UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

# Digital Coding of Speech Using Code Excited Linear Prediction

by

Jason Paul Woodard
B.A., M.Sc.

*A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy at the University of Southampton*

November 1995

SUPERVISOR: Dr. Lajos Hanzo
M.Sc., Ph.D, SMIEEE

**This thesis is dedicated to:**

My parents Joy and Bill and my brother Lee

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

# DIGITAL CODING OF SPEECH USING CODE EXCITED LINEAR PREDICTION

by Jason Paul Woodard

In this thesis the coding of narrow-band speech at rates between four and sixteen thousand bits per second has been studied. The work has concentrated on the Code Excited Linear Prediction (CELP) algorithm, and on means of improving this algorithm.

Conventional CELP coders-decoders (codecs) employing forward adaptive linear prediction analysis have been studied at bit rates between 4 and 8 kilobits per second (kbits/s). These CELP codecs offer good quality reconstructed speech due to their Analysis-by-Synthesis (AbS) structure. This AbS structure has been investigated in detail and ways of extending the AbS loop, and hence improving the quality of the codecs, have been studied.

The recent move towards digital mobile communication systems, and the popularity of such systems, means that a very important aspect of many speech codecs is their sensitivity to bit errors between the encoder and decoder. Various methods of improving this sensitivity are investigated, and also a new method of measuring the sensitivity is proposed to allow channel and speech codecs to be finely matched.

Traditional forward adaptive CELP codecs have a buffering frame size of 20 or 30 ms, and a delay of the order of 70 ms. This delay can cause problems, especially over networks, and so recently a 16 kbits/s CELP codec with a delay of less than 2 ms has been standardised. The extension of this codec to operate as a variable rate codec from 16 down to 8 kbits/s has been studied. Also other low delay CELP codecs operating at bit rates as low as 4 kbits/s with delays of less than 10 ms have been proposed and studied.

## Acknowledgements

I would like to thank my supervisor Dr. Lajos Hanzo for his continual help, enthusiasm and encouragement during my work. I am also grateful to all my colleagues in the Communications Group, both past and present, for their friendship and their help.

I must also thank the Department of Education in Northern Ireland for their financial support of my work.

Finally I must thank my parents for all their support during my studies.

# List of Publications

The following publications have been made as a result of this research:

1. **J.P. Woodard and L. Hanzo** "A Dual-Rate Algebraic CELP Based Speech Transceiver", in the Proceedings of the IEEE Conference on Vehicular Technology, held in Stockholm on 8-10 June 1994, pages 1690-1694.

2. **J.P Woodard and L. Hanzo** "A Reconfigurable Speech Transceiver", in the Proceedings of the IEEE Second International Workshop on Mobile Multimedia Communications, held in Bristol on 11-13 April 1995, Session B1 Paper 5.

3. **J.P. Woodard and L. Hanzo** "A Re-configurable Cordless Telecommunications Scheme", in the Proceedings of Speech and Channel Coding for Transmission (Codierung für Quelle, Kanal und Übertragung), held in Munich on 26-28 October 1994, pages 95-104.

4. **J.P. Woodard and L. Hanzo** "Improvements to the Analysis-by-Synthesis Loop in CELP Codecs", in the Proceedings of the IEE Sixth International Conference on Radio Receivers and Associated Systems, held in Bath on 26-28 September 1995, pages 114-118.

5. **L. Hanzo, R. Lucas and J.P. Woodard** "Automatic Repeat Request Assisted Cordless Telephony" to be published in the Proceedings of the International Conference on Universal Personal Communications, 6-10 November 1995 in Tokyo.

6. **L. Hanzo and J.P. Woodard** "An Intelligent Multimode Voice Communications System for Indoor Communications", to be published in the IEEE Transactions on Vehicular Technology, November 1995.

7. **J.P. Woodard, J.M. Torrance and L. Hanzo** "A Low-Delay Multimode Speech Terminal", to be published in the Proceedings of the IEEE Conference on Vehicular Technology, 28 April - 1 May 1996 in Atlanta.

# Contents

**Abstract** **ii**

**Acknowledgements** **iii**

**List of Publications** **iv**

**1 Introduction** **1**

1.1 Background . . . . . 1

1.2 Organization of Thesis . . . . . 2

**2 The Speech Signal and Common Codecs** **4**

2.1 The Basic Properties of Speech . . . . . 4

2.2 Measuring Speech Quality . . . . . 7

2.2.1 Time Domain Comparisons . . . . . 8

2.2.2 Frequency Domain Comparisons . . . . . 9

2.3 Commonly Used Speech Codecs . . . . . 10

2.3.1 Waveform Codecs . . . . . 11

2.3.2 Source Codecs . . . . . 14

2.3.3 Hybrid Codecs . . . . . 15

2.4 Information Theory and the Rate Distortion Function . . . . . 20

2.4.1 Entropy and Mutual Information . . . . . 20

2.4.2 Channel Capacity . . . . . 23

2.4.3 The Rate Distortion Function . . . . . 24

2.4.4 Applications to Speech Coding . . . . . 26

**3 Low Complexity CELP Codecs** **31**

3.1 Introduction . . . . . 31

3.2 General Coder Structure . . . . . . . . . . . . . . . . . . . . . . . 31
3.3 The Short-Term Synthesis Filter . . . . . . . . . . . . . . . . . . . . 33
3.4 The Error Weighting Filter . . . . . . . . . . . . . . . . . . . . . . . 39
3.5 The Adaptive Codebook Search . . . . . . . . . . . . . . . . . . . . 40
3.6 Fixed Codebook Search . . . . . . . . . . . . . . . . . . . . . . . . 43
3.7 Decoder Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
3.8 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

**4 Optimization of the CELP Codec Parameters 48**
4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
4.2 Calculation of the Excitation Parameters . . . . . . . . . . . . . . . 49
4.2.1 Full Codebook Search Theory . . . . . . . . . . . . . . . . . 49
4.2.2 Sequential Search Procedure . . . . . . . . . . . . . . . . . . 51
4.2.3 Full Search Procedure . . . . . . . . . . . . . . . . . . . . . 52
4.2.4 Sub-Optimal Search Procedures . . . . . . . . . . . . . . . . 54
4.2.5 Quantization of the Codebook Gains . . . . . . . . . . . . . 56
4.3 Calculation of the Synthesis Filter Parameters . . . . . . . . . . . . 59
4.3.1 Bandwidth Expansion . . . . . . . . . . . . . . . . . . . . . 60
4.3.2 Least Squares Techniques . . . . . . . . . . . . . . . . . . . 60
4.3.3 Optimization via Powell's Method . . . . . . . . . . . . . . . 64
4.3.4 Simulated Annealing and the Effects of Quantization . . . . . . 65

**5 The Error Sensitivity of CELP Codecs 70**
5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
5.2 Improving the Spectral Information Error Sensitivity . . . . . . . . . 71
5.2.1 LSF Ordering Policies . . . . . . . . . . . . . . . . . . . . . 71
5.2.2 The Effect of FEC on the Spectral Parameters . . . . . . . . . 74
5.2.3 The Effect of Interpolation . . . . . . . . . . . . . . . . . . . 75
5.3 Improving the Error Sensitivity of the Excitation Parameters . . . . . 76
5.3.1 The Fixed Codebook Index . . . . . . . . . . . . . . . . . . 77
5.3.2 The Fixed Codebook Gain . . . . . . . . . . . . . . . . . . . 77
5.3.3 Adaptive Codebook Delay . . . . . . . . . . . . . . . . . . . 79
5.3.4 Adaptive Codebook Gain . . . . . . . . . . . . . . . . . . . . 79
5.4 Matching Channel Coders to the Speech Coder . . . . . . . . . . . . 80

5.5 Conclusion . . . . . . . . . . 85

**6 A Variable Rate Low Delay CELP Codec 86**

6.1 Introduction . . . . . . . . . . 86
6.2 The G728 16 kbits/s Low Delay CELP Codec . . . . . . . . . . 89
6.2.1 The Synthesis Filter and Backward LPC Adaption . . . . . . . . . . 90
6.2.2 Backward Gain Adaption . . . . . . . . . . 93
6.2.3 The Weighting Filter . . . . . . . . . . 95
6.2.4 Excitation Vector Quantization . . . . . . . . . . 96
6.2.5 The Complexity and Performance of the G728 Codec . . . . . . . . . . 98
6.3 Reducing the Rate of a G728 Codec By Varying the Vector Size . . . 99
6.4 The Effects of Long Term Prediction . . . . . . . . . . 102
6.5 Closed-Loop Codebook Training . . . . . . . . . . 109
6.6 A Variable Rate Codec with a Constant Vector Size . . . . . . . . . . 114
6.7 PostFiltering . . . . . . . . . . 116
6.8 Conclusion . . . . . . . . . . 118

**7 Low Delay CELP Codecs at 8 to 4 kbits/s 119**

7.1 Introduction . . . . . . . . . . 119
7.2 Improvements Due to Increasing Codebook Sizes . . . . . . . . . . 120
7.3 Forward Adaption of the Short Term Synthesis Filter . . . . . . . . . . 121
7.4 Forward Adaption of the Long Term Predictor . . . . . . . . . . 123
7.4.1 Initial Experiments . . . . . . . . . . 123
7.4.2 Quantization of Jointly Optimized Gains . . . . . . . . . . 125
7.4.3 Voiced/Unvoiced Switched Codebooks . . . . . . . . . . 129
7.5 Low Delay Codecs at 4-8 kbits/s . . . . . . . . . . 131
7.6 A Low Delay ACELP Codec . . . . . . . . . . 135
7.7 Conclusion . . . . . . . . . . 140

**8 Conclusions and Further Work 142**

8.1 Summary and Conclusions . . . . . . . . . . 142
8.2 Suggestions for Further Work . . . . . . . . . . 144

**Bibliography 146**

# Chapter 1

# Introduction

## 1.1 Background

The coding of speech signals digitally has been an area of much research in recent years. It is important because of the many advantages digital signals have over their analogue counterparts. They are less prone to interference, and privacy and encryption can be more easily provided in digital systems. Also many complex functions which would be difficult to implement in an analogue circuit can be built into digital systems. However if the analogue speech signal is converted directly into a digital signal through sampling and quantization the resulting bit rate required is too high for many applications. For example narrow-band, telephone quality, speech is typically sampled at 8 kHz, and if linear quantization is used then to maintain good quality each sample must be quantized with 12 bits. This results in a total of 96 thousand bits being used to code each second of speech. This bit rate of 96 kbits/s can be reduced to 64 kbits/s fairly simply through the use of non-linear quantization, but this is still too high for many applications.

For this reason there has been much research into more efficient digital representations of low bandwidth speech. These attempt to code the speech with as few bits as possible while keeping the distortion introduced by the coding as low as possible. The digital signal representing the speech can then for example be transmitted in a much lower bandwidth than would be required if compression were not employed. Thus complex speech coding schemes are now commonly used in various digital mobile phone systems [1] to allow many users to enjoy good quality speech communications over limited bandwidth channels.

In this thesis we have studied the coding of narrow-band speech signals at low to medium rates (between four and sixteen thousand bits per second), and the application of speech codecs in digital mobile communications systems.

## 1.2 Organization of Thesis

This thesis is organized into eight chapters. Chapter 2 discusses briefly the basic properties of speech signals which allow it to be efficiently compressed, and the mechanics of speech production. We also consider the problem of measuring the quality of the speech produced by codecs, and define the distortion measures used in our work. Then we give a brief discussion of the three classes of speech codec that are commonly used. Finally we close the chapter with a discussion of rate distortion theory and its application to speech coding.

In Chapter 3 we consider in detail Code Excited Linear Predictive (CELP) [2] codecs. These codecs have been widely used to produce good quality speech at bit rates as low as 4 kbits/s. We describe the general structure used, and give details of various aspects of the codecs. In their original form CELP codecs are tremendously complex and impractical for use in real time systems. Many methods of reducing this complexity to allow real time implementation have been proposed, and one such method is described in Chapter 3.

CELP codecs are able to offer good speech at low bit rates due to their use of a structure called Analysis-by-Synthesis (AbS). In Chapter 4 we considers methods of improving the performance of CELP codecs by extending the AbS loop and re-optimizing various parameters which are transmitted to the decoder.

Many speech codecs are used over radio links, and an important aspect of such codecs is their sensitivity to errors in the bit stream between the encoder and the decoder. In Chapter 5 we discuss means of measuring and improving this error sensitivity, so that the codecs can be properly utilised in wireless communication systems (especially mobile radio systems).

An important aspect for some applications is the delay introduced by a speech codec. A recently standardised 16 kbits/s CELP codec [3] has a delay of less than 2 ms, and this codec is described in detail in Chapter 6. We then go on to describe how this codec can be modified to produce several variable rate low delay coding schemes operating between 16 and 8 kbits/s.

In Chapter 7 we continue our study of low delay codecs by extending one variable rate codec from Chapter 6 down to 4 kbits/s. We also study several other variable rate low delay codecs operating between 8 and 4 kbits/s, and demonstrate that good quality speech can be achieved with a low delay and a low coding rate. Finally in Chapter 8 we give the conclusions of our research, and give suggestions for further work.

# Chapter 2

# The Speech Signal and Common Codecs

In this chapter we describe the properties of speech which allow it to be efficiently compressed. We then discuss the problem of assessing the quality of the reconstructed speech produced by codecs with an objective measure, and introduce the objective measures which are used in our work. Next we briefly describe the three classes of codecs which are commonly used in speech compression. Finally we introduce the ideas of information theory and the rate distortion function, and apply these to the speech signal in order to estimate theoretical limits on what speech quality can be achieved at given bit rates.

## 2.1 The Basic Properties of Speech

Speech is produced when air is forced from the lungs through the vocal cords and along the vocal tract. The vocal tract extends from the glottis (the opening in the vocal cords) to the mouth, and in an average man is about 17 cm long [4]. It can be considered as a non-uniform acoustic tube whose cross sectional area varies from zero to 20 cm$^2$, and whose shape changes with time depending on the positions of the tongue, lips, jaw and the velum. This acoustic tube introduces short-term correlations (of the order of 1 ms) into the speech signal, and can be thought of as a filter with broad resonances called formants. The frequencies of these formants are controlled by varying the shape of the tract [4]. For some sounds, called nasal sounds, the velum

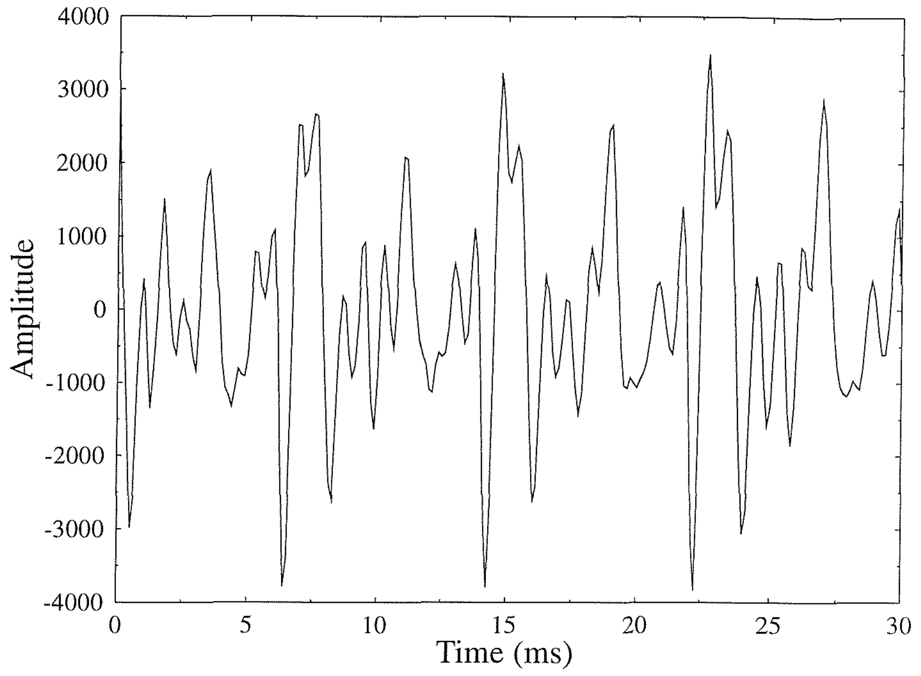

Figure 2.1: Typical Segment of Voiced Speech

is lowered and so the nasal tract is coupled to the vocal tract and it too plays a part in shaping the frequency spectrum of the sound produced. Modelling the vocal tract as a short-term linear filter is an important part of many speech coders, and will be discussed in detail later.

The vocal tract filter is excited by the air forced into it through the vocal cords. Speech sounds can be broken into three classes depending on their mode of excitation.

- Voiced sounds are produced when the vocal cords vibrate open and closed, thus interrupting the flow of air from the lungs to the vocal tract and producing quasi-periodic pulses of air as the excitation. The rate of the opening and closing gives the pitch of the sound. This can be adjusted by varying the shape of, and the tension in, the vocal cords, and the pressure of the air behind them. Voiced sounds show a high degree of periodicity at the pitch period, which is typically between 2 and 20 ms. This long-term periodicity can be seen in Figure 2.1 which shows a segment of voiced speech sampled at 8 kHz. Here the pitch period is

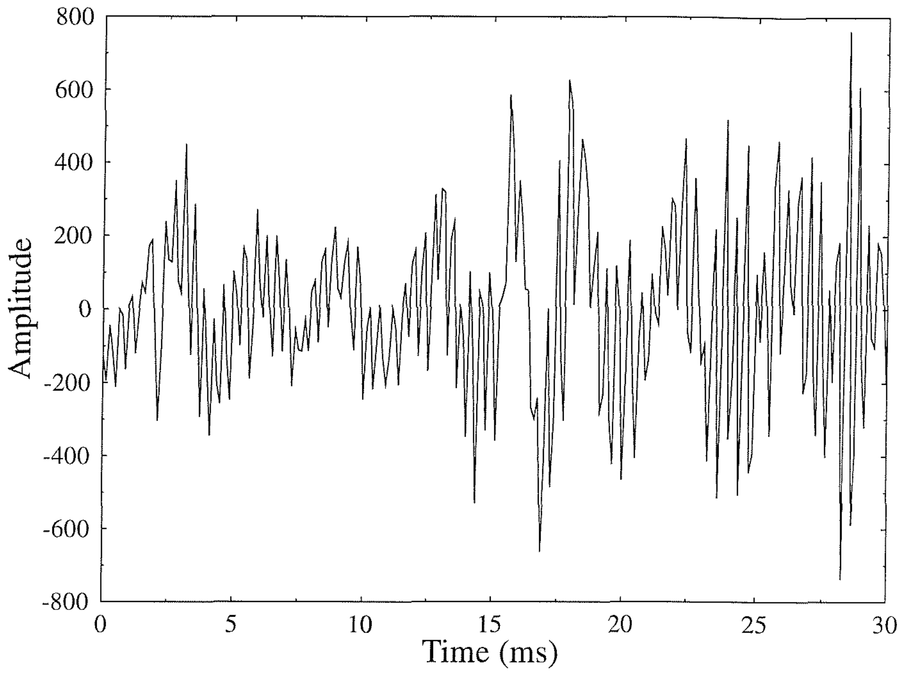

Figure 2.2: Typical Segment of Unvoiced Speech

about 8 ms or 64 samples. The power spectral density for this segment is shown in Figure 2.4.

- Unvoiced sounds result when the excitation is a noise-like turbulence produced by forcing air at high velocities through a constriction in the vocal tract while the glottis is held open. Such sounds show little long-term periodicity as can be seen from Figures 2.2 and 2.5, although short-term correlations due to the vocal tract are still present.

- Plosive sounds result when a complete closure is made in the vocal tract, and air pressure is built up behind this closure and realised suddenly.

Some sounds cannot be considered to fall into any *one* of the three classes above, but are a mixture. For example voiced fricatives result when both vocal cord vibration and a constriction in the vocal tract are present.

Although there are many possible speech sounds which can be produced, the shape of the vocal tract and its mode of excitation change relatively slowly, and so speech can be considered to be quasi-stationary over short periods of time (of the order of 20 ms). We can see that speech signals show a high degree of predictability, due sometimes to the quasi-periodic vibrations of the vocal cords and also to the resonances of the vocal tract. Speech coders attempt to exploit this predictability in order to reduce the data rate necessary for good quality voice transmission.

## 2.2 Measuring Speech Quality

In this section we discuss various objective distortion measures used with speech codecs, and give details of the ones we have used. Such objective measures are important for the following reasons. Firstly they provide a convenient way to compare and report the performance of different codecs. Secondly they can be useful to the designer of a speech codec who wants to know reliably how small changes affect performance, so that the codec can be optimized. Finally in Analysis by Synthesis (AbS) coders, which are the main class of low to medium bit rate speech coders, the encoder attempts to minimise an objective distortion measure. Obviously we would like this objective cost function to give a good indication of how the speech will sound subjectively.

Ultimately the quality of a speech coder must be judged by playing its reconstructed speech to impartial listeners, and asking their opinion. These subjective opinions can be quantized in various ways, and one commonly used measure is the Mean Opinion Score (MOS). When using this method listeners are asked to grade the reconstructed speech quality as excellent, good, fair, poor, or bad. The five possible grades are given a mark from 5 for excellent down to 1 for bad. The MOS for a codec is the mean of these marks. An MOS of 4 or more corresponds to toll quality speech in which the coding noise is difficult to detect. An MOS of between 3 and 4 corresponds to communications quality speech in which the coding noise is easily detectable, but not bad enough to hinder natural conversation.

The problem with subjective measures such as the MOS is that they are time consuming and expensive to carry out. Also the results can vary widely depending on the selection of the listeners and the speech samples used. Therefore an objective measure which could reliably predict some properly controlled subjective measure would be extremely valuable.

### 2.2.1 Time Domain Comparisons

Traditional methods of measuring the quality of reconstructed speech have concentrated on the sample to sample difference between the original and reconstructed signals. The most common is the Signal to Noise Ratio (SNR) which is given by

$$SNR = \frac{\sum_n s(n)^2}{\sum_n \{\hat{s}(n) - s(n)\}^2} \tag{2.1}$$

where s(n) are the original speech samples, $\hat{s}(n)$ are the reconstructed samples, and the summations are carried out over the entire waveforms. The most obvious problem with this measure is that it implies a listener stores an entire speech utterance before making a quality judgement based on the whole waveform. This is obviously untrue. Another problem is that in real speech quality tests the noise in quiet parts of the reconstructed signal is much more important than during periods when $\hat{s}(n)$ is high. This is not reflected in the SNR and it is found [5] that the basic SNR gives a very poor prediction of how distorted a reconstructed signal will sound.

A much improved alternative is the segmental signal to noise ratio (SEGSNR), which is found by splitting the waveforms into short segments (typically around 20 ms long) and finding the signal to noise ratio for each segment. These segment signal to noise ratios are expressed in decibels and then averaged to give the segmental SNR. As the average is found after the signal to noise ratios have been converted into the logarithmic domain, the SEGSNR is effectively a geometric average of the SNRs of the segments of speech. Therefore the very high SNRs which come from segments with high signal levels do not mask the perceptually important performance during quiet periods of speech. This segmental SNR is the main distortion measure we have used.

It is found in tests on human hearing that noise can be entirely "masked" by signals with a similar frequency [6, 7]. This means that noise of certain frequencies can be masked by the speech signal, and so will be inaudible. The perceptually weighted segmental signal to noise ratio attempts to allow for this by giving extra weighting to noise of frequencies where the speech content is low. This is done using a filter of the form

$$H(z) = \frac{A(z)}{A(z/\gamma)} \tag{2.2}$$

where $A(z)$ is the linear prediction error filter given by Equation 2.5, and $\gamma$ is a constant between 0 and 1 depending on the extent of the weighting. It is this perceptually weighted error which is usually minimised in Analysis-by-Synthesis (AbS) coders, and it has recently been proposed [8] as a measure for grading the quality of synthesized speech. We use this distortion measure in Chapter 4 when considering re-optimization techniques in AbS coders.

All of the measures above concentrate on the sample by sample difference between the original and synthesized speech, and it is this which the designer of a codec, or an AbS coder itself, attempts to minimise. However although it it is a sufficient condition for two signals to sound alike that their waveforms be identical, it is not a necessary condition. Consider for example a speech waveform and its inverse. Very few people will be able to hear the difference between the two signals, but the SNR will be extremely poor. Thus time domain measures work well for waveform codecs where the coder tries to match the reconstructed speech to the input waveform as closely as possible. However for lower rate codecs a strict waveform matching process is not practical, and it is useful to complement time domain distortion measures with frequency domain measures.

### 2.2.2 Frequency Domain Comparisons

The most commonly used frequency domain distortion measure for speech coding systems, and the third distortion measure we have used, is the Cepstrum Distance (CD) [9]. This is defined as

$$CD = \frac{10}{\log_e 10}\sqrt{\{C_x(0) - C_y(0)\}^2 + 2\sum_{i=1}^{N}\{C_x(i) - C_y(i)\}^2} \tag{2.3}$$

where $C_x(i)$ and $C_y(i)$ are the linear prediction cepstrum coefficients of the original and the reconstructed signal, and $N$ is the maximum order of the coefficients. The linear prediction filter used to model the vocal tract gives a representation of the smoothed spectrum of the speech from which it was derived. To find the cepstrum coefficients we use the original or the reconstructed speech to derive a set of linear prediction coefficients (for details of how this is done see Chapter 3). The cepstrum coefficients can then be derived iteratively from the corresponding filter coefficients [10].

The CD gives an approximation to the root mean square value of the difference between the logarithmic smoothed power spectra of the original and reconstructed speech signals. It is shown in [10] that $N = 3p$, where $p$ is the order of the linear prediction filter used, gives an accurate approximation. It is found [11] that for medium and high bit rate codecs the cepstrum distance can give a good prediction of the MOS.

Studies of how the ear works and of experiments in the masking of one sound by another [7] have lead to quite detailed formulae describing how we hear sounds. These can be used [5, 12] to predict exactly how the noise introduced by a speech coder will be masked by the speech itself, and should give a better indication of the subjective quality of coded speech than the weighted segmental SNR described above. Recently [13, 14] such measures have been used as the cost function to be minimised in AbS coders, with good results reported.

## 2.3 Commonly Used Speech Codecs

In this section we discuss briefly the main speech coding techniques which are used today, and those which may be used in the future. We also discuss the many different aspects of a speech codec which determine its suitability for a particular application. The two most obvious are the quality of its reconstructed speech and the bit rate necessary to produce this speech. Other aspects which may be equally important depending upon the intended application are

- The robustness of the codec to transmission errors between its encoder and decoder. This is very important for speech codecs which are to be used over radio channels, especially the particularly hostile mobile radio channel. We discuss this issue in detail in Chapter 5.

- The delay introduced by the coding-decoding process. This can be important in two way communication systems, especially when echoes are present. Most low bit rate codecs have high delays of around 50 or 100 ms, but in Chapters 6 and 7 we propose several codecs with much lower delays.

- The complexity of the codec, ie the computational effort (often expressed in terms of millions of operations per second) required to implement the coding and decoding algorithms in real time. Sometimes speech coders are so complex that it is virtually impossible to implement them in real time given the hardware of the day. Generally the more complex the codec is, the more expensive it will be and the more power it will consume.

- How well the codec responds to asynchronous tandems, both to itself and to other speech codecs. This is important because the speech codec may be connected to a network, and other codecs may be used at other points in the network.
- How well the codec copes with non-speech signals, such as music or signalling tones from a network or a data modem.

Details of these factors, and how the performance of speech codecs can be measured in terms of them, can be found in [15].

Obviously no single speech codec, or class of codecs, will be ideal in terms of all these factors. Different types of codecs have different advantages and disadvantages, and often no one codec will be able to match all the design parameters. In order to simplify the description of speech codecs they are often broadly divided into three classes — waveform codecs, source codecs and hybrid codecs. These three classes of codecs are described below.

### 2.3.1 Waveform Codecs

Waveform codecs attempt, without using any knowledge of how the signal to be coded was generated, to produce a reconstructed signal whose waveform is as close as possible to the original. This means that in theory they should be signal independent and work well with non-speech signals. Generally they are low complexity codecs which produce high quality speech at rates above about 16 kbits/s. When the data rate is lowered below this level the reconstructed speech quality that can be obtained degrades rapidly. Detailed descriptions of various aspects of waveform coding can be found in [16]. Here we give only a brief description of a few well known waveform codecs.

The simplest form of waveform coding is Pulse Code Modulation (PCM), which merely involves sampling and quantizing the input waveform. Narrow-band speech is typically band-limited to 4 kHz and sampled at 8 kHz. If linear quantization is used then to give good quality speech around twelve bits per sample are needed, giving a bit rate of 96 kbits/s. This bit rate can be reduced by using non-uniform quantization of the samples. Generally for any given input signal an optimum quantizer can be designed using a set of simultaneous equations [4], which must usually be solved iteratively using for example the Max-Lloyd algorithm. However such a quantizer will be optimum only for the particular probability density function for which it was designed — if the variance of the input signal is changed its performance will be severely degraded. For speech signals the input variance is not known in advance, and changes with time anyway. Therefore instead of using a Max-Lloyd type non-linear quantizer, an approximation to a logarithmic quantizer is often used. Such quantizers give a signal to noise ratio which is almost constant over a wide range of input levels, and at a rate of eight bits/sample (or 64 kbits/s) give a reconstructed signal which is almost indistinguishable from the original. Such logarithmic quantizers were standardised in the 1960's, and are still widely used today. In America $\mu$-law companding is the standard, while in Europe the slightly different A-law compression is used [16]. They have the advantages of low complexity and delay with high quality reproduced speech, but require a relatively high bit rate and have a high susceptibility to channel errors.

A commonly used technique in speech coding is to attempt to predict the value of the next sample from the previous samples. It is possible to do this because of the correlations present in speech samples due to the effects of the vocal tract and the vibrations of the vocal cords, as discussed previously. If the predictions are effective then the error signal between the predicted samples and the actual speech samples will have a lower variance than the original speech samples. Therefore we should be able to quantize this error signal with fewer bits than the original speech signal. This is the basis of Differential Pulse Code Modulation (DPCM) schemes — they quantize the *difference* between the original and predicted signals.

Although the predictors in such coders can be fixed, with coefficients derived from the long-term statistics of speech, much better results are achieved if adaptive predictors which can follow the changing nature of the speech to be coded are used. In forward adaptive predictors a segment, typically around 20 ms, of the speech to be coded is stored in a buffer and the coefficients to be used by the predictor are calculated from this data. These coefficients are then sent to the decoder as side information. In backward adaptive predictors the coefficients to be used are derived from the previously reconstructed speech, and hence are available to both the encoder and decoder without the transmission of any side information. Generally forward adaption will give a higher prediction gain than backward adaption, but requires the transmission of side information about the predictor coefficients used and introduces a significant delay because of the buffering of the input speech. In this thesis we have studied codecs using both forward and backward adaption of the predictors.

In the mid 1980's the CCITT standardised an Adaptive DPCM (ADPCM) codec which used a backward adaptive pole-zero predictor, with six zeros and two poles [17]. The error signal between the input speech samples and the output from this predictor is quantized with four bits per sample, using a backward adaptive quantizer, to give a bit rate of 32 kbits/s. This coder gives speech quality similar to 64 kbits/s log-PCM, and is more robust to channel errors than PCM. However its coding delay and complexity are higher than PCM, although still relatively low when compared to many other codecs.

An important sub-class of DPCM codecs are Delta Modulation codecs [18], which quantize the difference signal described above with only one bit per sample. This means that the bit rate of the codec will be equal to the rate at which the input speech waveform was sampled, and typically this is much higher than the Nyquist rate. The effect of such oversampling is to increase the correlation between adjacent speech samples, and so the variance of the prediction error is much reduced and one bit quantization of this error can be effective. The performance of delta modulation can be improved by using an adaptive one bit quantizer. Such Adaptive Delta Modulation (ADM) codecs give speech quality equivalent to 32 kbits/s ADPCM or 64 kbits/s PCM when operating at 48 kbits/s. They have the advantage of one bit data words, are much more robust to transmission errors than either PCM or ADPCM, and are significantly simpler than the CCITT standard ADPCM. However at 32 kbits/s in error free conditions their reproduced speech quality is slightly inferior to ADPCM [19].

The waveform codecs described above all code speech with an entirely time domain approach. Frequency domain approaches are also possible, and have certain advantages. For example in Sub-Band Coding (SBC) the input speech is split into a number of frequency bands, or sub-bands, and each is coded independently using for example an ADPCM like coder. At the receiver the sub-band signals are decoded and recombined to give the reconstructed speech signal. The advantages of doing this come from the fact that the noise in each sub-band is dependent only on the coding used in that sub-band. Therefore we can allocate more bits to perceptually important sub-bands so that the noise in these frequency regions is low, while in other sub-bands we may be content to allow a high coding noise because noise at these frequencies is less perceptually important. Even when perceptual effects are ignored some gain is possible [16] because speech signals have a non-flat spectrum. Adaptive bit allocation schemes may be used to further exploit these ideas. Sub-band codecs tend to produce communications to toll quality speech in the range 16-32 kbits/s. Due to the filtering necessary to split the speech into sub-bands they are more complex than simple DPCM coders, and introduce more coding delay. However the complexity and delay are still relatively low when compared to most hybrid codecs (see Section 2.3.3).

Another frequency domain waveform coding technique is Adaptive Transform Coding (ATC), which uses a fast transformation (such as the discrete cosine transformation) to split blocks of the speech signal into a large numbers of frequency bands. The number of bits used to code each transformation coefficient is adapted depending on the spectral properties of the speech, and toll quality reproduced speech can be achieved at bit rates as low as 16 kbits/s.

### 2.3.2 Source Codecs

Source coders operate using a model of how the source was generated, and attempt to extract, from the signal being coded, the parameters of the model. It is these model parameters which are transmitted to the decoder. Source coders for speech are called vocoders, and work as follows. The vocal tract is represented as a time-varying filter and is excited with either a white noise source, for unvoiced speech segments, or a train of pulses separated by the pitch period for voiced speech. Therefore the information which must be sent to the decoder is the filter specification, a voiced/unvoiced flag, the necessary variance of the excitation signal, and the pitch period for voiced speech. This is updated every 10-20 ms to follow the non-stationary nature of speech.

The model parameters can be determined by the encoder in a number of different ways, using either time or frequency domain techniques. Also the information can be coded for transmission in various different ways. Vocoders tend to operate at around 2.4 kbits/s or below, and produce speech which although intelligible is far from natural sounding. Increasing the bit rate much beyond 2.4 kbits/s is not worthwhile because of the inbuilt limitation in the coder's performance due to the simplified model of speech production used. The main use of vocoders has been in military applications where natural sounding speech is not as important as a very low bit rate to allow heavy protection and encryption.

### 2.3.3 Hybrid Codecs

Hybrid codecs attempt to fill the gap between waveform and source codecs. As described above waveform coders are capable of providing good quality speech at bit rates down to about 16 kbits/s, but are of limited use at rates below this. Vocoders on the other hand can provide intelligible speech at 2.4 kbits/s and below, but cannot provide natural sounding speech at any bit rate. Although other forms of hybrid codecs exist, the most successful and commonly used are time domain Analysis-by-Synthesis (AbS) codecs. Such coders use the same linear prediction filter model of the vocal tract as found in LPC vocoders. However instead of applying a simple two-state, voiced/unvoiced, model to find the necessary input to this filter, the excitation signal is chosen by attempting to match the reconstructed speech waveform as closely as possible to the original speech waveform. AbS codecs were first introduced in 1982 by Atal and Remde [20] with what was to become known as the Multi-Pulse Excited (MPE) codec. Later the Regular-Pulse Excited (RPE) [21], and the Code-Excited Linear Predictive (CELP) [2] codecs were introduced. These coders will be discussed briefly here. More details about all of them can be found in [1], and we describe CELP codecs further in the next chapter.

A general model for AbS codecs is shown in Figure 2.3. The synthesis filter is usually an all pole, short-term, linear filter of the form

$$H(z) = \frac{1}{A(z)} \tag{2.4}$$

where

$$A(z) = 1 - \sum_{i=1}^{p} a_i z^{-i} \tag{2.5}$$

is the prediction error filter determined by minimising the energy of the residual signal produced when the original speech segment is passed through it (see Chapter 3 for details). The order $p$ of the filter is typically around ten. This filter is intended to model the correlations introduced into the speech by the action of the vocal tract.

