

**School of Information Technology and Electrical  
Engineering**

**ELEC 4801 THESIS PROJECT**

*Thesis: Speech Compression Using Wavelets*

**Name: Nikhil Rao**

**Student No: 33710745**

**Supervisor: Dr John Homer**

*Submitted for the degree of  
Bachelor of Engineering (Elec Hons)  
October 2001*

Nikhil Rao,  
5 Mandell Close,  
Coopers Plains,  
Qld 4108

October 18, 2001

Prof. Simon Kaplan,  
Head of School of Information Technology  
and Electrical Engineering,  
The University of Queensland,  
St Lucia, Qld 4067

Dear Professor Simon Kaplan,

In accordance with the requirements of the degree of Bachelor of Engineering (Honours) in the division of Electrical Engineering, I present the following thesis entitled "*Speech Compression Using Wavelets*". This work was performed under the supervision and guidance of Dr. John Homer.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text, or references and has not been previously submitted for a degree at the University of Queensland or any other institution.

Sincerely,

Nikhil Rao

# Acknowledgements

I would like to express my sincere thanks to my thesis supervisor, Dr John Homer. Dr Homer provided me with necessary support, advice, facilities and enthusiasm required to successfully complete this thesis.

His efforts in helping me in the development of the project, through technical difficulties and in search for relevant literature are much appreciated.

I would also like to thank Shane Goodwin, the supervisor of the Signal and Image Processing Labs for installing the Wavelet Toolbox software to assist in my learning.

# Abstract

Speech compression is the technology of converting human speech into an efficiently encoded representation that can later be decoded to produce a close approximation of the original signal. This thesis presents a new algorithm to compress speech signals using Discrete Wavelet Transform (DWT) Techniques.

Wavelet analysis is the breaking up of a signal into a set of scaled and translated versions of an original (*or mother*) wavelet. Taking the wavelet transform of a signal decomposes the original signal into wavelets coefficients at different scales and positions. These coefficients represent the signal in the wavelet domain and all data operations can be performed using just the corresponding wavelet coefficients.

In this thesis a Wavelet based speech coder is implemented in software using Matlab 6's Wavelet Toolbox. The major issues concerning the design of this Wavelet based speech coder are choosing optimal wavelets for speech signals, decomposition level in the DWT, thresholding criteria for coefficient truncation and efficient encoding of truncated coefficients.

The performance of the wavelet compression scheme on both male and female spoken sentences is compared. On a male spoken sentence the scheme reaches a signal-to-noise ratio of 17.45 db and a compression ratio of 3.88, using a level dependent thresholding approach. A significant advantage of using wavelets for speech coding is that the compression ratio can easily be varied, while most other techniques have fixed compression ratios.

Finally some enhancements to the wavelet compression technique are suggested to improve reconstructed speech signal quality and compression ratios.

# Contents

<b>1.</b>	<b>Introduction</b>	<b>8</b>
<b>2.</b>	<b>Theory</b>	<b>10</b>
2.1	Introducing Wavelets	10
2.2	Wavelets vs. Fourier Transforms	10
2.2.1	General Concepts	10
2.2.2	Time-Frequency Resolution	12
2.3	Examples of Wavelets	14
2.4	The Discrete Wavelet Transform	15
2.4.1	Vanishing Moments	16
2.5	The Fast Wavelet Transform Algorithm	17
2.5.1	Implementation Using Filters	17
2.5.2	Multilevel Decomposition	19
2.6	Signal Reconstruction	21
<b>3.</b>	<b>Literature Review</b>	<b>22</b>
3.1	Wavelet Speech Compression Techniques	22
3.1.1	Introduction	22
3.1.2	Choice of Wavelet	22
3.1.3	Wavelet Decomposition	23
3.1.4	Truncation of Coefficients	24
3.1.5	Encoding Coefficients	24
3.1.6	Detecting Voiced vs. Unvoiced Speech Frames	25
3.1.7	Performance Relative to Other Coding Schemes	26
3.2	Real Time Audio Compression Examples	26
<b>4.</b>	<b>System Implementation</b>	
4.1	Design Overview	29
4.2	Software Implementation	30
4.2.1	Wavelet Functions and Transforms	30
4.2.2	Speech Signals	30

4.2.3	Calculating Thresholds	31
4.2.4	Encoding zero-valued Coefficients	32
<b>5.</b>	<b>Results</b>	<b>33</b>
5.1	Optimal Decomposition Level in Wavelet Transforms	33
5.2	Retained Energy in First N/2 Coefficients	34
5.3	Voiced, Unvoiced and Mixed Frames	35
5.4	Performance Measures	37
5.5	Performance of Recorded Speech Coding	39
5.6	Performance of Real Time Speech Coding	40
5.7	Varying Truncation % vs. Performance	41
<b>6.</b>	<b>Discussion</b>	<b>43</b>
6.1	Decomposition Level in Wavelet Transforms	43
6.2	Optimal Wavelet Selection	43
6.3	Compression Score & Size of Frames	44
6.4	Compression Score & Decomposition Level	45
<b>7.</b>	<b>Future Work</b>	<b>46</b>
7.1	Enhancing Quality	46
7.2	Improving Compression Ratios	46
<b>8.</b>	<b>Conclusion</b>	<b>48</b>
	<b>References</b>	<b>49</b>
	<b>Appendix – Code Listings</b>	<b>52</b>

# List of Figures

2.1	WFT Resolution	13
2.2	Wavelets Resolution	13
2.3	Different Wavelet Families	14
2.4	Filtering Operation of DWT	18
2.5	Decomposition of DWT Coefficients	19
2.6	Level 3 Decomposition of Sample Signal S	20
2.7	Wavelets Reconstruction	21
3.1	Tree-structured DWT	27
3.2	(a) A Clarinet Signal, (b) Tonal Signal, (c) Residual Signal	28
4.1	Design Flow of Wavelet Based Speech Coder	29
5.1	Original Speech Signal and Reconstructed Approximations at Different Scales	33
5.2	A Voiced Speech Segment and its DWT at Scale 5	35
5.3	An Unvoiced Speech Segment and its DWT at Scale 5	36
5.4	A Mixed Speech Segment and its DWT at Scale 5	36
5.5	Truncation % vs. Threshold	41
5.6	SNR vs. % Truncation	41
5.7	PSNR vs. % Truncation	42
5.8	NRMSE vs. % Truncation	42

# List of Tables

4.1	Values for M	31
5.1	Percentage of Energy Concentrated in $N/2$ Coefficients	34
5.2	Performance of Recorded Speech Compression	39
5.3	Performance of Recorded Speech Compression	40
6.1	Frame Size vs. No of Coefficients	44
6.2	Decomposition Level vs. No of Coefficients	45

# Chapter 1

## Introduction

Speech is a very basic way for humans to convey information to one another. With a bandwidth of only 4kHz, speech can convey information with the emotion of a human voice. People want to be able to hear someone's voice from anywhere in the world – as if the person was in the same room.

As a result a greater emphasis is being placed on the design of new and efficient speech coders for voice communication and transmission. Today applications of speech coding and compression have become very numerous. Many applications involve the real time coding of speech signals, for use in mobile satellite communications, cellular telephony, and audio for videophones or video teleconferencing systems. Other applications include the storage of speech for speech synthesis and playback, or for the transmission of voice at a later time. Some examples include voice mail systems, voice memo wristwatches, voice logging recorders and interactive PC software.

Traditionally speech coders can be classified into two categories: waveform coders and analysis/synthesis vocoders (from “voice coders”). Waveform coders attempt to copy the actual shape of the signal produced by the microphone and its associated analogue circuits [9]. A popular waveform coding technique is pulse code modulation (PCM), which is used in telephony today.

Vocoders use an entirely different approach to speech coding, known as parameter-coding, or analysis/synthesis coding where no attempt is made at reproducing the exact speech waveform at the receiver, only a signal perceptually equivalent to it. These systems provide much lower data rates by using a functional model of the human speaking mechanism at the receiver. One of the most popular techniques for analysis-synthesis coding of speech is called Linear Predictive Coding (LPC).

Some higher quality vocoders include RELP (Residual Excited Linear Prediction) and CELP (Code Excited Linear Prediction) [6].

This thesis looks at a new technique for analysing and compressing speech signals using wavelets. Very simply wavelets are mathematical functions of finite duration with an average value of zero that are useful in representing data or other functions.

Any signal can be represented by a set of scaled and translated versions of a basic function called the “mother wavelet”. This set of wavelet functions forms the wavelet coefficients at different scales and positions and results from taking the wavelet transform of the original signal. The coefficients represent the signal in the wavelet domain and all data operations can be performed using just the corresponding wavelet coefficients.

Speech is a non-stationary random process due to the time varying nature of the human speech production system. Non-stationary signals are characterised by numerous transitory drifts, trends and abrupt changes. The localisation feature of wavelets, along with its time-frequency resolution properties makes them well suited for coding speech signals.

In designing a wavelet based speech coder, the major issues explored in this thesis are:

- i. Choosing optimal wavelets for speech,
- ii. Decomposition level in wavelet transforms,
- iii. Threshold criteria for the truncation of coefficients,
- iv. Efficiently representing zero valued coefficients and
- v. Quantising and digitally encoding the coefficients.

The performance of the wavelet compression scheme in coding speech signals and the quality of the reconstructed signals is also evaluated.

# Chapter 2

## Theory

This section presents some background information on wavelets and wavelet transforms, including their implementation using filters. This theory is necessary for understanding the material that follows in the Literature Review and certain other chapters of the thesis.

### 2.1 Introducing Wavelets

The fundamental idea behind wavelets is to analyse according to scale. The wavelet analysis procedure is to adopt a wavelet prototype function called an *analysing* wavelet or *mother* wavelet. Any signal can then be represented by translated and scaled versions of the mother wavelet.

Wavelet analysis is capable of revealing aspects of data that other signal analysis techniques such as Fourier analysis miss, aspects like trends, breakdown points, discontinuities in higher derivatives, and self-similarity. Furthermore, because it affords a different view of data than those presented by traditional techniques, it can compress or de-noise a signal without appreciable degradation [7].

### 2.2 Wavelet vs. Fourier Analysis

#### 2.2.1 General Concepts

In the well-known Fourier analysis, a signal is broken down into constituent sinusoids of different frequencies. These sines and cosines (essentially complex exponentials) are the basis functions and the elements of Fourier synthesis.

Taking the Fourier transform of a signal can be viewed as a rotation in the function space of the signal from the time domain to the frequency domain.

Similarly, the wavelet transform can be viewed as transforming the signal from the time domain to the wavelet domain. This new domain contains more complicated basis functions called wavelets, mother wavelets or analysing wavelets.

Mathematically, the process of Fourier analysis is represented by the **Fourier transform**:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt \quad (2.1)$$

which is the sum over all time of the signal  $f(t)$  multiplied by a complex exponential.

The results of the transform are the Fourier coefficients  $F(\omega)$ , which when multiplied by a sinusoid of frequency  $\omega$ , yield the constituent sinusoidal components of the original signal.

A wavelet prototype function at a scale  $s$  and a spatial displacement  $u$  is defined as:

$$\Psi_{s,u}(x) = \sqrt{s} \Psi \left[ \frac{(x-u)}{s} \right] \quad (2.2)$$

Replacing the complex exponential in Equation 2.1 with this function yields the **continuous wavelet transform** (CWT):

$$C(s, u) = \int_{-\infty}^{\infty} f(t) \sqrt{s} \Psi \left[ \frac{(x-u)}{s} \right] dt \quad (2.3)$$

which is the sum over all time of the signal multiplied by scaled and shifted versions of the wavelet function  $\psi$ . The results of the CWT are many wavelet coefficients  $C$ , which are a function of scale and position. Multiplying each coefficient by the appropriately scaled and shifted wavelet yields the constituent wavelets of the original signal.

The basis functions in both Fourier and wavelet analysis are localised in frequency making mathematical tools such as power spectra (power in a frequency interval) useful at picking out frequencies and calculating power distributions.

