Source-Channel Coding for CELP Speech Coders

J.A. Asenstorfer B.Ma.Sc.(Hons),B.Sc.(Hons),B.E.,M.E.

University of Adelaide
Department of Electrical and Electronic Engineering.

Thesis submitted for the Degree of Doctor of Philosophy.

September 26, 1994

Awarded 1995

Contents

List of Figures

List of Tables

Abstract

Low rate speech coders have recently undergone extensive development. However little effort has gone into developing low rate coders for noisy channels.

This thesis is concerned with methods for protecting speech coding parameters transmitted over noisy channels. A linear prediction (LP) coder is employed to remove the short term correlations of speech. Protection of two sets of parameters are investigated: the short term spectral (LP) information and the excitation sequence. Errors in the LP coefficients cause particularly annoying distortions in the reconstructed speech.

Both techniques proposed for protecting spectral information employ the line spectral pair (LSP) representation of LP coefficients. The first scheme relies on the monotonicity of the LSP representation to detect errors, and uses a vector codebook for substitution of corrupted coefficients. The second scheme uses a multi-stage vector quantizer at the transmit end. Error protection is employed on the most significant bits of the codebook index and non-redundant pseudo-gray coding on the least significant bits.

To mask errors in the excitation sequence, trellis source coding is examined as a possible error robust technique. This form of excitation for modern LP analysis by synthesis coders is not well known. Trellis encoding is investigated specifically for rates less than 1 bit per sample (bps). An efficient encoding is achieved using the M,L algorithm for trellises of 64 to 128 states. Subjective tests reveal that for trellis excitation at less than 1 bps, regular pulse excitation is not perceptibly different from a purely stochastic excitation. Significant perceptual improvement is found on training the codebook. An efficient analysis by synthesis training method for regular pulse trellis excitation codebooks is presented. The segmental signal to noise ratio for the proposed coder is 5.3dB for a transmission bit rate of 5.5kb per second, which is superior than the original multipulse coders.

Declaration

J. Asenstorfer

Acknowledgements

I would like to acknowledge the guidance and suggestions made by Dr. Bill Cowley, Dr. Bruce Davis and Professor Mike Miller during the course of the thesis. Their helpful comments on thesis preparation are also gratefully acknowledged.
I would also like to thank my colleagues at the University of Adelaide for their support and friendship over the past years.

Chapter 1

Speech Coders for Noisy Channels

1.1 Introduction

The aim of this thesis is to investigate error correction and error masking procedures suitable for speech coders operating over additive white Gaussian noise and burst noise channels. Many modern mobile communications systems are subject to channels that vary greatly in signal strength and interference levels. Much interest has recently been shown in mobile satellite communications which allows users to communicate from any position under the satellite's footprint into the public switched telephone network. It is necessary to operate within narrow bandwidths in order to maximise the usage of the satellite transponder and make the scheme economically viable. It is this class of channels that is of primary interest in this thesis. The narrow bandwidths employed by modern speech coders for these kinds of channels make them very susceptible to deep fades. At the receiver end a deep fade is associated with a burst of errors. Although random errors are covered in the investigation the burst noise channel is of major importance as it is characteristic of the mobile communications channel.

It has become quite evident that due to the natural redundancy of speech it is far more efficient to base error protection for bursty channels on reconstruction techniques and methods of selected bit error protection, rather than brute force classical error correction coding. These issues will be discussed at length within the main body of the thesis.

As a vehicle for study this thesis examines the robustness of the class of Code Excited Linear Predictive (CELP) coders [1] [2] [3] [4] [37] [5] [6] [7] [8] [9] [10] [11] [103], Multipulse [12] [13] [14] [15] and Regular Pulse Excited coders [16] [17] [18] with respect to errors suffered during transmission of the coder parameters. A generic CELP coder was chosen as the main vehicle for the investigation as it is capable of delivering near toll quality speech at the rates of the order of 7 kilobit per second. CELP coders, discussed in more detail in Chapter 2, comprise a class of well understood algorithms. Key features include a linear predictive all pole recursive prediction filter [19] and stochastic noise sources used to excite the predictive filter. A United States Federal Standard 1016 [3] [1] has been set for CELP coding. The algorithm is also in the public domain which allows the comparison of developed coders with the Standard. The Multipulse coder developed by Araseki [13] and the Regular Pulse Excited coder developed by Kroon [16] are also used as reference models for the thesis.

A simplified version of the CELP coder without the long term predictor, similar to multipulse coders, was studied to determine the suitability of trellis encoding of the excitation sequence for the predictive filter at low bit rates. It was found that an implementation based on regular pulse excitation proved to be the best option in terms of computational load.

A considerable amount of literature exists on CELP, Multipulse and Regular Pulse Excited coders and their performance over good channels, which make them good vehicles to study in the context of designs to counter channel noise. Only a limited amount of work, see for example ([20] [8] [37] [21]), has been done in the area of making these coders robust to noise. Some of the results of these efforts, are discussed in the overview below.

1.2 Thesis Goals and Historical Perspective

Most of the effort in CELP coding has been in the area of extracting as much performance as possible from coders on a clear channel at a given bit rate.

The traditional view of making digital data robust to noisy channels is to compress the source code to as low a rate as possible and then use a forward error correcting code to obtain robustness. This approach was soon shown to be wasteful as not all parameters are as sensitive as others to noise. An important paper by Cox et al. [20] investigated the sensitivity of the various parameters to channel noise. The most sensitive parameters to noise are the linear prediction coefficients (LPC) parameters which are normally transformed to line spectral pairs (LSP's) before quantisation [31]. These parameters give the short term spectral information; errors give annoying whistling and banging sounds. The next most sensitive parameter to noise is the stochastic codebook gain. This is followed by the adaptive codebook index, the gain and finally the stochastic codebook index. Cox went on to discuss some correction strategies based on the redundancy of the speech signal rather than specifically adding coding redundancy. The techniques discussed include substitution of previously correctly received frames, and replacing LSP parameters found to be in error. Cox also considered Gray coding of quantiser indices to minimise the effect of single bit errors.

The main instrument for burst noise error correction/masking was found by noting that speech is highly redundant and if a frame[1], or subframe, was found to be corrupted beyond correction, various substitutions of previous frames' parameters and/or excitation parameters are possible. Error correction/masking is made possible through natural redundancy within natural speech and redundancy specifically added.

The ultimate goal of the thesis was to develop a speech coder that operated at 5 kilobits per second or below that had excellent noise immunity. The particular aspects that were addressed in order to achieve this goal were:

- the short term spectral information was to be encoded in such a way that either the errors could easily be masked at the receiver, or that it was robustly encoded at the transmitter giving direct immunity to channel noise;
- to use a fractional sample pitch predictor to enhance encoded speech quality and use pitch tracking algorithms to eliminate gross pitch errors caused by transmission bit errors;
- to use a trellis encoding scheme for the excitation sequence due to its alleged robustness to bit errors;
- and post processing to improve perceived reception quality.

Not all of these goals were met, in particular long term prediction using a pitch predictor was not incorporated into the final coder design due to difficulties with the trellis excitation scheme. A description of the contents of the thesis and the historical context in which the investigations were undertaken is given below.

With a view to the final speech coder design an interesting and robust pitch estimation

[1]A frame of speech is typically 20 to 30 milliseconds of speech which is commonly sampled at 8 kHz for coding applications.

algorithm was designed in chapter 3. Beside the pitch[2] estimation aspect, pitch tracking could be useful for detecting gross pitch errors, assuming that a pitch predictor type speech coder was used.

Many CELP coders use a pitch predictor but it was noted that a non-integer pitch predictor [2] [22], gave superior results. An aim of the thesis was to use the concepts of dynamical systems theory [23] [24] to extract pitch information. Tishby [25] explored the validity of describing the voice generation process as a nonlinear dynamical system and applying modern analytical concepts in describing the speech waveform. It was hoped that a dynamical systems approach would give good pitch information. Dynamical systems generate an orbit in an arbitrary space, called phase space, as the system evolves. The orbits generated by the speech waveform were studied to obtain pitch estimates.

Poincaré investigated dynamical systems by observing how a phase space track pierced a plane (section in higher dimensions) from a given direction. Grebogi, Ott & York [66] illustrate the idea. Ott [73] discusses the idea with a little more detail. The idea of a Poincaré section is useful in this application for finding points close to each other in different cycles of the orbit and extracting pitch information at the same time. This technique worked providing that the orbit did not vary too much within the window of observation.

A correlational approach to match the shapes of the orbits in multiple dimensions was chosen as the best approach to obtain the pitch. The developed fractional sample pitch estimator was presented in part to the Speech Science and Technology (SST) conference in Brisbane December 1992 [50]. Feed-back from the conference allowed refinement of the algorithm to make it simpler and more robust. A useful voiced/unvoiced indicator was also developed.

One of the best pitch estimators developed in recent years is that of Medan [76]. When the pitch estimator in the thesis is compared to Medan's many similarities become apparent although the one in the thesis was developed from another view point. One obvious advantage of using multiple dimensions was found in the improvement of the Absolute Magnitude Difference Function [26] (AMDF) when three dimensions were used for estimating the integer sample pitch.

An estimate of the pitch was also found to be useful for the post processing of speech.

[2]Pitch in this context was taken to mean the distance between glottal pulses, or the point where the speech waveform appeared to almost repeat itself.

Kroon [4] used spectral shaping for the post processing of speech and Transcoso [27] used a frequency domain approach to improve the perceived quality of CELP type coders. Other authors have used harmonic processing to remove unwanted noise from noisy speech [28]. A new method of pitch synchronous post processing of noisy speech was investigated at the end of Chapter 6. Implementation of the post processing required pitch information, preferably transmitted from the source end. The post processing improved the perceived quality of the reconstructed speech.

A primary interest of the thesis was the burst noise case common on mobile communications channels. Work by Perkis [21] and the author with colleagues (Rowe et. al. [51]) have studied the effects of burst noise and the masking of errors when entire frames were rendered unusable. A key factor to good performance in a burst noise environment was not only error correction for random errors but also more importantly error detection to determine when an encoded frame is deemed useless.

The parameters transmitted over the channel may be considered to be in different classes depending on their sensitivity to errors. As the short term spectral information is very sensitive to errors, it was dealt with separately using two different approaches to improve performance. The short term spectral information must be well protected to minimize the perceived annoying tonal distortions.

Two chapters of the thesis, Chapters 4 and 5, each present a different approach, to increasing the spectral parameters robustness to noise. Chapter 4 investigated a scheme of scalar quantising the line spectral pairs (LSP's) [30] [31] [32] at the transmitting terminal and then using a vector quantiser [33] at the receiver terminal to match the received scalar quantised LSP's with entries in the LSP vector codebook. If a component of the received LSP vector was deemed to be in error then it was corrected using components from the sample vector stored in the codebook. This method made use of the natural redundancy within speech by noting that a limited number of LSP vector shapes can describe most voice vectors with small distortion and that significant correlations exist between LSP vector components. The application of using a vector codebook to mask errors in the transmission of vectors is novel. Traditionally if an error was found in the LSP's they were sorted, or the previous frame's LSP's substituted [20] [21] [51]. The scheme described in the thesis was enhanced, so that the vector codebook performing the error masking was trained on-line and thus avoided the need for a large static codebook at the receiver. The superiority of the proposed scheme was clearly demonstrated.

In Chapter 5 the vector quantisation process was moved to the transmitting end. This application of vector quantisation is the more traditional one. Several attempts have been made at designing vector quantisation codebooks for the short term spectral information:Ozawa [22], Kroon [34], Grass [35]. However rather than just using vector quantisation to reduce the bit rate, and then adding an error correcting code to obtain integrity of the transmitted data, a more detailed study was undertaken. By using the inherent redundancy within speech and error masking properties of the auditory system a method of unequal error protection was possible, where the most significant bits of the codebook index was forward error protected and the least significant bits used to index vectors in close proximity to each other. Use of vector quantisation to reduce the bit rate for speech parameters for the noisy channel scenario has not been extensively studied elsewhere.

Various authors have attempted to design joint source channel coders for vector quantisation. The most notable work in the area has been done by Ayanoglu and Gray [38] and Farvardin [39] [40]. Each of the joint source channel coders perform well under the conditions they were designed for, that is, at a particular channel error rate. Secker [41] applied Ayanoglu's design algorithm to the encoding of LSP parameters. Although for a given bit rate significant performance improvement was found for the codebooks designed for a particular bit error rate, when the channel was error free a performance degradation was observed compared to source coders designed for the error free channel. Farvardin also observed similar degradation on the error free channel in his articles.

Zeger and Gersho [42] investigated generalised Gray coding to add non-redundant error minimization. Conceptually, indices to codebooks or other quantisers should be organised such that if an error occurs during transmission the reconstructed values (despite an erroneous index) will result in a vector/value as close as possible to the original vector/value. This idea is developed further in the thesis where the most significant bits were forward error protected and the less significant bits Gray coded. Coupled with these concepts an implementation that could run in real time was sought. Due the very large codebooks required to encode the short term spectral information (indices of ≈24 bits) a structure was placed on the codebook. The structures investigated were the binary decision tree, and the split and multistage vector quantisers; these structured vector quantisers are covered in some detail in the book by Gersho and Gray [33], which covers most aspects of vector quantisation.

The encoded stochastic excitation sequence has low sensitivity to errors which cause wideband noise. Due to the lower noise sensitivity the stochastic coding was considered separately, as a careful coding procedure without any further protection was conjectured to be adequate for the encoded excitation sequence. This did indeed prove the case.

Salami [8] [37] considered embedding forward error correction coding into the speech coder to protect only certain bits to achieve robustness. This idea of unequal error protection also makes use of the fact that various parameters are more sensitive to errors than others.

Chapter 6 deals with a trellis excitation mechanism for CELP coders. The main rationale for using a trellis coding mechanism is to limit the range of the effect of bit errors within the excitation sequence. Salami [43] has commented on the usefulness of making the excitation sequence (the stochastic codebook excitation) robust to channel errors. The concept of using trellis excitation for predictive waveform coding was briefly mentioned by Stewart [44] who called it *hybrid tree coding.* The new model described in the thesis advanced the concept considerably. The reason for the localization of error events is briefly described as follows. Trellis coding generates a bit stream which is used to generate a sequence of states within a state machine[3] and subsequently produces a series of excitation vectors at the receiver. The memory of the state machine together with the memory of the synthesis filter at the receiver determines the duration of any corruptive effect of channel errors in the excitation bit stream. For the moderate sized trellises used, the state memories were small. As the new model has small localised effects for bit errors, it has a large advantage over the standard CELP model which uses large vectors and a single index. In the standard CELP model an error event in the index, causes an incorrect vector to be chosen resulting in a complete vector[4] of incorrect excitation, in addition to the memory hangover effect into the following excitation vector. For modest trellis sizes the trellis coder's error propagation was only marginally longer than the memory hangover effect.

The important issue of codebook/trellis training is also covered in some detail in Chapter 6. Due to the nature of the regular pulse excitation chosen in the implementation, standard optimization techniques [44] [45] were not suitable. This led to stochastic (probabilistic) optimization techniques such as simulated annealing which has been investigated by various authors [46] [47] [48]. Simulated annealing was investigated for

[3]A trellis with 128 states describes the states and transitions of a finite state machine with a 7 bit memory.

[4]Typically the length of excitation vectors in CELP are of the order of 40 samples.

training the codebooks developed in the thesis, but the training time was far too great and a new random search algorithm was developed. The algorithm as developed here will not work well in low dimensions in unmodified form, however it had distinct rate of convergence advantages in higher dimensions. It was found that the codebook/trellis entries had a considerable effect on the subjective quality of the reproduced speech. The implementation details of a CELP coder that used the (M,L) algorithm [49] for trellis encoding of the source is covered in some detail in the latter part of the chapter.

Chapter 7 models the error processes involved in receiving erroneous bits for the encoded excitation stream and discusses the parameters involved in selecting a suitable trellis. Limitations of the trellis encoded excitation were also determined, in particular, the regular pulse excitation model. It was found that the trellis encoded excitation, as implemented, had a lower bit rate limit of 1/4 bit per speech sample.

Speech coder design for noisy channels is then discussed in the final chapter with recommendations and indication of areas where future work would be of interest.

Two publications [51] [50] have come out of this work to date, where the co-authored work [51] addressed packet substitution schemes suitable for CELP coders and the sole authored [50] covered preliminary results of the super resolution pitch estimator.

Chapter 2

Linear Predictive Coders Background

2.1 Introduction

When coding the waveform of speech two basic strategies are available: frequency domain representation and time domain coding. The experimental work in this thesis is based on time domain coding. In particular the work in the thesis is based on the highly successful Linear Prediction Coding which is described in some detail by Makhoul in a tutorial paper [52].

The purpose of this chapter is to give the reader background theory and establish the nomenclature that is used in the following chapters.

Time domain waveform coding is based on a series of equally spaced, time samples of speech. Linear prediction attempts to estimate the next speech sample based on a weighted sum of previous speech samples. The prediction uses an all pole model of the speech process and is designed to be optimal in the least squared error sense. By using an all pole model, computationally efficient solutions for the coefficients can be utilized; solutions for the model's coefficients, given the speech time series, are described below.

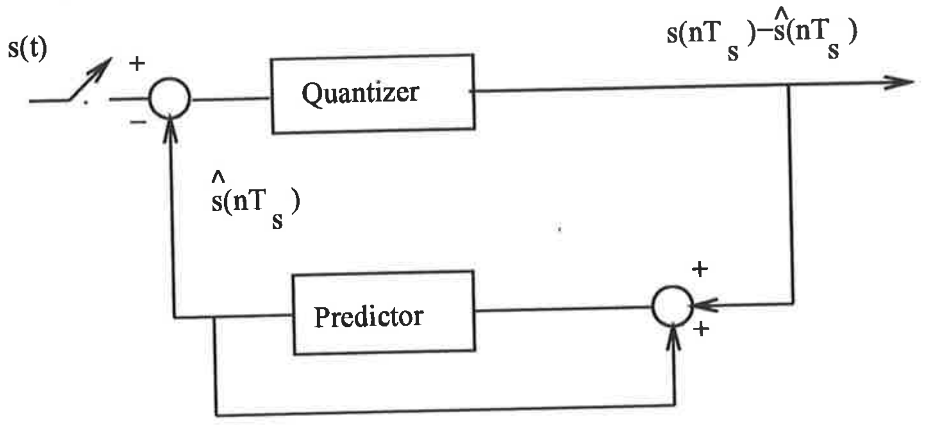


Figure 2.1: Differential Pulse Code Modulator.

2.2 Linear Predictive Models

The simplest form of prediction is that used by Differential Pulse Code Modulation (DPCM). This coder uses a predictor which is fixed and employs n previous speech samples to predict the next. The difference between the in-coming speech sample and the predicted sample is quantised and transmitted. Figure 2.1 illustrates a block diagram of the system.

Consider a predictor that operates on regularly sampled speech, with sampling period T_s. Typically these simple predictors use one, two or three previous samples. For illustrative purposes, a model using the two most recent speech samples will be developed. The predicted speech sample $\hat{s}(nT_s)$ at time nT_s which tries to estimate the true speech sample $s(nT_s)$ is given by,

$$\hat{s}(nT_s) = As((n-1)T_s) + Bs((n-2)T_s) \tag{2.1}$$

where A and B are the predictor coefficients to be determined. The error between the predicted value and the true sample value may be written as;

$$\begin{aligned} e(nT_s) &= s(nT_s) - \hat{s}(nT_s) \\ &= s(nT_s) - As((n-1)T_s) - Bs((n-2)T_s) \qquad (2.2) \\ & \qquad (2.3) \end{aligned}$$

The mean square error (mse) is to be minimized. Mean square error can be written in terms of the mathematical expectation $E[.]$ as,

$$mse = E[e^2(nT_s)] \tag{2.4}$$

By finding where $\partial mse/\partial A$ and $\partial mse/\partial B$ are equal to 0, the optimal values for A and B can be found. The differentiation of the mean square error with respect to A and B is equivalent to using the principal that for minimum prediction error the expectation of the data used for prediction and the error must be orthogonal, which allows us to write;

$$\begin{aligned} E[(s(nT_s) - As((n-1)T_s) - Bs((n-2)T_s))(s((n-1)T_s))] &= 0 \\ E[(s(nT_s) - As((n-1)T_s) - Bs((n-2)T_s))(s((n-2)T_s))] &= 0 \end{aligned} \tag{2.5}$$

The expectation $E[s(nT_s)s((n-1)T_s)]$ can be estimated using the auto-correlation function. Ideally the auto-correlation function should be evaluated over an extremely long segment of speech as it should represent the correlation value for all speech for the true expectation. That is,

$$E[s(nT_s)s((n-1)T_s)] \approx R(T_s) = \sum_{n=1}^{N} s(nT_s)s((n-1)T_s) \tag{2.6}$$

where N takes on a very large number in order to represent speech in general. In principle the two coefficients of Equation 2.1 should model the correlations of all possible speech. This requires the calculation of the autocorrelation function for all possible speech. In practice a representative large number of samples are used to estimate the true correlations.

Using the approximate evaluation of expectations the Equations 2.5 can be rewritten;

$$\begin{aligned} R(T_s) - AR(0) - BR(T_s) &= 0 \\ R(2T_s) - AR(T_s) - BR(0) &= 0 \end{aligned} \tag{2.7}$$

Some algebraic manipulation solves A and B in terms of the auto-correlation functions;

$$\begin{aligned} A &= \frac{R(T_s)R(2T_s) - R(T_s)R(0)}{R^2(T_s) - R^2(0)} \\ B &= \frac{R^2(T_s) - R(0)R(2T_s)}{R^2(T_s) - R^2(0)} \end{aligned} \tag{2.8}$$

As the prediction coefficients A and B are not updated but calculated once they must be representative for all speech. It is found that as the delay increases the speech waveform decorrelates quickly which limits the use of higher order fixed predictors.

If a higher prediction order is now employed for a fixed block size of N samples of speech the prediction may be written,

$$s(n) \approx \alpha_1 s(n-1) + \alpha_2 s(n-2) + \alpha_3 s(n-3)+, \ldots, +\alpha_p s(n-p) \tag{2.9}$$

The notation has been altered to be less clumsy as a constant sampling period T_s may be assumed. The constant weighting coefficients α_i in the above expression are called the linear prediction coefficients. The prediction $\hat{s}(n)$ depends on p past values of the time series $s(n)$.

$$\hat{s}(n) = \alpha_1 s(n-1) + \alpha_2 s(n-2) + \alpha_3 s(n-3) +, \ldots, + \alpha_p s(n-p) \tag{2.10}$$

The error between the prediction and the next sample can then be written;

$$\begin{aligned} e_n &= s(n) - \hat{s}(n) \\ &= s(n) - \sum_{i=1}^{p} \alpha_i s(n-i) \end{aligned} \tag{2.11}$$

If α_i is redefined as $-\alpha_i$ the prediction error becomes,

$$e_n = \textstyle\sum_{i=0}^{p} \alpha_i s(n-i) \quad a_0 = 1 \tag{2.12}$$

The α_i are determined using the mean square error criterion,

$$E[e_n^2] = E[(\sum_{i=0}^{p} \alpha_i s(n-i))^2] \tag{2.13}$$

The optimum value of the estimate $\hat{s}(n)$ is found when $E[e_n^2]$ is minimized, which can be found by setting to 0 each component of the partial differential of mean square error with respect to α_i;

$$\frac{\partial E[e_n^2]}{\partial \alpha_i} = 0 \quad i = 1, 2, 3, \ldots, p \tag{2.14}$$

A block of speech can be windowed by weighting each sample in the range $0 \leq n \leq W$ and setting sample values identically to 0 outside of this window. Expectation can be estimated using the auto-correlation function,

$$R(i-k) = \sum_{n=0}^{W+p-1} s(n-i)s(n-k) \tag{2.15}$$

If $m = n - i$ this may be rewritten as,

$$R(i-k) = \sum_{m=0}^{W-1-(i-k)} s(m)s(m+i-k) \tag{2.16}$$

We also note that;

$$\begin{aligned} R(j) &= \sum_{n=0}^{W-1-j} s(n)s(n+j) \\ &= \sum_{n=j}^{W-1} s(n)s(n-j) \end{aligned} \tag{2.17}$$

A set of p dimensional first order equations result from Equation 2.14 which can be expressed as,

$$\begin{bmatrix} R(0) & R(1) & R(2) & \cdots & R(p-1) \\ R(1) & R(0) & R(1) & \cdots & R(p-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R(p-1) & R(p-2) & R(p-3) & \cdots & R(1) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} = - \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(p) \end{bmatrix} \tag{2.18}$$

A solution can be found for this set of equations if the matrix is positive definite, which is guaranteed if windowing is used for the block of data.

Another method which can be used for determining the α_i, is known as the covariance method [53]. Consider a block of data of size L, then;

$$\phi(i,k) = \textstyle\sum_{n=0}^{L-1} s(n-i)s(n-k) \quad \begin{array}{l} i = 1,2,\ldots,p \\ k = 1,2,\ldots,p \end{array} \tag{2.19}$$

The set of simultaneous equations that give the α_i can be written as,

$$\begin{bmatrix} \phi(1,1) & \phi(1,2) & \phi(1,3) & \cdots & \phi(1,p) \\ \phi(2,1) & \phi(2,2) & \phi(2,3) & \cdots & \phi(2,p) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi(p,1) & \phi(p,2) & \phi(p,3) & \cdots & \phi(p,p) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} = - \begin{bmatrix} \phi(1,0) \\ \phi(2,0) \\ \vdots \\ \phi(p,0) \end{bmatrix} \tag{2.20}$$

The matrix is symmetric as $\phi(i,k) = \phi(k,i)$.

The two methods give very similar results and as the auto-correlation method always produces stable filters it was chosen for this work.

Given the prediction error e_n, the exact time series $s(n)$ can be reconstructed.

When a windowed discrete Fourier transform is performed on a block of ≥ 200 speech samples the power spectrum shows peaks during voicing. Voicing is a term used when vowels and vowel type sounds are uttered. Unvoiced speech refers to sounds like "s" and "f". The peaks found in the power spectra of voiced speech are known as formant peaks and the apex of the peak occurs at the formant frequency. The frequencies that are 3 dB down on the formant peaks can be used to measure the formant bandwidth. The linear predictive filter attempts to model the envelope of the power spectrum. Linggard [54] discusses spectrograms and formants with respect to utterances of different sounds.

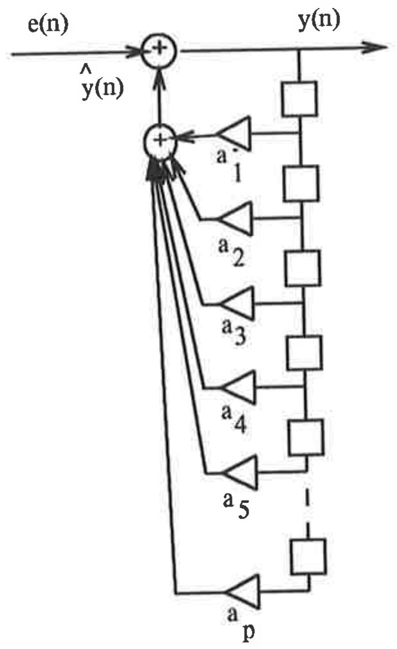


Figure 2.2: All Pole Auto-Recursive Prediction Filter.

The prediction filter illustrated in Figure 2.2 shows the digital filter implementation. The transfer function of the filter may be expressed as,

$$H(z) = \frac{1}{A(z)} \tag{2.21}$$

Finding the roots of $A(z)$ determines the poles of $H(z)$.

$$A(z) = 1 + \alpha_1 z^{-1} + \alpha_2 z^{-2} + \ldots + \alpha_p z^{-p} \tag{2.22}$$

The roots of $A(z)$ are the same as the roots of the equation,

$$z^p + \alpha_1 z^{p-1} + \alpha_2 z^{p-2} + \ldots + \alpha_p = 0 \tag{2.23}$$

This equation normally has $p/2$ complex conjugate pair solutions. If the solutions are written as,

$$\begin{aligned} z_i &= r_i e^{j\omega_i} \\ \bar{z}_i &= r_i e^{-j\omega_i} \end{aligned} \tag{2.24}$$

then the formant frequency is given by,

$$f_i = \frac{\omega_i}{2\pi} = \frac{1}{2\pi T_s} \arg(z_i) \tag{2.25}$$

and the formant bandwidth bw_i will be given by,

$$bw_i = \frac{1}{\pi T_s} |\log r_i| \quad \text{Hz} \tag{2.26}$$

If the order of the analysis p is smaller than the true order of the system some formant peaks close to each other will converge into one formant peak. If the model order matches the true order of the system the formant peaks match the peaks in the short term Fourier spectrum. When the model order is greater than the true order some of the formant peaks will not be true formants of the speech. For more details see Saito and Nakata Chapter 4 [53].

2.3 Autocorrelation of Speech

Central to predictive models of speech is the concept of the autocorrelation of speech which is covered in some detail in this Section. Autocorrelations are determined over a window of size W of speech samples $s(n)$. Weighting of the speech samples within the window is ignored at this stage. Letting $R(j)$ denote the autocorrelation for a sample delay j;

$$R(j) = \sum_{n=j}^{W-1} s(n)s(n-j) \tag{2.27}$$

Two speech waveforms of the words *fine* and *juice* are shown in Figure 2.3. For illustrative purposes the autocorrelation functions for the vowel 'i' in fine and fricative 's' sound in juice are shown in Figure 2.4. The data was obtained using sampling at 8 kHz and quantizing to 16 bits.

The quasi-periodic nature of the vowel sound in *fine* is quite evident in the speech waveform near the beginning of the utterance. The periodic structure towards the end of the utterance is the nasal 'n'. The periodic nature of the vowel 'i' is quite evident in the corresponding autocorrelation. The fricative 's' sound in *juice* is found at the end of the utterance. Fricatives are characterised by lower power than voiced sounds, like vowels and nasals, and display almost random like amplitude. The corresponding autocorrelation indicates correlations but at high frequencies, represented by correlations occurring within a few samples. Some fricatives exhibit a random autocorrelation.

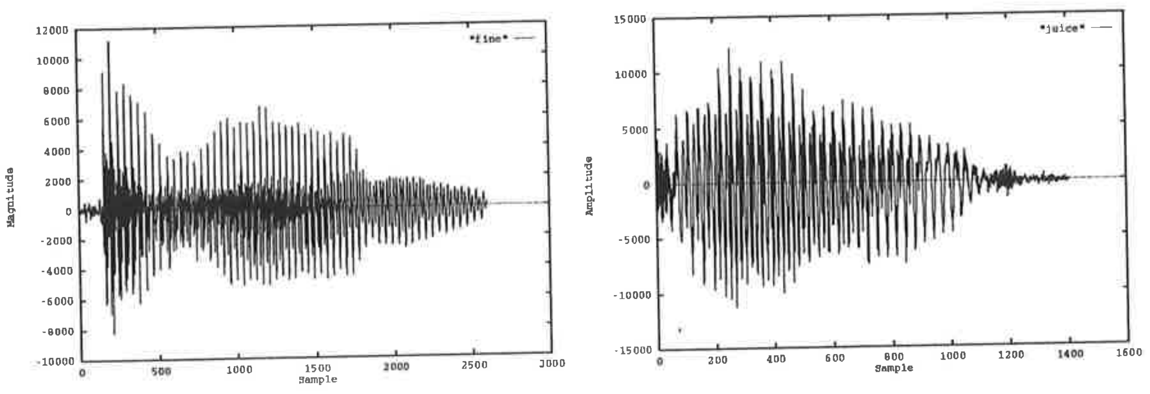


Figure 2.3: Speech Waveforms: fine and juice

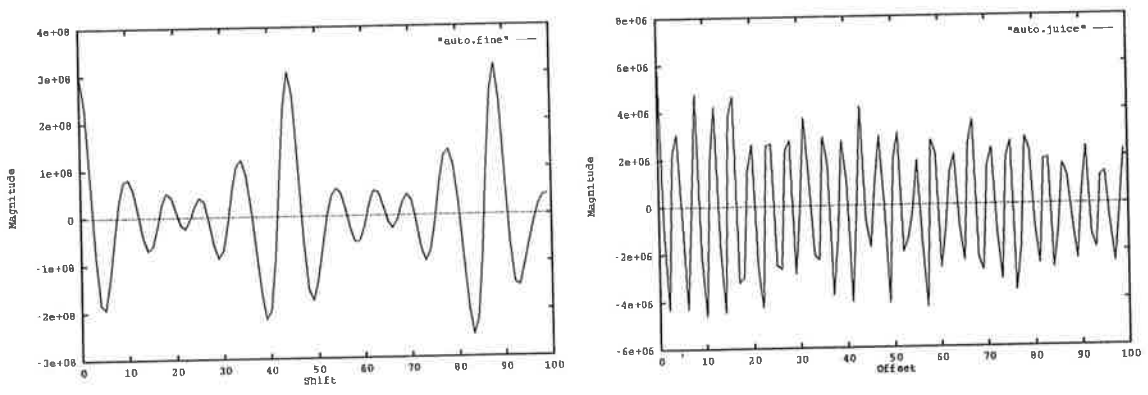


Figure 2.4: Autocorrelations: 'i' in fine and 's' in juice

2.4 Generic Code Excited Linear Prediction

In the class of linear predictive coders, Code Excited Linear Predictive (CELP) coders [1] [3] [4] [29] [84] [117] give the best subjective performance at lower bit rates; less than 8 Kbit/sec. A block diagram of the CELP coder is illustrated in Figure 2.5. The linear prediction coefficients a_i's, the adaptive gain and the stochastic gain are scalar quantised and transmitted as integers. The adaptive index and the stochastic index are also transmitted as integers. Channel errors may corrupt any of these parameters and mechanisms must be implemented to reduce the effect of these errors. Specifically the thesis addresses the protection of the a_i's and the indices by using different coding techniques.

The CELP coder is based on analysing a frame of speech and extracting the linear predictive coefficients (LPC's) for the frame. A speech frame is typically 16 to 25 milliseconds in length and sampled at 8 kilo-samples per second. Once the linear predictive coefficients have been extracted, the original waveform can be reconstructed exactly by the all pole synthesis filter provided the exact residual, obtained from filtering the speech using the LPC coefficients in the (finite impulse response) analysis filter, is known. The residual is the error between the predicted speech sample and the actual speech sample. Encoding of the residual is difficult as it must be done using the minimum number of bits whilst maintaining the ability to reproduce a faithful copy of the original waveform. The encoded residual may also be called the innovation sequence for the synthesis filter.

An analysis by synthesis approach is performed to encode the excitation that emulates the true residual. Fundamentally the approach attempts to reconstruct the original waveform given the LPC filter coefficients, gains (for amplitude matching) and a codebook of trial excitation vectors. The synthesised waveform that best matches the original waveform is chosen and the index for the chosen codebook entry transmitted as the encoded residual. A further refinement is incorporated by using a past innovation sequence to predict the current innovation sequence. It is well known that a previous pitch period is a good estimate of the present pitch period, particularly when a vowel is uttered, and as such a previous innovation sequence should provide information about the present pitch period. Similarity between pitch periods for voiced speech is well illustrated using the autocorrelation function for blocks of speech of 200 to 400 samples. The autocorrelation function shows distinct maxima when shifted by a pitch period. Many pitch estimation algorithms exploit the similarity from one pitch period

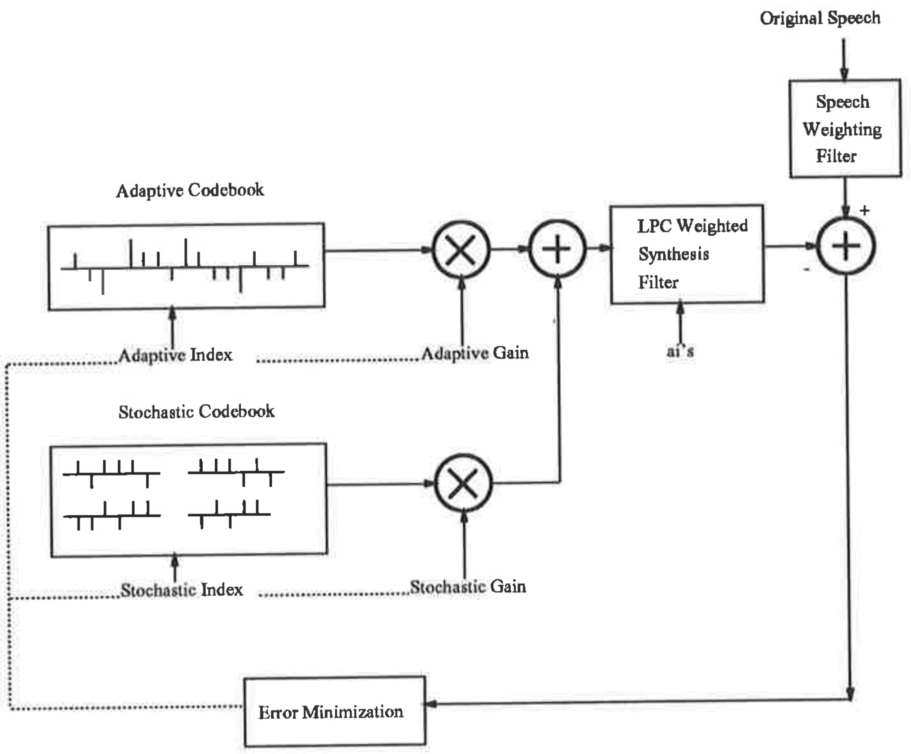


Figure 2.5: Block Diagram of Basic CELP Structure

to the next in the time domain for their operation. Rabiner and Schafer discuss the time domain pitch structure of speech in their book [55].

There are two main philosophies used with this *adaptive codebook*. In the first method the codebook contains a finite length of previous innovation and a pitch filter with a small number of taps is used to extract a weighted average innovation which is then added to the innovation from the stochastic codebook. The combined innovation is then appended to the adaptive codebook and the oldest section of innovation discarded. The second approach is to maintain the adaptive codebook as mentioned above but allow the contribution to the next block of innovation to come from any location within the codebook. The contribution however must be a vector of contiguous samples. Inclusion of an adaptive codebook requires an additional, separate, search for the best innovation: firstly the best adaptive codebook innovation must be found using analysis by synthesis, then secondly the best innovation from the stochastic codebook needs to be found. It is the combined innovation that is used in the synthesis process and subsequently matched to the original waveform. Transmitted parameters thus include quantised LPC coefficients, indices for the adaptive and stochastic codebooks and the gains associated with the adaptive and stochastic codebooks.

Figure 2.5 shows the block diagram for the generic CELP coder. The selected optimum vectors from the adaptive and stochastic codebooks are scaled and summed to produce an innovation sequence or residual. The resulting residual is then passed through the synthesis filter, to reproduce an estimate of the original speech. It was shown [29] that the auditory system can tolerate higher levels of noise in regions of higher amplitude (generally the formant regions) than regions of low amplitude in the frequency domain, which led to the development of a weighting filter. Atal and Schroeder [29] described an efficient realisation of the weighting filter by minimizing the subjective loudness of the quantisation noise. A frequency weighted error may be written as;

$$\epsilon = \int_0^{f_s} |S(f) - \hat{S}(f)| W(f) df \tag{2.28}$$

where $S(f)$ and $\hat{S}(f)$ are the Fourier transforms of the original and synthetic speech signal and f_s is the sampling frequency. The weighting function is chosen to de-emphasise the high energy formant regions in the spectrum.

A suitable weighting is produced by a filter that has the form;

$$W(z) = \frac{A(z)}{A(z/\gamma)} \tag{2.29}$$

where $A(z) = 1 - P(z)$ and $P(z)$ is the predictor polynomial determined from the linear predictor coefficients.

$$P(z) = \sum_{k=1}^{p} \alpha_k z^{-k} \tag{2.30}$$

The number of poles for the model is given by p, and α_i are the predictor coefficients. γ is chosen by the degree to which the formant regions should be de-emphasised in the error spectrum. Decreasing γ increases the bandwidth of the poles and the increase ω is given by;

$$\omega = \frac{-f_s}{\pi} ln(\gamma) \tag{2.31}$$

where f_s is the sampling frequency.

When $\gamma = 0$, $W(z) = A(z)$ and the output noise has the same envelope as the original spectrum while when $\gamma = 1$, $W(z) = 1$ which is equivalent to no weighting at all. The idea is to force the quantisation noise into regions of relatively high concentration of speech energy (formant regions) as the auditory system will tolerate larger errors there than in the *valleys* between formant regions. The incoming speech is weighted with filter $W(z)$ and then compared with the weighted synthetic speech. Generally, γ in Equations 2.32 2.29 takes on values between 0.7 and 0.9. The synthetic weighted speech is generated using a filter of the form,

$$H_w(z) = \frac{1}{A(z/\gamma)} \tag{2.32}$$

The mean square error between the synthesised weighted speech and weighted original speech is used as the performance measure in the minimization algorithm to select the most appropriate indices from the codebooks.

A further difficulty arises in the coder because of the memory of the synthesis filter, refer to Figure 2.2. At the receiver terminal, the speech synthesis procedure still has values in the filter memory at the end of a block of speech. When running in continuous mode, the memory will still have samples from the previous block which are retained on the start of synthesis of the following block. This aids in the smoothing between blocks. This architecture implies that the impulse response of the weighting filter $H_w(z)$ must be removed from the new incoming block of weighted speech prior to comparison with the weighted synthesised speech. In doing this, memory hangover effects are eliminated in the receiver's synthesis procedure, that is, the samples in the receiver synthesis filter are accounted for.

All of the parameters in the CELP model are quantised. The LP coefficients need to be quantised to limit the number of bits transmitted. Non-linear quantisers Lloyd [56], Max [57] are known to produce lower mean square error in quantisation, and algorithms have been developed to give the quantisation levels for a particular source. Non-linear quantisers are generally chosen to represent the quantisation levels for the CELP parameters. Decisions need to be made as to the number of bits required to encode each of the parameters while maintaining acceptable reproduction performance. The number of bits chosen for the various parameters is known as bit allocation.

As the basic structure of the CELP coder has now been briefly outlined, typical bit allocations for transmission over the channel will be illustrated. Cox [20] describes bit allocations for an 8 kbps CELP coder shown in Table 2.1.

Parameter	Framesize	Bits
LPC Predictor (10th order)	18 ms	36
Codebook (sign,index,gain)	4.5 ms	1+10+5
Pitch Predictor (gain,index)	4.5 ms	3+7
Unused Bits	18 ms	4

Table 2.1: Bit Allocation for Cox's CELP coder

The bit allocations for the various parameters are fairly typical for a CELP coder of this bit rate. It should be noted that the speech frame/block was broken into four sub-frames and that the codebook and pitch predictor information was determined four times for the frame.

Another bit allocation scheme is discussed in Perkis [21] for a CELP coder operating at 5 kbps. Bit allocation is indicated in Table 2.2.

Parameter	Framesize	Bits
LPC Predictor (10th order)	30 ms	32
Codebook (index,gain)	5 ms	5+5
Adaptive c-b (gain,index)	5 ms	3+7

Table 2.2: Bit Allocation for Perkis' CELP coder

Note that Table 2.2 implies that the speech frame size was 240 samples for a sampling rate of 8 kilo-samples per second. This frame size approaches the upper limit of the frame sizes used by CELP coders, as for larger frame sizes the assumption in the LPC analysis of statistical stationarity begins to become invalid. Also the breaking up of the frame into 6 sub-frames is unusual in that the norm is to use 4 sub-frames. For each subframe a new index into each codebook is required. This implies that the stochastic codebook must be small to keep the bit rate low. The short sub-frames assist in coping with the nonstationary spectrum.

There is a great degree of flexibility in bit allocation for the model parameters. However, care should be taken when allocating bits as the number of bits assigned to a parameter has a direct bearing on the quantisation resolution. If too few bits are assigned to a parameter, significant distortions in the synthesised speech may result. However, the finer the quantisation the greater the number of bits to be transmitted. This in turn means that the bit rate increases and a greater channel bandwidth is required for transmission. A delicate trade-off is called for with respect to bit rate and auditory performance of the reproduced, synthetic speech.

To determine the trade-offs, listening tests must be performed on the speech coder in order to obtain subjective performance. Objective measures are generally unreliable, however for some parameters objective functions do exhibit a reasonable correlation with subjective evaluations. An example of this is where LPC's encoding and performance can be checked using the spectral distortion objective measure [36]. Spectral distortion is covered in more detail in Chapter 4, Section 4.4.

Chapter 3

Pitch Estimation Based on Chaos Theory

3.1 Introduction

Pitch estimation is a parameter of considerable importance in the processing of speech. Pitch is defined for the thesis as the distance between glottal pulses in the production of voiced sounds, for example vowel like sounds.

A speech coder usually operates on blocks of information also called frames. Another term used in the literature, because of the blocked nature, is "packet". The block or frame terminology will be primarily used when discussing the processing at the transmit end and the term packet will be principally used when discussing the the received frame. A frame of data is generally of ≈20 ms duration which gives, for a sampling rate of 8 kHz, 160 samples. A frame may be broken down into smaller equal sized units of contiguous samples, called subframes, for processing.

Pitch is of interest in three primary areas in the thesis:

1. frame (packet) substitution for severely corrupted received packets,
2. long term prediction in CELP coders,
3. post processing of the synthesised speech.

These three items will be discussed in turn.

CELP coders process a block of speech at a time, obtaining parameters to describe it and then transmitting a block or packet of parameters. In blocked voice communications there is always the possibility that gross errors, such as the loss of a complete parameter block, may occur. Generally this kind of problem has been studied in the context of a missing packet in packet voice communications. For example see Goodman [58]. At the receive end when a packet is lost, it is necessary for the receiver to present something to the listener to give the perception of smooth uninterrupted speech. The missing packet or block must be determined from past speech, or, past and future packets if buffering and extra time delay is acceptable. The missing block is estimated from the data available and a substitution of the estimated block for the missing block is performed. Block substitution is useful if blocks are lost at a rate greater than about one in a hundred. If block losses are very rare events, the substitution of silence or pink noise[1] is acceptable [58].

The principle behind block substitutions is the exploitation of the natural redundancy that occurs in speech. In particular use is made of the facts that one pitch period appears very similar to the next during voiced speech and during unvoiced speech the waveform is very noise like and uncorrelated. These features were illustrated in the previous chapter. Substitution mechanisms fall into two broad categories:

1. *Pattern Matching* — This method looks at M samples prior to the missing block as a template and then looks back in history to find a good match for the template and collects L samples following (where L is the length of the block). The L samples are then substituted for the missing block.

2. *Pitch Detection* — If P is the length of a pitch period in samples prior to block loss, the P samples may be repeated till the missing block is filled. If no pitch can be found, the speech is considered unvoiced and the previous block is used to substitute for the missing block.

The work in this chapter is based on the second method of substitution. The second method was chosen in order to eliminate any further encoding and decoding delay in the speech codec. Pattern matching generally requires sufficient buffering into the past to find a good match or extra buffering to receive the following packet to perform an interpolation between the two packets.

Long term prediction for the CELP coder may use an adaptive codebook or a pitch

[1]Noise with a power spectrum proportional to $1/frequency$ is known as pink noise.

predictor; this was explained in more detail in Chapter 2. It has been found [2] that pitch estimators that can estimate to within a fraction of a sample and then perform interpolation, yield a better perceptual quality than estimators that yield pitch values to integral samples.

Post processing of the received speech is generally performed using frequency domain analysis based on the pitch period. Transcoso [27] describe such a technique where the spectrum of the reconstructed speech is analysed in terms of harmonics of the pitch period. Energies found at non-harmonic frequencies are attenuated and the speech is then converted back to the time domain. This process is claimed to improve the perceived quality. A time domain post processing technique was investigated and is presented at the end of Chapter 6.

The application of the pitch estimator to the substitution problem is covered in the latter sections of this chapter.

3.1.1 A Novel Approach to Pitch Estimation

Obtaining pitch estimators that are inherently stable and accurate has been a problem for many years and as yet remains an unsolved problem. This Chapter discusses an approach to pitch estimation that uses ideas from chaos and modern dynamical systems theory and the experimental methods used in analysing these systems.

A new reliable and robust super resolution pitch estimator, which has promising performance and yields an accuracy to 1/10 of a sample will be discussed. As the multidimensional phase space representation (explained below) is obtained by simple delays, the computational load is not great although greater reductions in processing may be possible using decimation of the low pass filtered input signal.

Extra information about the speech waveform makes the pitch estimation and pitch tracking, the process of obtaining a smooth pitch contour, an easier task. Estimation of whether speech is voiced or unvoiced is also useful, as unvoiced (noise-like) speech does not display a pitch period.

Two new speech waveform classification methods based on zero crossings were presented. These methods based on the zero set, the spacing between successive zero crossings of the speech waveform, and the Devil's staircase yield useful classification data. They however have a disadvantage in that it must be ensured that the speech does not have a dc bias. The zero set method was less sensitive as small amplitude signals can be ignored.

New voicing indicators were devised which were based on: direction of travel in the two dimensional phase plane orbit, the autocorrelation function's decorrelation shift, the zero set and the traditional root mean square signal power. Determination of the decorrelation time, the first time that the autocorrelation function of the time domain waveform for the frame goes through zero, was necessary for the phase space construction so this parameter was obtained for "free".

A sliding window approach was taken to obtain a continuous pitch contour. When no pitch was found 10 new samples were added to the buffer and then pitch processing was performed. During voiced sounds where a pitch period was apparent, one third of the estimated pitch period in new samples were added to the buffer. All correlations were performed only on samples already in the buffer. This method ensured minimum buffering delays.

3.2 The Non-Linear Dynamical Model

It has been shown by various authors, for example Tishby[25], that speech shows the properties of a non-linear dynamical system. In this section the concepts of non-linear dynamical systems will be reviewed; for more detail see: Percival and Richards [59], Drazin [23] Chapters 1 and 2, and Wiggins [24] Chapter 1.

Let a dynamical system be represented by the general equation;

$$\begin{array}{l} \dot{x} = f(x, u(t)) \quad x \in \Re^n \quad u \in \Re^p \quad t \in \Re \\ \text{where } \dot{x} \equiv \frac{dx}{dt} \end{array} \tag{3.1}$$

where:

- $\Re^n$ is the n-dimensional Euclidean space,
- $f : U \longrightarrow \Re^n$ where U is an open set, described below, on $\Re^n$,
- x are the dependent variables and the space that they form is known as phase space,
- $u(t)$ is a possible p dimensional control space.

An open sphere $S_r(x_0)$ with center x_0 and radius r is the subset of a $\Re^n$ defined by,

$$S_r(x_0) = \{x : d(x, x_0) < r\} \tag{3.2}$$

where $d(x, x_0)$ is the Euclidean distance between the points x and x_0.

A subset G of a Euclidean space is an *open set* if, given any point x in G there exists a positive real number r such that $S_r(x)$ is contained in or equals the set G; $S_r(x) \subseteq G$. This implies that an open subset G of $\Re^n$ is logically equivalent to a union of open spheres.

Note that $f(.)$ is a vector function. The trajectory or orbit, traced by the point $x(t)$, moves through phase space; this is described in a little more detail below. A limit cycle occurs if a stable periodic solution exists to which all starting points in the phase space move towards in the limit $t \to \infty$. $u(t)$ is the driving force of the system. As $u(t)$ varies, an established limit cycle may change its shape and/or period. It was assumed that the glottal pulse shape remained reasonably constant, only the period between pulses changed, and the vocal tract changed slowly during voicing. Real speech exhibits only slow variations during voicing, making one pitch period very similar to the next. The assumption implies that the time domain waveform of voiced speech can be used to construct an almost periodic trajectory in phase space. It was the period of the limit-like cycles that was estimated in this work. Natural speech has the difficulty that $u(t)$ varies as well as the x, due to changes in driving force and vocal tract parameters.

A trajectory $\Phi \equiv \Phi(t)$ is a curve in $\Re^n$ which is a solution of the above system of equations. It is a vector function which maps an interval of time to $\Re^n$. That is,

$$\Phi : I \to \Re^n \tag{3.3}$$

and satisfies

$$\begin{aligned} \dot{\Phi}(t) &= f(\Phi(t), u(t)) \\ \Phi(t_0) &= x_0 \\ \dot{\Phi}(t_0) &= f(x_0, u(t_0)) \end{aligned} \tag{3.4}$$

The set of points $\Phi(t)$ is the solution curve to the above set of equations. The goal is to understand the overall geometry of all solution curves. As $u(t)$ is altered the solution curves will differ. A trajectory or orbit is the locus of the point $\Phi(t)$. There is in general a family of trajectories which satisfy the system of equations. Members of the family of trajectories after a period of time appear to follow an asymptotic regime.

This regime, neglecting the transient behaviour, will be called an attractor. The limit cycle described above is one possible form of attractor.

It is the limit like cycles that are of most interest as it will be seen that these attractors are associated with voiced sounds. True chaotic behaviour indicates a transition to a fricative which is unvoiced and as such not of great interest to us. The actual details of individual trajectories may be quite different. Despite this, the vector field is fixed and it is this field which can be used to describe attracting sets for the trajectories.