The synthesis filter may also include a pitch filter to model the long-term periodicities present in voiced speech. Alternatively these long-term periodicities may be

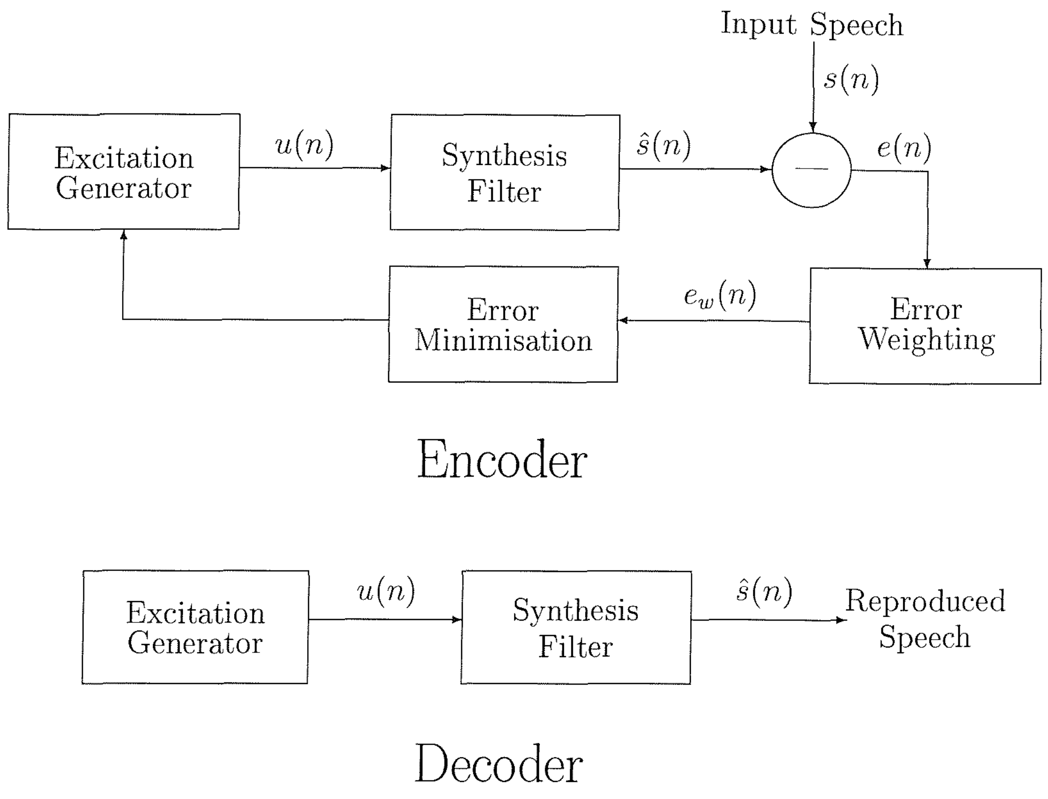

Figure 2.3: AbS Codec Structure

exploited by including an adaptive codebook in the excitation generator so that the excitation signal $u(n)$ includes a component of the form $Gu(n - \alpha)$, where $\alpha$ is the estimated pitch period. Generally MPE and RPE codecs will work without a pitch filter, although their performance will be improved if one is included. For CELP codecs however a pitch filter is extremely important, for reasons discussed below.

The error weighting block is used to shape the spectrum of the error signal in order to reduce the subjective loudness of this error [22]. This is possible because, as described earlier, the error signal in frequency regions where the speech has high energy will be at least partially masked by the speech. The weighting filter usually takes the same form as Equation 2.2, and emphasises the noise in the frequency regions where the speech content is low. Thus minimising the weighted error concentrates the energy of the error signal in frequency regions where the speech has high energy. Therefore the error signal will be at least partially masked by the speech, and so its subjective importance will be reduced. Although such weighting tends to slightly decrease the SNR of the codec, it produces a significant improvement in the subjective quality of the reconstructed speech.

The distinguishing feature of AbS codecs is how the excitation waveform $u(n)$ for the synthesis filter is chosen. Conceptually every possible waveform is passed through the filter to see what reconstructed speech signal this excitation would produce. The excitation which gives the minimum weighted error between the original and the reconstructed speech is then chosen by the encoder and used to drive the synthesis filter at the decoder. It is this 'closed-loop' determination of the excitation which allows AbS codecs to produce good quality speech at low bit rates. However the numerical complexity involved in passing every possible excitation signal through the synthesis filter is huge. Usually some means of reducing this complexity, without compromising the performance of the codec too badly, must be found (see Chapter 3 and [1]).

The differences between MPE, RPE and CELP codecs arise from the representation of the excitation signal $u(n)$ used. In multi-pulse codecs $u(n)$ is given by a fixed number of non-zero pulses for every frame of speech. The positions of these non-zero pulses within the frame, and their amplitudes, must be determined by the encoder and transmitted to the decoder. In theory it would be possible to find the very best values for all the pulse positions and amplitudes, but this is not practical due to the excessive complexity it would entail. In practice some sub-optimal method of finding the pulse positions and amplitudes must be used. Usually the positions are found one at a time as follows. Initially all the pulses are assumed to have zero amplitude except one. The position and amplitude of this first pulse can then be found. Then using this information the position and amplitude of the second pulse can be determined. This continues until all the pulses have been found. Once a pulse position is determined it is fixed, but the amplitudes of the previously found pulses can be re-optimized at each stage [23]. The quality of the reconstructed speech possible from MPE codecs is largely determined by how many non-zero pulses are used in the excitation. However this is constrained by the bit-rate necessary to transmit information about the pulse positions and amplitudes. Typically about 4 pulses per 5 ms are used, and this leads to good quality reconstructed speech and a bit-rate of around 10 kbits/s.

Like the MPE codec the Regular Pulse Excited (RPE) codec uses a number of non-zero pulses to give the excitation signal $u(n)$. However in RPE codecs the pulses are regularly spaced at some fixed interval, and the encoder needs only to determine the position of the first pulse and the amplitude of all the pulses. Therefore less information needs to be transmitted about pulse positions, and so for a given bit rate the RPE codec can use many more non-zero pulses than MPE codecs. For example at a bit rate of about 10 kbits/s around 10 pulses per 5 ms can be used in RPE codecs, compared to 4 pulses for MPE codecs. This allows RPE codecs to give slightly better quality reconstructed speech quality than MPE codecs. However they also tend to be more complex. The pan-European GSM mobile telephone system [1] uses a simplified RPE codec, with long-term prediction, operating at 13 kbits/s to provide toll quality speech.

Although MPE and RPE codecs can provide good quality speech at rates of around 10 kbits/s and higher, they are not suitable for rates much below this. This is due to the large amount of information that must be transmitted about the excitation pulses' positions and amplitudes. If we attempt to reduce the bit rate by using fewer pulses, or coarsely quantizing their amplitudes, the reconstructed speech quality deteriorates rapidly. Currently the most commonly used algorithm for producing good quality speech at rates below 10 kbits/s is Code Excited Linear Prediction (CELP). This approach was proposed by Schroeder and Atal in 1985 [2], and differs from MPE and RPE in that the excitation signal is effectively vector quantized. The excitation is given by an entry from a large vector quantizer codebook, and a gain term to control its power. Typically the codebook index is represented with about 10 bits (to give a codebook size of 1024 entries) and the gain is coded with about 5 bits. Thus the bit rate necessary to transmit the excitation information is greatly reduced – around 15 bits compared to the 47 bits used for example in the GSM RPE codec.

Originally [2] the codebook used in CELP codecs contained white Gaussian sequences. This was because it was assumed that long and short-term predictors would be able to remove nearly all the redundancy from the speech signal to produce a random noise-like residual. Also it was shown that the short-term probability density function (pdf) of this residual was nearly Gaussian. Schroeder and Atal found that using such a codebook to produce the excitation for long and short-term synthesis filters could produce high quality speech. However to choose which codebook entry to use in an analysis-by-synthesis procedure meant that every excitation sequence had to be passed through the synthesis filters to see how close the reconstructed speech it produced would be to the original. This meant the complexity of the original CELP codec was much too high for it to be implemented in real-time – it took 125 seconds of

Cray-1 CPU time to process 1 second of the speech signal. Since 1985 much work on reducing the complexity of CELP codecs, mainly through altering the structure of the codebook, has been done. Also large advances have been made with the speed possible from DSP chips, so that now it is relatively easy to implement a real-time CELP codec on a single, low cost, DSP chip. Several important speech coding standards have been defined based on the CELP principle, for example the American Department of Defence (DoD) 4.8 kbits/s codec [24], and the CCITT low-delay 16 kbits/s codec [3]. We give a detailed description of CELP codecs in the next chapter.

The CELP coding principle has been very successful in producing communications to toll quality speech at bit rates between 4.8 and 16 kbits/s. The CCITT standard 16 kbits/s codec produces speech which is almost indistinguishable from 64 kbits/s log-PCM coded speech, while the DoD 4.8 kbits/s codec gives good communications quality speech. Recently much research has been done on codecs operation below 4.8 kbits/s, with the aim being to produce a codec at 2.4 or 3.6 kbits/s with speech quality equivalent to the 4.8 kbits/s DoD CELP. We will briefly describe here a few of the approaches which seem promising in the search for such a codec.

The CELP codec structure can be improved and used at rates below 4.8 kbits/s by classifying speech segments into one of a number of types (for example voiced, unvoiced and transition frames) [25]. The different speech segment types are then coded differently with a specially designed encoder for each type. For example for unvoiced frames the encoder will not use any long-term prediction, whereas for voiced frames such prediction is vital but the fixed codebook may be less important. Such class-dependent codecs have been shown to be capable of producing reasonable quality speech at rates down to 2.4 kbits/s [26]. Multi-Band Excitation (MBE) codecs [27] work by declaring some regions in the frequency domain as voiced and others as unvoiced. They transmit for each frame a pitch period, spectral magnitude and phase information, and voiced/unvoiced decisions for the harmonics of the fundamental frequency. Originally it was shown that such a structure was capable of producing good quality speech at 8 kbits/s, and since then this rate has been significantly reduced (see for example [28]). Finally Kleijn has suggested an approach for coding voiced segments of speech called Prototype Waveform Interpolation (PWI) [29]. This works by sending information about a single pitch cycle every 20-30 ms, and using interpolation to reproduce a smoothly varying quasi-periodic waveform for voiced speech segments. Excellent quality reproduced speech can be obtained for voiced speech at rates as low as 3 kbits/s. Such a codec can be combined with a CELP type codec for the unvoiced segments to give good quality speech at rates below 4 kbits/s.

## 2.4 Information Theory and the Rate Distortion Function

In this section we briefly discuss the basic ideas of information theory and the rate distortion function, and how they apply to speech coding. A more complete description of the mathematics behind rate distortion theory can be found in [30], and many of the ideas discussed here are described in more detail in Appendices C and D of [16].

### 2.4.1 Entropy and Mutual Information

Much of rate distortion theory is based on the concepts of the entropy of a source, and the average mutual information between two sources. These concepts are defined here.

Consider first a discrete time and amplitude source $x$ with no memory, whose output $x(n)$ at any time $n$ can be one of $N$ possible symbols $x_1, x_2 \cdots x_N$. If each symbol $x_j$ has a probability $P_j$ of occurring at each time instant then the information received upon finding $x(n) = x_j$ is defined as

$$i(j) = -\log_2 P_j \ \text{ bits.} \tag{2.6}$$

The entropy of the source is then defined as the average information it gives

$$H(x) = -\sum_{j=1}^{N} P_j \log_2 P_j \ \text{ bits/symbol} \tag{2.7}$$

and lies in the range $0 \leq H(x) \leq \log_2 N$. Shannon's noiseless coding theorem states that the minimum rate $R$ necessary for the perfect (ie noiseless) transmission of a source with entropy $H(x)$ is given by

$$R = H(x) + \epsilon \ \text{ bits/symbol} \tag{2.8}$$

where $\epsilon$ is a positive number which can be made arbitrarily close to zero. Thus the entropy of a discrete source can be thought of as the transmission rate necessary for perfect coding of the source.

Next consider two discrete variables $x \in (x_1, x_2 \cdots x_N)$ and $y \in (y_1, y_2 \cdots y_M)$, with probability distributions $P_j$ and $Q_k$ and a joint probability distribution $P(j,k)$. The mutual information received upon finding $x(n) = x_j$ and $y(n) = y_k$ is defined as

$$i(j;k) = i(j) - i(j|k). \tag{2.9}$$

Here $i(j|k) = -\log_2 P(j|k)$ is the information one receives when told $x(n) = x_j$ if it is already known that $y(n) = y_k$. If we use the definitions of $i(j)$ and $i(j|k)$ given above then we can rewrite $i(j;k)$ as

$$\begin{aligned} i(j;k) &= i(j) - i(j|k) \\ &= -\log_2 P_j + \log_2 P(j|k) \\ &= \log_2 \frac{P(j|k)}{P_j} \\ &= \log_2 \frac{P(j,k)}{P_j Q_k}. \end{aligned} \tag{2.10}$$

The average mutual information is then given by the average of $i(j;k)$ ie

$$\begin{aligned} I(x;y) &= \sum_{j,k} P(j,k) i(j;k) \\ &= \sum_{j,k} P(j,k) log_2 \frac{P(j,k)}{P_j Q_k} = I(y;x). \end{aligned} \tag{2.11}$$

This average mutual information can be thought of as the average information reception of a value of $x$ gives about the value of $y$, or vice-versa. If we define the conditional entropies $H(x|y)$ and $H(y|x)$ as

$$H(x|y) = -\sum_{j,k} P(j,k) \log_2 P(j|k) \tag{2.12}$$

and

$$H(y|x) = -\sum_{j,k} P(j,k) \log_2 P(k|j) \tag{2.13}$$

then the average mutual information can be written as

$$I(x;y) = H(x) - H(x|y) = H(y) - H(y|x). \tag{2.14}$$

So far we have considered only discrete amplitude sources. For a continuous amplitude source the absolute entropy (ie the data rate necessary to reconstruct the signal with zero distortion) is infinite. However for a source with a continuous probability distribution function $p(x)$ we can define its *differential* entropy as

$$h(x) = -\int_{-\infty}^{\infty} p(x)\log_2 p(x)\;dx. \tag{2.15}$$

This is analogous to Equation 2.7 for the absolute entropy of a discrete source. Note however that the differential entropy of a continuous source gives us no information about the rate necessary to transmit the signal $x$, and depending on the form of $p(x)$ it can be positive, negative or zero. It can be shown that, for a given variance $\sigma^2$, the maximum differential entropy is given by the Gaussian pdf

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-x^2}{2\sigma^2}\right) \tag{2.16}$$

which has a differential entropy

$$h(x) = \log_2\sqrt{2\pi e\sigma^2}. \tag{2.17}$$

The average mutual information between two continuous variables $x$ and $y$ can be defined in a similar way to the discrete case with summations over the discrete variables being replaced with integrations over the continuous variables, and absolute entropies replaced with differential entropies.

The discussion above has considered only memoryless sources. For a source with memory we can take account of the memory by arranging blocks of T successive output symbols into vectors, and considering the probability of the source generating these vectors rather than individual symbols. If $P(\underline{x})$ is the probability of the source producing a given vector of length $T$ then the entropy of a discrete source is defined as

$$H(x) = -\lim_{T\Rightarrow\infty}\frac{1}{T}\sum_{\text{all }\underline{x}} P(\underline{x})\log_2 P(\underline{x}) \tag{2.18}$$

and similarly for continuous sources. The average mutual information definitions for memoryless sources given above can be generalised in a similar way.

It is almost always impossible to calculate the differential entropy of a continuous source with memory. However for a Gaussian source we can generalise Equation 2.17 to

$$h(x) = \log_2\sqrt{2\pi e\gamma^2\sigma^2} \tag{2.19}$$

where $\gamma^2$ is the spectral flatness measure of the source [16], defined as

$$\gamma^2 = \frac{\exp\left(\frac{1}{2\pi}\int_{-\pi}^{\pi} \ln S(\omega)\ dw\right)}{\frac{1}{2\pi}\int_{-\pi}^{\pi} S(\omega)\ dw} \tag{2.20}$$

and $S(\omega)$ is the power spectral density of the signal. The spectral flatness measure can also be written as

$$\gamma^2 = \frac{\eta^2}{\sigma^2} \tag{2.21}$$

where $\eta^2$ is the minimum error energy that can be produced by linear prediction of the signal. Thus $\gamma^2$ is the inverse of the maximum prediction gain possible through linear prediction. Note that $\gamma^2 \leq 1$ , with equality only for sources with a white spectrum (ie memoryless sources). This means that the differential entropy of a Gaussian source is reduced if the source has memory, and this is also true for other types of sources. Also, as with memoryless sources, the differential entropy of a source with a given variance and spectral flatness measure is maximised if the source has a Gaussian $p(x)$.

## 2.4.2 Channel Capacity

Consider a channel with input $x$ and output $y$, where $x$ and $y$ can be either continuous or discrete variables. The average mutual information $I(x;y)$ can be thought of as the average amount of information reception of a symbol $y$ gives us about the value of $x$. This depends not only on the conditional probabilities $p(y|x)$ for the channel, but also on the source probability distribution. Therefore the capacity of a given channel is defined as the maximum value of $I(x;y)$ that can be obtained for that channel over all possible source probability distributions. The significance of the channel capacity as defined above is given by Shannon's noisy channel coding theorem, which states that data can be transmitted over the channel with arbitrarily low error probability provided that the rate is less than the capacity.

For an analogue channel the average mutual information can be expressed in terms of differential entropies as

$$I(x;y) = h(x) - h(x|y) = h(y) - h(y|x) \tag{2.22}$$

where $h(x)$ is the source entropy, $h(y)$ is the destination entropy, $h(x|y)$ is the average information per symbol lost to the channel (the equivocation) and $h(y|x)$ is the error entropy, or the entropy of the noise introduced by the channel. Now consider an

Additive White Gaussian Noise (AWGN) channel. As the channel noise is Gaussian its error entropy will be given by

$$h(y|x) = \log_2 \sqrt{2\pi \mathrm{eN}} \tag{2.23}$$

where $N$ is the noise variance. Therefore the channel capacity (in bits per symbol) is given by

$$C = \max\left(I(x,y)\right) = \max\left(h(y) - \log_2 \sqrt{2\pi \mathrm{eN}}\right). \tag{2.24}$$

Thus to realise the channel capacity we should use an input distribution which maximises the received entropy $h(y)$. If the received *signal* variance is $S$, the total (ie signal plus noise) received variance will be $S + N$, and the maximum possible $h(y)$ will be

$$h(y) = \log_2 \sqrt{2\pi \mathrm{e}(\mathrm{S}+\mathrm{N})}. \tag{2.25}$$

Thus

$$C = \max(h(y) - \log_2 \sqrt{2\pi \mathrm{eN}}) = \log_2 \sqrt{\frac{S+N}{N}} \text{ bits/symbol.} \tag{2.26}$$

The total capacity (in bits/second) for a channel of bandwidth $B$ Hertz is the above capacity multiplied by $2B$ ie

$$C = B \log_2 \left(1 + \frac{S}{N}\right) \text{ bits/second.} \tag{2.27}$$

This is the famous Shannon-Hartley capacity theorem for an AWGN channel. Channels with non-Gaussian additive noise will have smaller error entropies and therefore higher capacities.

### 2.4.3 The Rate Distortion Function

The rate distortion function $R(D)$ defines, for a given source and distortion measure, the transmission rate $R$ necessary to be able to reconstruct the source signal with an average distortion less than or equal to $D$. Like the capacity of a channel it is defined in terms of the average mutual information $I(x;y)$, where now $x$ represents the original signal and $y$ represents the decoded signal. For a given mapping between $x$ and $y$, $I(x;y)$ represents the average information flow between the two. The rate distortion function is defined as the *minimum* possible value of $I(x;y)$, where the minimisation is carried out over all the mappings between $x$ and $y$ which give an average distortion less than or equal to $D$ between the two. Contrast this with the definition of the channel capacity which is the *maximum* possible value of $I(x;y)$ when the mapping between $x$ and $y$ is fixed, but the input distribution is varied.

Generally the rate distortion function is not known for most sources and distortion measures. However for the mean square error distortion function some results are known, and we consider only this measure. For a memoryless Gaussian source with variance $\sigma^2$ we have

$$\begin{aligned} R(D) &= \frac{1}{2}\log_2\left(\frac{\sigma^2}{D}\right) && 0 \leq D \leq \sigma^2 \\ &= 0 && \text{otherwise.} \end{aligned} \tag{2.28}$$

This means that in theory it is possible to code a memoryless Gaussian source with a signal to noise ratio of 6.02 dB for every bit per symbol which is used. For other memoryless sources $R(D)$ curves can be calculated numerically, and it can be shown that the rate distortion function for a Gaussian source upper bounds $R(D)$ for all other sources with the same variance. For example a memoryless source with the Gamma pdf (which is a close approximation to the long-term pdf of speech signals) can be coded with an SNR of 8.53 dB at the rate of 1 bit/sample [31], compared to an SNR of 6.02 dB for a Gaussian source at the same rate.

For sources with memory the rate necessary to reproduce the source with a given distortion is always less than the rate for a similar source with no memory. For a coloured Gaussian source with a power spectral density $S(\omega)$, $R(D)$ can be calculated using the following equations

$$\begin{aligned} D(\phi) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \min\left(\phi, S(\omega)\right) d\omega \\ R(\phi) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \max\left(0, \frac{1}{2}\log_2 \frac{S(\omega)}{\phi}\right) d\omega. \end{aligned} \tag{2.29}$$

This means that a level $\phi$ is chosen, depending on the required rate/distortion. In the frequency regions where $\phi \geq S(\omega)$, known as the stop-bands, no information is transmitted. For such regions, to minimise the average distortion, the decoder should set the reconstructed power spectral density to zero. Therefore in the stop-bands the average distortion is equal to the original PSD $S(\omega)$. In the frequency regions where $S(\omega) \geq \phi$, known as the pass-bands, the distortion is equal to $\phi$ and the transmission rate is $\log_2 \sqrt{S(\omega)/\phi}$.

For small distortions, ie if $\phi$ is such that $S(\omega) \geq \phi$ for all $\omega$, Equation 2.29 can be simplified to

$$R(D) = \frac{1}{2}\log_2 \frac{\sigma^2\gamma^2}{D} \tag{2.30}$$

where $\sigma^2$ is the variance and $\gamma^2$ is the spectral flatness measure of the source. For a memoryless source $\gamma^2 = 1$ and Equation 2.30 reduces to Equation 2.28.

The SNR (in dB) of the reconstructed signal is given by $10\log_{10}(\sigma^2/D)$ and so using Equation 2.30 we see that the maximum SNR possible when coding at a rate of $R$ bits/sample is, for large $R$

$$\begin{aligned} \mathrm{SNR}_{\mathrm{max}} &= 2R * 10\log_{10}(2) - 10\log_{10}\gamma^2 \\ &= T_B + T_P \end{aligned} \tag{2.31}$$

where

$$T_B = 2R * 10\log_{10}(2) \approx 6R \tag{2.32}$$

and

$$T_P = 10\log_{10}\frac{1}{\gamma^2}. \tag{2.33}$$

From Equations 2.21 and 2.33 we see that $T_P$ can be thought of as the best possible gain (in dB) that can be produced by linear prediction of the signal.

As in the memoryless case, for non-Gaussian sources the exact form of $R(D)$ is not known. However it can be shown that for a source with a given power spectral density, $R(D)$ will be less than or equal to the rate distortion function for a Gaussian source with the same PSD.

### 2.4.4 Applications to Speech Coding

Rate distortion theory assumes that the source we are coding is stationary, with a power spectral density known at both the encoder and decoder. Speech however is non-stationary and can only be considered to be quasi-stationary for short periods of time of the order of 20 ms. Also explicit rate distortion functions are known only for sources with a Gaussian distribution, which is not a good model for the long-term pdf of speech signals. Nevertheless we can use the theory to give some idea of the optimum performance possible from a speech coder, and how such an optimum coder will behave. For example in [32] the predictions of rate distortion theory, assuming a Gaussian source, are shown to agree reasonably well with the results from real speech coders.

Equation 2.31 gives the maximum possible $SNR$ for a stationary Gaussian source (for small distortions) in terms of the data rate (per sample) and the maximum possible prediction gain $T_P$ of the signal. This gain was taken by O'Neal in [33] to be 21 dB (after work by Atal and Schroeder). Thus if speech was a stationary Gaussian source we could write for large rates $R$

$$\mathrm{SNR}_{\mathrm{max}} \approx 21 + 6\mathrm{R} \quad (\mathrm{dB}). \tag{2.34}$$

For lower rates (such that $\phi > \min S(\omega)$) the above equation will not be valid, and we must use Equation 2.29 to calculate the SNR possible for a given rate. We did this with about seven seconds of speech data, sampled at 8 kHz, obtained from two male and two female speakers. The speech data was split into 256 sample segments, and we used the Fast Fourier Transform (FFT) on the Hamming-windowed samples to find the power spectrum $S(\omega)$ for each segment. Then for each segment an iterative procedure was used with Equation 2.29 to find $\phi$ and hence $D$ for a given rate.

The spectra of two typical segments, one voiced and the other unvoiced, are shown in Figures 2.4 and 2.5. Also shown in these figures as the dashed lines are the functions $\min(\phi, S(\omega))$ which give the power spectra of the noise in an optimum encoder. The values of $\phi$ have been set to give a rate of one bit/sample, and we found that at this rate the SNRs were about 21 dB for the voiced speech, and 14 dB for the unvoiced speech. The voiced segment can be coded with a lower distortion than the unvoiced segment because of its greater predictability – we found that $T_P = -10\log_{10}\gamma^2$ was 20 dB for the voiced speech and 15 dB for the unvoiced speech.

Figure 2.6 shows the predicted maximum segmental SNR against the data rate. This was calculated by finding the SNR in decibels for each speech segment as described above, and then averaging. We also calculated the maximum prediction gain $T_P$ in a similar way and found that it was 20.9 dB, agreeing well with the value used in [33]. Notice that for rates above about 1.5 bits/sample the curve in Figure 2.6 becomes approximately a straight line as predicted by Equation 2.34.

In the discussion above we have considered each 256 sample (32 ms) segment of speech to be a stationary Gaussian signal. We now discuss how these assumptions are likely to affect our results. Firstly, although the long-term statistics of speech closely match the Gamma pdf, the short-term statistics are approximately Gaussian [34]. Therefore assuming the 32 ms segments of speech to be Gaussian will probably not distort our results too badly. The non-stationarity will have a greater effect, and

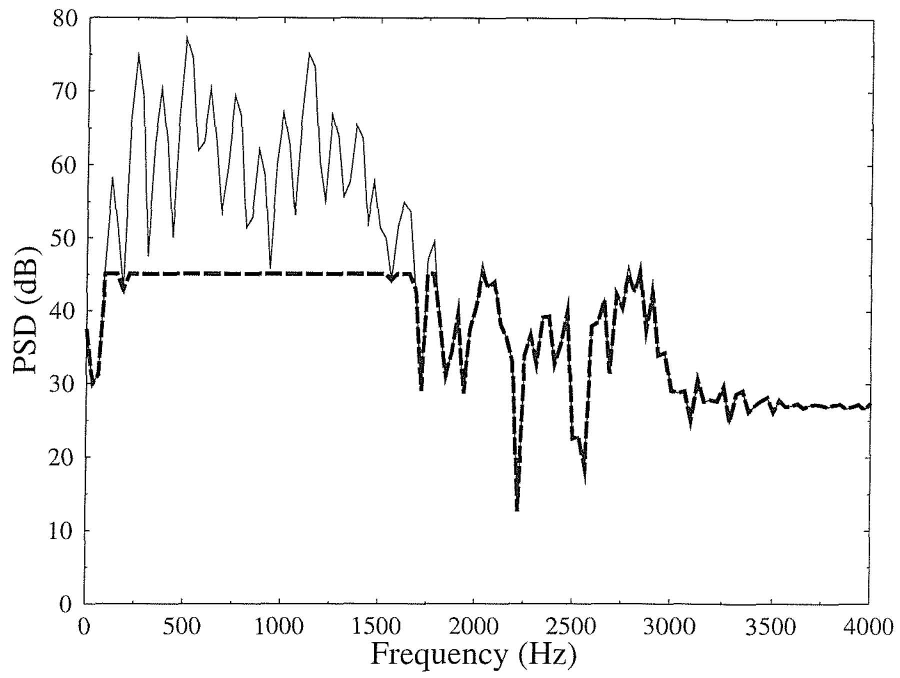

Figure 2.4: Power Spectrum Density for a Segment of Voiced Speech

will result in the true 'maximum SNR' function for speech lying somewhere below that drawn in Figure 2.6. Thus the maximum SNR values we have calculated give an upper bound for the SNR that could be obtained with a real speech coder.

We can produce a tighter bound by trying to approximate how the non-stationary nature of speech will affect our results. The first difference will be that for a speech coder to obtain a prediction gain close to $T_P$ it will need to send side information about the current spectrum of the signal to the decoder. The rate necessary for this side information (say $\hat{R}$ bits/sample) will reduce the effective rate $R$ of the coder. The side information necessary to support short-term linear prediction is about 1/8 bits per sample, and we take this as the necessary rate $\hat{R}$. Secondly the prediction gain possible will be reduced below $T_P$ because of the non-stationary nature of speech, and also because only limited information about the present correlations in the signal is sent in the side information (ie the gain will be dependent on $\hat{R}$). For example for the speech file described earlier, the calculated value of $T_P$ is 21 dB, but the gain achieved

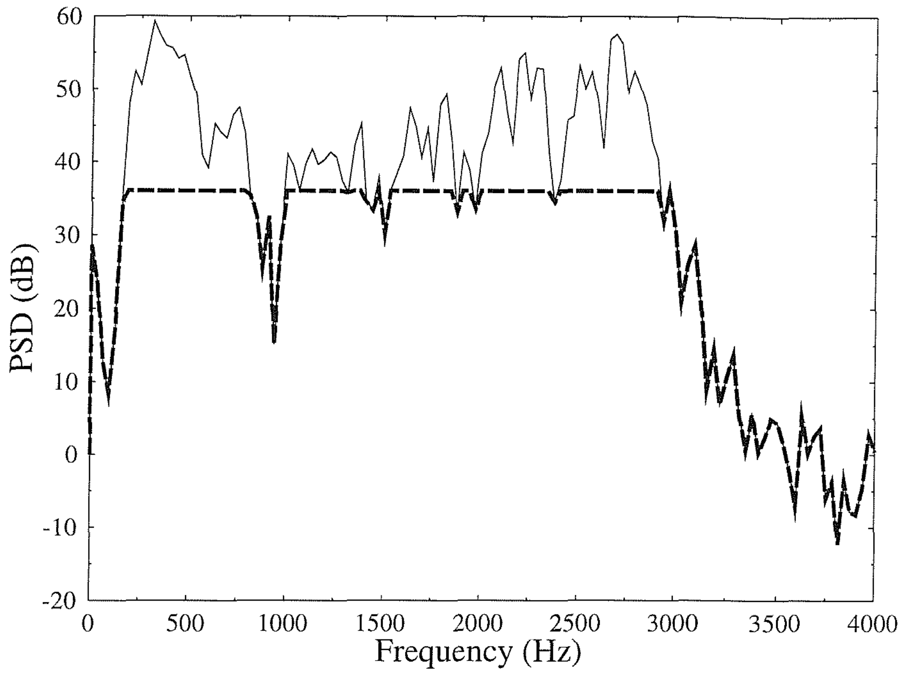

Figure 2.5: Power Spectrum Density for a Segment of Unvoiced Speech

with short-term linear prediction (of order 10) is only 17 dB.

At bit rates above about 1.5 bits per sample Equation 2.34 gives a good approximation to the maximum segmental SNR possible for a speech codec, provided we take into account the effects mentioned above. For a 16 kbits/s codec the bit rate is 2 bits per sample and so the effective rate $R$ is about 1.875 bits per sample. Thus, using 4 dB as the value of the reduction of the prediction gain $T_p$, the maximum segmental SNR predicted for a 16 kbits/s speech codec is about 28 dB. At rates of 1 bit per sample and less Equation 2.34 is no longer accurate, and so we must use Figure 2.6 to estimate the maximum segmental SNR of speech codecs at these rates. Also the effect of the reduction in $T_p$ will be less significant than the 4 dB figure used above, because of the fall of the maximum SNR figures below $T_p + 6R$. We take a decrease of about 2 dB to be typical at low rates. These assumptions mean that the effective rate $R$ for a 4.7 kbits/s coder will be about 0.45 bits/sample, giving a maximum segmental SNR of around $17.5 - 2 = 15.5$ dB. Similarly we predict a maximum possible segmental

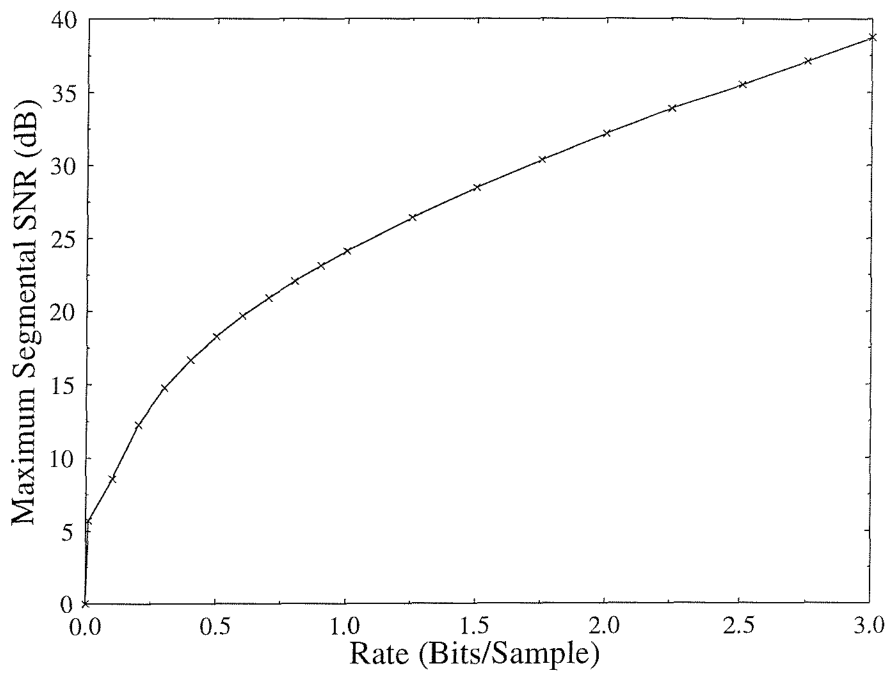

Figure 2.6: Predicted Maximum Possible Segmental SNR

SNR of about 19 dB at 7.1 kbits/s. It is interesting to compare these figures with those obtained for the real speech coders, operating at the same rates, described later.

# Chapter 3

# Low Complexity CELP Codecs

## 3.1 Introduction

In the previous chapter we introduced Analysis-by-Synthesis codecs, and in particular the Code Excited Linear Predictive (CELP) codec. In this chapter we give details of the form of CELP codecs, and the techniques used by encoders and decoders to produce good quality reconstructed speech. We also describe some of the methods which can be used to reduce the computational complexity associated with CELP codecs. In this and the next two chapters we have concentrated on relatively high delay codecs using forward adaption of the synthesis filter. We simulated two such CELP codecs operating at 4.7 and 7.1 kbits/s, and our results are reported here. Low delay backward adaptive CELP codecs are described in Chapters 6 and 7.

## 3.2 General Coder Structure

CELP coders use an Analysis-by-Synthesis (AbS) scheme in which the information to be transmitted to the receiver is largely determined in a closed-loop fashion so that the signal reconstructed by the decoder is as close as possible to the original speech. A block diagram of the structure often used in CELP codecs is shown in Figure 3.1. These differ from the general AbS codec structure shown in Figure 2.3 in two ways. Firstly the excitation signal $u(n)$ is given by the sum of the outputs from two code-books. The adaptive codebook is used to model the long-term periodicities present in voiced speech, while the fixed codebook models the random noise-like residual signal

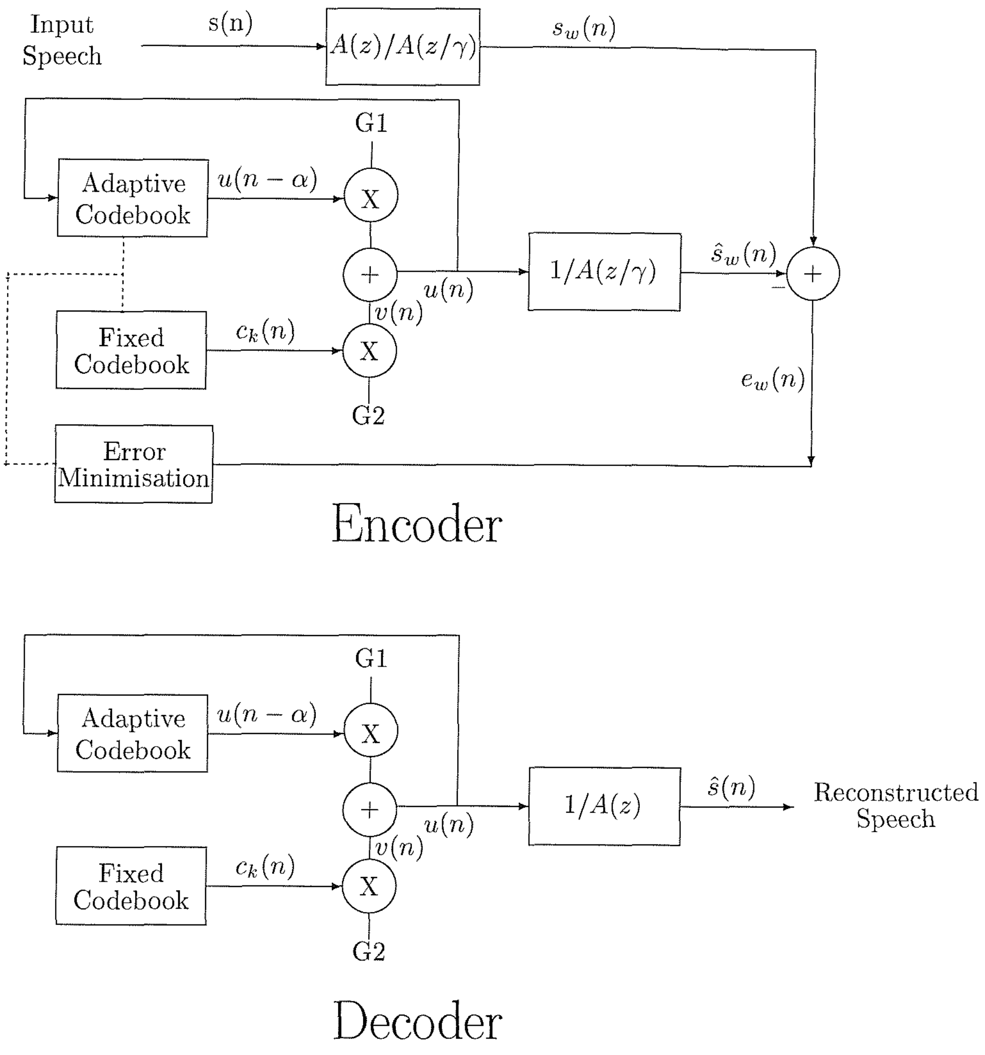

Figure 3.1: CELP Codec Structure

which remains after both long and short-term prediction. The second difference is that the error weighting filter in Figure 2.3 has been moved so that the input speech signal $s(n)$ and the reconstructed speech signal $\hat{s}(n)$ are both separately weighted before their difference is found. This is permissible because of the linear nature of the weighting filter, and is done because it makes the determination of the codebook parameters less complex. With a synthesis filter of the form $1/A(z)$, and an error weighting filter $A(z)/A(z/\gamma)$, we get the filters shown for the encoder in Figure 3.1. The filter $1/A(z/\gamma)$ in the encoder is called the weighted synthesis filter – when fed with an excitation signal it produces a weighted version $\hat{s}_w(n)$ of the reconstructed speech $\hat{s}(n)$.