The most important difference between these two kinds of transforms is that individual wavelet functions are localised in space. In contrast Fourier sine and cosine functions are non-local and are active for all time  $t$ .

This localisation feature, along with wavelets localisation of frequency, makes many functions and operators using wavelets “sparse” when transformed into the wavelet domain. This sparseness, in turn results in a number of useful applications such as data compression, detecting features in images and de-noising signals.

### **2.2.2 Time-Frequency Resolution**

A major draw back of Fourier analysis is that in transforming to the frequency domain, the time domain information is lost. When looking at the Fourier transform of a signal, it is impossible to tell when a particular event took place.

In an effort to correct this deficiency, Dennis Gabor (1946) adapted the Fourier transform to analyse only a small section of the signal at a time – a technique called windowing the signal [14]. Gabor’s adaptation, called the Windowed Fourier Transform (WFT) gives information about signals simultaneously in the time domain and in the frequency domain.

To illustrate the time-frequency resolution differences between the Fourier transform and the wavelet transform consider the following figures.

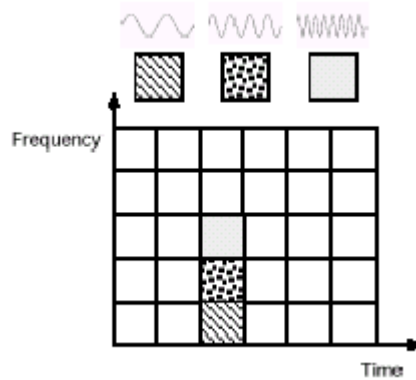


Figure 2.1: WFT Resolution

Figure 2.1 shows a windowed Fourier transform, where the window is simply a square wave. The square wave window truncates the sine or cosine function to fit a window of a particular width. Because a single window is used for all frequencies in the WFT, the resolution of the analysis is the same at all locations in the time frequency plane.

An advantage of wavelet transforms is that the windows vary. Wavelet analysis allows the use of long time intervals where we want more precise low-frequency information, and shorter regions where we want high-frequency information. A way to achieve this is to have short high-frequency basis functions and long low-frequency ones.

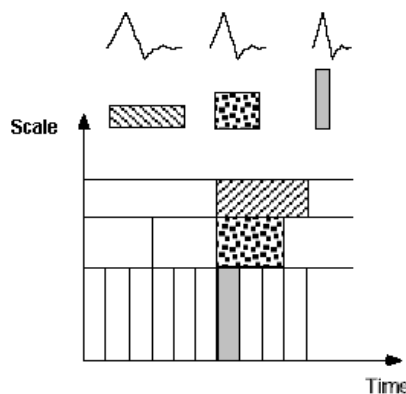


Figure 2.2: Wavelets Resolution

Figure 2.2 shows a time-scale view for wavelet analysis rather than a time frequency region. Scale is inversely related to frequency. A low-scale compressed wavelet with rapidly changing details corresponds to a high frequency. A high-scale stretched wavelet that is slowly changing has a low frequency.

## 2.3 Examples of Wavelets

The figure below illustrates four different types of wavelet basis functions.

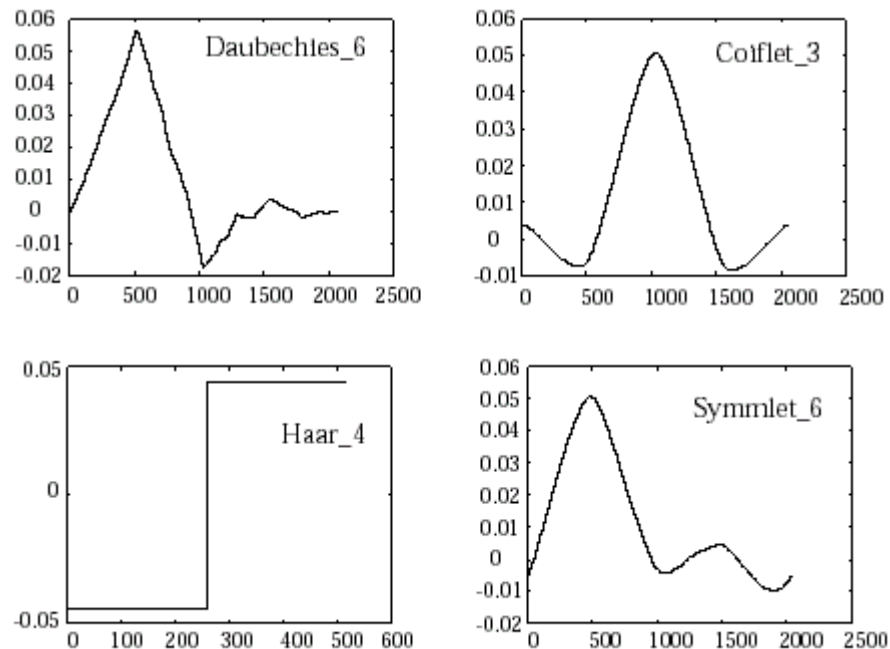


Figure 2.3: Different wavelet families [7]

The different families make trade-offs between how compactly the basis functions are localised in space and how smooth they are.

Within each family of wavelets (such as the Daubechies family) are wavelet subclasses distinguished by the number of filter coefficients and the level of iteration. Wavelets are most often classified within a family by the number of *vanishing moments*. This is an extra set of mathematical relationships for the coefficients that must be satisfied. The extent of compactness of signals depends on the number of vanishing moments of the wavelet function used. A more detailed discussion is provided in the next section.

## 2.4 The Discrete Wavelet Transform ([12], [14], [16], [17])

The Discrete Wavelet Transform (DWT) involves choosing scales and positions based on powers of two – so called dyadic scales and positions. The mother wavelet is rescaled or “dilated”, by powers of two and translated by integers.

Specifically, a function  $f(t) \in L^2(\mathbf{R})$  (defines space of square integrable functions) can be represented as

$$f(t) = \sum_{j=1}^L \sum_{k=-\infty}^{\infty} d(j, k) \psi(2^{-j}t - k) + \sum_{k=-\infty}^{\infty} a(L, k) \phi(2^{-L}t - k) \quad (2.4)$$

The function  $\psi(t)$  is known as the mother wavelet, while  $\phi(t)$  is known as the scaling function. The set of functions  $\{\sqrt{2^{-l}} \phi(2^{-l}t - k), \sqrt{2^{-j}} \psi(2^{-j}t - k) \mid j \leq L, j, k, L \in \mathbf{Z}\}$ , where  $\mathbf{Z}$  is the set of integers, is an orthonormal basis for  $L^2(\mathbf{R})$ .

The numbers  $a(L, k)$  are known as the approximation coefficients at scale  $L$ , while  $d(j, k)$  are known as the detail coefficients at scale  $j$ .

The approximation and detail coefficients can be expressed as:

$$a(L, k) = \frac{1}{\sqrt{2^L}} \int_{-\infty}^{\infty} f(t) \phi(2^{-L}t - k) dt \quad (2.5)$$

$$d(j, k) = \frac{1}{\sqrt{2^j}} \int_{-\infty}^{\infty} f(t) \psi(2^{-j}t - k) dt \quad (2.6)$$

To provide some understanding of the above coefficients consider a projection  $f_l(t)$  of the function  $f(t)$  that provides the best approximation (in the sense of minimum error energy) to  $f(t)$  at a scale  $l$ . This projection can be constructed from the coefficients  $a(L, k)$ , using the equation

$$f_l(t) = \sum_{k=-\infty}^{\infty} a(l, k) \phi(2^{-l}t - k).$$

As the scale  $l$  decreases, the approximation becomes finer, converging to  $f(t)$  as  $l \rightarrow \infty$ . The difference between the approximation at scale  $l + 1$  and that at  $l$ ,  $f_{l+1}(t) - f_l(t)$ , is completely described by the coefficients  $d(j, k)$  using the equation

$$f_{l+1}(t) - f_l(t) = \sum_{k=-\infty}^{\infty} d(l, k) \psi(2^{-l}t - k).$$

Using these relations, given  $a(L, k)$  and  $\{d(j, k) \mid j \leq L\}$ , it is clear that we can build the approximation at any scale. Hence, the wavelet transform breaks the signal up into a coarse approximation  $f_L(t)$  (given  $a(L, k)$ ) and a number of layers of detail  $\{f_{j+1}(t) - f_j(t) \mid j < L\}$  (given by  $\{d(j, k) \mid j \leq L\}$ ). As each layer of detail is added, the approximation at the next finer scale is achieved.

#### 2.4.1 Vanishing Moments

The number of vanishing moments of a wavelet indicates the smoothness of the wavelet function as well as the flatness of the frequency response of the wavelet filters (filters used to compute the DWT) [20].

Typically a wavelet with  $p$  vanishing moments satisfies the following equation [17]:

$$\int_{-\infty}^{\infty} t^m \psi(t) dt = 0 \quad \text{for } m = 0, \dots, p-1,$$

or equivalently,

$$\sum_k (-1)^k k^m c(k) = 0 \quad \text{for } m = 0, \dots, p-1.$$

For the representation of smooth signals, a higher number of vanishing moments leads to a faster decay rate of wavelet coefficients. Thus, wavelets with a high number of

vanishing moments lead to a more compact signal representation and are hence useful in coding applications.

However, in general, the length of the filters increases with the number of vanishing moments and the complexity of computing the DWT coefficients increases with the size of the wavelet filters.

## 2.5 The Fast Wavelet Transform Algorithm

The Discrete Wavelet Transform (DWT) coefficients can be computed by using Mallat's Fast Wavelet Transform algorithm. This algorithm is sometimes referred to as the *two-channel sub-band coder* and involves filtering the input signal based on the wavelet function used.

### 2.5.1 Implementation Using Filters

To explain the implementation of the Fast Wavelet Transform algorithm consider the following equations:

$$\phi(t) = \sum_k c(k)\phi(2t - k) \quad (2.7)$$

$$\psi(t) = \sum_k (-1)^k c(1 - k)\phi(2t - k) \quad (2.8)$$

$$\sum_k c_k c_{k-2m} = 2\delta_{0,m} \quad (2.9)$$

The first equation is known as the *twin-scale relation* (or the dilation equation) and defines the scaling function  $\phi$ . The next equation expresses the wavelet  $\psi$  in terms of the scaling function  $\phi$ . The third equation is the condition required for the wavelet to be orthogonal to the scaling function and its translates.

The coefficients  $c(k)$  or  $\{c_0, \dots, c_{2N-1}\}$  in the above equations represent the impulse response coefficients for a low pass filter of length  $2N$ , with a sum of 1 and a norm of  $\frac{1}{\sqrt{2}}$ .

The high pass filter is obtained from the low pass filter using the relationship  $g_k = (-1)^k c(1 - k)$ , where  $k$  varies over the range  $(1 - (2N - 1))$  to 1.

Equation 2.7 shows that the scaling function is essentially a low pass filter and is used to define the approximations. The wavelet function defined by equation 2.8 is a high pass filter and defines the details.

Starting with a discrete input signal vector  $s$ , the first stage of the FWT algorithm decomposes the signal into two sets of coefficients. These are the approximation coefficients  $cA_1$  (low frequency information) and the detail coefficients  $cD_1$  (high frequency information), as shown in the figure below.