Eckmann [68] defines an attractor in terms of a phase flow (the set of all possible motions in the vector field). For more details on (phase) flows see for example, Percival & Richards [59] Chapter 3.

Eckmann defines an attractor as follows, [68].

Definition: An *attractor* for the flow T^t is a compact set X satisfying;

1. X is invariant under $T^t : T^tX = X$
2. X has a shrinking neighbourhood, i.e. there is an open neighbourhood U of X, $U \supset X$ such that $T^tU \subset U$ for $t > 0$ and $X = \bigcap_{t>0} T^tU$
3. The flow T^t on X is *recurrent* and *indecomposable.* Recurrent means T^t is nowhere transient on X: If U is an open set in V and if $U \cap V \neq \emptyset$ then there are arbitrarily large values for t such that $T^tx \in X \cap U$ when $x \in X \cap U$. In-decomposable means X cannot be split into two nontrivial closed invariant pieces.

For more details see Eckmann's [68] paper. The issue of interest to us is the attractor's limiting set. By observing selected phase space points on the trajectories and noting the time of the next closest pass to the selected points as the trajectory moves through phase space an estimate of the period of the trajectory can be made. Extra information was also available about the nature of the trajectory. The phase volume of a dissipative system decreases and the volume decreases to 0 as $t \to \infty$ implying that in the limit x is inside the set X.

The dynamical system may be discretized by describing it via a map,

$$x_{t+1} = F(x_t, u_t) \quad \text{where } t \in Z \tag{3.5}$$

The iterated map $F(.)$ takes a vector of dependent variables x_t which is a point in the Euclidean space $\Re^m$ and an independent variable $t \in Z$ (Z is the set of integers) to yield the next point. The next point x_{t+1} is also a point in $\Re^m$. This implies that time is discretized in the map and made to take on integer values. The function $F(.)$ describes the path of the trajectory in discrete steps. If the starting point x_0 is known application of $F(x_0)$ gives x_1. Iteration of the procedure plots the entire trajectory. A problem arises in experimental systems in how to determine the phase space. Fortunately the application of theorems by Whitney[69] and Takens [70] make it possible to construct a phase space portrait by taking a single sampled time series of a variable S(t). Phase space may be constructed using tuples $[S(t), S(t+\tau), S(t+2\tau), \ldots]$ where τ is an arbitrary delay; this will discussed in more detail below.

A primary issue is determining the dimension or number of dependent variables x required to describe the system. The term dimension in this context denotes the dimension of phase space.

Using an argument from Takens[70], a vector field can be characterized by the trajectories within the field. In particular by observing dynamical symmetries of the trajectories it is possible to reduce the dimensionality $\Re^n$ of the embedding space to $\Re^m$ where $m < n$ of the trajectory space. The kinds of symmetries that may be used for example are translations in time of one trajectory to another possible trajectory.

The dimension of the trajectory provides a lower bound to the number of variables required to model the system. Many methods have been presented to determine the number of dimensions but the most popular is the method presented by Grassberger and Procaccia[63].

The number of dimensions used in the following work was guided by Tishby's [25] work. Work also by Condie [64] [65] suggests that the required number of independent variables for speech is around four. A range of dimensions from 1D to 6D were considered to determine the most accurate and efficient dimensionality for the task at hand.

3.3 Determining the Delay

In a noise free system the delays (τ_i) used to construct the multidimensional phase space from the single time series may be totally arbitrary. The only proviso is that the delays for the second, third, fourth, etc, dimension are monotonically increasing.

This implies $\tau_0 < \tau_1 < \tau_2 < \ldots < \tau_{n-1}$ are the arbitrary delays used with the time series $S(t)$ to construct the n-dimensional tuple space $[S(t), S(t+\tau_0), S(t+\tau_1), S(t+\tau_2), \ldots, S(t+\tau_{n-1})]$. Generally a delay τ is chosen in terms of integral number of samples and each τ_i is made an integral multiple of τ. Experiments show that changes to the delay alter the appearance of the attractor which is easily witnessed in a two dimensional representation. Varying the delay tends to *rotate* the attractor [71].

Ideally the delay should decorrelate the different dimensions in phase space, this was addressed by Fraser and Swinney in their work [71]. They used the concept of mutual information from information theory.

Information theory defines entropy in terms of probabilities of messages. Frazer and Swinney considered the messages as the values that measurements of the attractor took. It is known that strange attractors are ergodic [2] [72] and have a well defined asymptotic probability distribution. They define the entire set of messages $\{s_1, s_2, s_3, \ldots, s_n\}$ as S, and the associated probabilities as $\{P_s(s_1), P_s(s_2), \ldots, P_s(s_n)\}$, where $P_s(s_1)$ is the probability density at s_1. The entropy can then be defined by,

$$H(S) = -\sum_{i=1}^{n} P_s(s_i) \log P_s(s_i) \tag{3.6}$$

For a time series $x(t)$, the minimum dependence of $x(t+\tau)$ on $x(t)$ is of interest as the appropriately chosen τ will rotate the attractor to give a broad profile. They assigned, $[s, q] = [x(t), x(t+\tau)]$ to describe a coupled system (S, Q). Mutual information may be described as the measure of uncertainty in measuring q given the message s was s_i. The mutual information $I(Q, S)$ is given by,

$$I(Q, S) = H(S) + H(Q) - H(Q, S) \tag{3.7}$$

where $H(S, Q)$ is given by,

$$H(S, Q) = -\sum_{i,j=1}^{n} P_{sq}(s_i, q_j) \log P_{sq}(s_i, q_j) \tag{3.8}$$

Frazer and Swinney then went on to describe a recursive algorithm to estimate the probabilities. However to be accurate enough a great number of points are necessary to get a good estimate of τ and with a low number of points, the estimate of τ was very inaccurate.

They showed that the delay that gave the mutual information minimum produced in general better estimates of decorrelation than finding the first point of decorrelation

[2]Ergodic: the time average tends to the ensemble average in the limit $T \rightarrow \infty$.

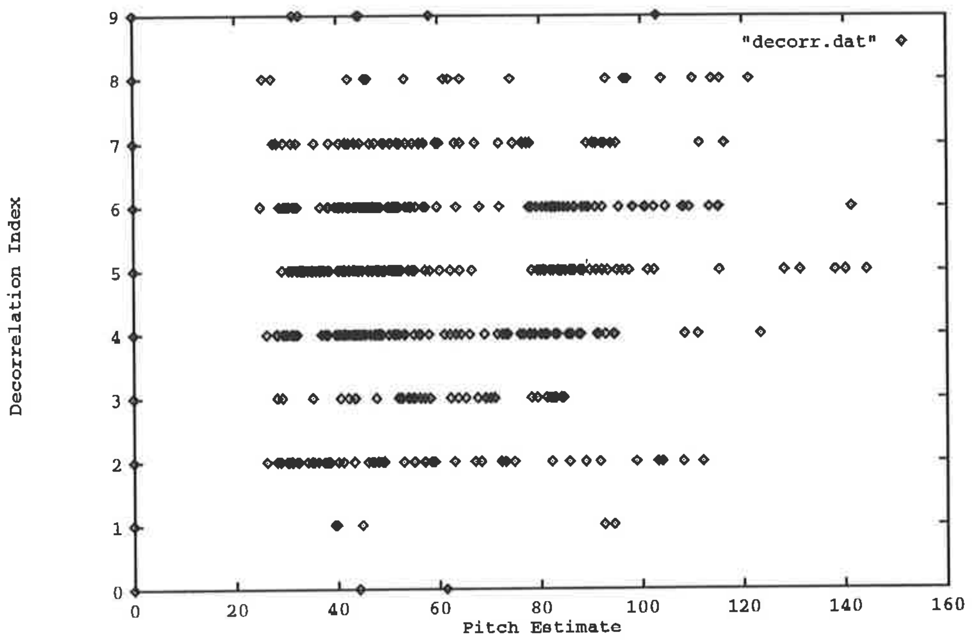


Figure 3.1: Decorrelation Lags in Samples versus Pitch Estimate

(zero crossing) of the autocorrelation function. The first zero of the autocorrelation function gives only linear independence of the phase space coordinates. Due to the non-stationarity of speech, only a low number of sample points for an attractor could be obtained; hence the autocorrelation decorrelation time [3] was used.

The resulting attractors for two dimensional phase space qualitatively looked very similar to the phase plane constructed from the analytic signal [4]. The author undertook experiments, to determine if the different delays altered the pitch estimate. It was found that essentially the same pitch estimate was found for delays greater than the decorrelation value. For delays less than the decorrelation value a wide variance in the pitch estimates was found.

A follow on experiment was undertaken to determine if the decorrelation delay value τ had a relation to pitch, thus avoiding the calculation of the autocorrelation function. The data was obtained for four separate utterances (two males and two females) each uttering two phonetically balanced sentences described in the following section. The pitch estimates were determined manually as described in Section 3.10.

[3]The first time that the autocorrelation function of the speech segment goes through 0.

[4]The analytic signal is constructed from couple $[x(t), \tilde{x}(t)]$, where $\tilde{x}(t)$ is the Hilbert transform of $x(t)$.

As can be seen from Figure 3.1 there does not appear to be any simple relationship between pitch and decorrelation time. However what did become clear was that the decorrelation time was predominantly between 4 to 6 samples. From visual inspection of the speech waveform and correlating it with decorrelation times, when the decorrelation time was less than 3 or greater than 7 the speech segment appeared to be predominantly unvoiced or semi-voiced.

On the basis of these findings the decorrelation delay search was constrained to be less than 10 samples, (for the sampling rate of 8 kHz), limiting the search space in the implementation.

3.4 Speech Database

The pitch estimation experiments used a speech database of eight Harvard test sentences [74] which were recorded in a low noise environment at Telecom Research Laboratories, Clayton Victoria. The sampling rate was 8 kHz and 16 bit scalar quantisation was used.

The sentences were:

male1 The Navy attacked the big task force. The hat brim was wide and too droopy.

female1 These days a chicken leg is a rare dish. The juice of lemons makes fine punch.

male2 The wagon moved on well oiled wheels. March the soldiers past the next hill.

female2 Kick the ball straight and follow through. Her purse was full of useless trash.

3.5 Initial Findings

The utterances detailed in the previous section were sampled at the constant rate of 8 kHz to obtain the time series, $S(t)$, which was used in the experiments described in this chapter. A phase space tuplet $[S(t), S(t+\tau), S(t+2\tau), \ldots]$ was created, where the time delay τ may be arbitrary for noiseless data but for noisy data it can be shown [60], that a good choice is between one tenth and one half of the mean orbital period, or, alternatively the mutual information decorrelation time described above could be used. Experimentally however, the decorrelation procedure described in the Section

Determining the Delay was found to be more useful as it gave consistent estimates of the pitch and was easily estimated.

For an overview of the techniques and terminology used in the study of chaotic systems see for example, May [67], Argoul [60], and Grebogi [66]. A dynamical system with 5 variables will have some orbit in five dimensional phase space. However the orbit may tend toward an attractor that may be embedded in only 3 dimensions, thus reducing the number of variables required to describe the system.

This work focussed on the periodic characteristic of the attractor rather than the classification of the strange attractors in speech. This approach differs slightly from the general work done previously on chaotic attractors. Another difference to previous experimental work was the number of samples available for a particular attractor. Sampling of the original speech was at 8 kHz and assuming a pitch of 80 Hz held for five pitch periods it was found, only 500 samples were available on a particular attractor before the next sound was formed. Chen [61] commented on the amount of data that should be available for numerical algorithms estimating an attractor's parameters, and comments that for orbits with 10–100 samples per orbit, about 5–50 orbits are necessary for a correlation dimension D of 2. Correlation dimension is a method of determining the phase space dimension, see Grassberger & Procaccia [63]. This was of considerable concern as large quantities of data are not available due to the time varying nature of speech. It should be noted that Tishby [25] used 20 segments of voiced speech to obtain an adequate number of samples for his experiments. His investigations indicated that voiced speech over a range of speakers had an embedding dimension of 3–5. He also noted that unvoiced speech had a higher embedding dimension but his figures may be a little unreliable due to the limited number of samples used.

Despite the above problems, experiments were undertaken by the author to determine if it was possible to determine the pitch period from phase space. Initial attempts at finding the pitch utilized packets of 200 samples of speech. If 200 samples of voiced speech are plotted in two dimensions (2D) the periodic nature of the attractor is evident. It was also noted that at the pitch period, the orbit tracks came within very close proximity of each other. Automation of a scheme for finding the points closest to each other in different orbits gave an estimate of pitch.

The main tool for obtaining the pitch estimate was the concept of a Poincaré map; see for example Wiggins [24] Chapter 1. Application of the Poincaré map to pitch estimation is novel and has not been found in the literature. The two principal forms

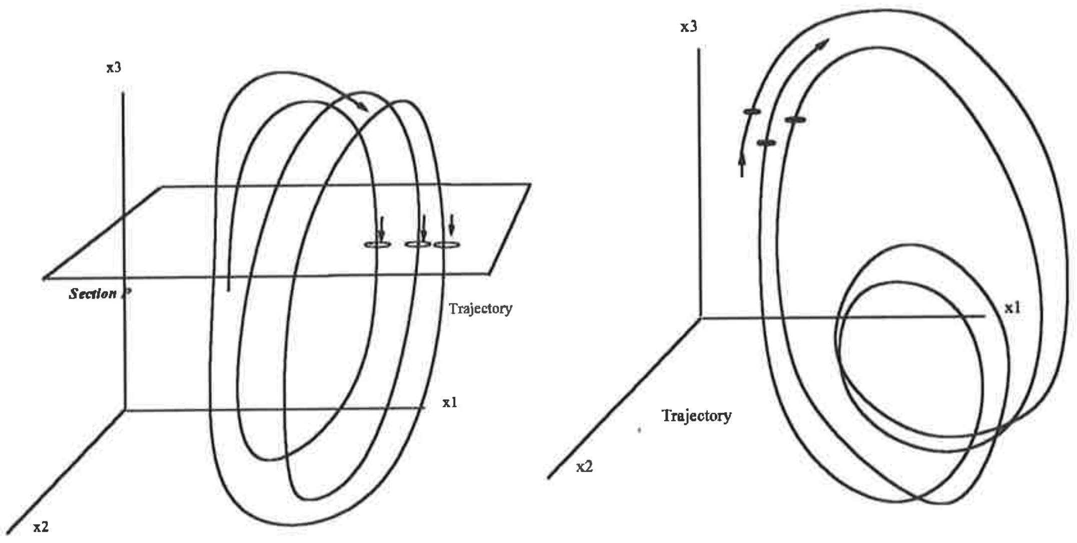


Figure 3.2: A Poincaré Section and a Poincaré Map generated using a *Stroboscope*

of the map are discussed below and indications made why the second form gave better results than the first. In what follows the term attractor is used to cover all forms of attractors.

3.5.1 Poincaré Sections

Figure 3.2 shows the conceptual construction mechanism of a Poincaré section. The idea was to find clusters of points close to each other in the section and measure the section piercing times to obtain pitch estimates. This basic concept worked provided that the speech did not vary too much during the window of observation.

Three dimensions were assumed, initially, to be adequate for pitch estimation. This choice was vindicated by later experiments. The three dimensions represented three dimensions of phase space which were constructed using a 3-tuple according to the method outlined above; $\{S(t), S(t+\tau), S(t+2\tau)\}$ where the delay between samples was taken to be integer but otherwise allowed to vary. The 3-tuple is referred to as a 3D sample.

To simplify the discussion consider initially only one plane, taken to be the semi-infinite half plane X $\geq$ 0. Each time a pair of 3D samples fell on either side of the plane, when the orbit was going in an anti-clockwise direction, a linear interpolation was made to determine the approximate piercing point of the phase space track. Figure 3.2 shows an example trajectory piercing a section in 3–space. The points determined by the intersection of the orbit and the section (from a given direction) produces the map, which is the object of interest. A collation was then made of the three pairs of closest piercings, and the number of samples between piercings within a pair gave a pitch estimate for that pair.

Naturally a simple technique such as this does not suffice in practice, as it is not possible to determine the orientation in phase space of the attractor. To alleviate this problem three half planes for the 3D data were used and the resulting data analysed, rejecting cases where the orbit track was at too oblique an angle to the section. This method gave good results for sounds like *i* in *wide* but became unreliable when the voiced sound changed, producing a range of fractional pitch periods. The fractional pitch periods were characterised by; 1/3, 1/2, 2/3, 3/4, 1, 4/3, 3/2, 2. At times some 1/8 fractions appeared. Although this was interesting it was not followed up because a method that could discriminate the true pitch period was required. The "true" pitch period was determined by visual inspection in order to evaluate the performance of the algorithm. The method by which pitch estimation is performed manually by visual inspection is covered in Section 3.10.

3.5.2 Poincaré Section in Time

Another way of creating a Poincaré section is to try and freeze the position of a notional point travelling along the orbit using a time section. This is akin to using a stroboscope to freeze the position of an harmonic oscillator by adjusting the strobe time till it closely approximates the period of the system. Figure 3.2 shows a trajectory with points marked to indicate the points frozen by a hypothetical stroboscope. To determine how closely the orbits match, a mean square error is calculated between points determined in the present period and those of the previous period for an ensemble of sections (all with the same time lag). An ensemble was generated by using an ensemble of stroboscopes all set to the same period but at fixed phase increments. The minimum of the averages of mean square error (over the ensemble), occurred when the best match between orbits was found. The best correlation occurred when the most accurate time lapse was chosen and this gave the estimate of the pitch. The expression for this process is given below. This method has similarities to that of Medan [76] although there are significant differences. The main difference is that Medan et al's method requires an accurate estimate of the window (of pitch length) while the present does not, a reasonable estimate is sufficient.

The expression for the pitch estimate is given by;

$$P_{sr} = \min_{winlo<\lambda<winh} \left\{ \frac{\sum_{i=\varrho}^{\varrho+P_{est}} \left[x(t_i,\tau) - x(t_{i+\lambda},\tau)\right]^2}{P_{est}} \right\} \tag{3.9}$$

where the value of λ that minimizes the expression is the super resolution pitch estimate P_{sr}. P_{est} is an integral pitch estimate, and *winlo* and *winhi* are the lower and upper limits for the range of λ. τ is the delay associated with the construction of the x–tuples. ρ is the array offset for the start of the search.

The technique described above constitutes a considerable computational burden and as such, a method for reducing the burden was required. A traditional pitch estimator is the Absolute Mean Difference Function (AMDF) which is described in many texts on speech processing; see for example Deller [26]. Using the multi-dimensional phase space notion, pitch estimates were obtained using a multidimensional AMDF. The multi-dimensional AMDF is given by,

$$AMDF = \frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{N} |x_i(t) - x_i(t-j)| \tag{3.10}$$

where M is the window length and N is the number of dimensions.

The AMDF estimate, which gave a resolution to the nearest time sample in the series, was used to obtain a window of search for the more computationally intensive section search, P_{sr}, indicated above.

To compare the AMDF function for one and three dimensions a series of experiments were performed to determine their relative performances. For illustrative purposes a voiced section of the word *fine* uttered by a high pitched female is used to visually observe the qualitative differences. Figure 3.3 illustrates how the three dimensional version often found the pitch period with the absolute minimum while the one dimensional version found double the pitch period. On average the three dimensional version found the period more reliably, however, both the one and three dimensional versions did have the problem of not always finding the true pitch period. This problem can easily be visualized by noting the multiple minima in the functions and that the absolute minimum may not correspond to the pitch estimate.

Experiments were also performed to determine if the three dimensional version of the AMDF had greater noise immunity than the one dimensional AMDF. To this end

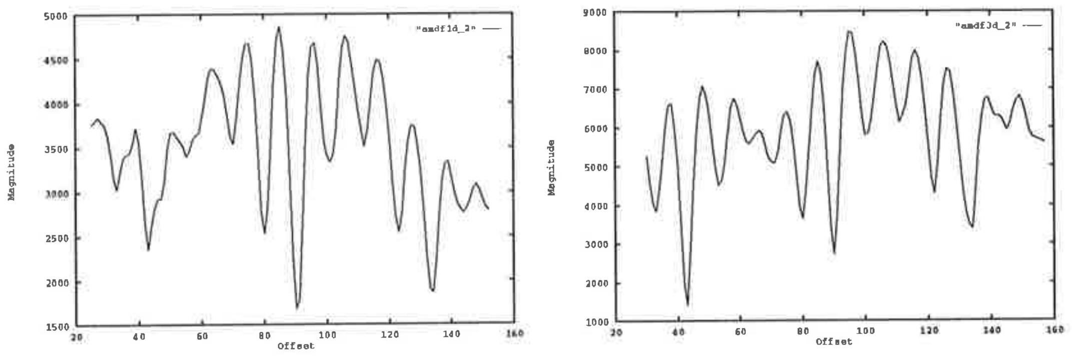


Figure 3.3: Comparison of 1 Dimensional and 3 Dimensional AMDF's.

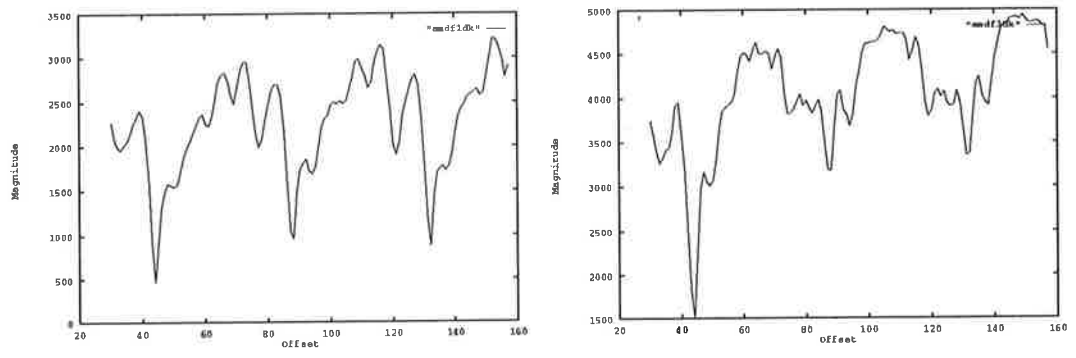


Figure 3.4: Reference 1 Dimensional and 3 Dimensional AMDF's for Noise Tests.

a voiced section of fine was used and the noiseless reference AMDF's are shown in Figure 3.4.

To test the performance of the AMDF's in noise, two types of noise were used. Firstly Additive White Gaussian Noise was added to the entire utterance of the word *fine*, uttered by the female speaker and the reference segment AMDF extracted and plotted. The noise had a standard deviation of 1500 in the same units as the original speech which had peak amplitude of approximately 8000 units over the extracted segment. As can be seen from Figure 3.5 there is not a great advantage in using the three dimensional version.

The second type of noise test was the case of a tone super-imposed on speech. This is a common problem with speech recorded "in the field". The tone frequency was set to 100 Hz with a peak amplitude of 2000 units. For tone noise the three dimensional AMDF did perform better. For the test utterance the 2000 units level was just at the

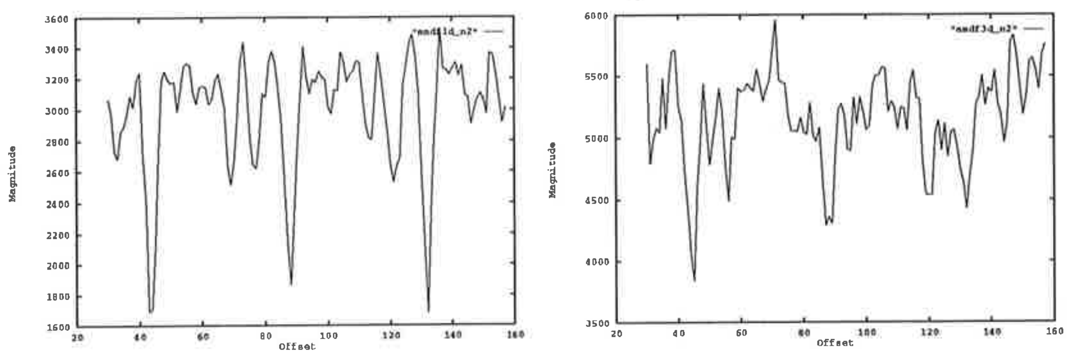


Figure 3.5: Comparison of 1 Dimensional and 3 Dimensional AMDF's of Speech with Additive White Gaussian Noise.

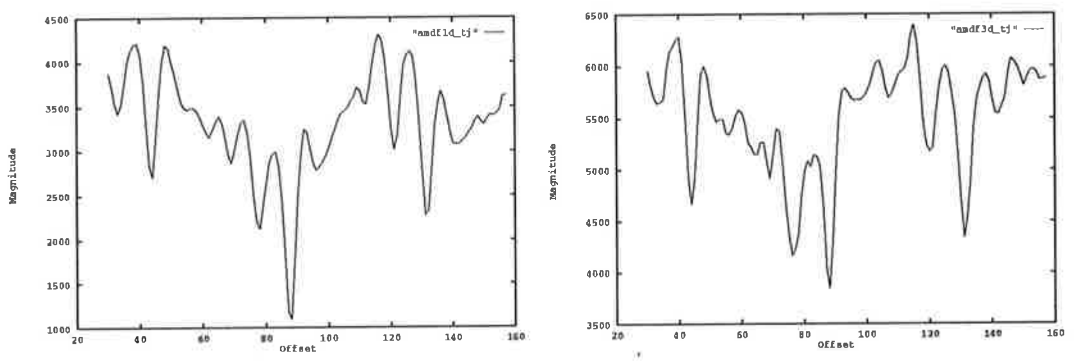


Figure 3.6: Comparison of 1 Dimensional and 3 Dimensional AMDF's with 100 Hz Tone Noise.

threshold where the three dimensional AMDF began to fail; this past the threshold where the one dimensional AMDF failed. The effect of tone noise is illustrated in Figure 3.6.

It was found that for the 3D and higher dimensional AMDF a fairly stable estimate was made. However the AMDF was found to give rise to rather characteristic errors such as pitch halving or doubling and in those cases the pitch estimate had to be adjusted.

Most of the errors made by the AMDF functions can be classed as gross pitch errors which can be corrected by using pitch tracking mechanisms, described in more detail later. An another useful technique is to use local minima picking of the AMDF function.

To aid in determining the correct *valley* of the AMDF function an array of significant *valleys* is determined. The position of *valleys* are determined from the minimum of each individual valley that crosses the threshold ($AMDF_{thresh}$) set by,

$$AMDF_{thresh} = AMDF_{min} + \frac{AMDF_{ave} - AMDF_{min}}{2} \tag{3.11}$$

where $AMDF_{min}$ is the global minimum of the AMDF and $AMDF_{ave}$ is the average of the AMDF for the current speech segment.

The relationship between the *valleys* is checked for harmonic relations and for finding the *valley* consistent with the previous estimate.

More details on the use of the array of minima is covered in Section 3.8 on "Pitch Tracking".

To assist in the task of determining good pitch estimates it became necessary to classify the speech into a limited number of classes with certain characteristics. Classification of speech is covered below.

3.6 Speech Classification

Speech classification was considered important to assist in determining if the utterance was voiced or unvoiced. It also assisted in identifying locations where pitch doubling or halving was likely to occur, that is near transitions from voiced to semi-voiced or unvoiced.

The objective was to construct a simple speech classification system that was pitch and speaker independent. A simple technique that has been used in the past is zero crossing. Zero crossing however did not give all the information that was required and some ideas from fractal theory were investigated. Fractal theory gives ways of classifying "rough" contours. In fractal theory, the set of zero crossings is called the zero set. The zero set however gives no information about amplitude between zero crossings, so another method was developed, the *Devil's Staircase*. For details of the more traditional *Devil's Staircase* see Feder [62].

For use in actual applications zero crossing detectors are considered unreliable because if a dc drift occurs in the speech waveform low speech power sounds like *s* may be incorrectly classified. A traditional root mean square (rms) power measure of the speech waveform can be valuable when used in conjunction with zero crossing information by setting a dead band where any speech below a certain level does not have zero crossing information evaluated. The rms power was available for *free* from the autocorrelation function, which was evaluated for the first ten integral sample shifts as part of the determination of the waveform's decorrelation phase shift, as described in Section 2.3.

Another useful parameter in classifying speech was the decorrelation time which was determined to obtain the sample delays for the phase space representations. Principally, if the decorrelation time was one sample, it was safe to classify the speech as unvoiced.

Detailed discussions follow describing: the *Zero Set* which gives a parameter describing fine structure of the speech waveform as it crosses the zero amplitude axis, *Devil's Staircase* which gives information on zero crossing spacing, and *Orbit Direction Change Indicator*.

The orbit direction change indicator returns a useful parameter which determines the number of times the direction of rotation changed in travelling around the two dimensional projection of the higher dimensional orbit (attractor) in a known time interval. This parameter gave some interesting information about the speech but also displayed classification ambiguities.

3.6.1 The Zero Set

To obtain a classification for voiced and unvoiced speech it is quite evident that the rate of zero crossings is important. Unvoiced speech is characterized by a very high zero crossing rate. A very simple extension to the zero crossing concept was based on the number of samples between consecutive crossings, to obtain two parameters: fine structure and coarse structure. After experimentation it was determined that fine structure be classed as that which has less than four samples between consecutive crossings (sampling at 8 kHz) and if greater it would be classed as coarse. It was desirable to have a continuous estimator that gave a *real number* to classify *coarse* and *fine* to simplify the overall pitch estimation algorithm and avoid batch counting over a window of speech. The recursive form used to estimate the *fine* parameter (x_n), is given by,

$$x_{n+1} = y + \alpha(x_n) \tag{3.12}$$

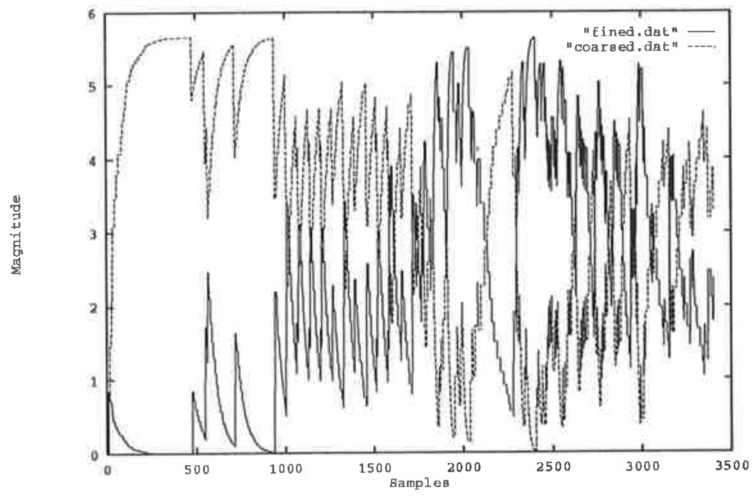


Figure 3.7: Structure Parameters for *wagon*: *fine* parameter solid line, and *coarse* parameter broken line.

where,

$$y = \begin{cases} 1 & \text{if no. of samples between this crossing and last} < 4 \\ 0 & \text{if number of samples} \geq 4 \end{cases} \qquad (3.13)$$

Equation 3.12 is evaluated at every zero crossing n and x_{n+1} is the value of the parameter at the (n+1)th zero crossing. The constant of 4 samples between zero crossings was determined by trial and error and it was found that 3 also performs satisfactorily. The multiplier α determines the *memory* of the parameter by regulating the decay. If α was too great the response of the parameter to the actual conditions was too slow. A value of 0.85 was found to be useful for the current application. This method gives a geometric like series and the range of values taken on by x_n extends from 0 when $y_n = 0\ \forall n$, to $\frac{1}{1-\alpha}$ when $y_n = 1\ \forall n$.

The parameter for *coarse* was defined analogously. Although the contour of the parameter was rough it proved valuable, see Figure 3.7. For a sibilant like *s* the *fine* parameter takes on values around 9.

3.6.2 The Devil's Staircase

Traditionally the Devil's staircase was used to describe the weight to the left of a point as it crosses the Cantor Set from left to right, see for example Schroeder [75]. In this thesis the Devil's staircase was used primarily by the author to summarize information about local amplitude maxima, however extra information about spacing between local maxima was also given. The staircase constructed here was not the same as the traditional staircase. It was constructed for this application using,

$$M_I = \sum_{n=1}^{N} \Phi_n \tag{3.14}$$

$$\Phi_n = \begin{cases} max\{|s_n|\,; z_{j-1} < |s_n| \leq z_j\} & \text{if a zero crossing occurred} \\ 0 & \text{otherwise} \end{cases}$$

where to obtain the staircase, the maximum of the absolute value of the speech samples s_n between the zero crossings z_{j-1} and z_j is accumulated when a zero crossing occurs otherwise a 0 is added. The index n is used to index speech samples in the block of size N.

It was found by the author that for different speakers recorded under similar conditions the average slope of the staircase was similar and that the slope for voiced and unvoiced speech was also similar as illustrated in Figure 3.8. Where there is no speech the slope is almost zero and where voicing changes within a word such as around *g* in *wagon*, the slope decreases in the region of change.

A disadvantage of this method was that it was very sensitive to zero (dc) offset of the speech waveform.

3.6.3 Orbit Direction Change Indicator

The rationale behind determining the direction of rotation as the orbit is traced out in phase space is that voiced sounds generally have fairly simple attractors while chaotic attractors tend to display many loops. Unvoiced sounds like *s* and *sh* should display near random phase and hence many direction changes. The method of determining the direction of rotation was effected by taking the previous 2D phase space point and the current phase point and performing a vector cross product. To illustrate the concepts involved, two dimensional phase plots, of the 'i' sound in *fine* and the 's' sound in *juice* are given in Figure 3.9. The orbital structure of the 'i' sound is quite evident while the 's' sound shows a lack of structure.

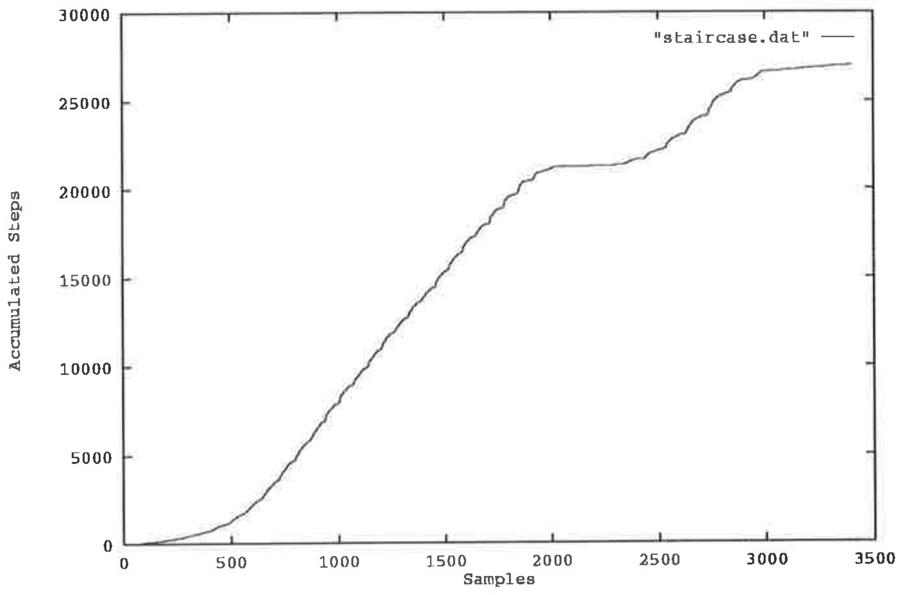


Figure 3.8: The Devil's Staircase for *wagon*.

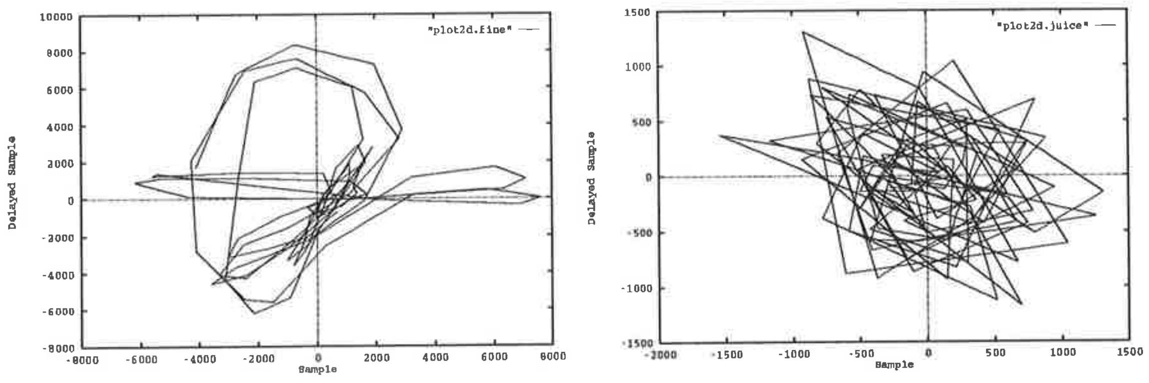


Figure 3.9: Two Dimensional Phase Plots of the sounds: 'i' in fine and 's' in juice.

Let i be the unit vector in the x_0 direction and j be the unit vector in the x_1 direction, where i and j are orthogonal and let k be the unit vector orthogonal to both i and j.

Using n for the time index, the direction of rotation of the phase trajectory can be found from the sign (sgn) of;

$$\begin{aligned} d_n &= sgn[(ix_0(n) + jx_1(n)) \times (ix_0(n-1) + jx_1(n-1))] && (3.15) \\ &= sgn(ky(n)) && (3.16) \end{aligned}$$

where $\times$ is the cross product. This expression was implemented as,

$$d_n = sgn[k \begin{vmatrix} x_0(n) & x_1(n) \\ x_0(n-1) & x_1(n-1) \end{vmatrix}] \qquad (3.17)$$

Let I be an indicator function then the direction change parameter can be given by,

$$D_n = \frac{\sum_{n=0}^{N-1} I(d_n, d_{n-1})}{N} \qquad (3.18)$$

where N is the number of samples in the block and I is defined as;

$$I(d_n, d_{n-1}) = \begin{cases} 1 & d_n \neq d_{n-1} \\ 0 & d_n = d_{n-1} \end{cases} \qquad (3.19)$$

The sign of $ky(n)$ gave the direction of rotation relative to the previous sample. A plot of changes in orbital direction over a window of 100 samples is shown in Figure 3.10. The y-coordinate parameter D_n was determined by counting each direction change and then dividing by the size of the window. The sentence spoken by a female was:

These days a chicken leg is a rare dish.

An interesting observation is that this parameter takes on large values for glottal stops and dentals, generally showing them as distinct peaks. Unfortunately it also takes on large values during other features including some types of voiced sounds making it unsuitable for distinguishing voiced and unvoiced features without further information.

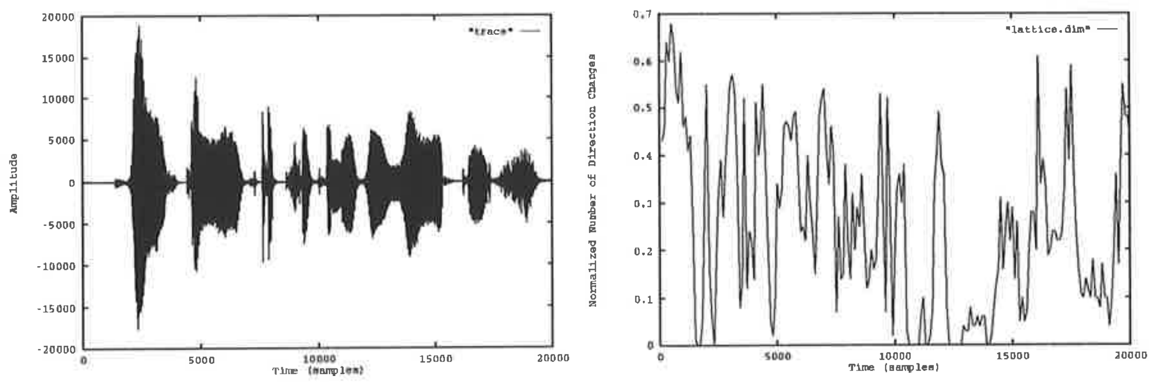


Figure 3.10: Speech Waveform and Normalized Direction Changes.

This new and innovative indicator is useful in classifying speech and empirical evidence over the utterances of two male and two female speakers indicate that values ≥ 0.45 for the direction change parameter (D_n) Figure 3.10 is a reasonable threshold; giving one parameter for an unvoiced speech hypothesis.

3.7 Voicing Estimator

One of the key features in a pitch estimator is to determine when speech is voiced and when it is unvoiced. One of the classic voicing indicators is the rms power of the speech within a window. Used on its own this parameter performs quite well, but it does classify sounds such as *sh* as voiced, or if the threshold is increased it fails to indicate softly voiced speech.

It was discovered that the direction change parameter was useful in determining unvoiced speech which is generally characterised by a high direction change parameter and low rms power. However sounds such as *sh* were not classified correctly. The decorrelation time of the autocorrelation was empirically determined to be an effective parameter in labelling unvoiced speech however a slight modification was necessary. If the speech was not decorrelated by nine integral sample shifts the decorrelation time was taken to be one, that is unvoiced. The reason for this correction was that general speech was found to almost always have shorter decorrelation times than nine samples, however low frequency *hum* sometimes found as low level background noise has a longer decorrelation time.

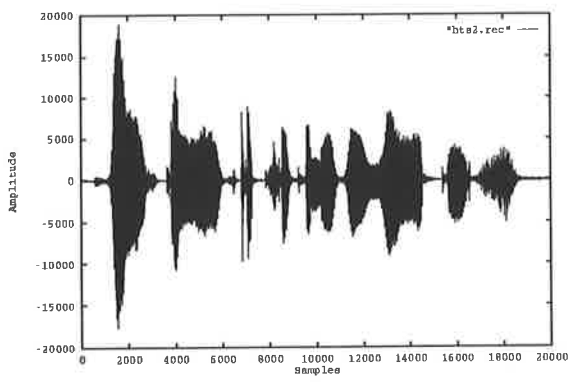
A first parse of a rule set for *not voiced* may be: if the rms power was below 600 units, and/or the decorrelation time was one sample then the speech was unvoiced. Units were measured as 1 quantisation level when the recorded speech was quantised to 16 bits such that no overload distortion occurred.

A more sophisticated voicing decision was based on a rule set that was determined by trial and error over a large number of trial runs over the speech data base. The rule set was divided into two main parts: the first independent of previous voicing estimates and the second depending on whether the previous estimate indicated voicing or non-voiced. The rule set can be summarized as:

1. If the rms power level was below 600 then classify as unvoiced,
2. If the fine structure variable was above 7.5 then classify as unvoiced,
3. If the fine structure variable was above 6.8 and the rms power less than 900 then classify as unvoiced,
4. If the decorrelation time was one sample then classify as unvoiced,
5. If the previous voicing estimate was *unvoiced*:
    (a) If rms power was less than 2600 and the number of valleys found by the AMDF was greater than or equal to 4, classify as unvoiced;
    (b) If rms power was less than 2600 and the fine structure variable was greater than 5, classify as unvoiced;
    (c) If rms power was less than 3500 and the number of valleys found by the AMDF was greater than 5, classify as unvoiced.

Rules using the slope of the Devil's Staircase were initially included to good effect, however the speech used for the work was recorded under excellent conditions and showed no dc bias on the waveform. In the case of field recording this could not be guaranteed so the Devil's Staircase construction could not be deemed to be accurate as it is sensitive to zero crossings and dc bias. In the interest of making the voiced/unvoiced indicator robust only the fine structure parameter was used as it was less sensitive to dc bias, particularly when a dead zone of 1000 ($\pm$500) quantisation units around zero was incorporated.

All thresholds were determined empirically from utterances from two females and two males as described earlier. Sixteen bit sampling was used to give an amplitude range of



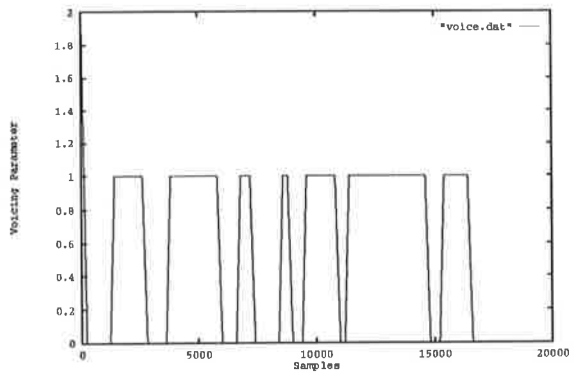


Figure 3.11: Speech Waveform and Voicing Indicator.

± 32000 although the recorded amplitude rarely exceeded 20000. A high rms threshold together with the autocorrelation decorrelation value of 1 indicated sounds like *sh* as unvoiced, whereas if the high threshold was solely used, low level voicing would be indicated as unvoiced.

A typical voicing indication is shown in Figure 3.11, for the sentence, " *These days a chicken leg is a rare dish.*", uttered by a female.

3.8 Pitch Tracking

Ideally for a perfectly periodic signal each pitch estimate will be correct and for the AMDF each peak would be distinct and identical. However in practice because of the time varying nature of speech the pitch estimate will not always be correct in that multiples or fractions of the pitch period may be chosen. When the first formant[5] in the speech segment is large and close to the pitch period beat effects can occur, which affect the pitch estimate.

Pitch tracking in the algorithm was done in conjunction with the coarse AMDF pitch estimator prior to handing the estimate to the fine resolution estimator. The main function of pitch tracking was to eliminate gross pitch errors such as pitch halving, doubling or tripling. For continuous voicing a smooth pitch contour should result.

Tracking the pitch depended on the threshold set for the AMDF (Equation 3.10) and

[5]The spectral envelope of speech shows resonant frequencies, the nominal center frequencies of these resonant peaks are known as formant frequencies.

also the pitch estimate from the median filter that tracked the last five nonzero estimates. The median filter used a moving window of the last five non-zero pitch estimates which, if the pitch estimates were treated as noisy, returned an estimate that minimized the absolute error of estimation.

If the absolute error is denoted A_e then;

$$\begin{aligned} A_e &= E[|x-c|] && (3.20)\\ &= \int_{-\infty}^{c}(c-x)f(x)\,dx + \int_{c}^{\infty}(x-c)f(x)\,dx && (3.21)\end{aligned}$$

where $f(x)$ is the probability density function of the pitch estimates x. An estimate c is to be made such that the absolute error is minimized;

$$\begin{aligned} \frac{\partial A_e}{\partial c} &= \int_{-\infty}^{c} f(x)\,dx - \int_{c}^{\infty} f(x)\,dx && (3.22)\\ &= 2F(c)-1 && (3.23)\\ &= 0 && (3.24)\end{aligned}$$

The median is thus the value where the cumulative distribution $F(x) = 1/2$. This is implemented by ranking the values in magnitude and taking the middle element as the estimate.

The first estimate of the pitch was given by the ordinate at the minimum of the AMDF. The pitch estimate and the AMDF minimum value were checked against the estimate from the median filter and the AMDF threshold. If the AMDF minimum value was below the threshold and the pitch estimate is within 30% of the median, the estimate was accepted. If it failed these conditions, an auxiliary estimate of the pitch was made. Auxiliary estimates were determined by searching the AMDF for a local minimum in a window of $\pm 30\%$ of the previous pitch estimate. If the AMDF value at this auxiliary ordinate was greater than twice the value obtained for the original estimate the auxiliary estimate was immediately labelled as unlikely, although this was a rare event. This scheme effectively eliminated almost all cases of pitch doubling, tripling and frequency doubling and tripling.

In the rare event where no decision could be reached, the original and auxiliary estimates were compared to the median filter's estimate and the median filter's estimate slightly modified by 10% up or down depending on the closer of auxiliary or original estimates. The modified median filter's estimate was then taken as the coarse estimate.

The estimated pitch always had to lie within 30% of the last estimated pitch period within the duration of a voiced segment. If the fine resolution pitch search could not find the true minimum within 25% of the last estimate an unvoiced decision was returned.

3.9 The Pitch Estimation Algorithm

The pitch estimation algorithm consisted of the following units:

Low Pass Filter The incoming speech was filtered with a low pass filter with a cut-off frequency of 800 Hz.

Parameters Calculate the autocorrelation function and find the decorrelation time as described above. This also gave access to rms power. Calculate the direction change parameter and the fine structure parameter.

Voicing If the section of speech was considered unvoiced, (see the Voicing Estimator section above), the voicing parameter was set to 0 and the pitch estimate was returned as 0, else if *voiced* the voicing parameter was set to 1 and the pitch determined.

AMDF Search An AMDF search was undertaken in the number of dimensions chosen (typically 3) and returned an integral sample estimate of the pitch period. Often the estimate would be within a sample or two of the final pitch estimate, but not always. The search window can be reduced by making use of the fact that the pitch variation almost never exceeds 25%, [76] although this restriction was lifted if the previous estimate was 0, ie. unvoiced.

Section Search The estimate from the AMDF search was used to give a window to search for the non-integral pitch estimate. The window was 15% above and below the AMDF estimate. A golden-mean section search was used to find the estimate although a gradient technique could have been used especially for higher dimensions. The well known Lagrange interpolation formula (second order) was used to interpolate sample values. (See equation 3.28).

Pitch Tracking The pitch estimate, if the last section was voiced, was fed into the median filter to assist pitch tracking.

The overall algorithm is as follows:

1. Input the data, if the last speech segment was unvoiced, input 10 new samples else, one third of the previous pitch period in new samples. Low pass filter the samples.
2. Calculate the fine structure parameter using the unfiltered samples and use the filtered samples for all steps below.
3. Calculate the autocorrelation function for 10 delays to determine the decorrelation delay and the rms power.
4. Determine the orbit direction change parameter.
5. Evaluate the AMDF.
6. Perform the voicing evaluation using the rule base given in Section 3.7.
7. Perform peak picking on the AMDF; this consists of:
   (a) If one peak with value less than 3000 accept the estimate.
   (b) Check the absolute minimum and the number of detected valleys. If estimate is within 25% of the previous estimate release estimate.
   (c) If multiple valleys check for harmonic structure. If a structure consistent with 25% of previous estimate is found between any valleys, return the pitch period that displays multiples.
   (d) If no consistent structure can be found return an estimate of 0.
8. Pass estimate to the fine resolution section search. The returned value must be within 15% of the value handed in and 25% of the previous estimate, if not, another search is attempted using the *running median* from the median filter. If still not within specification, class the segment as unvoiced.
9. Feed estimate into the median filter if the segment was voiced and return the estimate to the user. If the segment was classed unvoiced return 0 to the user.

10. Goto Step 1.

3.10 Results from the Pitch Estimator

Testing a pitch estimator of this kind without using synthetic speech is difficult. Utterances of different isolated words from four speakers, two male and two female, were chosen to test the estimator. Manual measurements of the pitch (described below) throughout each word were made as a guide to judge if the pitch estimator was operating correctly.

A trial run of the estimator was performed using up to six phase space dimensions. The experiment was run for four words, uttered by one of two females and two males, showing definite voicing. The pitch estimator was configured to work with one through to six phase space dimensions and the estimates were produced every 200 samples. Mean estimates were determined from the six estimators for each 200 sample block and then deviations were determined and averaged for each uttered word. The same statistics were determined for three to six dimensions and the results compared with the previous data. (See Table 3.1). Values in the tables are given in samples. The results indicated that using more than three dimensions was not justified as similar values were obtained for higher dimensions which required more processing.

The AMDF proved to be more robust with respect to noise and consistency of estimates in three dimensions than lower dimensions but showed no further improvement beyond three dimensions. The section search also showed little improvement above three dimensions. It was found that an L_∞ norm performed slightly better than the sum of magnitude referred to previously for the multidimensional AMDF. The L_∞ norm is given by,

$$\max_i(x_i - y_i) \quad \text{for } i = 1, \ldots, N \tag{3.25}$$

for two arbitrary N dimensional points.

Tests were performed for three dimensions with and without lowpass filtered data files, and the major difference found was the smoothness of the final pitch contour.

To further increase the robustness (to avoid the 7/8 pitch frequency sometimes present) a variation was used in the calculation of the inter–pitch distance function. Four different *phases*, relative to the pitch period, and lengths of speech segments were used

word	standard dev. for dimensions 1–6	standard dev. for dimensions 3–6
wagon	6.23	0.18
purse	0.42	0.18
droopy	0.54	0.04
fine	0.21	0.02

Table 3.1: Average Standard Deviations on Pitch Estimates for Different Dimensionality: for speech sampled at 8 kHz.

in calculating the distance. Each phase was started at a predetermined fixed point in the pitch period and only every fourth multidimensional point taken in the distance calculation. The four phases were staggered so that most of the multidimensional points in a pitch period were accounted for. The four distances are then summed and averaged. The true pitch was characterized by a high correlation for any segment within consecutive pitch periods. This argument extends to different length segments, as a sample segment a little longer than a pitch period should still find a strong correlation to the previous pitch period.