In this chapter we consider only systems in which forward-adaptive filtering is used. For such systems the input speech is split up into frames for processing, where a frame is of the order of 20 ms long. The frames are usually further divided into sub-frames, with around 4 sub-frames per frame. The short-term synthesis filter coefficients are determined and transmitted once per frame, while the adaptive and fixed codebook parameters are updated once per sub-frame. The 4.7 kbits/s codec we have simulated has a frame length of 30 ms with 4 sub-frames of 7.5 ms each, while our 7.1 kbits/s codec has a frame length of 20 ms with 5 ms long sub-frames.

The encoding procedure generally takes place in three stages. First the coefficients of the short-term synthesis filter $1/A(z)$ are determined for the frame by minimising the residual energy obtained when the input speech is passed through the inverse filter $A(z)$. Then for each sub-frame first the adaptive and then the fixed codebook parameters are calculated using a closed-loop approach. We give details of the procedures used for these three stages below.

## 3.3 The Short-Term Synthesis Filter

As discussed in Chapter 2 the vocal tract is responsible for altering the frequency spectrum of its excitation and producing the resonant, or formant, frequencies in speech. The short-term synthesis filter models this effect and introduces short-term correlations into the reproduced speech. In this section we discuss the form of this filter and methods of calculating and quantizing its coefficients.

For many sounds, especially non-nasal voiced sounds, the vocal tract can be modelled as an all-pole linear filter [4]. However when the glottis is opened the nasal tract is coupled to the vocal tract, and zeros are introduced into the speech spectrum. Thus nasal sounds tend to have spectral zeros, as do some non-nasal unvoiced sounds. Nevertheless, although some attempts have been made to use pole-zero models for the synthesis filter (see for example [35]), almost all codecs use a simple all-pole model. This is because of the ease with which the parameters of the all-pole filter can be calculated. Also if the number of poles is high enough even zeros in the speech spectrum can be approximately represented. Therefore in our work we have used only all-pole synthesis filters.

In forward adaptive systems the predictor coefficients are calculated from the input speech and sent to the decoder as side information. Initially the input speech is buffered and split into frames. Then for each frame a set of $p$ coefficients are found to produce an all-zero prediction filter $P(z)$ which, using the previous $p$ input speech samples $s(n-1)$, $s(n-2)\cdots s(n-p)$, predicts a value $\tilde{s}(n)$ for the present speech sample $s(n)$. $P(z)$ takes the form

$$P(z)=\sum_{i=1}^{p} a_i z^{-i} \tag{3.1}$$

and is related to the prediction error filter $A(z)$, which is also known as the inverse filter, by

$$A(z)=1-P(z)=1-\sum_{i=1}^{p} a_i z^{-i}. \tag{3.2}$$

The prediction filter coefficients $a_1, a_2\cdots a_p$ are chosen to minimise the energy of the error signal $e(n)=\tilde{s}-s(n)$. It is then *assumed* that these are the coefficients that best represent the spectrum of the speech in the all-pole synthesis filter $H(z)=1/A(z)$. We will consider the validity of this assumption in the next chapter.

The error energy $E$ is given by

$$\begin{aligned} E &= \sum_n e^2(n) \\ &= \sum_n (s(n)-\tilde{s}(n))^2 \\ &= \sum_n \left(s(n)-\sum_{i=1}^{p} a_i s(n-i)\right)^2. \end{aligned} \tag{3.3}$$

Our task is to calculate $a_i, a_2\cdots a_p$ so that $E$ is minimised. To do this we set $\partial E/\partial a_i = 0$ for $i=1,2\cdots p$ which leads to a set of $p$ simultaneous equations for the coefficients $a_i$:

$$\sum_{k=1}^{p} a_i \phi(i,k)=\phi(i,0) \quad \text{for i} = 1,2\cdots \text{p} \tag{3.4}$$

where

$$\phi(i,k)=\sum_n s(n-i)s(n-k). \tag{3.5}$$

To find the synthesis filter coefficients we must calculate the values $\phi(i,k)$ and use them to solve Equation 3.4. The exact form of the equations to be solved depends very much on the limits over which the residual error signal $e(n)$ is summed in Equation 3.3 when finding the error energy $E$ to be minimised. Two different approaches are possible – we can consider the input speech signal $s(n)$ to be of infinite duration and window the error signal $e(n)$ to a finite interval, or vice-versa. Windowing the input signal $s(n)$ to be non-zero only for $0 \leq n \leq L-1$, where $L$ is the analysis frame length, leads to the auto-correlation approach. Conversely if we sum the error signal $e(n)$ only over the range $0 \leq n \leq L-1$ in Equation 3.3 then we have the covariance approach.

Because the covariance method minimises the actual error signal $e(n)$ over the analysis frame length, and does not involve any modification of the input speech signal, it tends to lead to higher prediction gains than the autocorrelation approach. However when $L \gg p$, as is typically true for the short-term synthesis filter analysis, the differences are small. Also it can be shown [1] that the autocorrelation approach leads to a set of values for $\phi(i,k)$ such that $\phi$ is a symmetric Toeplitz matrix. This allows Equation 3.4 to be solved very efficiently [36] using the Levinson-Durbin algorithm, and also ensures [37] that the resulting all-pole synthesis filter will be stable. These are important advantages and so the autocorrelation method is commonly used in speech coding applications, and is the method we have used in our studies.

The vocal tract can be modelled as a series of uniform lossless acoustic tubes [4, 38]. It can then be shown that for a digital all-pole synthesis filter to approximate the effect of such a model of the vocal tract its delay should be at least twice the time required for sound waves to travel along the tract. For a vocal tract of length 17 cm and a sampling rate of 8 kHz this corresponds to the order $p$ of the filter being at least 8. Generally a few extra taps are added to help the filter cope with effects not allowed for in the lossless tube model, such as spectral zeros and losses in the vocal tract. We simulated the effect of changing the order $p$ on the prediction gain of the inverse filter $A(z)$. We used about eleven seconds of speech data obtained from two male and two female speakers. The speech was sampled at 8 kHz and split into 20 ms frames. For each frame the filter coefficients were calculated using the autocorrelation approach on the Hamming windowed speech data, and the prediction gain was calculated and converted into decibels. Here the prediction gain is defined as the energy of the original speech samples $s(n)$ divided by the energy of the prediction error samples $e(n)$. The overall prediction gain was taken as the average of the decibel gains for all the 20 ms frames in the speech file.

The results of our simulations are shown in Figure 3.2. Also shown in this figure is the variation of the segmental SNR of a CELP codec with the order $p$ of its synthesis filter. The filter coefficients were calculated for 20 ms frames as described above,

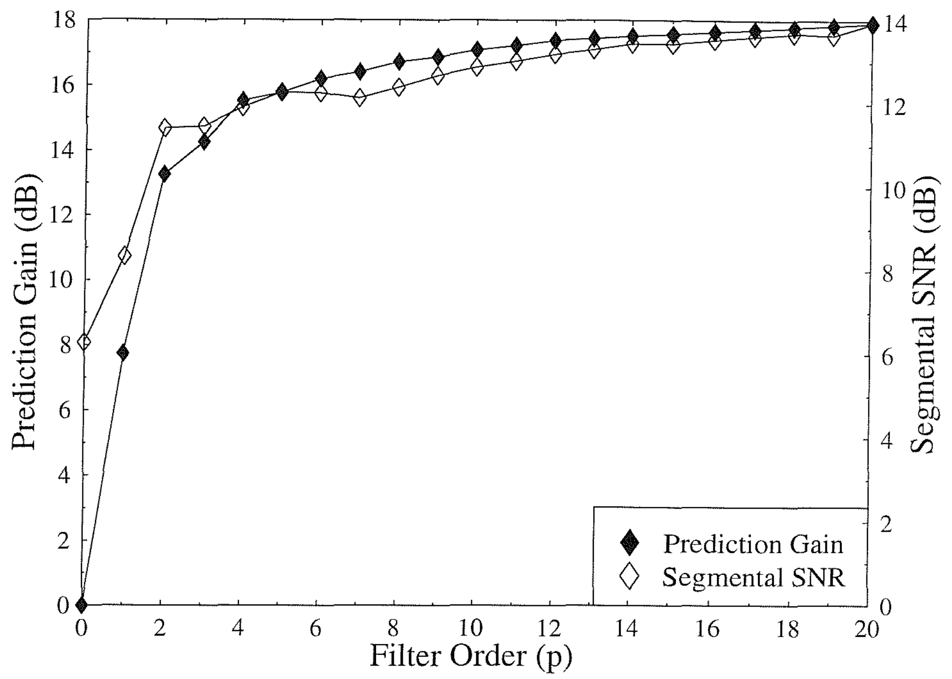

Figure 3.2: Variation of LPC Performance with Order p

and were left unquantized. The excitation parameters for the codec were determined identically to our 7.1 kbits/s codec as described later, except no error weighting was used. It can be seen that both the prediction gain of the inverse filter and the segmental SNR of the codec increase as the order of the synthesis filter is increased. However, in a forward adaptive system, each synthesis filter coefficient used requires side information to be sent to the decoder, and so we wish to keep their number to a minimum. We chose $p = 10$ as a sensible compromise between a high prediction gain and a low bit-rate.

The rate required to transmit information about the synthesis filter also depends on how often this information is updated, ie on the frame length $L$. We carried out similar simulations to those described above to see how the frame length affected the prediction gain of the inverse filter and the segmental SNR of a CELP codec. The order $p$ of the filter was fixed at $p = 10$ and the coefficients were calculated using Hamming windowed speech frames of length L samples. However the prediction gain and the

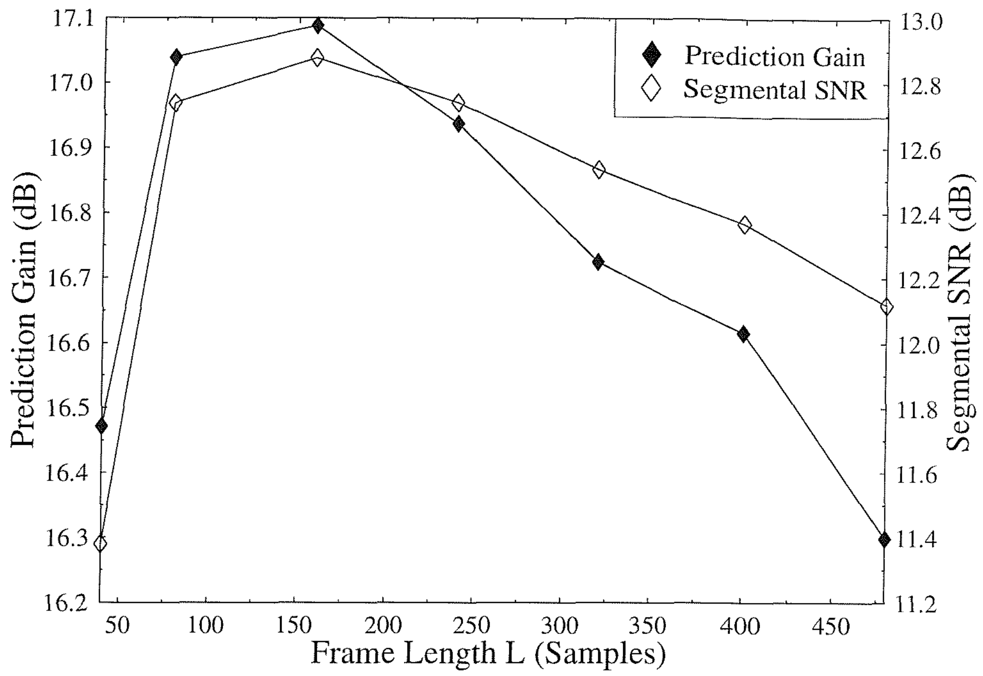

Figure 3.3: Variation of LPC Performance with Analysis Frame Length L

segmental SNR were calculated using frames 20 ms long to find the gains/SNRs which were converted into decibels and averaged. This was done to try and ensure a fair comparison within our results, which are shown in Figure 3.3. It can be seen that for very short analysis frame lengths both the prediction gain and the segmental SNR are well below the best values found. This is probably because we have used the autocorrelation method of analysis, and for small values of $L$ we do not have $L \gg p$ and so inaccuracies are introduced due to the windowing of the input speech signal. The best values of the prediction gain and the segmental SNR are given for $L = 160$, which corresponds to a 20 ms frame length. For larger size frames there is a gradual decrease in the performance of the filter due to the non-stationary nature of speech.

The synthesis filter coefficients must be quantized in order to be sent to the decoder. Unfortunately the filter coefficients themselves are not suitable for quantization because the frequency response of the synthesis filter is very sensitive to changes in them. This means even a small change in the values of the coefficients when they are quantized can lead to a large change in the spectrum of the synthesis filter. Also it is difficult to ensure that a given set of coefficients will produce a stable synthesis filter. Thus although the autocorrelation approach guarantees a stable filter, this stability could be easily lost through direct quantization of the filter coefficients. Therefore before quantization the coefficients are converted into another set of parameters from which they can be recovered but which are less sensitive to quantization noise and which allow stability to be easily guaranteed. Some schemes use reflection coefficients, which are related to the lossless tube model of the vocal tract and are calculated as a by-product of using the Levinson-Durbin algorithm to solve Equation 3.4. Using these coefficients the stability of the synthesis filter can be easily ensured by limiting the magnitude of all the coefficients to be less than one. Typically the reflection coefficients are transformed, using the inverse-sine transformation or log-area ratios, before quantization.

Another representation of the short-term filter coefficients, and the one we have used, is the set of Line Spectrum Pairs (LSPs) [39], or Line Spectral Frequencies (LSFs) [40]. These are derived as follows. A given inverse filter $A(z)$ of order $p$ can be arbitrarily extended to order $p+1$ by letting the '$p+1$'th reflection coefficient be $+1$ or $-1$. This corresponds to complete closure or opening at the '$p+1$'th stage in the acoustic tube model. The two resulting polynomials have all their zeros interlaced on the unit circle, and the frequencies of these zeros (between 0 and $\pi$) give the LSFs. It can be shown that the synthesis filter derived from a set of LSFs will be stable if the LSFs are ordered ie $\mathrm{LSF}_1 < \mathrm{LSF}_2 < \mathrm{LSF}_3$ etc. Thus the stability of the synthesis filter can be easily ensured when using LSFs. In our codecs the ten LSFs derived from the 10th order linear prediction filter are quantized with 34 bits using the scalar quantizer designed for use with the DoD 4.8 kbits/s codec [24]. This led, using the simulation conditions described earlier with $p = 10$ and $L = 160$, to the prediction gain for the inverse filter dropping from 17.1 dB to 16.7 dB and the segmental SNR of the codec dropping from 12.9 dB to 12.4 dB. Recently vector quantizers have been used to represent LSFs with fewer bits than are necessary for scalar quantizers (see for example [41]). Although we have not tried using such quantizers it should be possible to reduce the bit-rate of our codecs by around 0.5 kbits/s by using vector quantization. Such quantizers are capable of accurate representation of the LSFs and so the only penalty to be paid is an increase in the complexity of the encoder. This is a possible area for future work.

Generally the synthesis filter parameters are calculated by the encoder and transmitted to the decoder once per frame. We can however use some form of interpolation to update the filter coefficients every sub-frame, and minimise large changes in them from one frame to the next, without increasing how often they are transmitted. We use the scheme described in [1]. For our 7.1 kbits/s codec the frame length is 20 ms and each frame contains four sub-frames. However the LPC parameters are calculated using a Hamming window with an analysis frame length of 25 ms (or five sub-frames), so that the LPC windows for subsequent frames overlap by one sub-frame. The positioning of the LPC analysis windows are arranged so that the centre of the 4th sub-frame to be coded coincides with the centre of the Hamming window, and so the quantized LSFs are used directly in this sub-frame. The LSFs for the other three sub-frames are found using linear interpolation between the present set of quantized LSFs and the previous set. This interpolation led, using the simulation conditions described earlier with the ten LSFs quantized and an update rate of 20 ms (160 samples), to the prediction gain for the inverse filter increasing from 16.7 dB to 17.0 dB and the segmental SNR of our 7.1 kbits/s codec rising from 12.4 dB to 12.9 dB. A similar arrangement is used for the 4.7 kbits/s codec – the frame length is 30 ms with four 7.5 ms sub-frames, but the LPC analysis window is 37.5 ms long.

## 3.4 The Error Weighting Filter

CELP codecs choose the excitation signal for the synthesis filter in a closed-loop manner with the aim of minimising the error between the original and the reconstructed speech. The theory of auditory masking suggests that the noise in the formant regions (where the speech has high energy) can be partially or totally masked by the speech signal. Therefore it makes sense to try and concentrate the error energy in these formant regions. This is the function of error weighting filters in AbS codecs.

These error weighting filters should emphasise noise in the frequency regions where the speech has low energy, and de-emphasise the noise in the formant regions. The most commonly used filter, and the one we have used in our work with high delay codecs, is of the form

$$W(z) = \frac{A(z)}{A(z/\gamma)} \tag{3.6}$$

where $A(z)$ is the LPC inverse filter and $\gamma$ is a bandwidth expansion factor between 0 and 1. We used $\gamma = 0.8$ for our 7.1 kbits/s codec, and $\gamma = 0.9$ for the 4.7 kbits/s codec. This error weighting produced a significant increase in the subjective quality of the reconstructed speech, at the expense of a decrease in the codec's segmental SNR. For example for our 7.1 kbits/s codec the segmental SNR is 12.9 dB with no weighting, but drops to 12.1 dB with error weighting.

Recently other forms of error weighting have been suggested for speech codecs. For example in the CCITT 16 kbits/s [3] codec a filter

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} \tag{3.7}$$

is used where $\gamma_1 = 0.9$ and $\gamma_2 = 0.6$. We used this weighting filter in our work with low delay codecs in Chapters 6 and 7. In [13] an explicit auditory model is used to try and take account of the details known about psychoacoustics and masking.

## 3.5 The Adaptive Codebook Search

The excitation signal $u(n)$ is determined every sub-frame, ie four times a frame, and is chosen to minimise the mean square weighted error $E_w$ over the sub-frame, where

$$E_w = \frac{1}{N} \sum_{n=0}^{N-1} e_w^2(n) \tag{3.8}$$

and $N$ is the number of samples per sub-frame. In the coders we have simulated the sub-frames are 5 ms long for the 7.1 kbits/s codec and 7.5 ms long for the 4.7 kbits/s codec. Therefore $N$ is 60 for the low-rate codec and 40 for the high-rate codec.

The excitation is given by

$$u(n) = v(n) + G_1 u(n - \alpha) \tag{3.9}$$

where $G_1$ and $\alpha$ are the adaptive codebook gain and delay and $v(n)$ is the fixed codebook signal. The adaptive codebook models the long term periodicity, ie the pitch, of the speech, and produces an output which is a scaled version of some previous excitation. It is equivalent to a one tap predictor with v(n) as its input, u(n) as its output and a transfer function $L(Z) = \frac{U(z)}{V(z)} = \frac{1}{1-G_1 z^{-\alpha}}$ .

In an optimal coder the fixed codebook and the adaptive codebook parameters would all be optimized together in order to minimize $E_w$. However in practice this is not usually done due to the complexity it would involve. Instead a sub-optimal approach is used and the adaptive codebook parameters are determined first, by assuming that the fixed codebook signal $v(n)$ is zero. Then

$$u(n) = G_1 u(n-\alpha). \tag{3.10}$$

In every sub-frame $\alpha$ and $G_1$ are chosen to minimize the weighted error $e_w(n)$. This is given by

$$\begin{aligned} e_w(n) &= s_w(n) - \hat{s}_w(n) \\ &= s_w(n) - (h(n) * u(n) + \hat{s}_o(n)) \\ &= s_w(n) - \left(\sum_{i=0}^{n} u(i)h(n-i) + \hat{s}_o(n)\right) \end{aligned} \tag{3.11}$$

where $h(n)$ is the impulse response of the weighted synthesis filter $1/A(z/\gamma)$ and $\hat{s}_o(n)$ is the zero input response of the filter due to its memory of the input in the previous sub-frame. Substituting Equation 3.10 into Equation 3.11 gives

$$\begin{aligned} e_w(n) &= s_w(n) - \left(G_1 \sum_{i=0}^{n} u(i-\alpha)h(n-i) + \hat{s}_o(n)\right) \\ &= x(n) - G_1 y_\alpha(n) \end{aligned} \tag{3.12}$$

where

$$x(n) = s_w(n) - \hat{s}_o(n) \tag{3.13}$$

is the target for the adaptive codebook search, and

$$y_\alpha(n) = \sum_{i=0}^{n} u(i-\alpha)h(n-i) \tag{3.14}$$

is the convolution of the adaptive codebook signal $u(n-\alpha)$ with the impulse response of the weighted synthesis filter. Thus, ignoring the fixed codebook contribution, we can write the weighted mean square error as

$$E_w = \frac{1}{N}\sum_{n=0}^{N-1} \left(x(n) - G_1 y_\alpha(n)\right)^2. \tag{3.15}$$

Setting $\partial E_w / \partial G_1 = 0$ gives the optimum gain $G_1$ for a given delay $\alpha$ as

$$G_1 = \frac{\sum_{n=0}^{N-1} x(n) y_\alpha(n)}{\sum_{n=0}^{N-1} y_\alpha^2(n)}, \tag{3.16}$$

and substituting this into Equation 3.15 gives the minimum mean square weighted error for a given delay as

$$\begin{aligned} E_w &= \frac{1}{N}\left(\sum_{n=0}^{N-1} x^2(n) - \frac{\left(\sum_{n=0}^{N-1} x(n) y_\alpha(n)\right)^2}{\sum_{n=0}^{N-1} y_\alpha^2(n)}\right) \\ &= \frac{1}{N}\left(\sum_{n=0}^{N-1} x^2(n) - X\right) \end{aligned} \tag{3.17}$$

where

$$X = \frac{\left(\sum_{n=0}^{N-1} x(n) y_\alpha(n)\right)^2}{\sum_{n=0}^{N-1} y_\alpha^2(n)}. \tag{3.18}$$

To find the optimum adaptive codebook parameters we calculate the value of $X$ for all possible delays $\alpha$, and the delay which maximises $X$ is chosen. The corresponding gain is then given by Equation 3.16.

Note that the past excitation signal $u(n-\alpha)$ is only available for $n-\alpha < 0$. When $n - \alpha > 0$ the 'past excitation' is part of the excitation for the current sub-frame and so is not yet known. Therefore for delays less than the sub-frame length $N$ only the first $\alpha$ values of $u(n-\alpha)$ are available. We make up the rest of the values by repeating the available pattern, ie taking $u(n-2\alpha)$ for $\alpha < n < 2\alpha - 1$ etc, until the range $0 \leq n \leq N-1$ has been covered.

The computational load required to calculate the convolution $y_\alpha(n)$ for all possible values of the delay $\alpha$ would be large if they were all calculated independently. Fortunately this can be avoided by calculating the convolution for the lowest value of $\alpha$ and then using an iterative procedure to find $y_\alpha(n)$ for all the other necessary values of $\alpha$ [1]. This iterative procedure is possible because the adaptive codebook codeword for a delay $\alpha$ is merely the codeword for the delay $\alpha - 1$ shifted by one sample, with one new value $u(-\alpha)$ introduced, and one old value $u\,(N-\alpha)$ discarded. This is true except for delays less than the sub-frame length $N$, for which the iterative procedure is complicated slightly because of the repetition described above used to make up the codewords.

In our codecs the delay can take any integral value from 20 to 147, and so is represented with seven bits. The gain $G_1$ is non-uniformly quantized with three bits, giving a total of ten bits per sub-frame for the adaptive codebook information.

## 3.6 Fixed Codebook Search

In the final stage of its calculations the coder finds the fixed codebook index and gain which minimise $E_w$. Taking the fixed codebook contribution (which was ignored in the last section) into account the weighted error signal is given by

$$\begin{aligned} e_w(n) &= s_w(n) - \hat{s}_w(n) \\ &= s_w(n) - (\hat{s}_o(n) + G_1 y_\alpha(n)) - G_2 c_k(n) * h(n) \\ &= \tilde{x}(n) - G_2 c_k(n) * h(n) \end{aligned} \tag{3.19}$$

where

$$\tilde{x}(n) = s_w(n) - \hat{s}_o(n) - G_1 y_\alpha(n) \tag{3.20}$$

is the target for the fixed codebook search, $c_k(n)$ is the codeword from the fixed codebook and $G_2$ is the fixed codebook gain. Thus

$$\begin{aligned} E_w &= \frac{1}{N}\sum_{n=0}^{N-1} e_w^2(n) \\ &= \frac{1}{N}\sum_{n=0}^{N-1} (\tilde{x}(n) - G_2[c_k(n) * h(n)])^2 . \end{aligned} \tag{3.21}$$

Setting $\partial E_w / \partial G_2 = 0$ gives the optimum gain for a given codeword $c_k(n)$ as

$$\begin{aligned} G_2 &= \frac{\sum_{n=0}^{N-1} \tilde{x}(n)[c_k(n) * h(n)]}{\sum_{n=0}^{N-1}[c_k(n) * h(n)]^2} \\ &= \frac{\tilde{C}_k}{\xi_k} \end{aligned} \tag{3.22}$$

where

$$\tilde{C}_k = \sum_{n=0}^{N-1} \tilde{x}(n)[c_k(n) * h(n)] \tag{3.23}$$

and

$$\xi_k = \sum_{n=0}^{N-1} [c_k(n) * h(n)]^2. \tag{3.24}$$

Physically $\xi_k$ is the energy of the filtered codeword, and $\tilde{C}_k$ is the correlation between the target signal $\tilde{x}(n)$ and the filtered codeword. In the search for the fixed codebook parameters the values of $\xi_k$ and $\tilde{C}_k$ are calculated for every codeword $k$ and the optimum gain for that codeword is calculated using Equation 3.22. The gain is quantized to give $\hat{G}_2$ which is substituted back into Equation 3.21 to give

$$\begin{aligned} E_w &= \frac{1}{N}\sum_{n=0}^{N-1}\left(\tilde{x}(n) - \hat{G}_2[c_k(n) * h(n)]\right)^2 \\ &= \frac{1}{N}\left(\sum_{n=0}^{N-1}\tilde{x}^2(n) - 2\hat{G}_2\sum_{n=0}^{N-1}\tilde{x}(n)[c_k(n) * h(n)] + \hat{G}_2^2\sum_{n=0}^{N-1}[c_k(n) * h(n)]^2\right) \\ &= \frac{1}{N}\left(\sum_{n=0}^{N-1}\tilde{x}^2(n) - 2\hat{G}_2\tilde{C}_k + \hat{G}_2^2\xi_k\right). \end{aligned} \tag{3.25}$$

The term $T_k = \hat{G}_2(2\tilde{C}_k - \hat{G}_2\xi_k)$ is calculated for every codeword, and the index which maximises it is chosen. This index along with the quantized gain is then sent to the decoder.

Traditionally the major part of a CELP coder's complexity comes from calculating the correlation $\tilde{C}_k$ and energy $\xi_k$ for every codebook entry. From Equations 3.23 and 3.24 these are given by

$$\begin{aligned} \tilde{C}_k &= \sum_{n=0}^{N-1}\tilde{x}(n)[c_k(n) * h(n)] \\ &= \sum_{n=0}^{N-1}\psi(n)c_k(n) \end{aligned} \tag{3.26}$$

and

$$\begin{aligned} \xi_k &= \sum_{n=0}^{N-1}[c_k(n) * h(n)]^2 \\ &= \sum_{i=0}^{N-1}c_k^2(i)\phi(i,i) + 2\sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1}c_k(i)c_k(j)\phi(i,j) \end{aligned} \tag{3.27}$$

where

$$\begin{aligned} \psi(i) &= \tilde{x}(i) * h(-i) \\ &= \sum_{n=i}^{N-1}\tilde{x}(n)h(n-i) \quad \text{For } i = 0\cdots N-1 \end{aligned} \tag{3.28}$$

and

$$\phi(i,j) = \sum_{n=max(i,j)}^{N-1} h(n-i)h(n-j) \quad \text{For } i,j = 0\cdots N-1. \tag{3.29}$$

The functions $\psi(i)$ and $\phi(i,j)$ can be calculated once per sub-frame, but then $\xi_k$ and $\tilde{C}_k$ must be calculated for each codeword. This involves a large number of additions and

| Pulse Number i | Amplitude | Possible Position $m_i$ |
|---|---|---|
| 0 | +1 | 0,8,16,24,32,40,48,56 |
| 1 | -1 | 2,10,18,26,34,42,50,58 |
| 2 | +1 | 4,12,20,28,36,44,52 |
| 3 | -1 | 6,14,22,30,38,46,54 |

Table 3.1: Pulse Amplitudes and Positions for the 4.7 kbits/s Codec

| Pulse Number i | Amplitude | Possible Position $m_i$ |
|---|---|---|
| 0 | +1 | 1,6,11,16,21,26,31,36 |
| 1 | -1 | 2,7,12,17,22,27,32,37 |
| 2 | +1 | 3,8,13,18,23,28,33,38 |
| 3 | -1 | 4,9,14,19,24,29,34,39 |

Table 3.2: Pulse Amplitudes and Positions for the 7.1 kbits/s Codec

multiplications by the elements of $c_k(n)$. Several schemes, for example binary pulse excitation [42], have been proposed to simplify these calculations by using codebooks where most of the entries $c_k(n)$ are zero, thus greatly reducing the number of additions necessary to find $\xi_k$ and $\tilde{C}_k$. Also if the non-zero elements of the codebook are equal to +1 or -1 then no multiplications are necessary and $\tilde{C}_k$ and $\xi_k$ can be calculated by a series of additions and subtractions.

In our codecs we use the algebraic codebook structure which was originally proposed in [43]. Each codeword $c_k(n)$ has only 4 non-zero pulses, which have amplitudes of either +1 or -1. Also each non-zero pulse has a limited number of positions within the codeword where it can lie. The amplitudes and possible positions within the codeword for each of the four pulses are shown in Table 3.1 for our sub-frame size 60 4.7 kbits/s codec, and in Table 3.2 for our sub-frame size 40 7.1 kbits/s codec. In both codecs each pulse can take up eight positions, and so the chosen positions can be represented with three bits each, giving a total of twelve bits per sub-frame to represent the codebook index. The gain sign is represented with one bit and its magnitude is quantized with four bits using logarithmic quantization. This gives a total of 17 bits per sub-frame for the fixed codebook information.

The algebraic codebook structure has several advantages – it does not require any storage and is robust to channel errors. Most importantly it allows the values $\tilde{C}_k$ and $\xi_k$ to be calculated very efficiently. From Equations 3.26 and 3.27

$$\tilde{C}_k = \psi(m_0) - \psi(m_1) + \psi(m_2) - \psi(m_3) \tag{3.30}$$

| | |
|---|---|
| Line Spectrum Frequencies | 34 |
| Adaptive Codebook Delays | 28 (4*7) |
| Adaptive Codebook Gains $G_1$ | 12 (4*3) |
| Fixed Codebook Index $k$ | 48 (4*12) |
| Fixed Codebook Gains $G_2$ | 20 (4*5) |
| Total | 142 |

Table 3.3: Bits Allocated per Frame

and

$$\begin{aligned}\xi_k &= \phi(m_0, m_0)\\ &+ \phi(m_1, m_1) - 2\phi(m_1, m_0)\\ &+ \phi(m_2, m_2) + 2\phi(m_2, m_0) - 2\phi(m_2, m_1)\\ &+ \phi(m_3, m_3) - 2\phi(m_3, m_0) + 2\phi(m_3, m_1) - 2\phi(m_3, m_2) \qquad (3.31)\end{aligned}$$

where $m_i$ is the position of the pulse number $i$. By changing only one pulse position at a time $\tilde{C}_k$ and $\xi_k$ can be calculated using four nested loops. In the inner loop $\tilde{C}_k$ is updated with one addition, and $\xi_k$ with three multiplications and four additions. This allows for a very efficient codebook search.

## 3.7 Decoder Structure

The previous four sections have described the structure of our encoder. In the decoder the codebook information transmitted from the encoder is used to find an excitation signal $u(n)$. If there are no channel errors this will be identical to the excitation signal $u(n)$ in the encoder. It is then passed through a synthesis filter $1/A(z)$ to give the reconstructed speech signal $\hat{s}(n)$ as shown in Figure 3.1. The parameters of the synthesis filter are determined from the line spectrum frequencies transmitted from the coder, using interpolation as described in Section 3.3.

The number of bits allocated per frame for the various pieces of information to transmitted from the encoder to the decoder is summarized in Table 3.3. A total of 142 bits per frame are transmitted, which with a 20 ms frame length gives a rate of 7.1 kbits/s, and with a 30 ms frame length gives a rate of just over 4.7 kbits/s.

| | |
|---|---|
| CELP Codebook Search | 300,000 |
| ACELP Codebook Search | 15 |
| LPC Analysis | 0.75 |
| Adaptive Codebook Search | 7 |

Table 3.4: Encoder Complexity (MFLOPs)

## 3.8 Conclusion

In this chapter we have described in detail the general framework of CELP codecs, and a particular codebook structure which allows an efficient codebook search. Table 3.4 shows the approximate complexity, in terms of millions of floating point operations per second (MFLOPs), of the various stages of the encoding procedure for the 7.1 kbits/s codec. Also shown is the complexity for a non-sparse 12-bit fixed codebook search. As can be seen the fixed codebook search accounts for the majority of the complexity in the encoder, and the algebraic codebook structure gives a huge reduction in this complexity. In total the encoding procedure we have described requires approximately 23 MFLOPs, with most operations being spent on the two codebook searches. The decoder does not have to do any codebook searches but merely filters the selected excitation through the synthesis filter. As a result it is much less complex and requires only about 0.2 MFLOPs.

The two codecs described here were tested with the speech file described earlier. The 4.7 kbits/s codec produced good communications quality speech with a segmental SNR of 10.5 dB while the 7.1 kbits/s codec produced speech which was noticeably more transparent and had a segmental SNR of 12.1 dB.

# Chapter 4

# Optimization of the CELP Codec Parameters

## 4.1 Introduction

In the previous chapter we discussed the general structure of CELP codecs. This largely closed-loop structure is used in order to produce reconstructed speech which is as close as possible to the original speech. However there are two exceptions to an entirely closed-loop approach which are used in most CELP codecs. The first is in the determination of the synthesis filter $H(z)$, which is simply assumed to be the inverse of the short-term linear prediction error filter $A(z)$ which minimises the energy of the prediction residual. This means that although the excitation signal $u(n)$ is derived taking into account the form of the synthesis filter, no account is taken of the form of the excitation signal when the synthesis filter parameters are determined. This seems like an obvious deficiency, and means for example that the synthesis filter may try to take account of long-term periodicities which would be better left to the adaptive codebook to deal with.

The second departure from a strict closed-loop approach in most CELP codecs is in the determination of the codebook parameters. Rather than the adaptive and fixed codebook parameters being determined together to produce an overall minimum in the weighted error signal, the adaptive codebook delay and gain are determined first by assuming that the fixed codebook signal is zero. Then, given the adaptive codebook signal, the fixed codebook parameters are found. This approach is taken in order to reduce the complexity of CELP codecs to a reasonable level. However it seems obvious that it must lead to some degradation in the reconstructed speech quality.

In this chapter we discuss ways of overcoming the two exceptions to the closed-loop approach described above, and try to improve the quality of the reconstructed speech from our codecs while maintaining a reasonable level of complexity. We have concentrated our studies on the 4.7 kbits/s forward adaptive ACELP codec described in the previous chapter, although the techniques described will be applicable to other AbS codecs.

## 4.2 Calculation of the Excitation Parameters

In this section we discuss the procedure traditionally used for the adaptive and fixed codebook searches in CELP codecs, and ways in which this procedure can be improved. First the theory behind a full search procedure is given. Then we describe how the equations derived for a full search reduce to those in Sections 3.5 and 3.6 derived for the usual sequential determination of the codebook parameters. In Section 4.2.3 we describe the full search procedure, its complexity and the results it gives. Section 4.2.4 describes various sub-optimal approaches which can be used, and finally Section 4.2.5 describes the quantization of the codebook gains.

### 4.2.1 Full Codebook Search Theory

Consider the weighted error $e_w(n)$ between the weighted input speech and the weighted reconstructed speech. This is given by

$$\begin{aligned} e_w(n) &= s_w(n) - \hat{s}_w(n) \qquad (4.1) \\ &= s_w(n) - \hat{s}_o(n) - G_1 y_\alpha(n) - G_2[c_k(n) * h(n)] \end{aligned}$$

where the symbols used here have the same meaning as in Chapter 3. To recap $s_w(n)$ is the weighted input speech, $\hat{s}_o(n)$ is the zero input response of the weighted synthesis filter due to its input in previous sub-frames, $G_1$ is the adaptive codebook gain, $y_\alpha(n) = h(n) * u(n-\alpha)$ is the filtered adaptive codebook signal, $G_2$ is the fixed codebook gain, $c_k(n)$ is the fixed codebook codeword and $h(n)$ is the impulse response of the weighted synthesis filter.