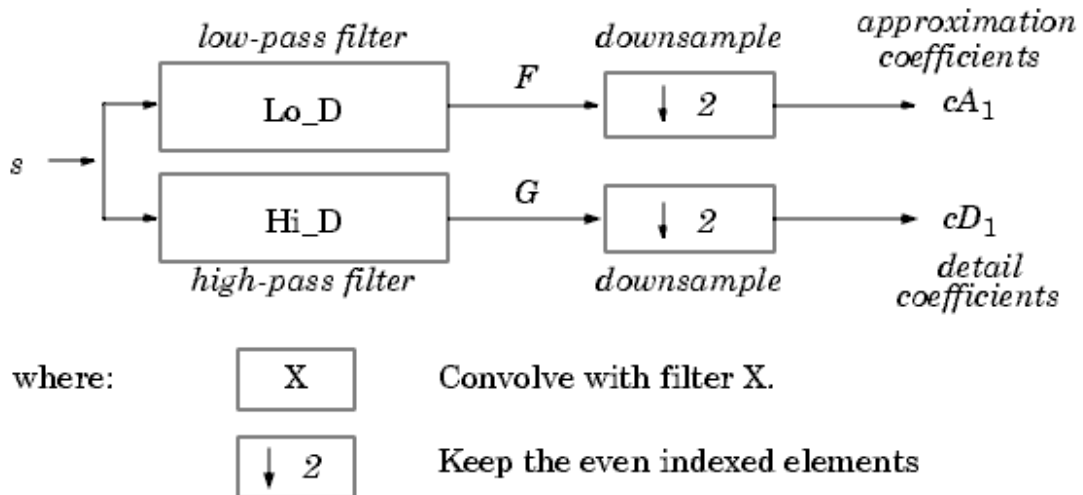


Figure 2.4: Filtering operation of the DWT [15]

The coefficient vectors are obtained by convolving  $s$  with the low-pass filter  $Lo\_D$  for approximation and with the high-pass filter  $Hi\_D$  for details. This filtering operation is then followed by dyadic decimation or down sampling by a factor of 2.

Mathematically the two-channel filtering of the discrete signal  $s$  is represented by the expressions:

$$cA_1 = \sum_k c_k s_{2i-k}, \quad cD_1 = \sum_k g_k s_{2i-k} \quad (2.10)$$

These equations implement a convolution plus down sampling by a factor 2 and give the forward fast wavelet transform.

If the length of each filter is equal to  $2N$  and the length of the original signal  $s$  is equal to  $n$ , then the corresponding lengths of the coefficients  $cA_1$  and  $cD_1$  are given by the formula:

$$\text{floor}\left(\frac{n-1}{2}\right) + N \quad (2.11)$$

This shows that the total length of the wavelet coefficients is always slightly greater than the length of the original signal due to the filtering process used.

### 2.5.2 Multilevel Decomposition

The decomposition process can be iterated, with successive approximations being decomposed in turn, so that one signal is broken down into many lower resolution components. This is called the wavelet decomposition tree.

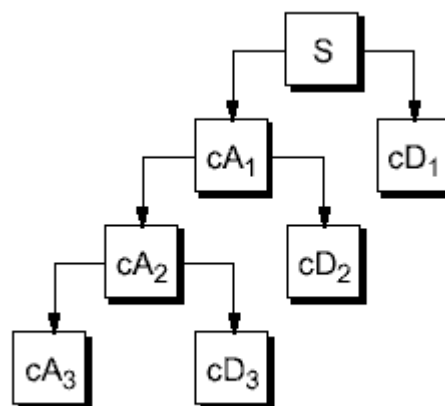


Figure 2.5: Decomposition of DWT coefficients

The wavelet decomposition of the signal  $s$  analysed at level  $j$  has the following structure  $[cA_j, cD_j, \dots, cD_1]$ .

Looking at a signal's wavelet decomposition tree can reveal valuable information. The diagram below shows the wavelet decomposition to level 3 of a sample signal  $S$ .

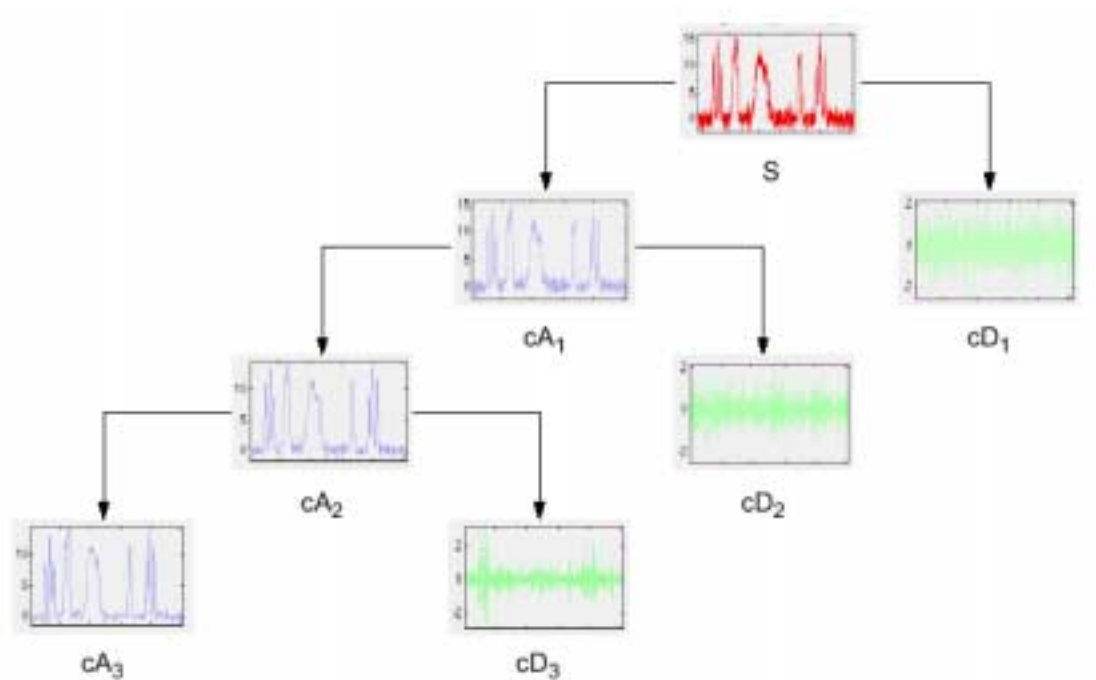


Figure 2.6: Level 3 Decomposition of Sample Signal  $S$  [15]

Since the analysis process is iterative, in theory it can be continued indefinitely. In reality, the decomposition can only proceed until the vector consists of a single sample. Normally, however there is little or no advantage gained in decomposing a signal beyond a certain level. The selection of the optimal decomposition level in the hierarchy depends on the nature of the signal being analysed or some other suitable criterion, such as the low-pass filter cut-off.

## 2.6 Signal Reconstruction

The original signal can be reconstructed or synthesised using the inverse discrete wavelet transform (IDWT).

The synthesis starts with the approximation and detail coefficients  $cA_j$  and  $cD_j$ , and then reconstructs  $cA_{j-1}$  by up sampling and filtering with the reconstruction filters.

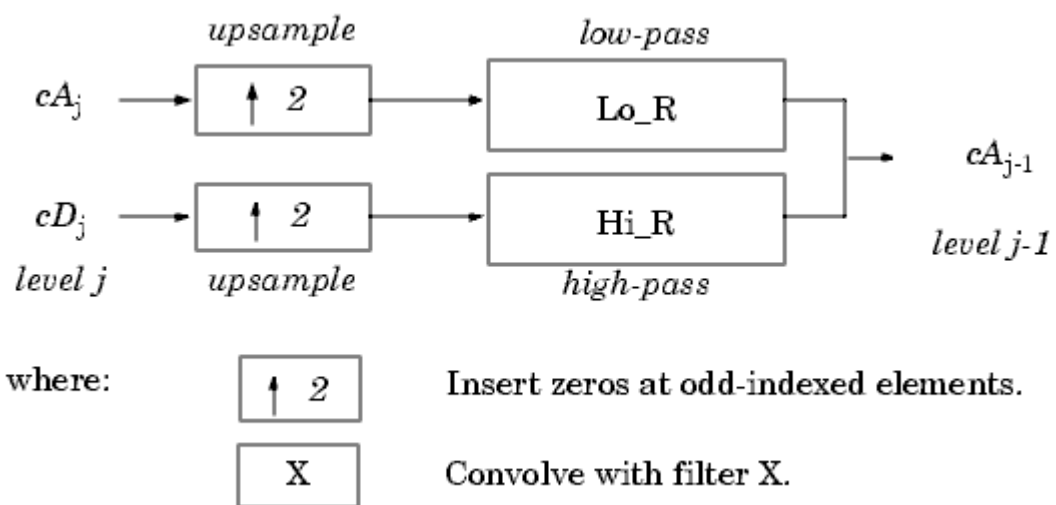


Figure 2.7: Wavelets Reconstruction [15]

The reconstruction filters are designed in such a way to cancel out the effects of aliasing introduced in the wavelet decomposition phase. The reconstruction filters (Lo\_R and Hi\_R) together with the low and high pass decomposition filters, forms a system known as *quadrature mirror filters* (QMF).

For a multilevel analysis, the reconstruction process can itself be iterated producing successive approximations at finer resolutions and finally synthesising the original signal.

# Chapter 3

## Literature Review

This chapter summarises all the relevant literature researched during the course of this thesis. It presents certain approaches used by many researchers to compress speech signals using wavelets. It also compares the performance of wavelet speech signal compression with other common speech coding techniques. Finally two “state of the art” real time high quality audio compression examples based on wavelets are illustrated.

### **3.1 Wavelet Speech Compression Techniques**

#### **3.1.1 Introduction**

The idea behind signal compression using wavelets is primarily linked to the relative scarceness of the wavelet domain representation for the signal. Wavelets concentrate speech information (energy and perception) into a few neighbouring coefficients [11]. Therefore as a result of taking the wavelet transform of a signal, many coefficients will either be zero or have negligible magnitudes.

Data compression is then achieved by treating small valued coefficients as insignificant data and thus discarding them. The process of compressing a speech signal using wavelets involves a number of different stages, each of which are discussed below.

#### **3.1.2 Choice of Wavelet**

The choice of the mother-wavelet function used in designing high quality speech coders is of prime importance. Choosing a wavelet that has compact support in both time and

frequency in addition to a significant number of vanishing moments is essential for an optimum wavelet speech compressor [10].

Several different criteria can be used in selecting an optimal wavelet function. The objective is to minimise reconstructed error variance and maximise signal to noise ratio (SNR). In general optimum wavelets can be selected based on the energy conservation properties in the approximation part of the wavelet coefficients.

In [1] it was shown that the Battle-Lemarie wavelet concentrates more than 97.5% of the signal energy in the approximation part of the coefficients. This is followed very closely by the Daubechies D20, D12, D10 or D8 wavelets, all concentrating more than 96% of the signal energy in the Level 1 approximation coefficients.

Wavelets with more vanishing moments provide better reconstruction quality, as they introduce less distortion into the processed speech and concentrate more signal energy in a few neighbouring coefficients. However the computational complexity of the DWT increases with the number of vanishing moments and hence for real time applications it is not practical to use wavelets with an arbitrarily high number of vanishing moments [20].

### **3.1.3 Wavelet Decomposition**

Wavelets work by decomposing a signal into different resolutions or frequency bands, and this task is carried out by choosing the wavelet function and computing the Discrete Wavelet Transform (DWT) [5]. Signal compression is based on the concept that selecting a small number of approximation coefficients (at a suitably chosen level) and some of the detail coefficients can accurately represent regular signal components.

Choosing a decomposition level for the DWT usually depends on the type of signal being analysed or some other suitable criterion such as entropy. For the processing of speech signals decomposition up to scale 5 is adequate [1], with no further advantage gained in processing beyond scale 5.

### 3.1.4 Truncation of Coefficients

After calculating the wavelet transform of the speech signal, compression involves truncating wavelet coefficients below a threshold. An experiment conducted on a male spoken sentence [11], shows that most of the coefficients have small magnitudes. More than 90% of the wavelet coefficients have less than 5% of the maximum value.

This means that most of the speech energy is in the high-valued coefficients, which are few [11]. Thus the small valued coefficients can be truncated or zeroed and then be used to reconstruct the signal. This compression scheme provided a segmental signal-to-noise ratio (SEGSNR) of 20 dB, with only 10% of the coefficients.

Two different approaches are available for calculating thresholds. The first, known as **Global Thresholding** involves taking the wavelet expansion of the signal and keeping the largest absolute value coefficients. In this case you can manually set a global threshold, a compression performance or a relative square norm recovery performance. Thus only a single parameter needs to be selected.

