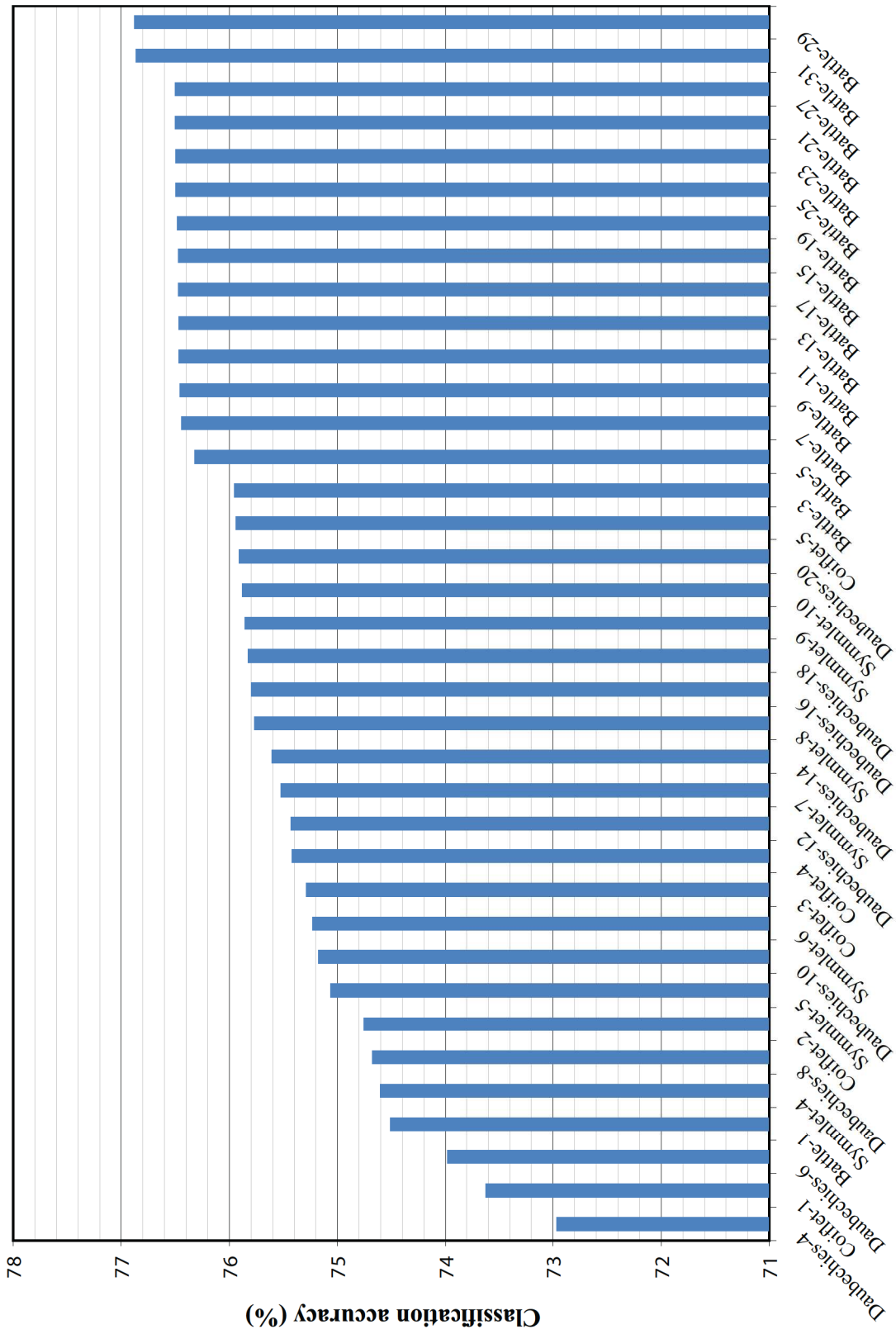


FIGURE 10.6: Performance of the WPC-SVM classifier on the cubic-chirp problem

10.3.3 Radar transmitter problem

The error rate (misclassified signals divided by total signals) for the WPT-SVM classifier on the radar transmitter classification problem is given in this section. Due to space considerations only a selected set of wavelets are plotted. The error rate for the Battle-Lemarié wavelet is given in figure 10.7. The error rate for the Daubechies wavelets, Symmlets and Coiflets are given in figure 10.8, figure 10.9 and figure 10.10.

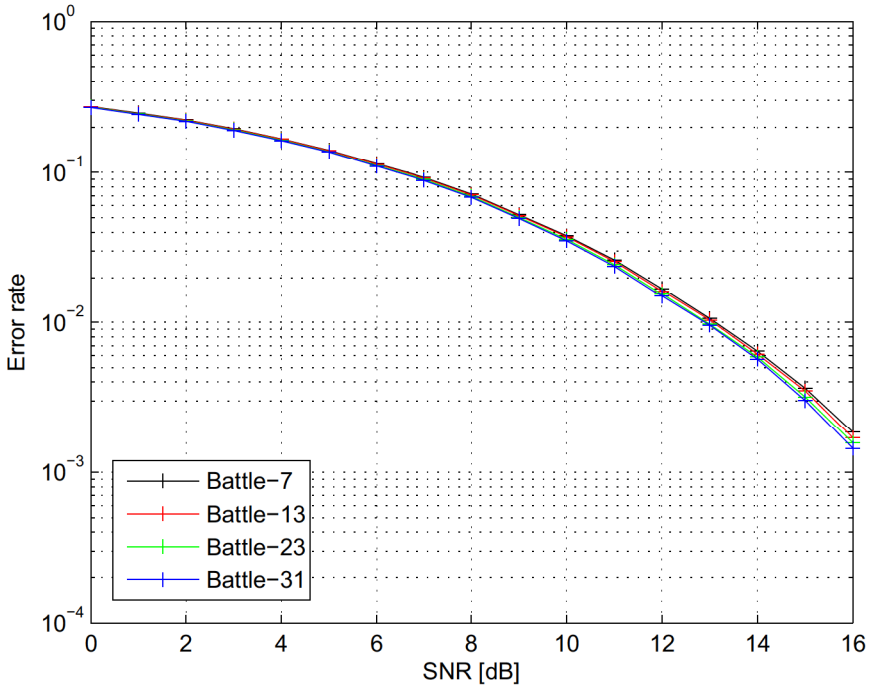


FIGURE 10.7: Error rate for the WPT-SVM classifier on the radar transmitter problem using Battle-Lemarié wavelets.

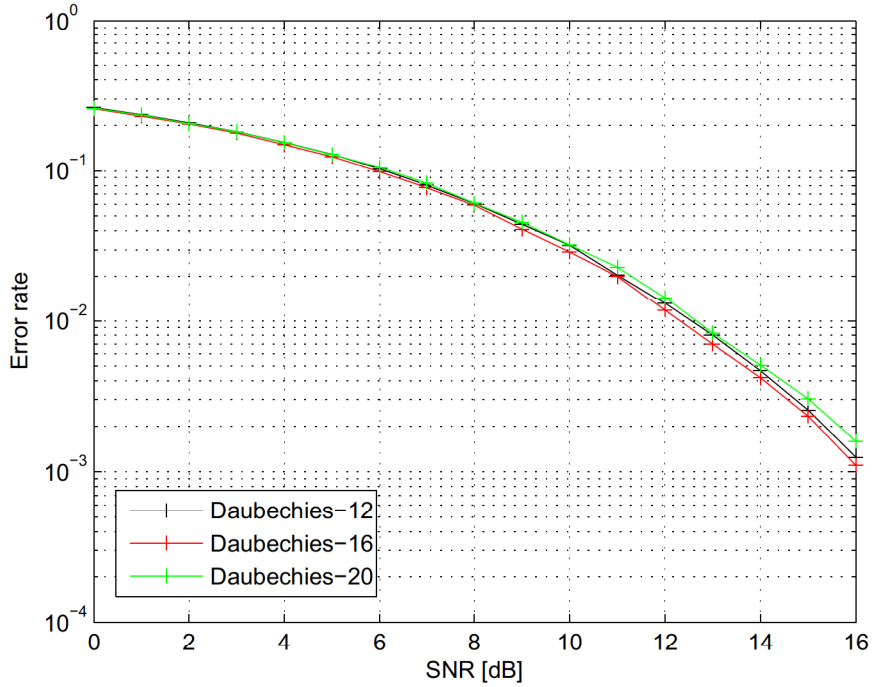


FIGURE 10.8: Error rate for the WPT-SVM classifier on the radar transmitter problem using Daubechies wavelets.