The search procedure attempts to find the values of the adaptive codebook gain $G_1$ and delay $\alpha$ and the fixed codebook index $k$ and gain $G_2$ which minimise the mean square error $E_w$ taken over the sub-frame length $N$. This is given by

$$\begin{aligned}
E_w &= \frac{1}{N}\sum_{n=0}^{N-1} e_w^2(n) \\
&= \frac{1}{N}\sum_{n=0}^{N-1} \left(x(n) - G_1 y_\alpha(n) - G_2[c_k(n) * h(n)]\right)^2 \\
&= \frac{1}{N}\left(\sum_{n=0}^{N-1} x^2(n) + G_1^2 \sum_{n=0}^{N-1} y_\alpha^2(n) + G_2^2 \sum_{n=0}^{N-1}[c_k(n) * h(n)]^2 - 2G_1 \sum_{n=0}^{N-1} x(n) y_\alpha(n)\right. \\
&\quad \left. -2G_2 \sum_{n=0}^{N-1} x(n)[c_k(n) * h(n)] + 2G_1 G_2 \sum_{n=0}^{N-1} y_\alpha(n)[c_k(n) * h(n)]\right) \qquad (4.2)
\end{aligned}$$

where $x(n) = s_w(n) - \hat{s}_o(n)$ is the target signal for the codebook search, referred to as the LTP target in Section 3.5. We can rewrite this formula as

$$\begin{aligned}
E_w &= \frac{1}{N}\left(\sum_{n=0}^{N=1} x^2(n) + G_1^2\xi_\alpha + G_2^2\xi_k - 2G_1C_\alpha - 2G_2C_k + 2G_1G_2Y_{\alpha k}\right) \\
&= \frac{1}{N}\left(\sum_{n=0}^{N-1} x^2(n) - T_{\alpha k}\right) \qquad (4.3)
\end{aligned}$$

where

$$T_{\alpha k} = 2\left(G_1C_\alpha + G_2C_k - G_1G_2Y_{\alpha k}\right) - G_1^2\xi_\alpha - G_2^2\xi_k \qquad (4.4)$$

is the term to be maximised by the codebook search. Here

$$\xi_\alpha = \sum_{n=0}^{N-1} y_\alpha^2(n) \qquad (4.5)$$

is the energy of the filtered adaptive codebook signal and

$$C_\alpha = \sum_{n=0}^{N-1} x(n) y_\alpha(n) \qquad (4.6)$$

is the correlation between the filtered adaptive codebook signal and the codebook target $x(n)$. Similarly $\xi_k$ is the energy of the filtered fixed codebook signal $[c_k(n)*h(n)]$, and $C_k$ is the correlation between this and the target signal. Finally

$$Y_{\alpha k} = \sum_{n=0}^{N-1} y_\alpha(n)[c_k(n) * h(n)] \qquad (4.7)$$

is the correlation between the filtered signals from the two codebooks. With this notation we have tried to emphasis what codebook the variables are dependent on.

For example, once the weighted synthesis filter parameters are known, $\xi_\alpha$ depends only on which delay $\alpha$ is chosen for the adaptive codebook, whereas $Y_{\alpha k}$ depends on the indices $\alpha$ and $k$ used for both the adaptive and fixed codebooks.

The codebook search must find the values of the indices $\alpha$ and $k$, and the gains $G_1$ and $G_2$, which maximise $T_{\alpha k}$ and so minimise $E_w$. For a given pair of indices $\alpha$ and $k$ we can find the optimum values for $G_1$ and $G_2$ by setting the partial derivatives of $T_{\alpha k}$ with respect to $G_1$ and $G_2$ to zero. This gives

$$\frac{\partial T_{\alpha k}}{\partial G_1} = 2C_\alpha - 2G_2 Y_{\alpha k} - 2G_1 \xi_\alpha = 0 \tag{4.8}$$

and

$$\frac{\partial T_{\alpha k}}{\partial G_2} = 2C_k - 2G_1 Y_{\alpha k} - 2G_2 \xi_k = 0. \tag{4.9}$$

Solution of these two linear simultaneous equations gives the optimum values of the gains, for given codebook indices, as

$$G_1 = \frac{C_\alpha \xi_k - C_k Y_{\alpha k}}{\xi_\alpha \xi_k - Y_{\alpha k}^2} \tag{4.10}$$

and

$$G_2 = \frac{C_k \xi_\alpha - C_\alpha Y_{\alpha k}}{\xi_\alpha \xi_k - Y_{\alpha k}^2}. \tag{4.11}$$

The full search procedure must for every pair of codebook indices $\alpha$, $k$ find the terms $\xi_\alpha$ $\xi_k$ $C_\alpha$ $C_k$ and $Y_{\alpha k}$, and use these to calculate the gains $G_1$ and $G_2$. These gains can then be quantized and substituted into Equation 4.4 to give $T_{\alpha k}$ which the coder must maximise by the proper choice of $\alpha$ and $k$.

### 4.2.2 Sequential Search Procedure

In this section we discuss how the equations derived above relate to those in Sections 3.5 and 3.6 for the sequential search procedure which is usually employed in CELP codecs. In this sequential search the adaptive codebook parameters are determined first by assuming $G_2 = 0$. Substitution of this into Equation 4.8 gives

$$G_1 = \frac{C_\alpha}{\xi_\alpha} = \frac{\sum_{n=0}^{N-1} x(n) y_\alpha(n)}{\sum_{n=0}^{N-1} y_\alpha^2(n)} \tag{4.12}$$

as in Equation 3.16. If we then substitute the values $G_1 = C_\alpha/\xi_\alpha$ and $G_2 = 0$ into Equation 4.4 the term to be maximised becomes

$$T_{\alpha k} = \frac{C_\alpha^2}{\xi_\alpha} = \frac{\left(\sum_{n=0}^{N-1} x(n) y_\alpha(n)\right)^2}{\sum_{n=0}^{N-1} y_\alpha^2(n)} \tag{4.13}$$

as in Equation 3.18.

Once the adaptive codebook parameters have been determined they are assumed constant during the fixed codebook search. The LTP target $x(n)$ is updated to give the fixed codebook target $\tilde{x}(n)$ where

$$\tilde{x}(n) = x(n) - G_1 y_\alpha(n), \tag{4.14}$$

and for each codebook index $k$ the energy $\xi_k$ and the correlation $\tilde{C}_k$ between $\tilde{x}(n)$ and the filtered codewords are found. The correlation term $\tilde{C}_k$ is given by

$$\begin{aligned} \tilde{C}_k &= \sum_{n=0}^{N-1} \tilde{x}(n)[c_k(n) * h(n)] \\ &= \sum_{n=0}^{N-1} (x(n) - G_1 y_\alpha(n))\,[c_k(n) * h(n)] \\ &= C_k - G_1 Y_{\alpha k}. \end{aligned} \tag{4.15}$$

Substitution of this into Equation 4.9 gives

$$G_2 = \frac{\tilde{C}_k}{\xi_k} \tag{4.16}$$

as in Equation 3.22, and the term to be maximised becomes

$$\begin{aligned} T_{\alpha k} &= 2G_1 C_\alpha + 2G_2(C_k - G_1 Y_{\alpha k}) - G_1^2 \xi_\alpha - G_2^2 \xi_K \\ &= 2G_1 C_\alpha - G_1^2 \xi_\alpha + 2G_2 \tilde{C}_k - G_2^2 \xi_k. \end{aligned} \tag{4.17}$$

Now as $G_1$ and $\alpha$ are fixed we can ignore the first two terms above and write the expression to be maximised by the fixed codebook search as $G_2(2\tilde{C}_k - G_2\xi_k)$, as in Section 3.6.

### 4.2.3 Full Search Procedure

We describe here the procedure used to perform a full codebook search to find the minimum possible weighted error $E_w$. Although such a full search is not a practical method for use in real speech coders, it does give us an upper bound to the improvements which can be obtained over the sequential search approach.

In order to perform a full search of the two codebooks the coder must calculate the value of $T_{\alpha k}$ using Equation 4.4 for every possible pair of codebook indices $\alpha$ and $k$, and select the indices which maximise $T_{\alpha k}$. This means we must calculate $\xi_\alpha$ and $C_\alpha$

for every adaptive codebook codeword, $\xi_k$ and $C_k$ for every fixed codebook codeword, and $Y_{\alpha k}$ for every pair of codewords. All the necessary values of $C_\alpha$, $\xi_\alpha$, $C_k$ and $\xi_k$ are calculated in the normal sequential search procedure. The extra complexity of the full search comes from calculating $Y_{\alpha k}$ for all values of $\alpha$ and $k$.

Using a similar approach to that used to find $\tilde{C}_k$ in the normal search, $Y_{\alpha k}$ can be written as

$$\begin{aligned} Y_{\alpha k} &= \sum_{n=0}^{N-1} y_\alpha(n)[c_k(n) * h(n)] \\ &= \sum_{n=0}^{N-1} c_k(n)[y_\alpha(n) * h(-n)] \\ &= \sum_{n=0}^{N-1} c_k(n)\Omega_\alpha(n) \end{aligned} \tag{4.18}$$

where $\Omega_\alpha(n)$ is given by

$$\Omega_\alpha(n) = \sum_{i=n}^{N-1} y_\alpha(i)h(i-n). \tag{4.19}$$

Thus, once $\Omega_\alpha(n)$ is known, using the algebraic codebook structure allows $Y_{\alpha k}$ to be calculated using four additions for each fixed codebook index $k$. Using four nested loops and updating the position of one pulse only in each loop allows us to find $Y_{\alpha k}$ very efficiently. Also because of the nature of the filtered adaptive codebook signal $y_\alpha(n)$ we can find $\Omega_\alpha(n)$ efficiently using an iterative procedure.

We simulated a full search codec in order to see what degradation the sequential approach gave compared to the ideal full search. We measured the performance of the codec using the segmental SNR and the weighted SNR measures as defined in Chapter 2. The delay $\alpha$ of the adaptive codebook was allowed to take any integer value between 20 and 147, and a twelve bit algebraic fixed codebook was used as described in Section 3.6. We found that quantizing the codebook gains with quantizers designed for the normal codec masked the improvements obtained with the full search. Therefore for all our simulation results reported here and in the next section neither $G_1$ nor $G_2$ were quantized. We consider quantization of the gains in Section 4.2.5.

We found, for four speech-files containing speech from two male and two female speakers, the full search procedure improved the average segmental SNR of our 4.7 kbits/s ACELP codec from 9.7 dB to 10.8 dB. A similar improvement was seen in the average weighted SNR – it increased from 7.3 dB to 8.2 dB. The reconstructed speech using the full search procedure sounded more full and natural than that obtained using the sequential search procedure.

However these gains are obtained only at the expense of a huge increase in the complexity of the codec. Even with the techniques described above to allow the full search to be carried out efficiently, such a codec is almost sixty times more computationally demanding than a codec using the standard approach. Therefore in the next Section we describe some sub-optimal approaches to the codebook search, with the aim of keeping the improvement in the reconstructed speech quality we have seen with the full codebook search, but reducing the complexity of the search to a reasonable level.

### 4.2.4 Sub-Optimal Search Procedures

The full search procedure described in the previous section allows us to find the best combination of the codebook indices $\alpha$ and $k$. However this method is unrealistically complex, and in this section we describe some sub-optimal search strategies.

One such search procedure, which we refer to here as "Method A", is to follow the sequential approach and find $G_1$ and $\alpha$ by assuming $G_2 = 0$, and then find $G_2$ and $k$ while assuming $G_1$ and $\alpha$ are fixed. Then once $\alpha$ and $k$ have been determined we can use Equations 4.10 and 4.11 to jointly optimize the values of the codebook gains. To do this we need to know $C_\alpha$, $\xi_\alpha$, $C_k$, $\xi_k$ and $Y_{\alpha k}$ for the chosen indices. The values of $C_\alpha$, $\xi_\alpha$ and $\xi_k$ will be known from the codebook searches, and $C_k$ can be found from $Y_{\alpha k}$ and $\tilde{C}_k$ using Equation 4.15. The main computational requirement for the update of the gains is therefore the calculation of $Y_{\alpha k}$ for the given $\alpha$ and $k$, and this is relatively undemanding. In fact updating of the codebook gains given the codebook indices increases the complexity of the codec by about only two percent. Using the same speech-files described earlier we found this update of the gains increased the average segmental SNR of the codec from 9.7 dB to 10.1 dB, and the average weighted SNR from 7.3 dB to 7.5 dB.

Another possible sub-optimal approach to the codebook searches is to find the adaptive codebook delay $\alpha$ using the usual approach (ie by assuming $G_2 = 0$), and then use only this value of $\alpha$ during the fixed codebook search in which $G_1$, $G_2$ and $k$ are all determined. This is similar to an approach suggested in [44] where a very small (32 entries) fixed codebook was used, and a one tap IIR filter was used instead of the adaptive codebook. For our codec we find $\xi_k$, $C_k$ and $Y_{\alpha k}$ for every fixed codebook index $k$ using the approach with four nested loops described in Sections 3.6 and 4.2.3. The values of $\xi_\alpha$ and $C_\alpha$ are known from the adaptive codebook search, and so we can use Equations 4.10 and 4.11 to find $G_1$ and $G_2$, and then calculate $T_{\alpha k}$ using Equation 4.4. The value of $k$ which maximises $T_{\alpha k}$ is chosen as the fixed codebook index. We refer to this joint codebook search procedure as "Method B".

This Method B search allows the fixed codebook entry to be selected taking full account of the possible variations in the magnitude of the adaptive codebook signal. If we could trust the initial value of $\alpha$ calculated assuming $G_2 = 0$ to be correct, then it would give identical results to the full search procedure. However it is much less computationally demanding than the full codebook search, and increases the complexity of the normal codec by only about 30%. In our simulations we found that it increased the average segmental SNR from 9.7 dB to 10.3 dB. Similarly the average weighted SNR increased from 7.3 dB to 7.8 dB. Thus this approach gives significant gains over the normal sequential search, but still does not match the results of the codec using the full search procedure.

The differences between the results using the full codebook search, and those described above, must be due to differences in the adaptive codebook delay $\alpha$ chosen. We therefore tried a procedure to recalculate, or update, this delay once the fixed codebook index $k$ is known. We refer to this final sub-optimal search procedure as "Method C", and it works as follows. The adaptive codebook delay is initially chosen assuming $G_2 = 0$. Then the fixed codebook index is found by calculating $G_1$, $G_2$ and $T_{\alpha k}$ for every $k$, and choosing the index $k$ which maximises $T_{\alpha k}$ as in the Method B search. Then once $k$ is known we update the delay $\alpha$ by finding $G_1$, $G_2$ and $T_{\alpha k}$ for each possible $\alpha$, and choosing the delay $\alpha$ which maximises $T_{\alpha k}$. To do this we need to know $\xi_\alpha$, $C_\alpha$, $\xi_k$, $C_k$ and $Y_{\alpha k}$ for all values of $\alpha$ and the value of $k$ chosen during the fixed codebook search. As explained previously $\xi_\alpha$, $C_\alpha$, $\xi_k$ and $C_k$ will all be known already, and so we must calculate $Y_{\alpha k}$ for all possible values of $\alpha$ and a fixed $k$.

This procedure to update the adaptive codebook delay once the fixed codebook index is known increases the complexity of the codec by about a further 10% relative to the complexity of the normal codec. It improved the average segmental SNR for our four speech-files to 10.6 dB, and the average weighted SNR to 7.8 dB.

The performance of the search procedures we have described in this section, along

| | Segmental SNR | Weighted SNR | Complexity |
|---|---|---|---|
| Normal Sequential Search | 9.7 | 7.3 | 1 |
| Method A | 10.1 | 7.5 | 1.02 |
| Method B | 10.3 | 7.8 | 1.3 |
| Method C | 10.6 | 7.8 | 1.4 |
| Full Search | 10.8 | 8.2 | 60 |

Table 4.1: Performance and Complexity of Various Search Procedures

with the normal and the full search methods, is shown in Table 4.1 in terms of the average segmental and weighted SNRs. Also shown are the complexities of codecs using these search procedures relative to a codec using the normal sequential search. It can be seen that the joint codebook search Method A gives a significant improvement in the codec's performance with very little extra complexity. Also we can see that Method C, the most complex sub-optimal search procedure tried, increases the codec's complexity by only 40% but gives reconstructed speech, in terms of the segmental SNR at least, very similar to that using the much more complex full search procedure.

The investigations we have reported in this section have ignored the effects of quantization of the codebook gains $G_1$ and $G_2$. However in any real coder we must somehow quantize these gains for transmission to the decoder. This is discussed in the next section.

### 4.2.5 Quantization of the Codebook Gains

In this section we study ways of quantizing the codebook gains $G_1$ and $G_2$ to try and maintain the improvements we see with our various codebook search procedures. This was necessary because we noticed, especially for female speakers, quantization of the gains had a much more serious effect in the codecs with improved search procedures than for the normal codec. This meant that the improvement which arose from the new search procedures was largely lost when quantization was considered. For example for one of our speech-files, containing the sentence "To reach the end he needs much courage" spoken by a woman, the segmental SNR of the normal codec with no quantization was 11.45 dB. With quantization of both gains this was only slightly reduced to 11.38 dB. The codec using the joint search procedure Method C gave a segmental SNR with no quantization of 12.45 dB. However with quantization this fell to 11.67 dB, meaning that the increase in the segmental SNR due to the improved search procedure fell from 1 dB without quantization to 0.3 dB with quantization.

There are several possible reasons for this effect. The most obvious is that when the gains are calculated in a different way their distributions change and so quantizers designed using the old distributions will be less effective. Also it may just be that the gains calculated with the improved search procedures are more sensitive to quantization than those calculated normally.

Notice that Equation 4.10 gives the optimum value of $G_1$ only if $G_2$ is given by Equation 4.11. When we quantize $G_2$ the optimum value of $G_1$ will change. We can find the best value of $G_1$ by substituting the quantized value of $G_2$, ie $\hat{G}_2$, into Equation 4.8. This gives

$$G_1 = \frac{C_\alpha - \hat{G}_2 Y_{\alpha k}}{\xi_\alpha}. \tag{4.20}$$

Similarly if the adaptive codebook gain has been quantized to give $\hat{G}_1$ then the optimum value of $G_2$ becomes

$$G_2 = \frac{C_k - \hat{G}_1 Y_{\alpha k}}{\xi_k}. \tag{4.21}$$

We set about improving the quantization of the gains for the codec using our best sub-optimal search procedure ie Method C. A speech-file, containing about eleven seconds of speech spoken by two men and two women, was used to train our quantizers. None of the speakers, or the sentences spoken, were the same as those used to measure the performance of the codec. Distributions for the two gains were measured using our training data when neither of the gains were quantized. We were then able to train quantizers using the Max-Lloyd algorithm [16].

There is a problem with the adaptive codebook gain $G_1$ because while most values of $G_1$ are between +1.5 and -1.5, a few values are very high. If we use all these values with the Max-Lloyd algorithm then the resulting quantizer will have several reconstruction levels which are very high and rarely used. We found that for an eight level quantizer trained using all the unquantized values of $G_1$, half the reconstruction levels were greater than 3 or less than -3. Using such a quantizer gives a serious degradation in the segmental SNR of the reconstructed speech. To overcome this problem the values of $G_1$ must be cut down to some reasonable range. The DoD [24] codec uses the range -1 to +2, and we tried this and also the range -1.5 to +1.5, which was suggested by the PDF of our data.

Another problem when using the Max-Lloyd algorithm to design a quantizer for

| | Segmental SNR | Weighted SNR |
|---|---|---|
| Normal Codec | 9.5 | 7.1 |
| Improved Search and Quantization | 10.0 | 7.5 |

Table 4.2: Performance of Search Procedures with Quantization

$G_1$ is that one reconstruction level tends to get allocated very close to zero where the PDF of the gains is low. We overcame this problem by splitting the values of $G_1$ into positive and negative values, and running the Max-Lloyd algorithm separately on each half of the data. Using these techniques we were able to design quantizers for $G_1$ which outperformed the quantizer designed for the normal codec.

Our normal codec used a four bit logarithmic quantizer for the magnitude of $G_2$, with the sign being allocated an additional bit. We also used the Max-Lloyd algorithm to design a five bit quantizer for $G_2$ using the distribution derived from our training data.

We ran our simulation of the codec with $G_1$ calculated using Equation 4.10, and quantized, and then $G_2$ calculated using Equation 4.21. Using this technique we were able to derive distributions for $G_2$ when $G_1$ was quantized with various quantizers. Similarly we were able to find distributions for $G_1$ when $G_2$ was quantized with various quantizers. These distributions were then used to train quantizers for $G_1$ to use in conjunction with those already designed for $G_2$, and vice-versa. We tried quantizing $G_1$ first using various different quantizers, and then using the specially trained quantizer for $G_2$. Similarly we tried quantizing $G_2$ first and then using various specially trained quantizers for $G_1$. The best results were obtained when $G_2$ was calculated first and quantized with the normal logarithmic quantizer, before $G_1$ was calculated using Equation 4.20 and quantized using a Max-Lloyd quantizer trained with gains cut to the range -1 to +2. Such a quantization scheme improved the segmental SNR for the female speech file described earlier from 11.67 dB to 11.97 dB. The improvement was less significant for the two male speech-files, but on average using the improved quantization scheme gave a segmental SNR of 10.0 dB and a weighted SNR of 7.5 dB. These figures should be compared to an average segmental SNR of 9.9 dB, and a average weighted SNR of 7.4 dB, when using the normal quantizers.

The average segmental SNR and weighted SNR for our four speech-files using the codec with the normal search procedure and gain quantizers, and the codec with the improved search procedure (Method C) and quantization, are shown in Table 4.2. It can be seen that on average the improved search procedure and quantization gives an increase in the segmental SNR of about half a decibel, and the weighted SNR increases by 0.4 dB. The improvements are similar for both the male and female speech-files, and in informal listening tests we found that the reconstructed speech for the improved search procedure again sounded more full and natural than that for the normal search procedure.

Next we discuss methods of improving the performance of our 4.7 kbits/s forward adaptive ACELP codec by re-calculating the synthesis filter parameters after the excitation signal $u(n)$ has been determined. However in Chapter 7 we return to joint codebook search procedures, and discuss using Method A and Method B described earlier to improve the performance of low delay backward adaptive CELP codecs.

## 4.3 Calculation of the Synthesis Filter Parameters

In the previous section we discussed ways of improving the determination of the codebook parameters which give the excitation signal $u(n)$. At the decoder this excitation signal is passed through the synthesis filter $H(z)$ to give the reconstructed speech $\hat{s}(n)$. As described in Chapter 3 $H(z)$ is usually simply assumed to be the inverse of the prediction error filter $A(z)$ which minimises the energy of the prediction residual. It is well known that this is not the ideal way to determine the synthesis filter parameters. For example when the pitch frequency is close to the frequency of the first formant, which commonly happens for high-pitched speakers, the methods of spectral analysis described in Chapter 3 tend to give spectral envelopes with sharp and narrow resonances [45]. This leads to amplitude booms in the reconstructed speech which can be annoying.

In this section we discuss ways of improving the synthesis filter $H(z)$ to maximise the SNR of the reconstructed speech. Initially for simplicity the filter coefficients were not quantized. Also the technique described in Chapter 3 of overlapping the LPC analysis frames, and interpolating the Line Spectrum Frequencies between frames, was not used. This discarding of LSF interpolation means that the filter coefficients for the weighted synthesis filter change only once per frame rather than every subframe. Therefore the energy of the filtered fixed codebook signals, ie $\xi_k$, needs to be computed only once per frame, and so the complexity of the fixed codebook search is dramatically reduced. This reduces the overall complexity of the codec by about 40%.

### 4.3.1 Bandwidth Expansion

One well known and relatively simple way of improving the synthesis filter parameters is to use bandwidth expansion [45]. In this technique the filter coefficients $a_k$, produced by the autocorrelation or covariance analysis of the input speech, are replaced by $a_k \gamma^k$ where $\gamma$ is some constant less than one. This has the effect of expanding the bandwidth of the resonances in the transfer function of the synthesis filter, and therefore helps reduce the problems mentioned above which occur when the pitch frequency is close to the first formant frequency.

The constant $\gamma$ can be expressed as [45]

$$\gamma = \exp(-\sigma \pi T) \tag{4.22}$$

where $T$ is the sampling interval and $\sigma$ is the bandwidth expansion in Hertz. We tried using a 15Hz expansion, which corresponds to $\gamma$ =0.9941, and found that this improved the segmental SNR of our 4.7 kbits/s codec (with no LSF quantization or interpolation) from 9.90 dB to 10.59 dB. Also it is reported [46] that such an expansion improves the robustness of a codec to channel errors, and so we used bandwidth expansion in our studies on error sensitivity in Chapter 5. Note that like all the results quoted in this section those above were obtained for a speech-file containing one sentence each from two male and two female speakers.

### 4.3.2 Least Squares Techniques

Given an excitation signal $u(n)$ and a set of filter coefficients $a_k$, $k = 1, 2 \cdots p$, the reconstructed speech signal $\hat{s}(n)$ will be given by

$$\hat{s}(n) = u(n) + \sum_{k=1}^{p} a_k \hat{s}(n-k). \tag{4.23}$$

We wish to minimise $E$, the energy of the error signal $e(n) = s(n) - \hat{s}(n)$, where $s(n)$ is the original speech signal. $E$ is given by

$$E \;=\; \sum_n (s(n) - \hat{s}(n))^2$$

$$= \sum_n \left( s(n) - u(n) - \sum_{k=1}^{p} a_k \hat{s}(n-k) \right)^2$$

$$= \sum_n \left( x(n) - \sum_{k=1}^{p} a_k \hat{s}(n-k) \right)^2 \tag{4.24}$$

where $x(n) = s(n) - u(n)$ is the 'target' signal. For a given frame this target is fixed once the excitation signal has been determined. The problem with Equation 4.24 is that $E$ is given in terms of not only the filter coefficients but also the reconstructed speech signal $\hat{s}(n)$ which of course also depends on the filter coefficients. Therefore we cannot simply set the partial derivatives $\partial E/\partial a_i$ to zero and obtain a set of $p$ simultaneous linear equations for the optimal set of coefficients.

One approach which has been used in Multi-Pulse Excited codecs [47, 48] is to make the approximation

$$\hat{s}(n-k) \approx s(n-k) \tag{4.25}$$

in Equation 4.24, which then gives

$$E \approx \sum_n \left( x(n) - \sum_{k=1}^{p} a_k s(n-k) \right)^2 . \tag{4.26}$$

We can then set the partial derivatives $\partial E/\partial a_i$ to zero for $i = 1, 2 \cdots p$ to obtain a set of $p$ simultaneous linear equations as shown below

$$\frac{\partial E}{\partial a_i} = -2 \sum_n \left( x(n) - \sum_{k=1}^{p} a_k s(n-k) \right) s(n-i) = 0 \tag{4.27}$$

so

$$\sum_{k=1}^{p} a_k \sum_n s(n-i)s(n-k) = \sum_n x(n)s(n-i) \tag{4.28}$$

for $i = 1, 2, \cdots, p$. Similarly to Chapter 3 two different approaches are possible depending on the limits of the summations in Equation 4.28. If we consider $s(n)$ and $u(n)$ to be of infinite duration and minimise the energy of the error signal $e(n)$ from $n = 0$ to $n = L - 1$, where $L$ is the analysis frame length, the summations in Equation 4.28 are from $n = 0$ to $L - 1$ and we have a covariance like approach [1]. Alternatively we can consider $s(n)$ and $u(n)$ to be non-zero only for $0 \leq n \leq L - 1$, which leads to an autocorrelation like approach [1] where the simultaneous equations to be solved become

$$\sum_{k=1}^{p} a_k \sum_{n=0}^{L-1-|k-i|} s(n)s(n+ \mid k-i \mid) = \sum_{n=0}^{L-1-i} s(n)x(n+i) \quad . \tag{4.29}$$

We tried these two approaches, both with and without windowing of $s(n)$ and $u(n)$, in our 4.7 kbits/s codec. We found that the updated filter coefficients were, in terms of the SNR of the reconstructed speech, usually worse than the original coefficients. This is because of the inaccuracy of the approximation in Equation 4.25. To obtain any improvement in the segmental SNR of the reconstructed speech it was necessary in each frame to find the output of the synthesis filter with the original and updated filter coefficients, and transmit the set of coefficients which gave the best SNR for that frame. Using this technique we found that the updated filter coefficients were better than the original coefficients in only about 15% of frames, and the segmental SNR of the codec was improved by about 0.25 dB.

These results were rather disappointing, so we tried to find an improved method of updating the synthesis filter parameters. One possibility comes to light if we write Equation 4.24 in a matrix notation

$$E = |\underline{x} - \underline{\underline{\hat{S}}}\,\underline{a}|^2 \tag{4.30}$$

where

$$\underline{x} = \begin{pmatrix} s(0) - u(0) \\ s(1) - u(1) \\ \vdots \\ s(L-1) - u(L-1) \end{pmatrix} \tag{4.31}$$

$$\underline{\underline{\hat{S}}} = \begin{pmatrix} \hat{s}(-1) & \hat{s}(-2) & \cdots & \hat{s}(-p) \\ \hat{s}(0) & \hat{s}(-1) & \cdots & \hat{s}(-p+1) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{s}(L-2) & \hat{s}(L-3) & \cdots & \hat{s}(L-1-p) \end{pmatrix} \tag{4.32}$$

and

$$\underline{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix}. \tag{4.33}$$

Note that here we have set the elements of $\underline{x}$ and $\underline{\underline{\hat{S}}}$ assuming that we are using the covariance like approach, but similar equations can be written for the autocorrelation approach. We must try to find a set of coefficients $\underline{a}$ such that

$$\underline{\underline{\hat{S}}}\,\underline{a} \approx \underline{x} \tag{4.34}$$

Similar problems occur in many areas of science and engineering and are solved using Least Squares (LS) methods [49]. The usual technique is to assume that the 'data' matrix $\underline{\underline{\hat{S}}}$ is known perfectly, and that the 'observation' vector $\underline{x}$ is known only approximately. Then a set of coefficients $\underline{a}$ are found such that

$$\underline{\underline{\hat{S}}}\ \underline{a} = \underline{x} + \underline{\Delta x} \tag{4.35}$$

and $|\underline{\Delta x}|^2$ is minimised. One method of solving the LS problem is to use what are called the 'normal equations'

$$\underline{\underline{\hat{S}}}^T \underline{\underline{\hat{S}}}\ \underline{a} = \underline{\underline{\hat{S}}}^T \underline{x}. \tag{4.36}$$

These equations are equivalent to those in Equation 4.28. However in our problem it is the data matrix $\underline{\underline{\hat{S}}}$ which is known only approximately, and the observation vector $\underline{x}$ which is known exactly. Therefore it seems obvious that the usual Least Squares technique will not be ideal for our purposes.

In recent years a relatively new technique called Total Least Squares (TLS) [50] has been applied to several problems, see for instance [51]. In this method errors are assumed to exist in both $\underline{\underline{\hat{S}}}$ and $\underline{x}$ and we find a set of coefficients $\underline{a}$ such that

$$\left(\underline{\underline{\hat{S}}} + \underline{\underline{\Delta \hat{S}}}\right) \underline{a} = \underline{x} + \underline{\Delta x} \tag{4.37}$$

where $\left|\left(\underline{\underline{\Delta \hat{S}}} \parallel \underline{\Delta x}\right)\right|_F^2$ is minimised. Here $|\ .\ |_F^2$ denotes the squared Frobenius norm of a matrix ie the sum of the squares of the matrix's elements, and $\left(\underline{\underline{\Delta \hat{S}}} \parallel \underline{\Delta x}\right)$ is a matrix constructed by adding $\underline{\Delta x}$ to $\underline{\underline{\hat{S}}}$ as the $p+1$th column of the new matrix.

The solution $\underline{a}$ of the TLS problem can be found using the singular value decomposition of $\left(\underline{\underline{\hat{S}}} \parallel \underline{x}\right)$ [50]. We tried this technique, but found that it was not useful because a very large number (about 95%) of the sets of filter coefficients it gave resulted in unstable synthesis filters.

One final Least Squares method we tried was the Data Least Squares (DLS) technique [52]. Here all the errors are assumed to lie in the data matrix $\underline{\underline{\hat{S}}}$, and a set of coefficients are found such that

$$\left(\underline{\underline{\hat{S}}} + \underline{\underline{\Delta \hat{S}}}\right)\ \underline{a} = \underline{x} \tag{4.38}$$

This is much closer to what we want in our situation, and again the solution can be found using singular value decomposition. However we found that the filter coefficients produced were very similar to those given by the TLS technique, with again about 95% of the updated synthesis filters being unstable. Therefore unfortunately neither the TLS nor the DLS update are practical solutions for our problem.

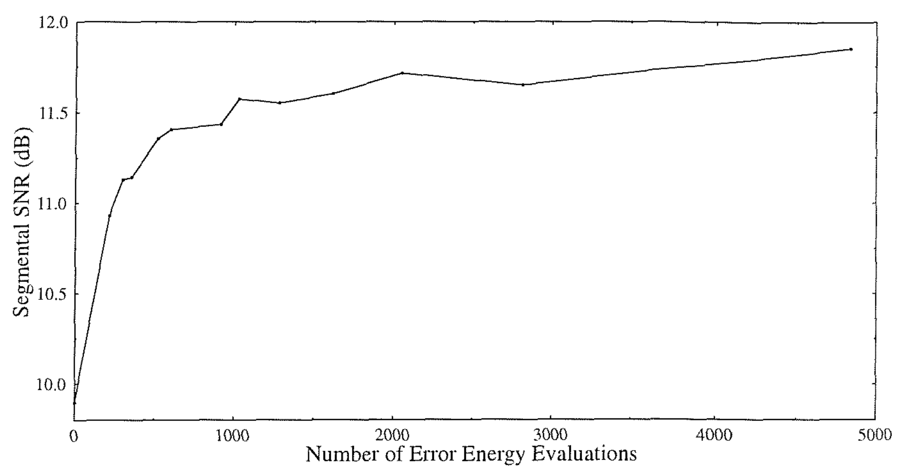

Figure 4.1: Powell Optimization Performance

### 4.3.3 Optimization via Powell's Method

Given our input speech signal $s(n)$, the filter's excitation $u(n)$ and the reconstructed speech memory $\hat{s}(-p), \hat{s}(-p+1), \cdots, \hat{s}(-1)$, the error energy $E$ is a function of the $p$ filter coefficients. Thus we can consider $E$ as a $p$ dimensional function which we wish to minimise. There are many different methods [49] for the minimisation of multidimensional functions, and we tried using the direction set, or Powell's, method [49]. This method works by iteratively carrying out a series of one dimensional line minimisations, and attempting to find a series of 'conjugate' directions for these minimisations so that the minimum along one direction is not spoiled by subsequent movement along the others. At each iteration a line minimisation is carried out along each of $p$ directions, and then the $p$ directions are updated to try and obtain the ideal conjugate directions. See [49] for details. The process ends when the decrease in $E$ during a particular iteration is less than some given fractional tolerance. When this happens it is assumed that we have settled into a minimum, which we hope is the global minimum of $E$. In our simulations the line minimisations were carried out using Brent's method [49]. This does a series of evaluations of $E$ for various sets of filter coefficients, and hunts down the minimum along a particular direction using either a golden section search or parabolic interpolation.

We tried this Powell optimization for various values of the fractional tolerance

| | Segmental SNR |
|---|---|
| Codec with no Interpolation or BW Expansion | 9.90 |
| Codec with Least Squares Optimization | 10.13 |
| Codec with LSF Interpolation only | 10.49 |
| Codec with Bandwidth Expansion Only | 10.59 |
| Codec with Interpolation and BW Expansion | 11.29 |
| Codec with Powell Optimization | 11.85 |

Table 4.3: Performance of Various Synthesis Filter Determination Techniques

which controls when the process of iterations should end. A good indicator of the complexity of minimisation procedures, such as Powell's method, is the number of times the function $E$ to be minimised is evaluated. Every 100 evaluations are approximately as complex as the whole encoding process in our standard ACELP codec. Figure 4.1 shows how the segmental SNR of our 4.7 kbits/s codec with a Powell optimization of the synthesis filter varies with the number of evaluations of $E$ carried out. The best SNR we were able to obtain was 11.85 dB, which was about 2 dB better than the segmental SNR of the codec without interpolation of the LSFs. However as shown in Table 4.3 this difference is much reduced if we use bandwidth expansion and interpolation of the LSFs in the codec, and these method are much less complex than the Powell's update. The Powell optimization is not a realistic option for a real codec, but it does give us an idea of the absolute best performance we can expect from updating the synthesis filter parameters. We see that without LSF quantization this is only about half a decibel better than a codec with LSF interpolation and bandwidth expansion.

### 4.3.4 Simulated Annealing and the Effects of Quantization

In any real coder it is necessary to quantize the synthesis filter parameters for transmission to the decoder. It is not clear whether this need for quantization will make updating the LPC parameters more or less worthwhile. On one hand the quantization may mask and reduce the improvement due to the update, but on the other hand the updating algorithm can take account of the quantization when it is choosing a set of filter parameters and this may lead to the update having more effect.

We decided to start our investigation of the effects of updating the synthesis filter parameters with quantization of the LSFs by finding an upper limit to the improvement possible. The Powell optimization method is meant to work on functions of continuous variables and so is not suitable when we consider quantization of the LSFs. Instead we used the technique of simulated annealing [49], which is more suitable for discrete optimization.