A comparison of the performance of the multidimensional AMDF the super resolution estimators were made against manually determined pitch estimates. Manual determination of pitch was made by measuring the distance between prominent features on successive pitch periods. A minimum of three inter pitch measurements had to agree before it was deemed that the correct pitch was determined. The resolution of measurement was to 1/2 a millimeter when the speech waveform was displayed such that 1mm represented 1.2 samples. An example section of speech (the voiced part of the word juice) is shown in Figure 3.12. The closeness of the estimator to the visually determined pitch period (the points in the figure) is evident. The high resolution pitch contour has a smoother pitch track than the AMDF.

Figure 3.13 shows the speech waveform for the word *wagon* spoken by a male speaker together with the pitch contour. It should be noted that the pitch estimates were formed only for the last pitch period in a 200 sample window, as that was the criterion required for the packet reconstruction problem. Waveform reconstruction only requires the last pitch period in the packet in order to generate the next packet of identical pitch periods. Waveform reconstruction and substitution are covered in some detail in the following sections. This gives the pitch contour a piecewise linear appearance.

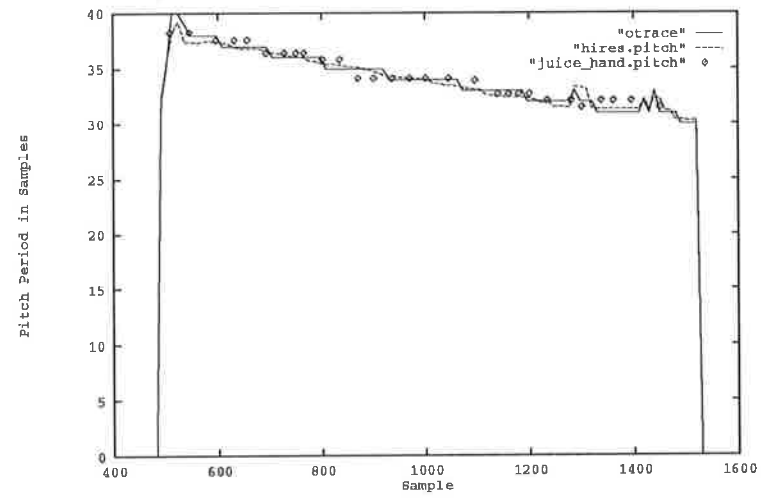


Figure 3.12: Pitch Contour for *juice*.

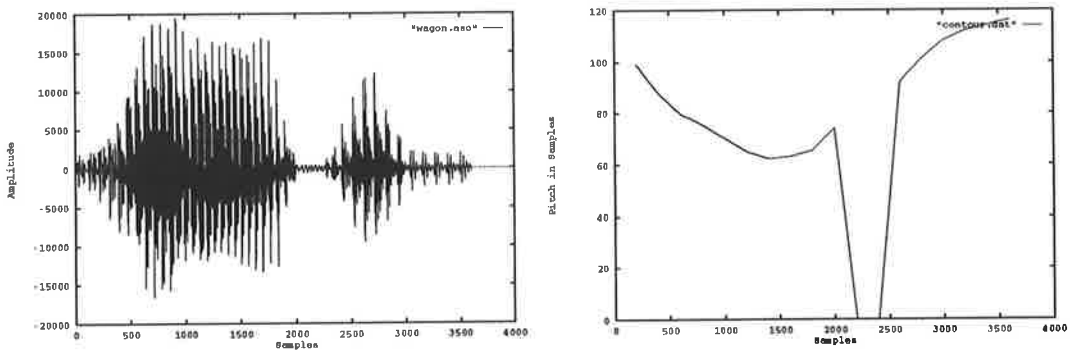


Figure 3.13: The Speech Waveform and Pitch Contour for *wagon*.

Various metrics for measuring waveform *closeness* were investigated for the fine resolution section search including; mean square error, mean absolute error and correlation. All methods produced similar but not exactly the same results. The pitch estimate is a statistical estimate between two consecutive pitch periods. In effect two slightly different pitch periods are cross correlated and as such an accuracy of a tenth of a sampling period seemed reasonable. To obtain greater accuracy, interpolation (compression or expansion) by a small amount of one of the pitch periods should be performed followed by the cross correlation process. This would of necessity increase the computational load.

Medan et al[76] imply that greater resolution is possible in their technique, however the author has reservations as two pitch periods of possibly different periods are cross correlated. It should be pointed out that the Medan method is very good and has less computational load than the author's method. The final formulation is similar in many respects to the Medan method.

For an accuracy of a tenth of a sampling period in the technique developed here an average of approximately seven iterations for finding the fractional part of a sample was required in the Poincaré section search.

Both the scheme presented and Medan's scheme for finding pitch periods have a robustness to noise as the entire pitch period is used in the estimation process. Both methods give substantial improvement over integer pitch period estimation however it would appear that Medan's scheme is superior when computational load is also considered.

The speech waveform of the female utterance *The juice of lemons makes fine punch* is shown in Figure 3.14. This waveform was analysed using the authors algorithm and pitch tracks plotted for the AMDF (Figure 3.15) and for the super resolution algorithm (Figure 3.16). The discrete nature is quite evident in the pitch track of the AMDF as it only made integer estimates. The smoother pitch track can be seen in Figure 3.16 although there are some locations where the pitch was not smooth. Visual observation of these locations (see for example samples around 17300 in Figure 3.16) show a large change in the speech waveform shape and amplitude which corresponded to a voiced/unvoiced transition. The pitch period by visual techniques indicate that approximately 48 samples would be a better estimate. The amplitude variations were adjusted so that the rms power of the segments being correlated matched. This was however not adequate. A possible solution to this problem may be to correlate backwards and for-

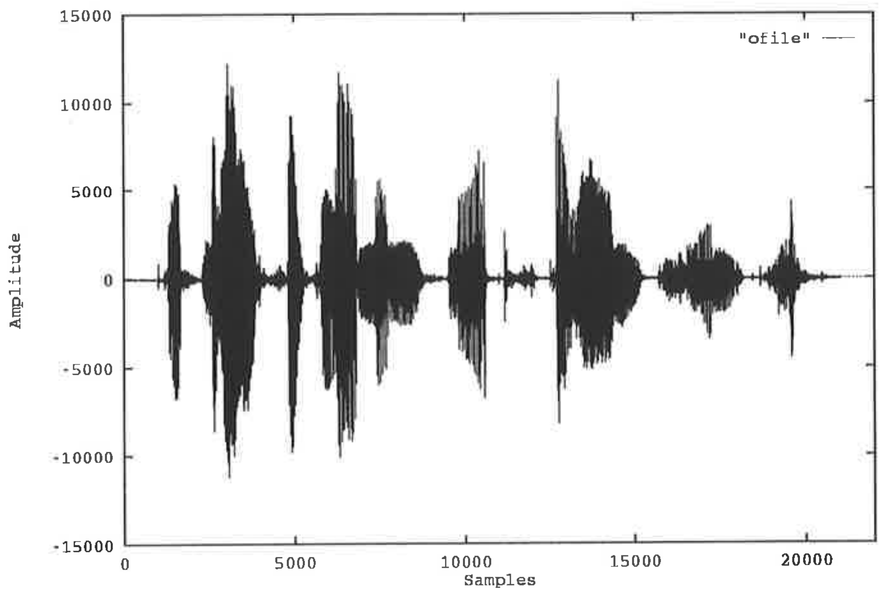


Figure 3.14: Speech Waveform the Female Utterance: *The juice of lemons makes fine punch.*

wards (the method used in this chapter only correlated with past samples). This would allow a test for consistency with pitch periods either side of the one being tested. Not withstanding the occasional error the super resolution technique gives a better pitch track.

Figures 3.17 and 3.18 show pitch contours for larger segments of speech spoken by a male and a female where the pitch estimate is the last pitch period estimate in the next block of 200 samples. This information is required for frame substitution methods which are discussed in the following section.

3.11 Waveform Substitution

Sometimes in the transmission of encoded speech a frame is received with too many errors in order for the waveform to be constructed from the parameters within the frame. In these cases a possible method of reconstruction for the frame is to use information obtained from the last correctly received packet. An assumption was made that the frame following the *lost*[6] one was not available. If it was, an additional delay of

[6]A *lost* frame is one that could not, due to corrupted parameters, be reconstructed.

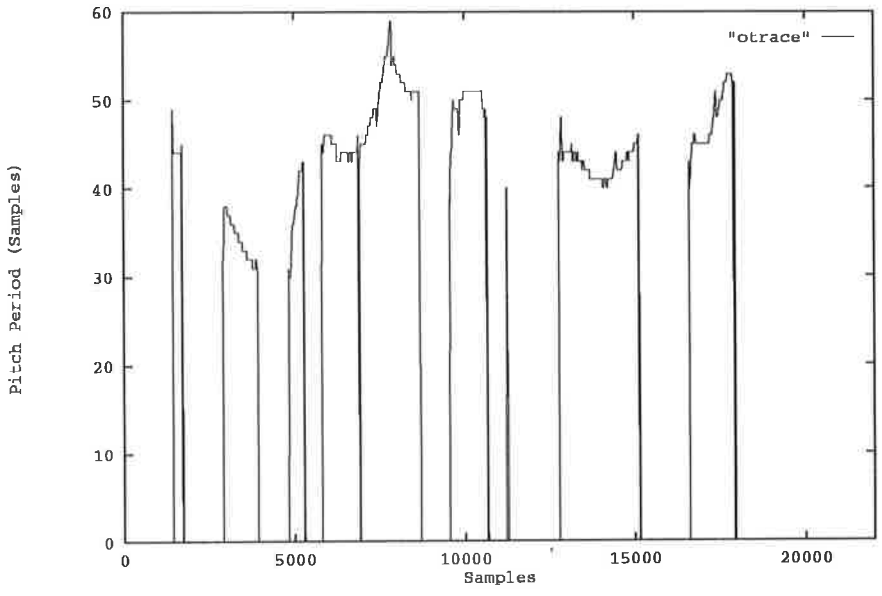


Figure 3.15: AMDF Pitch Contour.

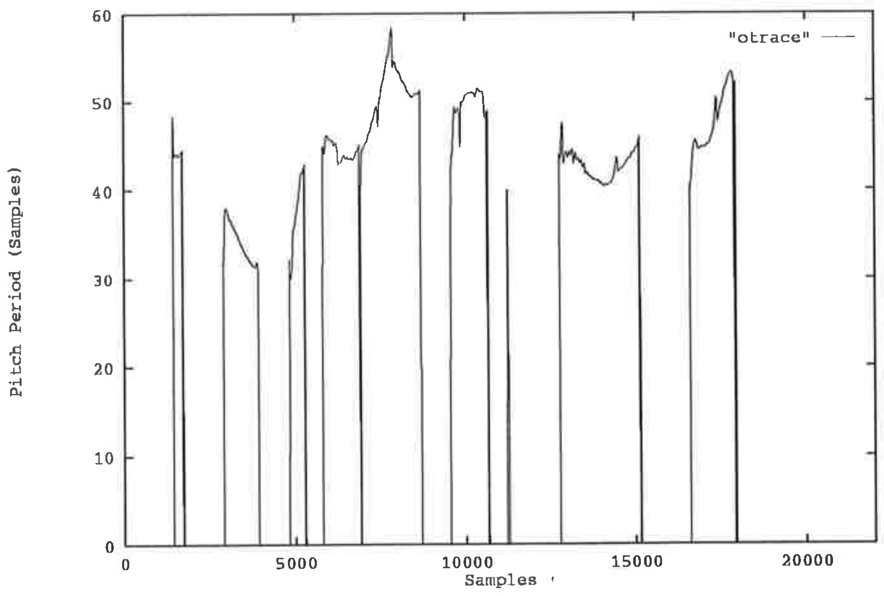


Figure 3.16: Super-resolution Pitch Contour.

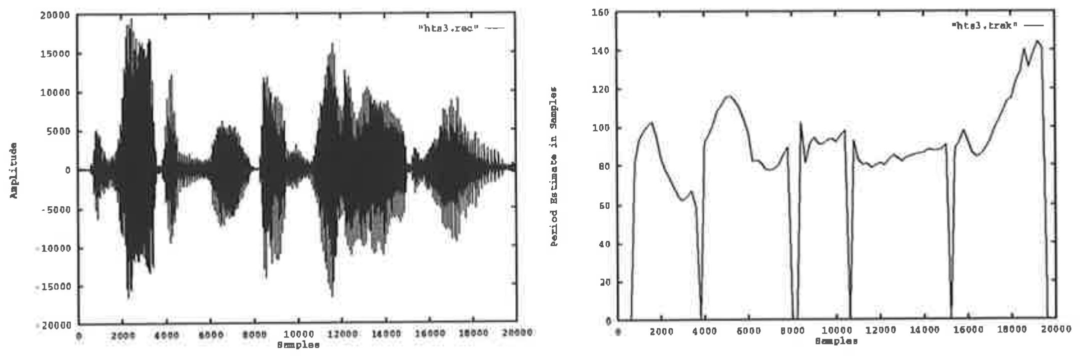


Figure 3.17: Speech Waveform and Pitch Contour for the Male Utterance: *The wagon moved on well oiled wheels.*

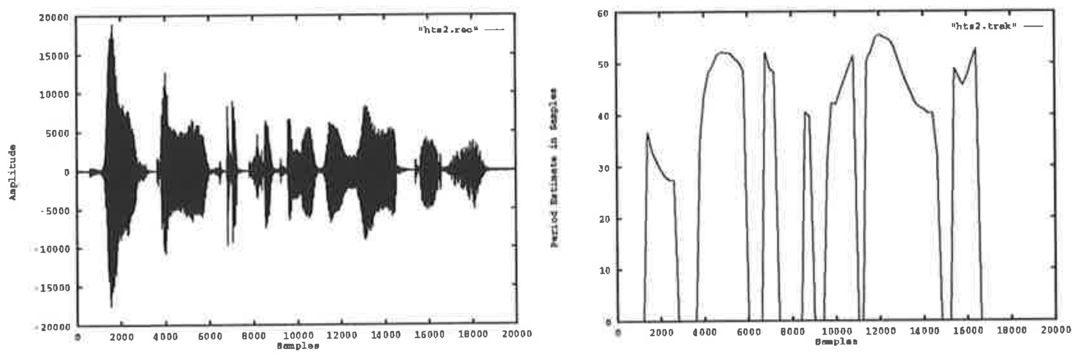


Figure 3.18: Speech Waveform and Pitch Contour for the Female Utterance: *These days a chicken leg is a rare dish.*

another frame must reckoned into the system. However if the next frame was available, an interpolation between the two frames bordering the *lost* could be made. Due to delay constraints only the previous frame is available. The restriction required that a frame was received as good or bad and substitution effected prior to the reception of the next frame.

If the previous packet was unvoiced, no special relationship holds with regard to pitch and thus phase of the waveform. A complete replication of the previous frame of speech would appear to be reasonable.

If the previous frame was voiced, a possible method of substitution is to replicate the last pitch period as it is the latest information available about the state of the speech. The pitches are phase aligned at the boundary to ensure a smooth transition.

In both substitution cases the amplitude has to be interpolated to the following correctly received frame to avoid discontinuities in the amplitude envelope. For the voiced speech substitution an additional smoothing of the pitch period into the following correctly received frame is necessary. This is done using a sample interpolation method.

As indicated in an earlier section, the substitution scheme developed in this section was based on pitch replication for voiced speech. A frame in this context was 25 ms of speech or 200 samples. The pitch estimation mechanism used was the system described above together with the auxiliary functions to determine voicing.

A discussion follows covering: the replication of the last correctly received pitch period for voiced frame substitution, substitution for unvoiced frames and the interpolation technique for splicing together a substituted voiced frame followed by another voiced frame.

3.11.1 Replication of the Last Pitch Period

In principle replication of the last pitch period was quite straight forward. The pitch estimate determined the number of samples N to be stepped back into the previous frame. The N samples were then repeated until the frame was filled. Any extra samples from the N samples in excess from filling the frame were retained. Within any frame the rms power level can change substantially. To remedy this shortcoming, the rms power level of the substituted frame was determined and the following received frame's rms power level was also determined. The levels of the newly received frame were adjusted using a raised cosine function over the range R of the first 120 samples of the new frame.

$$w_f(i) = \tfrac{1-cos[(i\pi)/R]}{2} P_r \quad \text{for } 0 \leq i < R \tag{3.26}$$

where $w_f(i)$ is the ith weighting factor and P_r is a power ratio determined from the difference in rms power levels between the substituted frame and the rms level of the new frame.

This implied that the frame following the substituted frame had its power level adjusted to ensure that there was no unusual jump in the speech envelope. If the frame following the substituted frame was unvoiced or silent the pitch period was extended and weighted by a decreasing exponential to almost silence before the following frame's waveform was used. The weighting function used in this case was given by,

$$w_f(i) = \tfrac{1+cos[(i\pi)/R]}{2} e^{(-i/600)} P_d \quad \text{for } 0 \leq i < R \tag{3.27}$$

where P_d is the difference in rms levels. This function gives a beginning and ending *tail* similar to the raised cosine but the transition in the middle is slower.

3.11.2 Unvoiced Substitution

Unvoiced substitution simply repeated the entire last frame. The possibilities for the following frame are, silence, more unvoiced, or voiced. In the case of silence or more unvoiced speech the rms power measure of the frames were used to give the weighting factor for an increasing or decreasing raised cosine function over 0.6 of a frame length (for a 20ms frame) for a smoother amplitude transition. The more demanding situation was when the good frame following a substitution frame was voiced. In this event the difference in the power levels between the frames was generally considerable. A raised cosine function was also used here to handle the transition. Fortunately the levels of unvoiced speech tended to be considerably lower than voiced speech so the small weighting factors at the start of the raised cosine compress the wave significantly, reducing the phase problem of beginning somewhere in the middle of a pitch period.

3.11.3 Splicing

The term splicing is used to describe the process of joining the new frame to the substituted frame so that there is no phase discrepancy at the joint with respect to the pitch period.

If the substituted frame was unvoiced section and this was followed by a voiced frame the joining of the frames only required amplitude adjustment, which was covered in an earlier section. The event of a voiced speech substitution followed by an unvoiced frame also presented little problem and was covered previously.

When a voiced frame was substituted and a voiced frame followed, the amplitude envelope was adjusted using the raised cosine function, but the phase of the pitch period may not have agreed. When a voiced substitution occurred a further 100 samples of substitution was kept aside for processing of the next frame. On arrival of the next good frame the amplitude of the packet was normalized to that of the previous substituted frame and a correlation performed with the extra 100 samples kept aside. The maximum of the cross correlation was noted and denoted the correlation-index. If the correlation-index was less than 1/2 of the current pitch estimate, samples were dropped from the received frame and the packet waveform was interpolated to make up the correct number of samples. The interpolation was performed using a second order form of the well known Lagrangian interpolation formula.

$$\begin{aligned} y(n) = {} & \frac{(n-n_2)(n-n_3)\ldots(n-n_N)}{(n_1-n_2)(n_1-n_3)\ldots(n_1-n_N)}y(n_1) + \\ & \frac{(n-n_1)(n-n_3)\ldots(n-n_N)}{(n_2-n_1)(n_2-n_3)\ldots(n_2-n_N)}y(n_2) + \\ & \vdots + \\ & \frac{(n-n_1)(n-n_2)\ldots(n-n_N)}{(n_N-n_1)(n_N-n_2)\ldots(n_N-n_{N-1})}y(n_N) \end{aligned} \qquad (3.28)$$

If the correlation-index was greater than 1/2 the estimated pitch period an extra fraction of a pitch period was substituted and the waveform of the following frame was compressed over 150 samples to ensure the correct number of samples by the completion of the frame.

In practice it was found that the adjustment was rarely more than 5 samples and very rarely more than 10 samples. The compression case occurred almost exclusively in the higher pitched speaker's case where the compression occurred over a few pitch periods and caused little pitch distortion. The expansion of the pitch period in the lower pitched speakers tended to be a small percentage change in the pitch.

3.11.4 Memory Considerations for CELP Coders

The major problem with the packet substitution scheme in the CELP scenario was the memory of the CELP coder structure. In experiments performed by the author and others [51], it was found that a simple repetition of the previously received error free parameter frame followed by the next good frame produced in most cases an acceptable result. The memory of the CELP coder performed an interpolation from one frame into the next with a satisfactory performance. The above technique though more *accurate* was left with the problem of how to update the memory elements; but more importantly, the computational overhead was not warranted. Perceptually CELP parameter substitution caused minimal audible distortion.

For the general CELP case it is recommended to repeat the previous frame's parameters for the substitution process and then follow on with the next error free frame. This does not however rule out the method developed above for methods that can afford the extra processing or have little to no memory from one block to the next.

3.12 Conclusion

A new reliable and robust super resolution pitch estimator, which has promising performance and yields an accuracy to 1/10 of a sample was discussed. As the multidimensional phase space representation was obtained by simple delays, the computational load is not great although greater reductions in processing would be possible using decimation of the low pass filtered speech waveform. It was interesting to note that a large number of dimensions in representing the speech did not significantly improve performance. One of the greatest gains found was the use of three dimensions of phase space for the AMDF which improved its reliability and accuracy of its estimates.

Two new speech waveform classification methods based on zero crossings were presented. The Devil's staircase despite giving useful information was deemed not robust enough for the general speech coder environment.

The new voicing indicators based on direction of travel in the two dimensional phase plane and orbit and the autocorrelation function's decorrelation shift were robust and used for voicing indication. The decorrelation time (shift) was determined for estimating the time shift required for the multidimensional phase space representation of the speech and as such it was not necessary to determine it separately. A short decorrelation time indicated an essentially uncorrelated signal. The orbit direction parameter was determined from a cross product of the vector from the origin to the current two dimensional representation of the speech and the vector from the origin to the previous two dimensional speech sample. A large number of orbit direction parameters of the same sign indicates a simple orbit, such as a limit cycle, generally associated with voiced speech. This parameter proved valuable in classifying the speech waveform. Further investigation of this parameter for the analytic speech signal may prove fruitful.

A frame substitution methodology was developed which provides a technique of error recovery in the extreme event of catastrophic[7] loss of information within a frame.

It is important to note that if the frame substitution is to be used with CELP type coders the memory effects of the long term predictor must be taken into account. If the vocoder has no long term memory the scheme described above can be used, however if the substitution occurs for more than one contiguous block it may be desirable to substitute spectrally coloured noise or gradually add in noise to avoid an unnatural sustained *tone*. This approach is also useful for the CELP vocoder structure where successive substitutions can be made with successively *flatter* spectral envelopes.

[7]By catastrophic it is meant that there is not enough information left to reconstruct the encoded waveform.

Chapter 4

Vector Quantiser and the Missing Data Problem

4.1 Introduction

In many low rate speech coders use is made of the Line Spectral Pairs (LSP) representation of the poles in the all pole model of Linear Predictive Coding (LPC) of speech. LSP's are a projection of the poles (a_i's) of the LPC model onto the unit circle, and thus have units of radians. LSP's have the property that they uniquely, to within a gain factor, specify the spectral shape of the speech waveform segment; and they are *monotonically increasing.* Soong and Juang [30] discuss the details of LSP's. If the LSP's are quantised to the U.S Federal Standard 1016 for CELP speech coders [1] [3], they are arranged to be at least 15 Hz apart. This together with specific one dimensional quantisers designed for each individual LSP component provides good performance in encoding, in terms of distortion of the reproduced speech.

A problem arises when a block of encoded speech is transmitted and a channel error causes an incorrect reconstruction of one of the LSP values. Let $\omega = \{\omega_0, \omega_1, \ldots \omega_{p-1}\}$ represent the LSP components. Note that the quantisation range of ω_{i-1} partially overlaps the quantisation range of ω_i so that an incorrectly received index may swap the ordering of the reconstructed LSP values. If on reception the values of the reconstructed LSP's are no longer in order it is deduced that a channel error/s have caused the non-monotonic sequence. Traditionally the solution has been to repeat the spectral shape from the last successfully received block or sort the LSP's into ascending order. This issue will be covered in more detail later.

It is proposed that this problem is similar to the classical missing data problem, see for example Krzanowski [77]. From statistical analysis, if the LSP components were completely uncorrelated it would be possible to maintain an average of each LSP component and when a component was found missing, substitute the average. This may not always result in a valid datum as the components must be monotonically increasing. A calculation of the covariance matrix (Equation 4.1) for the LSP components treated as a vector, demonstrates that considerable cross correlation exists between vector components; the off diagonal components are substantial. It should be noted that this matrix is not a covariance matrix in the conventional sense but will be called a *covariance* matrix for ease of discussion. This implies the use of conditional probabilities, conditional distributions and/or regression analysis. Let

$$C(x) = \begin{pmatrix} E[(x_1 - m_1)^2] & E[(x_1 - m_1)(x_2 - m_2)] & \cdots & E[(x_1 - m_1)(x_N - m_N)] \\ E[(x_2 - m_2)(x_1 - m_1)] & E[(x_2 - m_2)^2] & \cdots & E[(x_2 - m_2)(x_N - m_N)] \\ \vdots & \vdots & & \vdots \\ E[(x_N - m_N)(x_1 - m_1)] & E[(x_N - m_N)(x_2 - m_2)] & \cdots & E[(x_N - m_N)^2] \end{pmatrix} \tag{4.1}$$

where $E[.]$ denotes mathematical expectation, m_i denotes the mean of the ith vector component and N is the dimension of the vector.

Noting that the joint probability function is dependent, one may write

$$\begin{aligned} E\left[g(y)|x = t\right] &= \int_{-\infty}^{\infty} g(y) f_y(y|x = t)dy && (4.2) \\ &= \frac{\int_{-\infty}^{\infty} g(y) f_{xy}(x, y)dy}{\int_{-\infty}^{\infty} f_{xy}(x, y)dy} && (4.3) \end{aligned}$$

where $f_{x,y}$ is the joint density function. Clearly the components of the vector that are in the ascending monotonic sequence will assist in determining the erroneous components analogous to the g(**y**) in the above equation. If the term **y** was used for the missing vector components instead of $g(\cdot)$ in the above expression, then

$$E\left[y|x = t\right] \tag{4.4}$$

is the center of mass for that part of the joint distribution that corresponds to $x = t$; this is the best mean square estimate given $x = t$.

Calculation of these estimates imply that the joint distribution for the source is known. In general this is not known and may indeed vary from speaker to speaker. The problem then is to estimate the distribution on-line from the incoming frames, as the transmission of statistical data is to be avoided.

It is well known that vector quantisation can be used to describe data sets in the sense of classification. Vector quantisation can utilize linear and non-linear correlations between vector components in constructing the quantiser set.

4.2 Why Use Vector Quantisation?

Shannon's work predicted that using a block quantiser it was possible to perform source encoding approaching the rate-distortion bound and is certainly better than scalar quantisation. The lack of good design techniques for vector quantisers until the presentation of the now well known k-means [78] algorithm hindered use of vector quantisers. The k-means algorithm is also now known as the generalised Lloyd algorithm. The k-means algorithm also has problems in that complexity grows exponentially with codebook size.

A vector quantiser must have a codebook associated with it that uniquely describes nearest neighbour cells called Voronoi regions in $\Re^n$ space. The Voronoi regions are defined by a nearest neighbour rule to a *representative* point in the Voronoi region such that if any arbitrary point falling within the Voronoi region is represented by the *representative* point a minimum amount of distortion results. The Voronoi regions fill the space that they are embedded in. If the distortion measure is the mean square error (the distortion measure used for this work), then the optimum *representative* point is the standard centroid found by,

$$\overline{y_j} = \frac{\sum_{i=1}^{M} x_i S_j(x_i)}{\sum_{i=1}^{M} S_j(x_i)} \tag{4.5}$$

where the Voronoi region membership M is assumed for Voronoi region R_j and,

$$S_j(x_i) = \begin{cases} 1 & \text{if } x_i \in R_j \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

Denote the distortion measure $d(.,.)$, then x is in Voronoi region R_j when,

$$R_j \subset \{x : d(x, y_j) \leq d(x, y_i) \, \forall \, i \neq j\} \tag{4.7}$$

If $Q(\cdot)$ represents the quantisation process then an arbitrary point is represented by y_i by the rule,

$$\begin{array}{ll} Q(x) = y_j & \text{iff } x \in R_j \quad \text{i.e.} \\ d(x, y_j) \leq d(x, y_i) & \text{for all } i \neq j \end{array} \tag{4.8}$$

These relationships have consequences that have bearing on the following discussions, they are:-

1. Each y_i is uniquely associated with and defines a Voronoi region R_i;
2. $R_i \cap R_j = \emptyset$ for all $i \neq j$, that is, none of the Voronoi regions intersect;
3. A codebook is an enumerated list of y_i's that represent the minimum distortion quantisation for the range of $\mathbf{x}$'s in R_i;
4. Each Voronoi region is a convex polytope, due to the nearest neighbour rule;
5. If the random $\mathbf{x}$ vectors are assumed to be from Euclidean space $\Re^k$, the faces of the polytopes are k-1 dimensional hyperplanes.

Vector quantisers naturally exploit any linear correlation between vector components without the necessity of scaling and rotation of coordinate axes if using scalar quantisation. They also exploit non-linear dependencies by only placing Voronoi regions where they are required for the quantisation process. As the polytopes associated with the Voronoi regions are space filling they naturally find an optimum packing in N-space. These polytopes normally pack much more efficiently than corresponding N-cubes, resulting from scalar quantisation, so that a quantisation gain is achieved. A vector quantiser may also exploit the fact that the probability density function is not uniform and tailor the number of cells it packs into a region to minimize encoding distortion.

For more details on vector quantisation see Gersho and Gray [33] Chapters 10 and 11.

4.2.1 Generation of a Vector Quantiser Using a Training Set

Ideally to design a vector quantiser the distribution should be known. In general this is not the case and use must be made of a clustering algorithm. The most commonly used algorithm is the one described by Linde, Buzo and Gray [78] which is a k-means type of algorithm. A major issue in codebook design is the complexity of the design and quantisation implementation forced on the user by the randomness of the design data. No natural structure arises in the output vectors y_i's and so a complete codebook search to find the minimum distortion output vector is necessary. Various schemes have been found to improve the speed of operation of the encoding procedure but they are all inferior to the *full-search* codebook. Gersho and Gray cover a range of techniques in their book [33]. Makhoul, Roucos and Gish [79] also provide an overview of different codebook design methods.

Following the argument of Gersho [80], some comments can be made about the structure of random codebooks generated by clustering algorithms such as the k-means algorithm.

Let the training set TS be given by,

$$TS = \{x_i\} \quad i = 1, \ldots, M \tag{4.9}$$

for M random vectors, and assume that $M \gg N$, where N is the number of output vectors in the codebook. The clustering algorithm determines the N output vectors.

The quantiser Q is called optimal *with respect to* TS if the following two conditions hold:

Condition 1

$$\overline{y_j} = \frac{\sum_{i=1}^{M} x_i S_j(x_i)}{\sum_{i=1}^{M} S_j(x_i)} \tag{4.10}$$

Condition 2

$$\|Q(x_i) - x_i\| \leq \|y_j - x_i\| \quad \forall j \neq i \quad i = 1, 2, \ldots, M \quad j = 1, 2, \ldots, N \tag{4.11}$$

Condition 1 was mentioned earlier and implies that y_j should be the centroid of all training vectors x_i associated with the set R_j and condition 2 implies that all $x_i \in R_j$ define the Voronoi cell. Voronoi cells that fill the quantisation space are defined by the output set $Y = \{y_1, y_2, y_3, \ldots, y_N\}$.

An algorithm that produces a locally optimal partition of $\Re^k$ space will generate a random codebook Y given a random source. This will almost surely be the case if speech source is used for the training set. Gersho states two conditions of disorder (repeated here for convenience).

Weak Condition of Disorder: *Every subset of n output points from Y with $n \leq k$, forms an $n \times k$ matrix which is of rank n with probability one.*

Strong Condition of Disorder: *Each subset of $n \leq k$ output points of Y is continuously distributed in nk dimensions with probability one.*

For a more detailed discussion see Gersho [80].

The *consequence* of interest to us from these conditions is that if a k-1 dimensional cut is taken of k dimensional Euclidean space, no two centroids will lie on the cut.

4.2.2 Some Useful Optimal Vector Quantisation Results

Gersho [81] studied the case of asymptotically optimal vector quantisation. The asymptotic situation arises where N, the number of output vectors in the set Y, is very large. The objective function in determining quantiser performance, of interest to the present argument, is the per letter distortion D :

$$D = \frac{1}{k} E\|x - Q(x)\|^r \tag{4.12}$$

where $\|\cdot\|$ denotes the l_2 norm, $E[\cdot]$ is the mathematical expectation over the joint distribution $p(x)$, r denotes the rth power distortion and $E\|x\|^r$ is assumed finite.

Using various assumptions he was able to show that the minimum distortion for the asymptotic case is given by,

$$D(N) = C(k,r)N^{-\beta}\|p(x)\|_{k/k+r} \tag{4.13}$$

where $p(x)$ is the distribution density of x and the notation:

$$\|p(x)\|_\alpha = \left[\int [p(x)]^\alpha dx\right]^{1/\alpha} \tag{4.14}$$

is introduced. Zador also obtained the same result earlier. The coefficient $C(\cdot,\cdot)$ is known as the coefficient of quantisation. A remarkable result is that this coefficient is independent of the joint probability distribution. Gersho also conjectured that;

$$C(k,r) = \inf_{H_k \in R^k} \frac{1}{k} \frac{\int_{H_k} \|x - y\|^r dx}{[V(H_k)]^{1+r/k}} \tag{4.15}$$

where $\mathbf{y}$ is the centroid of the regular polytope H_k and $V(H_k)$ is its volume. (r represents the rth power distortion.)

The distortion integral was minimized by optimizing the choice of the asymptotic output point density function which was found to be proportional to $p^{1/(1+\beta)}$. This result implies that (using Gersho's words)

> ... each region R_i of the partition makes an equal contribution to the distortion for an optimal quantiser.

The argument may be extended to the more general case where the number of output vectors N is not very large and the assumptions made in deriving the result above are not valid. Intuitively it may be argued that the general polytope cells represented by the output vectors y_i have a smaller volume if the joint probability density for that cell is high. The reason that this conjecture is made is that practical quantisers tend to show that the quantiser produces small cells in regions of higher probability density. Gersho and Gray [33] pp.373–378 show an example quantiser of two independent identically distributed Gaussian variables. They also note that the theoretical asymptotic high resolution approximation is not far from the actual performance, especially at higher dimensions.

Lookabaugh and Gray [82] looked into a quantitative approach of why a vector quantiser performs better than repeated scalar quantisation for the same source. They classify three advantages of vector quantisation over scalar quantisation following notions presented by Makhoul et. al [79]. The three advantages as presented by them are:

1. *Space-filling* advantage. This corresponds to the increased packing densities available at higher dimensions and this also corresponds to Zador's coefficient of quantisation and Gersho's conjecture that the polytope that produces minimum distortion in the dimension of interest is of primary importance in evaluating the coefficient of quantisation. (Refer to Equation 4.13);
2. *Shape* advantage. The advantage here is that the polytopal Voronoi region is free to take on any shape that best describes the joint probability density function. It should be noted that repeated scalar quantisations force a rectangular structure to the cells, where the dimensions of the sides of the rectangles will depend on the variances of the marginal distributions;
3. *Memory/Correlation* advantage. Lookabaugh and Gray use the term memory, but the term correlation will be used here to bring to attention that a correlation exists between the components of the vectors that are under consideration. Repeated scalar quantisation does not consider correlations between successive samples. Thus it may be said that vector quantisation models the joint distribution density including correlational effects.

It should be noted that the repeated scalar quantiser and the vector quantisers were considered to have the same number of output points.

Recalling Gersho's conjecture Equation 4.15 and using Zador's result, Lookabaugh and Gray presented expressions for the ratio of distortion due to repeated scalar quantisation to vector quantisation. The equations are repeated here for convenience.

Let Δ(k,r) be the advantage, in terms of distortion, of vector quantisation over repeated scalar quantisation. k is the dimension of the vector and r is the rth power distortion.

$$\Delta(k,r) = \frac{C(1,r)\|\overline{p}(x)\|_{1/1+r}}{C(k,r)\|p(x)\|_{k/(k+r)}} \tag{4.16}$$

under the assumption that the source is stationary and $\overline{p}(x)$ is the marginal density. Also define

$$p^*(x) = \prod_{i=0}^{k-1} \overline{p}(x_i) \tag{4.17}$$

the distribution that would result if the vector coordinates were independent.

They then present the equation,

$$\Delta(k,r) = F(k,r)S(k,r)M(k,r) \tag{4.18}$$

where F(k,r) is the *space-filling* advantage, S(k,r) is the *shape* advantage and M(k,r) is the *correlation* advantage.

The space filling advantage is straight forward, but the shape and correlation advantages are of special interest to the missing data problem.

The *shape* advantage given by:

$$S(k,r) = \frac{\|\overline{p}(x)\|_{1/(1+r)}}{\|p^*(x)\|_{k/(k+r)}} \tag{4.19}$$

depends only on the marginal probability densities and can be shown to be ≥ 1 for $k \geq 1$ and $r > 0$.

The *correlation* advantage is given by:

$$M(k,r) = \frac{\|p^*(x)\|_{k/(k+r)}}{\|p(x)\|_{k/(k+r)}} \tag{4.20}$$

which depends on the k dimensional joint distribution density function. Generally this can be very difficult to calculate, but note that M(k,r) = 1 only when the components are independent and identically distributed. Thus the vector quantiser takes the joint distribution function into account in its structure.

4.3 Heuristic Argument

A heuristic argument for a new approach to the missing data problem for incomplete vectors of data follows in this section.

It is known that the optimum estimate for the missing data component from statistics under a mean square error criterion, is the expectation with respect to the joint density function given the other components. The question arises: what is the optimum choice of output vector, with respect to overall reproduction distortion, given that a vector quantiser codebook exists for the source?

Assume that only one vector element is missing. If a nearest neighbour search is performed for all of the known components of the vector, a k-1 dimensional cut is performed through the Voronoi (polytopes) in k dimensional space. It is known that the polytopes are convex and so a k-1 dimensional cut will still leave the polytope convex, due to the definition of convexity,

$$\alpha(x) + (1-\alpha)(y) \in S \quad \forall x, y \in S \tag{4.21}$$

where $0 < \alpha < 1$.

We can not choose two points such that all points between them are not inside the convex region, S. Consider the two cases where the original convex region and the cutting hyperplane generate a new convex region containing y and x (call this set **C**):

1. point $x \in C$ and point y on the cutting plane. Clearly all points between x and y are in the new convex region;
2. point $x \in C$ and point y in the previous existing convex region, not on the cutting plane. All points between x and y must be in the new convex region as all points C are in the new convex region and the previous convex region. We make use of the fact that the previous region was convex.

This implies that cutting one of the Voronoi regions does not result in *strange* behaviour such as generating unconnected regions, belonging to the one polytope, but now isolated on the k-1 dimensional hyperplane. As the polytopes are space filling, the k-1 dimensional cut will result in a covering of the cutting hyperplane, without regions that did not cut through some polytope.

From the conditions of disorder quoted above we can say that with probability 1 that a k-1 dimensional cut will result in at most one centroid on the hyperplane and at most one pair of centroids equidistant from the hyperplane. The distances of the centroids from the hyperplane will have a random distribution, so choosing the correct Voronoi region (output vector) will be probabilistic. If the assumption that the Voronoi region size depends on the probability density is made, smaller regions on the hyperplane correspond to regions of higher probability. Two things should be noted at this stage:

Firstly, for a random cut through a polytope a larger cross section is expected if the polytope is large. If we couple this with the distance from the centroid of the polytope, a reasonable estimator of the most likely representative Voronoi region is obtained. A possible estimator is the volume of the approximate k dimensional simplex generated by the vertices of the intersection of the polytope and the hyperplane and the centroid of the polytope. By approximate it is meant that the vertices are chosen so that the area of the intersection and the area of the simplex are equal. The volume is given by:

$$Vol(P) = \frac{1}{k!} \det \begin{vmatrix} 1 & v_{01} & v_{02} & \cdots & v_{0k} \\ 1 & v_{11} & v_{12} & \cdots & v_{1k} \\ 1 & v_{21} & v_{22} & \cdots & v_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & v_{k1} & v_{k2} & \cdots & v_{kk} \end{vmatrix} \tag{4.22}$$

where $v_i = (v_{i1}, \cdots, v_{ik})$ for $0 \leq i \leq k$ are the vertices of the simplex [83]. This volume relates to the distance from the centroid to the base and the size of the base, that is, the $k-1$ cross section of the polytope. Choosing the smallest volume thus gives the highest probability of choosing the correct polytope centroid (output vector). Unfortunately finding the vertices of the simplex other than the centroid, given by the codebook entry, is difficult.

Secondly, the smaller the Voronoi region the greater the probability that the minimum distance to a centroid d_{min} belongs to one of the small cells. Let the polytope (Voronoi region) be represented by a sphere that lies between the polytope's in-sphere (totally enclosed by the polytope) and circum-sphere (totally encloses the polytope).

If this sphere is cut by a $k-1$ dimensional plane there will be a point of minimum distance between the center of the sphere and the plane. Denote this point x and the center of the sphere c. We wish to determine the probability that this distance is less than some distance d_{min} obtained by some other polytope. This must be related to D the diameter of the representative sphere. Using the notation $P\{.\}$ for probability;

$$P\{|c-x| < d_{min}\} = P\{c < x < c + d_{min}\} + P\{c - d_{min} < x < c\}$$

$$= P_d \tag{4.23}$$

Evaluating P_d for a uniform distribution, as a cut through the polytope is uniform, and assuming the sphere diameter D,

$$\begin{aligned} P_d &= \frac{1}{D}\int_{c}^{c+d_{min}} dx + \frac{1}{D}\int_{c-d_{min}}^{c} dx \qquad (4.24) \\ &= \frac{2d_{min}}{D} \qquad (4.25) \end{aligned}$$

For a given d_{min} found by a nearest neighbour search to the hyperplane the probability is greater that the centroid in question belongs to a smaller polytope. Also if the correlation between the vector components is large the marginal density function will have a narrow peak. Consider two components x_i, x_j that are strongly correlated. The joint distribution density function will be compressed toward the $x_i = ax_j$ line. The greater the correlation the greater the compression of the distribution as the variance is small. It can thus be expected that regions of small cells occur near likely combinations of vector components.

Given the above arguments and that the area of the cuts of the polytopes are not readily available; the best decision of a polytope most closely satisfying the missing data problem defined by the hyperplane is the one that has its centroid closest to the plane. It is noted that this results in a probabilistic assignment of output vector to match the hyperplane and may not always correspond to the same output vector had traditional statistics with a mean square error criterion been used.

4.4 Codebook Generation

The source speech coder generates the ten dimensional LSP vectors on-line and the in-coming received vectors are to be compared with entries in a local vector quantiser codebook. Firstly, it should be noted that only the vectors that are detected as being in error need be compared to the codebook entries for the missing data estimation process. Secondly, because no codebook index is transmitted across the channel the number of entries in the codebook is not restricted to a power of two, the case for traditional vector quantiser source coding, as the constraint of bit efficiency on transmission is eliminated.

There are two possible strategies for the codebook generation:

1. *The codebook is fixed.* This requires a large source of mixed speakers so that the codebook is representative of all the possibilities of the source. As the codebook is fixed this mechanism provides the fastest alternative at the receiving end. Although it operates outside of the training sequence it cannot be corrupted by erroneous received vectors that have not been detected by the error detection strategy.

2. *The codebook is adaptive.* This method requires that the codebook is on-line and is continuously learning. An advantage of this method is that it is adaptive and hence will train itself to a speaker, making the output vectors more relevant for that particular speaker. Disadvantages are that extra computations must be performed for each received vector and that erroneous vectors received but not detected go into the training set and thus corrupt the quantiser. Vectors where errors are detected, are not used in training the codebook.

The source material used for this experiment consisted of thirteen speakers, of mixed ages and sex. All of the sentences uttered were phonetically balanced sentences. Overall approximately 150 seconds of digitized speech was used in the training and testing of the codebooks.

4.4.1 Fixed Codebook

The fixed codebook was constructed using the Linde, Buzo and Gray [78] version of the k-means algorithm. The source material consisted of the material mentioned above with the male and female voices interleaved. To obtain an *in training set* approximately 60 seconds of speech from two males and two females was used. For the *out of training set*, the rest of the above mentioned speech was used. Only codebooks of size up to N=128 were considered as the condition $M \gg N$ is a requirement for good training of the codebook: where M is the number of training vectors and in this case was about 2000 LSP vectors.

4.4.2 Adaptive Codebook

The description of the adaptive codebook given below references a new approach to constructing adaptive vector codebooks.

The adaptive codebook is primarily characterized by moving means (centroids) of the Voronoi regions and consequently an altering partition of $\Re^k$ space. If the variance statistics are maintained for each quantiser cell it is possible to merge two cells when the distortion (variance) drops below a threshold and simultaneously split the quantiser cell with the greatest variance maintaining a constant codebook size. This method allows rapid adaption to changing source data. Given the approximate distortion for each quantiser cell, the updating of the variance (distortion) statistics requires investigation for the case of merging and splitting. In order for analysis to proceed the asymptotic vector quantiser argument is followed.

In the merger case, two small cells S_j and S_i are combined to create a new volume, the sum of the two component parts. It should be noted that only adjacent cells are allowed to merge. The expression for the new volume is given by:

$$V(S_i) + V(S_j) = V(S_n) \tag{4.26}$$

where the new quantiser cell is designated S_n and $V(.)$ denotes volume. As the approximate variance for each original cell is known it is hypothesised that the variance for the new cell should be the sum of the original variances scaled by a factor $1/\rho$. Using the fact that in the asymptotic case (Gersho [81]) the distortion D for N quantiser cells is given by:

$$D = C(k,r) \sum_{i=1}^{N} p(y_i)[V(S_i)]^{1+r/k} \tag{4.27}$$

where $V(S_i)$ is the k dimensional volume of the quantiser cell for the r'th power distortion measure and $p(y_i)$ is the probability density at the centroid of quantiser region S_i. It was assumed that the probability density function was constant over the region. When $r = 2$ the distortion for the quantiser cell may be interpreted as a variance or second moment. Recall that C(k,r) is a constant dependent only on the geometrical shape of the quantiser cell. If the further assumption is made that the variance threshold is chosen so that the merger process causes negligible distortion, then:

$$p(y_i)V(S_i)^{\frac{k+2}{k}} + p(y_j)V(S_j)^{\frac{k+2}{k}} \approx \rho p(y_n)V(S_n)^{\frac{k+2}{k}} \tag{4.28}$$

If the further assumption is made that $p(x)$ is smoothly varying and that the merger regions are small,

$$p(y_i) \approx p(y_j) \approx p(y_n) \tag{4.29}$$

One further simplifying assumption is made by claiming that the two merging cells are of similar volume so, $V(S_i) \approx V(S_j)$.

This allows the approximation of Equation 4.28 to be written as,

$$2V(S_i)^{\frac{k+2}{k}} = \rho[2V(S_i)]^{\frac{k+2}{k}} \tag{4.30}$$

Evaluating ρ gives the scaling factor $1/\rho = 0.871$ for a $k = 10$ dimensional vector. The scaling factor is close to 1 and taking into account all of the assumptions made on the sum of the quantiser cells using 1 seems justified.

A similar argument can be performed for the splitting of the cell with the largest distortion; however the assumptions on the probability density and the size of the two resulting cells being equal becomes less tenuous. For this case the crude approximation of halving the variance for the parent cell and allocating this value to each of the new cells was made.

As the means move in the adaptive vector quantiser, the distortion/variance statistic $\hat{z}$, is best estimated by a recursive estimator;

$$\begin{aligned} g(m) &= z + \alpha g(m-1) && (4.31) \\ z &= (x - y_i)^2 && (4.32) \\ \hat{z} &= g(m)(1-\alpha) && (4.33) \end{aligned}$$

where m is an index for each new vector x that is quantised as a member of the cell that is represented by y_i and α determines the memory of the estimator.

The merging may be viewed another way if the codebook is of a small to moderate size and the centroids of the two cells to be merged, x_i and x_j, and the cell distortions may be considered relatively constant. Let the two cells to merge, S_i and S_j, have a membership of n_i and n_j and distortions D_i and D_j. If the new cell is denoted S_{ij} then;

$$n_{ij} = n_i + n_j \tag{4.34}$$

$$\bar{x}_{ij} = \frac{n_i\bar{x}_i + n_j\bar{x}_j}{n_i + n_j} \tag{4.35}$$

$$n_{ij}D_{ij} = \sum_{x\in S_{ij}} |x - \bar{x}_{ij}|^2 \tag{4.36}$$

$$= \sum_{x\in S_i} |x - \bar{x}_{ij}|^2 + \sum_{x\in S_j} |x - \bar{x}_{ij}|^2 \tag{4.37}$$

where,

$$\sum_{x\in S_i} |x - \bar{x}_{ij}|^2 = \sum_{x\in S_i} (|x|^2 - 2 < x, \bar{x}_{ij} > + |\bar{x}_{ij}|^2) \tag{4.38}$$

$$= n_i(D_i + |\bar{x}_i|^2) - 2n_i < \bar{x}_i, \bar{x}_{ij} > + n_i|\bar{x}_{ij}|^2 \tag{4.39}$$

$$= n_iD_i + n_i|\frac{n_i\bar{x}_i + n_j\bar{x}_i - n_i\bar{x}_i - n_j\bar{x}_j}{n_i + n_j}|^2 \tag{4.40}$$

$$= n_iD_i + \frac{n_in_j^2}{(n_i+n_j)^2}|\bar{x}_i - \bar{x}_j|^2 \tag{4.41}$$

The first step was performed by summing over the membership n_i of the cell and noting the $\sum_{x\in S_i} x = n_i\bar{x}_i$ and that the centroids are constants. The second step used substitution for $\bar{x}_{ij}$ and the last collected terms.

Using the above expression the distortion for the merged cell D_{ij} can be expressed as,

$$n_{ij}D_{ij} = n_iD_i + n_jD_j + \frac{n_in_j^2}{(n_i+n_j)^2}|\bar{x}_i - \bar{x}_j|^2 + \frac{n_jn_i^2}{(n_i+n_j)^2}|\bar{x}_j - \bar{x}_i|^2 \tag{4.42}$$

$$= n_iD_i + n_jD_j + \frac{n_in_j}{n_i+n_j}|\bar{x}_i - \bar{x}_j|^2 \tag{4.43}$$

This method was first described by Equitz [85] for use in his clustering algorithm. To compare this with the asymptotic form we let $n_i = n_j = n$ and $D_i = D_j = D$. Substituting,

$$2nD_{ij} = nD + nD + \frac{n^2}{2n}|\bar{x}_i - \bar{x}_j|^2 \tag{4.44}$$

which gives,

$$D_{ij} = D + \frac{1}{4}|\bar{x}_i - \bar{x}_j|^2 \tag{4.45}$$

The main difference in the two distortions is the term relating the squared distance between the two centroids. In this application, the centroids move so the derivations above are approximate but useful guidelines. The implemented algorithm used the simpler sum of individual distortions.

The threshold for merging and α were determined empirically for a good performance. A value of 0.9 for α was found to be serviceable. The thresholds for merging were allowed to be dynamic depending on the maximum of the average distortions of all of the cells. The distortion of merging the cell with the lowest distortion and its nearest neighbour is determined. If the incoming vector is to be quantised by the cell that displays the greatest distortion, the distortion in quantising the incoming vector is found and retained. If this quantising distortion is greater than twice the distortion caused by merging of the two candidate low distortion cells, perform the split/merge.

All of the available source material was used in the operation of the adaptive codebook. The adaptive codebook was also generated on-line by a modified version of the k-means algorithm using notions outlined above. The algorithm is described in the steps below:

1. Till N output vectors are defined, read the next incoming vector and if unique, place in a new cell;
2. With the next vector find the output vector that most closely matches it (nearest neighbour) in a mean square error sense and maintain the distortion value $Dist_{vec}$;
3. *If* the vector is to be quantised by the cell with greatest distortion (variance) *then* check to see if the average distortion of the candidate cell to be merged will be less than $2.0 * Dist_{vec}$;
4. *If* above true *then*: use the input vector as the new centroid for splitting the large distortion cell and initialize the variance, running sum and variance for the two new cells, and merge the cell with the lowest distortion and its closest neighbour, reinitializing the running sums, population count, and variance for the new cell; *else* add the input vector to the running sum for the cell, and increment the population count;
5. Using the running sum and population count, recalculate the new centroid, and if necessary update the variance;
6. Find the nearest neighbour of the quantiser cell with the smallest distortion;
7. Find cell with maximum distortion;
8. Goto step 2 and continue.

The algorithm as it stands clearly is adaptive as new populations of vectors can move the centroids in such a way that they follow the new population statistics. The additional feature of merging and splitting was added to ensure rapid innovation in the codebook. The number of codebook entries was always constant and could be set by the user. By adding the new feature the adaptive nature was improved, but a disadvantage was that more detailed book-keeping and statistics on variances of populations occupying cells had to be maintained.