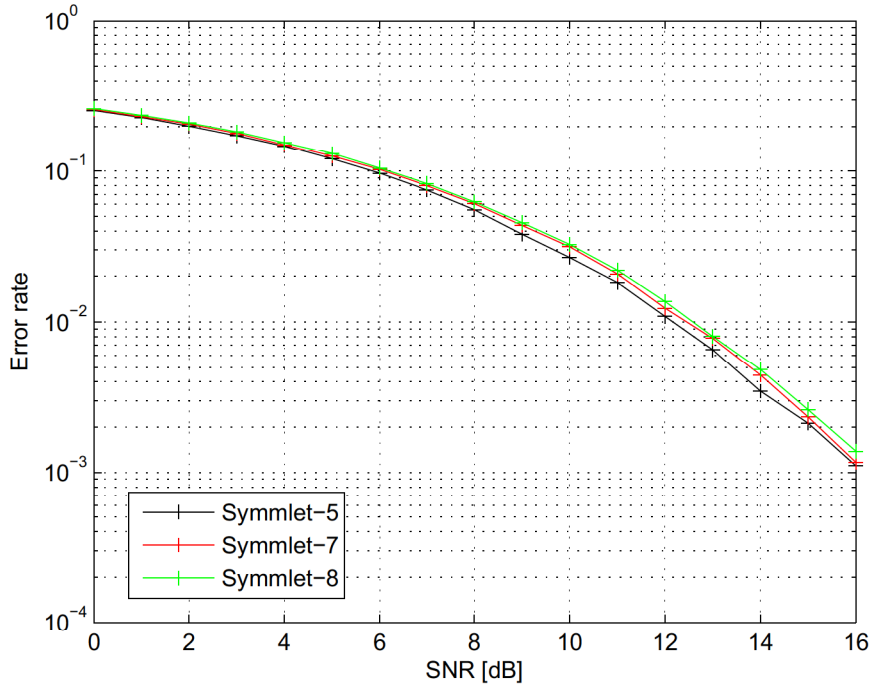


FIGURE 10.9: Error rate for the WPT-SVM classifier on the radar transmitter problem using Symmlet wavelets.

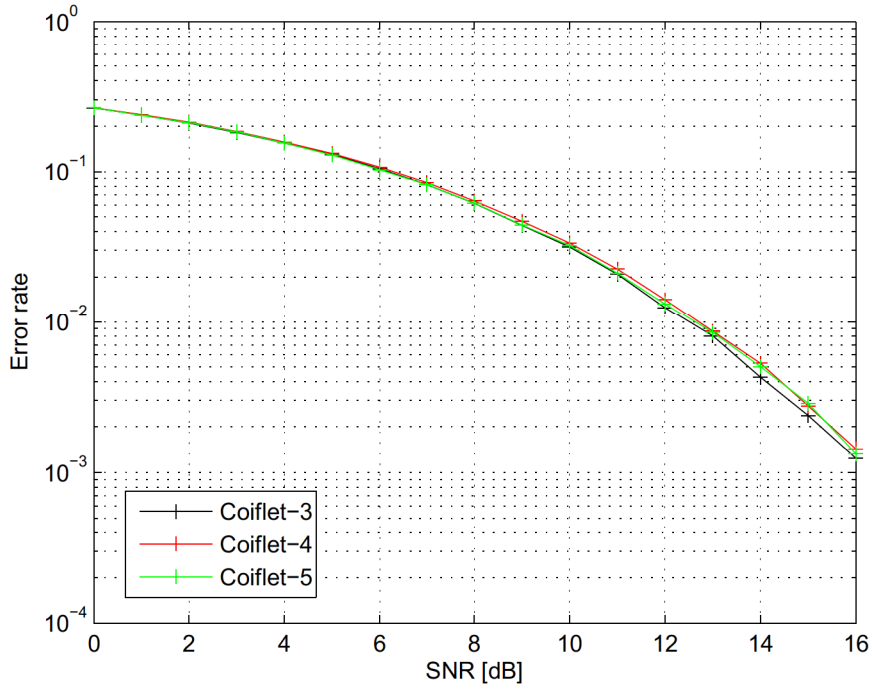


FIGURE 10.10: Error rate for the WPT-SVM classifier on the radar transmitter problem using Coiflet wavelets.

10.4 WPT-UNIVERSUM CLASSIFIER

The classification accuracy for the WPT-Universum classifier on the cubic-exponential chirp problem is given in figure 10.11. The classifier used a radially Gaussian kernel. The classification accuracy is plotted against the number of training samples. Because of the high running time the WPT-Universum classifier was not simulated on the radar transmitter problem.

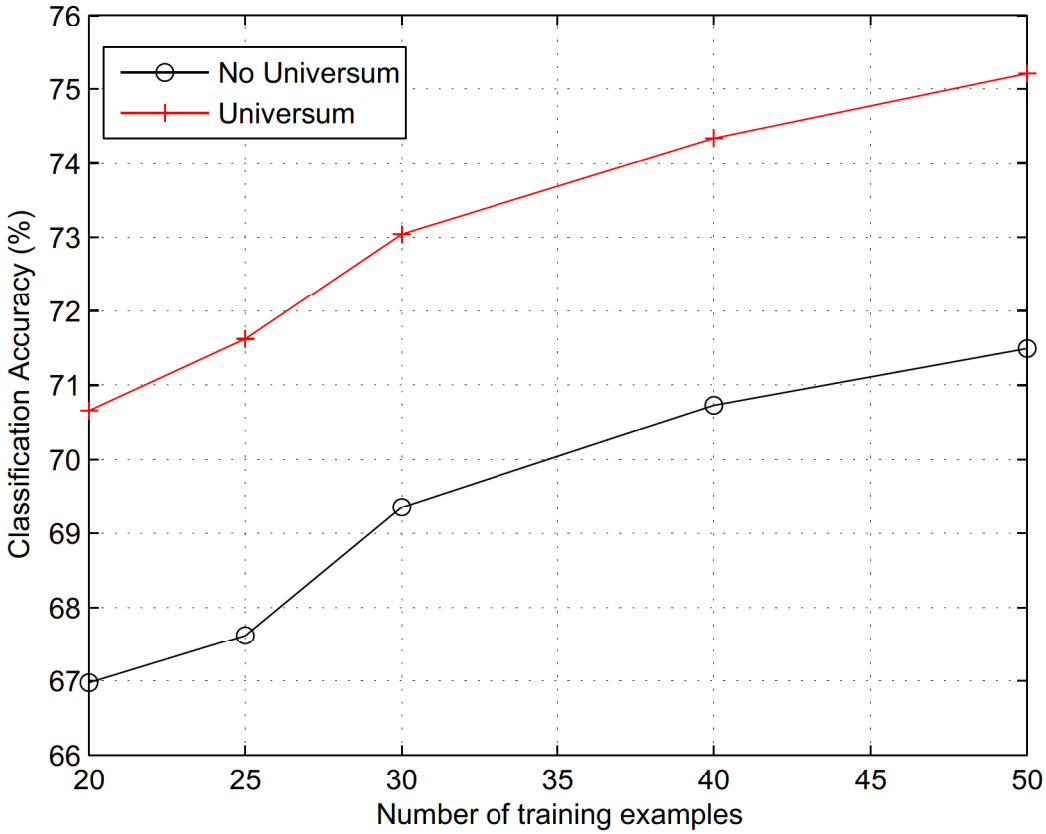


FIGURE 10.11: Classification results of the WPT-Universum classifier with a Gaussian kernel compared to the classification accuracy of the WPT-SVM classifier

10.5 DISCUSSION OF RESULTS

The results for the classifiers on the test problems are discussed in this section.

10.5.1 Cubic exponential chirp problem

The Cohen's class classifier achieved an accuracy of 79.16% on the cubic-exponential chirp problem. This is higher than the classification accuracy for the WPT-SVM for any wavelet (see figure 10.6). The highest classification accuracy for the WPT-SVM is 76.88% using the Battle-Lemarié-29 wavelet. The Battle-Lemarié wavelets performed significantly better on this problem than other wavelets and the performance increased with the order of the wavelet. It is illustrated in figure 10.11 that the WPT-Universum can attain a higher classification accuracy with fewer training samples than the WPT-SVM classifier. The complexity of the WPT-Universum classifier is however much larger since the Universum set is extremely large.

Because of this complexity the algorithm was not run on the radar transmitter problem. The EEG classification problem also did not have any acceptable datasets that could be used as a Universum.

10.5.2 EEG data

For the EEG data, only the WPT-SVM classifier was used because the Cohen's class classifier is infeasible for long signal lengths (discussed below in section 10.5.4). The WPT-SVM achieved a higher classification accuracy on the EEG data than the studies [3] and [4] (results in table 10.1 and 10.2). All wavelets outperformed the studies and the difference in classification accuracy between the different wavelets was small. It is unclear what caused the difference in performance between wavelets.

10.5.3 Radar transmitter data

For Battle-Lemarié wavelets (figure 10.7), the higher the order of the wavelet the higher classification accuracy is on the radar transmitter problem. The differences are however fairly small. It should be noted that the higher order wavelets have long filter lengths. In problems where the signal is longer, the effect of higher order would probably be more visible. For Daubechies wavelets (figure 10.8) the best performing wavelet was the Daubechies-16 wavelet (this was also the overall best performing wavelet on the problem set).