Simulated annealing works, as its name suggests, in analogy to the annealing (or slow cooling) of metals. When metals cool slowly from their liquid state they start in a very disordered and high energy state and reach equilibrium in an extremely ordered crystalline state. This crystal is the minimum energy state for the system, and simulated annealing similarly allows us to find the global minimum of a complex function with many local minima. The procedure works as follows. The system starts in an initial state, which in our situation is an initial set of quantized LSFs. A temperature like variable $T$ is defined, and possible changes to the state of the system are randomly generated. For each possible change the difference $\Delta E$ in the error energy between the present state and the possible new state is evaluated. If this is negative, ie the new state has a lower energy than the old state, then the system always moves to the new state. If on the other hand $\Delta E$ is positive then the new state has higher energy than the old state, but the system may still change to this new state. The probability of this happening is given by the Boltzmann distribution

$$\text{prob} = \exp\left(\frac{-\Delta E}{kT}\right) \tag{4.39}$$

where $k$ is a constant. The initial temperature is set so that $kT$ is much larger than any $\Delta E$ that is likely to be encountered, so that initially most offered moves will be taken. As the optimization proceeds the 'temperature' $T$ is slowly decreased, and the number of moves to states with higher energy reduces. Eventually $kT$ becomes so small that no moves with positive $\Delta E$ are taken, and the system comes to equilibrium in what is hopefully the global minimum of its energy.

The advantage of simulated annealing over other optimization methods is that it should not be deceived by local minima and should slowly make its way towards the global minimum of the function to be minimised. In order to guarantee that this happens it is important to ensure that the temperature $T$ starts at a high enough value, and is reduced suitably slowly. We followed the suggestions in [49] and reduced $T$ by 10% after every $100p$ offered moves, or every $10p$ accepted moves. The initial temperature was set so that $kT$ was equal to ten times the highest value of $\Delta E$ that was initially encountered. The random changes in the state of the system were generated

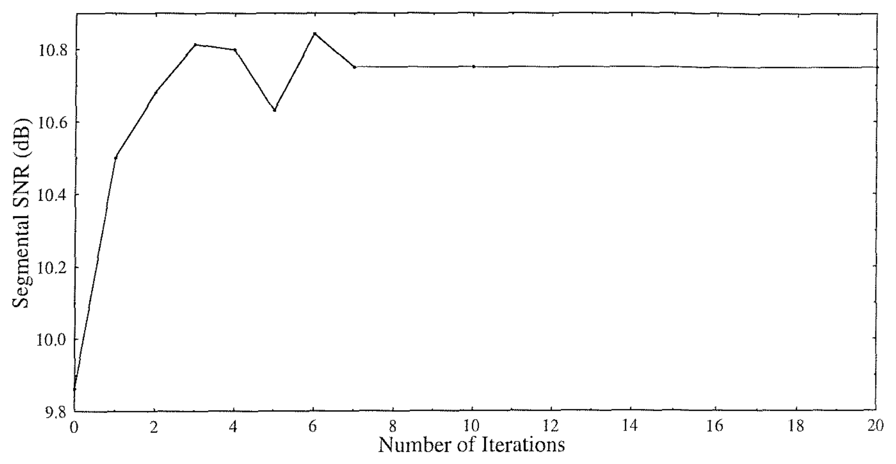

Figure 4.2: Performance of Quantized LSF Update Scheme

by randomly choosing an LSF and then moving it up or down by one quantization level, provided that this did not lead to an LSF overlap, as it is necessary to avoid unstable synthesis filters.

We found that we were able to improve the segmental SNR of our 4.7 kbits/s codec with quantization of the LSFs from 9.86 dB to 10.92 dB. Note furthermore that we were able to achieve almost the same improvement with a much simpler search technique described below. Rather than choose an LSF at random to modify, and accept some changes which increase the error energy as well as all those which reduce the energy, we cycled sequentially through all $p$ LSFs in turn. Each LSF was moved up and down one quantizer level to see if we could reduce the error energy. Any changes which reduced the error energy, but none which increased it, were accepted. This process can be repeated any number of times, with every testing of all $p$ LSFs counting as one iteration. The segmental SNR of our codec against the number of update iterations used is shown in Figure 4.2

We see that this method of updating the quantized synthesis filter parameters produces a segmental SNR of 10.8 dB after just three iterations. This is almost equal to the improvement produced by simulated annealing of the LSFs, and yet the complexity of the codec is increased by only about 80%. The improvement obtained (about 1 dB) is similar to that quoted in [48] of 10% in Multi-Pulse codecs at segmental SNRs of around 10 dB. However the method used in [48] required recalculating the

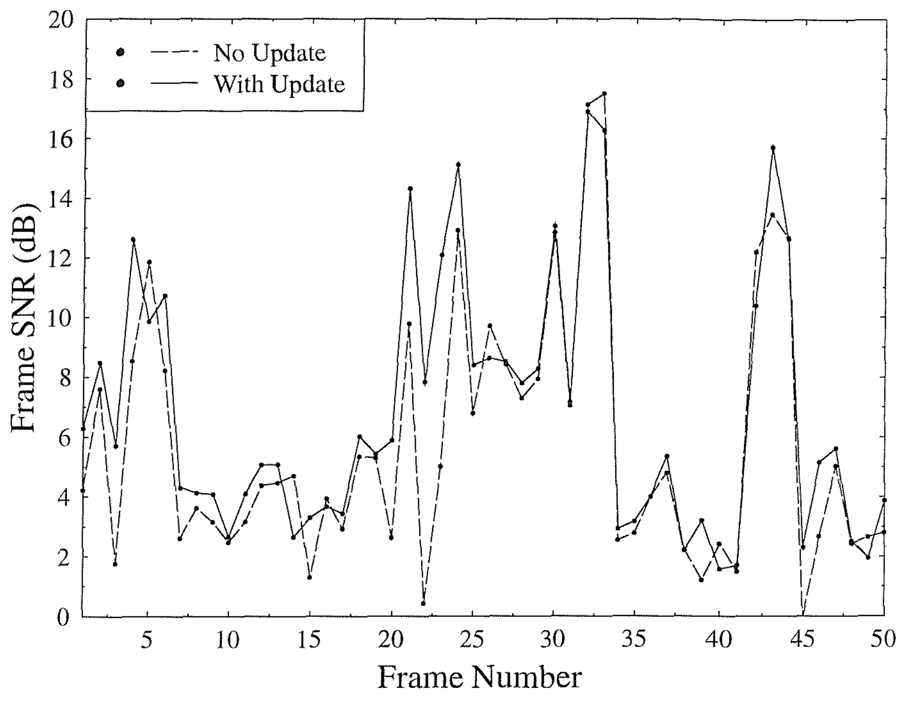

Figure 4.3: Effect of Update on Variation of SNR

excitation after the update of the synthesis filter parameters, and so approximately doubled the complexity of the codec.

As mentioned in [48] not only does the updating of the synthesis filter help to increase the average segmental SNR, but it also helps remove the very low minima in SNR that occur for some frames. This effect is shown in Figure 4.3 which shows the variation of SNR for a sequence of fifty frames for 4.7 kbits/s codecs with and without update of the synthesis filter. The update used three iterations of the scheme described above. These low minima that occur can be subjectively annoying and so it is helpful if they can be partially removed

It is also possible to update the synthesis filter so as to try and increase the weighted SNR for each frame. We tried this using the iterative scheme described above, and found that the improvement in the weighted segmental SNR available through an update saturated after just one iteration. The weighted segmental SNR increased from 7.18 dB to 7.43 dB, and the conventional segmental SNR increased from 9.86

| | Segmental SNR |
|---|---|
| Codec with no Interpolation or BW Expansion | 9.86 |
| Codec with Bandwidth Expansion Only | 9.89 |
| Codec with LSF Interpolation only | 10.31 |
| Codec with Interpolation and BW Expansion | 10.76 |
| Codec with Iterative Update | 10.75 |
| Codec with Simulated Annealing Update | 10.92 |

Table 4.4: Performance of Synthesis Filter Techniques with Quantization

dB to 10.08 dB.

The results described above comparing codecs with updated synthesis filter parameters to a codec with no update are reasonably good. However, as noted earlier for the codecs with no quantization of the LSFs, the results are not so impressive when compared to codecs using the techniques of bandwidth expansion and interpolation of the LSFs. This is shown in Table 4.4 Using both bandwidth expansion and interpolation of the LSFs gives a segmental SNR almost identical to that achieved using the iterative update algorithm. Also the interpolation and bandwidth expansion help remove the very low minima in the SNR in the same way that the update does. Although several papers [48, 53, 54] have appeared reporting reasonable improvements using various methods of update, to our knowledge none of them have considered the effects of LSF interpolation and bandwidth expansion. Our codec with the iterative update of the LSFs is about 10 % more complex than the codec with interpolation and bandwidth expansion. However the LSF interpolation scheme described in Section 3.3 increases the delay of the codec by two sub-frames, or 15 ms. Both interpolation (when used along with bandwidth expansion) and the iterative update scheme give very similar improvements in the performance of the codec. If a 15 ms increase in the delay of the codec is not important then the LSF interpolation can be invoked. However our iterative update scheme provides an alternative which gives similar results without increasing the delay of the codec, and is only slightly more complex.

The work reported in this chapter is summarised in [55]. In the next chapter we move on to investigating the error sensitivity of our 4.7 kbits/s ACELP codec.

# Chapter 5

# The Error Sensitivity of CELP Codecs

## 5.1 Introduction

As described in Chapter 3 CELP [2] seems to offer the most promising codecs for the next generation of mobile communication networks. CELP codecs are capable of producing good toll quality speech at low bit-rates with reasonable complexity. However almost equally important for a codec which is to be used over a radio channel is its ability to cope with random bit errors between the encoder and decoder. A mobile radio channel is particularly hostile [1] and when there is no line of sight path between the receiver and transmitter multi-path propagation leads to a channel which can be described by the Rayleigh distribution. Such a channel is not memory-less and deep fades of -20 dB, or more, are common. Such fades lead to error bursts and therefore it is necessary to use either interleaving, which attempts to randomise the bit errors, or a channel coder with good burst error correcting abilities. In any case a channel coder is essential for any speech coder which is to be used over a mobile radio channel at reasonable channel signal to noise ratios. However no channel coder will be able to remove all the bit errors without requiring an unreasonable bandwidth, and so even with channel coding it is important that the speech codec should be as robust as possible to errors.

In this chapter we describe several methods for improving the bit error sensitivity of our coder, and also how to measure the error sensitivity of the speech encoder output bits so that the matching channel coder can be carefully designed to give most protection to the bits which are most sensitive. The results of simulations which are reported refer to our 4.7 kbits/s codec, and similar results were found to apply to the 7.1 kbits/s codec.

## 5.2 Improving the Spectral Information Error Sensitivity

It has been noted [56, 57] that the spectral parameters in CELP coders are particularly sensitive to errors. There are many different ways to represent these parameters, but Line Spectral Frequencies (LSFs) [40] offer some definite advantages in terms of error robustness. One advantage is that the spectral sensitivities of the LSFs are localized [41], so that an error in a given LSF produces a change in the resulting spectrum only in the neighbourhood of the corrupted LSF. Another advantage is the ordering property of the LSFs. This means that for the synthesis filter to be stable, it is a necessary and sufficient condition that the LSFs from which it was derived are ordered, ie $LSF_1 < LSF_2 < LSF_3$ etc. Therefore if a set of LSFs are received which are not ordered, the decoder knows that there must be at least one error in the bits that represent these LSFs, and some action must be taken to rectify this error and produce a stable synthesis filter. It is this action which is studied here.

### 5.2.1 LSF Ordering Policies

There is a high correlation between the LSFs of successive frames. This means that, as reported in [57], occasionally the LSF set for a given frame can be replaced by the set from the previous frame without introducing too much audible distortion. Therefore one possible policy for dealing with frames where non-monotonic LSFs are received is to completely discard the LSFs which were received for that frame, and use those from the previous frame.

A better policy is to try and replace those LSFs which need to be, rather than all of them. In [58] when a non-monotonic set of LSFs is received, the two particular frequencies which cross over are replaced by the corresponding frequencies from the previous frame. Only if the resulting set of LSFs is still not ordered is the whole set replaced.

Several attempts have been made to try and identify which particular LSF is causing the instability, and then replace only it. In [59] use is made of the long-term statistics of the differences between adjacent LSFs in the same frame. If two frequencies cross over then an attempt is made to guess which one was corrupted, and in general the guess is right about eighty percent of the time. This "hit ratio" can be improved by including a voicing decision – in a frame of voiced speech the formants are sharper than in unvoiced frames, and so the spacings between adjacent LSFs are generally smaller.

Instead of trying to guess which LSF from a non-monotonic set is corrupted, and then replacing this LSF with the corresponding frequency from a previous frame, we tried to produce a monotonic set of LSFs by inverting various bits in the received bit-stream. Initially we try to determine which set of bits should be examined. For example if $LSF_i > LSF_{i+1}$ then we know that either $LSF_i$ or $LSF_{i+1}$ has been corrupted. When such a crossover is found we take the following steps

1. We check to see if $LSF_i > LSF_{i+2}$. If it is we assume that $LSF_i$ is corrupted and select the bits representing this LSF as those to be examined.

2. We check to see if $LSF_{i-1} > LSF_{i+1}$. If it is we assume $LSF_{i+1}$ is in error and select these bits to be examined.

3. If neither of the checks above indicate whether it is $LSF_i$ or $LSF_{i+1}$ which is corrupted then the bits representing both these LSFs are selected to be examined.

4. We try to correct the LSF crossover by inverting each bit, one at a time, from those to be examined. After each bit inversion the new value of $LSF_i$ or $LSF_{i+1}$ is decoded and checked to see if the crossover has been removed, and no new cross-overs introduced. If several possible codes are found then the one which gives the corrected LSFs as close as possible to their values in the previous frame is chosen.

5. If, as occasionally happens at high bit error rates, no single bit inversion can be found which corrects the LSF crossover, and introduces no new crossover, then we adopt the policy which is recommended in [60]. First $LSF_i$, then $LSF_{i+1}$, then both, and finally the entire LSF set, is replaced by those in the previous frame until a monotonic set is found.

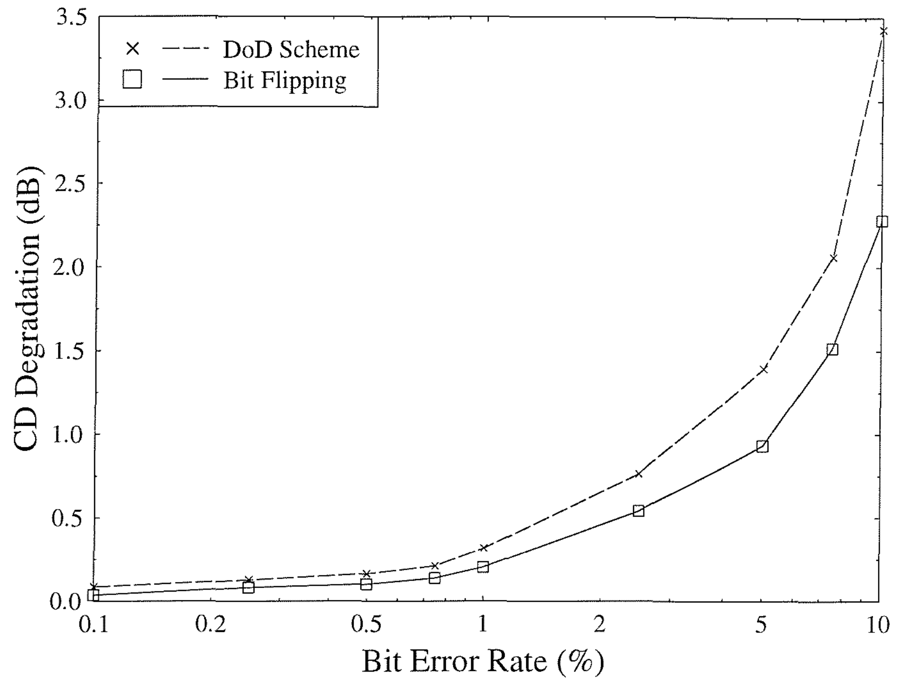

Figure 5.1: The CD Degradation Produced By Random Corruption of LSF Bits

We simulated the effect of the error correction scheme described above over a set of four sentences spoken by different speakers. The predictor coefficients were determined in a 4.7 kbits/s coder using the autocorrelation approach and a 15Hz bandwidth expansion was used. The LSFs were non-uniformly quantized with 34 bits. The Cepstral Distance (CD) [9] degradation produced by errors in the bits representing the LSFs is shown in the Figure 5.1. The dotted curve represent the effect of the scheme described in [58]. As can be seen our correction policy gives consistently better results, and a definite subjective improvement was heard in informal listening tests.

Also in [59] a table of "Hit ratio" figures is included to indicate how often the correct LSF for replacement was chosen at various bit error rates. The figures for the improved hit ratio which resulted when the voicing decision was used are reproduced in Table 5.1. Also shown in this table is the hit ratio for our scheme, ie how often the bit which was inverted was part of the codeword for the LSF which had actually been

| Bit error rate (%) | 0.1 | 1 | 2 | 2.5 | 3 | 4 |
|---|---|---|---|---|---|---|
| Atungsiri's Scheme | 100 | 80 | 80 | 82 | 79 | 80 |
| Our Scheme | 100 | 88 | 92 | 93 | 93 | 92 |
| Correct Bit Hit | 83 | 81 | 80 | 77 | 78 | 78 |

Table 5.1: Hit Ratios For Various Algorithms

corrupted. As can be seen our scheme performs significantly better than that reported in [59]. In the final row of the table are the figures for how often the correct bit is inverted when a non-monotonic set of LSFs is received. As can be seen the bit causing the LSF overlap is corrected about eighty percent of the time, and when this happens the effect of the bit error is completely removed. As about 30 percent of corrupted LSF bits produce LSF cross-overs, this means that about twenty five percent of all LSF errors can be entirely removed by the decoder.

### 5.2.2 The Effect of FEC on the Spectral Parameters

Although our scheme described above can remove the effect of channel errors on the LSF bits about twenty five percent of the time, the reconstructed speech is unacceptably distorted if the bit error rate among the LSF bits is above about one percent. Therefore some sort of error correction code is necessary if the coder is to be used at higher bit error rates. We found which of the LSF bits were most susceptible to errors by taking one LSF bit at a time and corrupting it ten percent of the time. The resulting degradations in the segmental SNR and the Cepstral Distance of the reconstructed speech were noted. The 13 bits which were least sensitive in terms of CD degradation all gave a degradation of less than 0.05 dB when corrupted ten percent of the time, and were left unprotected. The remaining 21 bits were protected with a (31,21,2) BCH code which was simulated as follows. If two or less errors were generated in the 31 bit code word then they were corrected, and if more than two errors were generated then we assumed that although the BCH code would be unable to correct these errors, it would at least be able to detect that the protected 21 bits may contain errors. Then in the decoding of the speech if an LSF crossover was found the decoder attempts to put it right by examining only unprotected bits, unless the BCH code indicates that the 21 protected bits may contain an error.

Thus the effect of including FEC on some of the LSF bits is not only that the most sensitive bits are completely protected (unless the code fails), but also when an LSF crossover occurs because of an error in one of the less sensitive bits, the bit flipping algorithm is much more likely to select the correct bit to toggle. In fact we found that for frames where the FEC had not failed, if an LSF crossover occurred it was correctly fixed almost one hundred percent of the time. In informal listening tests we found that for a bit error rate of 2.5% among the LSF bits the distortions produced were barely noticeable, and at 5% although the distortions were noticeable the reproduced speech was still of acceptable quality.

Recently an alternative means of improving the performance of speech and channel codecs, based on similar ideas, has been proposed [61]. This uses the ordering property of the LSFs, along with a specific property of multi-band excited codecs, to feed back information from the speech decoder to the channel decoder. The speech decoder indicates to the channel decoder if a set of received bits results in an LSF crossover, or is otherwise unlikely to be correct. The channel decoder can then use this information to help it decode the correct information from the received bit stream. Good results, in terms of the error correcting capability of the source aided channel decoder, are reported.

### 5.2.3 The Effect of Interpolation

In our codec the usual practice of employing interpolation between the present and the previous set of LSFs is used. This helps minimize sudden sharp changes in the short-term predictor filter coefficients between one frame and the next. However, as can be seen from Figure 5.2, it also leads to increased propagation of the effect of an LSF error from one frame to the next. The upper graph shows the average effect, in terms of degradation of the frame SNR and CD, of an error in one of the LSF bits in the coder with LSF interpolation.The bit is corrupted in frame 0 and the graph shows how the resultant degradation dies out from one frame to the next. In frame 1 the corrupted set of LSFs is used along with the present set to form the interpolated LSFs. Hence the effect of the error is almost as serious in the frame following the error as it is in the corrupted frame. After this the effect of the error quickly disappears.

Because of this error propagation it might be expected that the error sensitivity of the bits representing the LSFs could be improved by removing the interpolation. However we found that removing interpolation from the codec reduced its clear channel segmental SNR by about 0.5 dB, and at various error rates between 0.1% and

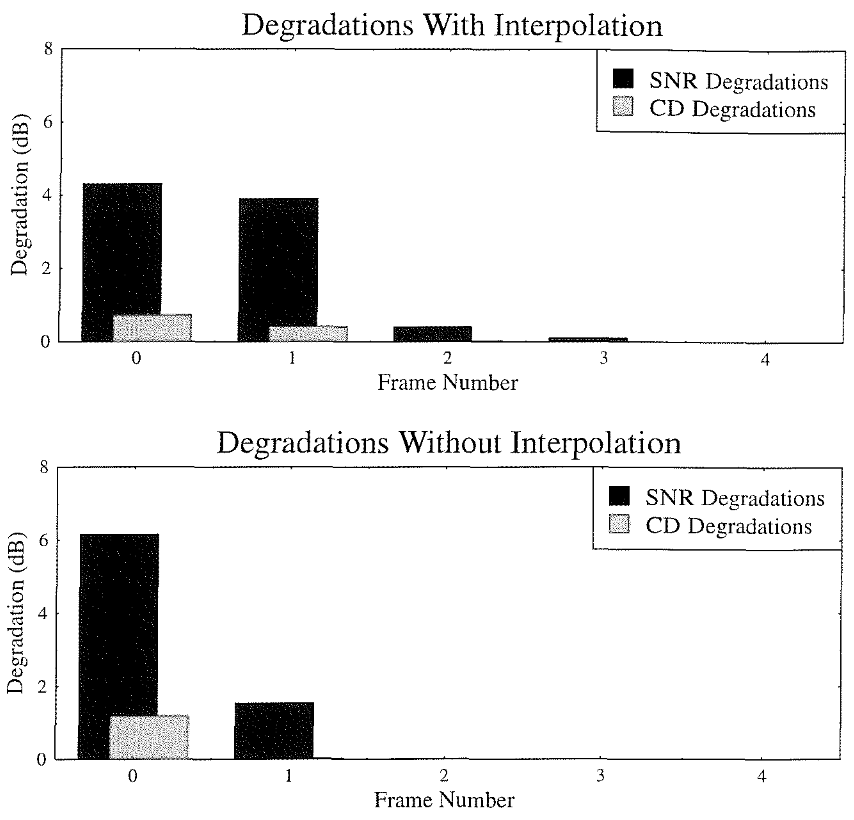

Figure 5.2: The Effect of Interpolation on Error Propagation

10% the resultant degradations are almost identical to those found in the coder with interpolation. The lower graph in Figure 5.2 shows the effect of an error (on the same LSF bit as was used in the upper graph) in the coder in which interpolation is not used. It can be seen that although the error propagation is reduced, the degradation in the frame which was corrupted is increased. This is because interpolation helps to smooth out the effect of an LSF error in the corrupted frame.

## 5.3 Improving the Error Sensitivity of the Excitation Parameters

Most of the bits transmitted by a CELP coder are used to represent the excitation for the synthesis filter. In our coder the information which must be sent to the decoder is

1. The fixed codebook index. Twelve bits per sub-frame are used.

2. The fixed codebook gain. Four bits are used to represent the magnitude, which is logarithmically quantized, and one bit is used to represent the sign.

3. The adaptive codebook delay. The delay can vary between 20 and 147 samples and so seven bits per sub-frame are needed to represent this information.

4. The adaptive codebook gain. Three bits per sub-frame are used.

The error sensitivity of this information, and ways of improving it, are discussed below.

### 5.3.1 The Fixed Codebook Index

The algebraic codebook structure used in our codec is inherently quite robust to channel errors. This is because if one of the codebook index bits is corrupted, the codebook entry selected at the decoder will differ from that used in the encoder only in the position of one of the four non-zero pulses, ie the corrupted codebook entry will be similar to the original. This is in contrast to traditional CELP coders which use a non-structured, randomly filled, codebook. In such codecs when a bit of the index is corrupted a new codebook address is decoded and the codebook entry used is entirely different to the original. Hence errors in the codebook index in such coders will be more significant than in ours. Such a codebook is used in [57] where SNR degradations of about 8 dB are recorded when a codebook index bit is corrupted in every frame. In our coder the corresponding degradation is only about 4 dB.

It is generally reported [58, 60] that errors in the fixed codebook index produce reconstructed speech in which the degradations are not perceptually annoying. Therefore the fixed codebook index is often left unprotected.

### 5.3.2 The Fixed Codebook Gain

The magnitude of the fixed codebook gain tends to vary quite smoothly from one sub-frame to the next. Therefore errors in the codebook gain can be spotted using a smoother to indicate, from the neighbouring gains, what range of values the present codebook gain should lie within. If a codebook gain is found which is not in this range then it is assumed to be corrupted, and replaced with some other gain.

We want a scheme which will spot as many errors in the codebook gain as possible, without introducing too many new errors by replacing gains which were not originally corrupted by the channel. After careful investigation of the effects of bit errors on the fixed codebook gain magnitude we implemented the following scheme. Every codebook gain quantizer level at the decoder is checked by calculating the mean and standard deviation of its two nearest neighbours. If the standard deviation of these neighbours is less than two quantizer levels then it is set equal to two. We then check to see if the present level is within 2.25 standard deviations of the mean calculated from its neighbours. If not it is assumed to be corrupted. When the codebook gain bits are corrupted with an error rate of 2.5% then this scheme spots almost 90% of the errors in the Most Significant Bit (MSB) of the gain level, while in error free conditions it falsely spots errors in only about 0.5% of the sub-frames. This false error spotting produces a small degradation in the decoder performance at zero bit error rate. However if some feedback between the channel decoder and the speech decoder is implemented so that the smoother is disabled in error free conditions, as suggested in [57], then this degradation is removed.

Another important aspect of the smoother is how gains which are thought to be corrupted are replaced. In [57] when a gain magnitude is thought to be in error it is replaced with the mean of its neighbours' magnitudes. However we found that a bit flipping scheme, similar to that used to correct LSF cross-overs, produced better results. When an error is spotted the decoder inverts all four bits, one at a time, in the received codeword for the gain magnitude. The single bit inversion which produces a decoded gain level as close as possible to the mean of its neighbours is chosen.

The effect of our smoother on the error sensitivity of the four bits per sub-frame representing the fixed codebook gain magnitude is shown in Table 5.2. This table shows the SNR degradation produced in 4.7 kbits/s codecs with and without smoothing when the bits shown are corrupted in every frame (the bits are corrupted for one sub-frame only per frame). As can be seen the smoothing improves the error sensitivity of all the bits, most especially the MSB in which most of the errors are spotted and corrected by the smoother.

The fixed codebook gain sign shows erratic behaviour and is not suitable for smoothing. This bit is among the most sensitive of the coder and should be well protected by the channel codec.

| Gain Bit | SNR Deg (dB) No Smoothing | SNR Deg (dB) With Smoothing |
|---|---|---|
| LSB | 1.4 | 1.3 |
| Bit 2 | 3.0 | 2.8 |
| Bit 3 | 6.2 | 4.8 |
| MSB | 10.5 | 2.1 |

Table 5.2: SNR Degradations For Fixed Codebook Gain Bits With and Without Smoothing

### 5.3.3 Adaptive Codebook Delay

Seven bits per sub-frame are used to encode the adaptive codebook delay, and most of these are extremely sensitive to channel errors. An error in one of these bits produces a large degradation not only in the frame in which the error occurred, but also in subsequent frames, and generally it takes more than ten frames before the effect of the error dies out.

If the adaptive codebook delay is chosen by the encoder by merely minimising the weighted mean square error of the reconstructed speech, its behaviour will be erratic and not suitable for smoothing. The delay can be forced to take on smooth behaviour by modifying the encoder to choose slightly sub-optimal delays. This then allows the decoder to use smoothing to minimise the effect of errors. However there is a noticeable clear channel degradation due to the sub-optimal delays chosen by the encoder.

Another approach [62, 57] is to use simulated annealing to assign codewords to delays so that common codewords have good neighbours. This means that when a common codeword is corrupted the new delay selected is such that the resultant degradation is minimised. This approach, along with smoothing, is used in the DoD 4.8 kbits/s standard [24], but as it has been studied extensively already we did not try it.

### 5.3.4 Adaptive Codebook Gain

The pitch gain is much less smooth than the fixed codebook gain, and is not suitable for smoothing. However its error sensitivity can be slightly increased by coding the quantizer level with a Gray code rather than the Natural Binary Code (NBC). The effect off this is shown in Table 5.3, which gives the SNR degradation for the two codes caused by bit errors (at a rate of 10%) in the three bits used to represent the gain in

| Gain Bit | SNR Deg (dB) NBC | SNR Deg (dB) Gray Code |
|---|---|---|
| Bit 1 | 1.9 | 1.9 |
| Bit 2 | 3.0 | 1.7 |
| Bit 3 | 5.3 | 4.8 |

Table 5.3: The Effect of Using A Gray Code For The LTP Gain

| Bit Numbers | Represents |
|---|---|
| 1 to 34 | LSFs |
| 35 to 41 | Adaptive Codebook Delay (Sub-frame 1) |
| 42 to 44 | Adaptive Codebook Gain (Sub-frame 1) |
| 45 to 56 | Fixed Codebook Index (Sub-frame 1) |
| 57 | Fixed Codebook Gain Sign (Sub-frame 1) |
| 58 to 61 | Fixed Codebook Gain (Sub-frame 1) |

Table 5.4: Bit Numbering

one sub-frame.

## 5.4 Matching Channel Coders to the Speech Coder

It is very clear that some bits are much more sensitive to channel errors than others, and so should be more heavily protected by the channel coder. However it is not obvious how the sensitivity of different bits should be measured. One commonly used approach [57] is, for a given bit, to invert this bit in every frame and measure the segmental SNR degradation which results. The error sensitivity of various bits for our coder measured in this way is shown in Figure 5.3. What information various bits represent is given in Table 5.4. Another similar approach [56] is to measure the degradations in both the SNR and the Cepstral Distance which result from systematic inversion of a given bit in every frame, and combine these measures to give an overall sensitivity measure.

The problem with these approaches is that they do not take adequate account of the different error propagation properties of different bits. This means that if instead of corrupting a bit in every frame it is corrupted randomly with some error probability then the relative sensitivity of different bits will change. We propose a new measure of error sensitivity. For each bit a graph similar to that in Figure 5.2 is found, ie we find the average SNR degradation for a single bit error in the frame in which the error occurs and in the following frames. The total SNR degradation is then found

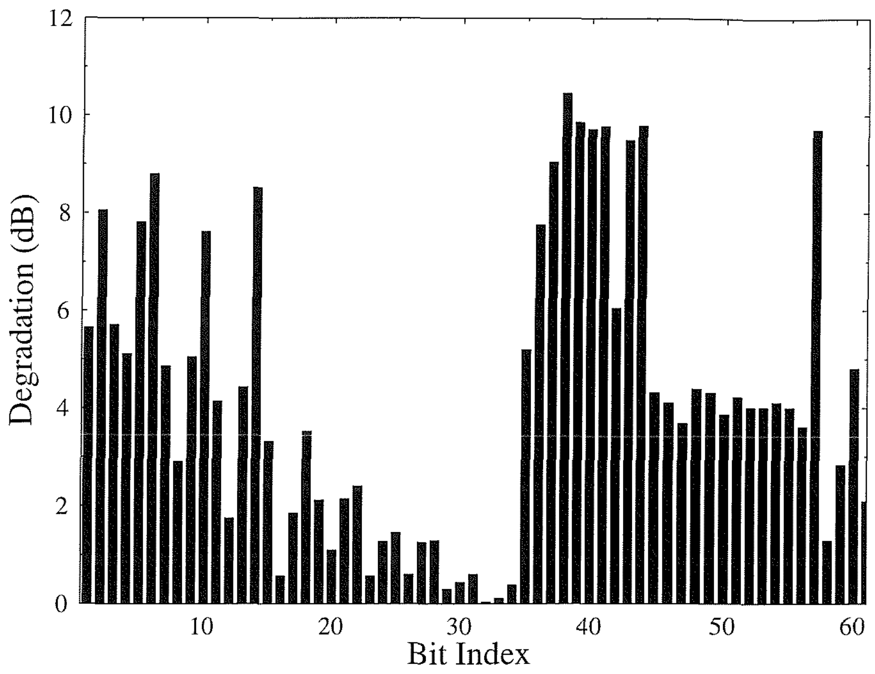

Figure 5.3: The SNR Degradations Due To 100% Bit Error Rate

by adding together the degradations in frames 0,1,2 etc. This total degradation is equivalent to the average SNR degradation which will be produced by a single error in a given bit. Of course the effect of a single error on the segmental SNR will be averaged out over all the frames of the speech file, so that for example if a bit with a total SNR degradation of 10 dB is corrupted once in a speech-file of 100 frames then the overall degradation in the segmental SNR will on average be 0.1 dB. The exact degradation depends very much on which frame the bit is corrupted in – corrupting a given bit in one frame of a speech-file can produce a much larger degradation in the segmental SNR for that file than corrupting the same bit in a different frame. This is shown in Figure 5.4 which gives the degradation in the segmental SNR produced by a single bit corruption, versus the frame in which the corruption takes place, for various different bits.