The second approach known as **By Level Thresholding** consists of applying visually determined level dependent thresholds to each decomposition level in the wavelet transform.

### 3.1.5 Encoding Coefficients

Signal compression is achieved by first truncating small-valued coefficients and then efficiently encoding them.

One way of representing the high-magnitude coefficients is to store the coefficients along with their respective positions in the wavelet transform vector [5]. For a speech signal of frame size  $F$ , taking the DWT generates a frame of size  $T$ , slightly larger than  $F$ . If only the largest  $L$  coefficients are retained, then the compression ratio  $C$  is given

by: 
$$C = \frac{F}{2L}.$$

Another approach to compression is to encode consecutive zero valued coefficients [11], with two bytes. One byte to indicate a sequence of zeros in the wavelet transforms vector and the second byte representing the number of consecutive zeros.

For further data compaction a suitable bit encoding format, can be used to quantise and transmit the data at low bit rates. A low bit rate representation can be achieved by using an entropy coder like Huffman coding or arithmetic coding.

### **3.1.6 Detecting Voiced vs. Unvoiced Speech Frames**

In speech there are two major types of excitation, voiced and unvoiced. Voiced sounds are produced when air flows between the vocal cords and causes them to vibrate [13]. Voiced speech tends to be periodic in nature. Examples of voiced sounds are English vowels, such as the /a/ in “bay” and the /e/ in “see”.

Unvoiced sounds result from constricting the vocal tract at some point so that turbulence is produced by air flowing past the constriction. Since unvoiced speech is due to turbulence, the speech is aperiodic and has a noise-like structure. Some examples of unvoiced English sounds are the /s/ in “so” and the /h/ in “he”.

In general at least 90% of the speech energy is always retained in the first  $N/2$  transform coefficients, if the speech is a voiced frame [1]. However, for an unvoiced frame the energy is spread across several frequency bands and typically the first  $N/2$  coefficients holds less than 40% of the total energy. Due to this wavelets are inefficient at coding unvoiced speech.

Unvoiced speech frames are infrequent. By detecting unvoiced speech frames and directly encoding them (perhaps using entropy coding), no unvoiced data is lost and the quality of the compressed speech will remain transparent.

### **3.1.7 Performance Relative to Other Coding Schemes**

In [11] a wavelet based speech coder achieved a 12.82 dB SNR on a male spoken sentence, at a bit rate of less than 4.8 kbits/s with 90% truncation. In comparison today's state of the art CELP (code excited linear predictive coding) at 4.8 Kbits/s has an SNR of 10-13 dB.

In [5] wavelet based compression was compared with two forms of Linear Predictive Coding (LPC-10 and LPC), Global System Mobile (GSM) and Adaptive Differential Pulse Code Modulation (ADPCM). The compression ratios for these schemes are 26:1, 12:1, 5:1 and 2:1 respectively. Using wavelets at these compression ratios gave higher SNR and better speech quality than Linear Predictive coding and comparable speech quality as the GSM and ADPCM schemes.

## **3.2 Real Time Audio Compression Examples**

In [20] the real-time implementation of a wavelet-based monophonic full-duplex audio coder using two T1 TMS320C31 DSP processors is described. The encoder uses a 33 MHz clock and takes about 92% of real time. The decoder uses a 40 MHz clock and requires 61% of real time. The implementation yields almost transparent compression of monophonic CD quality audio signals at bit rates of 64-78 Kbits/s.

The coder uses a tree-structured DWT as shown in Figure 3.1. This wavelet transform is known as the discrete wavelet packet and gives freedom in choosing the frequency resolution over all the frequency range according to the specified application needs. The numbers in the figure correspond to different frequency regions in KHz.

In this tree-structured wavelet transform, the detail or high frequency data is also decomposed giving the transform a binary tree structure. This tree has been designed in such a way as to closely match the frequency division of each transform band to that of the critical band of human ear, especially at the high frequency region.



The Clarinet is one example of a signal with strong periodic structures [3]. This portion is represented by sinusoidal basis functions. After removing the tones from the original signal a residual signal remains, which consists of transient and noise-like features. The wavelet transform is then performed on the residual. The figure below depicts a Clarinet signal with the harmonic and residual signals.

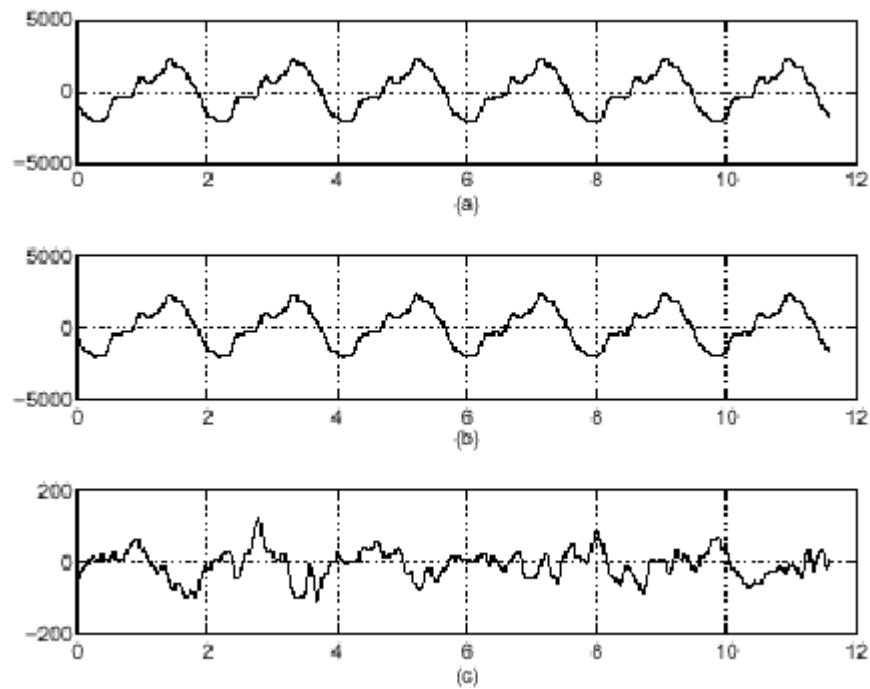


Figure 3.2: (a) A Clarinet Signal, (b) Tonal Signal, (c) Residual Signal [3]

This coder uses a combined method, as wavelet transforms do not provide compact representation for tonal signals. Existing coders using Fourier transforms perform better for harmonic analysis and hence wavelet analysis was done only on the residual signal.

# Chapter 4

## System Implementation

### 4.1 Design Overview

The design of the wavelet transform speech coder is based on the concepts covered in the Literature Review (section 3.1). The figure below illustrates the different processes involved in compressing speech signals using wavelets. In this thesis these different stages were designed and coded in software, using Matlab version 6, with the exception of the last two processes.

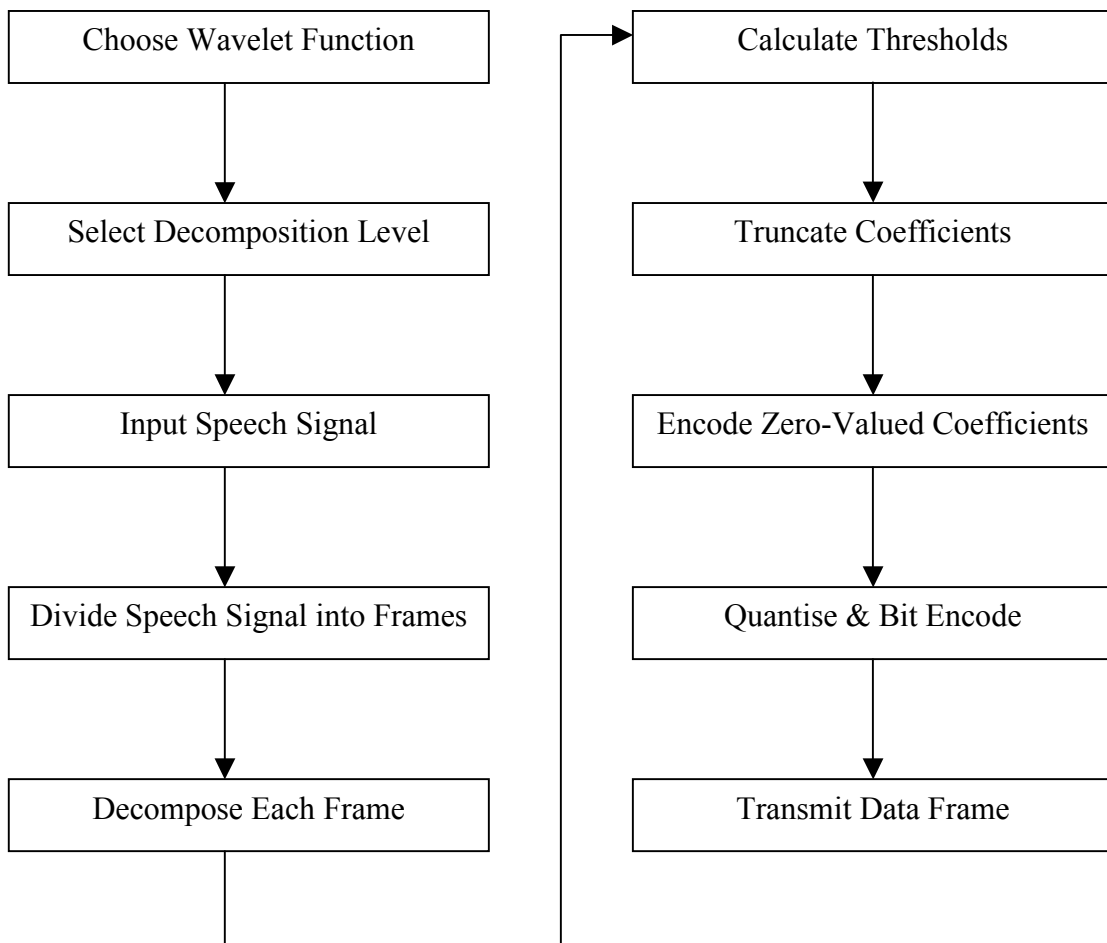


Figure 4.1: Design Flow of Wavelet Based Speech Coder

## 4.2 Software Implementation

The underlying source code for the implementation of the wavelet based speech coder is provided in the Appendices. This code listing however only contains the important functions and script files in the program that demonstrates the design of the speech compression software.

### 4.2.1 Wavelet Functions and Transforms

The capabilities of Matlab's Wavelet Toolbox were utilised in this section. The Wavelet Toolbox incorporates many different wavelet families and the coefficients of their QMF filters. From the literature reviewed it was decided to use the Haar and Daubechies wavelets for coding speech signals.

The Wavelet Toolbox's built in functions *dwt*, *wavedec*, *waverec* and *idwt*, were used to compute the forward and inverse wavelet transforms. *Wavedec* computes the multi-level decomposition of a signal and *waverec* reconstructs the signal from the coefficients. The script file for the multi-level decomposition implements a recursive algorithm, involving the circular convolution of the sample input signal vector and the wavelet filter coefficients followed by dyadic decimation.

### 4.2.2 Speech Signals

The sample speech files used for compression are .OD files. These files contain discrete signal values, which can be easily read in and played by Matlab at a sampling frequency of 8 KHz. Alternatively WAV files could also be used and processed.

Matlab 6 uses an interpreter to run any code written without actually compiling the code. Due to this it is far too slow to design and implement a real time speech coder in Matlab alone. Thus real time speech coding could only be simulated by dividing the input sample speech into frames of 20 ms (160 samples) and then decomposing and compressing each frame. For processing recorded speech however larger frame sizes

can be used. Since the speech files used in this design are of very short duration (few seconds), the entire speech vector could be decomposed without dividing it into frames.

### 4.2.3 Calculating Thresholds

For the truncation of small-valued transform coefficients, two different thresholding techniques are used, Global Thresholding and By-Level Thresholding.