A comparison between the Cohen's class classifier and WPT-SVM for the best wavelet for each class is given in figure 10.12. The Cohen's class classifier outperforms the WPT-SVM significantly at high signal to noise ratios. It is interesting that the WPT-SVM performs a little better than the Cohen's class classifier at low signal-to-noise ratios. This may be because high noise amplifies the effect of the cross-terms.

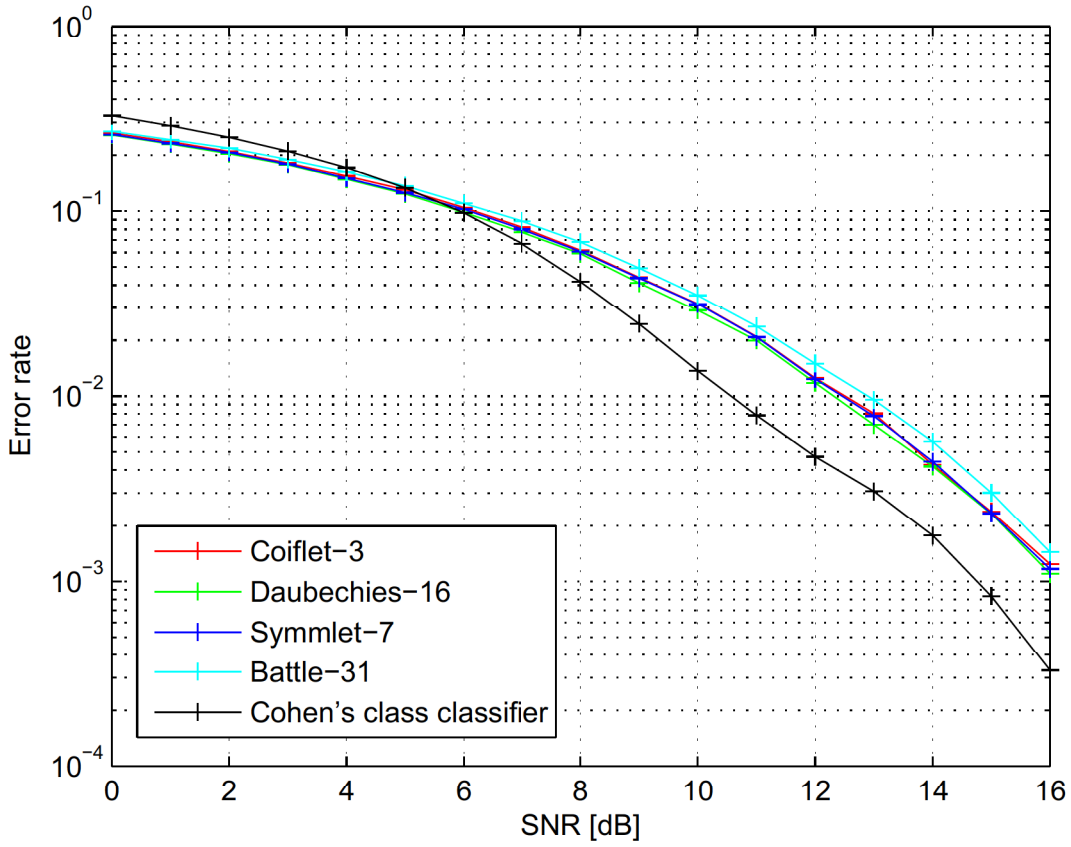


FIGURE 10.12: Comparison between the Cohen's class classifier and the WPT-SVM classifier

10.5.4 Complexity of algorithms

Both the computational complexity and memory requirements for the Cohen's class classifier is significantly higher than for the WPT-SVM. For a signal of length N, the Cohen's class classifer calculates a TFR of length N^2. For the WPT-SVM classifier, the size of the wavelet packet transform is approximately $(\log_2 N + 1)\, N$. This is illustrated in figure 10.13. The Cohen's class classifier also requires that the TFR be calculated multiple times during training. It is because of this complexity that the Cohen's class classifier was not tested on the EEG dataset.

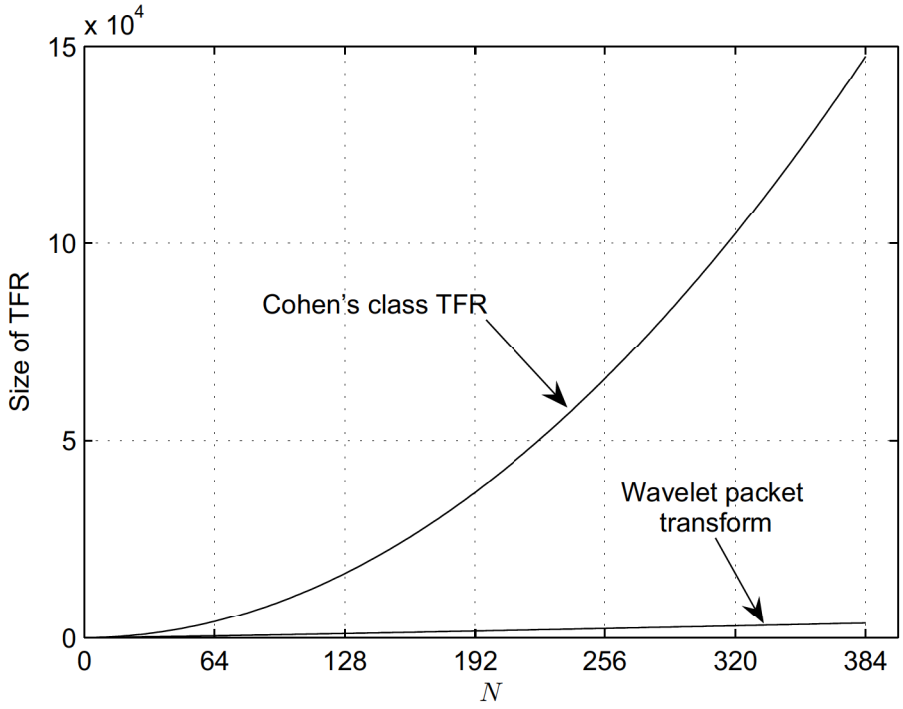


FIGURE 10.13: Size of the time-frequency representation for the Cohen's class classifier and the WPT-SVM classifier.

CHAPTER 11
CONCLUSION

11.1 SUMMARY

In this dissertation, radar transmitter identification was investigated. For this a radar transmitter model and several signal classifiers were developed. The radar transmitter model was developed as a pulse-forming network followed by an oscillator. A linear model for a magnetron oscillator was developed which incorporated frequency pushing. The linear model was used in the pulse forming network to develop an analytical solution based on the system of differential equations that describes a Guillemen E-type pulse-forming network.

In Chapter 7 a classifier was developed that improves on the series of classifiers developed in [11–13]. This classifier used a Cohen's class time-frequency representation with a radially Gaussian kernel. This kernel is described by a Bernstein expansion spread function introduced in Chapter 7. Constraints on the spread function were introduced to ensure that the spread function is continuous and differentiable. It was also proven that the constraints on the spread function can be added without increasing the complexity of the particle swarm optimisation algorithm. This classifier is compared to other classifiers in Chapter 10.

A new classifier using a wavelet packet transform was developed. In this classifier, the wavelet packet transform was calculated for each signal. The energy contribution for each of the points in the wavelet packet transform was then calculated. This was used with a support vector machine classifier. The classifier was compared to both the Cohen's class classifier with a cubic-exponential chirp (in Gaussian noise) test problem and on the radar transmitter

problem. On the cubic-exponential chirp problem, very high order Battle-Lemarié wavelets performed the best. The multiresolution filter coefficients for these wavelets are calculated in Section 6.5.1 and are listed in Appendix C. The classification accuracy of the wavelet packet classifier on this problem is slightly worse than the Cohen's class classifier. The complexity of this classifier was discussed in Chapter 10. This showed that the classifier has a much lower complexity than the Cohen's class classifier ($N \log_2 N$ compared to N^2).

The wavelet packet classifier was also tested on an EEG dataset (Section 10.3.1). The classification accuracy achieved with this classifier was much higher than the classification accuracy of two papers [3,4] that used the same dataset. The difference in performance between different wavelets was also much smaller than in the cubic-exponential chirp case.

On the radar transmitter problem, the wavelet packet classifier performed better than the Cohen's class classifier at low signal-to-noise ratios but worse on higher signal to noise ratios (Section 10.5.3). This is probably caused but the increased effect of cross-terms in the Cohen's class classifier at low signal-to-noise rations. The wavelet packet classifier also allowed for multiple signals per pattern to be incorporated.

A new classifier was developed in Chapter 9 that enables the prior information of a signal to be incorporated in the classification process with the aid of signals in the same domain. The disadvantage of this approach is that it has much higher complexity and little research has yet been done on Universum type classifiers. This concept however forms a good basis for future research.

11.2 FUTURE WORK

11.2.1 Universum classifiers

Classification using the Universum is a relatively new field. There are two main questions regarding Universum classifiers:

1. Selection of the Universum samples. Only a single study has been done regarding the selection of Universum samples [68]. The results however, remain inconclusive.