Figure 5.5 shows, for various bits, the average effect of a bit error in the frame in which the error occurred and in the following frames. The different error propagation

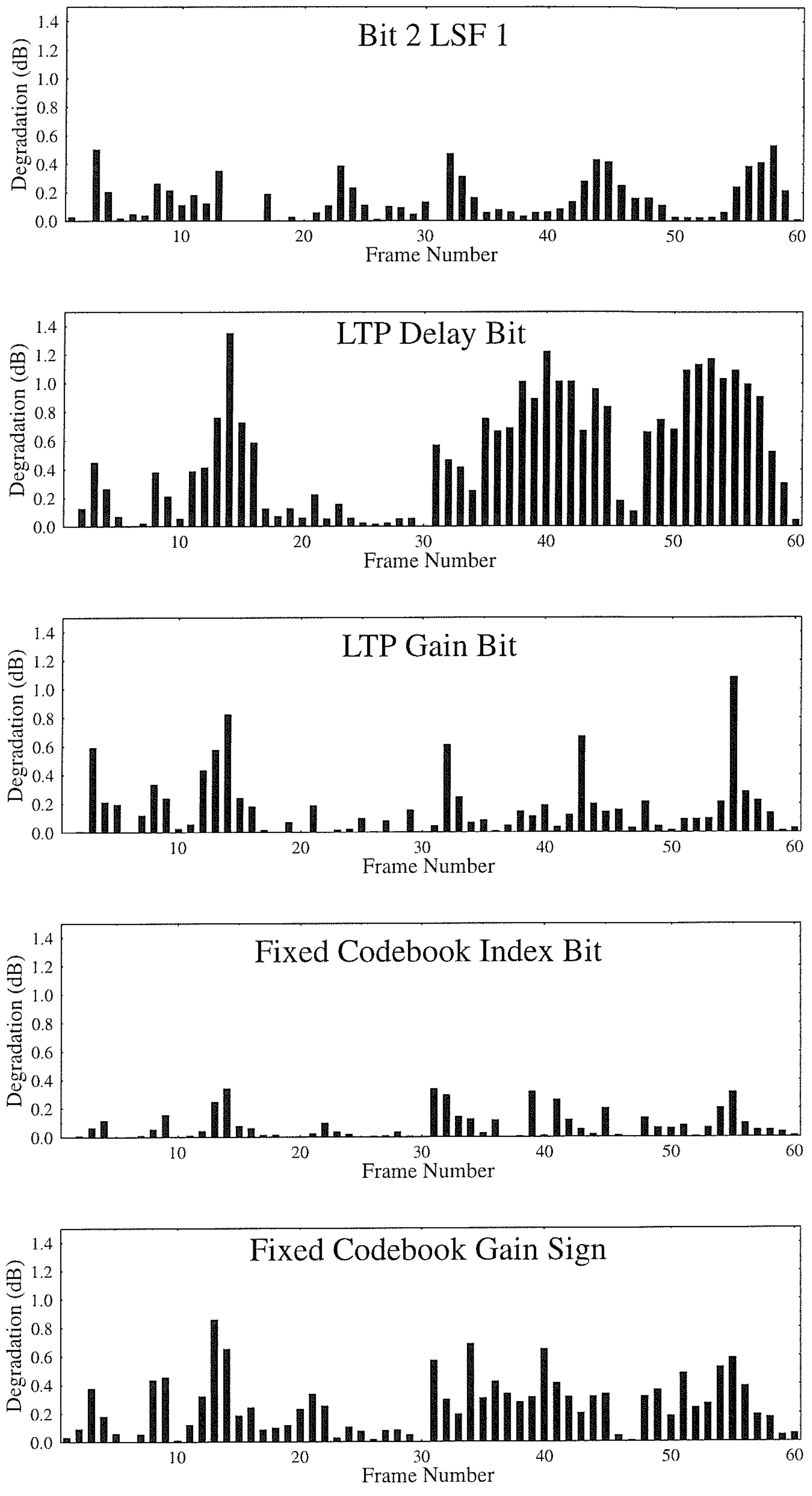

Figure 5.4: The Degradation Caused By Bit Errors In Different Frames

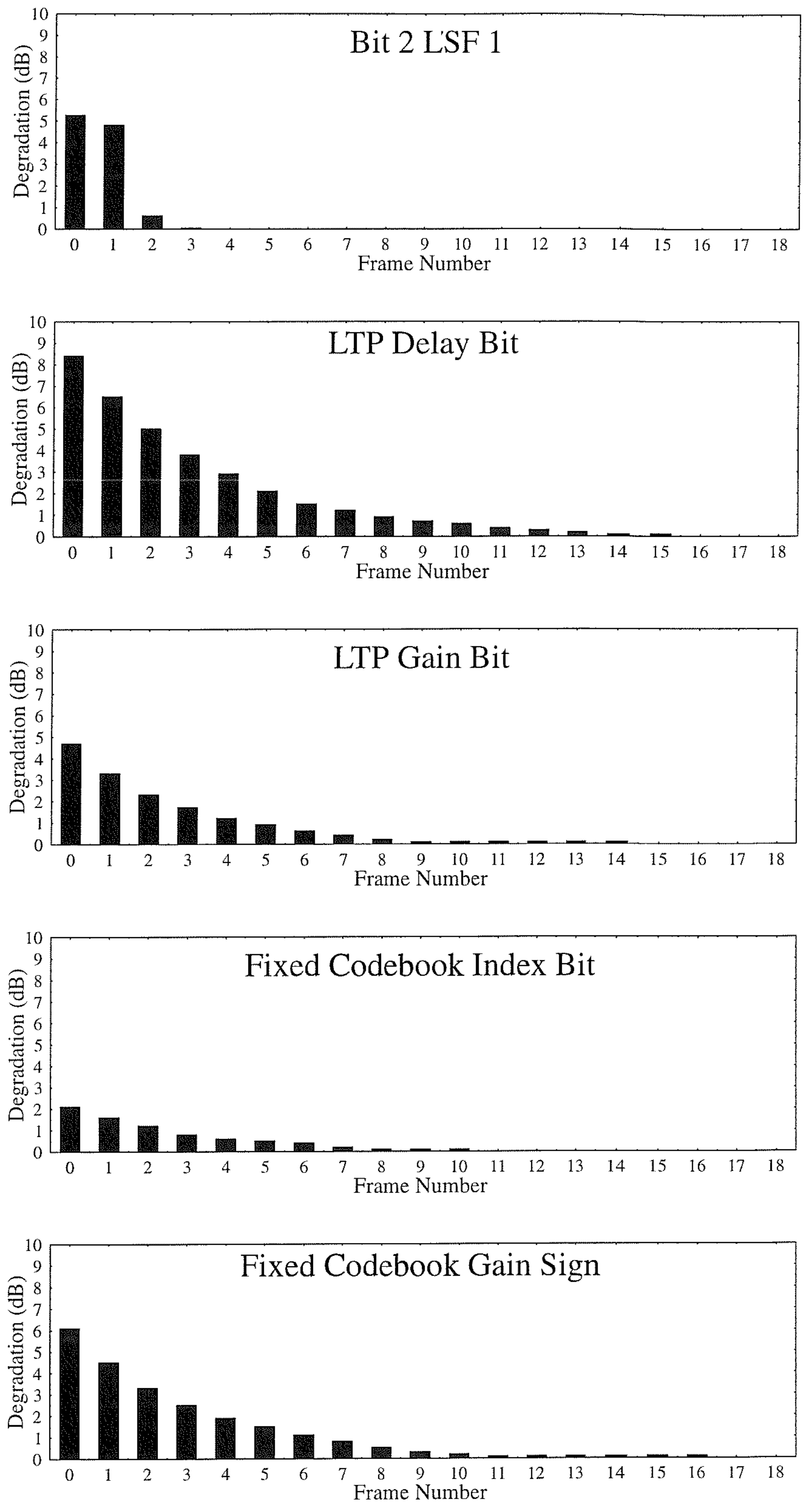

Figure 5.5: The SNR Degradation Propagation For Various Bits

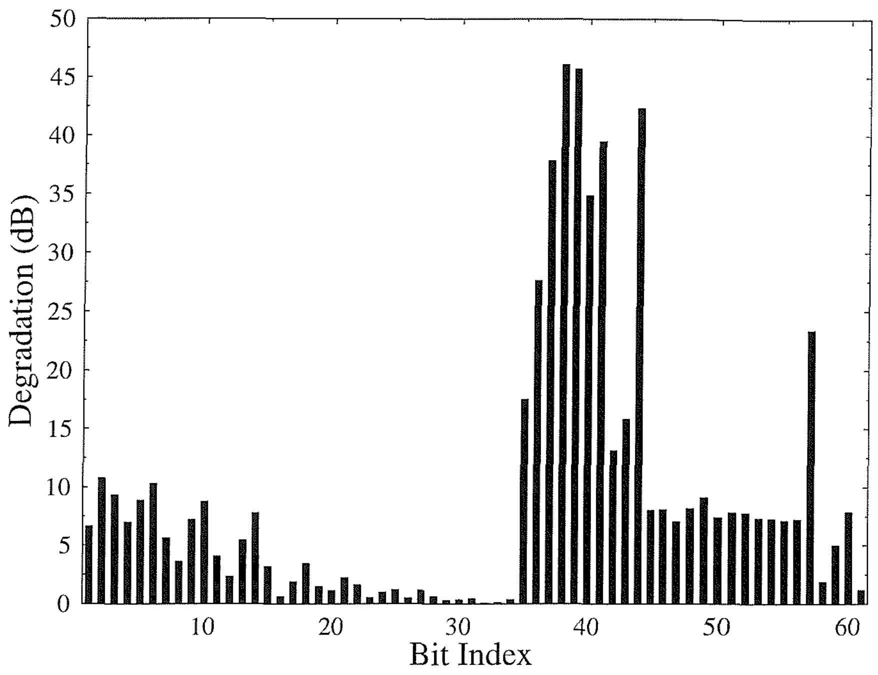

Figure 5.6: The Total SNR Degradation Due To Single Errors In Various Bits

properties of different bits can be clearly seen. For example an error in a bit representing an LSF has a significant effect only in the frame in which the error occurred and in the next two frames. Conversely an error in a bit representing the LTP delay gives a large degradation in the frame SNR, and this degradation is still significant 10 frames later. Figure 5.6 shows the total SNR degradation for single bit errors of the various bits. This graph is significantly different to that in Figure 5.3, in particular the importance of the adaptive codebook delay bits, because of their memory propagation properties, is much clearer.

Our error sensitivity figure is based on the total SNR degradation described above and on a similar measure for the total CD degradation. The two sets of degradation figures are combined and given equal weight by scaling each total SNR degradation by the maximum such degradation, and similarly for the total CD figures. The two sets of scaled degradation figures are then added together to give an overall sensitivity figure between 0 and 2. The higher this figure is the more sensitive the bit is deemed to be.

Our new scheme was tested as follows. The twelve most sensitive bits were determined using our scheme and that reported in [56]. These two sets of twelve bits contained four in common, which were removed to give two sets of eight bits. The two different sets were corrupted at a 5% bit error rate for various different speech files, and in all cases we found that both objectively (CD and SNR degradations), and in informal listening tests, the bits our scheme predicted would be most sensitive were much more sensitive than those predicted using the approach in [56].

## 5.5 Conclusion

In this chapter we have discussed the error sensitivity of the forward adaptive ACELP codec described in earlier chapters. We investigated various ways of improving the error sensitivity of the codec, and how the sensitivity of different bits could be compared in order to correctly match a channel coder to the speech coder. We have also shown how the degradations produced by errors propagate from one frame to another, and may persist for more than ten frames, and how the sensitivity of a given bit can vary significantly from frame to frame.

The error sensitivity improvement and measurement techniques we have described in this chapter were used to match our 4.7 kbits/s speech codec with a set of BCH error-correcting codes. The speech and error-correction codecs were used with 16-level QAM and a Packet Reservation Multiple-Access (PRMA) scheme to simulate a complete multiple-user mobile communication system. Similar studies were also carried out for a 6.5 kbits/s codec, which was identical to the 7.1 kbits/s codec described in Chapter 3 except it used six 5 ms sub-frames to make up a 30 ms frame instead of using four sub-frames per 20 ms frame. This extension of the frame length of the higher rate codec to be equal to the frame length of the low-rate codec was carried out for reasons of ease of implementation of the PRMA scheme. These simulations and our results are described in [63, 64].

# Chapter 6

# A Variable Rate Low Delay CELP Codec

In this chapter our work moves on from the forward adaptive, high delay, algebraic CELP codecs described earlier to higher rate and higher quality low delay codecs. In the next section we discuss why the delay of a speech codec is an important parameter, methods of achieving low delay coding and problems with these methods. Much of the work in this section is based on the recently standardised 16 kbits/s G728 Low Delay CELP codec [65, 3], and this is described in Section 6.2. We then describe our attempts to extend the G728 codec to produce a low delay, variable bit rate codec operating between 8 kbits/s and 16 kbits/s. In Section 6.4 we describe the improvements that can be achieved in such a codec by adding a Long Term Predictor, and in Section 6.5 we discuss means of training the codebooks used in our variable rate codec to optimise its performance. Section 6.6 describes an alternative variable rate codec which has a constant vector size. Finally in Section 6.7 we describe the postfiltering which is used to improve the perceptual quality of our codecs.

## 6.1 Introduction

The delay of a speech codec can be an important parameter for several reasons. In the public switched telephone network 4 to 2 wire conversions lead to echoes, which will be subjectively annoying if the echo is sufficiently delayed. Even if echo cancellers are used, a high delay speech codec makes the echo cancellation more difficult. Therefore if a codec is to be connected to the telephone network it is desirable that its delay should be as low as possible. Also the total delay introduced by any communications network must be below a certain limit, around 0.1s, before it becomes noticeable to the users of the network. When designing a communications system the delay will be kept below this limit if possible, and if the speech codec used has a low delay then other elements of the system, such as bit inter-leavers, will have more flexibility and should be able to improve the overall quality of the system.

The one-way coding delay of a speech codec is defined as the time from when a sample arrives at the input of the encoder to when the corresponding sample is produced at the output of the decoder, assuming the bit-stream from the encoder is fed directly to the decoder. This one-way delay is typically made up of three main components [3]. The first is the algorithmic buffering delay of the codec - the encoder operates on frames of speech, and must buffer a frame-lengths worth of speech samples before it can start encoding. The second component of the overall delay is the processing delay - speech codecs typically operate in just real time, and so it takes almost one frame length in time to process the buffered samples. Finally there is the bit transmission delay - if the encoder is linked to the decoder by a channel with capacity equal to the bit rate of the codec then there will be a further time delay equal to the codec's frame length while the decoder waits to receive all the bits representing the current frame.

From the above description the overall one-way delay of the codec will be equal to about three times the frame length of the codec. However it is possible to reduce this delay by careful implementation of the codec. For example if a faster processor is used the processing delay can be reduced. Also it may not be necessary to wait until the whole speech frame has been processed before we can start sending bits to the decoder. Finally a faster communications channel, for example in a time division multiplexed system, can dramatically reduce the bit transmission delay. Other factors may also result in the total delay being increased. For example the one sub-frame look-ahead used to aid the interpolation of the LSFs in our ACELP codecs described earlier will increase the overall delay by one sub-frame. Nonetheless, typically the one-way coding delay of a speech codec is assumed to be about 2.5 to 3 times the frame length of the codec.

It is obvious from the discussion above that the most effective way of producing a low delay speech codec is to use as short a frame length as possible. Traditional CELP codecs have a frame length of 20 to 30 ms, leading to a total coding delay of at least 50 ms. Such a long frame length is necessary because of the forward adaption of the short-term synthesis filter coefficients. As explained in Chapter 3 a frame of speech is buffered, LPC analysis is performed and the resulting filter coefficients are quantized and transmitted to the decoder. As we reduce the frame length, the filter coefficients must be sent more often to the decoder and so more and more of the available bit rate is taken up by LPC information. Although efficient speech windowing and LSF quantization schemes have allowed the frame length to be reduced to 10 ms (with a 5 ms look-ahead) in a candidate codec [66] for the CCITT 8 kbits/s standard, a frame length of between 20 and 30 ms is more typical. If we want to produce a codec with delay of the order of 2 ms, which was the objective for the CCITT 16 kbits/s codec [67], it is obvious that we cannot use forward adaption of the synthesis filter coefficients.

The alternative is to use backward adaptive LPC analysis. This means that rather than window and analyse present and future speech samples in order to derive the filter coefficients, we analyse previous quantized and locally decoded signals to derive the coefficients. These past quantized signals are available at both the encoder and decoder, and so no side information about the LPC coefficients needs to be transmitted. This allows us to update the filter coefficients as frequently as we like, with the only penalty being a possible increase in the complexity of the codec. Thus we can dramatically reduce the codec's frame length and delay.

As explained above backward adaptive LPC analysis has the advantages of allowing us to dramatically reduce the delay of our codec, and removing the information about the filter coefficients that must be transmitted. This side information is usually about 25 % of the bit rate of a codec, and so it is very helpful if it can be removed. However backward adaption has the disadvantage that it produces filter coefficients which are typically degraded in comparison to those used in forward adaptive codecs. The degradation in the coefficients comes from two sources [68]:

1. Noise Feedback - In a backward adaptive system the filter coefficients are derived from a quantized signal, and so there will be a feedback of quantization noise into the LPC analysis which will degrade the performance of the coefficients produced.

2. Time Mismatch - In a forward adaptive system the filter coefficients for the current frame are derived from the input speech signal for the current frame. In a backward adaptive system we have only signals available from previous frames to use, and so there is a time mismatch between the current frame and the coefficients we use for that frame.

The effects of noise feedback especially increases dramatically as the bit rate of the codec is reduced and means that traditionally backward adaption has only been used in high bit rate, high quality, codecs. However recently, as researchers have attempted to reduce the delay of speech codecs, backward adaptive LPC analysis has been used at bit rates as low as 4.8 kbits/s [69].

In the next section we describe the 16 kbits/s G728 low delay CELP codec, and in particular the ways it differs from the ACELP codecs we have used previously. We also attempt to quantify the effects of both noise feedback and time mismatch on the backward adaptive LPC analysis used in this codec.

## 6.2 The G728 16 kbits/s Low Delay CELP Codec

Block diagrams of the G728 encoder and decoder are shown in Figures 6.1 and 6.2. This codec operates on vectors of 5 speech samples, which are equivalent to the 40 or 60 sample sub-frames used in our ACELP codecs. However, as the LPC analysis in G728 is backward adaptive, there is no further buffering and so its algorithmic buffering delay is 0.625 ms, and its total delay will typically be less than 2ms. At 16 kbits/s there are 10 bits which can be used to represent each five sample vector, and as no LPC side information is needed these bits can be used entirely to quantize the excitation signal $u(n)$. A seven bit excitation shape codebook, equivalent to the 12 bit algebraic codebooks in our ACELP codecs, is used together with a 3 bit gain quantizer and backward gain adaption [70]. The shape and gain codebook indices are chosen in a closed-loop Analysis-by-Synthesis type search to minimise the weighted error between the reconstructed speech $\hat{s}(n)$ and the input speech $s(n)$.

The various parts of the G728 codec, and their performances, are described in more detail below. All the codec performance figures quoted are averaged over four different sentences, two of which are spoken by males and two by females. Each sentence is between two and three seconds long.

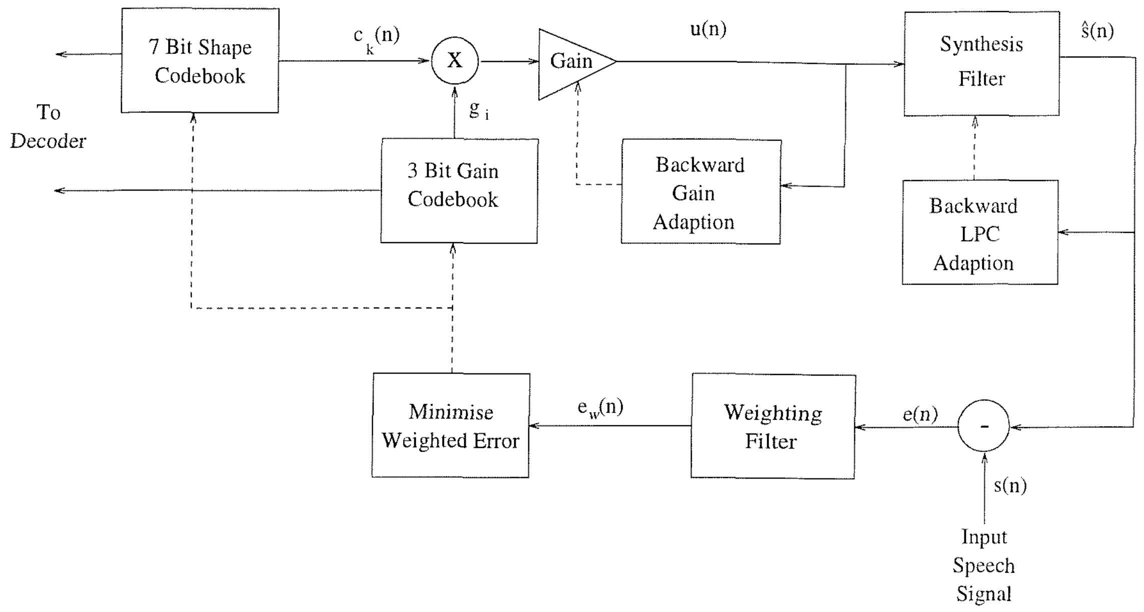

Figure 6.1: G728 Encoder

## 6.2.1 The Synthesis Filter and Backward LPC Adaption

The basis of the G728 codec is a high order all-pole backward adapted synthesis filter. As can be seen from Figures 6.1 and 6.2, no long term predictor or adaptive codebook is used. Instead a synthesis filter of order 50 is used to introduce long-term as well as short-term correlations into the reconstructed speech signal $\hat{s}(n)$. As no LPC side information needs to be transmitted, the only penalty in using such a high order filter is the resultant increase in the codec's complexity.

The synthesis filter coefficients are updated every four vectors, or 20 samples, from the previous reconstructed speech signal $\hat{s}(n)$. A hybrid recursive window, as shown in Figure 6.3, is used to window $\hat{s}(n)$ and find its autocorrelation values $R(0), R(1), \cdots R(50)$. The 'optimum' set of filter coefficients $a(k)$ is then computed from

$$\sum_{k=1}^{50} R(\mid i-k \mid)\, a_k \;=\; R(i) \quad \text{for i} = 1, 2 \cdots 50. \qquad (6.1)$$

This equation is solved using the Levinson-Durbin algorithm. Finally, to improve the performance of the codec over noisy channels, a 30Hz bandwidth expansion is applied to give the filter coefficients which are used for the next 4 vectors.

As can be seen from Figure 6.3 the windowing of the previous reconstructed speech samples places most weight on recent samples. The first 35 values of the windowing

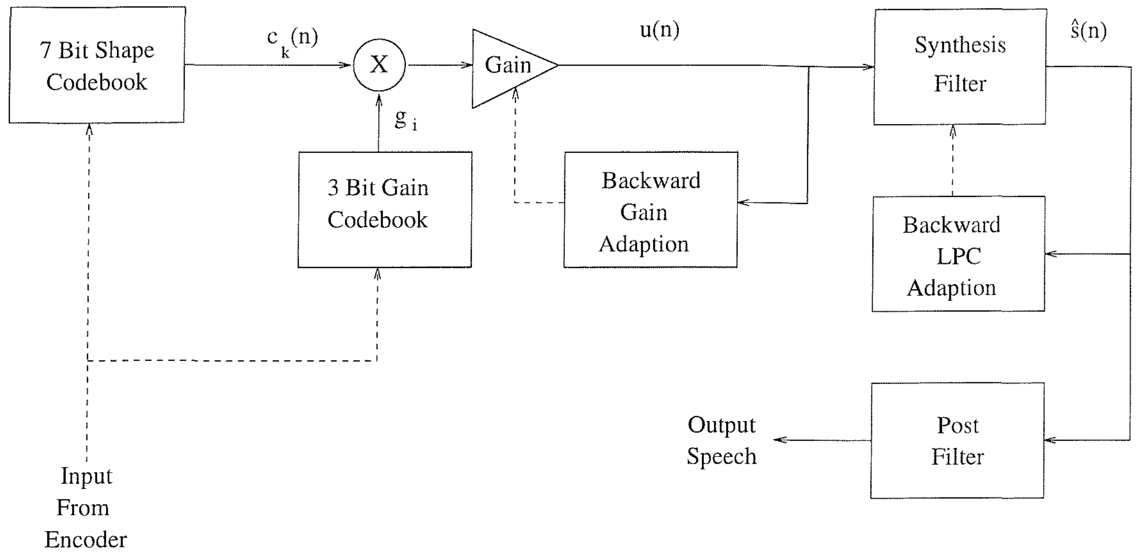

Figure 6.2: G728 Decoder

| Filter Order $p$ | $\Delta$ Prediction Gain (dB) | $\Delta$ Codec Segmental SNR (dB) |
|---|---|---|
| 10 | 0.0 | 0.0 |
| 25 | +0.68 | +0.70 |
| 50 | +1.05 | +1.21 |
| 75 | +1.12 | +1.41 |
| 100 | +1.11 | +1.46 |
| 150 | +1.10 | +1.42 |

Table 6.1: Relative Performance of the Synthesis Filter as $p$ is Increased

function are non-recursive, but for $n = -36, -37, -38, \cdots$ its values are given by $\beta\alpha^{-(n+L+1)}$ where $\alpha$ and $\beta$ are constants between 0 and 1, and $L$ is the length of the non-recursive section of the window. The window is effectively of infinite length, but due to its recursive nature it allows the autocorrelation values to be calculated very efficiently [65, 71].

The performance of the synthesis filter, in terms of its prediction gain and the segmental SNR of the G728 codec using this filter, is shown against the filter order $p$ in Figure 6.4 for a single sentence spoken by a female. Also shown in Table 6.1 is the increase in performance obtained when $p$ is increased above 10, which is the value most commonly used in AbS codecs. It can be seen that there is a significant performance gain due to increasing the order from 10 to 50, but little additional gain is achieved as $p$ is further increased.

We also tested the degradations in the synthesis filter's performance at $p = 50$ due

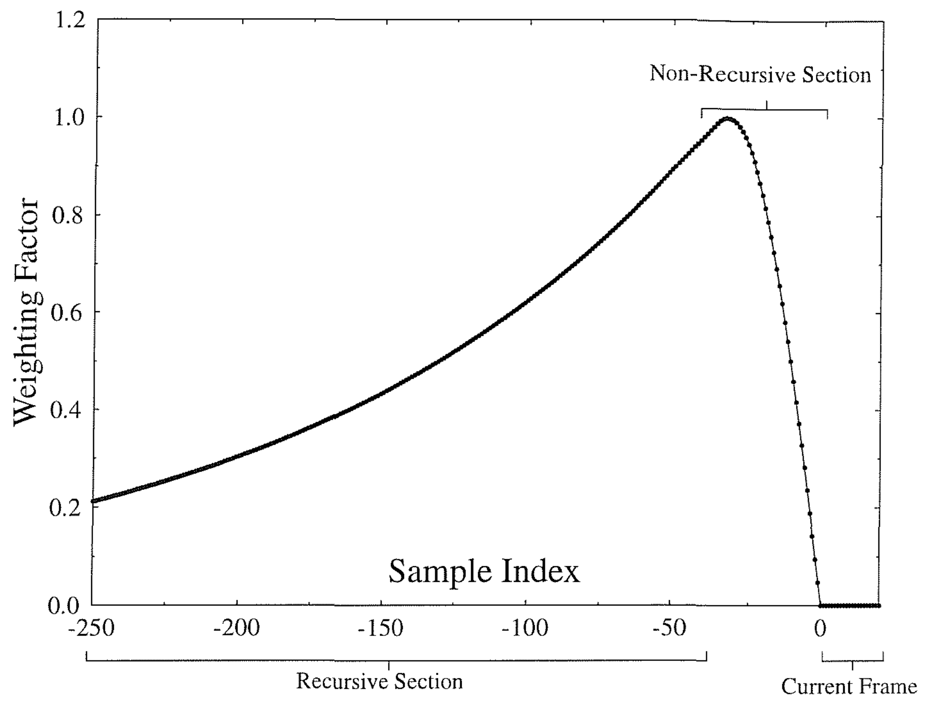

Figure 6.3: Windowing Function Used in the Backward Adaption of the Synthesis Filter

to backward adaption being used. This was done as follows. To measure the effect of quantization noise feedback we updated the synthesis filter parameters exactly as in G728 except we used the previous speech samples rather than the previous reconstructed speech samples. To measure the overall effect of backward adaption we updated the synthesis filter using both past and present speech samples. The improvements obtained, in terms of the segmental SNR of the codec and the filter's prediction gain, are shown in Table 6.2. We see that due to the high SNR of the G728 codec noise feedback has relatively little effect on the performance of the synthesis filter. The time-mismatch gives a more significant degradation in the codec's performance. Note however that the forward adaptive figures given in Table 6.2 could not be obtained in reality because they do not include any effects of the LPC quantization that must be used in a real forward adaptive system.

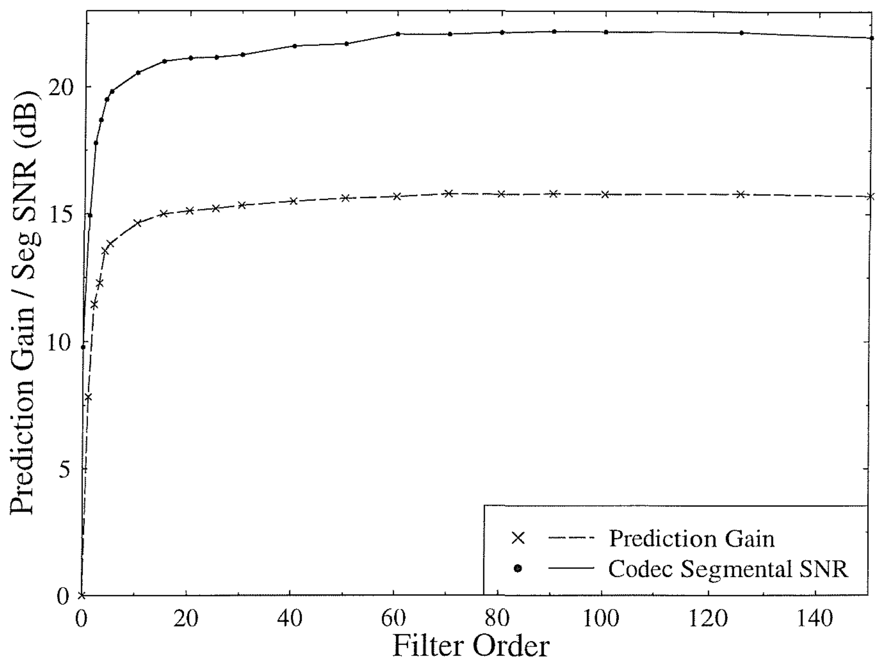

Figure 6.4: Performance of the Synthesis Filter in a G728-Like Codec

| | Δ Prediction Gain (dB) | Δ Codec Segmental SNR (dB) |
|---|---|---|
| No Noise Feedback | +0.50 | +0.18 |
| No Time Mismatch | +0.74 | +0.73 |
| Use Forward Adaption | +1.24 | +0.91 |

Table 6.2: Effects of Backward Adaption of the Synthesis Filter

### 6.2.2 Backward Gain Adaption

The performance of the gain quantization of the excitation vectors is enhanced by the backward gain adaption [70], as shown in Figures 6.1 and 6.2. This operates as follows. For each vector a predicted gain value $\hat{\sigma}$ is found, and the excitation signal $u(n)$ for the synthesis filter is then given by

$$u(n) = \hat{\sigma} g_i c_k(n) \qquad n = 0 \cdots vs - 1 \tag{6.2}$$

where $vs$ is the vector size in samples (5), $g_i$ $i = 1 \cdots 8$ is the 3-bit quantized gain, and $c_k(n)$ $k = 1 \cdots 128$ is the $k$'th entry from the 7 bit shape codebook. The predicted gain $\hat{\sigma}$ is found using an adaptive tenth order linear prediction filter in the logarithmic domain operating on values of the previous vectors' actual excitation gains $\sigma$. These excitation gains $\sigma$ are defined as the RMS of the excitation signal, ie

$$\sigma = \sqrt{\frac{1}{vs}\sum_{n=0}^{vs-1} u^2(n)} \tag{6.3}$$

where $u(n)$ is the quantized excitation as given in Equation 6.2. The gain predictor itself is adapted once every four vectors or 20 samples using backward adaption on the previous values of the excitation gains in the logarithmic domain. A hybrid-recursive window very similar to that shown in Figure 6.3 is used to find a set of autocorrelation values, which are then used by the Levinson-Durbin algorithm to give a set of predictor coefficients. A bandwidth expansion of 250Hz is then applied to make the gain adaption more robust to channel errors.

The effectiveness of the backward gain adaption can be seen from Figure 6.5 . This shows the PDFs, on a log scale for clarity, of the excitation vector's optimum gain both with and without gain adaption. Here the optimum vector gain is defined as

$$\sqrt{\frac{1}{vs}\sum_{n=0}^{vs} g^2 c_k^2(n)} \tag{6.4}$$

where $g$ is the unquantized gain chosen in the codebook search. For a fair comparison both PDFs were normalised to have a mean of one. It can be seen that gain adaption produces a PDF which peaks around one and has a shorter tail and a reduced variance. This makes the quantization of the excitation vectors significantly easier. Shown in Figure 6.6 are the PDFs of the optimum unquantized codebook gain $g$, and its quantized value, when backward gain adaption is used. It can be seen that most of the codebook gain values have a magnitude less than or close to one, but it is still necessary to allocate two gain quantizer levels for the infrequently used high magnitude gain values.

By training a split 7/3 bit shape/gain codebook, as described in Section 6.5, for G728-like codecs both with and without gain adaption we found that the gain adaption increased the segmental SNR of the codec by 2.7 dB, and the weighted segmental SNR by 1.5 dB. These are very significant improvements, especially when it is considered that the gain adaption increases the encoder complexity by only about 3%.

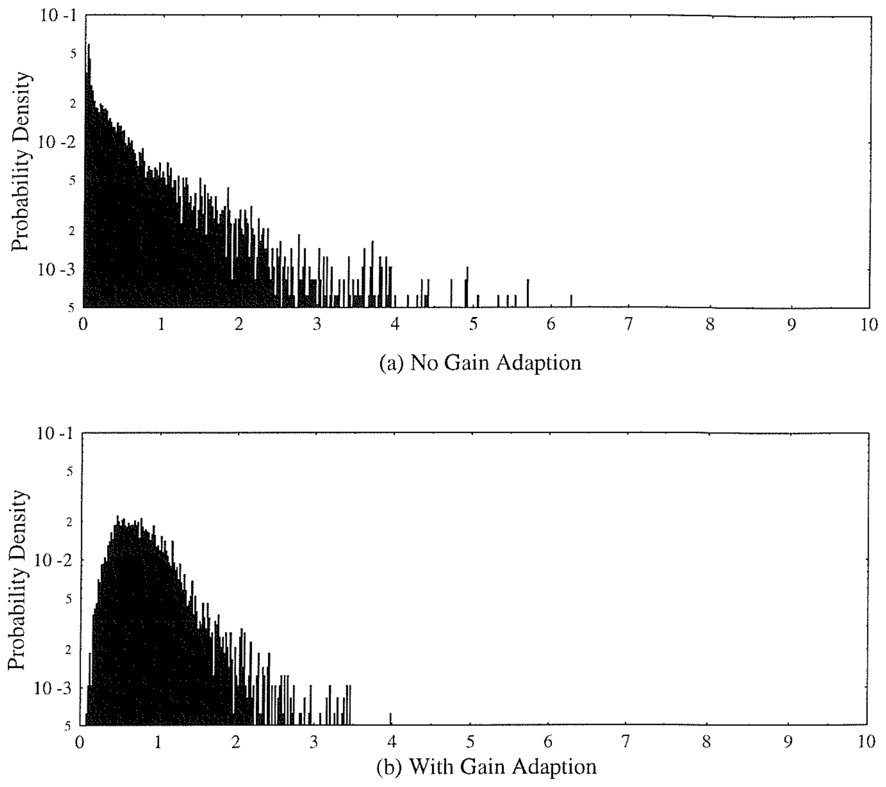

Figure 6.5: PDFs of the Normalised Codebook Gains With and Without Backward Gain Adaption

## 6.2.3 The Weighting Filter

The G728 encoder uses a weighting filter to improve the perceptual quality of the reconstructed speech by emphasising noise in the frequency regions where the speech has low energy, and de-emphasising noise in the formant regions. The weighting filter $W(z)$ has the form

$$W(z) = \frac{1 - \sum_{k=1}^{10} a_k \gamma_1^k}{1 - \sum_{k=1}^{10} a_k \gamma_2^k} \tag{6.5}$$

where $a_k$ are filter coefficients derived through LPC analysis of the input speech, and $\gamma_1$ and $\gamma_2$ are constants which control the amount of weighting. The values $\gamma_1 = 0.9$ and $\gamma_2 = 0.6$ are used. A filter order of 10 rather than 50 was used because a 50th order weighting filter was found to cause artifacts in the synthesised speech [3].

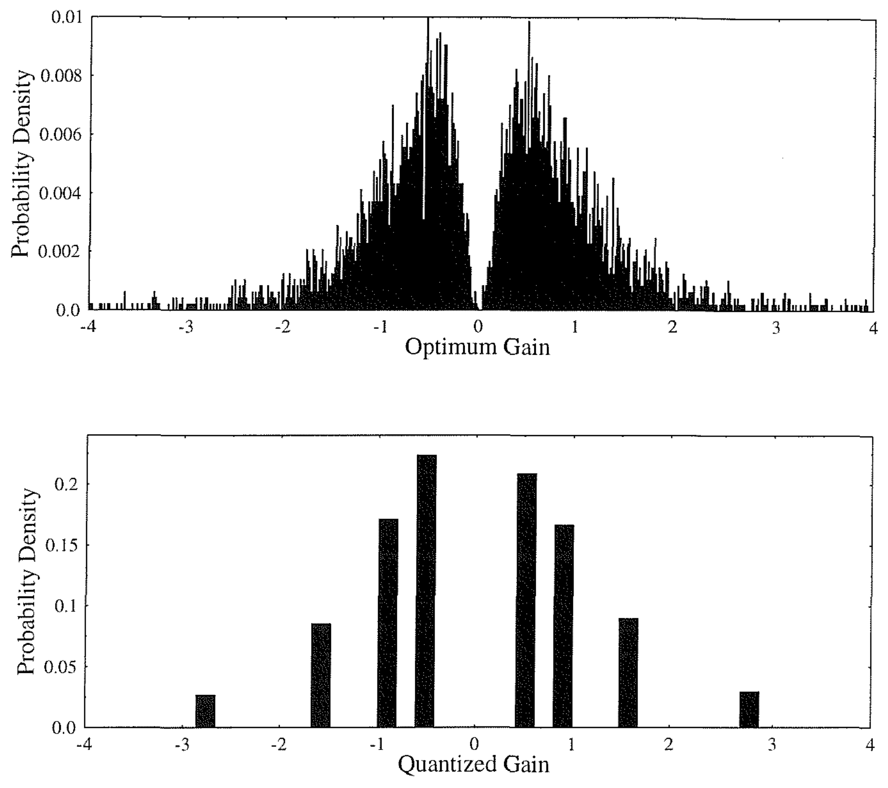

Figure 6.6: PDFs of the Optimum and Quantized Codebook Gain Values

### 6.2.4 Excitation Vector Quantization

At 16 kbits/s there are 10 bits which can be used to represent every 5 sample vector, and as the LPC analysis is backward adaptive these bits are used entirely to code the excitation signal $u(n)$ which is fed to the synthesis filter. The 5 sample excitation sequences are vector quantized using a 10 bit split shape-gain codebook. Seven bits are used to represent the vector shapes, and the remaining 3 bits are used to quantize the vector gains. This splitting of the 10 bit vector quantizer is done to reduce the complexity of the closed-loop codebook search. To measure the degradations that were introduced by this splitting we trained codebooks for a 7/3 bit shape/gain split vector quantizer, and a pure 10 bit vector quantizer. We found that the 10 bit vector quantizer gave no significant improvement in either the segmental SNR or the segmental weighted SNR of the codec, and increased the complexity of the codebook search by about 550% and the overall codec complexity by about 300%. Hence this splitting of the vector quantizer is a very efficient way to significantly reduce the complexity of the encoder.

The closed-loop codebook search is carried out as follows. For each vector the search procedure finds values of the gain quantizer index $i$ and the shape codebook index $k$ which minimise the squared weighed error $E_w$ for that vector. $E_w$ is given by

$$E_w = \sum_{n=0}^{vs-1} \left(s_w(n) - \hat{s}_o(n) - \hat{\sigma} g_i h(n) * c_k(n)\right)^2 \tag{6.6}$$

where $s_w(n)$ is the weighted input speech, $\hat{s}_o(n)$ is the zero-input response of the synthesis and weighting filters, $\hat{\sigma}$ is the predicted vector gain, $h(n)$ is the impulse response of the concatenated synthesis and weighting filters and $g_i$ and $c_k(n)$ are the entries from the gain and shape codebooks. This equation can be expanded to give

$$\begin{aligned}
E_w(n) &= \hat{\sigma}^2 \sum_{n=0}^{vs-1} \left(x(n) - g_i[h(n) * c_k(n)]\right)^2 \qquad (6.7)\\
&= \hat{\sigma}^2 \sum_{n=0}^{vs-1} x^2(n) \;+\; \hat{\sigma}^2 g_i^2 \sum_{n=0}^{vs-1} [h(n) * c_k(n)]^2 \;-\; 2\hat{\sigma}^2 g_i \sum_{n=0}^{vs-1} x(n)[h(n) * c_k(n)]\\
&= \hat{\sigma}^2 \sum_{n=0}^{vs-1} x^2(n) \;+\; \hat{\sigma}^2 \left(g_i^2 \xi_k - 2 g_i C_k\right)
\end{aligned}$$

where $x(n) = (s_w(n) - \hat{s}_o(n))/\hat{\sigma}$ is the codebook search target,

$$C_k = \sum_{n=0}^{vs-1} x(n)[h(n) * c_k(n)] \tag{6.8}$$

is the correlation between this target and the filtered codeword $h(n) * c_k(n)$, and

$$\xi_k = \sum_{n=0}^{vs-1} [h(n) * c_k(n)]^2 \tag{6.9}$$

is the energy of the filtered codeword $h(n) * c_k(n)$. Note that this is almost identical to the form of the term in Equation 3.25 which must be minimised in the fixed codebook search in our ACELP codecs.