The aim of Global Thresholding is to retain the largest absolute value coefficients, regardless of the scale in the wavelet decomposition tree. Global thresholds are calculated by setting the % of coefficients to be truncated.

Level dependent thresholds are calculated using the Birge-Massart strategy [15]. This thresholding scheme is based on an approximation result from Birge and Massart and is well suited for signal compression.

This strategy keeps all of the approximation coefficients at the level of decomposition  $J$ . The number of detail coefficients to be kept at level  $i$  starting from 1 to  $J$  are given by the formula:

$$n_i = \frac{M}{(J + 2 - i)^\alpha}$$

$\alpha$  is a compression parameter and its value is typically 1.5. The value of  $M$  denotes the how scarcely distributed the wavelet coefficients are in the transform vector. If  $L$  denotes the length of the coarsest approximation coefficients then  $M$  takes on the values in Table 4.1, depending on the signal being analysed.

<i>Scarce</i>	<i>M</i>
High	L
Medium	1.5*L
Low	2*L

Table 4.1: Values for M

Thus this approach to thresholding selects the highest absolute valued coefficients at each level.

#### **4.2.4 Encoding Zero-Valued Coefficients**

After zeroing wavelet coefficients with negligible values based on either calculating threshold values or simply selecting a truncation percentage, the transform vector needs to be compressed. In this implementation consecutive zero valued coefficients are encoded with two bytes. One byte is used to specify a starting string of zeros and the second byte keeps track of the number of successive zeros.

Due to the scarcity of the wavelet representation of the speech signal, this encoding method leads to a higher compression ratios than storing the non-zero coefficients along with their respective positions in the wavelet transform vector, as suggested in the Literature Review (Section 3.1.4). This encoding scheme is the primary means of achieving signal compression. In Matlab however, the coding of this compression algorithm using vectors results in relatively slow performance, with unacceptable delays for real time speech coding. This encoding process can be speeded up significantly by programming it in another language such as C++.

# Chapter 5

## Results

### 5.1 Optimal Decomposition Level in Wavelet Transforms

The figure below shows a sample speech signal and approximations of the signal, at five different scales. These approximations are reconstructed from the coarse low frequency coefficients in the wavelet transform vector.

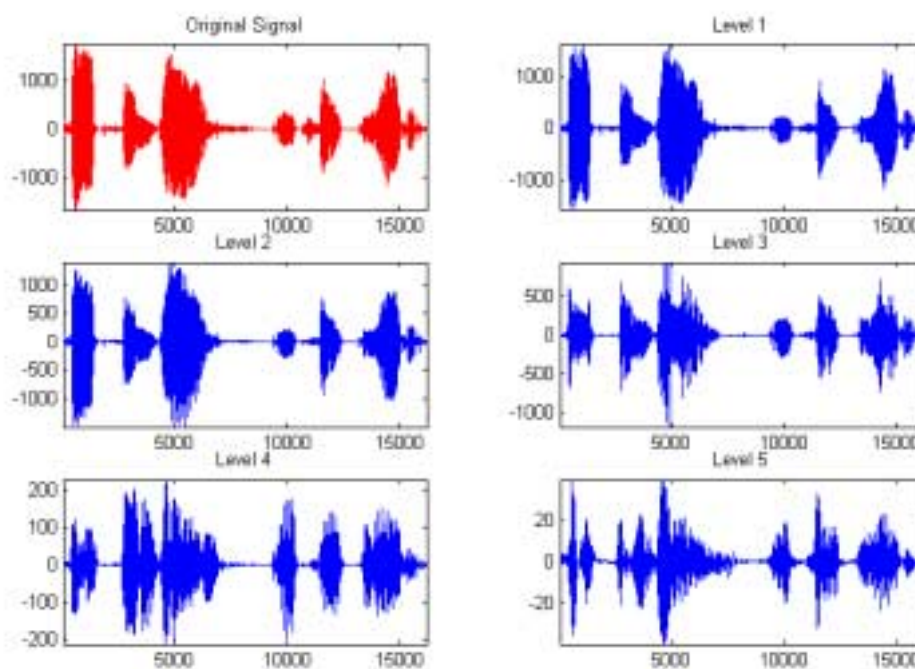


Figure 5.1: Original Speech Signal and Reconstructed Approximations at Different Scales

## 5.2 Retained Energy in First N/2 Coefficients

A suitable criterion for selecting optimum mother wavelets is related to the amount of energy a wavelet basis function can concentrate into the level 1-approximation coefficients.

A speech signal was divided into frames of size 1024 samples and then analysed using different wavelets. The wavelet transform was computed to scale 5. The table below shows the signal energy retained in the first N/2 transform coefficients, over the 16 frames. This energy is equivalent to the energy stored in the level 1-approximation coefficients.

<b>Frame</b>	<b>Haar</b>	<b>Db4</b>	<b>Db6</b>	<b>Db8</b>	<b>Db10</b>
1	88.92	94.30	95.40	94.24	94.47
2	89.35	94.59	95.06	96.33	95.71
3	93.45	96.12	96.63	96.71	96.28
4	96.20	98.74	98.77	99.09	99.21
5	94.15	99.02	98.46	97.27	98.65
6	93.72	99.56	99.50	99.38	99.37
7	94.81	98.46	98.42	98.61	98.71
8	37.58	36.49	37.52	34.28	35.44
9	77.10	76.05	79.63	81.07	80.04
10	89.64	89.71	90.17	90.27	90.62
11	35.86	31.36	31.24	49.90	61.30
12	89.95	93.19	90.91	91.79	93.39
13	87.43	85.61	85.26	86.22	88.22
14	93.03	95.47	95.61	95.31	95.91
15	93.68	99.01	99.25	99.19	99.34
16	94.14	97.90	99.55	98.96	98.68
<b>Avg. Energy</b>	<b>84.31</b>	<b>86.60</b>	<b>86.96</b>	<b>88.04</b>	<b>89.08</b>

Table 5.1: Percentage of Energy Concentrated in N/2 coefficients

### 5.3 Voiced, Unvoiced and Mixed Speech Frames

From the previous speech signal analysed three different types of speech segments can be identified, based on the amount of energy the wavelet concentrates in the first  $N/2$  coefficients.

The figures below show each speech frame with its wavelet decomposition at level 5, using the Daubechies 10 wavelet. The structure of the plotted wavelet transform vector is  $[cA_5, cD_5, cD_4, cD_3, cD_2, cD_1]$  and the lengths of the respective coefficients are given by  $(50, 50, 81, 144, 270, 521)$ . The total length of the coefficients is 1116, which is greater than the frame size of 1024.

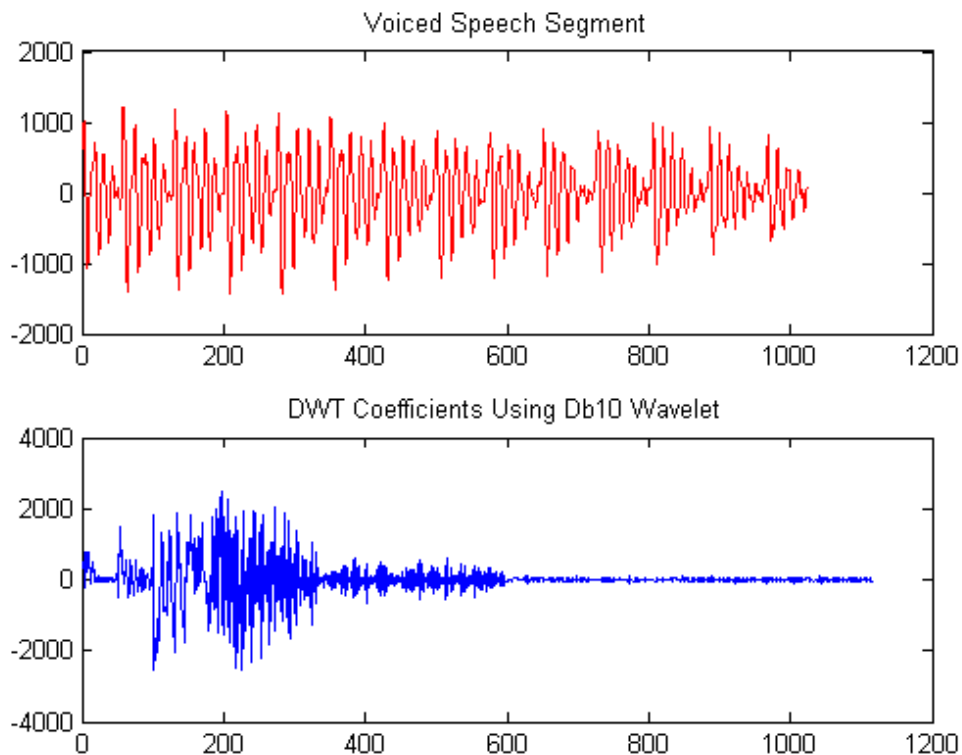


Figure 5.2: A Voiced Speech Segment and its DWT at Scale 5

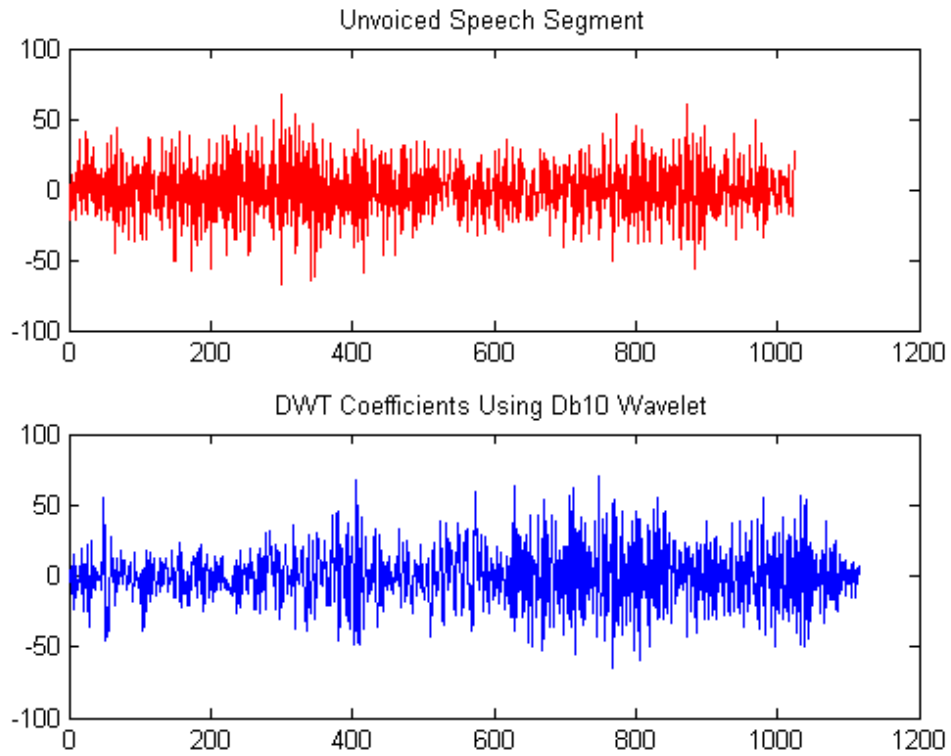


Figure 5.3: An Unvoiced Speech Segment and its DWT at Scale 5

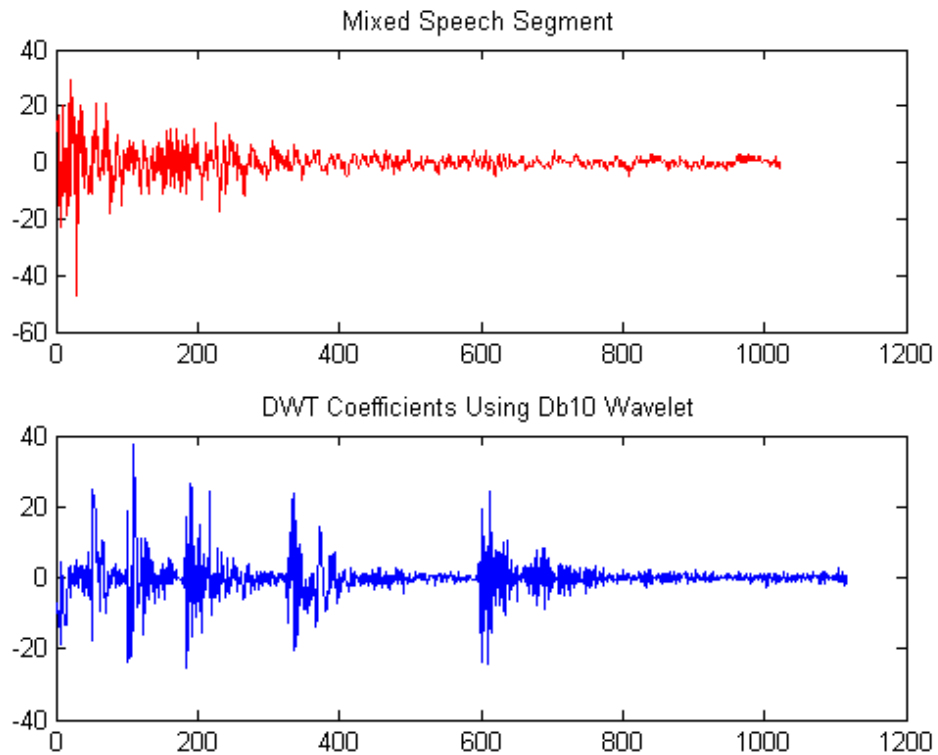


Figure 5.4: A Mixed Speech Segment and its DWT at Scale 5

## 5.4 Performance Measures

A number of quantitative parameters can be used to evaluate the performance of the wavelet based speech coder, in terms of both reconstructed signal quality after decoding and compression scores. The following parameters are compared:

- Signal to Noise Ratio (SNR),
- Peak Signal to Noise Ratio (PSNR),
- Normalised Root Mean Square Error (NRMSE),
- Retained Signal Energy
- Compression Ratios

The results obtained for the above quantities are calculated using the following formulas:

### 1. Signal to Noise Ratio [5]

$$SNR = 10 \log_{10} \left( \frac{\sigma_x^2}{\sigma_e^2} \right)$$

$\sigma_x^2$  is the mean square of the speech signal and  $\sigma_e^2$  is the mean square difference between the original and reconstructed signals.

### 2. Peak Signal to Noise Ratio [5]

$$PSNR = 10 \log_{10} \frac{NX^2}{\|x - r\|^2}$$

N is the length of the reconstructed signal, X is the maximum absolute square value of the signal x and  $\|x - r\|^2$  is the energy of the difference between the original and reconstructed signals.

### 3. Normalised Root Mean Square Error [5]

$$NRMSE = \sqrt{\frac{(x(n) - r(n))^2}{(x(n) - \mu_x(n))^2}}$$

$x(n)$  is the speech signal,  $r(n)$  is the reconstructed signal, and  $\mu_x(n)$  is the mean of the speech signal.

### 4. Retained Signal Energy

$$RSE = \frac{100 * \|x(n)\|^2}{\|r(n)\|^2}$$

$\|x(n)\|$  is the norm of the original signal and  $\|r(n)\|$  is the norm of the reconstructed signal. For one-dimensional orthogonal wavelets the retained energy is equal to the  $L^2$ -norm recovery performance.

### 5. Compression Ratio

$$C = \frac{Length(x(n))}{Length(cWC)}$$

cWC is the length of the compressed wavelet transform vector.

## 5.5 Performance of Recorded Speech Coding

A male and female spoken speech signals were decomposed at scale 3 and level dependent thresholds were applied using the Birge-Massart strategy (Section 4.2.3). Since the speech files were of short duration, the entire signal was decomposed at once without framing. A summary of the performance is given below for the different wavelets used.

### a) Male spoken sentence: "Cats and dogs each hate the other"

Wavelet	Zeros (%)	Retained Energy (%)	SNR	PSNR	NRMSE
Haar	79.2184	93.3054	11.7428	92.7803	0.2587
Db4	79.0749	97.8056	16.5870	97.6245	0.1481
Db6	79.0609	97.9450	16.8722	97.9098	0.1433
Db8	79.0381	98.0706	17.1463	98.1838	0.1389
Db10	79.0118	98.2008	17.4502	98.4877	0.1341

### b) Female spoken sentence: "The pipe began to rust while..."

Wavelet	Zeros (%)	Retained Energy (%)	SNR	PSNR	NRMSE
Haar	79.1756	90.2214	10.0972	91.9450	0.3127
Db4	79.0771	95.4705	13.4386	95.2864	0.2128
Db6	79.0560	95.9561	13.9378	95.7947	0.2007
Db8	79.0386	96.2727	14.2855	96.1318	0.1931
Db10	79.0111	96.3477	14.3866	96.2324	0.1909

### c) Compression Scores

Wavelet	Male	Female
Haar	3.5438	3.4658
Db4	3.7742	3.5657
Db6	3.8220	3.5988
Db8	3.8250	3.6352
Db10	3.8753	3.6446

Table 5.2 a), b), c): Performance of Recorded Speech Compression

## 5.6 Performance of Real Time Speech Coding

In this section the same speech files were used, except the signals were divided into frames of length 20ms to simulate real time coding. Each frame was then decomposed, truncated and compressed. A level 5, wavelet decomposition was applied along with global thresholds of close to 78-79% to enable comparisons with the previous results. A summary of the performance is given below for the different wavelets used.

### a) Male spoken sentence: "Cats and dogs each hate the other"

Wavelet	Zeros (%)	Retained Energy (%)	SNR	PSNR	NRMSE
Haar	78.8174	83.6182	9.1507	90.1882	0.3487
Db4	78.9216	89.4932	10.7339	91.8503	0.2880
Db6	78.7505	88.6178	11.0929	91.9349	0.2852
Db8	78.3683	88.4611	9.7060	90.8212	0.3242
Db10	77.9373	88.4156	9.6944	90.9158	0.3207

### b) Female spoken sentence: "The pipe began to rust while..."

Wavelet	Zeros (%)	Retained Energy (%)	SNR	PSNR	NRMSE
Haar	78.8531	85.7896	8.2749	90.1227	0.3857
Db4	79.0593	92.2694	10.5569	92.3711	0.2977
Db6	79.0560	92.2768	10.7696	92.5368	0.2921
Db8	78.9169	89.8339	8.5702	90.3742	0.3747
Db10	78.4119	88.8729	8.1457	90.1153	0.3860

### c) Compression Scores

Wavelet	Male	Female
Haar	2.3152	2.2919
Db4	2.1076	2.1765
Db6	1.9757	1.9544
Db8	1.7834	1.8182
Db10	1.6393	1.6588

Table 5.3 a), b), c): Performance of Real Time Speech Compression

## 5.7 Varying Truncation % vs. Performance

To see the effects of varying the truncation percentage on the reconstructed signal quality, a male spoken speech signal was decomposed at level 5, using the Db10 wavelet. Figure 5.5 shows the percentage of samples with a certain threshold value in the wavelet decomposition. From this figure it can be seen that most of the coefficients have small magnitudes.

Figures 5.6-5.8 show trends in the quality of a signal as a function of the truncation percentage. Hence a signal of a certain quality can be reconstructed, by choosing the appropriate threshold.

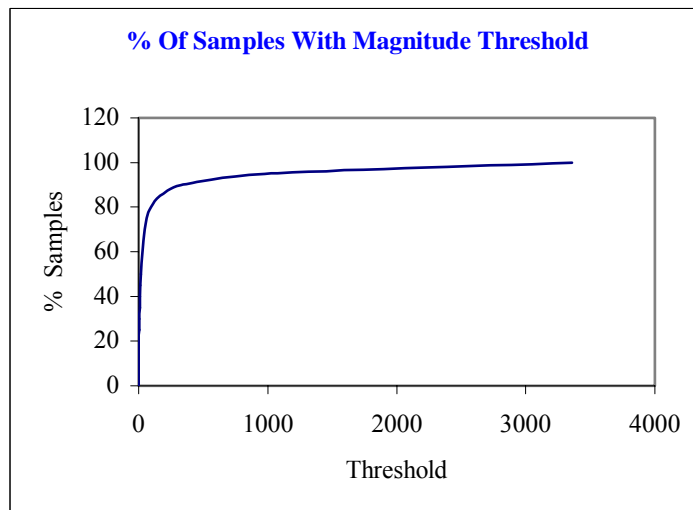


Figure 5.5: Truncation % vs. Threshold

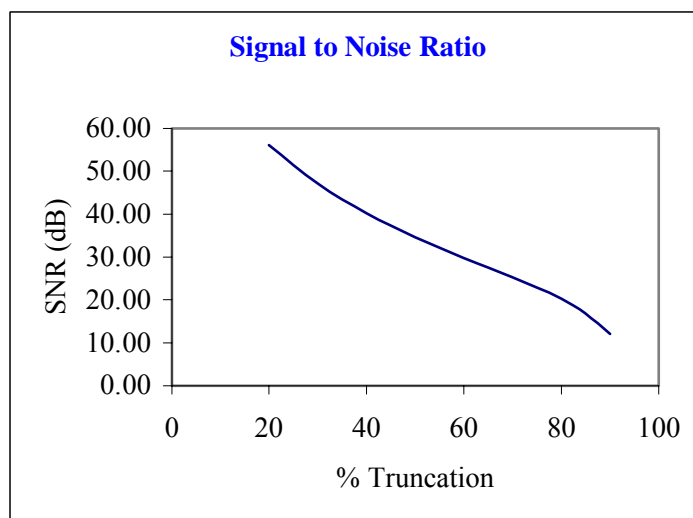


Figure 5.6: SNR vs. % Truncation

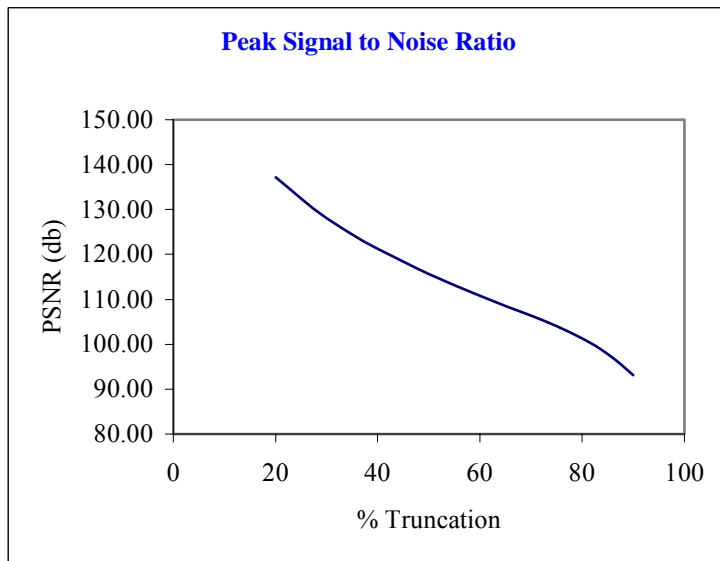


Figure 5.7: PSNR vs. % Truncation

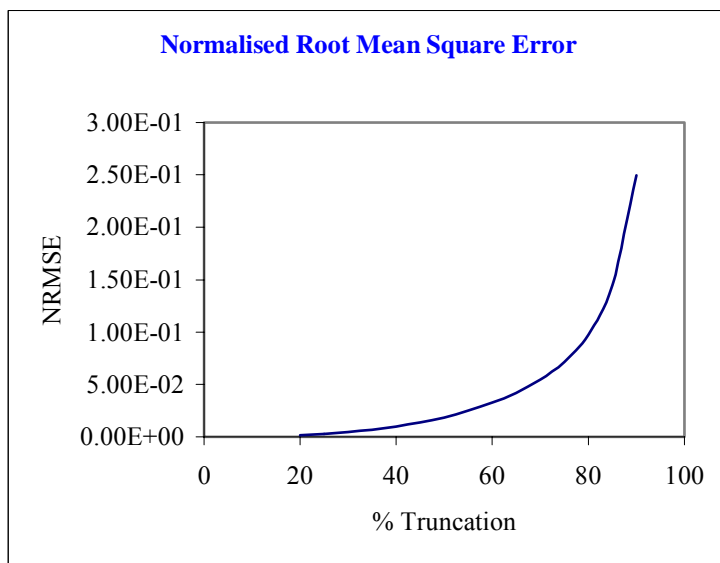


Figure 5.8: NRMSE vs. % Truncation

# Chapter 6

## Discussion

### 6.1 Decomposition Level in Wavelet Transforms

Choosing the right decomposition level in the DWT is important for many reasons. For processing speech signals no advantage is gained in going beyond scale 5 and usually processing at a lower scale leads to a better compression ratio.