2. Weighting of the Universum samples (i.e. selection of the $C_{\mathcal{U}}$ factor).

The current formulation of the Universum also treats the class labels of the Universum samples as unknown. There are however be cases where the labels of the Universum patterns may be known which could be used to improve the classification. The Universum classifier could also be extended for multi-kernel classification.

11.2.2 Usage of a redundant wavelet packet transform

The discrete wavelet transform and wavelet packet transform only represents translations at $t = n2^j$ (i.e. the coarser the approximation the less translation values there are). There is an extension to the discrete wavelet transform that makes provision for $t = n$ translations at every level. This is known as the redundant discrete wavelet transform (RDWT) [71]. It may be possible to extend the wavelet packet transform also along these lines and utilise this for classification purposes.

11.2.3 Extended to image recognition problems

The wavelet packet based classifier may be extended to image recognition problems. The wavelet packet transform is extended to the image case with a wavelet packet quadtree. An example of this is given for the familiar boat test image (figure 11.1). The first level decomposition is in figure 11.2.3 and the second level decomposition is in figure 11.3.
As can be seen from these figures, the majority of the energy is in the low-pass representation. This is usually the motivation for wavelets in image compression – the higher frequency components tends to be sparse. This however does not mean that the higher frequency components are useless for classification. The higher frequency components clearly shows the edges. Many computer vision algorithms use edge detection as a feature extraction step (the Mexican hat wavelet is often used for edge detection [19]). Features such as textures also tend to have a certain frequency response that is concentrated in a specific frequency band.

As an example of where this may be used, consider optical character recognition. Two test digits are given in figure 11.4 along with the first and second wavelet packet decomposition levels. This will be used as the features. The total size of the calculated features would be $N^2 log_2 N$ where N is the image width and height (which is extended again to be a power of 2).

FIGURE 11.1: Boat test image.

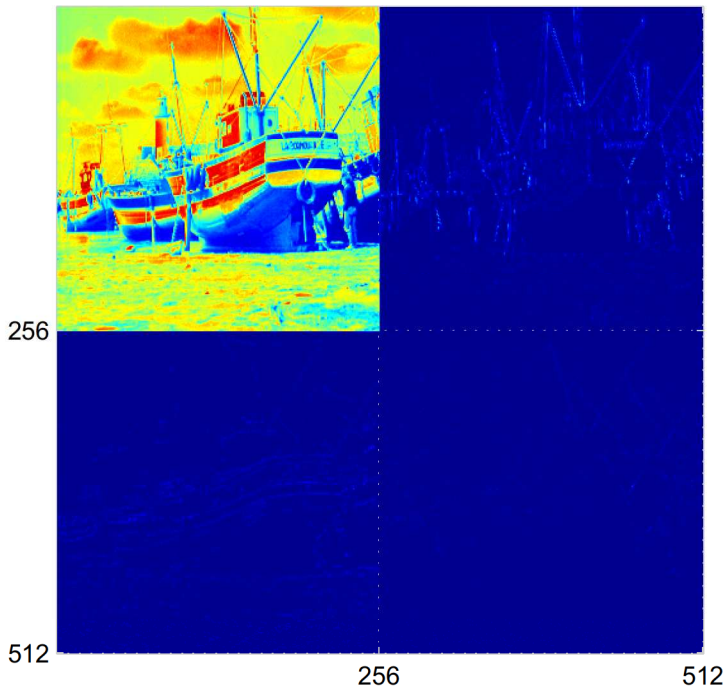


FIGURE 11.2: First level of a wavelet packet decomposition on the boat test image

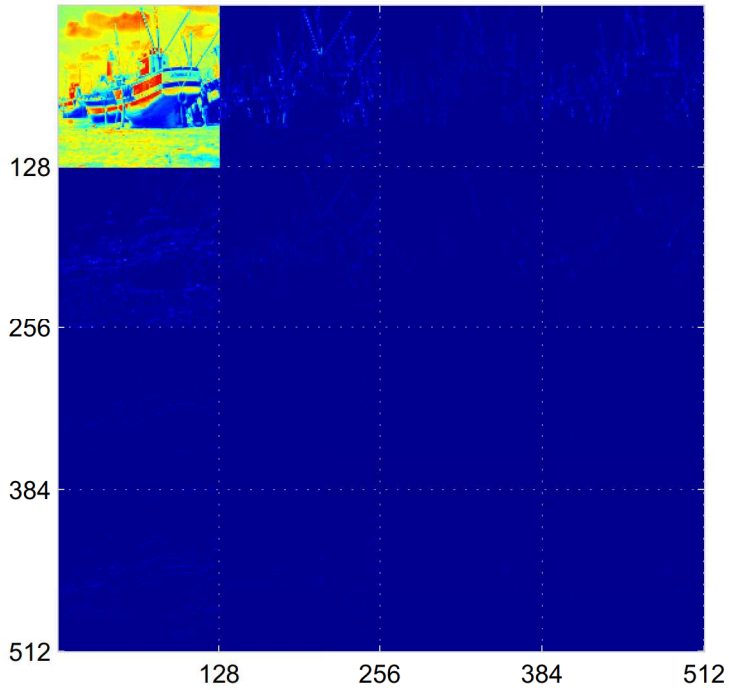


FIGURE 11.3: Second level of a wavelet packet decomposition on the boat test image

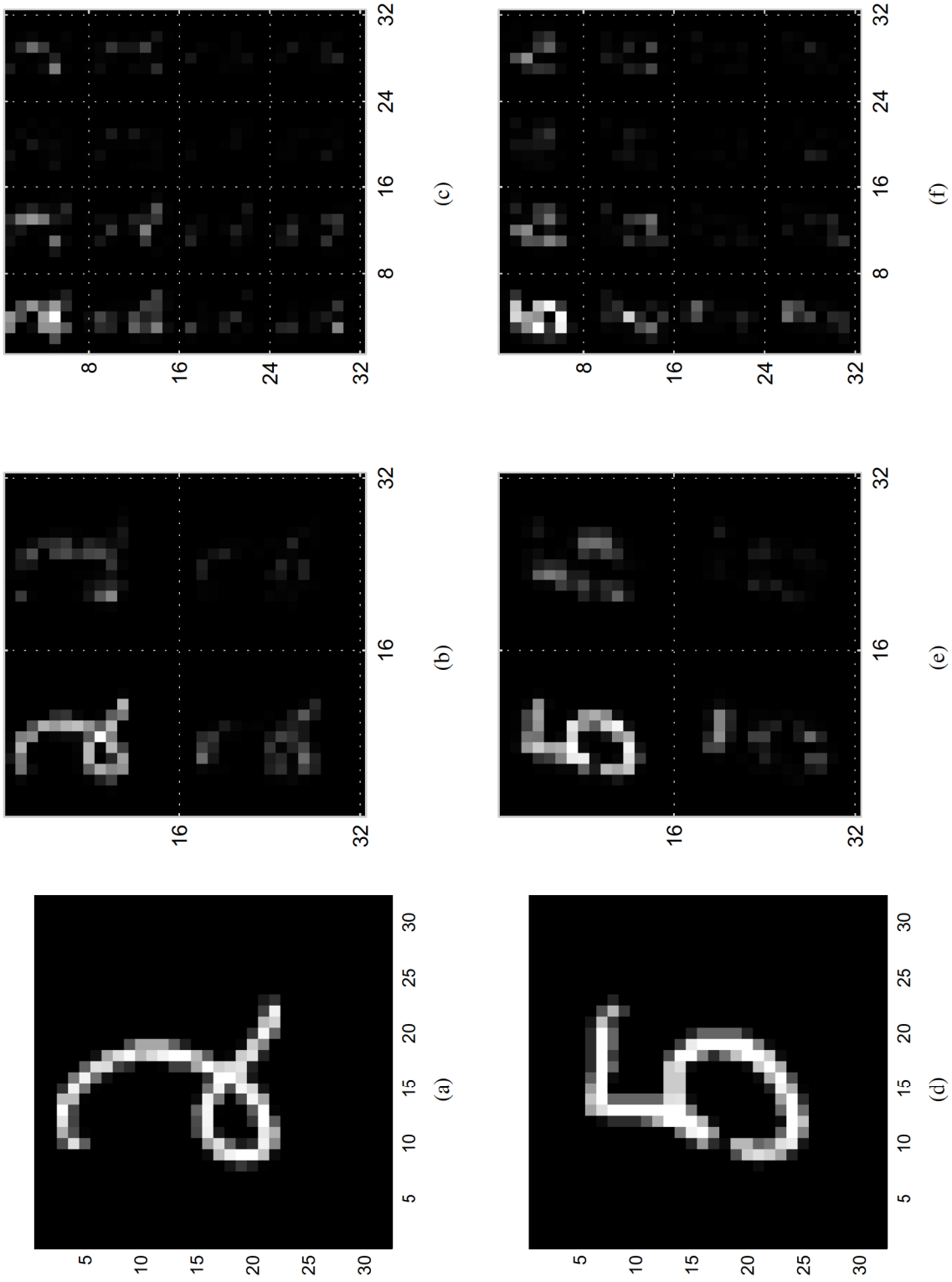


FIGURE 11.4: First and second level decomposition of MNIST digits.