4.5 Performance

In determining the performance of the vector quantiser as a useful device for the missing data problem, the fixed codebook approach was compared to the techniques in common usage in practical speech coders. The adaptive codebook and the fixed codebook were then compared. The measure chosen to quantify distortion was the Spectral Distortion Measure (SDM), as this method compares the spectral envelope between the linear predictor coefficients (LPC's) of the original speech and the reconstructed LPC's. This measure is often used in the literature [32] and has reasonable agreement with subjective tests. For these tests all of the LPC source data were in an LSP format and the spectral differences that channel errors caused were in the LSP representation. As a result of using this distortion measure conversions between LPC and LSP representations of the predictor coefficients were necessary for each vector.

The expression for the spectral distortion measure is given by:

$$\left(\int_0^{\omega_s/2} 10log_{10}\frac{S_\omega}{\hat{S}_\omega}d\omega\right)^{1/2} \tag{4.46}$$

Where: $\omega_s = 2\pi f_s$ and f_s is the sampling frequency and, S_ω and $\hat{S_\omega}$ are the spectral power responses for the original LPC predictor filter and the LPC predictor filter under test, respectively. It has been noted that a spectral distortion of less than 1 dB on average results in transparent coding, that is, the difference between the two LPC filters would be imperceptible to the ear [36].

For both the fixed codebook and the adaptive codebook, the missing data substitution follows an identical procedure, noting that the vector components must form a strictly monotonically increasing sequence:

1. The received vector $\omega_1, \omega_2, \ldots, \omega_{10}$ is tested for any components out of order;
2. All components out of order are marked as an erasure, that is, each ω_n and ω_{n+1} out of order; also the original received vector is saved ;

3. As erasures occur in pairs it can not be certain which of the two components out of order are in error if not both. The correction procedure first assumes the first component is in error and fixes the second at the received value, followed by the second component assumed in error and the first held fixed at the received value;

4. The first erasure is replaced by the vector component ω_n from the best matching vector obtained from a nearest neighbour search in the vector quantiser's codebook. The minimum distance output vector is noted, as is the distance. If there are other erasures in the received code vector, the vector elements associated with the erasures are not used in determining the minimum distance to a codebook entry;

5. The second erasure corresponding to ω_{n+1} is replaced by the corresponding component in the best matched vector obtained from a nearest neighbour search of the codebook. The minimum distance and the selected output vector is noted;

6. Of the two output vectors, the one which has the minimum distance to the hyperplane (caused by removing the erasure component) is selected as the best vector;

7. The most likely (best) codebook vector's component that corresponds to the erasure position is used as the substituted value. Monotonicity is checked and any anomalies resolved. Further erasures can be resolved by either sorting the vector elements into increasing order or by using the entire most likely codebook vector. There was little difference in performance.

The process can be illustrated by writing out the vectors;

$$\begin{array}{cccccccc} v_{i1} & v_{i2} & v_{i3} & v_{i4} & v_{i5} & v_{i6} & \cdots & v_{in} \\ r_1 & r_2 & r_3 & \varepsilon & r_5 & r_6 & \cdots & r_n \end{array} \tag{4.47}$$

The vector v_i in (4.47) represents the vector that most closely matched the received vector r which had the LSP's out of order in the 4th and 5th locations. ε indicates the trial erasure, that is, the co-ordinate that is not used in the vector match. The next step trials the 5th location in the received vector as an erasure.

$$\begin{array}{cccccccc} v_{j1} & v_{j2} & v_{j3} & v_{j4} & v_{j5} & v_{j6} & \cdots & v_{in} \\ r_1 & r_2 & r_3 & r_4 & \varepsilon & r_6 & \cdots & r_n \end{array} \tag{4.48}$$

Assume that the vector v_j in (4.48) produced the best match in the sense it was closest in a mean square error sense. The received vector would be corrected to 4.49 if criteria of monotonicity and spacing between LSP components were met, else the entire vector v_j would be used.

$$r_1 \quad r_2 \quad r_3 \quad v_{j4} \quad r_5 \quad r_6 \quad \cdots \quad r_n \tag{4.49}$$

The reason that the matching was only performed on the first erasures was to reduce processing and that the first two formants for human speech[1] would be expected between $\omega_2 - \omega_4$ and the second formant between $\omega_5 - \omega_7$ [84]. The most important formant for human ear perception is the first formant. Only one value is substituted rather than the whole vector because as Paliwal and Atal [36] elucidate the LSP representation has the nice property of only affecting the spectrum locally. That is, an incorrect LSP component will only affect the spectrum in the region associated with its frequency. Replacing a single component will result in a better spectral distortion measure than sorting, assuming a single corrupted LSP. If however no substitution can be made that meets the criterion of monotonicity and perhaps LSP spacing, the matched vector from the codebook is taken in its entirety to ensure a stable synthesis filter.

4.5.1 Common Current Techniques

Two common techniques used in present speech coder implementations are considered for comparison to the methods developed here. The first technique is to simply use, on detected error, the last correctly received LSP vector. This method guarantees that a valid vector is used and is extremely simple and exploits the fact that from frame to frame the LSP vectors tend to be similar. However as far as error contribution goes, the whole substituted vector was found to contribute significantly to the error unless the vectors happened to be identical. There was a reasonable chance that the vector would be quite similar particularly during voiced speech with a sustained vowel.

The second technique takes the received LSP vector in error and sorts the components of the vector into monotonic increasing order. This again results in a valid vector and guarantees a good match on the components of the vector that were not out of order. The error contribution in this case would only be over the two vector components that

[1]The first two formants are the most perceptible.

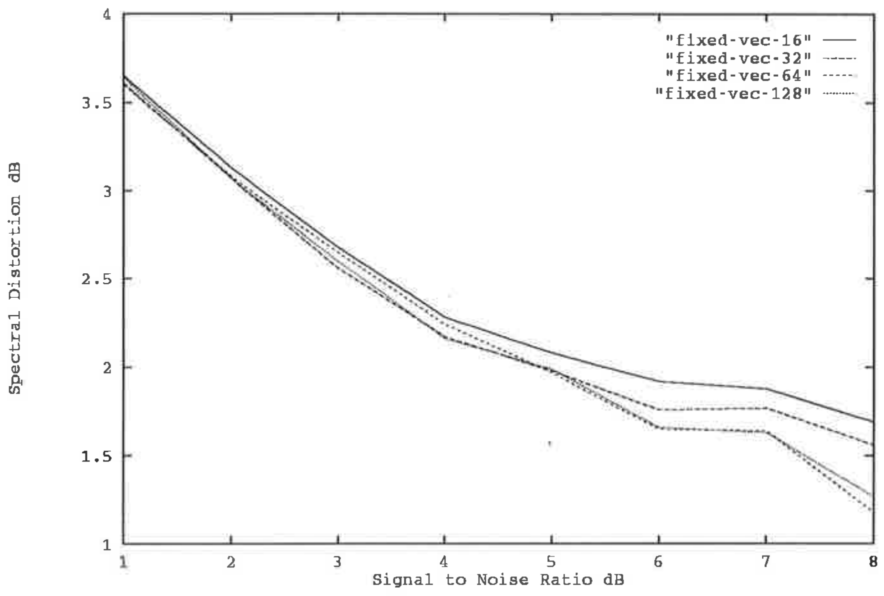


Figure 4.1: Fixed Vector Codebook Performance in Missing Data Estimation: the plot names indicate the size of the fixed codebook.

were out of order, assuming no other undetected errors. To ensure that no high level resonances occur, the sorted LSP's are arranged to be at least 15 Hz apart.

4.5.2 Comparison of Methods

For all of the experiments the speech data-base was converted to an LSP representation and then quantised using the US Federal Standard 1016 CELP quantisation tables. The channel simulation transmitted the quantised LSP vector through an additive white Gaussian noise (AWGN) channel where BPSK modulation and hard decision decoding were assumed. To effectively obtain more source vectors the vector source file was concatenated three times for the higher signal to noise ratios to obtain a significant number of error events. Signal to noise ratios were only simulated to 8 dB where there were still a small but significant number of errors ($\approx$80).

Figure 4.1 shows that a small increase in performance was achieved by going to a larger codebook. The mean square error was determined on a per LSP component basis (units of radians squared) and the signal to noise ratio refers to the additive white Gaussian noise channel that the quantised LSP indices were transmitted over. The spectral distortion was an average spectral distortion that occurred in the process of estimating

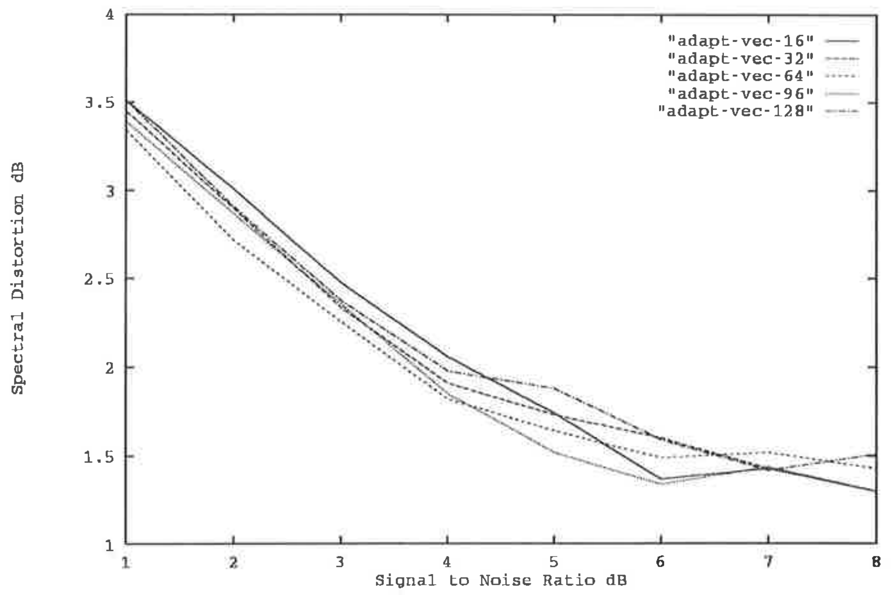


Figure 4.2: Adaptive Vector Codebook Performance in Missing Data Estimation: plot name extensions indicate the size of the codebook.

the reconstruction vector. Distortions due to no errors and undetected errors were not considered here. Most of the gain appears to be achieved with 64 vectors. The size of the codebooks double in size as an artifact of using the Linde, Buzo, Gray version of the k-means algorithm which performs a binary splitting at each level during codebook creation. [78]

Figure 4.2 shows an improvement with increasing codebook size for the adaptive codebook till it reaches approximately 100 vectors. A possible reason for the saturation was that the clustering procedure was not optimum and the on-line training of the codebook included received undetected erroneous vectors. Spectrally uncharacteristic cells may have been created that were not adjusted by correct data, as correct data never matched those cells. The uncharacteristic cells however were still available for matching in the missing data estimation process. The spectral distortion only accounted for the reconstruction process for vectors found in error.

From Figure 4.3 it is quite clear that the vector quantiser's performance is superior to the traditional methods of correcting channel corruption of LSP parameters. The distortion caused by undetected errors was dealt with separately and plotted as *undetected-errors*; this distortion component was in addition to all of the schemes represented here and was due entirely to undetected error vectors getting through the error

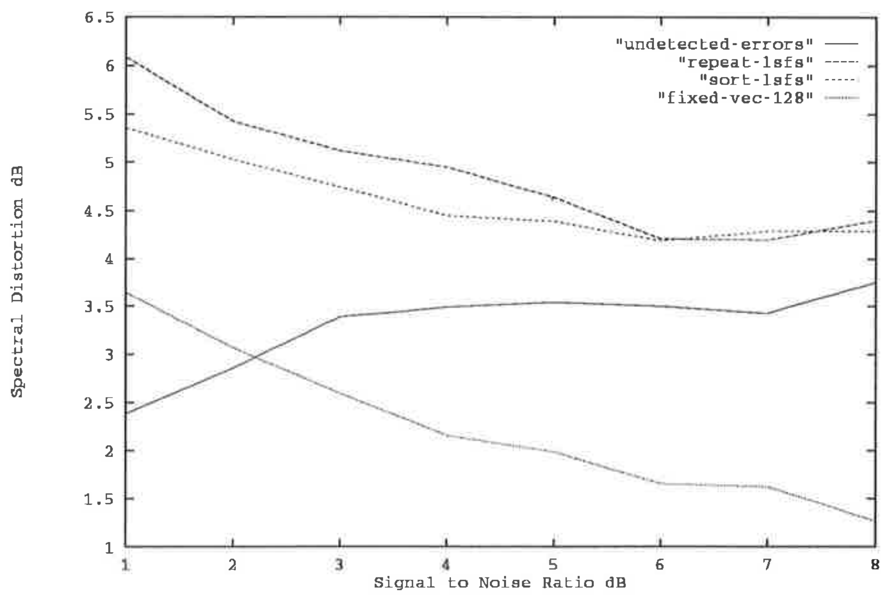


Figure 4.3: Comparison of Methods in Missing Data Estimation: 128 entry fixed vector codebook, sorting of the LSP components, repetition of entire LSP vectors and distortion due to undetected errors.

detection process. It should be noted, from the point of view of spectral representation for the corrupted frame, sorting of the LSP parameters in a monotonically increasing order and adjusting any pairs too close (avoid large resonances) performed better than repeating a previous frames spectral information for the lower signal to noise ratios. To obtain better performance it is clear that a better error detection scheme should be employed. It should be considered that the undetected error distortion and reconstruction distortion do not add directly but must be averaged over all vector error events.

It can be seen from Figure 4.4 that the LSP monotonicity as an error detection mechanism only detects between approximately 1/2 and 1/3 of erroneous vectors. More erroneous vectors were detected at higher bit error rates (lower signal to noise ratios) as any one vector of LSP parameters had a higher probability of having more than one bit error. At the lower channel error rates single bit errors do not necessarily cause an interchange in LSP ordering hence undetected errors occur. For better performance a more sophisticated error correction scheme and/or error detection scheme would be required. One possible method may be to encode the differences in angles of the LSP's and encode the last LSP's difference from π. On reconstruction any error will cause the sum of the differences to no longer equal π, within an allowed tolerance.

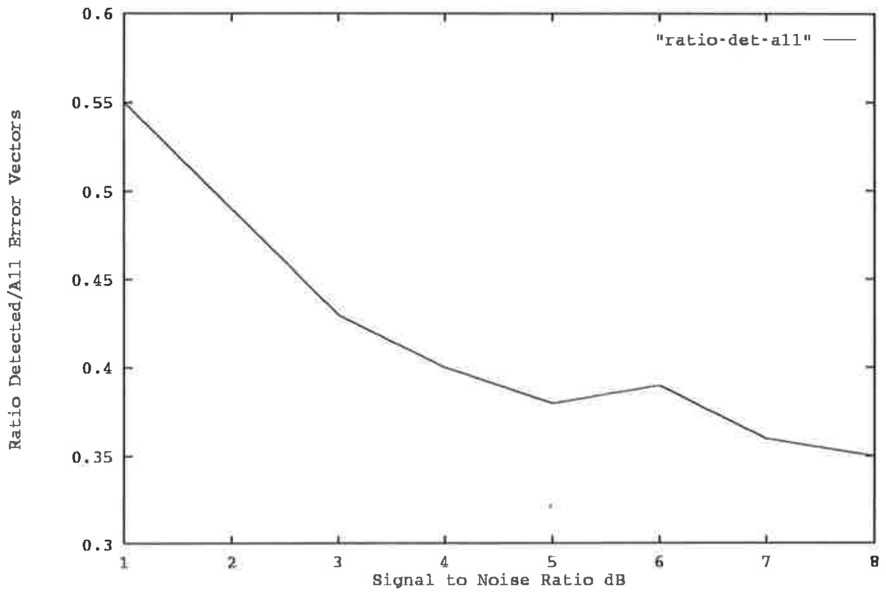


Figure 4.4: Ratio of Detected Error Vectors to all Vectors with Bit Errors.

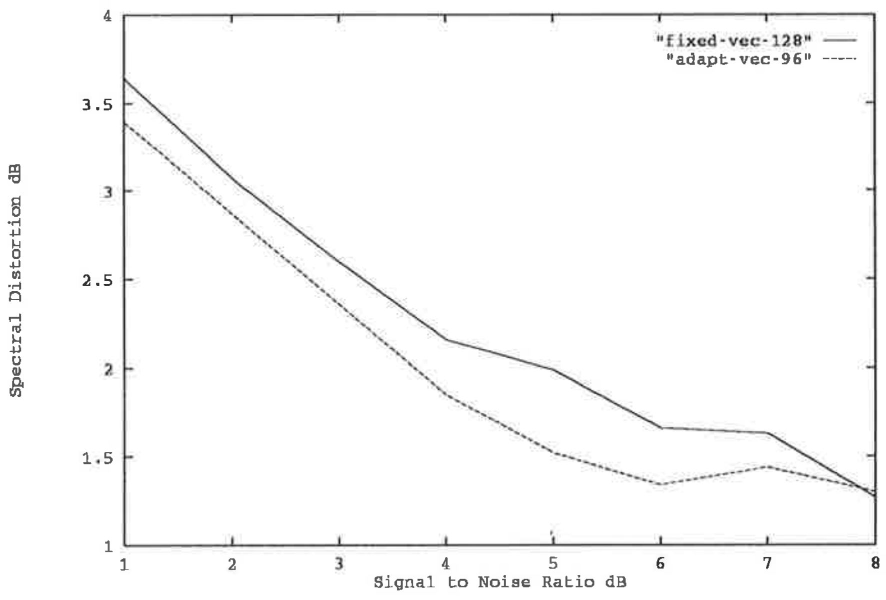


Figure 4.5: Comparison of a Fixed Vector Codebook and an Adaptive Codebook.

Figure 4.5 shows a comparison of the 96 entry adaptive codebook and the fixed vector codebook of 128 entries. It should be noted that at high signal to noise ratios the two schemes work similarly, however at lower signal to noise ratios the adaptive codebook has 0.5 dB advantage for a given performance level. The ability to adapt to a given speaker under the current conditions is a definite advantage for the adaptive codebook. This process also allows adaption to background noise which may be of benefit in some cases. The speakers in the speech file were chosen to interleave male and female speakers so that the adaptive codebook had to adapt relatively rapidly.

If a very robust system with small computational effort is required, the fixed codebook of 128 entries appears to be a good choice. However, caution must be taken in coming to this conclusion as the training set for the 128 entry codebook did not come from a large data base of different speakers and as such may not be as representative for a larger variety of speakers. A fixed codebook scheme may require different codebooks for different language speakers.

If flexibility in dealing with a wide range of speakers is required the adaptive codebook is the better alternative due to its on-line learning ability at the expense of extra computation.

4.6 Improving the Vector Quantiser Error Corrector

From the discussions above it should be clear that to improve the performance of the vector quantiser error corrector, particularly the adaptive version, a better error detection mechanism is necessary. In general digital communications cyclic codes are used extensively for error detection. From the experiments performed it is evident that undetected errors limit the possible performance of the proposed schemes. Principally the problem arises when an error causes an LSP value to change position but not enough to cause an ordering violation or proximity violation (must be >15 Hz apart). If a large burst of errors occurs the probability that ordering or proximity violations occur is much greater. A burst of errors is described by a starting error bit followed by a stream of errored or non-errored bits to a final error bit after which the burst is deemed finished before the next burst. This definition implies also an error-free guard length to be able to separate bursts in long codewords.

Many good references exist on error control coding, see for example [86] and [87]. An (n,k) codeword has a total length of n bits, k of which are information bits and $n-k$ of which are for parity. Codewords are generated by treating the information as a polynomial, dividing this polynomial by the generator polynomial of the code, and appending the remainder. On reception the complete codeword should be divisible by the generator polynomial with 0 remainder.

Cyclic codes have the following properties;

1. an (n,k) cyclic code can detect any burst of length $n-k$ or less,
2. the fraction of undetectable bursts of length $n-k+1$ is $2^{-(n-k-1)}$,
3. the fraction of undetectable bursts of length $l > n-k+1$ is $2^{-(n-k)}$.

Choosing the correct length code depends on the error detecting capability required and also the maximum increase in transmitted bit rate that can be tolerated. If 5 parity bits were allowed, the bit rate would be increased by 200 bits per second in the proposed design, (based on 40 frames per second.) It should be recalled that errors in the linear predictor coefficients are a major source of distortion in reconstructed speech.

If 5 parity bits were allowed the resultant code would be a (39,34) cyclic code. All error bursts of 5 bits or less would be detected, 1/16 bursts of length 6 would go by undetected and 1/32 of bursts of length greater than 6 would remain undetected.

This is clearly a far superior performance to the plain monotonicity check. The adaptive scheme could benefit greatly from the additional error detection as the number of corrupt vectors that would be used in training the codebook would be reduced, and hence the vectors in the codebook would remain more representative of the true on-line speaker's voice spectrum.

This simple error detection scheme does however have a price, in that the detection scheme does not yield any information about error location. Monotonicity, when violated, indicates an error and its approximate position. Cyclic code checks would only indicate the presence of a detected error. Without information about the location of the error it would be necessary to successively mark each LSP coefficient as an erasure and perform a pattern match to find the best match for that position. Of the ten resulting decisions the best is chosen as the final vector to use in its entirety or to extract the appropriate coefficient. The situation becomes more complicated if multiple errors are to be corrected, and probably the best technique to use in this case would be to substitute a complete vector or correct only the largest error in the lowest index position.

A suggested scheme is to use LSP encoding of the linear predictor coefficients and to couple this with an error detecting scheme. Gross errors would be detected by the property of monotonicity of the LSP representation and smaller errors will be detected and an assessment made of possible error by determining the nearest neighbour vector from the codebook. The erroneous vector would not be used for updating the adaptive codebook. This method would not be able to completely compensate for the previously undetected errors due to the approximation process that occurs in nearest neighbour matching.

4.7 Conclusions

A new method of error mitigation for Line Spectral Pairs encoding of the linear prediction coefficients of a CELP coder was presented in this chapter that resulted in lower spectral distortion than traditional methods in use. From the experiments conducted in this section together with the supportive introductory arguments, a vector quantiser was useful in the process of missing data estimation for this application. Central to the method was the concept of a hyperplane that cuts the multi-dimensional vector space. The hyperplane was generated by allowing the suspected error component of the vector to vary while holding the other components fixed.

Not only was the missing data estimated but the whole vector was approximated; however only the missing element was used, provided monotonicity was not violated. Monotonicity ensured that the LSP parameters would result in a minimum phase synthesis filter for the reconstruction of speech.

A saturation effect appears to occur with larger numbers of vectors in the quantisers' codebook. This may well be due to the effect of just taking the minimum distance centroid from the hyperplane without regard to the size of cell that the centroid is taken from. If the volume generated by the hyperplane cutting the Voronoi cell and the centroid is considered a more accurate estimate could be made. Using the normal distance of the centroid to the hyperplane, an on average good estimate resulted, but as the number of cells cut by the hyperplane increases the estimate procedure may degrade a little. The results indicated however that the method performed significantly better than the traditional sorting of LSP parameters.

A simple adaptive vector quantiser was seen to be an effective step in further improvement of the performance of the missing data estimation technique.

It was observed that the 96 entry adaptive codebook had superior performance to the 128 fixed vector codebook. The disadvantage in using the adaptive codebook was the additional computational load required for adapting the codebook. It was also observed that the 128 entry fixed codebook may not perform as well over a larger variety of speakers as indicated by the above experiments due to the limited database. A marginally larger codebook trained by a very large training set may provide improved performance.

As the adaptive codebook trains on-line no assumption is made about the nature of the speech transmitted across the channel and thus offers the greatest flexibility. If the additional computational load could be tolerated it would be the method of choice in LSP error reduction for the errored channel.

Chapter 5

Vector Quantiser Design for Spectral Information.

5.1 Introduction

Itakura [31], Soong and Juang [30], and Kang and Fransen [88] showed that Line Spectral Pairs (LSP) were the most efficient scalar quantisation technique for LPC information. Recently much work has been done in developing scalar quantisers for LSP information and the scalar technique was adapted for the US Federal Standard 1016 Code Excited Linear Predictive vocoder. Atal and coworkers [89] [36] use the term transparent quantisation to represent the level of quantisation required such that the quantisation process does not introduce any audible distortion in the coded speech. The parameter values that they suggest must be met for transparent coding is ≤ 1 dB average spectral distortion with no more than 2% outlier frames having spectral distortion greater than 2 dB. This implies that the scalar quantisation of LSP information must use 32–40 bits [36].

To reduce the bit rate further a vector quantisation scheme is required. There are several problems associated with a vector quantiser design. Assuming that approximately 25 bits are required to transparently encode the spectral information, the codebook requires 2^{25} entries. If *training vector* to *codebook vector* ratios of 15 to 50 are used, the training set will need to be of the order of 2^{30} vectors. Clearly this is an infeasible path to take if a training method such as the Linde, Buzo and Gray (lbg) algorithm is used which requires multiple iterations for convergence at each stage. Encoding source data under ideal conditions utilizes a full search, however for random codebook of this magnitude it is infeasible to achieve this in realtime[1]. Tree structured codebooks [33] provide a solution to the encoding computational load problem at the expense of quantisation performance. If the expense is of the order of 2 bits then the codebook expands to 2^{27} entries and the training set expands to 2^{32}. If the codebook size is represented by $||C||$, tree search encoding reduces the effort to $\approx log_2||C||$.

Another problem with vector quantisation in a noisy transmission scenario is the channel error problem. One bit in error in the transmitted index due to channel noise may cause a severe error in the choice that the decoder makes. The LSP representation of the LPC parameters has the property of monotonicity of the roots. This affords some channel error detection and various error masking schemes may be employed as covered in the previous chapter. Indices from vector quantiser codebooks do not naturally have this property and measures must be taken to increase the robustness in noisy channels. By taking the approach that the vector quantiser be designed in such a way as to cause minimum distortion at reconstruction time should the transmitted codebook index be corrupted, a degree of robustness can be obtained [39] [40] [42]. A nice property of vector quantisation is that a stable synthesis filter configuration is always chosen albeit the incorrect one if one or bits were in error in the index.

The work in this Chapter concentrated on designing a vector quantiser that required only moderate computational effort for encoding and was robust to channel errors. Tree structured vector quantisers were found to be useful in meeting the computational criterion. Apart from the generalized Lloyd algorithm, optimised vector quantisers have been designed using simulated annealing [90] in the past, especially if the vector quantiser had to meet special constraints, however a genetic algorithm [91] will be introduced to perform a similar function. The areas in which this work differs from the work in the literature is that the large codebooks are tree structured within a split vector quantiser and multistage vector quantiser to facilitate realtime coding and minimize storage. Also errors in the received codebook index are also considered.

5.2 Tree Structured Vector Quantiser Design

The first major decision was how best to represent the LPC information vector. For scalar quantisation it has been shown in the literature that the LSP representation is the most efficient. For the vector quantisation case it was not immediately obvious which

[1]Using current technology.

representation to use. For example transformed partial correlation coefficients [92] of the linear predictor gives a representation that ensures filter stability; they also have good interpolation properties. Reflection coefficients could also be used to construct the vector. This had the advantage that no transformation process would be required after finding the linear predictor coefficients using the Schur recursion. However there was a disadvantage in that an error in any of the reflection coefficients had an effect on the whole short term spectrum (*spectral leakage*)[36]. The spectral leakage effect precluded any weighted mean square error criterion for individual coefficients that may have helped in the design of the vector quantiser. It has been found by various authors that the LSP's have localized spectral sensitivity[36] [93]. This localization of the spectral sensitivity allows a weighted mean square error criterion that can be loosely related to the frequency spectrum and was instrumental in allowing split vector quantiser designs covered later. It was also a useful property for the multistage vector quantiser design, where the basic spectral envelope shape was found by the first quantiser and then modified by the second.

A tree structured design was undertaken to reduce the computational load associated with the full search strategy for large random codebooks. To obtain maximum performance from the codebook a weighted mean square error criterion was used, as other authors for example Paliwal and Atal [36], have indicated performance better than the mean square error criterion. The weighting Paliwal and Atal suggested was of the form,

$$d(f,\hat{f}) = \sum_{i=1}^{10}[w_i(f_i - \hat{f}_i)]^2 \tag{5.1}$$

where $d(.,.)$ was the distance measure, f_i and $\hat{f}_i$ were the test and reference vectors respectively, and w_i was the weighting coefficient for the ith component. Paliwal and Atal[36] define w_i as;

$$w_i = [P(f_i)]^r \tag{5.2}$$

where $P(f_i)$ is the LPC power spectrum magnitude associated with the test vector as a function of frequency, and r was an empirical constant equal to 0.15.

This weighting function gave good results for the split vector quantiser design; it reduced the codebook index size by 2 bits, but came with the price of higher computational load. Another weighting function that could be used with the mean square error was sought that allowed for fast computation. LSP spectral sensitivity was studied by Sugamura and Farvardin [32] who came to the conclusion that the first two LSP components were approximately 1.5 times more sensitive than the other eight components for spoken English and Japanese (for a ten pole LPC model). In viewing short term spectra of speech it was noted that in general the power levels at higher frequency were considerably reduced. It is well known that the human ear cannot resolve differences at high frequencies as well as it can at low. Paliwal and Atal [36] multiply their weighting function by an additional factor c_i:

$$c_i = \begin{cases} 1.0, & \text{for } \ 1 \le i \le 8 \\ 0.8, & \text{for } \ i = 9 \\ 0.4 & \text{for } \ i = 10 \end{cases} \tag{5.3}$$

Combining the weighting factors of Sugamura and Farvardin, and Paliwal and Atal, a new very simple weighting function may be given by:

$$w_i = \begin{cases} 1.5, & for\ 1 \le i \le 2 \\ 1.0, & for\ 3 \le i \le 8 \\ 0.8, & for\ i = 9 \\ 0.4 & for\ i = 10 \end{cases} \tag{5.4}$$

It should be noted that this weighting function was *fixed* for all frames to avoid a high computational load.

The structure for the tree vector quantiser was chosen to be an unbalanced tree arising from single node splitting during the design process. In this model a node (cell) in the tree was split if it was the largest contributor to the overall average distortion for that stage of the design. Splitting is generally performed [78] by perturbing each component of the selected centroid by δ and $-\delta$ to obtain two new candidate centroids. The generalized Lloyd algorithm [78] is then run for each of the new pair of centroids till convergence is reached. This process was also used in this design which added one new node to the tree for each iteration. The tree was searched again to determine the largest contributor to the average overall distortion and the process of splitting and optimization repeated.

Makhoul [79] argues that the most cost effective search technique for a vector quantiser is the binary search with an unbalanced tree as its performance is close to full search. The advantage of using the unbalanced tree is that clusters with low number of members are not split unnecessarily, wasting bits, where the extra bits could be used on clusters with large populations; hence reducing the overall average distortion.

The overall algorithm for the design of the tree may be stated as follows:-

1. Find the initial centroid of the whole data training set τ.

2. Find the greatest contributor to the average overall distortion and split the node i. Run the generalized Lloyd algorithm to obtain two new centroids.

3. Partition the training set associated with node i into two reduced training sets τ_{i0}, τ_{i1} associated with each of the newly determined centroids.

4. Associate the centroids and corresponding associated training sets with an index and increment the number of nodes n.

5. Continue by returning to step 2 until the desired size n of the codebook is reached.

The splitting process may be described by taking a centroid of a cell τ and a suitably small scalar perturbation δ. The two new trial centroids τ_i^0, τ_i^1 were given by:

$$\tau_i^0 = (\tau_{i0}^0 + \delta, \tau_{i1}^0 + \delta, \ldots, \tau_{in}^0 + \delta) \tag{5.5}$$

$$\tau_i^1 = (\tau_{i0}^1 - \delta, \tau_{i1}^1 - \delta, \ldots, \tau_{in}^1 - \delta) \tag{5.6}$$

An analysis of the cells constructed by the tree structured vector quantiser, gain an insight to the details of the design process and hence construct a more efficient quantiser.

The first observation about the tree construction is that on each decision in the binary tree the search space is broken into closed half spaces. The spaces will always be closed as the vector components in the space $\Re^n$ are limited. This implies that each decision is a *refinement* of a cell and once a vector is found to lie in a decision region no other larger region is searched. This allows the process of, once the location of the half space boundary has been determined, to divide the training data for the original cell into the two new half spaces. It should be noted that once the training data has been associated with a cell of the quantiser it no longer plays any part in training other cells *except* further refinements of that cell.

A well known theorem of convexity theory [94] states:

Theorem 1 *If the set X is the non–void intersect of a finite number of closed half spaces and is bounded then X is a convex polytope.*

Further refinement of a quantiser cell generates a new convex polytope. To characterise the quantiser cell use was made of the covariance matrix defined in Chapter 3;

$$C(x) = \begin{pmatrix} E[(x_1 - m_1)^2] & E[(x_1 - m_1)(x_2 - m_2)] & \cdots & E[(x_1 - m_1)(x_N - m_M)] \\ E[(x_2 - m_2)(x_1 - m_1)] & E[(x_2 - m_2)^2] & \cdots & E[(x_2 - m_2)(x_N - m_N)] \\ \vdots & \vdots & & \vdots \\ E[(x_N - m_N)(x_1 - m_1)] & E[(x_N - m_N)(x_2 - m_2)] & \cdots & E[(x_N - m_n)^2] \end{pmatrix} \tag{5.7}$$

The training process was used to estimate the expectations $E[.]$ followed by a principal component analysis. An approximate hyper–ellipsoid model of the quantiser cell was used to describe its basic dimensional features and in particular the principal axis of the cell. Many of the details of the analysis have been left out and only the salient results are presented in what follows.

From the definition of eigenvectors,

$$\lambda x = Ax \tag{5.8}$$

Multiplying both sides by x^T, rearranging and finding the maximum eigenvalue;

$$\lambda_{max} = \frac{\max(x^T Ax)}{x^T x} \tag{5.9}$$

where the maximization is performed over the set of all possible eigenvectors. This expression indicates that the direction of maximum variance is associated with the principal eigenvector.

Gersho [80] has pointed out that the line joining the centroids of two adjacent cells is normal to and bisected by the hyperplane boundary between the two cells. Given this fact and that the cell was modelled by the hyper–ellipsoid a cut, in the direction of the principal axis, resulting in a two dimensional section was used to analyse the effects of quantisation distortion when centroid splitting occurred.

It was assumed that the data set fell within the ellipse and to reduce the quantisation distortion further this cell was split to create two new cells. It was assumed that the probability density was uniform across the ellipsoid.

The basic property of splitting to be determined was whether the splitting should occur in the direction of the principal axis to ensure that the two quantiser cells would quantise the data with minimum distortion.

Intuitively the boundary between the two new cells should run through the center of the ellipse; this will initially be assumed and then proven.

As the line between the centroids and the boundary between the cells are normal, and using rotational symmetry of the ellipse, the intersection of the two lines must be invariant to the rotation. The only point that satisfies this condition is the centre of the ellipse.

The ellipse under study is oriented with the major axis coincident with x axis and the minor axis with the y axis. The equation of the ellipse is given by,

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{5.10}$$

The boundary between the two cells, denoted the *bisector*, will be allowed to rotate and it will be shown that the only orientations allowed that are consistent with all of the criteria are the minor and major axes. The direction vector of the bisector s can be described in terms of a point on the ellipse, for arbitrary x_1 as,

$$s = (x_1, \frac{b}{a}(a^2 - x_1^2)) \tag{5.11}$$

The expression for the centroid $c = (\bar{x}, \bar{y})$, found by integration for arbitrary x_1;

$$\begin{aligned} \bar{x} &= \int_A x\,dA & (5.12) \\ \bar{y} &= \int_A x\,dA & (5.13) \end{aligned}$$

where A is the area of integration which applied to the ellipse gives,

$$\begin{aligned} \bar{x} &= \frac{2a^2b}{3} - 2ab\arcsin(\frac{x_1}{a})(\frac{b}{a}[(a^2 - x_1^2)^{3/2} - a^3 - \frac{x_1^2}{3}(a^2 - x_1^2)^{1/2}]) & (5.14) \\ \bar{y} &= 0 + 2ab\arcsin(\frac{x_1}{a})(\frac{b^2}{2a^2}[a^2x_1 - \frac{x_1^3}{3} - \frac{1}{3}(a^2 - x_1^2)^{3/2}]) & (5.15) \end{aligned}$$

The dot product of the direction of the centroid c and the bisector must be 0 for the two to be orthogonal.

$$s \bullet c = 0 \tag{5.16}$$

When this expression is written out explicitly a solution for the bisector in the vicinity of the minor axis occurs when $x_1 = 0$, that is the line between the centroids corresponds to the major axis. The solution for the x component is obvious and for the y component we note $\arcsin(0) = 0$.

Now for consistency of argument it is to be determined that a bisection of the ellipse normal to the direction of the major axis will lead to minimum distortion coding when the ellipse is equally bisected, assuming a uniform probability distribution. It is assumed that the bisection occurs at a distance δ from the origin of the ellipse. Due to the 180 degree rotational symmetry of the ellipse it is only necessary for δ to take on positive values. To evaluate the distortion in coding for the mean square error criterion, the equivalent problem of the second moment of area about the new centroid may be determined.

This may be expressed in symbols as;

$$\begin{aligned} I_x &= \int_A (y - \overline{y})^2 \, dA && (5.17) \\ I_y &= \int_A (x - \overline{x})^2 \, dA && (5.18) \end{aligned}$$

The primary interest is the second moment of area about an axis parallel to the y-axis of the ellipse, I_y.[2] The expression for I_y for arbitrary δ can be found to be;

$$\begin{aligned} I_y &= \frac{4b^2}{9a^2}\Big(\frac{(a^2-\delta^2)^3}{\pi a/2 - \delta/a(a^2-\delta^2)^{1/2} - a\arcsin(\delta/a)} + \\ &\quad \frac{(2a^3 - (a^2-\delta^2)^{3/2})^2}{\pi a/2 + \delta/a(a^2-\delta^2)^{1/2} + a\arcsin(\delta/a)}\Big) && (5.19) \end{aligned}$$

It can be shown that I_y has a minimum for $\delta = 0$. This implies that the ellipse should be cut exactly in half to minimize the distortion.

Returning to the original hyper–ellipsoid; any two dimensional section of the original volume that includes the major axis of the ellipsoid results in an ellipse of the form discussed above. Due to symmetry considerations, the analysis for the two dimensional case also covers the higher dimensional case and the same conclusions can be drawn. The primary result of interest is finding the major axis and the splitting the centroid along the direction of this axis.

The principal axis was found simply using an indirect method not dependent on finding the eigenvalues. The principal eigenvector represents the direction of maximum

[2]The ellipse at this stage is assumed to be symmetrical about the x-axis.

variance. Use is made of an argument put forward by Householder [95] for the case when,

$$|\lambda_1| > |\lambda_2| \geq \ldots \tag{5.20}$$

where λ_n are the eigenvalues. The principal eigenvector λ_1 can be found by continuing the Krylov sequence,

$$x_{i+1} = Ax_i \tag{5.21}$$

starting with any convenient vector x_0. For computational convenience scale factors were introduced to manage overflow problems. The expression now became,

$$\rho_{i+1}x_{i+1} = Ax_i \tag{5.22}$$

where $\|x_i\| = 1$ for some norm. The case where the principal eigenvalue was not unique was indicated by the iteration above not converging. This possibility was checked in practice by starting with two different x_0's and checking for convergence using a mean square error threshold, ϵ, for the two final vectors. If the two vectors did not converge to the same value, within the selected ϵ factor, the more traditional splitting technique [78] was used. Splitting along the direction of the principal component was effected as follows:-

Assume that the principal component vector is given by,

$$a = (a_0, a_1, a_2, \ldots, a_n) \tag{5.23}$$

and a suitably small perturbation factor δ is chosen. The new trial centroids τ_i^0, τ_i^1 are given by,

$$\tau_i^0 = (\tau_{i0}^0 + \delta a_0, \tau_{i1}^0 + \delta a_1, \ldots, \tau_{in}^0 + \delta a_n) \tag{5.24}$$
$$\tau_i^1 = (\tau_{i0}^1 - \delta a_0, \tau_{i1}^1 - \delta a_1, \ldots, \tau_{in}^1 - \delta a_n) \tag{5.25}$$

It was found, referring to the following section, that slightly different codebooks were constructed with slightly different performances. It was not surprising that the two codebooks were similar as the generalized Lloyd algorithm moves the centroid to local optimum for both cases of splitting which is more dependent on the training data than the position of the initial centroids. Also note the characterization of the cell as a hyper-ellipsoid for the analysis above is an approximation to the true situation.

5.2.1 Initial Tree-Structured Vector Quantiser Design

Five vector quantiser design cases were investigated:

1. Mean Square Error distance measure with traditional splitting of centroids,
2. Atal/Paliwal Weighted Mean Square Error distance measure with traditional splitting of centroids,
3. Simple Weighted Mean Square Error distance measure with traditional splitting of centroids,
4. Atal/Paliwal Weighted Mean Square Error distance measure with principal axis splitting of centroids,
5. Simple Weighted Mean Square Error distance measure with principal axis splitting of centroids.

The simple weighting mentioned above was given by Equation 5.4. The original database of Line Spectral Frequency (frequency version of LSP's) had to be expanded to incorporate a greater variation of speakers to allow larger more meaningful codebooks to be constructed. The initial database consisted of two females and two males each uttering two Harvard test sentences resulting in approximately 40 seconds of speech.

The database was expanded to include:

1. The original 4 speakers,
2. Two female and two male speakers uttering a single test sentence (approximately 10 seconds of speech),
3. Two male and two female speakers uttering 3 Harvard test sentences each (approximately 54 seconds of speech).

The final database consisted of just over 100 seconds of speech uttered by six different females and six different males giving 3400 ten dimensional LSF vectors. Training ratios (the number of training vectors to codebook vectors) used by various authors vary from 50:1 to 15:1. Paliwal and Atal [36] for their work had access to a database of 60000 LSF vectors and used a training ratio of approximately 15:1. The database available for this work allowed a training ratio of approximately 14:1 for a 256 entry codebook and approximately 7:1 for a 512 entry codebook. The 512 entry codebook results consequently must be viewed with some suspicion. The results for the tree vector quantiser design are shown in Table 5.2.1. The maximum depth (Max Depth) refers to the maximum depth that the tree grew to, due to the unbalanced tree design employed, and gives an upper limit to the computational load required in the encoding process. The size of the codebook (Size) was chosen to be a power of 2 to make optimum use of the codebook index bits transmitted over the channel. Notations *mse*, *w-ap* and *w-s* were used to indicate: mean square error, weighted mean square error using Atal/Paliwal weighting, and simple weighting respectively. The terminology *p-axis* and *norm* refer to the splitting algorithm used in the construction of the tree. *p-axis* refers to a principal axis analysis performed at the node to determine the direction of splitting while *norm* refers to the normal or more generally used splitting method [78].

Also tabulated are the outliers associated with quantising using the tree quantiser. The first percentage is for outliers lying between 2–4 dB spectral distortion and the second for outliers > 4 dB.

It can be seen that the principal axis analysis improves performance marginally in some cases, and also it is dubious if the weighting assists. The principal axis analysis does give some valuable information however and this will be discussed later.

Max Depth	5	6	7	9	11
Codebook Size	32	64	128	256	512
w-s p-axis	3.630	3.323	2.996	2.683	2.341
Outliers	49% 38%	54% 29%	57% 19%	57% 12%	53% 6%
w-ap p-axis	3.619	3.320	2.994	2.7033	2.364
Outliers	50% 38%	54% 28%	56% 20%	58% 12%	52% 6%
mse norm	3.605	3.294	3.003	2.685	2.346
Outliers	49% 38%	52% 29%	58% 19%	58% 11%	54% 5%
w-pa norm	3.600	3.307	3.003	2.692	2.367
Outliers	50% 37%	55% 28%	58% 19%	57% 12%	53% 6%
w-s norm	3.624	3.303	2.990	2.693	2.343
Outliers	48% 38%	54% 28%	58% 19%	59% 11%	53% 6%

Table 5.1: Spectral Distortion and Percent Outliers for the Tree Structured Quantiser.

These values should be compared to the full search codebook to compare the loss in using a tree searched quantiser as against the full search technique. Table 5.2.1 gives the results of the full searched codebook and Figure 5.1 shows a graph comparing the average spectral distortions in dB for the two techniques.

Codebook Size	128	256	512
Outliers	61% 14%	61% 6%	49% 2%

Table 5.2: Spectral Distortion and Percent Outliers for the Fully Searched Quantiser.

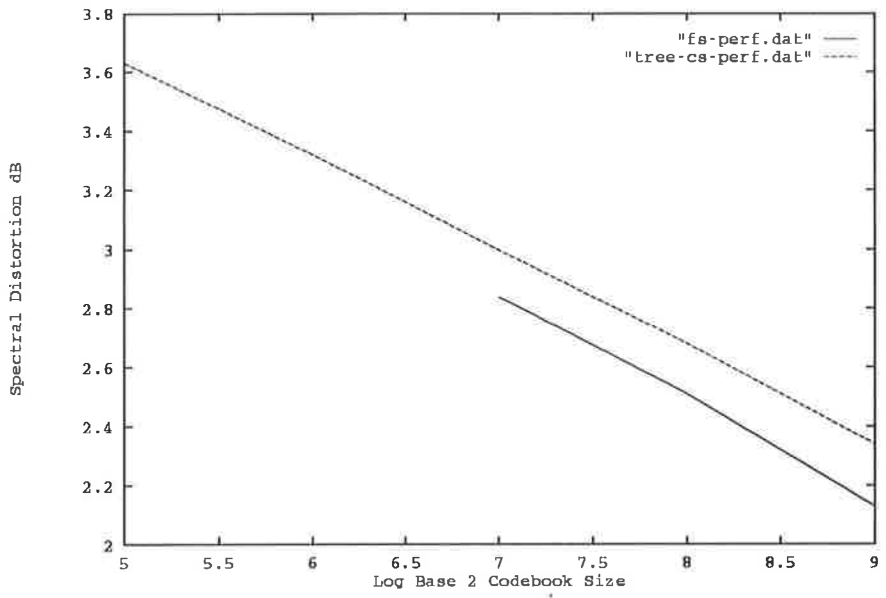


Figure 5.1: Average Spectral Distortion versus Log2 Codebook Size for Tree and Full Search Vector Quantisers: *fs* denotes full search and *tree* tree search.

A simple linear extrapolation on the weighted mean square error cases indicate that a codebook size of approximately 32000 vectors would be required to obtain transparent LSP encoding, that is, less than 1% average spectral distortion with less than 2% outliers in 2-4 dB range and none greater than 4db. It was noted that Atal and Paliwal [36] use of the order of 25 bits for their vector quantiser. Firstly it should be considered that the above estimates were optimistic due to the size of the training set, although the performance was also tested on an out of training set of 3500 vectors. Paliwal's database was also constructed using a stabilized covariance method to obtain the LPC coefficients including a 0.996 (10 Hz) bandwidth broadening factor. The database for these experiments was constructed using the windowed autocorrelation approach with a (15 Hz) bandwidth broadening factor together with the further restriction that no two Line Spectral Frequencies were allowed to be closer than 15 Hz. Spectral distortion was measured relative to the processed source data rather than the original unprocessed source LSP's.

It may be noted from Figure 5.1 that little performance was lost in using a tree search strategy in terms of quantisation performance.

The general format of a tree structured vector quantiser is that the index that is transmitted is built up sequentially from decisions made in going through the tree. Consider a binary tree; if an *upper* branch was chosen the bit in the index is set to a 0, else if the *lower* branch was chosen it is set to a 1. The next decision level is concatenated to the string of previous decisions. At the last decision the string representing the decision path is completed which is also the index to be transmitted over the channel.

Using the single node splitting design the tree may be unbalanced (general case) and as such the above technique would not work. Instead an index to be transmitted over the channel was assigned when the leaf node was reached. Each leaf node was given a unique index and at the receiver end the reproduction vector was determined from the received index and a look up table which provided a pointer into the tree-structured codebook. If the tree is drawn as a flat graph on paper the indexing could conceptually be assigned as consecutive numbers in going from left to right past the leaf nodes on the tree.

5.3 Vector Quantiser Design for Noisy Channels

Various authors have studied the problem of combined source/channel codebook design, for example Zeger [42] and Farvardin [39] [40].

Consider a k-dimensional vector quantiser as a mapping $q(.)$ that assigns a vector y to a vector x, $y = q(x)$. The vector $\mathbf{y}$, drawn from a finite alphabet codebook $\mathbf{C}$, becomes the reproduction vector for $\mathbf{x}$. The codebook $C = \{c_1, c_2, \ldots, c_M\}$ is associated with a partition $\mathbf{P}$ of the vector space. The partition is described by $P = \{S_1, S_2, \ldots, S_M\}$ where S_i are sets.

The mapping $q(.)$ is described by,

$$q(x) = c_i \quad \text{if } x \in S_i \quad i = 1, 2, \ldots, M \tag{5.26}$$

Now it is necessary to define a distortion measure so that it is possible to measure the distortion incurred by representing the vector $\mathbf{x}$ by $\mathbf{y}$. Let the distortion measure be represented by $d(x,y)$. The *average distortion* per vector is given by,

$$D_s(q) = \frac{1}{k}\sum_{i=1}^{M}\int_{S_i} p(x)d(x,c_i)dx \tag{5.27}$$

where $p(\mathbf{x})$ is the k-fold probability density function of the source. The rate R of the source coder is given by

$$R = \tfrac{1}{k}log_2 M \quad \text{bits/sample.} \tag{5.28}$$

For a given source and rate R a vector quantiser is optimal if there is no other mapping $q(.)$ that has a lower average distortion. The locally optimal lbg algorithm [78] gives a good codebook. In a practical communications system the codewords c_i are mapped into binary words (generally of fixed length) to be transmitted over the channel. Let us use the notation $b(c_i)$ for the assignment of an index to a codeword. Given this information the average distortion can be determined. For a fixed codebook, the distortion caused per reconstructed source sample due to channel errors on the transmitted index may be given by,

$$D_c(b) = \frac{1}{k}\sum_{i=1}^{M}\sum_{j=1}^{M} P(c_i)(b(c_j)|b(c_i))d(c_i,c_j) \tag{5.29}$$

where $P(c_i)$ is the *a priori* probability that codeword c_i is chosen.

The overall distortion caused by the vector quantiser and the channel is given by,

$$D(q;b) = \frac{1}{k}\sum_{i=1}^{M}\sum_{j=1}^{M} P(b(c_j)|b(c_i))\int_{S_i} p(x)d(x,c_j)dx \tag{5.30}$$

which in general is not equal to the sum of the source encoder distortion D_s and the channel distortion D_c.

If the distortion criterion is the squared error criterion then it can be shown that [39] [42],

$$D(q;b) = D_s(q) + D_c(b) \tag{5.31}$$

which is an extension of the scalar case.

To use this property the assumptions that must be made are: a squared error distortion measure and that the codevectors are given by the centroids of their respective regions S_i.

Farvardin [39] [40] goes on to the more general case where the codebook and channel conditions are jointly designed. Rewriting equation 5.30 by interchanging summations and integrations,

$$D(q;b) = \frac{1}{k}\sum_{i=1}^{M}\int_{S_i} p(x)\left\{\sum_{j=1}^{M} P(b(c_j)|b(c_i))d(x,c_j)\right\}dx \tag{5.32}$$

This expression can be seen to imply a vector quantiser design with a modified distortion criterion. Farvardin calls these vector quantisers channel optimized vector quantisers (COVQ). Results also show that COVQ will out perform index optimized vector quantisers [42].

In the present application there was a problem with COVQ in that the algorithm is for an unstructured codebook and it was not straight forward how these principles may be carried into the tree structured design. Recall the tree structured design was deemed necessary to speed up the encoding of source vectors in large codebooks[3]. Another problem arises in that the COVQ trades off overall channel performance for less accurate quantisation at the source end. Farvardin and Vaishampayan [40] showed that as the noise level increased (greater probability of channel error on a binary symmetric channel) the number of non-empty encoding regions decreases.

Generally mobile communications channels have a non-constant signal to noise ratio and have times when the error rate is very low. If the COVQ approach was used, the less accurate quantisation may be significant during periods of low noise. This implied that to get transparent coding of LSP's at low error rates the codebook needed to be substantially larger. Another approach suggested itself where the codebook was designed using the lbg algorithm and a codeword/index assignment space larger than the cardinality of the codebook was used. A codeword was chosen and the associated index was transmitted. The corrupted index was mapped back to a codeword/vector which was used as the reproduction vector.

Consider a concrete example. A codebook of $N = 2^n$ entries was constructed. An index space of $2N = 2^{n+1}$ was chosen. When a particular codeword c_j was chosen a unique index was assigned $b(c_j)$ from the index space. Only N of the possible 2N indices were required for this task. On transmission the index may have been corrupted into any one of the 2N possible indices. At the receive end a 2:1 mapping occurs back to the reproduction codebook of size N. In the noiseless channel case, the performance would be equal to the noiseless codebook design. By using a counting argument there are $N!$ ways of permuting the indices for the original book. For the new book there are $2N!$ ways of permuting the indices but as each vector was represented twice by two

[3]A large codebook is arbitrarily defined as 4096 entries or greater.

different indices, the swapping of those indices produces the same result at the receiver. Counting all interchanges of pairs one gets 2^N interchanges. The total number of unique permutations P_u for the reception space thus becomes,

$$P_u = \frac{2N!}{2^N} \tag{5.33}$$

This is a large number for large codebook sizes and intuitively one may suspect an improved performance. However simulations over noisy channels always indicated that a permutation of the original codebook indices performed better than the expanded index space method. If one or more index bits were error protected, coupled with permutation superior results were obtained. It appears the addition of the extra bit exposed to channel errors never paid off. The results of transmission over noisy channels is covered in more detail after the section on implementation details of index permutation.