Figure 5.1 shows the reconstructed approximation coefficients at different scales for the sample speech signal analysed. This figure shows that the original speech data is still well represented by the level 3 approximation. At higher levels the approximation data isn't as significant and hence does a poor job in approximating the input signal. Therefore if all the approximation data is to be kept, the ideal decomposition for this signal is level 3. Thus in section 5.5 a level 3 decomposition was computed as all approximation data is preserved by the Birge-Massart strategy.

### 6.2 Optimal Wavelet Selection

A suitable criterion used by [1] for selecting optimal wavelets, is the energy retained in the first  $N/2$  coefficients. Based on this criterion alone the Daubechies 10 wavelet preserves perceptual information better than all the other wavelets tested.

The Db10 wavelet also provides the highest SNR, PSNR, and lowest NRMSE, as shown in table 5.2. The Compression Scores shown in table 5.2 c) are primarily related to the percentage of truncation. However its value is also affected by the scarcity of the wavelet domain representation of the signal and also the mother function used. The Db10 wavelet has the highest no of vanishing moments of the wavelets tested and thus

provides the most compact signal representation. For both male and female voices the Db10 wavelet provides the greatest compression as revealed in table 5.2 c).

### 6.3 Compression Score & Size of Frames

To explain the effects of framing on compression score, consider a speech signal that is analysed using the Db10 wavelet at the 5<sup>th</sup> decomposition level. The table below shows the number of transform coefficients for different frame sizes.

<b>Fame Size</b>	<b>160</b>	<b>1024</b>	<b>16195</b>
Coefficients A5	23	50	524
D5	23	50	524
D4	27	81	1030
D3	36	144	2041
D2	54	270	4063
D1	89	521	8107
Total Coefficients	252	1116	16289
<b>Coeff/Frame Size</b>	<b>1.575</b>	<b>1.0898</b>	<b>1.0058</b>

Table 6.1: Frame Size vs. No of Coefficients

The number of coefficients at each level can be worked out using formula 2.11. An important deduction from this table is that as the frame size decreases the ratio of the wavelet coefficients to frame size increases. This means that the ratio of the wavelet representation of the signal to actual signal representation becomes larger for smaller frames.

Due to this compression ratios for real time speech coding using wavelets will be poorer than that achieved for pre-recorded speech signals, using no frames or large frame sizes. Therefore for the same truncation percentage coding pre-recorded speech (for speech synthesis or transmission at a later time), will lead to a better compression ratio and superior reconstruction quality than for real time speech coding. This is clearly shown by comparing the results of tables 5.5 and 5.6.

## 6.4 Compression Score & Decomposition Level

The former problem is worsened for real time coding by increasing the decomposition level in wavelet transforms. This occurs because the number of coefficients required to represent a given signal increases with the level of decomposition. Table 6.2 illustrates this for two different frame sizes.

<b>Frame Size</b>	<b>1024 samples</b>	
Decomposition Level	5	3
No of Coefficients	1116	1079
Coefficients/Frame Size	1.0898	1.0537
<b>Frame Size</b>	<b>160 samples</b>	
Decomposition Level	5	3
No of Coefficients	252	215
Coefficients/Frame Size	1.575	1.3438

Table 6.2: Decomposition Level vs. No of Coefficients

From the table it is obvious that the increase in the no of coefficients at higher decomposition levels has a greater impact on smaller frame sizes. Also higher wavelet decompositions requires more computation time, which should be minimised for real time speech coding.

From equation 2.11, it can be seen that as the size of the input signal  $n$  becomes smaller the  $N$  term in the equation, representing half the length of the wavelet filter coefficients (also equal to the number of vanishing moments) becomes more significant.

Thus using a wavelet with a large no of vanishing moments will start to deteriorate the performance of a real time speech coder after a certain filter length. From table 5.3, the Db6 wavelet gives better performance than all the other wavelets tested. Thus for real time speech coding using wavelets it is important to choose a wavelet function with less vanishing moments and also a reasonable decomposition level.

# Chapter 7

## Future Work

### 7.1 Enhancing Quality

Listening tests conducted on a male spoken sentence “Cats and dogs each hate the other”, using different wavelets revealed that the /s/ sound in the word “dogs” tends to be slightly distorted and if not heard carefully can be mistaken for just the singular term. The /s/ sound is an unvoiced excitation. The transforms for three different frames from this speech signal are analysed in section 5.3.

Figure 5.3 shows that for an unvoiced speech frame the wavelet coefficients are spread across all frequency bands. Therefore this frame will not undergo significant compression when a threshold is applied. If a relatively low value is used for the truncation threshold, the reconstructed signal frame will be severely distorted. Thus wavelets are inefficient at coding unvoiced speech frames.

By detecting unvoiced speech frames and directly encoding them using some form of bit encoding, like entropy coding, no unvoiced data is lost and there is only a marginal increase in the bit rate [1]. A cheap wavelet with few vanishing moments can be used to detect voiced or unvoiced speech frames, using the technique suggested in the Literature Review (Section 3.1.5).

### 7.2 Improving Compression Ratios

Further data compaction is possible by exploiting the redundancy in the encoded transform coefficients. A bit encoding scheme could be used to represent the data more efficiently. A common loss-less coding technique is *Entropy* coding. Two common entropy coding schemes are *Prefix coding* and tree-structured *Huffman coding*.

Both these forms of entropy coding require a prior knowledge of the nature of the source data, such as probability distribution of the source output data [8]. In practice however, probabilistic models are usually not known a priori. Thus a model of the data must be constructed from the data set itself.

An example of a data compaction code that encodes directly from the data stream without constructing an explicit model is the *Ziv-Lempel* code [2]. *Ziv-Lempel* coding is a universal variable-to-fixed-length data compaction code that is practical, has good performance and does not require an externally constructed source model. The use of such a scheme in the last stage of the wavelet transform speech coder (shown in figure 4.1) will enable the transmission of voice at low bit rates.

# Chapter 8

## Conclusion

Speech coding is currently an active topic for research in the areas of Very Large Scale Integrated (VLSI) circuit technologies and Digital Signal Processing (DSP). The Discrete Wavelet Transform performs very well in the compression of recorded speech signals. For real time speech processing however, its performance is not as good. Therefore for real time speech coding it is recommended to use a wavelet with a small number of vanishing moments at level 5 decomposition or less.

The wavelet based compression software designed reaches a signal to noise ratio of 17.45 db at a compression ratio of 3.88 using the Daubechies 10 wavelet. The performance of the wavelet scheme in terms of compression scores and signal quality is comparable with other good techniques such as code excited linear predictive coding (CELP) for speech, with much less computational burden. In addition, using wavelets the compression ratio can be easily varied, while most other compression techniques have fixed compression ratios.

# References

1. J.I. Agbinya, "Discrete Wavelet Transform Techniques in Speech Processing," *IEEE Tencon Digital Signal Processing Applications Proceedings*, IEEE, New York, NY, 1996, pp 514-519.
2. R.E. Blahut, *Digital Transmission of Information*, Addison-Wesley, New York, 1990.
3. A. Chen, N. Shehad, A. Virani and E. Welsh, "Discrete Wavelet Transform for Audio Compression," <http://is.rice.edu/~welsh/elec431/wavelets.html>, (current July. 16, 2001).
4. N.R. Chong, I.S. Burnett, J.F. Chicharo and M.M. Thomson, "Use of the Pitch Synchronous Wavelet Transform As a New Decomposition Method for WI," *Proc. Int'l Conf. Acoustics, Speech and Signal Processing*, Vol 1, IEEE, New York, NY, 1998, pp. 513-516.
5. E.B. Fgee, W.J. Phillips, W. Robertson, "Comparing Audio Compression using Wavelets with other Audio Compression Schemes," *IEEE Canadian Conference on Electrical and Computer Engineering*, IEEE, Edmonton, Canada, 1999, pp. 698-701.
6. A. Gersho, "Speech Coding," *Digital Speech Processing*, A.N. Ince, ed., Kluwer Academic Publishers, Boston, 1992, pp. 73-100.
7. A. Graps, "An Introduction to Wavelets," *IEEE Computational Sciences and Engineering*, <http://www.amara.com/IEEEwave/IEEEwavelet.html> (current Mar. 15, 2001).
8. S.Haykin, *Communication Systems*, John Wiley & Sons, New York, 2001.
9. J.N. Holmes, *Speech Synthesis and Recognition*, Chapman & Hall, London, 1988.

10. R. Kastantin, D. Stefanoiu, G. Feng, N. Martin and M. Mrayati, "Optimal Wavelets for High Quality Speech Coding," *Proc. Of EUSIPCO-94*, European Association for Signal Processing, Edinburgh, Scotland, 1994, pp. 399-402.
11. W. Kinsner and A. Langi, "Speech and Image Signal Compression with Wavelets," *IEEE Wescanex Conference Proceedings*, IEEE, New York, NY, 1993, pp. 368-375.
12. B. Lin, B. Nguyen and E.T. Olsen, "Orthogonal Wavelets and Signal Processing," *Signal Processing Methods for Audio, Images and Telecommunications*, P.M.Clarkson and H.Stark, ed., Academic Press, London, 1995, pp. 1-70.
13. L.R. Litwin, "Speech Coding with Wavelets," *IEEE Potentials*, Vol.17, No.2, May 1998, pp. 38-41.
14. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, Calif., 1998.
15. M. Misiti, Y. Misiti, G. Oppenheim and J. Poggi, *Matlab Wavelet Tool Box*, The Math Works Inc., 2000.
16. Y. Nievergelt, *Wavelets made easy*, Birkhäuser, Boston, 1999.
17. J. Ooi and V. Viswanathan, "Applications of Wavelets to Speech Processing," *Modern Methods of Speech Processing*, R.P. Ramachandran and R. Mammone, ed., Kluwer Academic Publishers, Boston, 1995, pp. 449-464.
18. A.H. Tewfik and M.Ali, "Enhanced Wavelet Based Audio Coder," *Conference Record of the 27<sup>th</sup> Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Nov. 1993, pp. 896-900.

19. M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall PTR, Englewood Cliffs, N.J, 1995.
  
20. V. Viswanathan, W. Anderson, J. Rowlands, M. Ali and A. Tewfik, "Real-Time Implementation of a Wavelet-Based Audio Coder on the T1 TMS320C31 DSP Chip," *5<sup>th</sup> International Conference on Signal Processing Applications & Technology (ICSPAT)*, Dallas, TX, Oct. 1994.

# Appendix – Code Listings

This appendix contains some of the important Matlab functions and script files developed during the course of this thesis project. To properly implement a real time speech coder, many of the underlying functions need to be compiled to C++ code for faster execution.