REFERENCES

[1] J. Weston, R. Collobert, F. H. Sinz, L. Bottou, and V. Vapnik, "Inference with the Universum," in *ICML* (W. W. Cohen and A. Moore, eds.), vol. 148 of *ACM International Conference Proceeding Series*, pp. 1009–1016, ACM, 2006.

[2] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[3] L. Guo, D. Rivero, J. A. Seoane, and A. Pazos, "Classification of EEG signals using relative wavelet energy and artificial neural networks," in *GEC Summit* (L. Xu, E. D. Goodman, G. Chen, D. Whitley, and Y. Ding, eds.), pp. 177–184, ACM, 2009.

[4] K. Polat and S. Günes, "Classification of epileptiform EEG using a hybrid system based on decision tree classifier and fast fourier transform," *Applied Mathematics and Computation*, vol. 187, no. 2, pp. 1017–1026, 2007.

[5] B. Gillespie and L. Atlas, "Optimization of time and frequency resolution for radar transmitter identification," in *Acoustics, Speech, and Signal Processing, 1999. ICASSP '99. Proceedings., 1999 IEEE International Conference on*, vol. 3, pp. 1341–1344, Mar 1999.

[6] B. Gillespie and L. Atlas, "Data-driven optimization of time and frequency resolution for radar transmitter identification," in *Proceedings of the SPIE - The International Society for Optical Engineering*, 1998.

[7] B. Gillespie and L. Atlas, "Optimizing time-frequency kernels for classification," *Signal Processing, IEEE Transactions on*, vol. 49, pp. 485–496, Mar 2001.

[8] J. Mclaughlin, *Applications of Operator Theory to Time-Frequency Analysis and Classification*. PhD thesis, Univ. Washington, Seattle, 1997.

[9] R. G. Baraniuk and D. L. Jones, "Signal-dependent time-frequency analysis using a radially Gaussian kernel," *Signal Processing*, vol. 32, pp. 263–284, 1993.

[10] M. Davy and C. Doncarli, "Optimal kernels of time-frequency representations for signal classification," in *Time-Frequency and Time-Scale Analysis, 1998. Proceedings of the IEEE-SP International Symposium on*, pp. 581–584, Oct 1998.

[11] M. Davy, C. Doncarli, and G. Boudreaux-Bartels, "Improved optimization of time-frequency-based signal classifiers," *Signal Processing Letters, IEEE*, vol. 8, pp. 52–57, Feb 2001.

[12] A. Gretton, M. Davy, A. Doucet, and P. Rayner, "Nonstationary signal classification using support vector machines," in *Statistical Signal Processing, 2001. Proceedings of the 11th IEEE Signal Processing Workshop on*, pp. 305–308, 2001.

[13] M. Davy, A. Gretton, A. Doucet, and P. Rayner, "Optimized support vector machines for nonstationary signal classification," *Signal Processing Letters, IEEE*, vol. 9, pp. 442–445, Dec 2002.

[14] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York, USA: Wiley-Interscience Publication, 2nd ed., 2001.

[15] S. Pittner and S. Kamarthi, "Feature extraction from wavelet coefficients for pattern recognition tasks," in *Neural Networks,1997., International Conference on*, vol. 3, pp. 1484–1489, Jun 1997.

[16] A. Kandaswamy and C. Kumar, "Neural classification of lung sounds using wavelet coefficients," *Computers in biology and medicine*, vol. 34, pp. 1484–1489, 2004.

[17] P. Jahankhani, V. Kodogiannis, and K. Revett, "EEG signal classification using wavelet feature extraction and neural networks," in *JVA '06: Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing*, (Washington, DC, USA), pp. 120–124, IEEE Computer Society, 2006.

[18] R. Learned and A. S. Willsky, "A wavelet-packet approach to transient signal classification," *Applied Computational and Harmonic Analysis*, vol. 2, no. 3, pp. 265–278, 1995.

[19] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*. London: Academic Press, 3rd ed., 2008.

[20] I. Daubechies, *Ten Lectures on Wavelets*, vol. Volume 61 of *CBMS/NSF Series in Applied Math*. Society for Industrial and Applied Mathematics, 1st ed., 1992.

[21] G. Kaiser, *A Friendly Guide to Wavelets*. Cambridge, MA, USA: Birkhauser Boston, 1st ed., 1994.

[22] G. W. Ewell, *Radar Transmitters*. New York, USA: McGraw-Hill, 1981.

[23] G. Glasoe and J. Lebacqz, *Pulse Generators*. New York, USA: McGraw Hill, 1st ed., 1949.

[24] M. I. Skolnik, *Radar handbook*. New York, NY: McGraw-Hill, 1970.

[25] A. P. Engelbrecht, *Computational Intelligence - An Introduction*. Chichester, UK: John Wiley & Sons, 2002.

[26] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. Chichester, UK: John Wiley & Sons, 2006.

[27] F. van den Bergh, *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, 2001.

[28] S. H. Friedberg, A. J. Insel, and L. E. Spence, *Linear Algebra*. Upper Saddle River, NJ: Prentice Hall, 4th ed., 2002.

[29] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, UK: Cambridge University Press, 1 ed., March 2000.

[30] V. N. Vapnik, *Statistical learning theory*. New York, USA: John Wiley & Sons, 1st ed., 1998.

[31] D. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, 1999.

[32] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Belmont, MA, USA: Athena Scientific, 1 ed., 1996.

[33] J. Dattorro, *Convex Optimization and Euclidean Distance Geometry*. Palo Alto, CA, USA: Meboo Publishing, 2008.

[34] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[35] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall, 1999. 2nd edition.

[36] V. Vapnik and C. Cortes, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 2005.

[37] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, 2000.

[38] B. Schölkopf, *Support Vector Learning*. PhD thesis, Technishen Universität Berlin, 1997.

[39] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, ACM Press, 1992.

[40] M. Minksy and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[41] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning* (B. Schölkopf, C. Burges, and A. Smola, eds.), ch. 11, pp. 169–184, Cambridge, MA: MIT Press, 1999.

[42] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001.

[43] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 07 2006.

[44] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," *Proc. IEEE Workshop on Neural Networks and Signal Processing*, pp. 276–285, 24-26 Sep 1997.

[45] J. C. Platt, "Using analytic qp and sparseness to speed training of support vector machines," in *Neural Information Processing Systems*, vol. 11, (Cambridge, MA, USA), pp. 557–563, MIT Press, 1999.

[46] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in kernel methods: support vector learning*, pp. 185–208, Cambridge, MA, USA: MIT Press, 1999.

[47] R.-E. Fan, P.-H. Chen, and C.-J. Lin, "Working set selection using second order information for training SVM," *Journal of Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.

[48] C.-C. Chang and C.-J. Lin, "Training ν-support vector classifiers: Theory and algorithms," *Neural Computation*, vol. 13, no. 9, pp. 2119–2147, 2001.

[49] L. Cohen, "Time-frequency distributions – a review," *Proceedings of the IEEE*, vol. 77, no. 7, pp. 941–981, Jul 1989.

[50] F. Hlawatsch and G. Boudreaux-Bartels, "Linear and quadratic time-frequency signal representations," *Signal Processing Magazine, IEEE*, vol. 9, pp. 21–67, Apr 1992.

[51] F. Hlawatsch and F. Auger, eds., *Time-Frequency Analysis: Concepts and Methods*. ISTE – John Wiley & Sons, Inc., 1 ed., 2008.

[52] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *Information Theory, IEEE Transactions on*, vol. 36, pp. 961–1005, Sep 1990.

[53] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis*, vol. 11, pp. 674–693, July 1989.

[54] F. Auger, P. Flandrin, P. Gonçalvès, and O. Lemoine, *Time-Frequency Toolbox for use with MATLAB*. Centre National de la Recherche Scientifique (CNRS), Lyon, France, July 2008.

[55] S. K. Mitra, *Digital signal processing: a computer-based approach*. New York, NY, USA: McGraw-Hill, 1998.

[56] A. N. Akansu and R. A. Haddad, *Multiresolution signal decomposition - transforms, subbands and wavelets*. New York, NY, USA: Academic Press, 2nd ed., 2001.

[57] R. R. Coifman, Y. Meyer, and M. V. Wickerhauser, "Wavelet analysis and signal processing," in *Wavelets and their applications*, pp. 153–178, Boston, MA: Jones and Bartlett, 1992.

[58] E. W. Weisstein, *CRC concise encyclopaedia of mathematics*. Boca Raton, Fla, USA: CRC Press, 1999.

[59] G. G. Lorentz, *Bernstein polynomials*. New York, NY, USA: AMS Chelsea Publishing, 1986.

[60] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, pp. 151–160, 1986.

[61] L. Rajagpoal and S. Roy, "Design of maximally-flat FIR filters using the bernstein polynomial," *Circuits and Systems, IEEE Transactions on*, vol. 34, pp. 1587–1590, Dec 1987.