5.4 Genetic Algorithms

Genetic Algorithms (GA's) [91] are a method of guided random search used for non-linear function optimization. Another random search technique that is quite popular is *simulated annealing.* The term random search does not imply that the search is a directionless search, however it is not a gradient technique in the standard calculus sense. Gradient techniques are good on a narrow class of problems and will in general arrive at an answer faster than the random searches, but they also have a high chance of falling into local minima. From experience it was found that genetic algorithms produced a good solution fairly quickly when compared to simulated annealing, although an *optimal* solution may take just as long.

To describe Genetic Algorithms (GA's) emphasis is placed on how they differ from traditional optimization methods.

1. Traditional methods work on the actual parameters in a parameter set, while GA's work on a coding of the parameter set (usually a string of 1's and 0's). The length of the string depends on the number of parameters and the resolution required.

2. GA's always work from a population of points (strings) rather than a single point. The population size may vary from problem to problem.

3. GA's use a cost function that is evaluated for a complete string. The method is blind in that only the cost function for the complete string is used without any

extra knowledge of the function space in terms of derivatives or other auxiliary information.

4. GA's use probabilistic transition rules to move from point (string) to point (string). Note, *simulated annealing* also uses probabilistic transition rules rather than deterministic rules.

GA's work with a coding of the parameter set (usually a binary coding) to produce a string. Rather than use a single string a whole population of strings is used. Associated with each string is a cost function value. The GA will try to optimize the cost function over the entire (encoded) parameter set simultaneously. The value of any parameter may be obtained at any time by going to the correct string position and decoding it. A direct correspondence is made between a parameter and a string position. Any single parameter may be encoded as any number of bits.

Having established that the GA works with a set of strings of encoded parameters it is now necessary to discuss how transitions are made. A key concept in GA's is that they move from one generation to the next, hopefully producing a new generation more fit or having a better cost function value (on average) than the previous generation.

The standard simple GA has three basic probabilistic operators:

1. Reproduction,
2. Crossover,
3. Mutation.

Each of these operators will be described in turn.

Reproduction is based entirely on the fitness of the string. A sum of all the fitness values for a generation is calculated giving a total fitness. Each individual's fitness is then calculated as a percentage of the total fitness. A mechanism akin to a roulette wheel with the slots proportional to each individual's fitness percentage is used for selection; where the "ball" lands, that individual is reproduced for the next generation. The roulette wheel is spun for each new member of the following generation. This method ensures that the fittest strings survive to the next generation but also allows other *genetic* material to survive and be passed on.

The next phase is *crossover*. This mechanism is similar to the concept of breeding. Strings are randomly selected according to some probability level and placed into a mating pool. Pairs of strings are mated at random. The mating process consists of finding a random crossing site (cs) between the fist position on the string and the last; $cs \in [1, \ldots, l-1]$ for a string of length l. The partial string after the crossing site is then swapped between the two mating members. Consider an example;

$$\begin{aligned} S_1 &= 010011|10100 \\ S_2 &= 101010|11001 \end{aligned} \tag{5.34}$$

After crossover at the marked site;

$$\begin{aligned} \hat{S}_1 &= 010011|11001 \\ \hat{S}_2 &= 101010|10100 \end{aligned} \tag{5.35}$$

is obtained.

The operations of reproduction and crossover are very simple operations involving nothing more than string copies and partial string swapping. These operations are the primary GA operators. *Mutation* is a secondary operation which consists in the simplest case of randomly swapping a 1 for a 0 or vice versa within a string. Mutation operates with a very low probability of occurrence $\approx 1-3\%$ on all of the strings in the new population. This operation ensures that *genetic* material is not lost forever within the population and may introduce new genetic material.

The reasons why this scheme works is subtle and more information can be found in Goldberg's book [91]. In summary the algorithm strives toward a more fit population as poor performers (lethals) have a low probability of contributing to the next generation. Higher performing string segments tend to propagate from one generation to the next.

5.5 Genetic Algorithms Applied to Vector Quantiser Design

Having introduced the concept of Genetic Algorithms (GA's) the adaption to solve the vector quantiser design for a noisy channel is now discussed.

The simplest problem to solve was how best to arrange the transmitted indices such that, should a channel error occur, the distortion due to an incorrect codevector was minimized. This problem was basically a permutation type problem. The encoding of the strings for the GA will be discussed first followed by the development of the cost function.

The reproduction phase of the GA did not pose a problem as it only copied strings. One of the major problems with GA's is the encoding of the parameter set. In the general case following Goldberg's book [91] the indices for the vector quantiser codebook would be represented as a binary code. However for this problem the value of the parameter (codebook index) was not to be changed, only the order in which the parameters appeared in the string positions. Uniqueness of each index must be assured and all indices must be present in each string. A general argument in the genetic algorithm literature is that the best encoding of the parameters is to use a binary string, but in this permutation case it was of no advantage as only the ordering of the indices was important. It was considered undesirable that crossovers between strings occurred inside the representation of a codebook index. To reduce the computational load, the string was constructed from an alphabet of the same cardinality as the index space size. If there were to be 32 different indices on the channel each index from 1 to 32 appeared once and only once in a random place in each string. Each genetic string had to contain the complete set of codebook indices. The vectors in the codebook remained stationary and did not alter their relative positions.

Crossover however did pose a problem because if partial strings were swapped without attention to the problem of replication and deletion of indices from the *genetic* string an invalid permutation resulted. This problem was addressed by various authors where one of the most successful techniques was developed by Goldberg and Lingle [96]. The method they developed was called Partially Matched Crossover (PMX). Under PMX two strings are aligned and two crossing sites are randomly determined. The two crossing sites define a matching section that is used in a position by a position exchange operation.

Another *new* crossover technique was developed by the author that ensured that absolute positions within the string were maintained with as much integrity as possible. The scheme finds two crossing sites at random for two strings that are aligned, as indicated in the binary example above. Crossover occurred just as in the simple genetic algorithm except now three tasks had to be performed on the resulting string: the missing indices in the string were determined, the duplication of indices were found, and for each duplicate pair one was chosen randomly and replaced by one of the missing indices. This scheme dubbed PMX2 was trialled with a simple sorting/permutation problem to determine which produced the correct answer with fewest computations. The simple sample problem was to sort 10 indices from an initial set of random permutations.

$$S_{sorted} = \{1, 2, 3, \ldots, 10\} \tag{5.36}$$

$$S_{random} = \{p_1, p_2, p_3, \ldots, p_{10}\} \tag{5.37}$$

The cost function was a square error of index value from the correct position,

$$f = \sum_{i=i}^{10} (i - p_i)^2 \tag{5.38}$$

It was found on repeated trials that PMX2 performed better than PMX for each trial.

Mutation was retained as a secondary operator in these experiments. It is not always used in permutation type problems. The mutation operation was simply to swap random pairs within a string. This scheme was simple to implement and maintained population diversity towards the end of an optimization run.

Another major issue was to determine the cost function. The candidate cost function was developed from Equation 5.29. For each vector in the codebook the corresponding index was compared with the indices of every other codevector. A variable Hamming distance[4] threshold was set to limit the number of errors that the system would be trained for. If a distance less than or equal to the threshold was found, the associated reproduction vector was extracted and subsequently the spectral distortion between the reproduction vector and the transmitted (reference vector) calculated. (A Hamming distance of two implies two errors, etc.) The spectral distortion for a particular reference vector was weighted by the probability that the reference vector was chosen from the book in the encoding process. It was also weighted by the binary symmetric channel probability of error for the Hamming weight found between indices; which must have been less than or equal to the threshold. This implied that the index assignment was determined for a particular number of allowed errors and for a particular channel error rate. As an example, a threshold may be set for up to three bit errors, on a binary symmetric channel with bit error rate 0.1.

The raw fitness values were processed before reproduction in order to eliminate *lethals* (those individuals with a particularly poor fitness score). This idea follows a reference in Goldberg [97] to work done by Forrest. The mean and standard deviation of the population fitness values were determined and a new fitness $\acute{f}$ for each string determined.

$$\acute{f} = f - (\bar{f} - c\sigma) \tag{5.39}$$

Where f is the fitness value, $\bar{f}$ is the mean fitness, σ is the standard deviation and c takes values between 1 and 3. In these experiments a value of 2 was chosen. Any fitness

[4]That is the Hamming distance between the binary representations of the codebook indices.

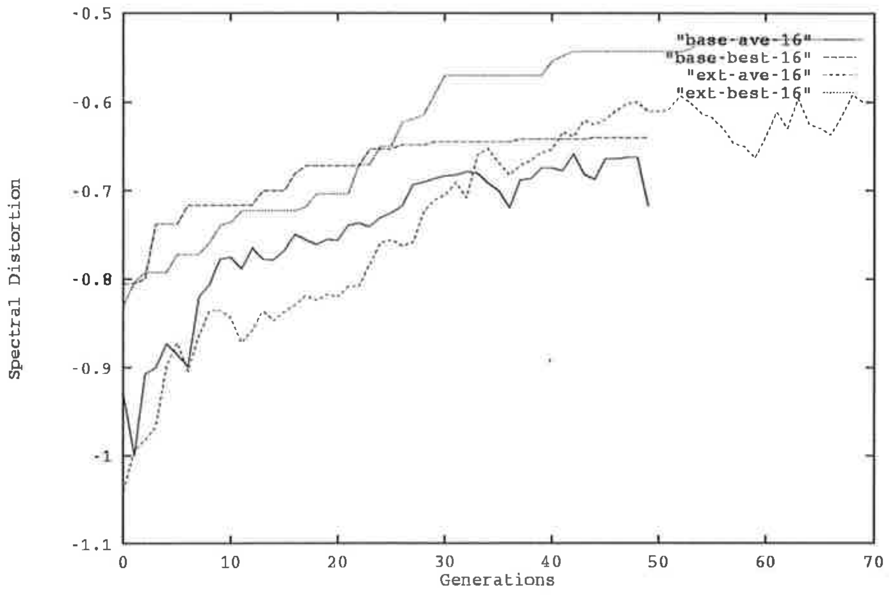


Figure 5.2: Weighted Negative Spectral Distortion versus Generation Number: *best* denotes the best of the population, *ave* denotes the population average.

value that was negative was set to zero, thus eliminating it from the reproduction step. This method had the effect of eliminating the worst individuals from the population.

For the case of designing a channel index assignment where the number of indices was greater than the number of codevectors certain constraints were placed on the structure. Each genetic string was twice the length required to represent the number of codewords. A strict 2:1 mapping was ensured by mapping the position in the string modulo the code-book-size to a codevector. For example if the codebook had 16 vectors the channel index assignment was in the range 0-31. A genetic string consequently had 32 positions. The 3rd position maps to the third codevector as did the 19th. The genetic algorithm tried to arrange the indices such that channel errors giving a false index, (*index-dash*), resulted in a mapping back to the codevector such that the distortion on reconstruction of the coefficients was minimized.

Figure 5.2 shows the results of the genetic algorithm optimization problem described above for a codebook of 16 entries. Note that the top of the graph represents better performance. The plots are of the purely permuted case *base* and the extended index space case *ext*. Also the current best solution (best in the population) and the population average are shown for both cases. Table 5.3 illustrates the results of using the designed codebooks of 64 and 128 entries over a simulated noisy channel for the binary

Error Rate	$\epsilon = 0.1$		$\epsilon = 0.05$	
Codebook Size	64	128	64	128
Ave. SD unperm	3.984	4.639	2.301	2.6943
Ave. SD perm	3.229	3.995	1.784	2.294
Ave. SD ext	3.700	4.522	2.106	2.636

Table 5.3: Average Spectral Distortion (SD) in dB for the Specified Error Rates.

symmetric channel error rates as specified.

The Table 5.3 shows the results in terms of spectral distortion (SD) in decibels due to reception of incorrect indices when transmitting vector indices over a binary symmetric channel with bit error rate 0.1 and 0.05. Each codebook was specifically designed for the given error rate. The term *unperm* refers to the index assignment that resulted from the original codebook design, *perm* to the "optimal" index assignment found for the natural size of the codebook and *ext* to the case of the "optimal" extended index space. The results from the permutation training in the genetic algorithm did not reflect the actual performance on a noisy channel.

It should be noted that the counter-intuitive result of the larger codebook giving inferior performance comes about due to the larger number of bits exposed to error as they are transmitted across the channel. If the number of bits likely to be in error is calculated for the 64 entry codebook with the 3400 test vectors and an error ratio of 0.1, approximately 2040 errors may be expected. For the 128 entry codebook indices transmitted over the same channel 2380 errors may be expected. The ratio of the spectral distortion for the 128 entry codebook to the spectral distortion for the 64 entry codebook is very close to the error ratio. The values obtained for the permuted indices were not as close to this ratio and which may be put down to two factors: the permuted indices were arranged under the assumption of a single error within an index, and the permutation was not globally optimal. At the higher error rates assumed here there was a reasonable chance that more than one error was found per index.

The effect of using a redundant bit to allow the permutation of indices to be more effective has been shown to be counter productive. In all cases studied the extended codebook performed worse than using the original size codebook and permuting the indices.

To improve the effectiveness of the permutation of the indices, an unequal error protection scheme was also incorporated. Results for this scheme are covered in the next section.

5.6 Simulation Results of a Protected Vector Quantiser on Noisy Channels

The permutation of the indices gave an improvement of approximately 0.5 dB over noisy channels with respect to the original unpermuted indices. To gain more performance it was necessary to provide error protection. Rather than protect the complete index it was possible to protect a part of the index, and use permutation of the indices under the assumption that some of the bits were protected.

On fading channels, the signal to noise ratio may fall to very low values. It was desirable to use a block code in the investigated application due to the blocked nature of the speech parameter extraction process and subsequent transmission thereof. This implied that a block code which could be decoded using soft decisions in a straight forward manner was desirable. Soft decision implies rather than using one threshold to select a 1 or a 0 on reception a set of thresholds are used. Theory says that if an infinite number of thresholds are used, performance should be 3 dB better than the single threshold case. If three bits are used (7 thresholds) the performance improvement is of the order 2.25 dB. This allows operation further into the noise.

The Golay code using the Chase algorithm (soft decision) was chosen as a candidate block code, for the unequal error protection. Clark and Cain [98] provide a good chapter (Chapt. 4), on soft decision decoding of block codes and also provide performance curves for soft decision decoding of the Golay and other codes.

To test the idea of using unequal error protection and permutation of the indices certain assumptions were made. The (24,12) Golay code was chosen for protecting some of the index bits and the Chase algorithm for decoding the received codewords. If 150 bits are assumed in a block of coded voice parameters the addition of unequal error protection of the type mentioned would increase the block size to 162 bits. This implies a 0.33 dB shift to the left on the *energy per bit to noise ratio* axis of the performance curves of the Chase algorithm (note this is the compensation factor for having to signal marginally faster for the same information transmitted per unit time). Using the above assumptions, if the channel bit error rate was $\epsilon = 0.06$ then the protected bits have a

Codebook Size	unperm	perm	ext perm
32	0.980	0.622	1.205
Outliers	0.1% 12%	4% 8%	3% 18%
64	1.409	1.022	1.872
Outliers	0.4% 16%	3% 14%	3% 24%
128	1.988	1.587	2.489
Outliers	0.7% 22%	3% 20%	3% 29%
256	2.505	2.292	3.037
Outliers	0.9% 27%	2% 26%	3% 33%

Table 5.4: Average Spectral Distortion in dB Due to Channel Errors ($\epsilon = 0.06$) with Three Index Bits Protected.

bit error rate $\acute{\epsilon} = 0.0034$

Tables 5.4 and 5.5 show the performance of various size codebooks designed with the assumption of a channel bit error rate of $\epsilon = 0.06$ with 3 bits error protected, except in the case where an extended codebook was used where only two bits were protected. This ensured that the same number of redundant bits were used in both scenarios.

Due to the inferior performance of the extended codebook, the extended codebook was dropped from consideration for the lower bit error rate case. Note, Table 5.5 used the same codebooks as those above, designed for the higher error rate channel, but subjected it to the lower error rate of $\epsilon = 0.03$.

The average spectral distortion against code book size is shown in Figure 5.3. The results in Tables 5.4 and 5.5 appear counter intuitive but it should be noted that no normalisation was performed. The ratio of protected bits to unprotected bits drops as the code book size increases and also the code book indices increase in size thus being exposed to more errors. If these two ratios are accounted for, the performance of the unpermuted index code book remains fairly constant whilst the permuted code book performance decreases. It appeared the larger the code book, the more difficult it was to find an optimal permutation.

There is a trade-off between error performance and the number of bits error protected. Any extra performance gained by index permutation is obtained for *free* without an increase in required channel bandwidth. An experiment was performed to determine the performance of the 128 entry codebook with an increasing number of bits protected.

Codebook Size	unperm	perm
32	0.523	0.323
Outliers	0.03% 7%	2% 4%
64	0.740	0.537
Outliers	0.3% 9%	1% 7%
128	1.018	0.798
Outliers	0.3% 11%	2% 10%
256	1.292	1.189
Outliers	0.7% 14%	1% 14%

Table 5.5: Average Spectral Distortion in dB due to Channel Errors ($\epsilon = 0.03$) with Three Index Bits Protected.

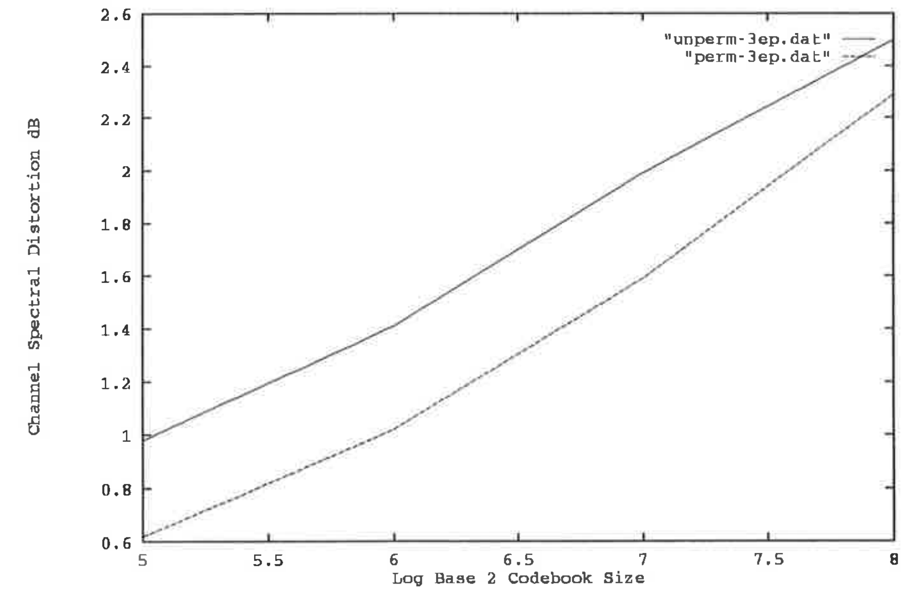


Figure 5.3: Average Spectral Distortion Versus Codebook Size for 3 Bits of the Index Error Protected: *unperm* denotes the unpermuted index case and *perm* denotes permutation of the indices.

No. Error Protected Bits	Ave. Spectral Distortion	Outliers 2–4 dB	Outliers >4 dB
0	3.100	1%	34%
2	2.389	2%	25%
3	1.587	3%	20%
4	1.388	1%	17%
5	1.022	1%	13%

Table 5.6: Average Spectral Distortion in dB due to Channel Errors ($\epsilon = 0.06$) with a Variable Number of Index Bits Protected.

Table 5.6 shows the results.

The values in Table 5.6 are shown in the graph Figure 5.4. It should be noted that the points form an upper bound on the actual performance which would above this curve. The code book of 128 entries required 7 index bits to be transmitted and from the graph approximately 4 bits protected appeared to be (the knee of the curve) a reasonable trade-off between distortion performance and number of bits protected.

The overall conclusion is that for transmission over channels with low signal to noise ratios it is desirable to protect most of the index bits, and allow permutation of index bits to provide distortion minimization in the case of errors in the unprotected bits.

5.7 Split Vector Quantisers

The split vector quantiser is a compromise on the ideal full search vector quantiser. The computational load $\tilde{C}$ and memory requirements $\tilde{M}$ for the full search vector quantiser is of the order;

$$\begin{aligned} \tilde{C} &= NL \\ \tilde{M} &= NL \end{aligned} \tag{5.40}$$

where N is the dimension of the vectors and L the number of entries in the codebook. The binary tree structure search solves the search problem but slightly aggravates the memory storage problem. The orders of magnitude (note for an unbalanced tree these equations give approximate estimates) are given by

$$\tilde{C} = 2Nlog_2L$$

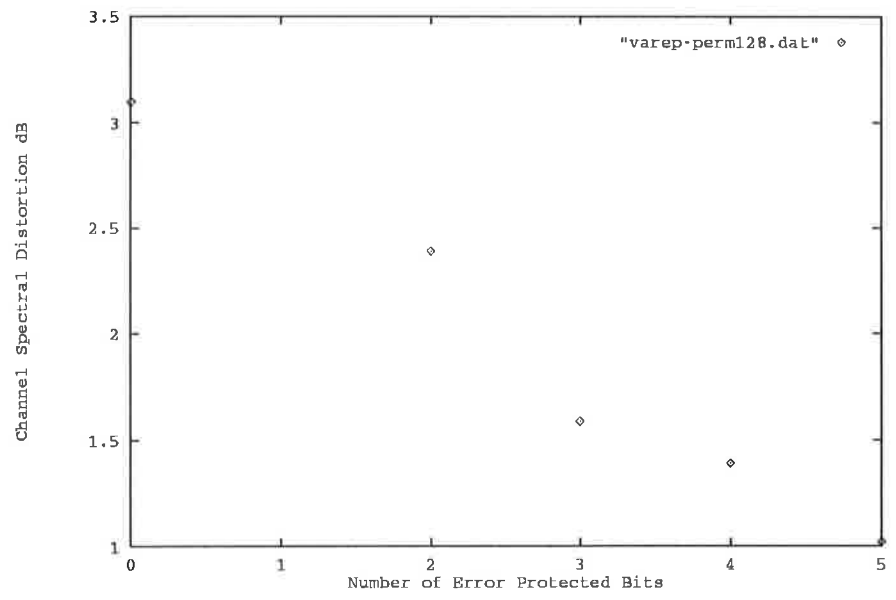


Figure 5.4: Average Spectral Distortion Versus the Number of Error Protected Bits.

$$\tilde{M} = 2N(L-2) \tag{5.41}$$

Clearly trees solve the search problem but do not aid the storage and attendant training problems of a large codebook. Atal and Paliwal [36] note that for transparent coding of LSP's 24–26 bits are required. Obviously the memory requirements for a tree structured codebook of the type above was out of the question. Another method of constructing the codebook had to be sought. The two most likely candidates were:

Cascade Codebooks. This technique is also called Multistage Quantisation. Multistage quantisation uses cascaded codebooks, as the name implies, where the input vector is crudely quantised with the first codebook and then a residual error vector is formed from the original input vector and the selected first stage vector. The residual error vector is then quantised from the next codebook. The channel information then comprises two indices which are used on reconstruction to generate two codevectors which are component-wise summed to give the final reconstructed vector.

Split Vector Codebooks. This method splits the vector to generate two or more vectors of smaller dimension than the original vector. Each vector portion is individually quantised and the indices from each codebook transmitted over the channel. For reconstruction the vector parts are recovered as codevectors and concatenated to reproduce the estimate of the original input vector.

The two alternatives are discussed in more detail below to determine the best path for future development.

The cascade arrangement used the generalized Lloyd (lbg) algorithm to design each stage, where the original input vectors were used for the first stage and the resulting residual vectors were pooled to form the training set for the second stage. The burden for the cascade codebooks (assuming two stages) was

$$\begin{aligned} \tilde{C} &= N(L_1 + L_2) \\ \tilde{M} &= N(L_1 + L_2) \end{aligned} \tag{5.42}$$

Where $L = L_1 L_2$. This compares to NL_1L_2 for the full search case. If the tree search strategy is used in conjunction with this structure the burdens become;

$$\begin{aligned} \tilde{C} &= 2Nlog_2L_1 + 2Nlog_2L_2 \\ \tilde{M} &= 2N(L_1 + L_2 - 4) \end{aligned} \tag{5.43}$$

This on the surface appeared to be good, however there were some significant problems. The main problem was the performance degradation. This occurred because the pooling of error residuals and construction of a subsequent codebook assumed that the relative probability density function within all clusters (cells) was the same. If indeed this was the case degradation would be small. Vector rotations may be used to remove linear dependencies and thus make the probability density functions for each cluster appear similar, but this implied a greater computational cost (proportional to N^2) and also memory storage for the rotation matrices of magnitude;

$$\tilde{M} = N^2 L_1 \tag{5.44}$$

Once the quantised vector was found in the residual error codebook it had to be rotated prior to its addition to the previous stage.

Work by Lee et. al [99] looked at making the multistage or cascade configuration more efficient. They argued that rotations were not really required but only scaling. This followed from a conjecture by Gersho [81] that the cells of the best quantisers are all of approximately the same shape (the tessellating polytope with minimum normalized moment of inertia). This also implies that they are close to spherical. Thus the only difference between the polytopes is that they vary in size and orientation. However the near spherical shape should remove the need of rotation leaving only the scaling factor.

Gersho [100] noted that in the cascade arrangement the vector components of successive stages tended to be less correlated and less statistically dependent than those of the input vector. This implies in the limit that the error stage is asymptotically a uniform vector quantiser such as a lattice. The analysis of Lee et al. [99] shows that this is the case.

Whitening of the error vectors implies that the procedure for allocation of indices to code vectors to combat channel noise will have reduced effectiveness because in a uniform quantiser, in the white noise case, all cells are chosen with equal probability and all adjacent cells have the same distortion metric between them.

Split vector quantisation was considered next. There are certain restrictions on where a split vector quantiser can be used. It is desirable that the two (or more) vector components can be quantised independently. For this to be true the distortion measure over the entire vector must be separable. This is true for mean square error and derivatives of it. Generally the codebooks cannot be designed independently unless the distortion measure is separable and the component vectors are statistically independent. Even if the distortion measure is separable a reduction in performance compared to the full search codebook would have to be predicted because of statistical dependence between vector components and also the reduction of dimensionality of the vector quantiser space.

Paliwal and Atal [36] noted that the spectral sensitivities of line spectral frequencies (LSF's) were localized which means there was little spectral leakage from one region to another. This makes LSF's ideal candidates for a split vector quantiser. The localized sensitivities also allows weighting of the LSF components allowing weighted mean square error distortions. It should be noted that Log Area Ratio and ArcSine Reflection Coefficient representations [79] of the LPC coefficients do not have this property. Paliwal and Atal's split vector quantiser was split into two parts, each with the same number of bits, and the first part comprised LSF's 1-4 and the second LSF's 5-10. This choice appeared to be an intuitive optimum configuration as the first 4 LSF's are around the principal formant region of speech to which the ear is most sensitive. However an experiment on the data set available was undertaken to verify if the results obtained by Paliwal and Atal were valid for the tree designs of this work. Principally the difference between this work and Paliwal and Atal's was the *tree* structure imposed

Split Low/High	Ave. Spectral Distortion	Outliers 2–4 dB	Outliers >4 dB
4/6	2.35	54%	6%
5/5	2.18	50%	3%
6/4	2.23	52%	3%

Table 5.7: Spectral Distortion and Percent Outliers for the Split Tree Structured Quantiser.

on the codebooks.

An approximate calculation of the burden for the split vector quantiser under the assumption that the vector dimension is halved in each codebook and that the number of entries in each codebook is the same gives;

$$\begin{aligned} N &= N_1 N_2 \\ L &= L_1 L_2 \\ \tilde{C} &= N log_2 L \\ \tilde{M} &= 2N(\sqrt{L} - 2) \end{aligned} \qquad (5.45)$$

A tree structure was assumed with independent quantisation for each stage. Note, the split vector quantiser used less memory than the cascade configuration.

To maximize performance of a split vector quantiser, the number of component sub-vectors should be minimized (because of inter-component dependencies) and the size of each codebook maximised (as much memory as the system can afford).

Table 5.7 shows the results of splitting the vector quantiser with differing numbers of components for each part and then determining the spectral distortion. Each part of the codebook was trained for 128 entries and used the same training set as used in the previous work.

In Table 5.7 the notation 4/6 for the split refers to the first four LSF components of the input vector were used for the first sub-vector (first codebook) and the following six components were used in the construction of the second codebook. The results obtained contrast with Paliwal and Atal's results for the full searched quantisers in that the tree searched codebooks had best performance with a 5/5 split. It was also noted that the split quantiser although it used 14 bits, it performed barely better than the 512 entry codebook (9 bits). This technique had dubious value when coupled with tree structured quantiser.

It was noted that during training of the tree structured codebook on the split vectors that the lower vector portion, that is the lower LSF's, showed a distinct principle direction 80% of the time, while the upper vector portion only 20% of the time. This implied that the upper sub-vector portions were more *white*[5] where tree splitting would be less effective. Comparing Paliwal and Atal's results with the result here demonstrated that the tree structuring was more sensitive to the vector quantiser's cell shapes than a fully searched code book. Another feature of the split vector quantiser should be considered, that is, the transmission of the indices over a noisy channel. Line spectral pairs have the nice property that a distortion of any one component is relatively localized, however an error in a codebook index no longer localizes the distortion. This implies that both indices need to be protected; the index for the first codebook more heavily than the second. The higher frequencies of the speech spectrum, as mentioned earlier, have lower power and the ear resolves them to a lesser degree, so the level of protection may be reduced on the second index.

5.8 Multistage Vector Quantisers for Noisy Channels

The design of the multi-stage vector quantiser for the noisy channel follows the same structure and procedures as discussed for the single stage tree structured vector quantiser except that the tree design proceeds for the vector residuals obtained from the previous stage. In the case of two stages, two indices are transmitted over the channel. A tree structured vector quantiser was constructed for the first stage of the multistage quantiser, but because the performance of the tree structured quantiser on *white noise* vectors is inferior than more correlated vectors, the second *error* vector quantiser was made a full search quantiser. To simplify the system, scaling of the second quantiser was not performed as suggested by Lee et al. [99] but may be incorporated for better performance at the cost of extra bits.

Figure 5.5 shows a block diagram of the two stage multi-stage vector quantiser used in the design.

The performance of the multistage quantiser is better than the split vector quantiser.

[5]As the vectors were white noise like with equal probability, the VQ cells appeared almost spherical.

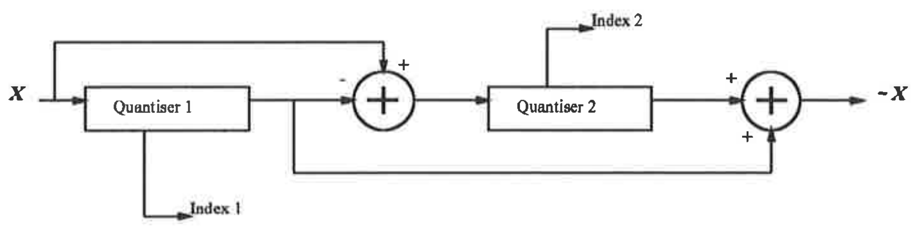


Figure 5.5: Multi-Stage Vector Quantiser Using Two Stages.

This can be seen from Table 5.8 and Figure 5.6, where the split vector quantiser performance is plotted as points for the 5/5 and 6/4 low/high vector split cases. It should be noted that a first stage quantiser of 512 entries followed by a 32 entry *error* quantiser (total 14 bits) performed slightly better than the first stage quantiser with 256 entries followed by another 256 entry *error* quantiser (total 16 bits). This implied that the first codebook should be made as large as practical followed by a smaller *error* codebook. Computationally this approach is attractive as most of the quantisation gain is made by the first (coarse) quantiser and the more computationally intensive full search second stage is performed over the smaller codebook. The transmission of the indices over the noisy channel using this quantiser also has advantages. A simple error detection scheme on the second quantiser index can assist the decision to use the finer quantisation or discard it. Assuming the first quantiser index is received correctly, the distortion can be no greater than that caused by the encoding process using the first (coarse) quantiser stage.

The actual quantising process performs a monotonicity check on the line spectral frequencies, however monotonicity cannot be used reliably as an error detection mechanism. An evaluation performed for the 512 entry coarse and 128 entry error quantiser vectors indicated that 90% of all combinations of vectors from the first and second codebooks gave valid encodings.

5.9 Conclusions

The novel idea of using principal component analysis for the splitting process in the design of tree structured quantisers, although not producing substantially better codebooks, did give some objective measure of the performance of the tree structured codebooks over full search codebooks. If most of the quantiser cells had a principal direction of the covariance matrix for the cell then the tree quantiser tended to perform

Book Configuration	Ave. Spectral Distortion	Outliers 2–4 dB	Outliers >4 dB
512-32	1.90	38%	1%
512-64	1.81	35%	0%
512-128	1.73	31%	0%
512-256	1.64	26%	0%
256-32	2.20	51%	3%
256-64	2.10	47%	2%
256-128	2.01	43%	2%
256-256	1.92	40%	1%

Table 5.8: Spectral distortion and percent outliers for the multistage tree structured quantiser. The two primary *coarse* quantisers have 512 and 256 entries; the *error* quantisers were allowed to vary in size.

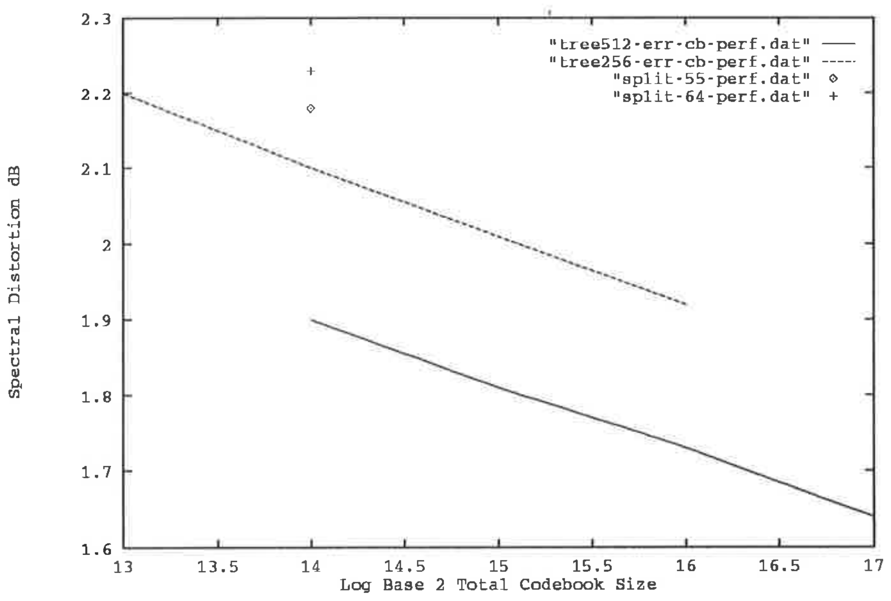


Figure 5.6: Average Spectral Distortion Versus the Total Codebook Size: The upper curve is for the 256 vector coarse quantiser and the lower curve for the 512 vector coarse quantiser, the points show the performance of the better split-vector quantisers.

well. This was best seen in constructing the *split* vector quantiser where the codebook for the upper sub-vector was *white* and only a small number of cells exhibited a strong principal component. The performance of the tree structured codebook was poor in this case. Whenever most cells in the vector quantiser showed a strong single principal component, performance of the overall codebook was good. The most likely explanation for this phenomenon is that the tree construction can only split a single cell into two regions. The direction of maximum variance is the most likely direction to split the cell such that encoding of the training set will have lower average distortion than the current level.

A new fully integrated method of design of source and channel coding of speech spectrum parameters was then presented. Factors that were considered included:

1. minimum bit rate, implying vector quantisation and line spectral pair representations;
2. small computational burden in finding the vector codebook index;
3. combined source and channel coding for vector codebooks;
4. unequal error protection;
5. small storage requirements;

all relative to the subjectively significant spectral distortion measure.

For tree structured quantisers the *multistage* vector quantiser gave a better performance than the *split* vector quantiser. Further improvement may be obtained by using a scaling for the second quantiser, but at the expense of more bits. The tree structured codebook also would appear to have a greater sensitivity to correlations between vector components (biassing against the split vector design), however a very large data set would be required to verify this.

If the multistage quantiser is chosen a cautious guess using extrapolation and adding in allowances (for the small training set) a coarse quantiser of 14 bits (16000 entries) and an error quantiser of 10 bits (1000) entries will in all likelihood meet the transparent coding specification. (Paliwal and Atal suggest 24 bits for their quantiser.) If simple error detection was used on the *error* quantiser index an additional 3 bits would be added for cyclic redundancy[6]. The cyclic redundancy would use the generator poly-

[6]See Chapter 4 of Lin and Costello [86]

nomial $x^3 + x + 1$. This code is capable of detecting any 2 random errors and any burst of contiguous error bits up to 3. If the channel bit error rate was $\epsilon = 0.01$ the probability of undetected error would be 2×10^{-4}. Of the 14 bits in the coarse quantiser index 9 or 10 would probably need to be protected. To protect these bits well it would probably be necessary to use a rate 1/2 code with soft decision decoding, thus an extra 9 or 10 redundant bits would be required. The number of bits to transmit the line spectral pairs representation reliably over a noisy channel now totals 36 or 37 bits. It may be possible to ignore the error introduced by the *error* quantiser and only check for monotonicity at the receiver, as the errors are only a fine adjustment. The level of spectral distortion caused by an incorrect error quantiser index still needs to be investigated using a much larger speech database and greater computational power than was available for the work in this chapter.

An advantage of using vector quantised indices and forward error correction on the most significant bits of the codebook index, is that some of the error correcting power of the code can be used for error detection. In the mobile communications environment deep fades and consequently large bursts of error are common place. The detection of a burst would allow other higher level error correction capabilities such as packet substitution as described in [51], [21]. The substitution of previously received *clean* packets for burst error corrupted packets yields acceptable results.

Chapter 6

Trellis Vector Excitation

6.1 Introduction

Various forms of code excited linear predictive (CELP) vocoders have been discussed at length in the literature. There has been a growing interest in how these vocoders perform on noisy channels, in particular the mobile communications channel. Handling the corrupted CELP parameters has been discussed in the literature [20]. The previous two chapters have been pre-occupied with the short term spectrum model parameters and reducing the effects of channel errors on these parameters. All of the CELP vocoders discussed in the literature use a vector codebook excitation. This implies that if the vector index is corrupted in transmission over the channel an incorrect excitation is fed into the linear predictive model. Surprisingly this has remarkably little perceptual effect until the error rates approach several percent. This work tries to reduce the effect even further by using vector trellis encoding of the excitation with the goal in mind to simultaneously increase the perceptual quality of the speech and to make it more robust to channel errors, thereby eliminating the need for error correction on the excitation stream.

It has been hypothesised [101] that trellis encoding of the excitation is more robust to channel errors as a corrupt bit generates only a short range corruptive effect as it travels through the shift register (state generator) used in reconstructing the excitation sequence. This concept will be explored in more detail below.

Fehn and Noll [102] have performed limited studies of trellis encoding of the linear prediction residual but not for a CELP type of vocoder configuration. Stewart, Gray and Linde [44] gave a very brief account of a basic analysis by synthesis method which they called *hybrid tree coders.* The work here advances the model considerably.

The chapter starts with an overview of the salient points of the CELP model which is then extended to the trellis excited case. An analysis of the trellis encoded case is given with special emphasis on regular pulse excitation.

Steele [109] noted in his discussion of regular pulse excited CELP structures that the use of a long term (pitch) predictor provides 2 dB improvement when the pulse spacing is one in four but does not provide any improvement in performance when the pulse space was one in two. The higher rate of pulse excitation modelled the true residual well enough. This observation justified the major design effort for rate 1/2 systems without pitch period excitation or self excitation. Only after implementation of the 1/2 bit per speech sample coder was the extension of the design to lower rates considered.

Two distinct types of trellis codebook were investigated. One codebook used two symbols per branch for a rate of 1/2 bit per residual symbol. The other codebook made use of the regular pulse concept where the single symbol on the trellis branch was followed by a zero; which yields a coding rate of 1/2 bit per excitation symbol. Subjective tests indicated that the regular pulse codebook performed as well as the two symbols per branch codebook. The regular pulse excitation was the method of choice as it halved the number of symbols that had to be optimized in training. Coding was also attempted at the lower rate of 1/4 bit per symbol, using the regular pulse excitation.

A discussion is given of the (M,L) algorithm in Section 6.4 which was used to encode the speech. Implementation details are also covered in this section.

Much effort went into the optimization of the trellis codebook to improve the perceived quality of the encoded speech. Optimization of the codebook had an audible effect on the reproduced speech making it more pleasant to listen to. A major problem in the optimization process was that segments of speech had to be encoded and compared to the weighted original thousands of times in order to optimize the trellis parameters. This was necessary as optimization used the analysis by synthesis process. The chapter describes the development of algorithms to overcome the computational load problem.

Performance issues are assessed to determine the best operational parameters for trellis excited linear predictive coders.

The chapter concludes with a discussion on post processing of the synthetic speech. Other authors [4] [103] have also considered post processing to improve the quality

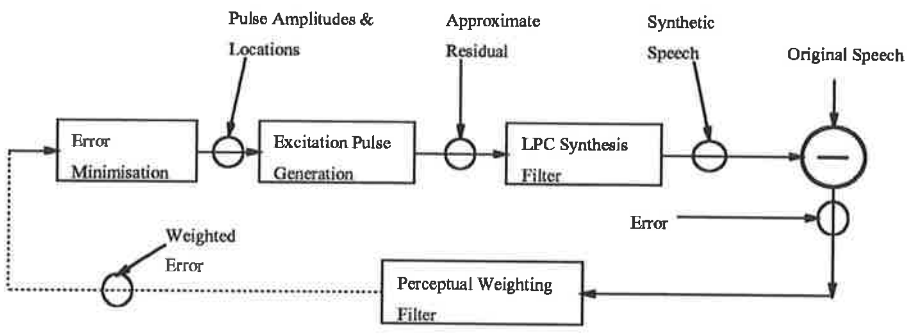


Figure 6.1: Simple Multi-Pulse Encoder.

of the received synthetic speech. The technique developed in this chapter however is unique as it uses harmonic processing in the time domain using synthesis filter impulse responses spaced at the pitch period. An audible subjective improvement in the speech quality occurs with the post processing.

6.2 The Code Excited Linear Prediction Vocoder Model

Many of the concepts of the modern CELP coder can find their roots from ideas presented by Atal and Remde [12]. In their paper they described a multipulse excitation mechanism for a linear predictive filter that used analysis by synthesis and auditory weighting to obtain natural sounding speech with a much simplified residual. Analysis by synthesis (refer to Figure 6.1) is effected by trial synthesis of each candidate vector followed by comparison to the original speech. The trial synthesis that is closest, usually in the mean square error sense, defines the residual encoding to be transmitted to the receiver. Quantisation error can be further reduced using perceptual masking by a *perceptual weighting* filter which is described in more detail below.

Linear predictive modelling of speech's short term spectral information is a fundamental component of the CELP model. Short term spectral modelling is normally executed using windowed, overlapped, speech segments of 20 ms to 30 ms duration. The overlap may be of the order of 5 ms. Most CELP implementations use a sampling rate of 8 kHz; this implies approximately 200 samples per analysis segment.

The short term spectral information analysis typically incorporates a 10 pole filter and only in a few cases are higher order models used. The recently developed low delay CELP uses a much larger order, however low delay CELP was not investigated in this work. Given that a filter $1/A(z)$ models the short term spectrum, the excitation sequence must be estimated in order to reconstruct the speech. In standard CELP the excitation sequence is determined as a gain and a codebook entry, where the codebook entry is a vector typically 1/4 the length of the analysis segment. The excitation sequence is determined using analysis by synthesis. Analysis by synthesis in CELP is essentially the same process as was described for the earlier multi-pulse coders. The closest match between a trial vector and the original speech is the selected vector, and subsequently the vector's codebook index and quantised gain are transmitted. Observation of the residual of LPC filtered speech reveals a quasi-periodicity due to the pitch period. CELP coders make use of this in various ways. One way is to utilize a long delay (*pitch*) filter of the form;

$$\frac{1}{P(z)} = \frac{1}{1 - \sum_{i=-q}^{r} b_i z^{-(D+i)}} \tag{6.1}$$

where D is the long term delay and typically three taps ($q = r = 1$) or one tap ($q = r = 0$) is used.

Another method that is used is called the adaptive codebook method. In this method, each time a sub-block of the speech segment has been analysed, the selected excitation sequence (sum of the contribution from the long term *adaptive* codebook and the short term *stochastic* excitation) is appended to the adaptive codebook and the oldest excitation sub-block of the same length is removed. The adaptive codebook thus holds a fixed size history of past excitation sequences. Excitation vectors are obtained from the adaptive codebook by performing a pattern match on n contiguous samples, where n is the length of the desired vector within the overlapped codebook. The position in the codebook that produces the best match is the index that is transmitted. Any vector, within an allowable range of the overlapped codebook, can be used as the matching vector which avoids any constraints on the index being directly related to the pitch period.

When the long term excitation information is included, the long term contribution to the excitation sequence is determined first, followed by the short term contribution from the stochastic vector codebook. Although the best performance would result from a combined search of both excitation sources, the calculations become prohibitively complex for real time application. Most CELP coders follow the sub-optimum procedure of determining the long term excitation component and its associated gain, followed by selection of the best vector from the stochastic codebook and determining its optimum gain.

Selection of the best excitation sequences traditionally was done using a mean square error technique, however better perceptual performance can be achieved by incorporating an auditory model into the mean square error criterion. An error weighting filter is used to weight the error to allow greater error tolerance around spectral peaks and less in spectral valleys. The weighting filter is given by;

$$W(z) = \frac{A(z)}{A(z/\gamma)} \tag{6.2}$$

$$= \frac{1 - \sum_{i=1}^{p} a_i z^{-i}}{1 - \sum_{i=1}^{p} a_i \gamma^i z^{-i}} \tag{6.3}$$

where p is the number of poles and γ typically takes values between 0.7 and 0.95. A value of approximately 0.9 is most common.

This form of weighting filter was discussed by Atal and Schroeder in [104] at some length. The parameter γ is chosen by the degree by which the formant regions should be de-emphasized in the error spectrum. Decreasing γ increases the bandwidth ω of the poles. The increase ω is given by,

$$\omega = \frac{-f_s}{\pi} \ln(\gamma) \tag{6.4}$$

where f_s is the sampling frequency. When $\gamma = 0$ the output noise from the weighting filter $W(z)$ has the same spectral envelope as the original speech and when $\gamma = 1$ no weighting occurs.

The CELP model may now be represented as shown in Figure 6.2 where the input speech is weighted by passing it through a perceptual weighting filter of the form;

$$Y(z) = \frac{A(z)}{A(z/\gamma)} \tag{6.5}$$

The trial excitation is passed through a perceptually weighted synthesis filter of the form;

$$H(z) = \frac{1}{A(z/\gamma)} \tag{6.6}$$

Referring to Figure 6.2, the transmitted parameters are: the index for the adaptive codebook, the gain for the adaptive codebook vector, the index for the stochastic codebook and its associated gain, and the a_i's (LSP representation).

A CELP coder compares the perceptually weighted synthetic signal with the perceptually weighted input speech. A mean square error minimization process is used to select

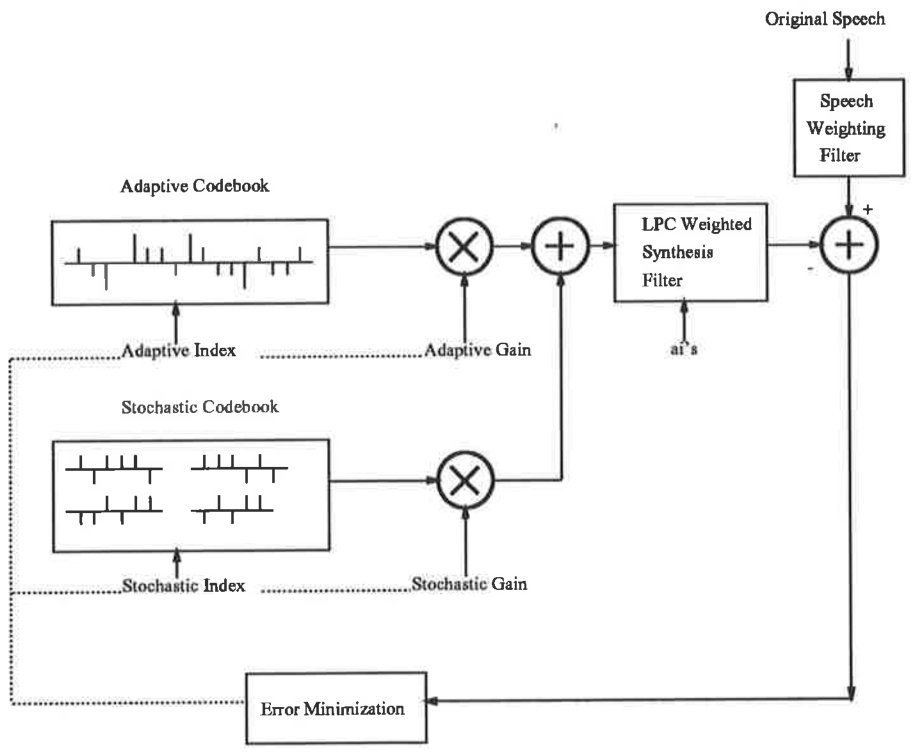


Figure 6.2: Block Diagram of a Generic Code Excited Linear Predictor Coder.

the optimum excitation vectors. Gain is determined inside the minimization loop for both the long term predictor and the short term predictor.

6.3 Analysis of the Trellis Encoding Process

Figure 6.3 illustrates the basic blocks of the trellis excitation process. The adaptive codebook of Figure 6.2 has been removed and the trellis generates symbols in place of the stochastic codebook. The trellis codebook generates symbols that feed into the linear predictor from which a codetree is generated. The method chosen to track through the tree is to implement (in effect) a bank of predictors that attempt to choose the best set of paths through the tree.

A processing step for a vector of n samples, consists of:

1. Input n samples to be coded,
2. For each filter 'i' of the M prediction filters;

(a) Check the state of the ith best path,

(b) Load the prediction filter memory from the path storage,

(c) Use the state from step (a) and the trellis to generate n symbols from the upper transition and feed into the i'th predictor to obtain n synthetic speech symbols,

(d) Find the mean square error between the synthetic speech symbols and the input perceptually weighted speech, and add to the accumulated error for the path so far;

(e) Store the new filter memory, the accumulated error, the new trellis state and a 0;

(f) Load the prediction filter memory from the path storage,

(g) Use the trellis to generate n symbols from the lower transition and feed into the i'th predictor to obtain n synthetic speech symbols,

(h) Find the mean square error between the synthetic speech symbols and the input perceptually weighted speech, and add to the accumulated error for the path so far;

(i) Store the new filter memory, the accumulated error, the new trellis state and a 1;

(j) Goto step a) till the best M paths have been processed.

3. Sort through the paths in the path storage and find the set of M paths which have the lowest accumulated error;

4. Repeat the above step steps (1 to 3) till the entire block has been processed;

5. Trace back through the tree on the best overall path and obtain the series of 1's and 0's which describes the best path,

6. Retain only the last path memory entry of the best path as the root of the code tree for the next block.

The gain analysis is performed open loop and the gain is used for the entire block of 200 samples. The series of 1's and 0's, which are transmitted to the receiver, trace a unique path through the code tree.

Analysis of the predictive trellis encoding procedure and in particular regular pulse trellis encoding in not covered in the literature. The approach and notation used in this work follows Steele's [109] arguments.