In the G728 codec the synthesis and weighting filters are changed only once every four vectors. Hence $\xi_k$ must be calculated for the 128 codebook entries only once every four vectors. The correlation term $C_k$ can be rewritten as

$$\begin{aligned}
C_k &= \sum_{n=0}^{vs-1} x(n)[h(n) * c_k(n)] \qquad (6.10)\\
&= \sum_{n=0}^{vs-1} c_k(n)\psi(n)
\end{aligned}$$

where

$$\psi(n) = \sum_{i=n}^{vs-1} x(i)h(i-n) \tag{6.11}$$

is the reverse convolution between $h(n)$ and $x(n)$. This means that we need to carry out only one convolution operation for each vector to find $\psi(n)$ and then we can find $C_k$ for each codebook entry $k$ with a relatively simple series of multiply-add operations.

The codebook search finds the codebook entries $i$=1-8 and $k$=1-128 which minimise $E_w$ for the vector. This is equivalent to minimising

$$D_{ik} = g_i^2 \xi_k - 2g_i C_k. \tag{6.12}$$

For each codebook entry $k$, $C_k$ is calculated and then the best quantized gain value $g_i$ is found. The values $g_i^2$ and $2g_i$ are pre-computed and stored for the 8 quantized gains, and these values along with $\xi_k$ and $C_k$ are used to find $D_{ik}$. The codebook index $k$ which minimises this, together with the corresponding gain quantizer level $i$, are sent to the decoder. These indices are also used in the encoder to produce the excitation and reconstructed speech signals which are used to update the gain predictor and the synthesis filter.

### 6.2.5 The Complexity and Performance of the G728 Codec

In the previous sub-sections we have described the operation of the G728 codec, except for its post-filter which is described in Section 6.7. The associated complexities of the various sections of the codec are shown in Table 6.3 in terms of millions of arithmetic operations (mostly multiplies and adds) per second. The weighting filter and codebook search operations are carried out only by the encoder, which requires a total of about 12.4 million operations per second. The post filtering is carried out only by the decoder which requires about 8.7 million operations per second. The full duplex codec requires about 21 million operations per second, and has been implemented on a single fixed point DSP (the Texas Instruments TMS320C50 for example).

We found that the codec gave an average segmental SNR of 20.1 dB, and an average weighted segmental SNR of 16.3 dB. The reconstructed speech was difficult to distinguish from the original, with no obvious degradations.

In the next section we discuss our attempts to modify the G728 codec to produce a variable bit rate 8-16 kbits/s codec which gives a graceful degradation in speech quality as the bit rate is reduced.

| | |
|---|---|
| Synthesis Filter | 5.1 |
| Backward Gain Adaption | 0.4 |
| Weighting Filter | 0.9 |
| Codebook Search | 6.0 |
| Post Filtering | 3.2 |
| Total Encoder Complexity | 12.4 |
| Total Decoder Complexity | 8.7 |

Table 6.3: Millions of Operations per Second Required by G728 Codec

## 6.3 Reducing the Bit Rate of a G728-Like Codec By Varying the Vector Size

Having detailed the G728 codec in the previous section we now describe our work in reducing the bit rate of this codec and producing an 8-16 kbits/s variable rate low-delay codec. The G728 codec uses 10 bits to represent each 5 sample vector. It is obvious that to reduce the bit rate of this codec we must either reduce the number of bits used for each vector, or increase the number of speech samples per vector. If we were to keep the vector size fixed at 5 samples then in an 8 kbits/s codec we would have only 5 bits to represent both the excitation shape and gain. Without special codebook training this leads to a codec with unacceptable performance. Therefore initially we concentrated on reducing the bit rate of the codec by increasing the vector size. In Section 6.6 we discuss the alternative approach of keeping the vector size constant and reducing the size of the codebooks used.

In this section at all bit rates we use a split 7/3 bit shape/gain vector quantizer for the excitation signal $u(n)$. The codec rate is varied by changing the vector size $vs$ used, from $vs = 5$ for the 16 bits/s codec to $vs = 10$ for the 8 kbits/s codec. For all the codecs we used the same 3 bit gain quantizer as in G728, and for the various shape codebooks we used randomly generated Gaussian codebooks with the same variance as the G728 shape codebook. Random codebooks with a Gaussian PDF were used for simplicity and because in the past such codebooks have been shown to give a relatively good performance [2]. We found that replacing the trained shape codebook in the G728 codec with a Gaussian codebook reduced the segmental SNR of the codec by 1.7 dB, and the segmental weighted SNR by 2 dB. However these losses in performance are recovered in Section 6.5 when we consider closed-loop training of our codebooks.

In the G728 codec the synthesis filter, weighting filter and the gain predictor are all updated every four vectors. With a vector size of 5 this means the filters are updated every 20 samples or 2.5ms. Generally the more frequently the filters are updated the better the codec will perform, and we found this to be true for our codec. However updating the filter coefficients more frequently significantly increases the complexity of the codec. Therefore we decided to keep the period between filter updates as close as possible to 20 samples as the bit rate of our codec is reduced by increasing the vector size. This means reducing the number of vectors between filter updates as the vector size is increased. For example at 8 kbits/s the vector size is 10 and we updated the filters every 2 vectors, which again corresponds to 2.5ms.

The segmental SNR of our codec against its bit rate as the vector size is increased from 5 to 10 is shown in Figure 6.7. Also shown in this figure is the segmental prediction gain of the synthesis filter at the various bit rates. It can be seen from this figure that the segmental SNR of our codec decreases smoothly as its bit rate is reduced, falling by about 0.8 dB for every 1 kbits/s drop in the bit rate.

As explained in the previous section, an important part of the codec is the backward adaptive synthesis filter. It can be seen from Figure 6.7 that the prediction gain of this filter falls by only 1.3 dB as the bit-rate of the codec is reduced from 16 to 8 kbits/s. This suggests that the backward adaptive synthesis filtering copes well with the reduction in bit rate from 16 to 8 kbits/s. We also carried out tests at 16 and 8 kbits/s, similar to those used for Table 6.2, to establish how the performance of the filter would be improved if we were able to eliminate the effects of using backward adaption ie the noise feedback and time mismatch. The results are shown in Tables 6.4 and 6.5 for the 16 kbits/s codec (using the Gaussian codebook rather than the trained G728 codebook used for Table 6.2 ) and the 8 kbits/s codec. As expected the effects of noise feedback are more significant at 8 than 16 kbits/s, but the overall effects on the codec's segmental SNR of using backward adaption are similar at both rates.

It has been suggested [68] that high order backward adaptive linear prediction is inappropriate at bit rates as low as 8 kbits/s. However we found that this was not the case for our codec and that increasing the filter order from 10 to 50 gave almost the same increase in the codec performance at 8 kbits/s as at 16 kbits/s. This is shown

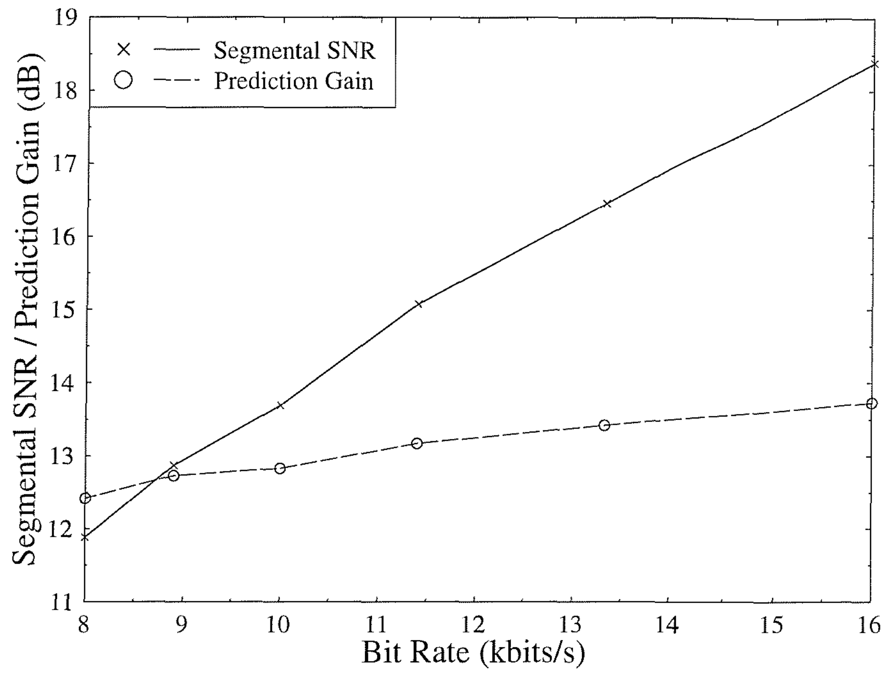

Figure 6.7: Performance of a Variable Rate G728 Like Codec

in Table 6.6.

Another important part of the G728 codec is the backward gain adaption. Figure 6.5 shows how at 16 kbits/s this backward adaption makes the optimum codebook gains cluster around one, and hence become easier to quantize. We found that the same was true at 8 kbits/s. To quantify the performance of the gain prediction we defined the following signal to noise ratio

$$SNR_{\text{gain}} = \frac{\sum \sigma_o^2}{\sum (\sigma_o - \hat{\sigma})^2}. \tag{6.13}$$

| | $\Delta$ Prediction Gain (dB) | $\Delta$ Codec Segmental SNR (dB) |
|---|---|---|
| No Noise Feedback | +0.74 | +0.42 |
| No Time Mismatch | +0.85 | +0.83 |
| Use Forward Adaption | +1.59 | +1.25 |

Table 6.4: Effects of Backward Adaption of the Synthesis Filter at 16 kbits/s

| | $\Delta$ Prediction Gain (dB) | $\Delta$ Codec Segmental SNR (dB) |
|---|---|---|
| No Noise Feedback | +2.04 | +0.75 |
| No Time Mismatch | +0.85 | +0.53 |
| Use Forward Adaption | +2.89 | +1.28 |

Table 6.5: Effects of Backward Adaption of the Synthesis Filter at 8 kbits/s

| | $\Delta$ Prediction Gain (dB) | $\Delta$ Codec Segmental SNR (dB) |
|---|---|---|
| 8 kbits/s p=10 | 0.0 | 0.0 |
| 8 bits/s p=50 | +0.88 | +1.00 |
| 16 kbits/s p=10 | 0.0 | 0.0 |
| 16 kbits/s p=50 | +1.03 | +1.04 |

Table 6.6: Relative Performance of the Synthesis Filter as $p$ is Increased at 8 and 16 kbits/s

Here $\sigma_o$ is the optimum excitation gain given by

$$\sigma_o = \sqrt{\frac{1}{vs} \sum_{n=0}^{vs} (\hat{\sigma} g c_k(n))^2} \tag{6.14}$$

where $g$ is the unquantized gain chosen by the codebook search and $\hat{\sigma}$ is the predicted gain value. We found that this gain prediction SNR was on average 5.3 dB for the 16 kbits/s codec, and 6.1 dB for the 8 kbits/s codec. Thus the gain prediction is even more effective at 8 kbits/s than at 16 kbits/s.

In the next section we discuss the addition of long term prediction to our variable rate codec.

## 6.4 The Effects of Long Term Prediction

In this section we describe the improvements in our variable rate codec that can be obtained by adding backward adaptive Long Term Prediction (LTP). This work was motivated by the fact that we found significant long term correlations remained in the synthesis filter's prediction residual, even when the pitch period was lower than the order of this filter. This can be seen from Figure 6.8, which shows the prediction residual for a segment of voiced female speech with a pitch period of about 45 samples. It can be seen that the residual has clear long term redundancies, which could be exploited by a long term prediction filter.

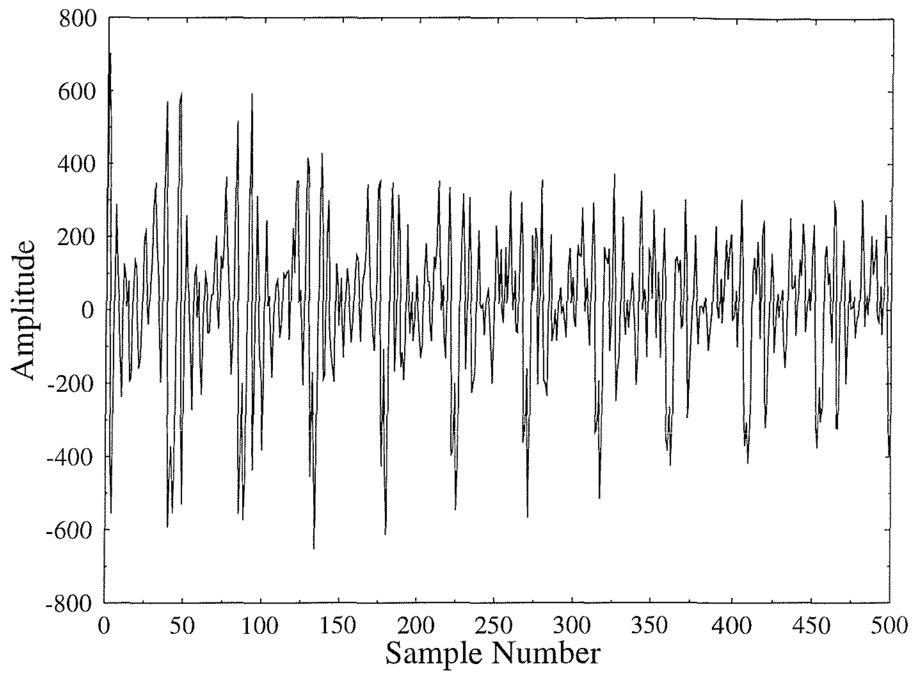

Figure 6.8: Synthesis Filter Prediction Residual in G728

In a forward adaptive system the short term synthesis filter coefficients are determined by minimising the energy of the residual signal found by filtering the original speech through the inverse synthesis filter. Similarly for open-loop LTP we minimise the energy of the long term residual signal which is found by filtering the short term residual through the inverse long term predictor. If $r(n)$ is the short term residual signal, then for a one tap long term predictor we want to determine the delay $L$ and gain $\beta$ which minimise the long term residual energy $E_{LT}$ given by

$$E_{LT} = \sum_n \left( r(n) - \beta r(n-L) \right)^2 . \tag{6.15}$$

The best delay $L$ is found by calculating

$$X = \frac{\left(\sum_n r(n) r(n-L)\right)^2}{\sum_n r^2(n-L)} \tag{6.16}$$

for all possible delays, and choosing the value of $L$ which maximises $X$. The best long term gain $\beta$ is then given by

$$\beta = \frac{\sum_n r(n)r(n-L)}{\sum_n r^2(n-L)}. \tag{6.17}$$

In a backward adaptive system the original speech signal $s(n)$ is not available, and instead we use the past reconstructed speech signal $\hat{s}(n)$ to find the short term synthesis filter coefficients. These coefficients can then be used to filter $\hat{s}(n)$ through the inverse filter to find the "reconstructed residual" signal $\hat{r}(n)$. This residual signal can then be used in Equations 6.16 and 6.17 to find the LTP delay and gain. Alternatively we can use the past excitation signal $u(n)$ in Equations 6.16 and 6.17. This approach is slightly simpler than using the reconstructed residual signal because the inverse filtering of $\hat{s}(n)$ to find $\hat{r}(n)$ is not necessary, and we found in our codec that the two approaches gave almost identical results.

Initially we used a one tap LTP in our codec. The best delay $L$ was found by maximising

$$X = \frac{\left(\sum_{n=-100}^{-1} u(n)u(n-L)\right)^2}{\sum_{n=-100}^{-1} u^2(n-L)} \tag{6.18}$$

over the range of delays 20 to 140 every frame. The LTP gain $\beta$ was updated every vector by solving

$$\beta = \frac{\sum_{n=-100}^{-1} u(n)u(n-L)}{\sum_{n=-100}^{-1} u^2(n-L)}. \tag{6.19}$$

We found that this backward adaptive LTP improved the average segmental SNR of our codec by 0.6 dB at 16 kbits/s, and 0.1 dB at 8 kbits/s. However the calculation of $X$ as given in Equation 6.18 for 120 different delays every frame dramatically increases the complexity of the codec. The denominator $\sum u^2(n-L)$ for delay $L$ need not be calculated independently, but instead can be simply updated from the equivalent expression for delay $L-1$. Even so if the frame size is 20 samples then to calculate $X$ for all delays increases both the encoder and the decoder complexity by almost 10 million arithmetic operations per second, which is clearly unacceptable.

Fortunately the G728 post-filter requires an estimate of the pitch period of the current frame. This is found by filtering the reconstructed speech signal through a tenth order short term prediction filter to find a reconstructed residual like signal. This signal is then low pass filtered with a cut-off frequency of 1 kHz and 4:1 decimated, which dramatically reduces the complexity of the pitch determination. The maximum value of the auto-correlation function of the decimated residual signal is then found to

| | Segmental SNR (dB) | Segmental Weighted SNR (dB) |
|---|---|---|
| No LTP | 18.43 | 14.30 |
| 1 Tap LTP | 19.08 | 14.85 |
| 3 Tap LTP | 19.39 | 15.21 |
| 5 Tap LTP | 19.31 | 15.12 |

Table 6.7: Performance of LTP at 16 kbits/s

| | Segmental SNR (dB) | Segmental Weighted SNR (dB) |
|---|---|---|
| No LTP | 11.86 | 8.34 |
| 1 Tap LTP | 12.33 | 8.64 |
| 3 Tap LTP | 12.74 | 9.02 |
| 5 Tap LTP | 12.49 | 8.81 |

Table 6.8: Performance of LTP at 8 kbits/s

give an estimate $\tau_d$ of the pitch period. A more accurate estimate $\tau_p$ is then found by maximising the autocorrelation function of the undecimated residual between $\tau_d - 3$ and $\tau_d + 3$. This lag could be a multiple of the true pitch period, and to guard against this possibility the autocorrelation function is also maximised between $\tau_o - 6$ and $\tau_o + 6$, where $\tau_o$ is the pitch period from the previous frame. Finally the pitch estimator chooses between $\tau_p$ and the best lag around $\tau_o$ by comparing the optimal tap weights $\beta$ for these two delays.

This pitch estimation procedure requires only about 2.6 million arithmetic operations per second, and is carried out at the decoder as part of the post-filtering operations anyway. So using this method to find a LTP delay has no effect on the decoder complexity, and increases the encoder complexity by only 2.6 million arithmetic operations per second. We also found that not only was this method of calculating the LTP delay much simpler than finding the maximum value of $X$ from Equation 6.18 for all delays between 20 and 140, it also gave better results. This was due to the removal of pitch doubling and tripling by the checking of pitch values around that used in the previous frame. The average segmental SNR and segmental weighted SNR for our codec at 16 kbits/s both with and without one tap LTP using the pitch estimate from the post-filter is shown in Table 6.7. Similar figures for the codec at 8 kbits/s are given in Table 6.8. We found that when LTP was used, there was very little gain in having a filter order any higher than 20. Therefore the figures in Tables 6.7 and 6.8 have a short term filter order of 20 when LTP is used.

Tables 6.7 and 6.8 also give the performance of our codec at 16 and 8 kbits/s when we use multi-tap LTP. As the LTP is backward adaptive we can use as many taps in the filter as we like, with the only penalty being a slight increase in complexity. Once the delay is known, for a $(2p+1)$'th order predictor the filter coefficients $b_{-p}, b_{-p+1}, \cdots, b_0, \cdots, b_p$ are given by solving the following set of simultaneous equations

$$\sum_{j=-p}^{j=p} b_j \sum_{n=-100}^{n=-1} u(n-L-j)u(n-L-i) = \sum_{n=-100}^{-1} u(n)u(n-L-i) \tag{6.20}$$

for $i = -p, -p+1, \cdots, p$. The LTP synthesis filter $H_{LTP}(z)$ is then given by

$$H_{LTP}(z) = \frac{1}{1 - b_{-p}z^{-L+p} - \cdots - b_0 z^{-L} - \cdots b_p z^{-L-p}}. \tag{6.21}$$

It can be seen from Tables 6.7 and 6.8 that at both 16 and 8 kbits/s the best performance is given by a 3 tap filter which improves the segmental SNR at both bit rates by almost 1 dB. Also because when LTP is used the short term synthesis filter order was reduced to 20, the complexity of the codecs is not significantly increased by the use of a long term prediction filter.

We found that it was possible to slightly increase the performance of the codec with LTP by modifying the signal $u(n)$ used to find the filter coefficients in Equation 6.20. This modification involves simply repeating the previous vector's excitation signal once. Hence instead of using the signal $u(-1), u(-2), \cdots, u(-100)$ to find the LTP coefficients, we use $u(-1), u(-2), \cdots, u(-vs), u(-1), u(-2), \cdots, u(-100+vs)$. This single repetition of the previous vector's excitation in the calculation of the LTP coefficients increased both the segmental and the weighted SNR of our codec at 16 kbits/s by about 0.25 dB. It also improved the codec performance at 8 kbits/s, although only by about 0.1 dB. The improvements that this repetition brings in the codec's performance seem to be due to the backward adaptive nature of the LTP - no such improvement is seen when a similar repetition is used in a forward adaptive system.

Shown in Figure 6.9 is the variation in the codec's segmental SNR as the bit rate is reduced from 16 to 8 kbits/s. The codec uses 3 tap LTP with the repetition scheme described above and a short term synthesis filter of order 20. Also shown in this figure is the equivalent variation in segmental SNR for the codec without LTP, repeated here from Figure 6.7. It can be seen that the addition of long term prediction to the codec

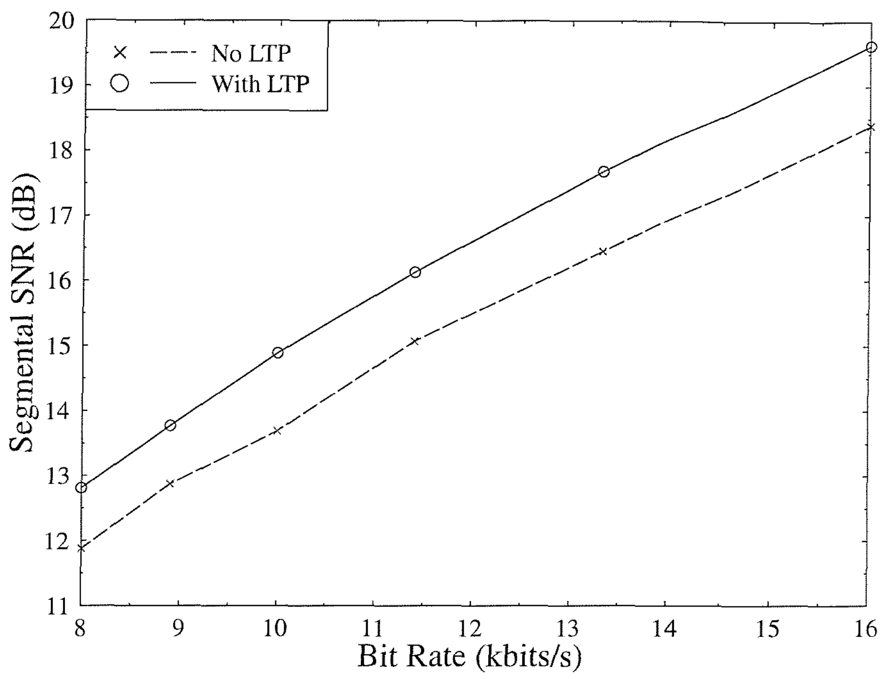

Figure 6.9: Performance of a 8-16 kbits/s Low Delay Codec With LTP

gives a uniform improvement in its segmental SNR of about 1 dB from 8 to 16 kbits/s. The effectiveness of the LTP can also be seen from Figure 6.10 which shows the long term prediction residual, in the 16 kbits/s codec, for the same segment of speech as was used for the short term prediction residual in Figure 6.8. It is clear that the long term correlations have been significantly reduced. It should be noted however that the addition of backward adapted long term prediction to the codec will degrade its performance over noisy channels. This aspect of our codec's performance is the subject of ongoing work [72].

Finally we tested the degradations in the performance of the long term prediction due to backward adaption being used. To measure the effect of quantization noise feedback we used past values of the original speech signal rather than the reconstructed speech signal to find the LTP delay and coefficients. To measure the overall effect of backward adaption as opposed to open-loop forward adaption we used both past and present speech samples to find the LTP delay and coefficients. The improvements

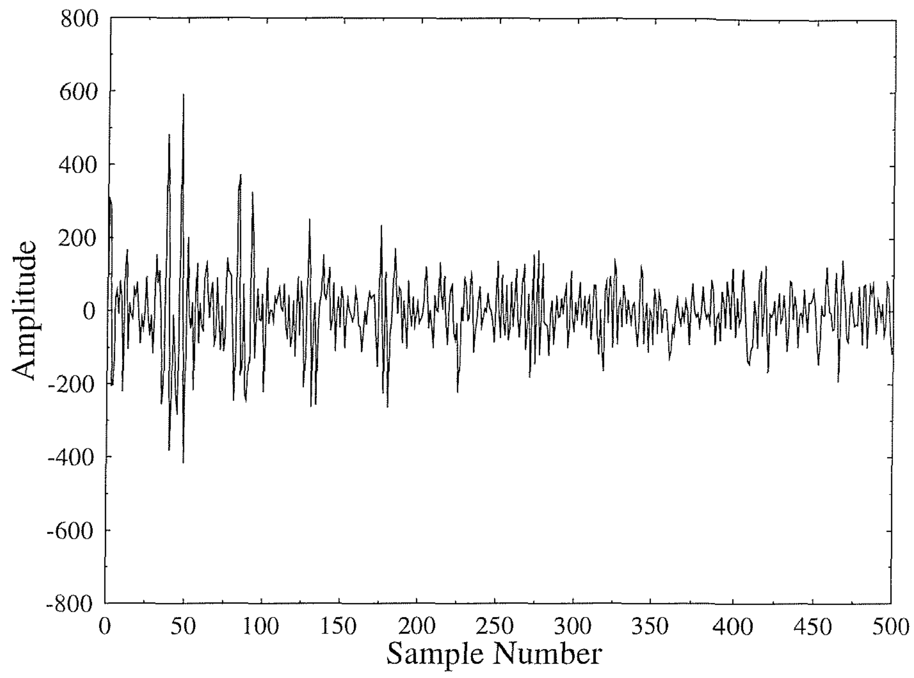

Figure 6.10: Long Term Filter Prediction Residual at 16 kbits/s

obtained in terms of the segmental SNR and the segmental weighted SNR are shown in Table 6.9 for the codec at 16 kbits/s, and Table 6.10 for the codec at 8 kbits/s. It can be seen that the use of backward adaption degrades the codecs performance by just under 1 dB at 16 kbits/s, and just over 1 dB at 8 kbits/s. At both bit rates noise feedback has very little effect, with most of the degradation coming from the time mismatch inherent in backward adaption.

| | $\Delta$ Segmental Weighted SNR (dB) | $\Delta$ Segmental SNR (dB) |
|---|---|---|
| No Noise Feedback | -0.03 | +0.01 |
| No Time Mismatch | +0.87 | +0.85 |
| Use Forward Adaption | +0.84 | +0.86 |

Table 6.9: Effects of Backward Adaption of the LTP at 16 kbits/s

| | Δ Segmental Weighted SNR (dB) | Δ Segmental SNR (dB) |
|---|---|---|
| No Noise Feedback | -0.18 | +0.02 |
| No Time Mismatch | +1.17 | +1.17 |
| Use Forward Adaption | +0.99 | +1.19 |

Table 6.10: Effects of Backward Adaption of the LTP at 8 kbits/s

## 6.5 Closed-Loop Codebook Training

In this section we describe the training of the shape and gain codebooks used in our codec at its various bit rates. In Sections 6.3 and 6.4 Gaussian shape codebooks were used, together with the G728 gain codebook. These codebooks were used for simplicity, and in order to provide a fair comparison between the different coding techniques used.

Due to the backward adaptive nature of the gain and synthesis filter and LTP adaption used in our codec it is not sufficient to generate a training sequence for the codebooks and use the Lloyd algorithm [73] to design the codebooks. This is because the codebook entries required from the shape and gain codebooks depend very much upon the effectiveness of the gain adaption and the LTP and synthesis filters used. However, because these are backward adapted, they depend on the codebook entries that have been selected in the past. Therefore the effective training sequence needed changes as the the codebooks are trained. Thus it is reported in [70] for example that in a gain-adaptive vector quantization scheme unless the codebook is properly designed, taking into account the gain adaption, the performance is worse than simple non-adaptive vector quantization.

We used a closed-loop codebook design algorithm similar to that described in [74]. A long speech file consisting of four sentences spoken by two males and two females is used for the training. Both the sentences spoken and the speakers are different from those used for the performance figures quoted in this chapter. The training process commences with an initial shape and gain codebook and codes the training speech as usual. The total weighted error $E_k$ from all the vectors that used the codebook entry $c_k(n)$ is then given by

$$E_k = \sum_{m \in N_k} \left( \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} (x_m(n) - g_m[h_m(n) * c_k(n)])^2 \right) \tag{6.22}$$

where $N_k$ is the set of vectors that use $c_k(n)$, $\hat{\sigma}_m$ is the backward adapted gain for vector $m$, $g_m$ is the gain codebook entry selected for vector $m$ and $h_m(n)$ is the impulse response of the concatenated weighting filter and the backward adapted synthesis filter used in vector $m$. Finally $x_m(n)$ is the codebook target for vector $m$, which with $(2p+1)$'th order LTP is given by

$$x_m(n) = \frac{s_{wm}(n) - \hat{s}_{om}(n) - \sum_{j=-p}^{j=p} b_{jm} u_m(n - L_m - j)}{\hat{\sigma}_m}. \tag{6.23}$$

Here $s_{wm}(n)$ is the weighted input speech in vector $m$, $\hat{s}_{om}(n)$ is the zero input response of the weighting and synthesis filters, $u_m(n)$ is the previous excitation and $L_m$ and $b_{jm}$ are the backward adapted LTP delay and coefficients in vector $m$.

Equation 6.22 giving $E_k$ can be expanded to give

$$\begin{aligned} E_k &= \sum_{m \in N_k} \left( \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} (x_m(n) - g_m[h_m(n) * c_k(n)])^2 \right) \qquad (6.24) \\ &= \sum_{m \in N_k} \left( \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} x_m^2(n) + \hat{\sigma}_m^2 g_m^2 \sum_{n=0}^{vs-1} [h_m(n) * c_k(n)]^2 \right. \\ &\qquad \left. -2\hat{\sigma}_m^2 g_m \sum_{n=0}^{vs-1} x_m(n)[h_m(n) * c_k(n)] \right) \\ &= \sum_{m \in N_k} \left( \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} x_m^2(n) + \hat{\sigma}_m^2 g_m^2 \sum_{n=0}^{vs-1} [h_m(n) * c_k(n)]^2 - 2\hat{\sigma}_m^2 g_m \sum_{n=0}^{vs-1} p_m(n) c_k(n) \right) \end{aligned}$$

where $p_m(j)$ is the reverse convolution between $h_m(n)$ and the target $x_m(n)$. This expression can be partially differentiated with respect to element $n = j$ of the codebook entry $c_k(n)$ to give

$$\frac{\partial E_k}{\partial c_k(j)} = \sum_{m \in N_k} \left( 2\hat{\sigma}_m^2 g_m^2 \sum_{n=0}^{vs-1} c_k(n) H(n,j) - 2\hat{\sigma}_m^2 g_m p_m(j) \right) \tag{6.25}$$

where $H_m(n,j)$ is the autocorrelation of the delayed impulse response $h_m(n)$ and is given by

$$H_m(i,j) = \sum_{n=0}^{vs-1} h_m(n-i) h_m(n-j). \tag{6.26}$$

Setting these partial derivatives to zero gives the optimum codebook entry $c_k^*(n)$ for the cluster of vectors $N_k$ as the solution of the set of simultaneous equations

$$\sum_{m \in N_k} \left( \hat{\sigma}_m^2 g_m^2 \sum_{n=0}^{vs-1} c_k^*(n) H_m(n,j) \right) = \sum_{m \in N_k} \left( \hat{\sigma}_m^2 g_m p_m(j) \right) \quad \text{for } j = 0, 1, \cdots, vs-1. \tag{6.27}$$

A similar expression for the total weighted error $E_i$ from all the vectors that use the gain codebook entry $g_i$ is

$$\begin{aligned} E_i &= \sum_{m \in N_i} \left( \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} (x_m(n) - g_i[h_m(n) * c_m(n)])^2 \right) \\ &= \sum_{m \in N_i} \left( \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} x_m^2(n) + g_i^2 \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} [h_m(n) * c_m(n)]^2 - \right. \\ &\qquad \left. 2 g_i \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} x_m(n)[h_m(n) * c_m(n)] \right) \end{aligned} \tag{6.28}$$

where $N_i$ is the set of vectors that use the gain codebook entry $g_i$, and $c_m(n)$ is the shape codebook entry used by the $m$'th vector. Differentiating this expression with respect to $g_i$ gives

$$\frac{\partial E_i}{\partial g_i} = \sum_{m \in N_i} \left( 2 g_i \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} [h_m(n) * c_m(n)]^2 - 2 \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} x_m(n)[h_m(n) * c_m(n)] \right) \tag{6.29}$$

and setting this partial derivative to zero gives the optimum gain codebook entry $g_i^*$ for the cluster of vectors $N_i$ as

$$g_i^* = \frac{\sum_{m \in N_i} \left( \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} x_m(n)[h_m(n) * c_m(n)] \right)}{\sum_{m \in N_i} \left( \hat{\sigma}_m^2 \sum_{n=0}^{vs-1} [c_m(n) * h_m(n)]^2 \right)}, \tag{6.30}$$

The summations in Equations 6.27 and 6.30 over all the vectors that use $c_k(n)$ or $g_i$ are carried out for all 128 shape codebook entries and all 8 gain codebook entries as the coding of the training speech takes place. At the end of the coding the shape and gain codebooks are updated using Equations 6.27 and 6.30, and then the codec starts coding the training speech again with the new codebooks. This closed loop codebook training procedure is summarised below

1. Start with an initial gain and shape codebook.
2. Code the training sequence using the given codebooks. Accumulate the summations in Equations 6.27 and 6.30.
3. Calculate the total weighted error of the coded speech. If this distortion is less than the minimum distortion so far keep a record of the codebooks used as the best codebooks so far.
4. Calculate new shape and gain codebooks using Equations 6.27 and 6.30.

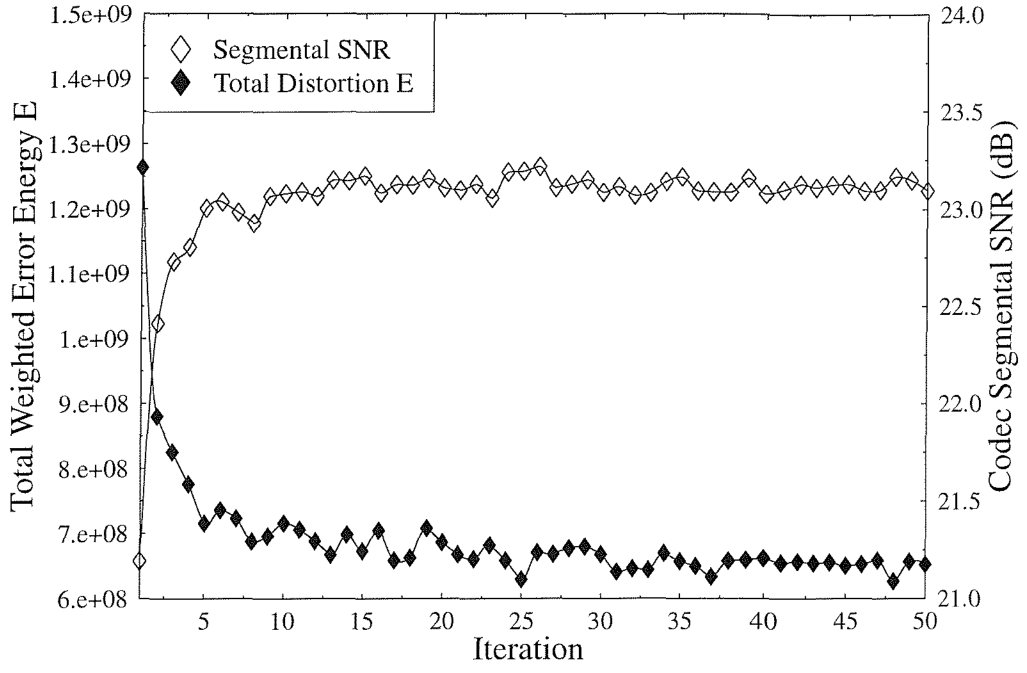

Figure 6.11: Codec's Performance as the Codebooks are Trained

5. Return to step 2.

Each entire coding of the training speech file counts as one iteration, and Figure 6.11 shows the variation in the total weighted error energy $E$, and the codec's segmental SNR, as the training progresses for the 16 kbits/s codebooks. From this figure it can be seen that this closed-loop training sequence does not give a monotonic decrease in the total weighted error from one iteration to the next. This is because of the changing of the codebook target $x_m(n)$, as well as the other backward adapted parameters, from one iteration to the next. However it is clear from Figure 6.11 that the training does give a significant improvement in the codec's performance. Due to the non-monotonic decrease in the total weighted error energy it is necessary during the codebook training to keep a record of the lowest error energy achieved so far, and the corresponding codebooks. If a certain number of iterations passes without this minimum energy being improved then the codebook training can be terminated. It can be seen from Figure 6.11 that we get close to the minimum within about 20 iterations.