[62] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. New York: Springer-Verlag, 2001.

[63] C. M. Bishop, *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press, 1996.

[64] V. Vapnik, *Estimation of Dependences Based on Empirical Data*. New York: Springer, 2nd ed., 2006.

[65] B. Schölkopf, P. Simard, A. J. Smola, and V. Vapnik, "Prior knowledge in support vector kernels," in *NIPS* (M. I. Jordan, M. J. Kearns, and S. A. Solla, eds.), The MIT Press, 1997.

[66] G. R. G. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble, "Kernel-based data fusion and its application to protein function prediction in yeast," in *Pacific Symposium on Biocomputing* (R. B. Altman, A. K. Dunker, L. Hunter, T. A. Jung, and T. E. Klein, eds.), pp. 300–311, World Scientific, 2004.

[67] O. Chapelle, B. Schölkopf, and A. Zien, eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[68] S. Chen and C. Zhang, "Selecting informative universum sample for semi-supervised learning," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2009.

[69] F. H. Sinz, O. Chapelle, A. Agarwal, and B. Schölkopf, "An analysis of inference with the universum," in *21th Neural Information Processing Systems Conference* (J. C. Platt, D. Koller, Y. Singer, and S. Roweis, eds.), (Cambridge, MA, USA), pp. 1369–1376, MIT Press, 09 2008.

[70] R. Andrzejak, K. Lehnertz, C. Rieke, F. Mormann, P. David, and C. Elger, "Indications of nonlinear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state," *Phys. Rev. E*, vol. 64, 2001.

[71] J. Fowler, "The redundant discrete wavelet transform and additive noise," *Signal Processing Letters, IEEE*, vol. 12, pp. 629–632, Sept. 2005.

APPENDIX A
MATHEMATICAL DEFINITIONS

A.1 FOURIER TRANSFORMS

A.1.1 Continuous Fourier transform

The continuous Fourier transform of $x(t)$ in terms of the angular frequency ω is defined as

$$\hat{x}(\omega) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt. \tag{A.1}$$

The inverse continuous Fourier transform is defined as

$$x(t) = \mathcal{F}^{-1}\{\hat{x}(\omega)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{x}(\omega)e^{j\omega t}d\omega. \tag{A.2}$$

The Fourier transform of a *scale change* $x(at)$ is

$$x(at) \Leftrightarrow \frac{1}{a}\hat{x}(\omega/a) \;\; a > 0. \tag{A.3}$$

This can be proven by

$$\mathcal{F}\{x(at)\} = \int_{-\infty}^{\infty} x(at)e^{-j\omega t}dt. \tag{A.4}$$

Making the substitution $v = at$ gives

$$\mathcal{F}\{x(at)\} = \frac{1}{a}\int_{-\infty}^{\infty} x(v)e^{-j\omega\frac{1}{a}v}dv \tag{A.5}$$

$$= \frac{1}{a}\hat{x}\left(\frac{\omega}{a}\right). \tag{A.6}$$

The Fourier transform of the *time-shift* is

$$x(t-t_0) \Leftrightarrow \hat{x}(\omega)e^{-j\omega t_0}. \tag{A.7}$$

This can be proven by

$$x(t-t_0) = \int_{-\infty}^{\infty} x(t-t_0)e^{-j\omega t}dt, \tag{A.8}$$

making the substitution $v = t - t_0$ gives

$$\begin{aligned}
\mathcal{F}\{x(t-t_0\} &= \int_{-\infty}^{\infty} x(v)e^{-j\omega(v+t_0)}dv && \text{(A.9)}\\
&= e^{-j\omega t_0}\int_{-\infty}^{\infty} x(v)e^{-j\omega v}dv && \text{(A.10)}\\
&= e^{-j\omega t_0}\hat{x}(\omega). && \text{(A.11)}
\end{aligned}$$

A.1.1.1 Fourier transform of selected functions

A rectangular function can be defined as

$$\Pi(t) = \begin{cases} 1 & -\frac{1}{2} \leq t \leq \frac{1}{2} \\ 0 & \textit{otherwise.} \end{cases} \tag{A.12}$$

The Fourier transform of this is

$$\begin{aligned}
\hat{\Pi}(\omega) &= \int_{-\infty}^{\infty} \Pi(t)e^{-j\omega t}dt && \text{(A.13)}\\
&= \int_{-\frac{1}{2}}^{\frac{1}{2}} e^{-j\omega t}dt && \text{(A.14)}\\
&= \frac{1}{-j\omega}e^{-j\omega t}\Big|_{\frac{1}{2}}^{\frac{1}{2}} && \text{(A.15)}\\
&= \frac{j\,e^{-\frac{j\,w}{2}}}{w} - \frac{j\,e^{\frac{j\,w}{2}}}{w} && \text{(A.16)}\\
&= \frac{2\,\sin\left(\frac{\omega}{2}\right)}{\omega}. && \text{(A.17)}
\end{aligned}$$

A.1.2 Discrete-Time Fourier Transform (DTFT)

The discrete-time Fourier transform is defined as [55]

$$\hat{x}(\omega) = \mathcal{F}_{DTFT}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}. \tag{A.18}$$

The inverse of the discrete time Fourier transform is defined as

$$x[n] = \mathcal{F}_{DTFT}^{-1}\{\hat{x}(\omega)\} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{x}(\omega)e^{j\omega n}d\omega. \tag{A.19}$$

APPENDIX B
MAXIMA SOURCE CODE

B.1 SOURCE CODE FOR THE SCALING MR FILTER CALCULATION

```
j : sqrt(-1);
pi: %pi;


/* This function calculates the z'nth value*/
zp(p,w) := trigsimp((-2)^(p-2)*(1/factorial(p-1))*diff(1/(sin(w/2))^2,w,p-2));


/* calculates phihat for a specific degree of n */
phin(n,w) := 1/((w/2)^(n+1)*sqrt(zp(2*(n+1),w)));
phins(n,w) := subst(w,tmp,expand(phin(n,tmp)));

/* calculates hat h for a specific n */
hn(n,w) := sqrt(2)*phins(n,2*w)/phins(n,w);
hnconj(n,w) := subst(w,tmp,expand(conjugate(hn(n,tmp))));

/* calculates g hat for a specific n */
gn(n,w) := exp(-j*w)*hnconj(n,w+pi);

/* calculate psin */
psin(n,w) := (1/sqrt(2))*gn(n,w/2)*phins(n,w/2);


/* calculate hm - inverse fourier transform */
hm(n,m) := integrate(hn(n,w)*1/(2*%pi)*exp(j*w*m),w,-%pi,%pi);
```

B.2 SOURCE CODE FOR THE CALCULATION OF THE MEXICAN HAT FOURIER TRANSFORM

```
j : sqrt(-1);

phi(t) := 2/sqrt(3)*1/sqrt(pi)*(1-t^2)*exp(-t^2/2);

phih(w,t) := integrate(phi(t)*exp(-j*t*w),t,-inf,inf);
```