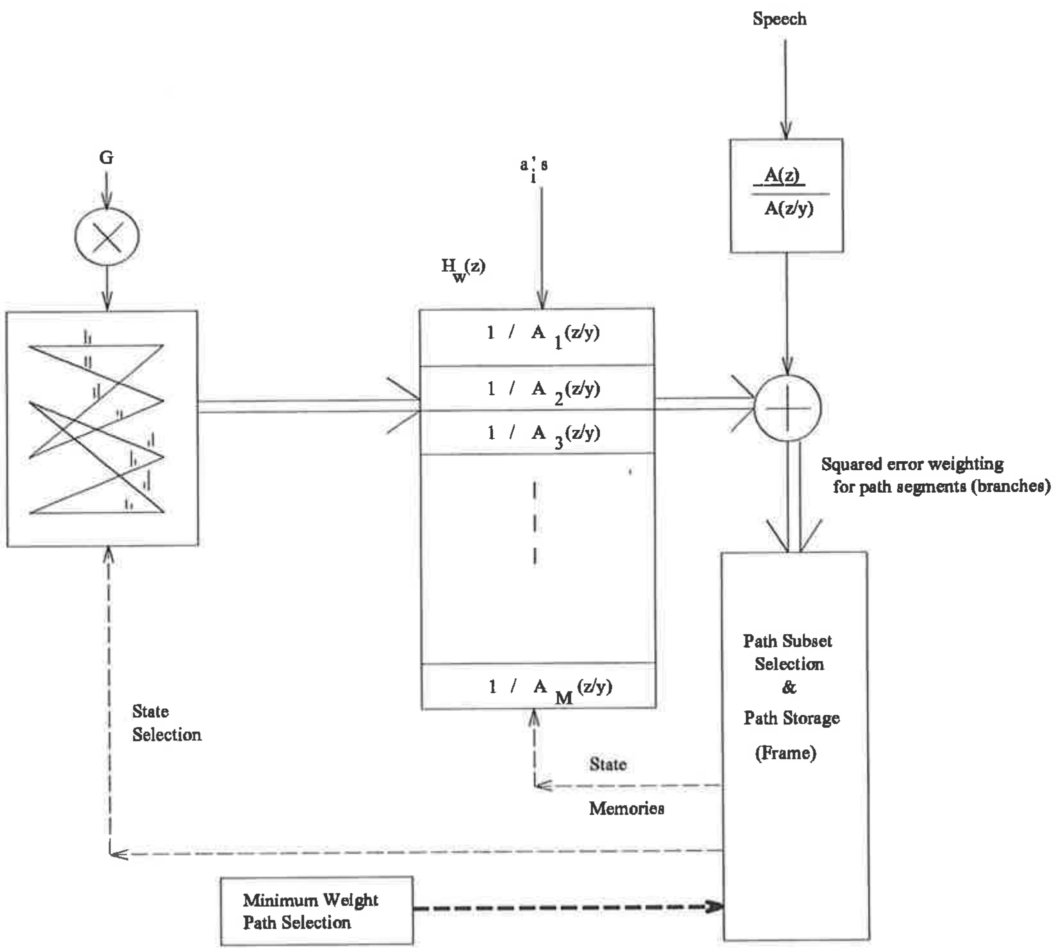


Figure 6.3: Block Diagram of the Trellis Code Excited Linear Predictor Coder.

Let an excitation frame of length N contain M pulses with amplitudes β_i at positions m_i. The excitation sequence is expressed as,

$$v(n) = \textstyle\sum_{k=0}^{M-1} \beta_k \delta(n - m_k) \quad n = 0, 1, \ldots, N-1 \tag{6.7}$$

where $\delta(.)$ is the Kronecker delta function. The residual prediction error is given by,

$$r(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \tag{6.8}$$

where for notational convenience the prediction coefficients are redefined as $a_k = -a_k$ in Equation 6.8. The perceptually weighted speech (refer back to Chapter 1) can be expressed in terms of the residual and past weighted speech samples;

$$s_w(n) = r(n) + \sum_{k=1}^{p} a_k \gamma^k s_w(n-k) \tag{6.9}$$

If $h_w(n)$ denotes the impulse response of the weighted speech synthesis filter, the weighted speech can be given by,

$$\begin{aligned} s_w(n) &= \sum_{i=-\infty}^{n} r(i) h_w(n-i) \\ &= \sum_{i=0}^{n} r(i) h_w(n-i) + s_0(n) \end{aligned} \tag{6.10}$$

The first term in Equation 6.10 is called the memoryless convolution of the residual and the filter $h_w(n)$.

Equation 6.10 can be expanded to yield a series of terms for Equation 6.9. A sketch of the procedure yields;

$$\begin{aligned} s_w(n) &= r(0)h(n) + r(1)h(n-1) + r(2)h(n-2) + \ldots + r(n)h(0) \\ a_1\gamma^1 s_w(n-1) &= a_1\gamma^1 r(0)h(n-1) + a_1\gamma^1 r(1)h(n-2) + a_1\gamma^1 r(2)h(n-3) + \ldots \\ &\quad + a_1\gamma^1 r(n-1)h(0) \\ a_2\gamma^2 s_w(n-2) &= a_2\gamma^2 r(0)h(n-2) + a_2\gamma^2 r(1)h(n-3) + a_2\gamma^2 r(2)h(n-4) + \ldots \\ &\quad + a_2\gamma^2 r(n-2)h(0) \\ &\vdots \\ a_p\gamma^p s_w(n-p) &= a_p\gamma^p r(0)h(n-p) + a_p\gamma^p r(1)h(n-p-1) + a_p\gamma^p r(2)h(n-p-2) + \ldots \\ &\quad + a_p\gamma^p r(n-p)h(0) \end{aligned} \tag{6.11}$$

Substituting back into Equation 6.9, $h_w(n)$ can be deduced,

$$h_w(n) = \delta(n) + \textstyle\sum_{k=1}^{p} a_k \gamma^k h_w(n-k) \quad \text{n=0,1,...,N-1} \qquad h(n) = 0 \text{ for } n < 0 \tag{6.12}$$

and the zero input response for the filter as,

$$s_0(n) = \sum_{k=1}^{p} a_k \gamma^k s_0(n-k) \tag{6.13}$$

which depends on the contents of the filter memory. This implies that the memory of the predictor from the last block/sub-block must be preserved for coding the next block/sub-block.

The weighted synthesised speech may be written in terms of the memoryless convolution,

$$\hat{s}_w(n) = v(n) * h_w(n) + \hat{s}_0(n) \tag{6.14}$$

Note the similarity between Equation 6.14 which uses the excitation and Equation 6.10 which uses the residual.

Now substituting the proposed excitation sequence $v(n)$ from Equation 6.7 into the previous equation,

$$\begin{aligned}\hat{s}_w(n) &= \sum_{i=0}^{n}(\sum_{k=0}^{M-1} \beta_k \delta(n-m_k) h_w(n-i)) + \hat{s}_0(n) \\ &= \sum_{k=0}^{M-1} \beta_k h_w(n-m_k) + \hat{s}_0(n)\end{aligned} \tag{6.15}$$

The error between the weighted input speech and the synthetic speech then becomes,

$$e_w(n) = s_w(n) - \hat{s}_0(n) - \sum_{k-0}^{M-1} \beta_k h_w(n-m_k) \tag{6.16}$$

The mean square weighted error for the block is given by,

$$\begin{aligned}E_w &= \sum_{n=0}^{N-1} e_w^2(n) \\ &= \sum_{n=0}^{N-1} [s_w(n) - \hat{s}_0(n) - \sum_{k=0}^{M-1} \beta_k h(n-m_k)]^2\end{aligned} \tag{6.17}$$

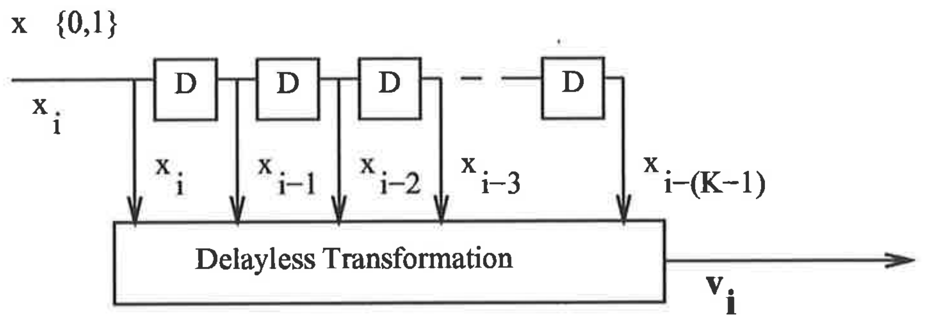


Figure 6.4: Convolutional Source Code Branch Symbol-Vector Generation.

If regular pulse excitation is used with one pulse β_k followed by $D-1$ zeroes, the m_k pulse positions may be written as,

$$m_i^{(j)} = j + iD \quad \begin{array}{l} j = 0, 1, \ldots, D-1 \\ i = 0, 1, \ldots, M-1 \\ \text{such that } MD = L \end{array} \tag{6.18}$$

The *phase* of the pulse within the D samples is given by j, but this feature was not used in this work.

If the pulse amplitudes can be individually chosen, use is made of the set of M simultaneous equations arising from,

$$\tfrac{\partial E_w}{\partial \beta_k} = 0 \quad k = 0, 1, \ldots, M-1 \tag{6.19}$$

For trellis coding the β_k are selected from a trellis structure such that the estimates $\acute{\beta}_k$ produce minimum distortion $d_N(.)$ on reconstruction of the speech. The distortion can be written,

$$d_N(s_w, \hat{s}_w) = \frac{1}{N} \sum_{n=0}^{N-1} d(s_w(n), \hat{s}_w(n)) \tag{6.20}$$

where $d_N(\cdot)$ could simply be the mean square error given by 6.17.

Referring to Figure 6.4, a set of K branch choices $x_i \in \{0, 1\}$ is used to generate the excitation sequence $v(n)$ which is the input to the predictor. Each branch v_i may be a sub-vector of excitation values,

$$v_i = (v_{Di+0}, v_{Di+1}, \ldots, v_{Di+D-1}) \tag{6.21}$$

For the regular pulse excitation considered here, using $j = 0$ in 6.18, v_i becomes,

$$v_i = (\acute{\beta}_{Di}, 0, \ldots, 0) \tag{6.22}$$

Trellis encoding assumes independent sub-sequences/branches so that a distortion can be calculated for each branch, which can then be accumulated for the entire path. The structure of the trellis and examples are given in the following section; see Figure 6.5.

The encoding procedure attempts to find a path through the trellis, defined by the branch choices x, by trying to match the β_k indirectly with a set $\acute{\beta}_k$. The indirection comes about because the weighted synthetic speech and the weighted input speech are used for the path selection process. For regular excitation from Equation 6.17,

$$d_N(s_w, \hat{s}_w) = \min_j \sum_{n=0}^{N-1} (s_w(n) - \sum_{k=0}^{M-1} \acute{\beta}_k^{(j)} h(n - Dk) - \hat{s}_0(n))^2 \tag{6.23}$$

where the minimization occurs over all possible paths j that have the $\beta^{(j)}$ as branch labels.

The Viterbi algorithm is not optimal due to the prediction having a significant effect beyond the current branch. If a sub-sequence is considered starting at time $\acute{n} = Dk$ the synthesised weighted speech for the branch is given, from (6.15), by

$$\hat{s}_w(\acute{n}) = \acute{\beta}_k^{(j)} h_w(n - Dk) + \hat{s}_0(\acute{n}) \tag{6.24}$$

where $\hat{s}_0(\acute{n})$ is the predictor zero input response at time Dk. The weighted speech for each branch depends on the path history.

At each decision stage of the Viterbi algorithm, or other breadth first trellis search algorithm, an entire path history is eliminated and it is possible that had the path been retained, it may have attained a lower overall distortion by the end of the block. If the paths are not discarded, a tree structure results where an exhaustive search is the only optimal approach.

Two strategies for encoding could be used: assume the simplified trellis structure and use the Viterbi algorithm [106], or an algorithm like the (M,L) algorithm [49] or the Simmons algorithm [107]; or use a metric first sequential encoding algorithm such as the stack algorithm or a modified Fano algorithm. Sequential encoding algorithms are described in Viterbi and Omura [105]. The approach used in the thesis used the (M,L) algorithm.

6.4 Trellis Encoding of the Excitation Residual

Trellis coding of the residual faces many practical problems. A major problem associated with the technique is determining the gain for the excitation vector as scaling of the input speech had to be performed prior to encoding. The Viterbi algorithm generally performs better than the (M,L) algorithm, however for large trellises it becomes impractical. There is the additional problem in the predictive coder case, that although the trellis format describes states of the encoder associated with the innovation sequence, additional states of the system due to the predictor memory must also be considered. This can easily be seen by observing that the output of the predictor depends on memory greater than the state memory. The impulse length of the synthesis filter, which is discussed at some length in the following chapter, is of the order of 100 samples for voiced speech which is much greater than the memory used in describing the states of the trellis codebook. So strictly speaking, the code formed by the predictor system described is a tree code.

For the source encoding problem, although an optimal encoding is desirable, a very good encoding should be quite acceptable. The (M,L) algorithm was chosen for encoding in this work as the encoding effort is constant across different sized trellises, and thus a realistic option for a real-time implementation.

The issue of whether it was better, performance-wise, to use a large trellis and the (M,L) algorithm or a smaller trellis and the Viterbi algorithm was also investigated and results are given below.

A node in the trellis is also known as a state and arcs connecting nodes are commonly called branches. A path from a node to another node is said to start at the parent node and end on the child node. Let N denote the number of states in the trellis. An example trellis of four states is illustrated in Figure 6.5. States are indicated by numbers in the illustration and branch symbols by alphabetic letters. Branch symbols may be scalar or vector quantities. Encoding of the source data is the process of obtaining the best match, according to some metric, between the predicted speech using the symbols on the branches and the source data. Generally the measure of choice is squared error distortion, but others were also investigated. The encoding procedure used metrics from two sources: the metric due to mean square error between the predicted samples and the input perceptually weighted speech samples, and the metric which consisted of accumulated branch metrics which was associated with the parent node. Each child node had a new accumulated metric associated with it, which was the sum of the two

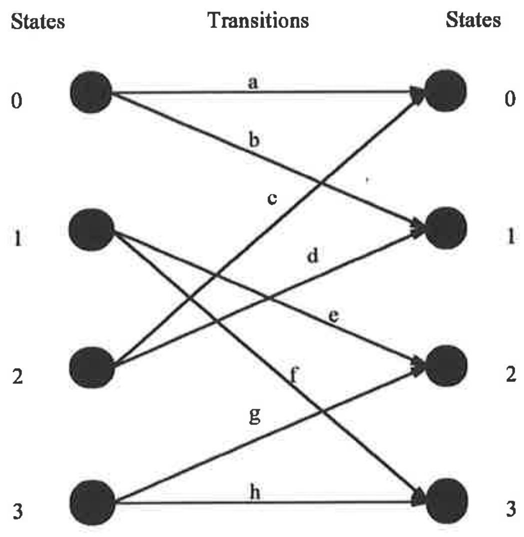


Figure 6.5: Example 4 State Trellis.

metrics outlined above.

The (M,L) algorithm:

1. Start at the root node and extend paths out on all branches from that node, determine the branch metrics and accumulate with the metric associated with the parent root node, and associate the new metrics with the child nodes;
2. Keep on extending active nodes until $\geq M$ child nodes exist, updating the accumulated metric for each new child;
3. (There are $\geq M$ nodes active at this stage) Sort the accumulated metrics of the new child nodes in ascending order (the smallest metric is known as the best metric) ;
4. Check to see if there is more than one path ending on the same child node and if so, choose the path with the lower accumulated metric and discard (label path as inactive) the path with the higher accumulated metric;
5. If the path depth (length in branches) is greater than L trace back the path with the lowest accumulated metric and release the symbol/s associated with the last (oldest) branch and mark the child node for that branch as the new root node, remove any surviving paths that do not terminate on the new root node;
6. If the size of the list of candidate paths is greater than M keep the M best paths and discard the rest;

State	Branch 1 Symbols	Branch 1 Next State	Branch 2 Symbols	Branch 2 Next State
0	1.87 0.33	0	0.03 -2.13	2
1	0.05 -0.23	0	1.03 0.55	2
2	-1.03 -0.55	1	-0.05 -0.23	3
3	-0.03 2.13	1	-1.87 -0.33	3

Table 6.1: Look up table representation for a trellis

7. Extend branches forward on the active paths;

8. If not at end of source data, Goto step 3.

The process of extension in the (M,L) algorithm is the process of associating the source data with the predicted speech waveform generated by the symbol/s labelling the branch. A metric was determined for each branch and became the branch metric. An accumulated metric was obtained by adding the branch metric to the previously accumulated metric associated with the parent node that the branch emanated from, and the new accumulated metric was associated with the new node that the branch terminated on.

On traceback to the terminal node the released symbol was a bit code for the selected branch. As transitions between states in the trellis were restricted to be via the branches, a continuous stream of bit symbols conveying branch selection information at the transmitting end allows path reconstruction at the receiver. Furthermore, the branches on the path have associated with them symbol/s for encoding the source data. The reconstructed predictor innovation stream consisted of the concatenation of the symbols labelling each branch traversed in the code tree.

An efficient method for representing a trellis is via a look-up table giving branch symbols and the next state. A sample look-up table for a trellis is shown in Table 6.1. This table is for a 1/2 bit per sample code, where the branch symbols are fed into the prediction filter to give two predicted speech samples. Each branch is represented by one bit, and each branch produces two predicted speech samples. Each branch is associated with a state transition where the new state is given by the *Next State* columns in the table.

The encoding of the excitation for the adaptive linear predictive filter was of the class of

Gain/Shape vector quantisers which required that the gain for a segment of excitation be determined followed by normalisation of the input speech. The normalized speech was then predictive trellis vector quantised. This strategy gave a very large effective *codebook* as the basic *shape* of a vector could be used with a variety of gain values. If the DoD US Standard 1016 (CELP coder) [3] is referenced it may be noted that the *Gain/Shape* vector quantisation philosophy is used for the stochastic codebook and also the adaptive codebook.

As the gain could not be determined accurately till after vector/path selection was made, an estimate had to be determined prior to encoding. The standard CELP allows the synthesised speech to be matched to the weighted original speech and consequently an optimal gain for that segment of synthesised speech can be determined. The (M,L) algorithm like the Viterbi algorithm [106] matches the synthesised speech to the weighted original on a symbol by symbol basis. For this process to work properly, the weighted input speech must be scaled prior to the matching minimization process.

A gain estimate may be determined using two different methods. Makhoul [52] suggested using the filter gain as determined from the synthesis filter analysis phase. The gain is found from the minimum value of the prediction error E_p given by;

$$E_p = R_0 + \sum_{k=1}^{p} a_k R_k \tag{6.25}$$

where a_k are the linear prediction coefficients. R_n are the correlation coefficients for the analysis block defined by;

$$R_i = \sum_{n=0}^{N-|i|} s_n s_{n+|i|} \tag{6.26}$$

where the input speech is given by s_n. Gain for the synthesis filter is found from the relation;

$$G^2 = E_p \tag{6.27}$$

Vocoders such as the U.S. standard LPC-10 [108] use the r.m.s. value of the input speech as the gain term.

A relatively large codebook was used in the present implementation to try and lessen some of the difficulties associated with an open loop gain determination. By having a large number of entries in the codebook not only was the shape catered for but also a small range of gains. The large or gross variance of gain was substantially covered by the open loop gain determination. A scaled version of the gain as described by Equation 6.27 was found to have the best subjective performance; further details of gain determination are covered in the following section.

6.4.1 Implementation Details of the Trellis Vector Excited LP Coder

Figure 6.3 shows the basic block structure of the trellis vector excited linear prediction coder. A block (200 samples, 25 ms) of speech was analysed to determine the 10 linear predictor coefficients (a_i's) and then divided into 4 sub-blocks[1]. Prior to encoding the perceptually weighted speech, the speech samples were scaled by a gain factor to *normalize* the input amplitude by determining the r.m.s. value for each subblock. Equivalently the symbols on the trellis branches may be scaled as shown in Figure 6.3. The r.m.s. value was used to scale the speech so that the new nominal r.m.s. input range was 4 instead of unity[2]. To further improve the performance, the codebook was re-optimized with normalized r.m.s. input range set to 4.

Informal listening tests compared the r.m.s. power as a gain factor to the gain factor obtained by Equation 6.27. It was found that the latter gain gave better subjective performance, hence it was chosen. This method eliminated some computation as the filter gain was a by-product of the filter analysis phase, and the one gain was used for the whole block. Any variation in gain over the block was handled by the trellis codebook. The codebook was then re-optimized for the new gain criterion.

The (M,L) algorithm, with different number of paths M, was used for investigating predictive trellis encoding of blocks of 200 speech samples. The length of the traceback L was chosen to be basically ignored as a complete block was processed prior to making decisions. This simplification reduced the computation required per encoded symbol as the traceback mechanism was invoked only once for the entire block. To ensure that the branch decisions for the last symbols in the block were at a reasonable level of confidence, further samples beyond the block were processed. The traceback was

[1]It was found after much testing that it was not necessary to break the input block into sub-blocks

[2]Informal tests indicated that the trellis coding performed far better when the range of the scaled input speech was greater than 1.

then effected and branch decision bits collected. The decision bits associated with the excess speech samples beyond the block boundary were discarded as they were processed again in the next block which had a new set of linear predictor coefficients. Investigations indicated that there was little advantage to process more than a few extra speech samples past the end of the block. The block to block matching process appeared to be quite satisfactory with essentially imperceptible audible evidence of block joining.

The (M,L) algorithm worked with a trellis which had N states and K branches from each state. In the example trellis, Figure 6.5 $N = 4$ and $K = 2$.

The synthesis process for the encoding mechanism was involved and justified using as few paths M in the (M,L) algorithm as possible, although generally a larger number of paths subjectively improved the synthetic speech a little. Each path in the encoder had associated with it:

1. the current trellis state for the path,
2. the accumulated metric for the path to that point,
3. the synthesis filter (updated each block) which consisted of:
    (a) the linear predictor coefficients,
    (b) and the filter memory.

Each synthesis filter maintained a history of chosen branch symbols in its memory and because the symbols were retained it was not necessary to remove the *memory* effect from each new section to be encoded. The memory was maintained for all states on all paths on the tree code, which required careful record management. As it was not known before hand which paths would be chosen by the (M,L) algorithm the filter memory contents had to be kept for *all* candidate paths even those that would be discarded at the next step.

To keep track of all of the information required at a node on the path, a record consisting of:

1. the current accumulated metric,
2. the current trellis state that the node is on,

3. the synthesis filter memory to this point on the path and,
4. forward and backward pointers for reconstruction of the best path,

had to be maintained for all states on the paths of the code tree.

At the end of coding a nominal block (the block plus extra speech samples) the best (lowest) metric was chosen as the best model of the speech residual. A trace-back was then effected along the best path to the beginning of the block. Each branch choice, upper or lower, was recorded for transmission over the channel. The stream of 1's and 0's received at the receiver end allowed reconstruction of the path through the trellis and hence the associated branch values.

The actual implementation used a doubly linked list tree structure where each node maintained the information enumerated above. The tree only grew from the nodes that were determined by the (M,L) algorithm to be the best *survivors*. An array of pointers was maintained which pointed to the end *leaf* nodes, that is the *survivor* candidate paths. The array also allowed efficient sorting of the accumulated metrics for each path to determine which paths to retain. It was found that this structure was quite efficient when compared to several other implementations such as using tables of pointers. Figure 6.6 illustrates the notional structure of the implementation.

When coding of a new block began, the contents of the synthesis filter had to be consistent with the contents of the best path through the code tree from the previous block. To ensure this, a simple mechanism was chosen where the optimum path from the previous block was given an accumulated metric value of 0 and this node then became the root node for the encoding of the next block. Associated with the root node was the memory contents of the synthesis filter so that the smooth transition into the next block of speech to be encoded was ensured. It should be noted that the node chosen as the terminating node for the previous speech block coincided with the block boundary and not the last node on the best path as it included the extra speech samples.

It was found that encoding several symbols past the end of a block made little to no improvement in the overall quality of the speech. The reason for this was probably two-fold. Firstly the speech samples past the end of the block were encoded by one synthesis filter and then by the synthesis filter for the following block, so the path choice near the boundary would have been at best a compromise. Secondly at low bit rates the quantisation process was noisy, resulting in most of the best paths having a

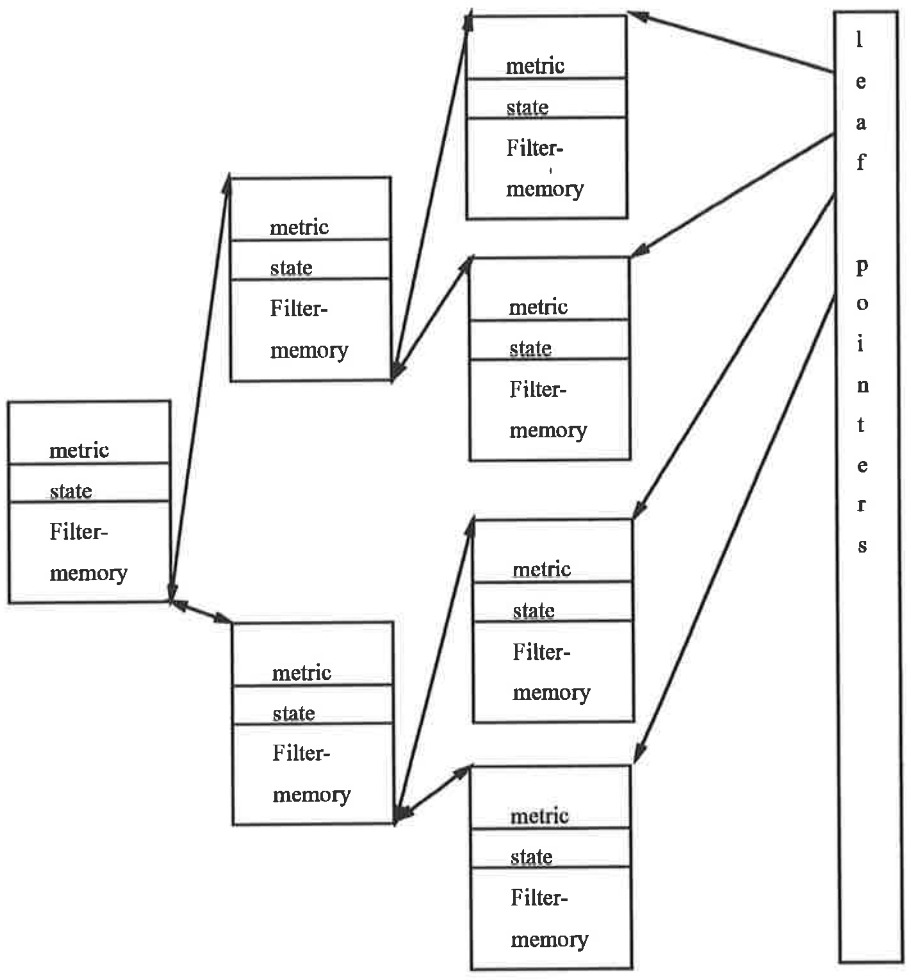


Figure 6.6: Block Diagram of the Doubly Linked Tree Structure. The blocks indicate records and the arrows pointers. Forward pointers were used for generating the tree and reverse arrows were used for trace-back to the root node.

very similar metric. Hence there was no great advantage in selecting, within the set of probable good paths, one path over the other. Noting these observations it was decided to reduce the overlap processing to reduce the computational effort; informal listening test confirmed minimal loss of performance.

As pointed out above, the encoded speech excitation was a stream of 1's and 0's that the receiver received over the channel. It may be considered that the receiver tracks states on the trellis by moving a window of size $\log_2 N$ over the received bit stream, where N is the number of states in the trellis. The bit pattern in the window defines the current state at the receiver and transitions from one state to another define which branch was taken.

Should a bit be received in error, the erroneous bit will propagate through the window as it moves along the bit stream and the maximum number of incorrect branch choices it can cause is $log_2 N + 1$. This effect causes a trade-off between having a large number of states to allow an accurate encoding and minimizing the effect of an error. There is another effect that must be considered when studying the error propagation, which is the memory effect of the speech synthesis filter due to its recursive nature.

An analysis of channel errors and the resulting waveform is covered in some detail in Chapter 6.

6.5 Code Book Optimization

The optimization of the trellis codebook is a very difficult problem. Stewart's [44] approach was considered too restrictive as it does not allow extension to the case of the regular pulse excitation [4] version of the codebook. There is also no guarantee that Stewart's approach will be near the global minimum. Two distinct types of trellis codebook were investigated. One codebook used two symbols per branch for a rate of 1/2 bit per residual symbol while the other codebook made use of the regular pulse concept.

A simplification was used in the design of the trellis codebooks by noting that speech waveforms tended to be quite symmetrical in the sense of the distribution of sample magnitudes about zero axis. To incorporate the symmetry, the trellis was constructed such that all branches had a complementary branch value in complementary positions on the trellis. This meant that the branch associated with "0" for the zero state had a complement in the branch associated with "1" for the "N-1"th state. The complement branch values were found by simply negating the branch values from the upper half of the trellis. This simplification had the advantage of cutting the number of free variables to be optimised by half.

Various techniques have been studied for the optimization of vector quantisation codebooks and trellis quantisation codebooks. Freeman [45] viewed the codebook optimization as a function space optimization problem. A Viterbi algorithm encoding procedure was assumed which resulted in the codebook function space to be essentially a convex surface. A convex surface can easily be optimized using gradient descent methods, although he argued that only those algorithms using first derivative information were viable, which converge relatively rapidly. It was pointed out by Freeman that the Viterbi algorithm assumption was critical to his argument and that the (M,L) algorithm would not produce the same convex surface.

The (M,L) algorithm was used for this work because it was considered that it could handle a large trellis in real-time. The parameter M sets the computational load for the encoding process independent of the number of states whereas the Viterbi algorithm's computational load grows exponentially[3] with the number of states.

To confirm Freeman's conjecture that the function profiles (sections or cuts) of the function surface would be far from convex, a set of experiments were performed. A segment of speech consisting of two phonetically balanced sentences, one spoken by a male speaker and the other by a female speaker, were used for determining the profiles. The segment, which consisted of 60,000 speech samples, was encoded for each trial trellis configuration and the overall average squared error determined. The function profiles are shown in Figure 6.7 for the direct codebook and Figure 6.8 for the regular pulse excitation codebook. The plotted variable is a particular random symbol value on a selected branch, while all other branch symbols in the trellis were held fixed. The vertical axis displays the average of the mean square error attained by the speech synthesis process.

Symbols in the four basic branch positions for the rate 1/2 coding were chosen for a section: upper branch first branch symbol, upper branch second branch symbol, lower branch first symbol and lower branch second symbol. There did not appear to be any relationship with the variable parameter's position in terms of upper or lower branch, or being the first or second symbol on a branch and the function profile. Clearly the

[3]Computational load is order $O(2^m)$, where m is the number of bits required to represent all states.

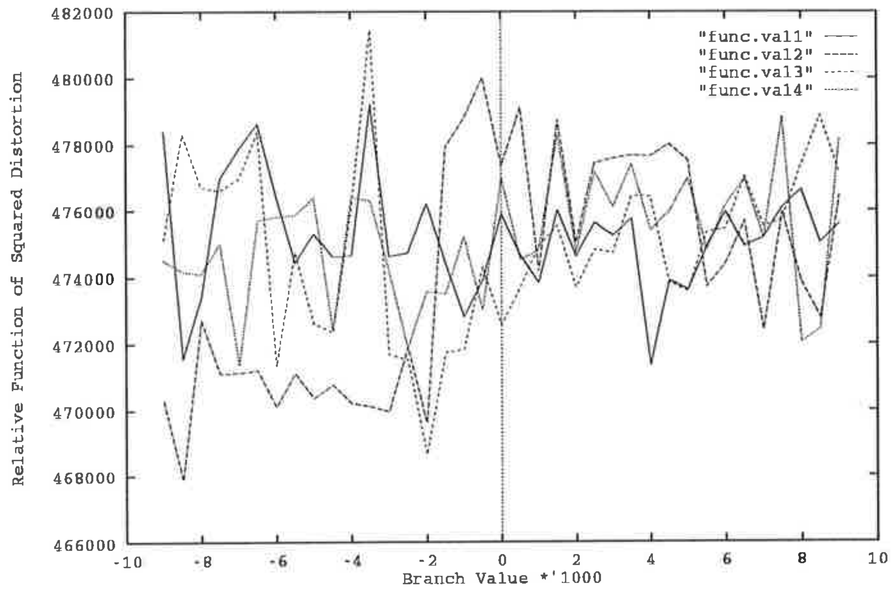


Figure 6.7: Four Profiles of the Trellis Evaluation Function; the function value was the average mean square error on encoding the source speech using the M,L algorithm for four randomly selected branches.

surface of this function is non-convex and displays *valleys* and *creases.*

Figure 6.8 also shows the *rough* nature of the surface of the function when regular pulse excitation was used. As the (M,L) algorithm performs a random walk over the trellis symbols many *cliffs* are possible where the chosen path set may change dramatically as the varying parameter is altered past a threshold. This property implied that the function was non-differentiable, therefore a gradient technique can not be used for optimization. Freeman used a conjugate gradient technique for the design of his trellis source coders, and assumed the use of the Viterbi algorithm for encoding which resulted in functions which were at least crudely convex. A predictor in this design invalidated Freeman's choice and conclusions about the Viterbi algorithm as the trellis structure was no longer valid and approximate convexity could no longer be assured.

6.5.1 Optimization for Rough Non-convex Functions

Two well known candidates for optimization of functions that are non-convex are simulated annealing [90] and genetic algorithms [91].

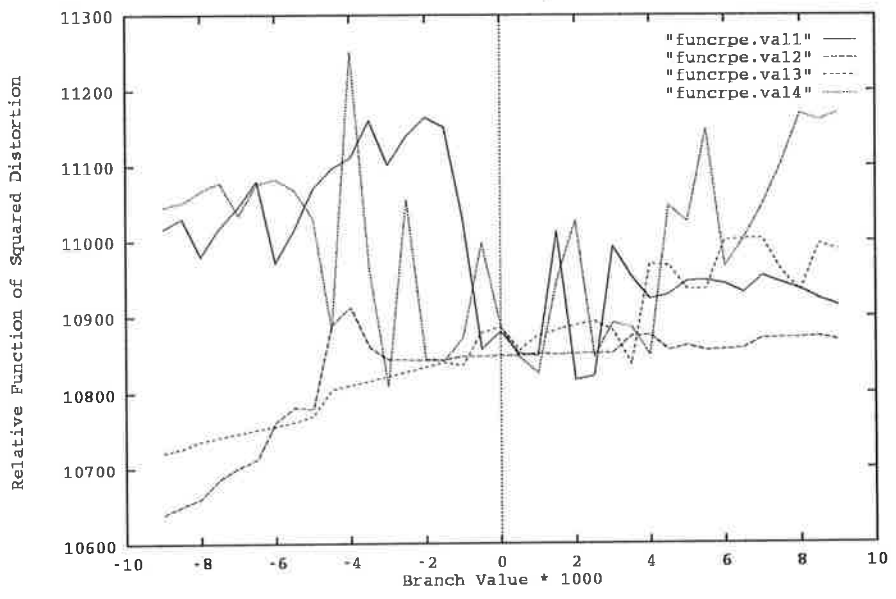


Figure 6.8: Four Profiles of the Trellis Evaluation Using Regular Pulse Excitation; evaluated for four randomly selected branches.

Simulated annealing is an analogue of physical annealing processes where systems which are heated and allowed to cool slowly produce a crystal structure in a low ground state energy. Statistical mechanics describes the average and fluctuations of energy of an ensemble of identical systems about a thermal equilibrium point for a particular temperature. The Gibbs function of energy is given by;

$$P_{en} = e^{-E(\{s_i\})/k_B T} \tag{6.28}$$

where P_{en} is the probability of the system with energy and configuration $E(\{s_i\})$, T is the current temperature and k_B is Boltzman's constant.

Five basic components are required to use a simulated annealing approach to solving functions (obtaining near global minima):

1. **Configuration space.** The configuration space must represent the space in which the solution is to be found. For the trellis optimization problem the multidimensional space is simply $\Re^N$ where N is 1/2 the number of branch symbols. (Recall the trellis was assumed to be symmetrical.)
2. **Reconfiguration strategy.** A set of rules must exist to move into another allowable configuration. For the trellis optimization a random perturbation was

applied to a randomly selected branch symbol. The perturbation was allowed to be positive or negative. As a large random perturbation was unlikely to be useful towards the end of the annealing process the perturbation was taken from a population of reducing variance and zero mean.

3. **Cost function.** The cost function is the function that the simulated annealing algorithm uses as the *Energy* term and as such must be well defined. The trellis optimization uses the average mean square error of the weighted synthesised speech compared to the original speech over a number of blocks of sample speech. The synthesis process uses previously determined branch parameters and the current trellis' selected branch symbols (current configuration).

4. **Cooling schedule.** The temperature at the start of the annealing should be high enough so that the parameters in configuration space can easily move between configurations. Cooling of the system should occur at a slow enough rate to ensure that the system is not frozen into a local minimum too early. Another aspect of cooling is to determine when the system has completely *frozen.* For the trellis problem many trial runs were required to determine a suitable starting temperature and cooling profile. The chosen profile was $T_{new} = \alpha T_{old}$ with $\alpha = 0.99$.

5. ***Metropolis* algorithm.** This part of the simulated annealing algorithm determines the probabilistic acceptance of the current trial configuration. The Metropolis algorithm is explained in more detail below.

The Metropolis algorithm may be described as follows: firstly evaluate the cost function to determine the new *Energy*, find the difference between the trial energy and the old energy (ΔE) and accept the new configuration according to the rules;

if $\Delta E < 0$ accept new configuration,

else accept with probability $P_M = e^{-\Delta E/T}$.

The standard Unix operating system *random*() function was used in the **else** case to select a random number (between 0 and 1) which if less than the probability P_M confirmed acceptance, else the new configuration was rejected.

Within the genetic algorithm optimization, a genetic string consisted of N substrings each representing a branch symbol. A complete genetic string consisting of the collection of substrings represented a complete trellis. A substring's position in the overall genetic string governed its position within the trellis. The genetic algorithm was described in some detail in Chapter 4. Each sub-string was a branch value bit encoded to some level of quantisation; the greater the resolution of the branch symbol, the greater the length of the substring.

Simulated annealing and a genetic algorithm were both applied to the optimization of the trellis branch values. In both cases the cost function that was evaluated was the average accumulated squared error of the synthesised waveform. The trellis used in the simulated trial encoding process was defined by, the last accepted configuration modified by the current test symbol.

There was no real difference in the overall function optimization for the two techniques as both appeared to fall into local minima. The genetic algorithm should be more robust to such problems, in particular *speciation (niche exploitation)* [91] could be used. In using speciation a number of gene population members would be required to search each crease and thus clusters of population members search the various creases. Covering the search space adequately, would imply a large gene population for this problem. A trial attempt at optimization gave very similar results to the simulated annealing algorithm probably because a smaller population (≈ 30) without speciation was used. Due to the larger over-heads associated with the genetic algorithm, particularly if sharing functions (for speciation) were to be incorporated, simulated annealing was chosen for optimization. However an attempt was made to improve the performance of the simulated annealing algorithm to decrease the computational effort.

Rutenbar [111] claims that some functions with:

> "a mostly flat landscape with numerous, densely packed gopher holes, each of widely varying depth ..."

may be impossible to anneal. From the profiles shown above this is not exactly the situation here but there appear to be many creases of varying depth and scale. To improve the performance a tactic of restarting the simulated annealing algorithm was chosen. Figure 6.9 shows the results: standard simulated annealing (which gave similar results to the standard genetic algorithm) is denoted as *con.ann* trace, simulated annealing with a restart after 20 step failures as *con2.ann*, and simulated annealing with a restart after 10 step failures as *con3.ann*. Restarts were determined by the rules:

1. if after N temperature adjustments no better solution was found since the last best configuration, restart with the best configuration found so far;

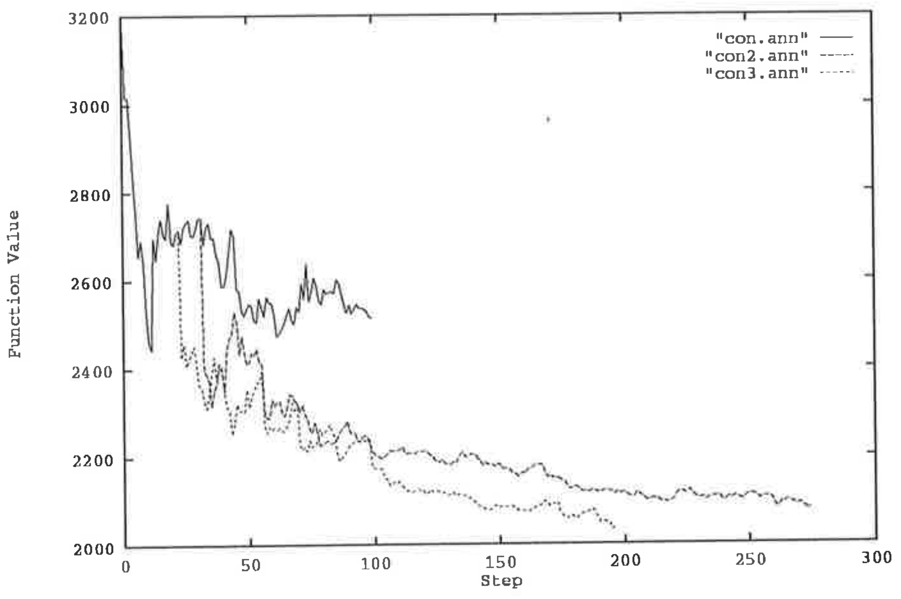


Figure 6.9: Evaluation Profile of Simulated Annealing Algorithms: without: *con* and with restarts: *con2* and *con3*.

2. if a better configuration is found than the current best, update the current best and reset the *failure to improve* counter.

Figure 6.9 was generated for the synthesis of 3 blocks (600 samples) of voiced speech data. This kept the computation time down to realistic values for the process of algorithm tuning. It was quite clear that the restart strategy improved the performance of simulated annealing for this problem. The number of function evaluations for each temperature step was lower bounded by 30 as the minimum number of evaluations and 512 as the upper limit. Each temperature step had to achieve 30 accepted reconfigurations or reach the limit of 512 evaluations before the next temperature step was taken. The strategy of a restart after 10 temperature trials with a *failure to improve* gave good results for this problem, however the computation time was still too great. For the traces in Figure 6.9 each trace required 10 hours CPU time on a SUN SPARC-station 2. The goal was to optimize the trellis for 6 seconds of speech, uttered by a male and a female speaker. This duration of speech totals 300 blocks of speech; clearly even the optimized simulated annealing algorithm would take too long to produce a good result on other than a super computer.

A different optimization technique was sought, suitable for standard architecture computers, which could cope with the target 6 seconds of speech.

Conjugate Stochastic Optimization

The target function which was to be minimized had a *surface* that may be viewed as being *rough*, which implied no gradient information could be utilized to find the minimum. An analysis of the obtained function profiles was undertaken to determine if there was any exploitable structure.

Rough non-differentiable surfaces are common place in nature and have recently been characterized using the concept of fractals [112]. If the function surface is viewed in terms of fractals it should have a random nature and show statistical self affinity. Self similarity is easier to explain so it will be explained first. Consider a curve $X(t)$ with independent variable t. Then

$$\text{Prob. of } X(t) - X(t_0) \ = \text{Prob. of } \tfrac{1}{r^H}(X(rt) - X(t)) \quad t_0 < t \quad r > 0 \tag{6.29}$$

where r is the scaling ratio.

The statistics of the curve are identical for the index t with respect to amplitude differences and the corrected amplitudes relative to the index rt. The power H governs the roughness and for H=1/2 the curve becomes that of Brownian motion.

Self affinity goes a step further and allows the different axes to have different scaling factors r_i.

The statistical nature of the function surface may be easily argued from the nature of the encoding algorithm where limited set of paths in the (M,L) algorithm move down the trellis. When a branch symbol's value reaches a particular threshold value it may no longer be a member of the path set at t_n although it was below (or above) the threshold. There is a high probability the branch symbol will still be used but it would now be used in a different context, with different symbols preceding and succeeding it. This threshold phenomenon causes a random jump to some other value of the evaluation function (mean square error value). To support the self affinity argument, the data that was available as the function cross-sections (Figure 6.8) was analysed to create a log-log plot of variance against step size, Figure 6.10. A linear plot is required to satisfy the properties of a self-affine surface. More formally the relation;

$$E[\Delta V(rt)] \propto r^{2H} E[\Delta V(t)] \tag{6.30}$$

should be satisfied where $E[.]$ is the expectation operation, t is the change in variable

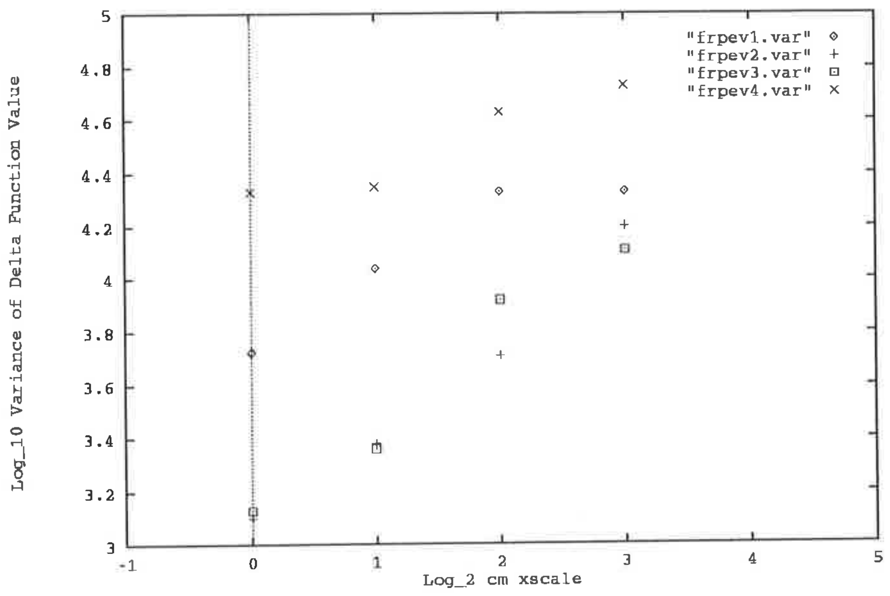


Figure 6.10: Plot to determine if the scale relationship required for self-affine curves was met by the mean square encoding error for regular pulse excitation.

parameter (Δt) and $V(.)$ is the function value. H is the exponent that is of interest in characterising the fractal dimension.

The graph of hand determined values obtained from Figure 6.8 is shown in Figure 6.10. Rough calculations from the limited data available indicate H is close to 1/2 (above and below) for the range of scales and show an approximate linear relation.

Generally to perform an accurate determination of fractal dimension the scaling relation should hold over a large range of scales and various probability moments should give consistent data [112].

The purpose of the elementary fractal analysis was simply to verify that the function surface did appear to be *rough* in the fractal sense in that the step variations occurred at a wide range of scales. This ruled out the possibility of using any derivative methods in finding an optimum. As the surface showed these characteristics, the aim of the optimization strategy was to develop an algorithm that could handle rough surfaces.

Some of the desirable features that the optimization algorithm had to incorporate were:

1. that it would not get *stuck* in a local minimum within its range of search,

2. that it did not require any gradient information,
3. that it would search the function surface *efficiently*,
4. that it would not be greedy and try to go to a minimum directly (thereby increasing the probability of getting stuck),
5. that it would cater for various scales of function variation.

Although Powell's (1964) [113] method does not necessarily use gradient information the surface to be optimized was too rough and the algorithm would in all probability go to a local minima all too readily.

It was considered important that the algorithm was not greedy so that it would not lock into a local minimum and search a limited region of the overall parameter space. A Brent's [114] type algorithm was avoided as there was a possibility that several minima could exist within the bracketing values.

The optimization procedure is summarized below where each variable parameter is considered varying along a coordinate axis in $\Re^n$:

1. Select a random coordinate axis and direction parallel to which the line of search occurs;
2. Choose a random starting point relative to the current position and generate a further M random steps from a distribution with a variance σ^2;
3. Evaluate the function at each of the M steps and if the value at the minimum is less than the current best, accept the value for the current configuration space. If the minimum value is greater than the best so far by some small percentage, accept it with a variable probability starting at 0.5 and decreasing ;
4. When the number of searches made at a particular scale reaches a set limit T, set $\sigma^2 = \sigma^2 * 0.5$;
5. If the number of iterations has reached the iteration limit or function improvement has reached the desired resolution limit, exit; else Goto Step 1.

The configuration space was defined so that there were boundaries (limits) for the values that any particular variable parameter may take. Steps giving values at which the function was evaluated, were generated from a uniform distribution which had

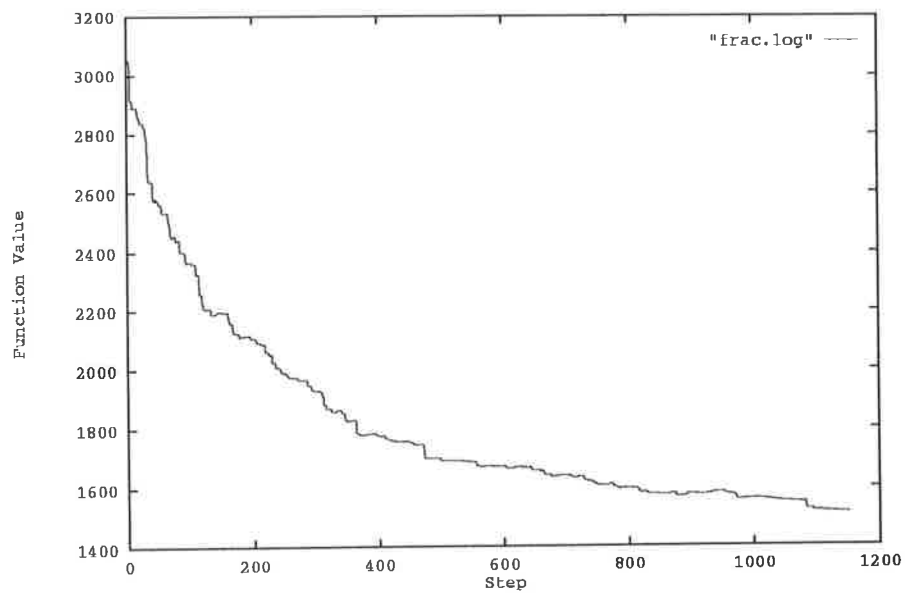


Figure 6.11: Evaluation Profile of Conjugate Stochastic Optimization Algorithm

an adjustable range which was directly related to σ^2. Adjusting the range scaled the average step size that the algorithm took in a direction. Steps for a line search were in a fixed direction which required the implementation to randomly choose an axis direction and keep stepping using a random step size. On average the step size expectation $E[stepsize]$ was 1/2 the range (R) set for the uniform distribution ($0 < stepsize < R$). To cater for the boundaries a simple approach was used that allowed the step to *bounce* off the boundary with the commensurate change in stepping direction.

The random selection of axis ensured a conjugate search direction which, together with the rule of not attempting to find an absolute minimum along the line of search, helped the algorithm avoid entrapment in local minima.

Use of random starting points along the line of search in the function space provided additional protection against getting stuck at a local minimum.

In implementing the algorithm the initial step size range was set to 1/2 the distance between the two boundaries and 7 search values were used along a line (axis) search. An average of 3 searches per axis was chosen before the average step size was again reduced by a factor of 1/2.

Figure 6.11 clearly shows the dramatic improvement of the conjugate stochastic (CS) algorithm compared to simulated annealing even with restarts (Figure 6.9) for the same problem. It should be noted that each step in the CS algorithm was 7 computations whilst in the simulated annealing algorithm each step comprised between 30-512 computations with an average of approximately 90 computations.

The range scaling was altered to 1/3 instead of the initial 1/2, however no improvement or substantial detriment was found, except that slightly fewer evaluations were required to find a good minimum. Fine tuning of the parameters was not attempted as a significant improvement (overall reduction of approximately two orders of magnitude in the number of function evaluations) was made over the other algorithms tried and it was considered unlikely another order of magnitude improvement could be achieved.

6.6 Coding Performance

Overall encoding performance is difficult to measure and is best done using listening tests. At a residual encoding rate of 1/2 bit per residual symbol, a ***hoarseness*** was evident in the reconstructed speech. A spectrogram of the reconstructed speech showed the typical banding due to the pitch, although with less definition than the original. A region of noise was evident between 1.5 and 2 kilohertz. The encoder implementation was checked for faults but none were found and it was concluded the noise was a phenomenon of the encoding mechanism.

The metric measure generally used in conjunction with the (M,L) algorithm was the mean square error. An attempt was made at finding a metric that gave superior subjective performance to mean square error. Each metric evaluation for the short vector was accumulated to obtain the accumulated metric for each path. A range of metrics were tried to determine if any improvements could be made on the mean square error measure. Letting the term *ref* denote the source or reference vector and the term *syn* denote the synthesised vector or trial vector, the variations attempted may be summarized as:

- $|ref - syn|$,
- $|ref - syn|^3$,

It was found that no other method substantially improved the quality of the speech and in most cases degraded it. Due to the simplicity of the mean square error measure

	32 States		64 States		128 States	
Number Paths	Ave. Best Metric	Ave. Spread	Ave. Best Metric	Ave. Spread	Ave. Best Metric	Ave. Spread
8	75,393	4,458	81,063	4,758	87,738	4,353
16	72,347	3,380	73,574	3,301	77,016	2,955
32	71,727	3,008	69,975	2,687	72,310	2,458
64			68,944	2,341	69,940	2,113
128					69,551	2,051

Table 6.2: Comparative Best Metric Performance of Trellises and Number of Paths: the spread indicated is between the best metric and ranked 16th path metric.

it was retained. It should be noted that the speech and the synthesised speech were perceptually weighted for the tests.