## Openfile.m

```
function sdata = openfile(fName);
% openfile : function to read a speech file with a .od extension
% call syntax: sdata = openfile(fName);
% -----
% Read sound file data into a column vector
sdata = dlmread(fName);
```

## Play.m

```
function play(M);
% PLAYFILE: Plays a sound file which is stored as a vector
% call syntax: playfile(M);
% -----
% Play sound file
soundsc(M, 8000, 8);
```

## Main.m

```
% Speech Compression Simulation Program

% User Inputs
fileName = 'c:\program files\matlab\work\s180.od';
wavelet = 'db10';

% Compress speech
[tC, tL, PZEROS, PNORMEN] = compress(fileName, wavelet);

% Decompress speech
rS = decompress(tC,tL, wavelet);
```

```
% Performance calculations
[SNR, PSNR, NRMSE] = pefcal(fileName, rS);
```

### Compress.m

```
function [tC, tL, PZEROS, PNORMEN] = compress(fileName, wavelet);
% Compress : compresses speech signals wavelet coefficients
% Inputs: speech signal file name, wavelet
% Outputs: compressed coefficients, length vector, compression score
% and retained energy
% Call syntax: [tC, tL, PZEROS, PNORMEN] = compress(fileName, wavelet);
% -----
```

```
% Initialise other variables
N = 5; % level of decomposition
ALPHA = 1.5; % compression parameter
SORH = 'h'; % hard thresholding
```

```
% Read speech file
sdata = openfile(fileName);
% Compute the DWT to level N
[C,L] = wavedec(sdata,N,wavelet);
% Calculate level dependent thresholds
[THR,NKEEP] = lvlThr(C,L,ALPHA);
% Compress signal using hard thresholding
%[XC,CXC,LXC,PERF0,PERFL2] = Trunc('lvd',C,L,wavelet,N,THR,SORH);
% Encode coefficients
cC = encode(CXC);
% Transmitted coefficients;
tC = cC;
% Transmitted coefficients vector length
tL = L;
% Percentage of zeros
PZEROS = PERF0;
% Retained energy
PNORMEN = PERFL2;
% Compression ratio with encoding
CompRatio = length(sdata)/length(tC)
```

### Decompress.m

```
function rSignal = decompress(tC,tL, wavelet);
% Decompress : uncompress DWT coefficients and reconstructs signal
% Inputs: encoded wavelet coefficients, coeff vector length
% Output: reconstructed signal
% Call syntax: rSignal = decompress(tC,tL, wavelet);
% -----
```

```
% Decode coefficients
rC = decode(tC);
% Reconstruct signal from coefficients
rSignal=waverec(tC,tL,wavelet);
```

Encode.m

```
function cC = Encode(C);
% Encode: function encodes consecutive zero valued coefficients
% Call syntax: cC = Encode(C);
% -----

% Initialise variables
zeroseq = 'false'; % True if previous array entry was a zero
zerocount = 0; % Count of no of zeros in sequence
j= 1; % Start index value for compressed coefficients
compC = [ ]; % compressed coefficients vector

% Start iterating thru array

for m=1:length(C)
    if (C(m) == 0) & (zeroseq == 'false') % First zero
        compC = [compC C(m)];
        j = j+1;
        zeroseq = 'true';
        zerocount = 1;
        % Reached end of array and last value is zero
        if m == length(C)
            compC = [compC zerocount];
        end
    elseif (C(m) == 0) & (zeroseq == 'true') % Sequence of zeros
        zerocount = zerocount + 1;
        % Reached end of array and last value is zero
        if m == length(C)
            compC = [compC zerocount];
        end
    elseif (C(m) ~= 0) & (zeroseq == 'true') % End of zeros
        compC = [compC zerocount C(m)];
        j = j+2;
        zeroseq = 'false';
        zerocount = 0;
    else % Non-zero entry
        compC = [compC C(m)];
        j = j+1;
    end
end
cC = compC;
```

Decode.m

```
function rC = Decode(cC);
% Decode: function to decode consecutive zero valued coefficients
% Call syntax: rC = Decode(cC);
% -----

% Initialise variables
dcompC = [ ]; % Empty reconstructed coefficients array
i = 1; % Initial index of loop

% Start iterating thru array
while i <=length(cC)
    if cC(i) ~= 0 % Non-zero entry
        dcompC = [dcompC cC(i)];
        i = i + 1;
    else % Zero entry
        count = cC(i+1);
        for m=1:count % Add zeros
            dcompC = [dcompC 0];
        end
        i = i + 2;
    end
end

rC = dcompC;
```

Pefcal.m

```
function [SNR, PSNR, NRMSE] = pefcal(fileName, rS);
% Pefcal: Performance Calculations function file
% Calculates Signal to Noise Ratio, Peak Signal to Noise Ratio
% and Normalized Root Mean Square Error

% Get original speech signal
origdata = openfile(fileName);
% Resize reconstructed signal for the mathematics to work
rS = rS(1:length(origdata));
% Signal to Noise Ratio
sqdata = origdata.^2;      % Square of original speech signal
sqrS = rS.^2;             % Square of reconstructed signal
msqdata = mean(sqdata);   % Mean square of speech signal
sqdiff = (sqdata-sqrS);   % Square difference
msqdiff = mean(sqdiff);   % Mean square difference
SNR = 10*log10(msqdata/msqdiff); % Signal to noise ratio

% Peak Signal to Noise Ratio
N = length(rS);          % Length of reconstructed signal
X = max(abs(sqdata));    % Maximum absolute square of orig signal
diff = origdata - rS;    % Difference signal
enddiff = (norm(diff))^2; % Energy of the difference between the
                        % original and reconstructed signal
PSNR = 10*log10((N*(X^2))/enddiff); % Peak Signal to noise ratio

% Normalised Root Mean Square Error
diffsq = diff.^2;        % Difference squared
mdiffsq = mean(diffsq);  % Mean of difference squared
mdata = mean(origdata);  % Mean of original speech signal
scaledsqS = (origdata - mdata).^2; % Squared scaled data
mscaledsqS = mean(scaledsqS); % Mean of squared scaled data
NRMSE = sqrt(mdiffsq/mscaledsqS); % Normalized Root Mean Square Error
```

## Comp.m

```
function [tC, tL, PZEROS, PNORMEN, cScore, nFrames] = comp(fileName, wavelet,
N, frameSize)
% Comp: function simulates real time compression of speech signals
% Inputs: speech signal file name, wavelet and frame size
% If frame size is 0 no frames are used
% Outputs: compressed coefficients and compression ratio
% Call Syntax: [tC, tL, PZEROS, PNORMEN, cScore, nFrames] = comp(fileName,
wavelet, N, frameSize)
```

```
% Calculate no of frames
fileSize = FileSize(fileName);
if frameSize == 0
    frameSize = fileSize;
end
numFrames = ceil(fileSize/frameSize);
```

```
% Initialise other variables
%tC = [ ]; % transmitted coefficients vector
tXC = [ ]; % uncompressed coefficients vector
%lenOrigC = 0; % length of original coefficients
PERF0V = [ ]; % vector of % truncation for each frame
PERFL2V = [ ]; % vector of % retained energy for each frame
```

```
for i=1:numFrames
    % Read a frame from the speech file
    sdata = FrameSelect(i,frameSize,fileName, fileSize);
    % Compute the DWT to level N
    [C,L] = wavedec(sdata,N,wavelet);
    % Calculate default thresholds
    [THR, SORH, KEEPAPP] = gblThr('cmp','wv',sdata);
    SORH = 'h';
    KEEPAPP = 0; % Can threshold approximation coefficients also
    % Compress signal using hard thresholding
```

```
[XC,CXC,LXC,PERF0,PERFL2]=Trunc('gbl',C,L,wavelet,N,THR,SORH,KEEPAPP);
% Encode coefficients
cC = encode(CXC);
% Transmitted coefficients
tXC = [tXC cC];
% Truncation % Vector
PERF0V = [PERF0V PERF0];
% Retained Energy Vector
PERFL2V = [PERFL2V PERFL2];
end
```

```
% Return Values
```

```

tC = tXC;
tL = tXC;
PZEROS = mean(PERF0V);
PNORMEN = mean(PERFL2V);
cScore = fileSize/length(tC);
nFrames = numFrames;

```

#### Decomp.m

```

function rSignal = decomp(tC,tL,wavelet,numFrames,frameSize);
% Decomp: function simulates real time decoding of signals
% Inputs: encoded wavelet coefficients, coeff vector length
% Call syntax: rSignal = decompress(tC,tL,numFrames,frameSize);
% Outputs: reconstructed signal
% Call Syntax: rSignal = decomp(tC,tL,wavelet,numFrames,frameSize);

% Initialise other variables
rS = [ ]; % reconstructed signal
frameSize = sum(tL)-frameSize; % frame size of DWT coefficients
% Decode coefficients
rC = decode(tC);
for i=1:numFrames
    % Range of frame
    R1 = (i-1)*frameSize + 1;
    R2 = i*frameSize;
    % Read coefficients in frame
    fC = rC(R1:R2);
    % Reconstruct frame signal
    X = waverec(fC,tL,wavelet);
    % Total reconstructed signal
    rS=[rS; X];
end
% Return output
rSignal = rS;

```

#### Filesize.m

```

function fSize = FileSize(fName);
% FileSize: counts no of samples in a speech file
% Call syntax: fSize = FileSize(fName);
% -----
data = OpenFile(fName);
fSize=length(data);

```

## Frameselect.m

```
function v = FrameSelect(fNum,fSize,fileName,fileSize);
% FrameSelect : reads a frame of data from a speech file into a column vector
% call syntax: v = FrameSelect(fNum,fSize,fileName);
% -----

% Read the corresponding frame from the sound file into a column vector
% range = [R1 C1 R2 C2] C1 = C2 = 0 since only one column
% R1 = First Value, R2 = Last Value
R1 = fSize*(fNum-1);
R2 = (fSize*fNum - 1);
R3 = R2;
% Adjust range value for last frame
if R2 >= fileSize
    R2 = fileSize - 1;
end
range = [R1 0 R2 0];
v = dlmread(fileName,"",range);
% If data for last frame is smaller than frame size
% Zero pad the frame
if R3~=R2
    N = (R3-R2);
    for i= 1:N
        v = [v;0];
    end
end
end
```

## Optimal.m

```
% Optimal Wavelet For Speech Compression
% This script file determines the percentage of Speech Frame Energy
% Concentrated by wavelets in the first N/2 Coefficients
% -----
% Inputs: speech signal file name, wavelet and frame size
% Outputs: compressed coefficients and compression ratio

% User Inputs
fileName = 'c:\program files\matlab\work\s180.od';
wavelet = 'db10';
frameSize = 160;

% Calculate no of frames
fileSize = FileSize(fileName);
if frameSize == 0
    frameSize = fileSize;
end
numFrames = ceil(fileSize/frameSize);

% Vector to Store Retained Energy of Each Frame
PREV = [];
% Step thru each frame and calculate Retained Energy
%for i=1:numFrames
    % Read a frame from the speech file
    sdata = FrameSelect(8,frameSize,fileName, fileSize);
    % Compute the DWT to level 5
    [C,L] = wavedec(sdata,5,wavelet);
    % Calculate Energy Retained in first N/2 Coefficients
    xC = C(1:(length(C)/2));
    RE = 100*(norm(xC))^2/(norm(C))^2;
    PREV = [PREV ; RE];
%end
PREV
```

## Voiced.m

```
% Vocied, Unvoiced and Mixed Frames
% This script file plots frames
% -----
% Inputs: speech signal file name, wavelet and frame size
% Outputs: compressed coefficients and compression ratio

% User Inputs
fileName = 'c:\program files\matlab\work\s180.od';
wavelet = 'db10';
frameSize = 1024;

% Calculate no of frames
fileSize = FileSize(fileName);
if frameSize == 0
    frameSize = fileSize;
end
numFrames = ceil(fileSize/frameSize);

% Read frame i from the speech file
i = 9;
sdata = FrameSelect(i,frameSize,fileName, fileSize);
% Compute the DWT to level 5
[C,L] = wavedec(sdata,5,wavelet);
% Calculate Energy Retained in first N/2 Coefficients
xC = C(1:(length(C)/2));
RE = 100*(norm(xC))^2/(norm(C))^2;
% Plot frame and wavelet transform coefficients
subplot(2,1,1); plot(sdata,'r'); title('Mixed Speech Segment');
subplot(2,1,2); plot(C); title('DWT Coefficients Using Db10 Wavelet');
```