APPENDIX C

BATTLE-LEMARIÉ FILTER COEFFICIENTS

Filter name	$n = \pm 0, \pm 1, \pm 2, \pm 3, \pm 4, \pm 5, \pm 6, \pm 7, \pm 8, \pm 9, \pm 10, \ldots$
Battle-1	0.817646, 0.397297, -0.069101, -0.051945, 0.016971, 0.009991, -0.003883, -0.002202, 0.000923, 0.000512
Battle-3	0.766130, 0.433923, -0.050202, -0.110037, 0.032081, 0.042068, -0.017176, -0.017982, 0.008685, 0.008201, -0.004354, -0.003882, 0.002187, 0.001882, -0.001104, -0.000927, 0.000560, 0.000462, -0.000285, -0.000232, 0.000146, 0.000118
Battle-5	0.747234, 0.442463, -0.037020, -0.129269, 0.029474, 0.061312, -0.021006, -0.032520, 0.014010, 0.018209, -0.009049, -0.010562, 0.005768, 0.006279, -0.003660, -0.003799, 0.002321, 0.002329, -0.001474, -0.001441, 0.000937, 0.000899, -0.000597, -0.000564, 0.000382, 0.000356, -0.000244, -0.000225, 0.000157, 0.000143
Battle-7	0.737430, 0.445715, -0.028912, -0.137555, 0.025205, 0.071541, -0.020406, -0.042204, 0.015656, 0.026279, -0.011602, -0.016897, 0.008425, 0.011113, -0.006050, -0.007433, 0.004319, 0.005037, -0.003076, -0.003449, 0.002188, 0.002380, -0.001557, -0.001653, 0.001109, 0.001155, -0.000791, -0.000810, 0.000564, 0.000570, -0.000403, -0.000403, 0.000288, 0.000285, -0.000206, -0.000202, 0.000148, 0.000144
Battle-9	0.731453, 0.447278, -0.023596, -0.141777, 0.021540, 0.077320, -0.018661, -0.048397, 0.015505, 0.032107, -0.012489, -0.021994, 0.009847, 0.015390, -0.007655, -0.010941, 0.005898, 0.007874, -0.004520, -0.005723, 0.003453, 0.004194, -0.002633, -0.003094, 0.002007, 0.002295, -0.001529, -0.001710, 0.001165, 0.001279, -0.000889, -0.000960, 0.000678, 0.000723, -0.000518, -0.000545, 0.000396, 0.000412, -0.000302, -0.000312, 0.000231
Battle-11	0.727435, 0.448144, -0.019885, -0.144192, 0.018638, 0.080824, -0.016808, -0.052443, 0.014669, 0.036230, -0.012470, -0.025886, 0.010392, 0.018892, -0.008534, -0.013998, 0.006936, 0.010494, -0.005598, -0.007941, 0.004496, 0.006057, -0.003599, -0.004649, 0.002876, 0.003589, -0.002295, -0.002783, 0.001831, 0.002167, -0.001460, -0.001693, 0.001164, 0.001326, -0.000929, -0.001042, 0.000741, 0.000821, -0.000592, -0.000647, 0.000473
Battle-13	0.724552, 0.448671, -0.017162, -0.145694, 0.016354, 0.083084, -0.015130, -0.055180, 0.013637, 0.039173, -0.012025, -0.028820, 0.010417, 0.021674, -0.008899, -0.016549, 0.007522, 0.012780, -0.006308, -0.009958, 0.005259, 0.007817, -0.004367, -0.006175, 0.003615, 0.004904, -0.002987, -0.003913, 0.002465, 0.003134, -0.002032, -0.002519, 0.001674, 0.002031, -0.001379, -0.001642, 0.001136, 0.001330, -0.000936, -0.001080, 0.000771
Battle-15	0.722384, 0.449017, -0.015085, -0.146689, 0.014533, 0.084616, -0.013679, -0.057097, 0.012607, 0.041313, -0.011407, -0.031041, 0.010162, 0.023866, -0.008939, -0.018637, 0.007782, 0.014719, -0.006721, -0.011728, 0.005768, 0.009412, -0.004927, -0.007598, 0.004193, 0.006166, -0.003559, -0.005026, 0.003015, 0.004113, -0.002551, -0.003377, 0.002156, 0.002781, -0.001821, -0.002297, 0.001537, 0.001901, -0.001298, -0.001577, 0.001095
Battle-17	0.720694, 0.449254, -0.013452, -0.147380, 0.013058, 0.085699, -0.012441, -0.058482, 0.011649, 0.042903, -0.010741, -0.032740, 0.009770, 0.025596, -0.008785, -0.020335, 0.007825, 0.016344, -0.006914, -0.013252, 0.006072, 0.010821, -0.005305, -0.008887, 0.004616, 0.007336, -0.004004, -0.006081, 0.003465, 0.005060, -0.002994, -0.004225, 0.002582, 0.003538, -0.002225, -0.002970, 0.001916, 0.002499, -0.001649, -0.002108, 0.001419

Filter name	$n = \pm 0, \pm 1, \pm 2, \pm 3, \pm 4, \pm 5, \pm 6, \pm 7, \pm 8, \pm 9, \pm 10, \ldots$
Battle-19	0.719340, 0.449425, -0.012134, -0.147879, 0.011844, 0.086490, -0.011384, -0.059512, 0.010786, 0.044108, -0.010086, -0.034059, 0.009320, 0.026970, -0.008525, -0.021718, 0.007730, 0.017698, -0.006956, -0.014552, 0.006221, 0.012049, -0.005536, -0.010035, 0.004905, 0.008398, -0.004332, -0.007057, 0.003815, 0.005952, -0.003353, -0.005037, 0.002941, 0.004274, -0.002577, -0.003637, 0.002256, 0.003101, -0.001973, -0.002650, 0.001725
Battle-21	0.718231, 0.449552, -0.011049, -0.148250, 0.010830, 0.087084, -0.010479, -0.060295, 0.010017, 0.045039, -0.009468, -0.035096, 0.008858, 0.028072, -0.008211, -0.022848, 0.007551, 0.018827, -0.006896, -0.015657, 0.006260, 0.013113, -0.005654, -0.011046, 0.005086, 0.009350, -0.004559, -0.007946, 0.004074, 0.006778, -0.003632, -0.005799, 0.003232, 0.004976, -0.002872, -0.004280, 0.002549, 0.003690, -0.002259, -0.003188, 0.002001
Battle-23	0.717306, 0.449648, -0.010142, -0.148535, 0.009972, 0.087542, -0.009698, -0.060904, 0.009335, 0.045772, -0.008897, -0.035923, 0.008405, 0.028964, -0.007875, -0.023778, 0.007325, 0.019771, -0.006770, -0.016596, 0.006222, 0.014032, -0.005690, -0.011933, 0.005182, 0.010197, -0.004703, -0.008749, 0.004256, 0.007533, -0.003841, -0.006506, 0.003460, 0.005634, -0.003111, -0.004891, 0.002793, 0.004256, -0.002505, -0.003710, 0.002244
Battle-25	0.716523, 0.449723, -0.009371, -0.148757, 0.009237, 0.087902, -0.009019, -0.061386, 0.008729, 0.046358, -0.008376, -0.036591, 0.007973, 0.029693, -0.007536, -0.024548, 0.007075, 0.020564, -0.006603, -0.017395, 0.006131, 0.014824, -0.005666, -0.012708, 0.005215, 0.010947, -0.004783, -0.009469, 0.004374, 0.008218, -0.003990, -0.007154, 0.003631, 0.006245, -0.003298, -0.005465, 0.002992, 0.004792, -0.002710, -0.004210, 0.002452
Battle-27	0.715852, 0.449783, -0.008708, -0.148934, 0.008601, 0.088189, -0.008425, -0.061773, 0.008189, 0.046832, -0.007901, -0.037137, 0.007569, 0.030296, -0.007204, -0.025192, 0.006816, 0.021233, -0.006414, -0.018077, 0.006006, 0.015509, -0.005599, -0.013386, 0.005200, 0.011611, -0.004813, -0.010112, 0.004441, 0.008837, -0.004088, -0.007746, 0.003754, 0.006808, -0.003441, -0.005998, 0.003148, 0.005295, -0.002877, -0.004684, 0.002625
Battle-29	0.715270, 0.449831, -0.008133, -0.149077, 0.008045, 0.088422, -0.007902, -0.062089, 0.007708, 0.047221, -0.007469, -0.037589, 0.007193, 0.030798, -0.006886, -0.025732, 0.006557, 0.021802, -0.006212, -0.018663, 0.005859, 0.016102, -0.005503, -0.013979, 0.005150, 0.012197, -0.004804, -0.010686, 0.004468, 0.009395, -0.004145, -0.008285, 0.003836, 0.007325, -0.003544, -0.006491, 0.003268, 0.005765, -0.003009, -0.005130, 0.002767
Battle-31	0.714760, 0.449871, -0.007629, -0.149194, 0.007556, 0.088614, -0.007437, -0.062350, 0.007276, 0.047544, -0.007077, -0.037966, 0.006844, 0.031220, -0.006585, -0.026191, 0.006304, 0.022287, -0.006007, -0.019167, 0.005700, 0.016617, -0.005389, -0.014498, 0.005076, 0.012715, -0.004767, -0.011197, 0.004463, 0.009896, -0.004169, -0.008772, 0.003885, 0.007797, -0.003614, -0.006946, 0.003355, 0.006201, -0.003111, -0.005546, 0.002880

APPENDIX D
DAUBECHIES AND SYMMLET WAVELETS

D.1 DAUBECHIES WAVELETS

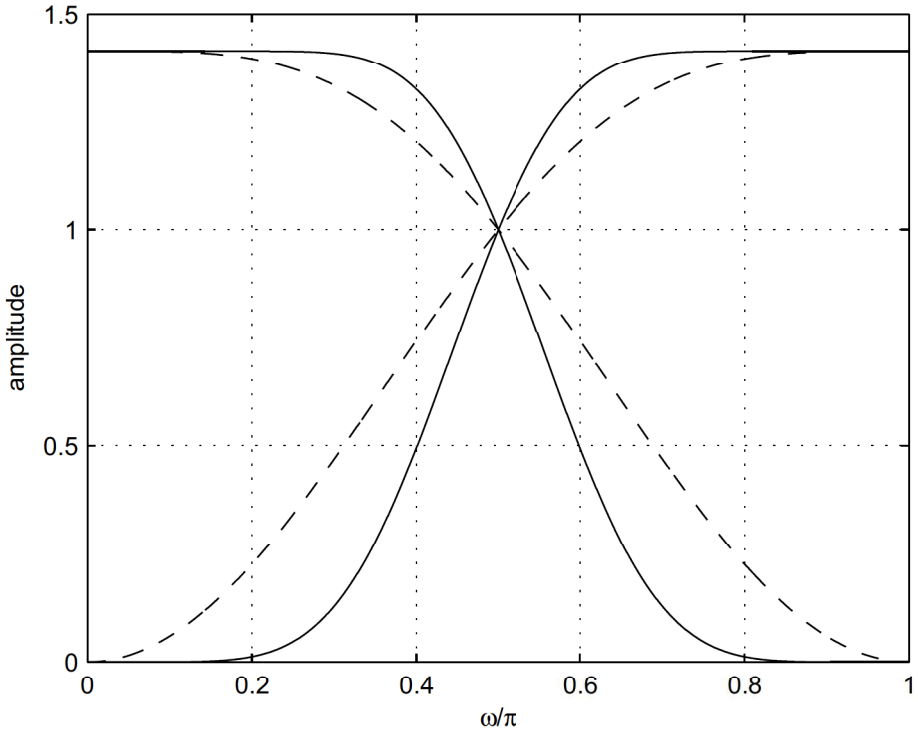


FIGURE D.1: Frequency response of the Daubechies 2 and Daubechies 7 conjugate filters.