Optimized trellises of different numbers of states were used to encode a source file of five different speakers of different ages and sexes and then compared in terms of the best average error metric. The number of paths used in the (M,L) algorithm was also allowed to vary. Results are presented in Table 6.2 for the rate 1/2 bit per speech sample.

Care must be taken in interpreting these results as a lower metric between trellises did not necessarily imply a better performance, for example, the difference in performance between 8 paths and 16 paths for the 128 state trellis was barely audible. It should also be noted that this average metric was taken over voiced and also unvoiced speech where comparison of the trellis matching may not be particularly meaningful.

Informal subjective tests using quantised a_i's were undertaken to attempt to evaluate the results found in the table. Listening to speech encoded with 16 paths for each of the trellises listed showed little difference between them except the 32 state trellis did produce slightly rougher speech. There was an improvement when 64 paths were used. The 64 state trellis with 64 paths and the 128 state trellis with 64 paths showed little difference between them and each produced less noisy speech when compared to their 16 path counter parts. For the trellises and path combinations listed above a slight preference was given to the 64 state 64 path[4] configuration.

Tests were also taken to confirm that one gain term based on prediction error for the

[4]This is equivalent to the Viterbi algorithm for 64 states.

whole block performed as well as 4 terms based on r.m.s. input level for each of the sub-blocks. It was found that there was no audible difference when only one gain term was used, which implied that the lower transmission bit rate for transmitting only one gain term could be used without producing any noticeable degradation.

It can be seen that all of the trellises show a knee in the average best metric characteristic when the number of paths equals 1/2 the number of states. This implied on face value that little is gained by going to full Viterbi encoding. However the sorting process in the (M,L) algorithm has a computational order of $M \log M$ while the add compare select mechanism, although slightly more complex, is of the order $2N$. (Where N is the number of states). Hence if $N/2$ paths or more are planned to be used in the trellis encoding full Viterbi encoding is attractive.

Figure 6.12 show the speech waveform of, *"The hat brim was wide and too droopy."*, uttered by a male speaker. The coded waveform using a trellis that employed a rate of 1/2 a bit per speech sample is shown in Figure 6.13. The trellis was optimized using the algorithm described in the chapter. It is clear that the coder tracks the waveform, although listening tests were necessary to evaluate the perceived accuracy. A magnified detail of the original and synthetic waveforms is shown in Figure 6.14 where it can be seen that the zero crossings were preserved quite well and the amplitude matching was reasonable. Some of the finer details of the original waveform however were not well reproduced.

Figure 6.15 shows the result of coding using the untrained trellis at a rate of 1/2 bit per speech sample. The untrained trellis used regular pulse excitation where the pulses came from a Gaussian sample population. The mean for the Gaussian population was set to zero and the maximum values of the trained trellis were taken as approximately three standard deviations for this population. The resultant standard deviation was 3000 quantisation units. It is quite clear from the waveform that the training produced superior results. Subjective tests clearly demonstrated that the training process was important in improving the quality of the reproduced speech.

When the code rate was lowered to 1/4 bit per speech sample the reproduced waveform showed gross distortions. Subjective tests verified that the encoding algorithm became unstable at these rates. Large sections of speech encoded at this rate proved to be totally unintelligible. Figure 6.16 shows the distortions quite evident in the waveform envelope.

Informal listening tests were performed to determine the effect of varying γ in the

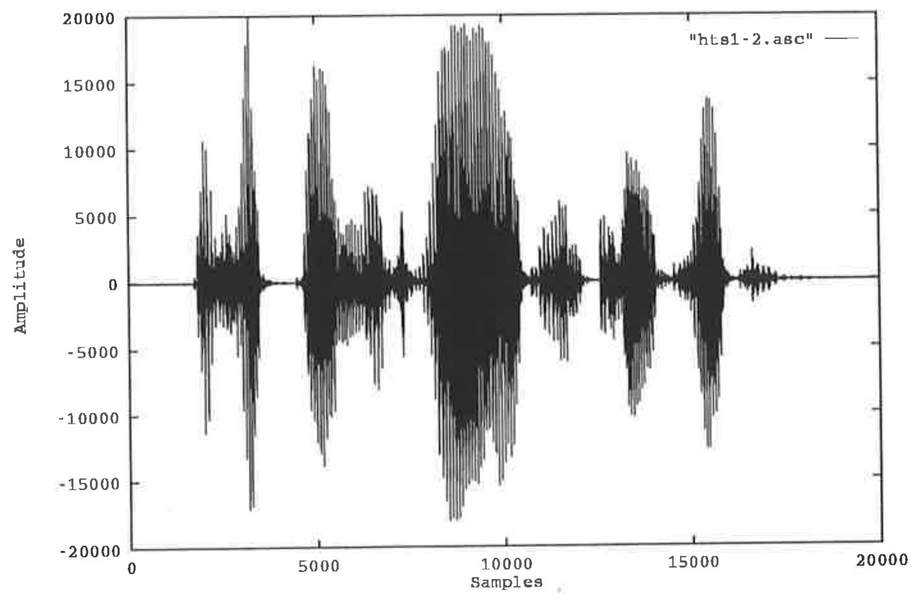


Figure 6.12: Original Waveform Uttered by a Male Speaker.

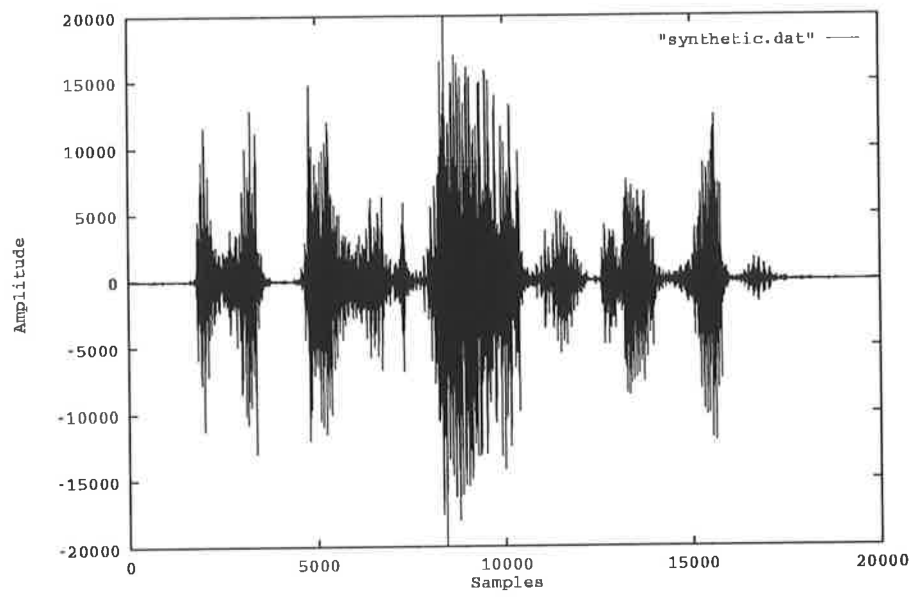


Figure 6.13: Coded Waveform for Rate 1/2 Bit per Sample Trellis.

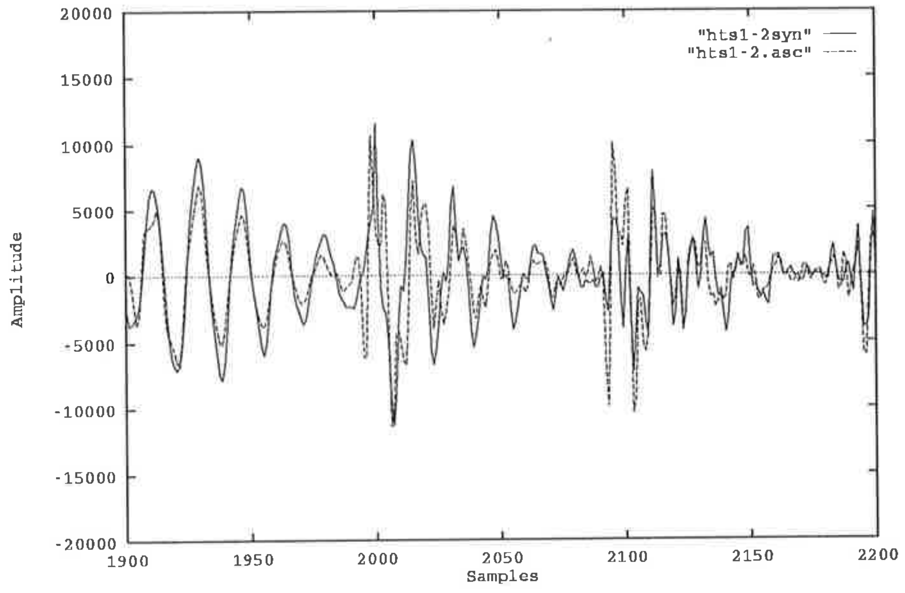


Figure 6.14: Detail Showing Comparison of Original and Synthetic Waveform.

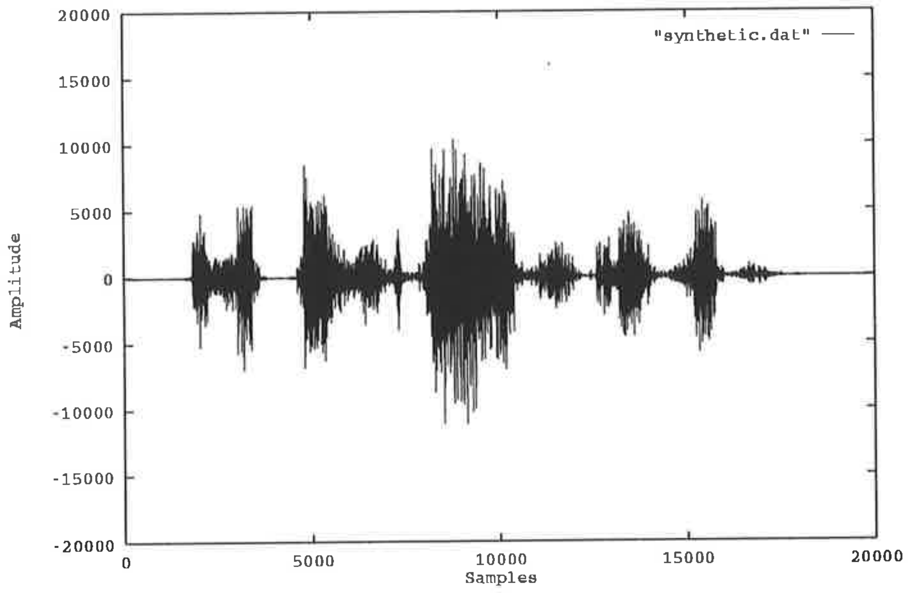


Figure 6.15: Coded Waveform for Rate 1/2 Bit per Sample Untrained Trellis.

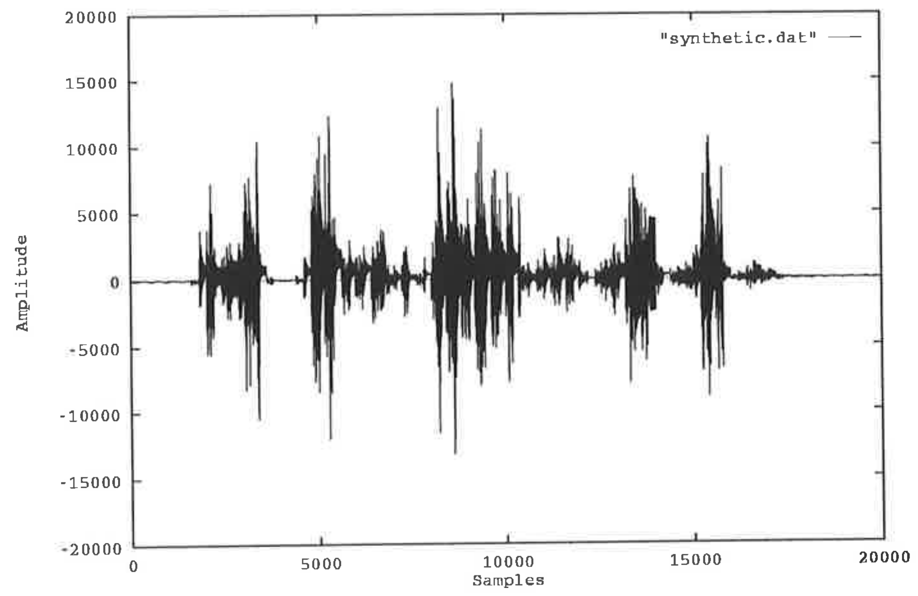


Figure 6.16: Coded Waveform for Rate 1/4 Bit per Sample Trellis.

perceptual weighting filter. The normal range used in CELP coders is from 0.75 to 0.9. Listening tests indicated that the larger values were preferable which prompted trials for values above the normal limit. A final range of 0.95 to 0.97 was chosen as yielding the most pleasant listening values. These values were significantly larger than those used with standard CELP or multipulse vocoders.

Comparing the coder developed here to the earlier designs showed that the trellis encoded residual encoder performed well. Speaker intelligibility was very good and recognizability was good although it displayed a slight hoarseness. A segmental signal to noise ratio objective evaluation indicated the developed coder had a performance of 5.3 dB at 5.5 kbps while the coder by Araseki et. al. had a performance of 5 dB at 6.5 kbps. The original regular pulse excitation coder as described by Kroon [16] operated at the significantly higher bit rate of 9.6 kbps and achieved a commensurately higher segmental to noise ratio of 13 dB.

The final bit rate for the fully quantised coder, excluding forward error correction bits, was determined to be approximately 5.5 kbps as follows:

1. synthesis filter poles information — 32 bits;

2. scaling gain term — 5 bits;

3. speech residual information — 100 bits;
4. giving a Total — 5,480 bits per second.

Speech frames consisted of 200 samples at a sampling rate of 8 kilo-samples per second. This implied a frame duration of 25 ms. The gain term for the residual did not require a sign as the trellis encoding accounts for the sign.

6.7 Post Processing of the Synthetic Speech Using Pitch Information

In coding the residual there were some sounds such as *eye* that the coder did not handle well at low bit rates; the sound was reproduced with a gravelly quality. In particular the pulses defining the start of period were not always reproduced well. It was noted that the coder did encode the start of the quasi-period but not with as large a pulse as was desirable. A post processing technique was considered which used pitch information transmitted by the encoder expressly for the purpose of post processing the speech. This overhead implies an extra 280 bits per second (1/25 ms $\times$ 7 bits) additional to the base transmission rate.

Use was made of the idea put forward by Granzow et. al. [110] that a single main pulse dominates the start of a quasi-period of the residual. The residual encoding mechanism attempts to replicate this situation with a varying degree of success. During unvoiced speech there was no real problem in replicating noisy speech as the ear has difficulty in establishing differences in the noisy part of speech. However waveform coders generally have trouble with voiced speech as they have difficulty in making the 'pitch' periods similar enough.

The post processing technique uses the idea of using single synthesis filter impulse responses spaced at linear interpolated *pitch periods* to simulate voiced speech; much like the LPC-10 vocoder. The trellis encoded speech data however has much more information encoded into it than the LPC-10 implementation. Using a maximum correlation approach the start of voicing could be found from the synthetic waveform and it's position marked.

Processing continued using the marker concept where the search for the next marker is always relative to the last marker. Interpolated pitch information was only used as a

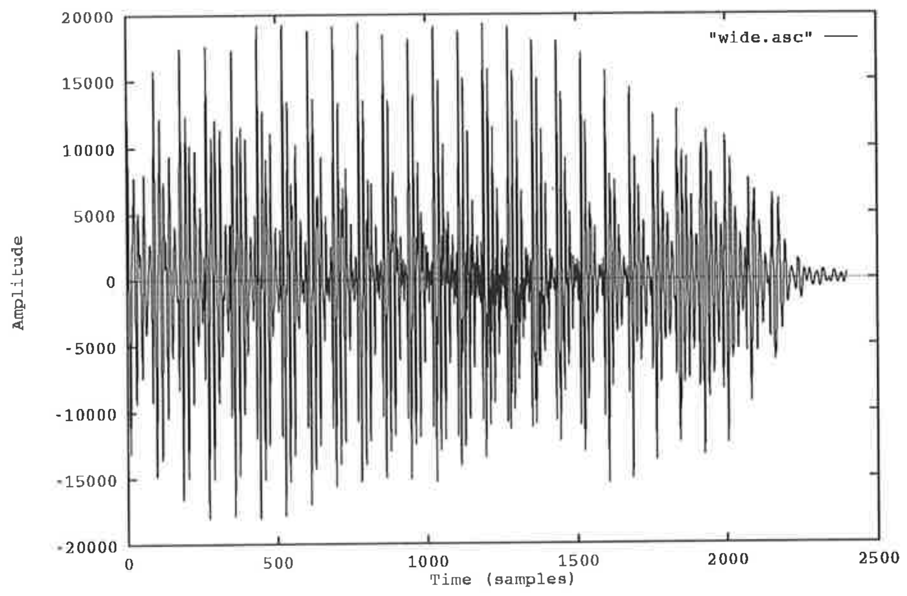


Figure 6.17: Original Waveform of *wide* Spoken by a Male Speaker.

guide or estimate of where the next marker should be. A region of a few samples before and after the estimated marker location was searched for a large cross-correlation with the impulse response of the synthesis filter. The location of the greatest correlation, positive or negative, located the next marker. This mechanism allowed for slight variations in the quasi-period and tracked the start of the periods quite well. The marking process continued to the end of all speech blocks considered to be voiced. The marking mechanism was automatically reset at the completion of a voiced sound, after which a start of voicing search based primarily on gain values was required.

A buffer of synthetic speech consisting of impulse responses from the synthesis filter was constructed using the now known period markers. The amplitude of the impulses was determined by multiplying the scaled gain parameters transmitted with the encoded speech by two.

Enhancement of the speech was effected by constructing a weighted sum of the impulse responses with the synthetic speech. This technique avoided the problem of the synthetic voice sounding synthetic as is the case with the LPC-10 vocoders. It should be noted that the addition process was synchronous with the synthetic speech which avoided difficulties due to phase misalignments between the two waveforms.

It can be seen from Figures 6.17 and 6.18 that the encoder had difficulty in replicating

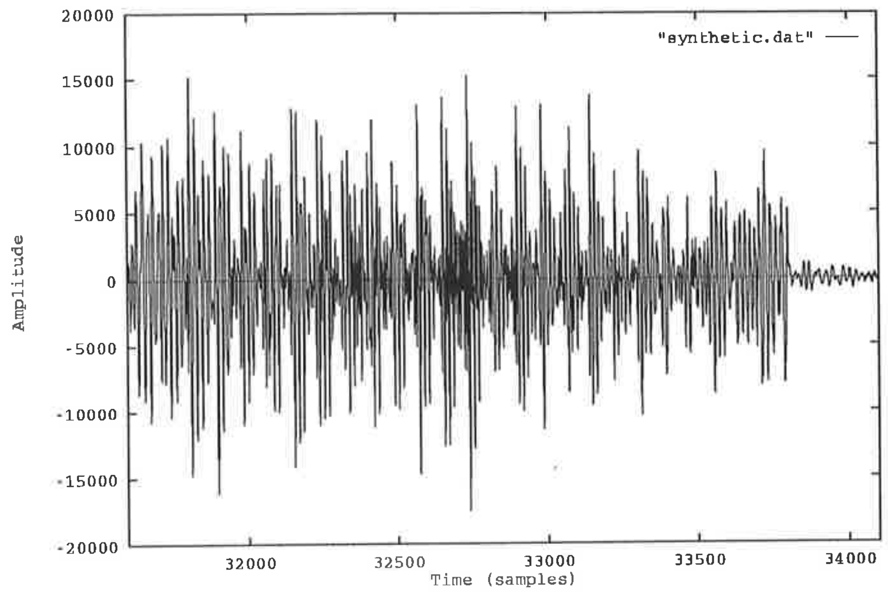


Figure 6.18: Reconstructed Coded Waveform of *wide.*

the waveform. (The capabilities of the encoder are discussed in more detail in chapter 6.) In particular the start of the periods was not always clearly reproduced. The suggested post processing was used to specifically try and enhance the start of the periods.

The enhanced speech shown in Figure 6.19 was constructed using a 30% magnitude weighting of the original speech and 70% weighting of the impulse response buffer. It was found that from informal listen tests that the weighting ratio 70% original to 30% impulse response buffer produced pleasant speech which was more *melodic* than the original synthesised speech without sounding too artificial.

Synthetic speech from the coder sounded *hoarse* while the speech from the impulse buffer sounded *buzzy* with no trace of hoarseness. Mixing the two sounds called for a trade off between the hoarseness but more natural sound and the buzzy synthetic sound. Listening tests were conducted for the original synthesised speech to pitch synthetic speech ratios ranging from 80%:20% to 50%:50% which was considered to be the acceptable performance region. The 70%:30% to 60%:40% mixture reduced the hoarseness without introducing too much buzziness improving the synthetic speech in terms of making it more pleasant to listen to. When the ratio reached 50%:50% the speech started to sound artificial with mild *buzzy* and *bubbly* artifacts.

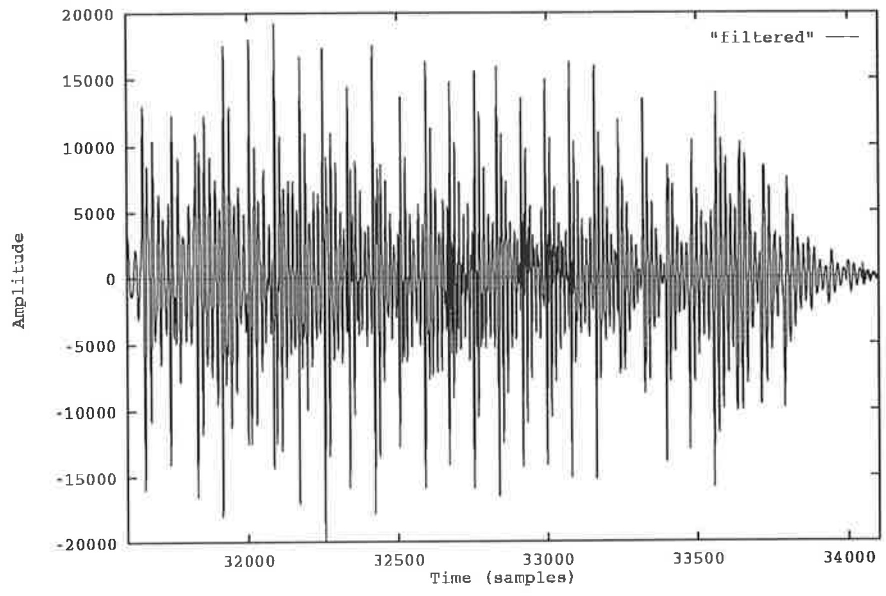


Figure 6.19: Enhanced Version: post processing of *wide*.

The post processing of the synthetic speech did make the speech more pleasant to listen to as the hoarseness of the synthetic speech did become tiring. The modest overhead of the extra 280 bits per second was deemed worthwhile to improve the perceived speech quality.

Chapter 7

Analysis of Trellis Coding

7.1 Introduction

This chapter deals with characterization of errors in the synthetic speech waveform due to trellis encoding of the residual. Two types of errors are addressed. Firstly the errors caused by channel errors and secondly errors due to the encoding process.

Marcellin and Fischer [101] comment that for their trellis source coding scheme, a received error only affects $\log_2 N + 1$ output symbols where N is the number of states in the trellis. This comment is no longer valid if a predictor, such as a speech synthesis filter, is driven by the trellis output symbols. In fact the impulse response of the synthesis filter dominates the length of the error event. What must also be considered is the different paths, and hence output symbols, generated by an error event. A sequence of incorrect states is generated by a bit error till the erroneous bit no longer has an effect on the state selection. These factors are discussed in more detail below. The term tree will be used in this chapter as a trellis of excitation values is used in conjunction with a predictor generates a code tree.

It was found in listening tests and observation of the synthetic speech waveform that the encoding process causes a very rapid degradation of the reproduced signal as the tree encoding of the residual was lowered from 1/2 bit per sample to 1/4 bit per sample. Listening tests of the reproduced speech revealed gross distortion, to the extent that the output was just recognisable as speech. In order to try and determine the mechanism causing the rapid degeneration of performance, a model was constructed that attempted to capture the fundamental dynamics of the encoding system.

The techniques developed here reference a new approach to the analysis of error events in tree source coding problems. The problems referenced here are not well covered in the literature.

7.2 Channel Error Analysis

The trellis encoded excitation is received as a stream of bits that determine the sequence of states as the trellis or tree is traversed. The traversal of the tree allows the receiver to generate a set of (branch) symbols which are fed into a synthesis filter which reconstructs the synthetic speech waveform.

State determination from the bit stream can be likened to moving a window of size $\log_2 N$ bits over the bit stream, where N is the number of states in the residual trellis. The bits in the window are viewed as a binary encoding of the current state. As a new bit enters the window the old state number is doubled and the new bit added; the resulting number is then taken modulo N. Clearly this constitutes a feed forward mechanism where bit errors in the received stream are not propagated indefinitely. The synthesis filter that the branch symbols (synthetic residuals) are fed into is however an infinite impulse response system. An error event thus in theory has an infinite memory. However in practice this is not the case as the synthesis filter memory is effectively flushed out during unvoiced speech and during *silent* periods. However an error event during voiced speech does have a significant effect as the speech power at that time is high.

To study the memory effect of the speech synthesis filter the decay rate of the impulse response was determined. Initial attempts of using bounding arguments (e.g theorem due to Gershgorin [115]) on the magnitude of the poles had to be abandoned as the bounds were not tight enough. A more direct approach was taken to determine the decay rate, where the impulse response was generated using the speech predictor filter and measured directly. The procedure used was to find the initial maximum of the response, followed by a skip of 100 samples, and then finding a series of consecutive local maxima.

The decay rate was defined in this context as the time taken for the impulse response to decay to 1/10 the initial maximum. It was deemed that the impulse would contribute little to waveform distortion once it had fallen to this level. The skip of 100 samples was found from empirical evidence to save computation time and used the observation that only the poles closest to the unit circle have an effect of long duration. A series of local maxima was found on the decay curve, to allow for the effect of beats. If more than one pole pair was located at the approximately the same distance from the unit circle, in a beat in the waveform results. The outcomes from the four primary test files of two males and two females each uttering 2 phonetically balanced sentences are given in Table 7.1.

File Name	Ave. Decay Rate (samples)	Ave. Pitch (samples)
hts1	101	85
hts2	98	44
hts3	80	97
hts4	73	44

Table 7.1: Average Decay Rates and Pitch Periods in Samples (8 kHz sampling): four speech files of male and female speakers were used.

In determining the values for the table, only voiced frames were considered as spurious pulses in speech residual for unvoiced speech had little to no audible effect. The regions of speech where the speech power was appreciable were generally voiced sections. Using this assumption only the voiced segments were considered. During voicing the poles of the synthesis filter tended to be closer to the unit circle and the impulse response had a longer duration. From the table it may be noticed that the female and male speakers' decay rates are similar. An erroneous pulse in synthesis of speech during a voiced section of speech had a similar effect whether the speaker was male or female. It should also be noted that the decay rate was much greater than the memory length ($\log_2 N$) for the candidate trellises. This implied the number of states (N) in the encoding trellis did not have a great bearing on the size of an error event when viewed with respect to the resulting synthetic speech waveform. Another effect present, which was harder to quantify, was the effect of scaled, phase-shifted and summed impulse responses representing a voiced pitch period. For voiced speech however, a reasonable assumption [110] is that the waveform quasi-period is dominated by a primary pulse.

Employing the idea that the output of the synthesis filter was the linear super-position of time shifted impulse responses, any major errors in the output waveform would be due to a large pulse contribution from a trellis branch. A large incorrect pulse may have been due to a large difference between the desired correct pulse and the incorrect pulse. The difference may come about as a difference between pulses of the same sign or between pulses of opposite signs.

An error event consists of a sequence of incorrect branch symbols before going on the correct track again. In order to estimate average error events all combinations of error and non-error paths must be compared. To simplify the analysis only the greatest difference between the error and non-error path's pulses were considered to be significant. This was a simplification but using linear super-position the largest difference will contribute the largest error to the final waveform. The decay rate on average, on a voiced section of speech, was much longer than a single bit error event so the exact position of the greatest difference could be ignored.

Statistics that characterise an error event were determined directly from the trellises and consequent tree codes. The optimized trellises of 32, 64 and 128 states were investigated. For each starting state an arbitrary error event may terminate in any other state. The maximum path differences were determined by pair wise comparison of all possible paths through the trellis on an error event. An algorithm to achieve all path pairs through the trellis may be described by:

1. **for** each possible starting state;
2. **for** each possible finish state;
3. With an input bit 0 and current start state, follow with a string of bits to assure termination in the current finish state, collect the sequence of branch symbols and mark this sequence-0;
4. With an input bit 1 and current start state, follow with the same string of bits as the step above to terminate in the current finish state, collect the sequence of branch symbols and mark this sequence-1;
5. Find the largest magnitude difference between sequence-0 and sequence-1;
6. Increment the count for each threshold bin for which the difference is greater than the threshold level;
7. End **for** loops.

This process resulted in the sum of single error events, determined from the comparison of all possible correct and error path pairs for the investigated trellis, where the maximum difference was greater than a given threshold. Before going further the circumstances of error events should be analysed to indicate why certain decisions were made in the characterization of the pulse errors.

Let the probability of decoding which results in an erroneous pulse, x_{dmax}, of magnitude greater than threshold T_{thresh} be expressed as;

$$P_E(x_{dmax} \geq T_{thresh}) = \sum_{n=0}^{\infty} P_E(x_{dmax} \geq T_{thresh}|n-errorevent)P(n-errorevent) \quad (7.1)$$

where x_{dmax} is the maximum magnitude difference between any pulse pair on comparing the correct path and error path.

An $n-errorevent$ is defined as an error burst of length n followed by a guard length of $K = \log_2 N$ non-errored bits, where N is the number of states in the trellis. An error burst is characterized by a start error bit followed by n-2 bits which may or may not be in error, followed by a terminating error bit. To qualify as a single burst, $(n-2) < K$. The guard length is necessary for the decoder to clear any erroneous bit out of the decoder memory. If a binary symmetric channel is assumed, P_E is given by;

$$\begin{aligned} P_E(x_{dmax} \geq T_{thresh}) \quad = \quad & P_E(x_{dmax} \geq T_{thresh}|1 - error\, event)P_e P_f^K + \\ & \sum_{k=0}^{n-2} P_E(x_{dmax} \geq T_{thresh}|(k+2) - error\, event) \times \\ & P_e^2 P_f^K C_k^{n-2} P_e^k P_f^{n-2-k} \end{aligned} \quad (7.2)$$

where P_e is the probability of receiving a bit in error, P_f is the probability that the bit is error *free*, and C_a^b denotes the combinatorial function. If the probability of channel bit error $P_e \ll 1$ only single error events need to be considered, that is, an error event has 1 bit in error followed by K error free bits.

It was desirable to describe the maximum error pulse information as an overall probability given that a single bit error event had occurred. To do this, the probability of being in any particular state needed to be determined. If the probability of being in a state at time t is denoted $P(S_j)$ and the probability of a transition from state S_i to state S_j over a path of $K+1$ bits as $P(S_{ij})$, the probability of the maximum error pulse being greater than the threshold T_{thresh}, given that there was an error event can be lower bounded by $P_{\acute{E}}$ where;

$$P_{\acute{E}}(x_{dmax} \geq T_{thresh}) \approx \sum_{i,j=1}^{N} P_{\acute{E}}(x_{dmax} \geq T_{thresh}|S_{ij})P(S_{ij}) \quad (7.3)$$

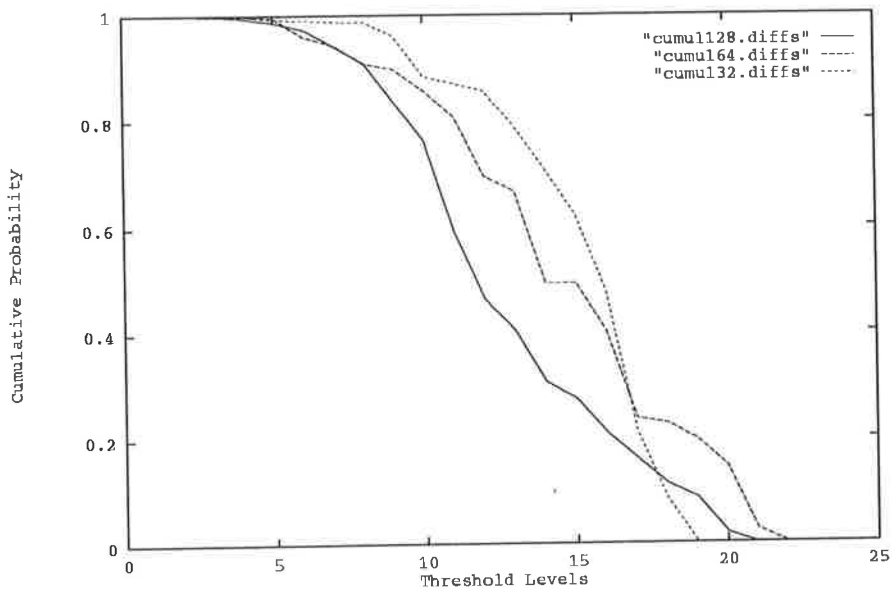


Figure 7.1: Cumulative Probability Distribution for Maximum Error Events Greater than a given Threshold: *cumul128, cumul64, cumul32* refer to 128 state, 64 state and 32 state trellises respectively.

The probability of being in a state is source dependent so a relatively long speech segment, of five people of mixed sexes and ages each uttering a single sentence, was encoded. The resulting state sequence was used to obtain relative probabilities of each state at an arbitrary time. An assumption was made that the finishing state on a transition was independent of the starting state. This assumption was reasonable provided the encoding rate was low (a relatively large number of input samples lay between the start state and the finish state of the encoding trellis). It was found in practice that at low rates the quantisation process was quite noisy, resulting in very similar accumulated metrics for the different candidate paths.

$$P_{\acute{E}}(x_{dmax} \geq T_{thresh}) \approx \sum_{i,j=1}^{N} P_{\acute{E}}(x_{dmax} \geq T_{thresh}|S_{ij})P(S_i)P(S_j) \tag{7.4}$$

The algorithm described above was run again with the accumulation process weighted by the probability of starting in a particular state S_i and the probability of ending in a particular finishing state S_j. Figure 7.1 shows the experimentally determined lower bound for the cumulative probability distribution of errors greater than a given threshold for pulse magnitude error for the source data investigated.

The average pulse magnitude from all branches in the trellis were also tabulated for comparative purposes in Table 7.2.

Number of states in Trellis	Average Pulse Magnitude
128	4.105
64	4.471
32	5.55

Table 7.2: Average Magnitude of Branch Values

From the graphs in Figure 7.1 it may be noted that for an error pulse of magnitude greater than 7 and less than 17, the 128 state trellis performs best. This is in the sense that for an error event pulse of magnitude 15 for example, the 128 state trellis presents an erroneous pulse of this magnitude or greater with a probability of approximately 0.26. The 64 state trellis yields a pulse on an error event of the same magnitude with probability approximately 0.46 and the 32 state trellis with approximate probability 0.63.

As an error event has a relatively long duration compared to the bit error determining the state, the larger code books have an advantage in the presence of channel errors. This is because the values on the branches of the trellis cover the range of values less coarsely than those of the smaller trellises. The average magnitude of the branch entries also supports this property of the quantisation process.

Table 7.2 shows that the average pulse magnitude was greater for the smaller codebooks, which corresponded to a coarser quantisation procedure.

7.2.1 Informal Listening Tests

Informal listening tests confirmed that the larger trellises did perform better than the smaller ones over a noisy channel. When the noise level produced an approximate 10% error rate, the voice decoder system was essentially noise driven with no one trellis showing any advantage over the other; however the speech was still intelligible. The U.S. Federal Standard 1016 CELP coder was also very distorted under these conditions and gave different distortion characteristics. Although the errors were only allowed to affect the codebook indices, fluctuations in the noise level were quite audible. The Federal CELP's output also had a bursty quality to it which the coder in the thesis did not exhibit. The bursty noise most likely came about because the long term predictor used incorrect indices in the reconstruction of voiced speech.

At an error rate of 5% it was clearer that the larger trellises performed better but not in the way expected. The quality of the noise was different rather than the perceived noise level as may have been predicted. The larger 128 state trellis had a more mellow sound under these conditions than the smaller 32 state trellis. The 32 state trellis in these conditions had a rather *brittle* sound quality.

7.3 Regular Pulse Excitation Prediction Error

The aim in this section is to quantify the absolute minimum error level achievable using regular pulse excitation. The actual coding case has greater error as the encoding process is non-ideal.

For an autoregressive model with predictor coefficients a_i the next sample in the time series is;

$$x_i = \sum_{j=1}^{L} a_j x_{i-j} + w_i \tag{7.5}$$

where L is the order of the model and the first term on the RHS predicts the new value. The difference between the actual value x_i and the predicted value is w_i the ideal innovation. If the time series is estimated $\hat{x}$ the above equation can be rewritten as;

$$\hat{x}_i = \sum_{j=1}^{L} a_j \hat{x}_{i-j} + d_i \tag{7.6}$$

where d_i is the innovation estimate usually selected from a finite alphabet.

Let us assume that up to time $i-1$ perfect prediction has occurred. The squared error at time i is,

$$\begin{aligned} \epsilon &= (x_i - \hat{x}_i)^2 \\ &= (w_i - d_i)^2 \end{aligned} \tag{7.7}$$

If regular pulse excitation with one pulse every D samples is assumed the next $D-1$ estimates are 0's. Only the case of $D=2$ is considered. For the $i+1$th time step the ideal and estimated signals are given by;

$$x_{i+1} = a_1 x_i + \sum_{j=2}^{L} a_j x_{i+1-j} + w_{i+1} \tag{7.8}$$

$$\hat{x}_{i+1} = a_1 \hat{x}_i + \sum_{j=2}^{L} a_j \hat{x}_{i+1-j} + 0 \tag{7.9}$$

The squared error between the $i+1$th sample and estimate is now found to be;

$$\begin{aligned}(x_{i+1} - \hat{x}_{i+1})^2 &= (a_1 x_i + w_{i+1} - a_1 \hat{x}_i)^2 \\ &= (a_1(w_i - d_i) + w_{i+1})^2 \end{aligned} \tag{7.10}$$

where a substitution was made for $x_i - \hat{x}_i$. The squared error for the predictor using regular pulse excitation over $D = 2$ symbols is denoted by $\epsilon_{i,i+1}$ and is given by;

$$\begin{aligned}\epsilon_{i,i+1} &= (x_i - \hat{x}_i)^2 + (x_{i+1} - \hat{x}_{i+1})^2 \\ &= (1 + a_1^2)(w_i - d_i)^2 + 2a_1 w_{i+1}(w_i - d_i) + w_{i+1}^2 \end{aligned} \tag{7.11}$$

Now the best possible estimate of the innovation d_i minimizes the error $\epsilon_{i,i+1}$. This is found by differentiating with respect to d_i;

$$\frac{\partial \epsilon_{i,i+1}}{\partial d_i} = -2(1 + a_1^2)(w_i - d_i) - 2a_1 w_{i+1} \tag{7.12}$$

and setting this to 0. Solving now for d_i gives,

$$d_i = w_i + \frac{a_1 w_{i+1}}{1 + a_1^2} \tag{7.13}$$

This value of d_i is then substituted back to evaluate $\epsilon_{i,i+1}$ to give;

$$\epsilon_{i,i+1} = \frac{w_{i+1}^2}{1 + a_1^2} \tag{7.14}$$

Equation 7.14 represents the absolute minimum error possible when using regular pulse excitation with pulses of the optimum size every second pulse.

It is quite evident that using innovation estimates from a finite alphabet can only result in even greater noise in the encoding process.

7.4 Model of the Coding Mechanism

In the process of attempting to lower the bit rate of the residual encoding procedure it was discovered that at rate 1/2 the results were acceptable, however at rate 1/4 (or

1/4 of a bit per residual sample) the encoding mechanism appeared to go unstable. The instability was quite severe to the point where the reproduced speech was barely intelligible. An attempt was made to model the encoding process to determine the mechanism causing the instability.

Several simplifying assumptions had to be made to extract a simple model, however the dynamics of the process still had to be maintained. The assumptions made were:

- the pattern matching (minimum mean square error) of a set of candidate synthetic waveforms, generated from a set of possible *branch symbols*, to the incoming speech waveform was considered important;
- encoding at low bit rates generated a high degree of quantisation noise and for a limited number of encoding paths in the (M,L) algorithm, the retained paths had very similar accumulated metrics (allowing removal of memory from the model);
- actual symbols on the branches of the trellis that were fed into the synthesis filter were from a random population of bounded magnitude bi-polar values;
- encoding was from a selection of 32 levels;
- the synthesis filter was low pass and crudely matched the shape of some voiced vowels in speech.

The first assumption pertains to the encoding mechanism where the minimum mean square error was used to make a decision on the quantisation level chosen for a particular encoding of the original waveform. The general form of the encoder used in the work operated at one bit per two residual samples which resulted in a small range between accumulated metrics for candidate paths. A small range of accumulated metrics indicated that no one path had a large advantage over any other; which implied that the encoding mechanism was *noisy*.

Taking the noisiness to the extreme, the paths could be considered of identical merit and that the encoding depended only on the mean square error of the *current* pattern match. If the waveform match only depended on the current input symbols and not past symbols, an independent selection for the current estimate could be made. Selection of the current estimate was made from a fixed size alphabet that was generated for each time step. The selected estimate of the residual was immediately fed back into the system by the estimate entering the synthesis filter from where it was used in encoding future waveform symbols. Recall that a regular pulse excitation was used where, for example, an encoding rate of 1 bit per three residual samples implied, the branch symbol (residual estimate) entered the synthesis filter followed by two zeroes. Past decisions, as governed by the contents of the synthesis filter, controlled the selection of the next branch symbol or in the model's case the next quantisation level from the random alphabet.

A full complement of 32 quantisation levels was allowed in the model as the typical encoding work of the thesis used 16 paths, which allowed 32 decisions for each pulse. The 32 decisions in the actual (M,L) algorithm may not be as generous as the model's 32 random levels between the positive and negative set limits, however the (M,L) algorithm makes use of many past decisions via the accumulated metric. An open selection from 32 levels was deemed to be a good compromise, to having to model the codebook state memory.

It was found that instabilities in the encoding mechanism were most audible during voiced sections of speech. During voicing the power level of speech is higher than during unvoiced speech, and differences in noise-like sounds are more difficult for the ear to distinguish than voiced sounds. Consequently the voiced case was of more interest to research than the unvoiced.

The synthesis filter of the model was made *voiced-like* in shape by using three pole filters with appropriately placed poles. The primary difference between the two synthesis filters used was the tail of the frequency response of the synthesis filter, where the one filter's Figure 7.2 fell off more rapidly with increasing frequency than the second filter, Figure 7.9.

A chirp signal was chosen as input to the encoding model so that the response of the encoding could be observed for a range of frequencies. The input waveform is illustrated in Figure 7.3.

The progression of Figures 7.4–7.7 illustrate the way the encoding of the system became noisier as the rate was lowered to the point where four residual samples were encoded using one branch sample; at which point instability was evident. This type of response was in agreement with the listening tests performed on the actual encoder.

Other factors found that affected the instability were: reduction of the number of quantisation levels with which the encoding was performed, and the degree of mis-match between the set limits of the range of the quantisation levels and the dynamic range of the input signal. The set limits on the allowable range of quantisation governed

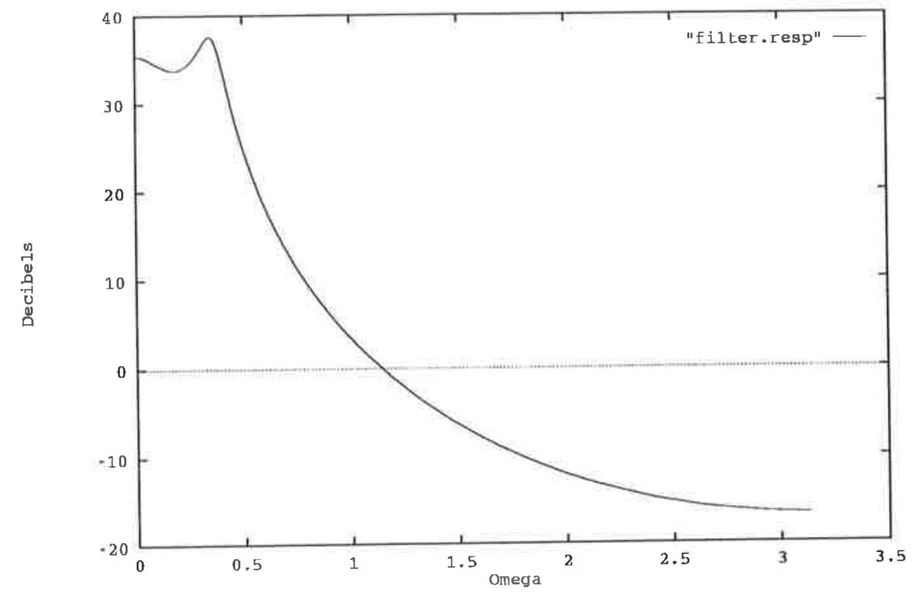


Figure 7.2: Frequency Response of Synthesis Filter 1 in the Encoding Model

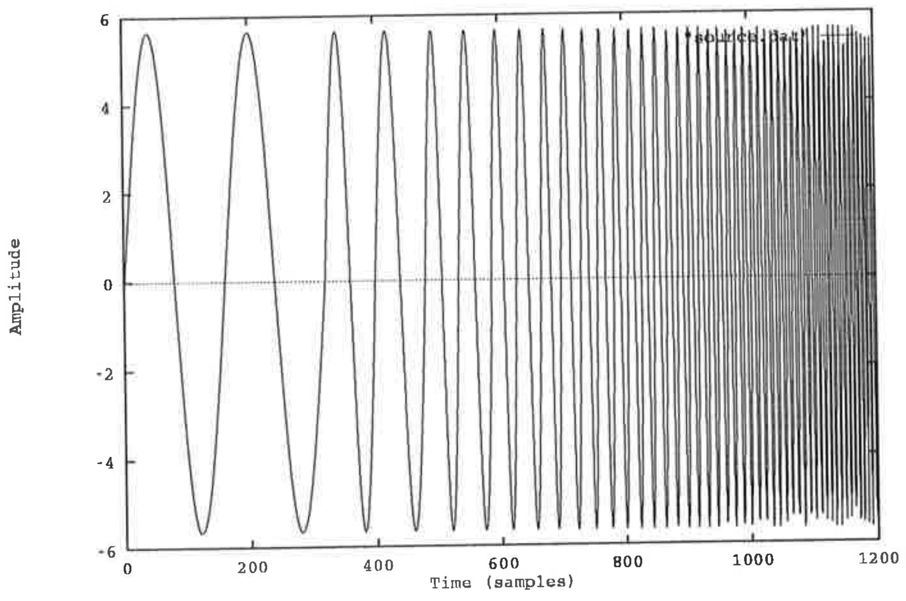


Figure 7.3: Chirp Input Waveform to be Encoded

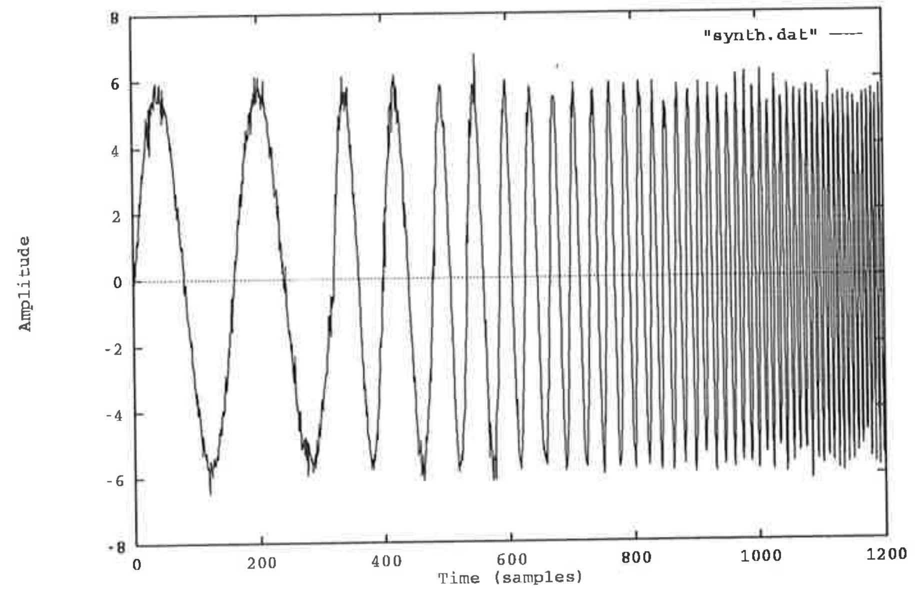


Figure 7.4: Reconstructed Waveform from Encoding Model; Residual Coding Rate 1.

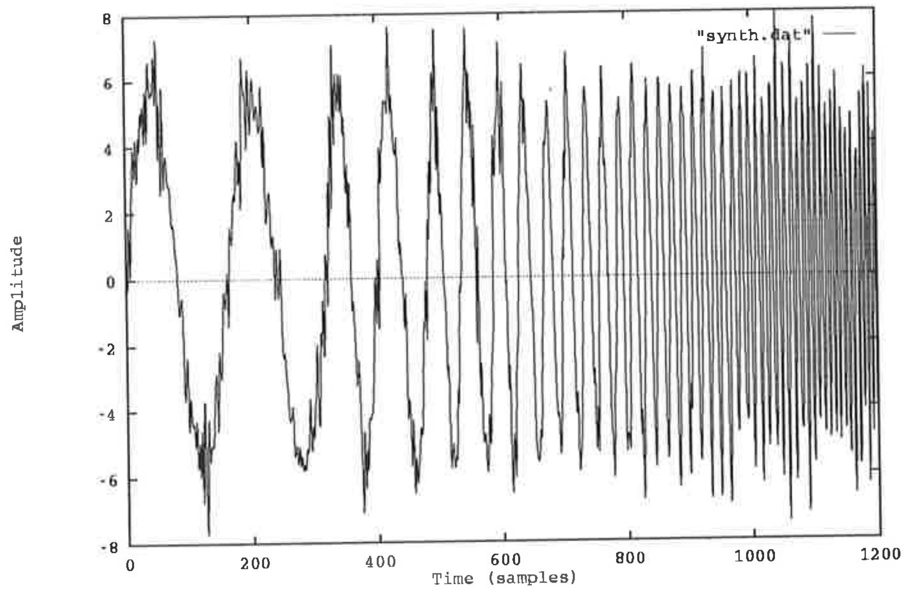


Figure 7.5: Reconstructed Waveform from Encoding Model; Residual Coding Rate 1/2.

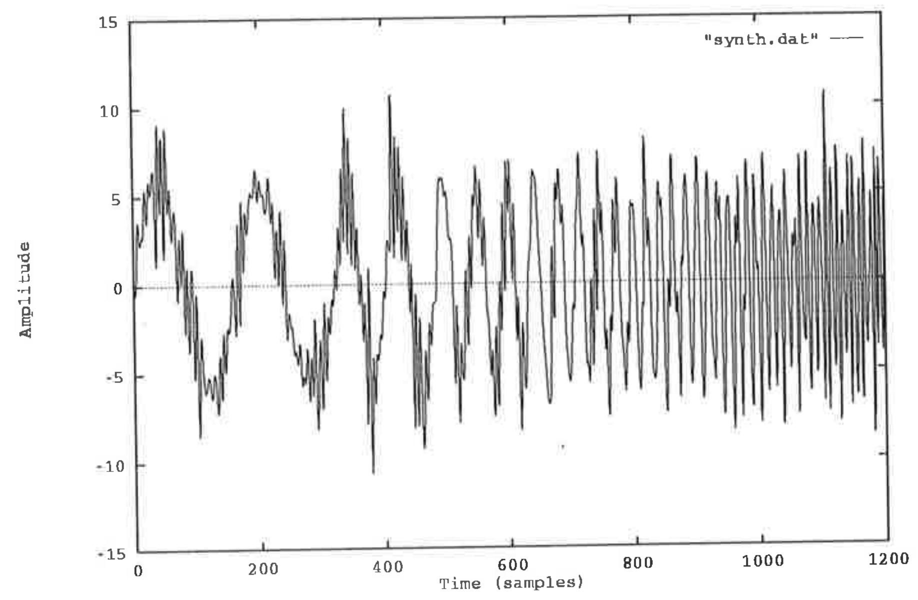


Figure 7.6: Reconstructed Waveform from Encoding Model; Residual Coding Rate 1/3.

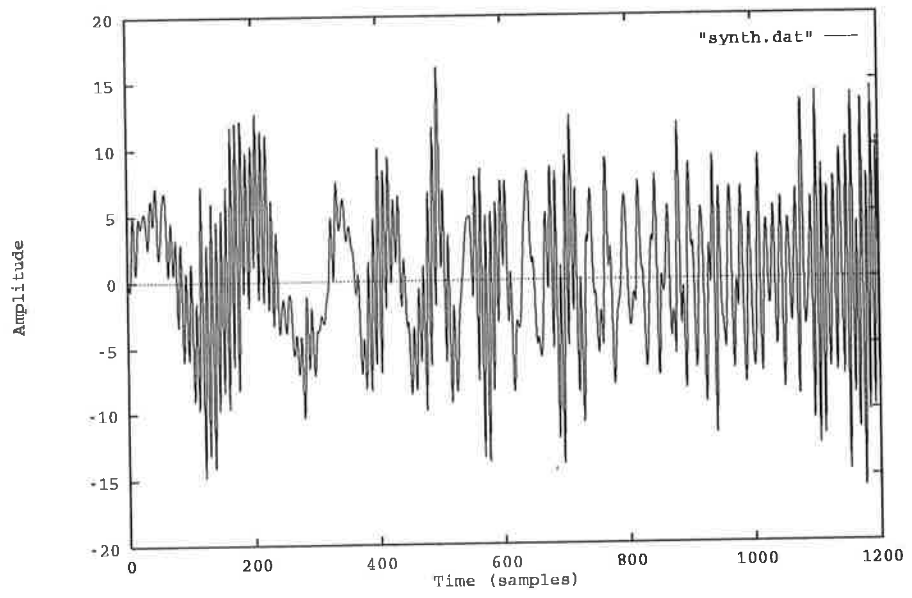


Figure 7.7: Reconstructed Waveform from Encoding Model; Residual Coding Rate 1/4.

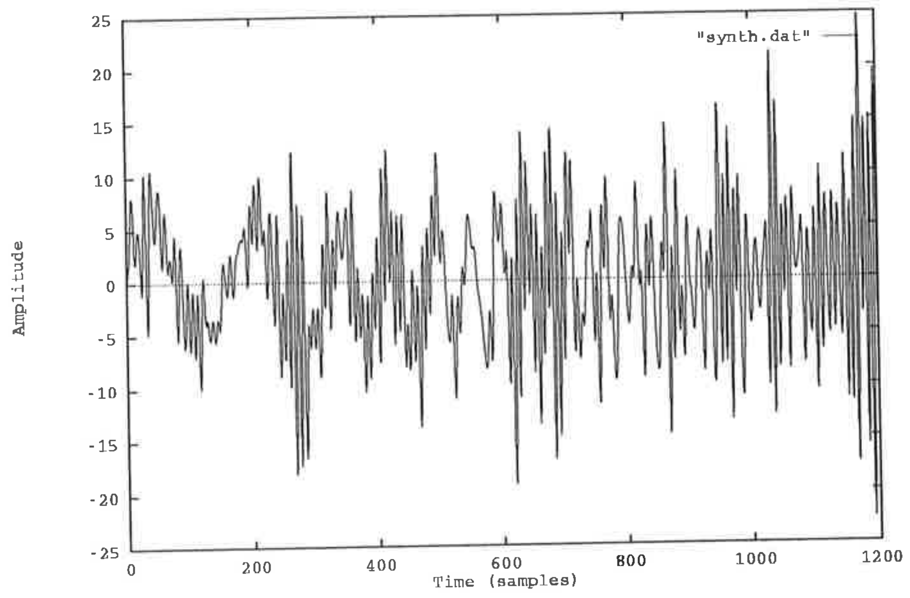


Figure 7.8: Encoding Rate 1/4 With a Greater Codebook-Range/Source Mismatch

that maximum possible pulse amplitude that could be fed into the synthesis filter. Figures 7.4, 7.5, 7.6, 7.7 had the range of the random quantisation levels set from -8 to $+8$ and the input waveform had an r.m.s. value of 4. Figure 7.8 illustrates the effect of setting the quantisation range from -12 to $+12$, causing a greater mismatch between the source and codebook.

Figures 7.9 and 7.10 illustrate the effect of the synthesis filter frequency response having a slower decay rate. Generally the encoding was better behaved but at the rate of 1/4 instability was again quite evident.

7.5 Conclusions

The four main factors that affected the output quality of encoding according to the encoding simulation were:

1. the actual encoding rate, (the number of branch symbols representing the number of residual samples);
2. the filter shape, where the greater the fall off rate the greater the distortion;

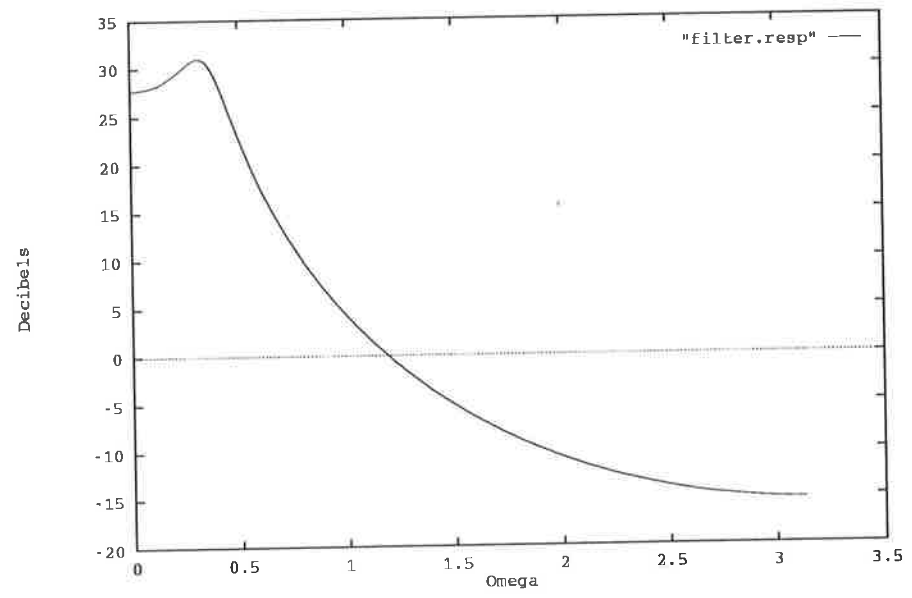


Figure 7.9: Frequency Response for Synthesis Filter 2

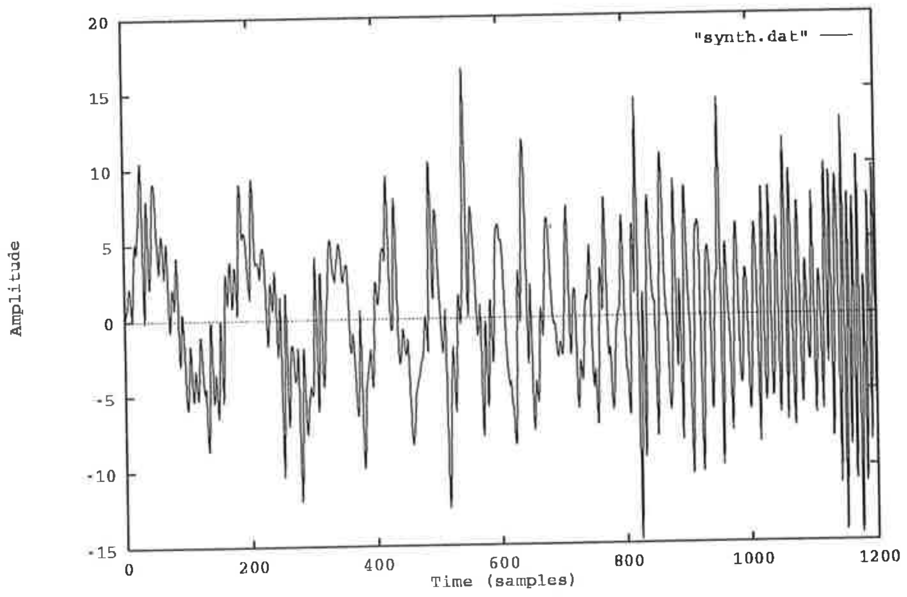


Figure 7.10: Encoding Simulation For Rate 1/4 with Filter 2

3. the number of quantisation levels, (corresponds roughly to the number of paths in the (M,L) algorithm);
4. the range of the quantisation levels with respect to the residual levels of the input waveform.

When the model was compared with the actual encoder the following factors were evident from informal listening tests:

1. as the rate for the encoder synthesis filter decreased, the reconstructed speech quality monotonically decreased,
2. the filter shape had a significant bearing on the output quality of the reconstructed speech,
3. encoding with a smaller number of paths did not have a significant impact on encoding performance around rate 1/2 (the larger codebooks/trellises had a slightly perceptible smoother sound),
4. mismatch in magnitude of the trellis entries relative to the input waveform residuals had a significant effect.

Mismatch in input sample magnitude range and trellis excitation range had a very pronounced effect. Trials were performed where the trellis codebook was trained for the input waveform having an r.m.s. of 4, and the actual input waveform scaled to different values. It was found that the best output was obtained when the input was correctly scaled, exhibiting least amount of audible distortion.

Open loop determination of the gains, made the gain estimation somewhat inaccurate. Modest scaling of the input waveform to r.m.s. values greater than nominal unity enabled the trellis to have shape and limited gain information. This made the system less sensitive to the open loop gain determination due to the larger dynamic range that the trellis excitation could encode.

Chapter 8

Conclusions

This thesis has dealt with making CELP type coders more robust with respect to channel errors. Experiments have shown that for CELP coders, the spectral information is most sensitive to channel errors followed by the gain for the stochastic code book followed by the other parameters. In particular trellis encoding of the excitation was undertaken in some detail.

A significant amount of effort was expended on finding a good pitch estimation algorithm as it was deemed important in the development of the coder. The post processing of speech discussed in the thesis used the pitch estimator but the coding algorithm did not due the instabilities of the encoding mechanism at low bit rates.

8.1 Pitch Estimation

Pitch synchronization can be very useful in a range of error recovery situations of encoded speech. Natural speech has a high redundancy and as pitch is one aspect, it is important that it is well understood. Pitch was taken to be the time, in samples, between glottal pulses. This corresponded to the quasi-periodicity of the speech waveform.

Completely robust and accurate pitch estimation is still an unsolved problem. The thesis presented a new method of obtaining pitch using a model based on dynamical systems theory. The methods used were based on a multidimensional phase space model which are usually used for determining the characteristics of chaotic systems. These methods however allowed determination of the pitch to super-resolution. Some concern must be expressed about the concept of super-resolution pitch estimation. Medan [76] claims super-resolution to arbitrary degree, however it should be noted that the number of samples within a quasi-period in the speech waveform, may change from one period to the next. This means that two pitch waveforms of different lengths are cross correlated. There are also situations where the pitch waveform change substantially from one pitch to the next. These observations of real speech call into question the accuracy of the claim of arbitrarily high resolution.

The final version of the pitch estimator developed in the thesis used a cross correlation matching technique not that different from the one developed by Medan, albeit the one described in the thesis used matching in three dimensions. There did not appear to be any great advantage of the multidimensional version and the computational effort was greater than Medan's. One advantage was that the search region for the pitch was reduced by employing the multidimensional AMDF in the first stage.

The model in the thesis used a simple rule base and a median filter to track the pitch and produce pitch contours. Tracking of the pitch at the receiving end was considered to be a possible useful method of error control for the pitch predictor parameters of the final coder. Papers in the literature [2] [103] have indicated that a pitch predictor structure in the CELP coder had to be based on fractional sample pitch estimates or multiple tap filters in order for the performance to match self excited codebooks. This was in line with the goal of obtaining a good fractional sample pitch estimate.

Packet waveform substitution was investigated for the memoryless coder case but not pursued with the CELP coder, as a previous frame's parameter substitution performed quite well; the subsequent correctly received frames interpolated well with the substituted frames. The extra processing required for waveform substitution instead of parameter substitution was not warranted for the CELP coder.

Pitch information was used in Chapter 6 to investigate post processing of the reconstructed speech waveform using pitch synchronicity. Pitch synchronicity meant that pitch events were tracked in the reconstructed speech so that the post processing of the speech was pitch synchronous and phase aligned with respect to the reconstructed speech waveform. It was discovered that some of the hoarseness of the speech could be traded off for a melodic although slightly synthetic sound. There was an overall improvement in the auditory quality in the sense that the speech was more pleasant to listen to.

8.2 Protection of the Short Term Spectral Information

Experiments [20] [51] have indicated that the spectral envelope information is very sensitive to channel errors and as such must be protected, or some other mechanism utilised to ensure that no gross spectral distortion occurred as a result of the channel errors. Bad errors are heard as *bangs, pops and clicks* which are very distracting and annoying to listeners.

Chapters 4 and 5 discussed in some detail the advantages of representing the the linear predictor coefficients in terms of Line Spectral Pairs (LSP's). Various useful properties of LSP's were discussed which made it clear why they are in common use in the low rate speech coding community. The properties of LSP's that are exploited in common CELP coders are:

1. they are the most efficient representation for quantisation (least number of bits required for quantisation),
2. the LSP values are monotonic, so out of order values indicate an error.

Error recovery techniques implicitly use the property that the effect of a perturbation of an LSP value is localised to a small range of frequencies in the reconstructed speech. This means that a small variation of an LSP value will only distort the spectral envelope in a small region associated with that particular LSP. Perturbation of the LP coefficients or reflection coefficients show distortion over the entire spectral envelope.

The two most commonly used error recovery strategies used in CELP coders at this time are:

1. repeat the spectral information from the last correctly received encoded speech frame,
2. sort the decoded LSP values so that they form a monotonic sequence.

Investigations undertaken in the thesis indicated that using the monotonicity criterion only captures a modest portion of errors (only those that cause an out of order sequence, approximately 1/3 to 1/2 of errors) and to improve performance beyond the base-line missed error distortion, additional measures need to be taken, such as extra error detection. This aspect of the performance of LSP monotonicity as an error detection mechanism is not well documented in the literature.

Two methods of error protection were investigated which had minimal impact on the bit rate for the speech frame. The goal was to produce a mechanism that was more robust to channel errors than the existing schemes in use.

8.2.1 Vector Quantiser as an Error Correction Mechanism

Error correction should always be taken in the context that it operates in. In speech coding spectral information is quantised and encoded from real values, and error correction in this case should be in the context of attempting to recover the spectral envelope with minimal error.

Generally vector quantisers are used to *encode* data for transmission over a channel. A novel proposal introduced in this work was to use a vector quantiser at the receiving end, where the vector codebook was either developed online or previously calculated. The monotonicity of the LSP coefficients were checked for errors and approximate error positions located. A pattern matching process was used to estimate the hypothesised erroneous data. The best match over the whole vector, excluding the hypothesised error, was selected as the most likely candidate vector and the component at the hypothesised error location then substituted. Generally two error location hypotheses and thus two rounds of matching were used.

From the experiments conducted in Chapter 4 a vector quantiser can be successfully applied in the process of missing data estimation. It was noted that not only the missing data was estimated but the whole vector was approximated; although only the missing element was used, provided monotonicity was not violated. Maintaining monotonicity ensured that the LSP parameters would result in a minimum phase synthesis filter for the reconstruction of speech with minimal spectral distortion. If no useful substitution could be made for the missing data, the closest matched vector from the vector codebook was used in unmodified form to ensure a stable reconstruction filter.

It was quite clear that a saturation effect in the performance gains occurred with larger numbers of vectors in the quantiser codebook. This may well have been due to the effect of just taking the minimum distance centroid from the hyperplane without regard to the size of cell that the centroid was taken from. If the volume (generated by the hyperplane cutting the voronoi cell and the cell's centroid) was taken into account, a more accurate estimate could have been made. Taking only the normal distance from the cutting hyperplane to the centroid, resulted on average with a good estimate, but as the number of cells cut by the hyperplane increased the estimate procedure may have degraded a little. The results however showed the method performed significantly better than sorting the LSP parameters.

A simple adaptive vector quantiser was seen to be an effective step in optimizing the performance of the technique. The training of the adaptive codebook added only a little extra computation, namely storage of a running sum of vector components, a running estimate of encoding distortions, a population count for each cell, and the recalculation of the resulting centroid. The major part of the computation was in performing mean square error matching of the received vector and the codebook vectors. Training for the online codebook was suboptimal, and further investigations yielded a better on-line training mechanism. The improved method tried optimizing training of the codebook by taking into account the statistics of the vectors represented by a cell. If two cells moved too closely together during operation, they were amalgamated and a new cell became free for training. This improvement helped in speeding up the adaption process. Some of the vector quantisers currently being constructed by the neural network community show useful characteristics. See for example reference [116].

It was observed that the 96 entry adaptive codebook performed slightly better than the 128 entry fixed codebook for the test data-base. Generally the 128 entry fixed codebook may not perform as well as indicated over a larger variety of speakers, while the adaptive codebook has an advantage in that it always trains to the particular speaker. It should also be noted that if a more sophisticated error detection scheme was used the adaptive method should improve as the number of undetected erroneous vectors used in training would decrease.

A brief quantitative discussion on using cyclic codes indicated that using only 5 parity bits would catch most error bursts, and allow better performance and training of the adaptive codebook. The cost of using only error detection without obtaining information about error location is more rounds of matching in the vector quantiser to estimate the error.

8.2.2 Protection of Vector Quantised Parameters

Quantisation at the source must always be undertaken relative to some performance criterion. Encoding of the spectral information can usefully employ the concept of just noticeable distortions as a measure of performance. If the spectral distortion between the encoded version of the spectral envelope and the original envelope is less than 1% the term transparent coding is used. Vector quantisation could achieve this level of performance with approximately 24 bits, however the memory usage of the codebook and the computational load for encoding the vectors would be enormous. To counter these two problems different architectures for vector quantisers were investigated: for the former problem split vector quantisers and multistage quantisation, and tree structures for the latter problem.

Tree structured vector quantisers dramatically reduce the computational burden of encoding. Implementation of a training algorithm which constructed a tree structure that gave minimal distortion compared to random full searched codebooks was addressed first. Experimental results indicated that an unbalanced tree gave good performance. It was also found that the tree structure sped up the training procedure relative to the full search codebook design. This occurred because once a decision region was chosen the training vectors were also partitioned and could no longer influence training of vectors outside of their partition. Once the tree encoding procedures were developed, the tree structure was imposed on the higher level architectural configurations of, split vector quantisation and multistage vector quantisation. Tree structured vector quantisers address the problem of the computational burden while the higher level architectural designs, split vector and multistage quantisation, address computational effort and the codebook size problem. Tree structures applied to the higher level structures is rarely done and has not been reported in the literature with respect to coding speech line spectral pairs.

Tree structures are well known in the literature where the general training procedure uses the generalised Lloyd algorithm and a splitting technique which involves perturbing the centroid of a cell by $\pm\delta$ to produce two new candidate centroids. To speed up the convergence of the generalised Lloyd algorithm within the tree structured quantiser, a principal component analysis of the mean square error variance data for the quantisation cells was undertaken.

The idea of using principal component analysis for the splitting process in the design of tree structured quantisers, although not producing substantially better codebooks, did give some objective measure of the performance of the tree structured codebooks over full search codebooks. If most of the quantiser cells had a unique principal eigenvector-vector for the cell's covariance matrix, the tree quantiser tended to perform well. This was best seen in constructing the *split* vector quantiser where the codebook for the upper sub-vector tended to be *white* and only a small number of cells exhibited a strong principal component. The performance of the tree structured codebook for the upper sub-vectors proved to be poor. Whenever most cells in the vector quantiser showed a strong single principal component, average performance of the codebook was good. The most likely explanation for this phenomenon is that the tree construction can only split a single cell into two sections. The direction of maximum variance is the most likely direction to split the cell such that encoding of the training set results in less average distortion than that obtained prior to splitting. If the underlying data vectors are white noise-like, no preferred direction for splitting is evident as there are no strong correlations in any one particular direction. As the vector quantiser in this case cannot make use of under-lying data correlations, its performance is degraded compared to data that is correlated.

A comparison of tree structured *multistage* vector quantisers and tree structured *split* vector quantisers indicated that the *multistage* option had a performance advantage of up to 5 dB. This was verified for the codebook sizes possible for the available speech database of 3400 ten dimensional LSF vectors. Further improvement may be obtained by using a scaling for the second quantiser in the *multistage* quantiser at the expense of more bits used in encoding the source. The tree structured codebook also appeared to have a greater sensitivity to relation-ships between vector components (biassing against the split vector design) than full searched codebooks, however a very large data set would be required to verify this.

The overall cost of transmitting the line spectral pairs as indices of a codebook over a noisy (error) channel was considered. If the multistage quantiser was chosen a conservative estimate using extrapolation and adding in allowances (for the small training set) a *coarse* quantiser of 14 bits ($\approx$ 16000 entries) and an *error* quantiser of 10 bits ($\approx$ 1000 entries) would in all likelihood meet the transparent coding specification. (Paliwal and Atal suggest 24 bits for their quantiser.) If simple error detection was used on the *error* quantiser index an additional 3 bits would be added for cyclic redundancy. The cyclic redundancy code would use the generator polynomial $x^3 + x + 1$. This code is capable of detecting any 2 random errors and any burst of contiguous error bits up to 3. If the channel bit error rate was $\epsilon = 0.01$ the probability of undetected error would be 2×10^{-4}. Of the 14 bits in the coarse quantiser index, 9 or 10 would probably need to be protected. To protect these bits well it would probably be necessary to use a rate 1/2 code with soft decision decoding, thus an extra 9 or 10 redundant bits would be required. The number of bits to transmit the line spectral pairs representation reliably over a noisy channel now totals 36 or 37 bits. It may be possible to ignore the error introduced by the *error* quantiser and only check for monotonicity at the receiver, as the errors are only a fine adjustment. The level of spectral distortion caused by an incorrect *error* quantiser index needs to be investigated.

An advantage of using vector quantised indices and forward error correction on the most significant bits of the codebook index, is that some of the error correcting power of the code could be used for error detection. In the mobile communications environment deep fades, consequently large error bursts are common place. The detection of a burst allows other higher level error correction capabilities such as packet substitution as described in [51], [21] to be applied. The substitution of previously *clean* packets for packets that have been destroyed by an error burst yields acceptable results.

8.2.3 Comparison of Techniques

To get transparent coding[1] scalar quantisation of line spectral pairs requires of the order of 32 to 34 bits. Any competing technique to the current well established procedures must provide a performance advantage with regard to channel errors, preferably with less bits.

The first scheme discussed had the advantage that it could be used with existing coders for improved performance. To significantly improve performance beyond this, more errors need to be detected which could be done by adding extra parity bits. This feature could be added to existing coders by extending the data frame. Most of the processing is performed at the receiving end at the cost of extra processing there.

The second scheme required more processing at the transmitting end in performing the encoding. It was shown that permuting of indices relative to the vectors that they represent had some performance gain that diminished as the codebook became larger. If permutation of indices was used in conjunction with error protection of some of the index bits better performance was achieved. If the multistage encoding technique was used in conjunction with the above techniques a viable alternative to scalar encoding is possible.

[1]Transparent coding requires the spectral distortion to be less than 1%.

Both of the suggested techniques used 37–39 bits for encoding to produce a performance superior to current spectral encoding methods. The overall bit rate increase in bit rate would be of the order 120-200 bits per second, assuming 25 ms speech frames.

8.3 Trellis Encoding of the Residual

Trellis encoding of the speech residual was undertaken in an attempt to encode speech at low bit rates. The (M,L) algorithm was singled out as a possible candidate for real time implementation of trellis coding as the encoding effort is dependent on the chosen number of paths and not the number of states in the trellis.

The implementation of the trellis encoding mechanism was non-trivial and posed more implementation and analytical problems than the relatively straight forward block mechanism. Each node on a path in the trellis encoding had to record the latest synthesis filter memory at that node. This process also allowed a smooth transition between blocks as the contents of the synthesis filter memory was correctly modelled (for the decoder) at all stages of encoding.

Informal listening tests indicated that a regular pulse excitation mechanism (at rate 1/2 this consisted of a random amplitude pulse followed by a 0) from the trellis performed similarly to fully populated vectors (for rate 1/2 this consisted of two random amplitude pulses) for each branch label. This reduced the storage for the regular pulse excitation trellis to 1/2 that of the equivalent fully populated rate 1/2 trellis.

In comparing the curves of residual encoding error for different numbers of paths M in the (M,L) algorithm, a substantial knee was found at $M = N/2$, (where N is the number of states in the trellis). This implied that setting the number of paths to 1/2 the number of states would result in the best effort versus performance trade-off. In practice informal listening tests indicated that it was not necessary to use this number of paths as the auditory performance improvement was slower than the excitation error curve.

8.3.1 Training of the Trellis

Training of the trellis presented a substantial challenge. If the (M,L) algorithm was used for encoding, the growth of the accumulated metrics was a non-linear function of branch values (where branch values are the control variables). Furthermore, if the branch values were associated with a regular pulse excitation mechanism, methods such as trellis training using the LBG [44] algorithm was impractical. The major difficulty was that a *synthesised* waveform had to be compared to the reference training waveform. The residual waveform could not be used to train a regular pulse excitation codebook as it could for full vector branch entries.

Due to the non-linearity of the (M,L) algorithm a gradient descent method of training for the codebook was unsuitable. The function profile did not show even approximate convexity. These characteristics suggested the use of random search based optimization functions such as simulated annealing and genetic algorithms. Although these algorithms did optimize the codebooks they did so extremely slowly even for the regular pulse excitation which halved the number of parameters compared to the rate 1/2 full excitation trellis. A further assumption of trellis symmetry was made to reduce the computational burden (number of independent variables) by a further factor of 2 and also aid the rate of convergence (suggested by Freeman [45]). This was based on the observation that the speech samples tended to be fairly symmetrically distributed about the zero amplitude axis.

Even with the reduction of the problem size, the simulated annealing and genetic algorithms proved to be too slow to be useful. A further reduction of effort by reducing the (M,L) algorithm's number of paths, M, to 8 to speed up encoding made the methods suitable only for small trellises.

The development of an optimization procedure based on conjugate direction random line optimization[2] proved to provide results as good as the above two optimization methods in two orders of magnitude less time.

The new algorithm allowed training of trellises of 128 states in approximately 140 hours of CPU time on a SUN Sparc Station 2. It should be noted that the algorithm allows an efficient implementation on a multi-processor computer of the multiple instruction, multiple data path type.

[2]Described in detail in Chapter 6

8.3.2 Trellis Encoding Performance

General results showed that the greater the number of paths the better the resulting synthetic speech. Also the greater the number of states of the encoding trellis the better the reconstructed speech quality; the 64 state trellis and 128 state trellis encodings were noticeably better than encodings from the 32 state trellis for fixed number of paths. Training of the trellises gave substantial improvement in the perceived quality of the speech. Tests also indicated that the lowest bit rate feasible for this type of encoder was 1/2 bit per sample. Lower bit rates produced very distorted speech.

8.4 Error Modelling of Trellis Encoded Residual

A new method of modelling of the error processes that occur in trellis encoding of the excitation proved a useful tool in determining the importance of different parameters.

Modelling of the effects of channel errors on the decoder gave a feel for the types of distortion and the extent of the distortions caused by bit errors. The sample speech files used in the thesis consisted of approximately 45% voiced speech. If the 128 state trellis was used for encoding of the speech with a pulse magnitude threshold setting of about 15 (considered a distracting level of error) the probability of error is read off the cumulative error curves as approximately 0.26. The product of the two probabilities gives approximately a probability of 12% that an error on the channel will cause a significant disturbance in the synthetic speech.

Listening tests confirmed that the larger codebooks produced better speech quality in the presence of noise but more in the quality of the noise rather than the perceived level. The larger codebooks gave a more mellow noise than the smaller codebooks for the same channel error rate.

Determining the dynamics of encoding from the simplified encoder model gave an insight as to the parameters that are important in ensuring that the encoder remains stable in service. The main factors found important were:

- the code rate,
- the synthesis filters frequency response,
- gain mismatch between input waveform and quantisation range,

- and the number of quantisation levels.

It is recommended that the trellis encoding mechanism should not be used below rate 1/2 as the quantisation becomes coarse and noisy causing instability. Filter frequency response is out of the designer's hands as the input speech waveform determines this. Gain matching is important but the deleterious effects of a bad mismatch may be somewhat overcome by training the trellis entries to encode over a larger gain variation than the traditional CELP. It should be noted that the traditional CELP algorithms adjust the codebook gains as part of the matching process and as such the input speech does not have to be scaled prior to encoding. Empirically it was found that for the trellis coder the reconstructed speech quality was best when the trellis incorporated an r.m.s gain range of 4 as well as the shape information.

8.5 Trellis Coder Overview

Some useful methods for improving the robustness of the spectral information over noisy channels were developed with the choice of extra processing at the receive end or at the transmit end using essentially the same bit rates. These techniques are suitable for any vocoder that encodes the short term spectral information in terms of linear predictor coefficients or an equivalent representation of them.

Post processing of the received speech to reduce the hoarseness, due to the distortion caused by encoding at low rates, was successful in the sense of making it more pleasant to listen to. Transmission of the extra pitch parameters, an extra 280 bits per second, may be questioned. However in the authors opinion the improvement was worth the cost of the bits.

The resultant speech encoder (using trellis residual encoding) developed in this thesis is best compared with the original multipulse coder designs which were the fore-runners to the modern CELP designs. These earlier versions did not have a long term predictor and tried, as does the present design, to emulate the true residual with a few pulses using analysis by synthesis techniques. Araseki [13] described a multipulse coder similar to the coder developed in this thesis, the simplified version of which is shown in Figure 8.1.

Comparison of the coder developed here to the earlier designs indicated that the trellis encoded residual encoder performed well. Speaker intelligibility was very good and rec-

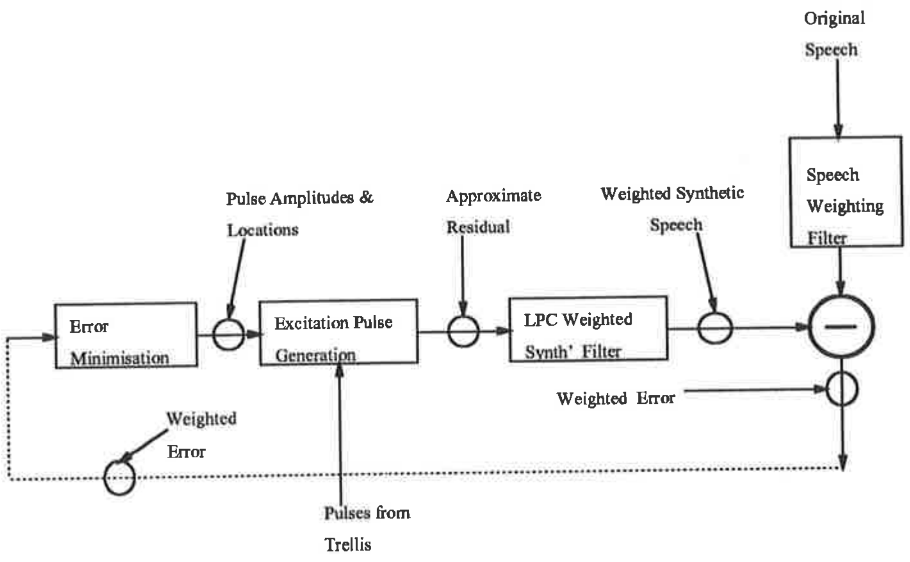


Figure 8.1: Simplified Block Diagram of Proposed Speech Encoder.

ognizability was good although it displayed a slight hoarseness. A segmental signal to noise ratio objective evaluation indicated that the developed coder had a performance of 5.3 dB at 5.5 kbps while the coder described by Araseki had a performance of 5 dB at 6.5 kbps. The original regular pulse excitation coder as described by Kroon [16] operated at the significantly higher bit rate of 9.6 kbps and achieved a commensurately higher segmental to noise ratio of 13 dB.

The final bit rate and bit allocation for the fully quantised coder, excluding forward error correction bits, was determined to be approximately 5.5 kbps as follows:

1. synthesis filter poles information — 32 bits;
2. scaling gain term — 5 bits;
3. speech residual information — 100 bits;
4. giving a Total — 5,480 bits per second.

Speech frames consisted of 200 samples at a sampling rate of 8 kilo-samples per second. This implied a frame duration of 25 ms. The gain term for the residual did not require a sign as the trellis encoding accounts for the sign.

It was found that the trellis excitation encoding method cannot be used for lower bit rates than those indicated above, although slightly lower rates may be possible if a long term predictor is included. This is a conjecture, bearing in mind that a long term predictor would make the input to the trellis coder (residual with periodic correlations substantially removed) more Gaussian like distribution with less amplitude fluctuation. This would assist indirectly in the gain matching problem.

A different tree source encoding procedure that used a combination of breadth first and metric first decoding could be useful in the predictive encoding environment where the assumption of being able to accurately make a branch selection on the basis of current values and past performance is invalid. In a predictive environment selection of an innovation pulse at time t may still have significant effect at time $t + 50$ in the future. If the rate of source coding for the predictive coder is high enough the innovation can effectively compensate for past incorrect choices and avoid instabilities.

Inclusion of a long term predictor would necessarily imply that the resulting residual must be encoded at rates lower than 1/4 bit per sample for similar or lower bit rates. Salami [117] has developed a block excitation coding method that displays good robustness over noisy channels. Investigations in this direction could prove more fruitful for lower bit rates.

References

[1] J.P. Campbell Jr., V.C. Welch, and T.E. Tremain, "An Expandable Error-Protected 4800 BPS CELP Coder. (U.S. Federal Standard 4800 BPS Voice Coder)", Proc. ICASSP 1989 pp. 735–738.

[2] P. Kroon and K. Swaminathan, "A High-Quality Real-Time CELP Coder", IEEE Journal on Selected Areas of Communications, Vol. 10, No. 5, pp. 850–857, June 1992.

[3] *Federal Standard 1016, National Communications Systems, Office of Technology and Standards.* General Services Administration, Office of Information Resources Management. USA.

[4] P. Kroon and E.F. Deprettere, "A Class of Analysis-by-Synthesis Predictive Coders for High Quality Speech at Rates Between 4.8 and 16 kbit/s", IEEE Journal on Selected Areas in Commun., Vol. 6, No. 2, pp. 353–363, Feb. 1988.

[5] R.C. Rose and T.P. Barnwell 3rd, "Quality Comparison of Low Complexity 4800 bps Self Excited and Code Excited Vocoders", ICASSP '87, pp. 1637 – 1640, 1987.

[6] W.B. Kleijn, D.J. Krasinski and R.H. Ketchum, "Improved Speech Quality and Efficient Vector Quantization in SELP", Proc. ICASSP '88, pp. 155 – 158, 1988.

[7] C.S. Xydeas, M.A. Ireton and D.K. Baghbadrani, "Theory and Real Time Implementation of a CELP Coder at 4.8 & 6.0 kbits/sec Using Ternary Code Excitation", IERE Conf. on Digital Processing of Signals in Communications, pp. 167–174, 1990.

[8] R. Salami, K.H.H. Wong, D. Appleby, and R. Steele, "A Robust Transformed Binary Vector Excited Coder with Embedded Error Correction Coding", IEE Colloquium, 9. Oct. 1989, Savoy Place London

[9] P. Kabal and R.P. Ramachandran, "Joint Optimization of Linear Predictors in Speech Coders", IEEE Trans. ASSP Vol. 37, No. 5 pp. 642–650, May 1989.

[10] W.B. Kleijn, P. Kroon, L. Cellario and D. Sereno, "A 5.85 kb/s Celp Algorithm for Cellular Applications", ICASSP '93, pp. II-596 – II-599, 1993.

[11] I.M. Trancoso and B.S. Atal, "Efficient Search Procedures for Selecting the Optimum Innovation in Stochastic Coders", Trans. ASSP. Vol. 38 No. 3 pp. 385 – 395, March 1990.

[12] B.S. Atal and J.R. Remde, "A New Model of LPC Excitation for Producing Natural-Sounding Speech at Low Bit Rate", Proc. 1982 ICASSP, pp. 614–617, 1982.

[13] T. Araseki, K. Ozawa, S. Ono and K. Ochiai, "Multi-Pulse Excited Speech Coder Based on Maximum Cross-Correlation Search Algorithm", pp. 23.3.1–23.3.5, 2nd IEEE Global Commun. Conf. 1983.

[14] A. Perkis and B. Ribbum, "A good Quality, Low Complexity 4.8 kbit/sec Stochastic Multipulse Coder", ICASSP '89, 1989.

[15] R.L. Zinser and S.R. Koch, "4800 & 7200 bit/sec. Hybrid Codebook Multipulse Coding", ICASSP '89, pp. 747 – 750, 1989.

[16] P. Kroon, Ed F. Deprettere and R.J. Sluyter, "Regular-Pulse Excitation – A Novel Approach to Effective and Efficient Multipulse Coding of Speech", IEEE Trans. ASSP, Vol. ASSP-34, No. 5. pp. 1054–1063, October 1986.

[17] R. Montagna and M. Omologo, "Some Results on Multipulse Linear Prediction Coding", Globecom '86 pp. 802 – 806, 1986.

[18] *13 kbps. Regular Pulse Excitation – Long Term Prediction – Linear Predictive Coder for use in the Pan European Digital Mobile Radio System*, GSM Recommendation 6.10. March 1988.

[19] J.R. Deller,Jr., J.G. Proakis and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals* pp. 273 – 280, Macmillan, 1993.

[20] R.V. Cox, W. Bastiaan Kleijn and P. Kroon, "Robust CELP Coders For Noisy Backgrounds and Noisy Channels", Proc. ICASSP 1989 pp. 739–742.

[21] A. Perkis, "Speech Coding For Mobile Satellite Communications; a novel scenario", ISSPA 90, Gold Coast, August 1990.

[22] K. Ozawa and T. Miyano, "4 kb/s Improved Celp Coder with Efficient Vector Quantization", ICASSP '91, pp. 213 – 216, 1991.

[23] P.G. Drazin *Nonlinear Systems*, Cambridge University Press, 1992.

[24] S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, Springer Verlag, 1990.

[25] N. Tishby, (1990) "A Dynamical Systems Approach to Speech Processing", S6b.5 ICASSP 90

[26] J.R. Deller,Jr., J.G. Proakis and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals* pp. 241 – 251, Macmillan, 1993.

[27] I.M. Trancoso and J.M. Tribolet, "Harmonic Post Processing of Speech Synthesised by Stochastic Coders", IEE Proc., Vol 136, Pt. I No. 2 pp. 141–144, April 1989.

[28] M. Kabrisky, S.K. Rogers and N.A. Bashir, "Reconstruction of Mutilated Speech", IEEE AES Magazine, pp. 39 – 43, Sept. 1989.

[29] B.S. Atal and M.R. Schroeder, "Optimizing Predictive Coders for Minimum Audible Noise", Proc. ICASSP 1979 pp. 453–455.

[30] F.K. Soong, and B.H. Juang, "Line Spectral Pairs (LSP) and Speech Data Compression", Proc. ICASSP, 1984.

[31] F. Itakura, "Line Spectrum Representation of Linear Predictive Coeffients of Speech Signals", J. Acoust. Soc. Am., vol. 57, pp S35(A), 1975.

[32] N. Sugamura and N. Farvardin, "Quantizer Design in LSP Speech Analysis-Synthesis", IEEE JSAC, vol. 6, pp. 432–440, Feb 1988.

[33] A. Gersho, & R.M. Gray *Vector Quantization* Kluwer Press 1992

[34] P. Kroon and B.S. Atal, "Strategies for Improving the Performance of CELP coders at Low Bit Rates", ICASSP '88, 1988.

[35] J. Grass and P. Kabal, "Methods of Improving Vector – Scalar Quantization of LPC Coefficients", ICASSP '91, pp. 657 – 660, 1991.

[36] K.K. Paliwal and B.S. Atal, "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame", Proc. ICASSP., pp. 661–664, 1991.

[37] R.A. Salami, K.H.H. Wong, R. Steele and D.G. Appleby., "Performance of Error Protected Binary Pulse Excitation Coders at 11.4 kbps Over Mobile Radio Channels", Proc. ICASSP'90, Albuquerque, USA, pp. 473–476 April, 1990

[38] E. Ayanoglu and R. Gray., “The Design of Joint Source and Channel Trellis Waveform Coders”, IEEE Trans. Inform. Theory, vol IT-33, pp. 855–865.

[39] N. Farvardin and V. Vaishampayan, “On the Performance and Complexity of Channel-Optimized Vector Quantizers” IEEE Trans. Inform. Theory, vol. IT-37, pp. 155–160 Jan. 1991

[40] N. Farvardin, “A Study of Vector Quantization for Noisy Channels”, IEEE Trans. Inform. Theory, vol. IT-36, pp. 799–809, July 1990.

[41] P. Secker and A. Perkis, “A Robust Speech Coder Incorporating Joint Source and Channel Coding Techniques”, Proc. SST’90 , pp. 46–51, Melbourne, 1990.

[42] K. Zeger and A. Gersho, “Pseudo-Gray Coding”, IEEE Trans. Comms, vol. 38, pp.2147–2158, Dec. 1990

[43] R.A. Salami, “Binary Code Excited Linear Prediction (BCELP): new approach to CELP coding of speech without codebooks.”, Electronics Letters, vol. 25 No. 6, pp. 401–403, 16th March 1989.

[44] L.C. Stewart, R.M. Gray, and Y. Linde, “The Design of Trellis Waveform Coders”, IEEE Trans. Comms., Vol. Com-30, No.4, pp. 702–710 Apr. 1982.

[45] G.H. Freeman, I.F. Blake and J.W. Mark. “Trellis Source Code Design as an Optimization Problem”, IEEE trans. Inform. Theory, Vol 34 No. 55 pp. 1226–1241 Sept 1988.

[46] C.R. Nasser and M.R. Soleymani, “Globally Optimal Trellis Quantizers”, ICASSP ’91 pp. 41 – 44, 1991.

[47] K. Rose, E. Gurewitz and G.C. Fox, “Vector Quantization and Deterministic Annealing”, IEEE Trans. Inform. Theory, Vol. 38, No. 4, pp. 1249 – 1257, July 1992.

[48] N-A Lu and D.R. Morrell, “Vector Quantizer Codebook Design Using Improved Simulated Annealing Algorithms”, ICASSP ’91, pp. 673 – 676, 1991.

[49] J.B. Anderson, “Limited Search Trellis Decoding of Convolutional Codes”, IEEE Trans. IT Vol. 35, No. 5, pp. 944–955 Sept. 1989.

[50] J.A. Asenstorfer, “Super Resolution Pitch Estimator Using Chaos Theory”, Proc. SST 1990.

[51] D. Rowe, A. Perkis, W.G. Cowley, J.A. Asenstorfer, "Error Masking in a Real Time Voice Codec For Mobile Satellite Communications", Proc. Speech Science and Technology (SST) 1990.

[52] J. Makhoul, "Linear Prediction: a Tutorial Review", Proc. IEEE, Vol. 63, No. 4, pp. 561–580, April 1975.

[53] S. Saito and K. Nakata, *Fundamentals of Speech Signal Processing* Academic Press 1985, Chapt. 4

[54] R. Linggard, *Electronic Synthesis of Speech*, Cambridge University Press, pp. 29–37, 1985.

[55] L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, pp. 135–161, 1978.

[56] S.P. Lloyd, "Least Squares Quantization of PCM", IEEE Trans. Inform. Theory, Vol. IT-28, No. 2, March 1982.

[57] J. Max, "Quantizing for Minimum Distortion", IRE Trans. Inform. Theory, Vol. IT-6, No. 1, pp. 7–12, March 1960.

[58] D.J. Goodman, G.B. Lockhart, O.J. Wasem, W. Wong, "Waveform Substitution Techniques for Recovering Missing Speech Segments in Packet Voice Communications", IEEE Trans. Accoustics, Speech and Signal Processing, Vol. ASSP-34, No. 6, pp. 1440–1447, December 1986

[59] I. Percival & D. Richards, *Introduction to Dynamics*, Cambridge University Press. 1989

[60] F. Argoul, et al "Chemical Chaos: From Hints to Confirmation", Accounts Of Chem. Research, Vol 20, Dec. 1987.

[61] P. Chen, "Empirical and Theoretical Evidence of Economic Chaos", Systems Dynamics Review Vol. 4 Numbers 1–2, 1988.

[62] J. Feder, *Fractals*, pp67-73, Plenum Press), 1988.

[63] P. Grassberger, & I. Procaccia "Measuring the Strangeness of Strange Attractors", Physica D 9 pp. 189–192, 1983.

[64] L. Condie,"Non-Linearity in Vowel Waveforms", Proc. SST 90, Melbourne 1990.

[65] L. Condie, "Phase Space Behaviour of Speech", Proc. SST 92, Brisbane 1992.

[66] C. Grebogi, E. Ott & J.A. Yorke, "Chaos, Strange Attractors, and Fractal Basin Boundaries in Nonlinear Dynamics", Science, Vol 238, Oct. 1987.

[67] R.M. May, "Simple Mathematical Models With Very Complicated Dynamics", Nature Vol. 261 June 10, 1976.

[68] J.P. Eckmann, "Roads to Turbulence in Dissipative Dynamical Systems" Reviews of Modern Physics Vol. 53, No. 4 Part I, Oct 1981

[69] H. Whitney, "Differentiable Manifolds" Ann. Math. No. 37. pp. 645–680, 1936.

[70] F. Takens, *Dynamical Systems and Turbulence, Warwick, 1980* , Vol 898 of *Lecture Notes in Mathematics*, edited by D.A.Rand and L.S. Young (Springer, Berlin, 1981) p. 366

[71] M.A. Fraser, & H.L. Swinney, "Independent Coordinates for Strange Attractors from Mutual Information" Phys. Rev. A. Vol. 33. Feb 1986

[72] J-P. Eckmann and D. Ruelle, "Ergodic Theory of Chaos and Strange Attractors", Reviews of Modern Physics, Vol. 57, No. 3, Part 1, pp. 617–656,

[73] E. Ott, "Strange Attractors and Chaotic Motions of Dynamical Systems" Reviews of Modern Physics, Vol. 53, No. 4, Part I, Oct 1981

[74] N. Kitawaki and H. Nagabuchi, "Quality Assessment of Speech Coding and Speech Synthesis", IEEE Communications Magazine, pp. 36–44, Oct. 1988.

[75] M. Schroeder, *Fractals, Chaos, Power Laws*, pp167–174, Freeman, 1991.

[76] Y. Medan, E. Yair, & D. Chazan, "Super Resolution Pitch Determination of Speech Signals", IEEE Trans. on Signal Proc. Vol. 39, No. 1. Jan. 1991.

[77] W.J. Krzanowski,*Principles of Multivariate Analysis: A User's Perspective*, Oxford University Press, pp. 316-318, 1988.

[78] Y. Linde, A. Buzo, & R.M. Gray, "An Algorithm for Vector Quantizer Design", IEEE Trans. Commun. Technol., vol. COM-28, pp. 84–95, Jan. 1980.

[79] J. Makhoul, S. Roucos, & H. Gish, "Vector Quantization in Speech Coding", Proc. IEEE, Vol. 73, pp. 1551–1588, 1985.

[80] A. Gersho, "On the Structure of Vector Quantizers" IEEE Trans. Information Theory, Vol. IT-28, No.2, March 1982

[81] A. Gersho, "Asymptotically Optimal Block Quantization" IEEE Trans. Information Theory, Vol. IT-25, No. 4, July 1979

[82] T.D. Lookabaugh, & R.M. Gray "High-Resolution Quantization Theory and Vector Quantizer Advantage" IEEE Trans. Information Theory, Vol. 35, No. 5, Sept. 1989

[83] J.H. Conway, & N.J.A. Sloane, "Voronoi Regions of Lattices, Second Moments of Polytopes, and Quantization", IEEE Trans. Information Theory, Vol. IT-28, No. 2, March 1982

[84] A. Perkis, & B. Ribbum, "Application of Stochastic Coding Schemes in Satellite Communication" in, *Advances in Speech Coding* Kluwer Academic, 1990.

[85] W.H. Equitz, "A New Vector Quantization Clustering Algorithm", IEEE Trans. Acoustics, Speech and Signal Processing, Vol. 37, No. 10, pp. 1568–1575. Oct. 1989.

[86] Shu Lin and D.J. Costello Jr., *Error Control Coding Fundamentals and Applications*, Prentice-Hall 1983.

[87] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, Vol. 1. North-Holland 1977.

[88] G.S. Kang and L.J. Fransen, "Application of Line-Spectrum Pairs to Low-Bit-Rate Speech Encoders", Proc. ICASSP. pp. 244–247, 1985.

[89] B.S. Atal, R.V. Cox and P. Kroon, "Spectral Quantization and Interpolation for CELP Coders", Proc. ICASSP. pp. 69–72, 1989

[90] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing", Science, vol. 21, pp. 671–680, May 1983.

[91] D.E. Goldberg, *Genetic Algorithms: in search, optimization & machine learning,* Addison-Wesely 1989.

[92] S. Saito and K. Nakata, *Fundamentals of Speech Signal Processing* Academic Press 1985 pp. 120–124.

[93] P. Kabal and R.P. Ramachandran, "The Computation of Line Spectral Frequencies using Chebyshev Polynomials", IEEE Trans. ASSP vol. 34, pp. 1419–1425, Dec 1986.

[94] H.G. Eggleston, *Convexity*, Cambridge University Press 1969. p 29.

[95] A. Householder,*The Theory of Matrices in Numerical Analysis*, Dover, 1988.

[96] D.E. Goldberg and R. Lingle, "Alleles, Loci, and the Travelling Salesman Problem", Proceedings of an International Conference on Genetic Algorithms and their Applications, pp. 154–159, 1985.

[97] D.E. Goldberg, *Genetic Algorithms: in search, optimization & machine learning*, pp. 123–124, Addison-Wesely 1989.

[98] G.C. Clark and J. Bibb Cain,*Error Correcting Coding for Digital Communications*, Chapt. 4, Plenum Press 1981.

[99] D.H. Lee, D.L. Neuhoff and K.K. Paliwal, "Cell-Conditioned Multistage Vector Quantization", Proc. ICASSP., pp. 653–656, 1991.

[100] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression* pp. 451–459, Kluwer Academic Publishers, 1992

[101] M.W. Marcellin and T.R. Fischer, "Trellis Coded Quantization of Memoryless and Gauss-Markov Sources", IEEE Trans. Comms. Vol 38, No. 1, pp. 82–93, Jan 1990.

[102] H.G. Fehn and P. Noll, "Multipath Search Coding of Stationary Signals with Applications to Speech", IEEE Trans. Comms., Vol. Com-30, No.4, pp. 687–701 Apr. 1982

[103] P. Kroon and B.S. Atal, "Quantization Procedures for the Excitation in Celp Coders", '87 ICASSP pp. 1649–1652, 1987.

[104] B.S. Atal and M.R. Schroeder, "Optimizing Predictive Coders for Minimum Audible Noise", Proc. ICASSP'79 pp. 453–455, 1979.

[105] A.J. Viterbi and J.K. Omura, *Principles of Digital Communication and Coding*, pp. 349–378, McGraw-Hill 1979.

[106] G.D. Forney, "The Viterbi Algorithm", Proc. IEEE, vol. 61, pp. 268–278, Mar. 1973.

[107] S.J. Simmons, "Breadth-First Trellis Decoding with Adaptive Effort", IEEE Trans. Comms., Vol. 38 No. 1, pp. 3–12 Jan. 1990.

[108] T.E. Tremain, "The Government Standard Linear Predictive Coding Algorithm: LPC-10", Speech Technology pp. 40–49 Apr. 1982.

[109] R. Steele (Ed), *Mobile Radio Communications*, Pentech Press, pp. 275–280. 1992.

[110] W. Granzow, B.S. Atal, K.K. Paliwal and J. Schroeter, "Speech Coding at 4KB/s and Lower Using Single-Pulse and Stochastic Models of LPC Exctation", Proc. ICASSP 1991 pp. 217–220.

[111] R.A. Rutenbar, "Simulated Annealing Algorithms: An Overview", IEEE Circuits and Devices Mag. pp. 19–26 Jan 1989.

[112] B.B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman 1983.

[113] F.S. Acton, *Numerical Methods that Work*, Harper and Row, 1970.

[114] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes*, Cambridge University Press, 1987.

[115] K.E. Atkinson, *An Introduction to Numerical Analysis*, Wiley 1978.

[116] F. Poirier, "Improving the Training and Testing Speed and the Ability of Generalization in Learning Vector Quantization – DVQ", ICASSP '91 pp. 649–652, 1991.

[117] R.A. Salami and D.G. Appleby, "A New Approach to Low Bit Rate Speech Coding with Low Complexity Using Binary Pulse Excitation (BPE)", IEEE *Workshop on Speech Coding for Telecomm., Vancouver, Canada*, September pp. 5–8, 1989.