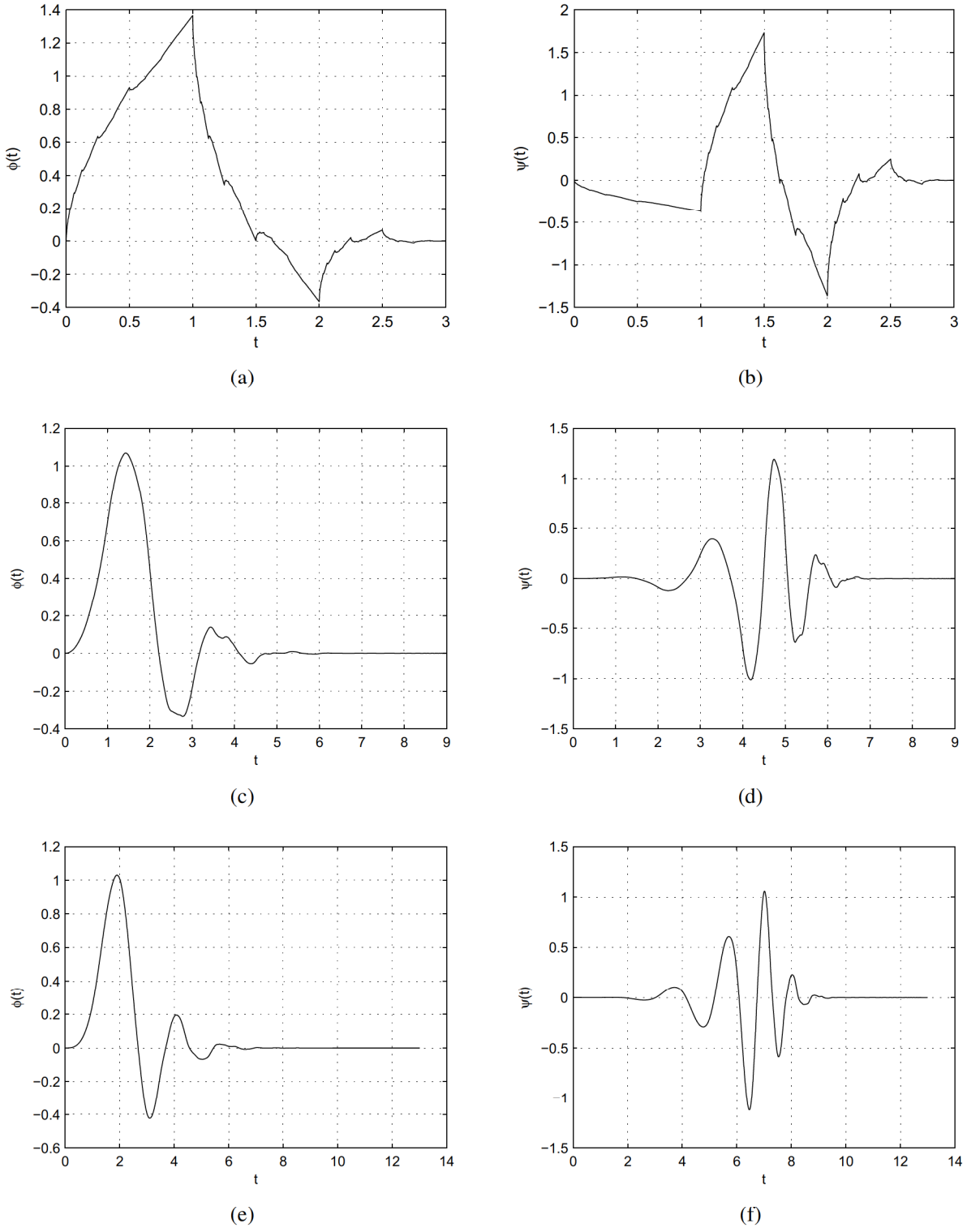


FIGURE D.2: Daubechies wavelets. (a), (b) Scaling and wavelet for Daubechies 2. (c), (d) Scaling and wavelet for Daubechies 5. (e), (f) Scaling and wavelet for the Daubechies 7.

D.2 SYMMLETS

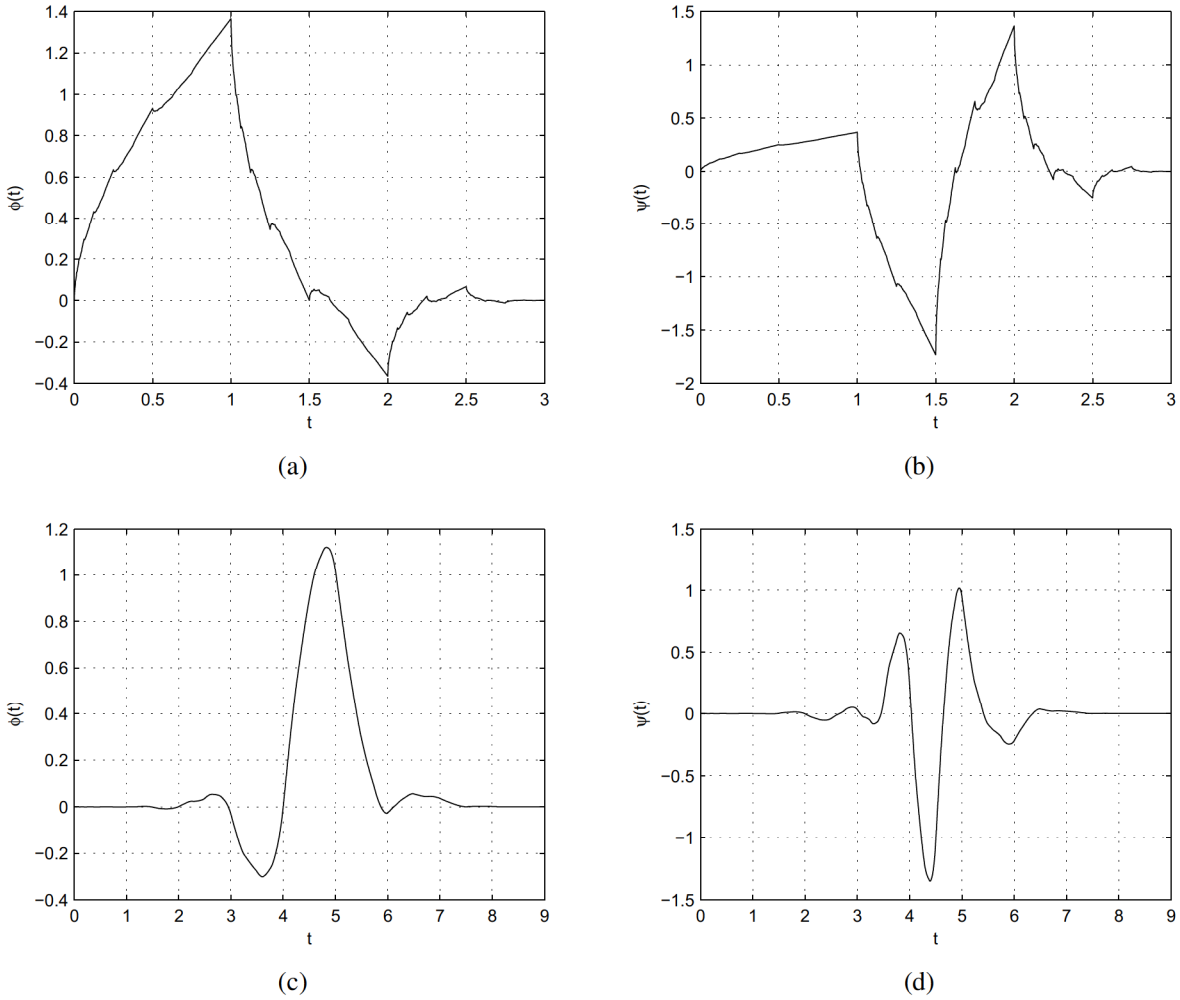


FIGURE D.3: Symmlet wavelet. (a),(b) The scaling function and wavelet for Symmlet 2. (c), (d) The scaling function and wavelet for Symmlet 5.