An important aspect in vector quantizer training can be the initial codebook used. In Figure 6.11 we used the G728 gain codebook and the Gaussian shape codebook as the initial codebooks. We also tried using other codebooks such as the G728 fixed codebook, and Gaussian codebooks with different variances, as the initial codebooks. However, although these gave very different starting values of the total weighted error $E$, and took different numbers of iterations to give their optimum codebooks, they all resulted in codebooks which gave very similar performances. Therefore we concluded that the G728 gain codebook, and the Gaussian shape codebook, are suitable for use as the initial codebooks.

We trained different shape and gain codebooks for use by our codec at all of its bit rates between 8 and 16 kbits/s. The average segmental SNR given by the codec using these codebooks is shown in Figure 6.12 for the 4 speech sentences which were not part of the training sequence. Also shown in this figure for comparison is the curve from Figure 6.9 for the corresponding codec with the untrained codebooks. It can be seen that the codebook training gives an improvement of about 1.5 to 2 dB across the codec's range of bit rates.

It can be seen from Figure 6.11 that a decrease in the total weighted error energy $E$ does not necessarily correspond to an increase in the codec's segmental SNR. This is also true for the codec's segmental weighted SNR, and is because the distortion $D$ calculated takes no account of the different signal energies in different vectors. We tried altering the codebook training algorithm described above to take account of this, hoping that it would result in codebooks which gave lower segmental SNRs. However the codebooks trained with this modified algorithm gave very similar performances to those trained by minimising $E$.

We also attempted training different codebooks at each bit rate for voiced and unvoiced speech. The voicing decision can be made backward adaptive based on the correlations in the previous reconstructed speech. A voiced/unvoiced decision like this is made in the G728 post-filter to determine whether to apply pitch post-filtering. We found however that although an accurate determination of the voicing of the speech could be made in a backward adaptive manner, no significant improvement in the codec's performance could be achieved by using separately trained voiced and unvoiced codebooks. This agrees with the results in [68] when fully backward adaptive LTP is used.

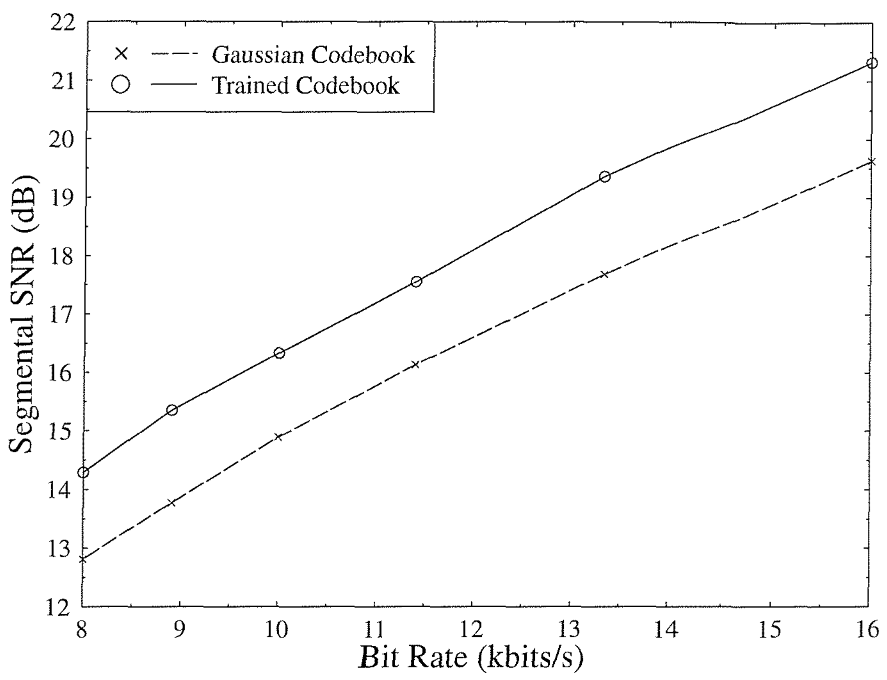

Figure 6.12: Performance of the 8-16 kbits/s Codec with Trained Codebooks

## 6.6 A Variable Rate Codec with a Constant Vector Size

In the previous sections we discussed a variable rate codec based on G728 which varied its bit rate by changing the number of samples in each vector. The excitation for each vector was coded with 10 bits. In this section we describe the alternative approach of keeping the vector size constant and varying the number of bits used to code the excitation. The bit rate of the codec is varied between 8 and 16 kbits/s with a constant vector size of 5 samples by using between 5 and 10 bits to code the excitation signal for each vector. We used a structure for the codec identical to that described earlier, with backward gain adaption for the excitation and backward adapted short and long term synthesis filters. With 10,9 or 8 bits to code the excitation we used a split vector quantizer, similar to that used in G728, with a 7 bit shape codebook and a 3,2 or 1 bit gain codebook. For the lower bit rates we used a single 7,6 or 5 bit

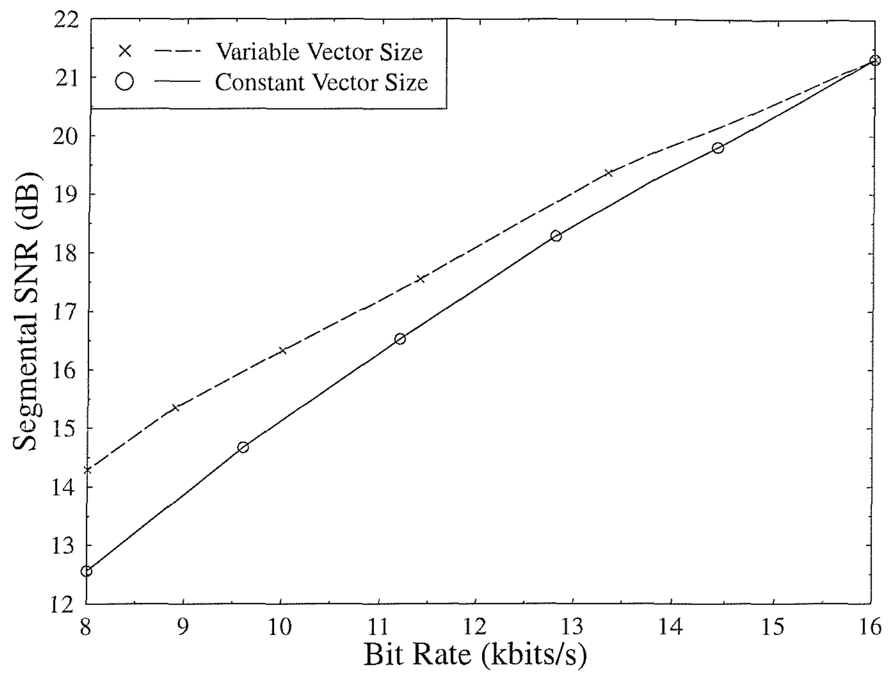

Figure 6.13: Performance of Variable Rate Codec with Constant Vector Size

vector quantizer to code the excitation. Codebooks were trained for the various bit rates using the closed loop codebook training technique described in Section 6.5.

The segmental SNR of this variable rate codec is shown in Figure 6.13. Also shown in this graph is the segmental SNR of the codec with a variable vector size, copied here from Figure 6.12 for comparison. At 16 kbits/s the two codecs are of course identical, but at lower rates the constant vector size codec performs worse than the variable vector size codec. The difference between the two approaches increases as the bit rate decreases, and at 8 kbits/s the segmental SNR of the constant vector size codec is about 1.75 dB lower than that of the variable vector size codec.

However, although the constant vector size codec gives lower reconstructed speech quality, it does have certain advantages. The most obvious is that it has a constant delay equal to that of G728, ie less than 2ms. Also the complexity of its encoder, especially at low bit rates, is lower than that of the variable vector size codec. This is because of the smaller codebooks used - at 8 kbits/s the codebook search procedure

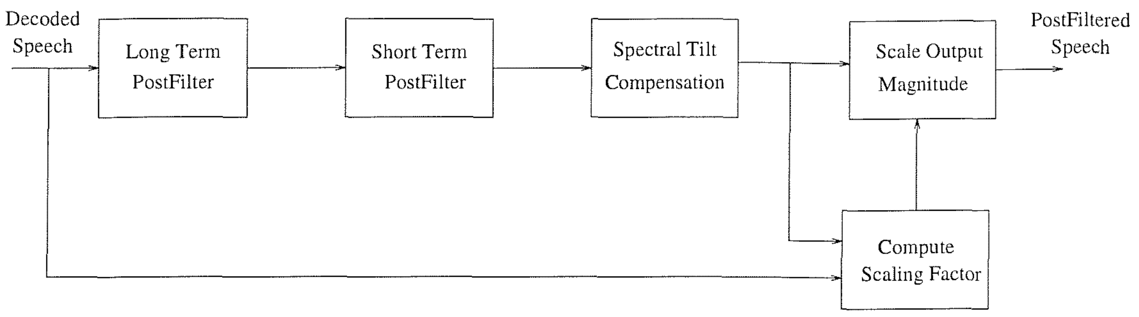

Figure 6.14: The G728 Adaptive PostFilter

has only to examine 32 codebook entries. Therefore for some applications this codec may be more suitable than the higher speech quality variable vector size codec.

## 6.7 PostFiltering

An adaptive postfilter is used with the G728 codec to improve the subjective quality of its reconstructed speech. Such postfilters have been used for several years with various speech codecs, and are described in detail in [75]. They work by emphasizing the formant and pitch peaks in the speech, and attenuating the valleys between these peaks. This reduces the audible noise in the reconstructed speech because, even with the noise shaping of the error weighting filter, it is in the valleys between the formant and pitch peaks that the noise energy is most likely to cross the masking threshold and become audible. Therefore attenuating the speech in these regions reduces the audible noise, and because our ears are not very sensitive to the speech intensity in these valleys only minimal distortion is introduced to the speech signal.

A block digram of the post filter used in the G728 codec, and our variable rate codecs, is shown in Figure 6.14. The long term postfilter has a transfer function

$$H_l(z) = \frac{1}{1+b}(1 + bz^{-p}) \tag{6.31}$$

where $p$ is the backward adapted estimate of the pitch period, calculated as explained in section 6.4, and the coefficient $b$ is given by

$$b = \begin{cases} 0 & \text{if } \beta < 0.6 \\ \lambda\beta & \text{if } 0.6 \leq \beta \leq 1 \\ \lambda & \text{if } \beta > 1 \end{cases} \tag{6.32}$$

where $\lambda$ is a parameter which controls the amount of long term postfiltering and $\beta$ is the tap weight for a single tap predictor with delay $p$, given by

$$\beta = \frac{\sum_{n=-100}^{-1} \hat{s}(n)\hat{s}(n-p)}{\sum_{n=-100}^{-1} \hat{s}^2(n-p)}. \tag{6.33}$$

If $\beta$ is less than 0.6 then the speech is assumed to be unvoiced and $b$ is set to zero, effectively tuning off the long term postfilter.

The short term postfilter is given by

$$H_s(z) = \frac{1 - \sum_{i=1}^{10} \tilde{a}_i \gamma_1^i z^{-i}}{1 - \sum_{i=1}^{10} \tilde{a}_i \gamma_2^i z^{-i}} \tag{6.34}$$

where $\gamma_1$ and $\gamma_2$ are tunable parameters which control the short term postfiltering and $\tilde{a}_i$ ,$i = 1, 2 \cdots 10$, are backward adapted short term synthesis filter parameters for a filter of order 10, which are derived as a by product of calculating the coefficients for the higher order synthesis filter. The all pole section of $H_s(z)$ emphasizes the formants in the reconstructed speech, and attenuates the valleys between these formants. However this filtering operation introduces an undesirable spectral tilt in the postfiltered speech, which leads to it sounding muffled. This spectral tilt is partially offset by the all zero section of $H_s(z)$.

The all zero section of $H_s(z)$ significantly reduces the muffling effect of the postfilter. However the postfiltered speech is still slightly muffled, and so a spectral tilt compensation block is used to further reduce this effect. This is a first order filter with a transfer function $1 - \mu k_1 z^{-1}$, where $\mu$ is a tunable parameter between 0 and 1, and $k_1$ is the first reflection coefficient calculated from the LPC analysis of the reconstructed speech. During voiced speech the postfilter introduces a low pass spectral tilt to the speech, but $k_1$ is close to -1 and so the spectral tilt compensation block introduces high pass filtering to offset this spectral tilt. During unvoiced speech the postfilter tends to introduce a high pass spectral tilt to the speech, but $k_1$ becomes positive and so the spectral tilt compensation block automatically changes to a low pass filter and again offsets the spectral tilt.

The final section of the postfilter scales the output so it has approximately the same power as the original decoded speech. The long term postfilter has its own gain control because of the factor $1/(1 + b)$ in $H_l(z)$. However the short term postfilter tends to amplify the postfiltered speech when the prediction gain of the short term filter is high, and this leads to the output speech sounding unnatural. The output scaling blocks remove this effect by estimating the average magnitudes of the decoded speech and the output from the spectral tilt compensation block, and determining a scaling factor based on the ratio of these average magnitudes.

The tunable parameters $\lambda$, $\gamma_1$, $\gamma_2$ and $\mu$ must be chosen to control the amount of postfiltering used. We want to introduce enough postfiltering to attenuate the audible coding noise as much as possible without introducing too much distortion to the postfiltered speech. In the G728 codec the parameters were chosen to minimise the coding noise after three tandems of the codec [3] because of the CCITT requirements regarding tandeming of the codec. The parameters were set to $\lambda = 0.15$, $\gamma_1 = 0.65$, $\gamma_2 = 0.75$ and $\mu = 0.15$. This postfilter does give some improvement to the speech quality of our variable rate codec after just one stage of coding of the speech, ie with no tandeming. However a greater improvement can be achieved by tuning the parameters for the best performance after one stage of coding. We used the parameters $\lambda = 0.5$, $\gamma_1 = 0.5$, $\gamma_2 = 0.8$, and $\mu = 0.5$, and found that with these parameters the postfilter produced a significant improvement in the subjective quality of the reconstructed speech.

## 6.8 Conclusion

In this chapter we have described a variable rate low delay codec which is compatible with the 16 kbits/s G728 codec at its highest bit rate, and exhibits a graceful degradation in speech quality down to 8 kbits/s. The bit rate can be reduced while the buffering delay is kept constant at 5 samples (0.625 ms), or alternatively better speech quality is achieved if the buffering delay is increased to 10 samples as the bit rate is reduced down to 8 kbits/s.

# Chapter 7

# Low Delay CELP Codecs at 8 to 4 kbits/s

## 7.1 Introduction

In the previous chapter we described the 16 kbits/s G728 low delay codec, and discussed modifying this codec to produce an 8-16 kbits/s variable rate codec. In this chapter we consider methods of improving the performance of the 8 kbits/s codec, while maintaining as low a delay and complexity as possible. The 8 kbits/s codec developed in Chapter 6 uses a 3 bit gain codebook and a 7 bit shape codebook with backward adaption of both the long and the short term synthesis filters, and gives an average segmental SNR of 14.29 dB. In Section 7.2 we describe the effect of increasing the size of the gain and shape codebooks in this codec while keeping a vector length of 10 samples. This is followed by Sections 7.3 and 7.4 where we consider the improvements that can be achieved, again while maintaining a vector length of 10 samples, by using forward adaption of the short and long term synthesis filters. Then in Section 7.5 we show the performance of three codecs, based on those developed in the earlier sections, operating at bit rates between 8 and 4 kbits/s. Finally in Section 7.6 we describe a codec, with a vector size of 40 samples, based on the algebraic codebook structure we described in Chapter 3. The performance of this codec is compared to the previously introduced low delay codecs from Section 7.5 and the higher delay ACELP codec described in Chapter 3.

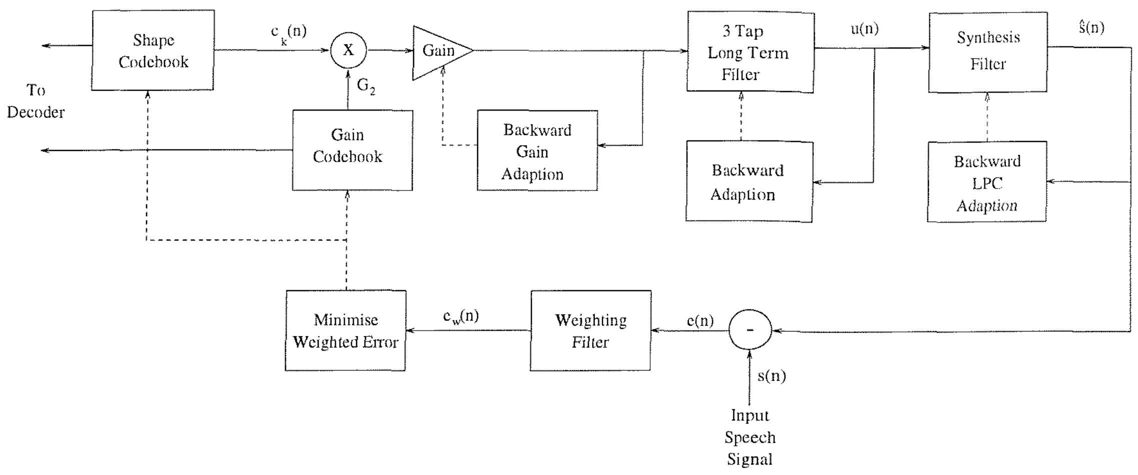

Figure 7.1: Scheme One Low Delay CELP Codec

| Gain Codebook Bits | Shape Codebook Bits | Segmental SNR (dB) |
|---|---|---|
| 3 | 7 | 14.29 |
| 4 | 7 | 15.24 |
| 5 | 7 | 15.62 |
| 3 | 8 | 15.33 |
| 3 | 9 | 16.12 |
| 4 | 8 | 16.01 |

Table 7.1: Performance of the Scheme One Codec with Various Size Gain and Shape Codebooks

## 7.2 Improvements Due to Increasing Codebook Sizes

In this section we use the same structure for the codec as was developed in Chapter 6 but increase the size of the shape and the gain codebooks. This codec structure is shown in Figure 7.1, and we refer to it as "Scheme One". We used 3 tap backward adapted LTP and a vector length of 10 samples with a 7 bit shape codebook, and varied the size of the gain codebook from 3 to 4 and 5 bits. Then in our next experiments we used a 3 bit gain codebook and trained 8 and 9 bit shape codebooks. Finally we attempted increasing the size of both the shape and the gain codebooks by one bit. In each case the new codebooks were closed-loop trained using the technique described in Section 6.5.

The segmental SNRs of this Scheme One codec with various size shape and gain codebooks is shown in Table 7.1. It can be seen that adding one bit to either the gain or the shape codebook increases the segmental SNR of the codec by about 1 dB.

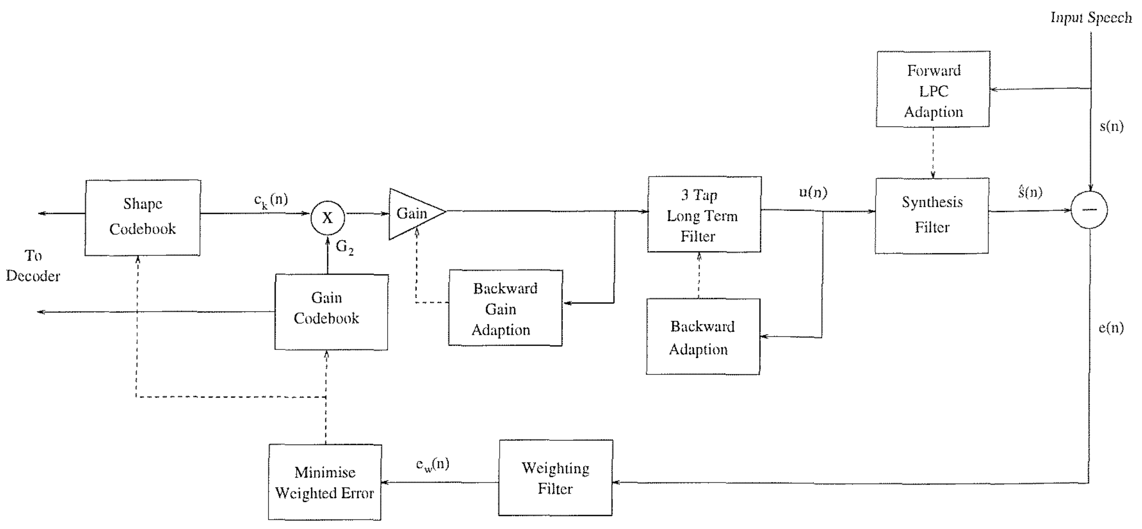

Figure 7.2: Scheme Two Low Delay CELP Codec

Adding two extra bits to the shape codebook, or one bit each to both codebooks, increases the segmental SNR by almost 2 dB.

## 7.3 Forward Adaption of the Short Term Synthesis Filter

In this section we consider the improvements that can be achieved in the vector size 10 codec by using forward adaption of the short term synthesis filter. In Table 6.5 we examined the effects of backward adaption of the synthesis filter at 8 kbits/s. However these figures gave the improvements that can be achieved by eliminating the noise feedback and time mismatch that are inherent in backward adaption when using the same recursive windowing function and update rate as the G728 codec. In this section we consider the improvements that could be achieved by significantly altering the structure used for the determination of the synthesis filter parameters.

The codec structure used is shown in Figure 7.2, and we refer to it as "Scheme Two". Its only difference from the 8 kbits/s codec developed in Chapter 6 is that we replaced the recursive windowing function shown in Figure 6.3 with an asymmetric analysis window which was used in a candidate codec for the CCITT 8 kbits/s standard [66, 76]. This window, which is shown in Figure 7.3, is made up of half a Hamming window and a quarter of a cosine function cycle. The windowing scheme uses a frame

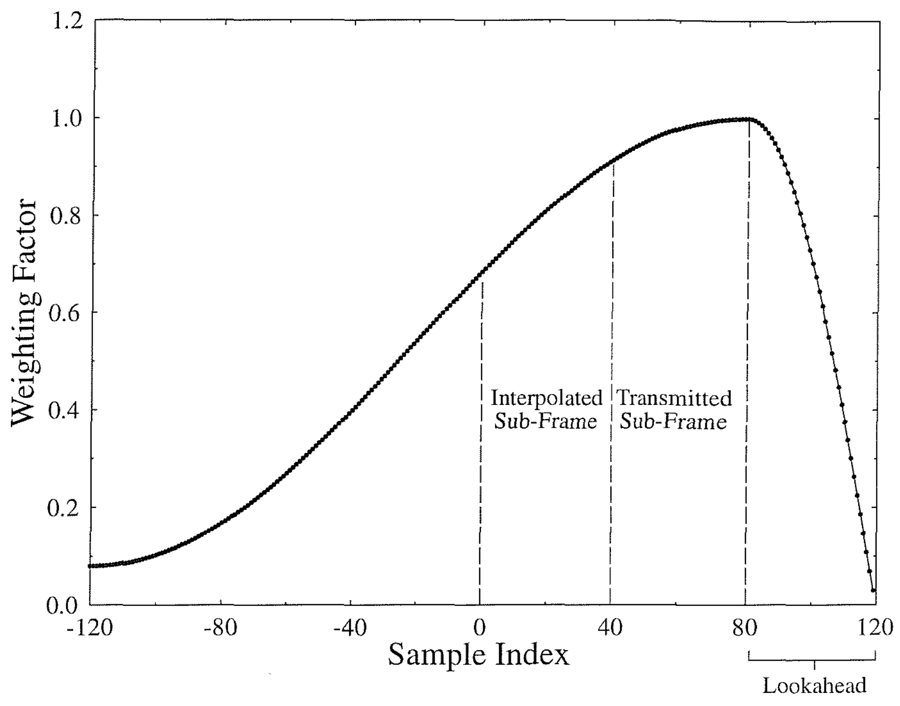

Figure 7.3: LPC Windowing Function Used in Candidate CCITT 8 kbits/s Codec

length of 10 ms (or 80 samples), with a 5 ms look-ahead. The 10 ms frame consists of two sub-frames, and a Line Spectral Frequency (LSF) interpolation scheme similar to that described in Chapter 3 is used.

We implemented this method of deriving the LPC coefficients in our codec. The vector length was kept constant at 10 samples, but instead of the synthesis filter parameters being updated every 20 samples, as in the Scheme One codec, they were updated every 40 samples using either the interpolated or transmitted LSFs. In the candidate 8 kbits/s CCITT codec [66] a filter order of ten is used and the ten LSFs are quantized with 19 bits using differential split vector quantization. However for simplicity, and in order to see the best performance gain possible for our codec by using forward adaption of the short term synthesis filter, we used the ten unquantized LSFs to derive the filter coefficients. A new 3 bit gain codebook and 7 bit shape codebook were derived for this codec using the codebook training technique described in Section 6.5. We found that this forward adaption increased the segmental SNR of the codec by only 0.8 dB, and even this rather small improvement would of course be reduced by the quantization of the LSFs. Using a 19 bit quantization scheme to transmit a new set of LSFs every 80 sample frame would mean using on average about 2.4 bits per 10 sample vector.

Traditionally codecs employing forward adaptive LPC are more resilient to channel errors than those using backward adaptive LPC. However a big disadvantage of using such a forward adaptive LPC scheme is that it would increase the delay of the codec by almost an order of magnitude. Instead of a vector length of 10 samples we would need to buffer a frame of 80 speech samples, plus a 40 sample look-ahead, to calculate the LPC information. This would increase the overall delay of the codec from under 4 ms to about 35 ms.

## 7.4 Forward Adaption of the Long Term Predictor

### 7.4.1 Initial Experiments

In this section we consider the gains in our codec performance which can be achieved using forward adaption of the Long Term Predictor (LTP) gain. Although forward adaption of the LTP parameters would improve the codec's robustness to channel errors, we did not consider forward adaption of the LTP delay because to transmit this delay from the encoder to the decoder would require around 7 extra bits per vector. However we expected to be able to improve the performance of the codec, at the cost of significantly fewer extra bits, by using forward adaption of the LTP gain.

The codec developed in Chapter 6 uses a three tap LTP with backward adapted values for the delay and filter coefficients. Initially we replaced this LTP scheme with an adaptive codebook arrangement, where the delay was still backward adapted but the gain was calculated as described in Section 3.5. This calculation assumes that the fixed codebook signal, which is not known until after the LTP parameters are calculated, is zero. The "optimum" adaptive codebook gain $G_1$, which minimises the weighted error between the original and reconstructed speech, was then given by Equation 3.16, which is repeated here for convenience:

$$G_1 = \frac{\sum_{n=0}^{vs-1} x(n) y_\alpha(n)}{\sum_{n=0}^{vs-1} y_\alpha^2(n)}. \tag{7.1}$$

Here $x(n) = s_w(n) - \hat{s}_o(n)$ is the target for the adaptive codebook search, $s_w(n)$ is the weighted speech signal, $\hat{s}_o(n)$ is the zero input response of the weighted synthesis filter, and

$$y_\alpha(n) = \sum_{i=0}^{n} u(i-\alpha)h(n-i) \tag{7.2}$$

is the convolution of the adaptive codebook signal $u(n-\alpha)$ with the impulse response $h(n)$ of the weighted synthesis filter, where $\alpha$ is the backward adapted LTP delay.

Again we trained new 7/3 bit shape/gain fixed codebooks, and used the unquantized LTP gain $G_1$ as given by Equation 7.1. However we found that this arrangement improved the segmental SNR of our codec by only 0.1 dB over the codec with 3 tap backward adapted LTP. Therefore we decided to try some of the joint adaptive and fixed codebook optimization schemes described in Section 4.2.4. These joint optimization schemes are described below.

The simplest optimization scheme, Method A from Section 4.2.4, involves calculating the adaptive and fixed codebook gains and indices as usual, and then updating the two gains for the given codebook indices $k$ and $\alpha$ using Equations 4.10 and 4.11, which are repeated here for convenience:

$$G_1 = \frac{C_\alpha \xi_k - C_k Y_{\alpha k}}{\xi_\alpha \xi_k - Y_{\alpha k}^2} \tag{7.3}$$

$$G_2 = \frac{C_k \xi_\alpha - C_\alpha Y_{\alpha k}}{\xi_\alpha \xi_k - Y_{\alpha k}^2}. \tag{7.4}$$

Here $G_1$ is the LTP gain, $G_2$ is the fixed codebook gain,

$$\xi_\alpha = \sum_{n=0}^{vs-1} y_\alpha^2(n) \tag{7.5}$$

is the energy of the filtered adaptive codebook signal and

$$C_\alpha = \sum_{n=0}^{vs-1} x(n) y_\alpha(n) \tag{7.6}$$

is the correlation between the filtered adaptive codebook signal and the codebook target $x(n)$. Similarly $\xi_k$ is the energy of the filtered fixed codebook signal $[c_k(n)*h(n)]$, and $C_k$ is the correlation between this and the target signal. Finally

$$Y_{\alpha k} = \sum_{n=0}^{vs-1} y_\alpha(n)[c_k(n) * h(n)] \tag{7.7}$$

is the correlation between the filtered signals from the two codebooks.

We studied the performance of this gain update scheme in our vector length 10 codec. A 7 bit fixed shape codebook was trained, but the LTP and fixed codebook gains were not quantized. We found that the gain update improved the segmental SNR of our codec by 1.2 dB over the codec with backward adapted 3 tap LTP and no fixed codebook gain quantization. This is a much more significant improvement than that reported in Section 4.2.4 for our 4.7 kbits/s ACELP codec, because of the much higher update rate for the gains used in our present codec. In our low delay codec the two gains are calculated for every 10 sample vector, whereas in the 4.7 kbits/s ACELP codec used in Chapter 4 the two gains are updated only every 60 sample sub-frame.

Encouraged by these results we also invoked the second sub-optimal joint codebook search procedure described in Section 4.2.4. In this search procedure the adaptive codebook delay $\alpha$ is determined first, by backward adaption in our present codec, and then for each fixed codebook index $k$ the optimum LTP and fixed codebook gains $G_1$ and $G_2$ are determined using Equations 7.3 and 7.4 above. The index $k$ which maximises $T_{\alpha k}$, given below in Equation 7.8, will minimise the weighted error between the reconstructed and the original speech for the present vector, and is transmitted to the decoder. This codebook search procedure was referred to as Method B in Section 4.2.4.

$$T_{\alpha k} = 2\left(G_1 C_\alpha + \hat{\sigma} G_2 C_k - \hat{\sigma} G_1 G_2 Y_{\alpha k}\right) - G_1^2 \xi_\alpha - \hat{\sigma}^2 G_2^2 \xi_k \tag{7.8}$$

We trained a new 7 bit fixed shape codebook for this joint codebook search algorithm, and the two gains $G_1$ and $G_2$ were left unquantized. We found that this scheme gave an additional improvement in the performance of the codec so that its segmental SNR was now 2.7 dB higher than the codec with backward adapted 3 tap LTP and no fixed gain quantization. Again this is a much more significant improvement than that which we found for our 4.7 kbits/s ACELP codec.

### 7.4.2 Quantization of Jointly Optimized Gains

The improvements quoted above for our vector size 10 codec when we use an adaptive codebook arrangement with joint calculation of the LTP and fixed codebook gains, and no quantization of either gain, are quite promising. Next we considered the quantization of the two gains $G_1$ and $G_2$. In order to minimise the number of bits used we decided to use a vector quantizer for the two gains. A block diagram of the coding scheme used is shown in Figure 7.4. We refer to this arrangement as "Scheme

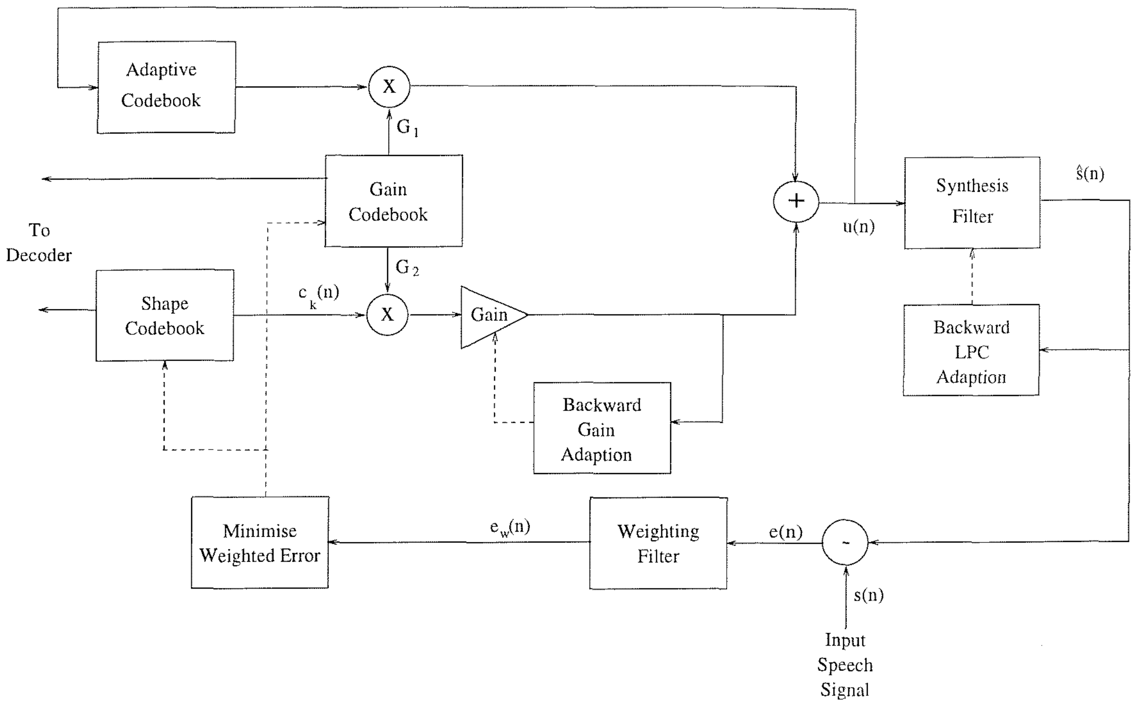

Figure 7.4: Scheme Three Low Delay CELP Codec

Three".

This Scheme Three codec with forward adaptive LTP was tested with 4,5,6 and 7 bit vector quantizers for the fixed and adaptive codebook gains and a 7 bit shape codebook. The vector quantizers were trained as follows. For a given vector quantizer level $i$ the total weighted energy $E_i$ for speech vectors using this level will be

$$E_i = \sum_{m\in N_i} \left( \sum_{n=0}^{vs-1} (x_m(n) - G_{1i} y_{\alpha m}(n) - G_{2i}\hat{\sigma}_m[h_m(n) * c_m(n)])^2 \right). \qquad (7.9)$$

Here $x_m(n)$, $y_{\alpha m}(n)$, and $h_m(n)$ are the signals $x(n)$, $y_\alpha(n)$, and $h(n)$ in the $m$'th vector, $\hat{\sigma}_m$ is the value of the backward adapted gain $\hat{\sigma}$ in the $m$'th vector, $c_m(n)$ is the fixed codebook entry $c_k(n)$ used in the $m$'th vector, $G_{1i}$ and $G_{2i}$ are the values of the two gains in the $i$'th entry of the joint vector quantizer, and $N_i$ is the set of speech vectors that use the $i$'th entry of the vector quantizer. As before $vs$ is the vector size used in the codec, which in our present experiments is ten.

Expanding Equation 7.9 gives

$$E_i = \sum_{m\in N_i} \left( X_m + G_{1i}^2 \xi_{\alpha m} + G_{2i}^2 \hat{\sigma}_m^2 \xi_{km} - 2G_{1i}C_{\alpha m} - 2\hat{\sigma}_m G_{2i} C_{km} + 2\hat{\sigma}_m G_{1i} G_{2i} Y_{\alpha km} \right) \qquad (7.10)$$

where $X_m = \sum_{n=0}^{vs-1} x_m^2(n)$ is the energy of the target signal $x_m(n)$, and $\xi_{\alpha m}$, $\xi_{km}$, $C_{\alpha m}$, $C_{km}$ and $Y_{\alpha km}$ are the values in the $m$'th vector of $\xi_\alpha$, $\xi_k$, $C_\alpha$, $C_k$ and $Y_{\alpha k}$ defined earlier.

Differentiating Equation 7.10 with respect to $G_{1i}$ and setting the result to zero gives

$$\frac{\partial E_i}{\partial G_{1i}} = \sum_{m \in N_i} (2G_{1i}\xi_{\alpha m} - 2C_{\alpha m} + 2\hat{\sigma}_m G_{2i} Y_{\alpha km}) = 0 \tag{7.11}$$

or

$$G_{1i} \sum_{m \in N_i} \xi_{\alpha m} + G_{2i} \sum_{m \in N_i} \hat{\sigma}_m Y_{\alpha m} = \sum_{m \in N_i} C_{\alpha m}. \tag{7.12}$$

Similarly differentiating with respect to $G_{2i}$ and setting the result to zero gives

$$G_{1i} \sum_{m \in N_i} \hat{\sigma}_m Y_{\alpha km} + G_{2i} \sum_{m \in N_i} \hat{\sigma}_m^2 \xi_{km} = \sum_{m \in N_i} \hat{\sigma}_m C_{km}. \tag{7.13}$$

Solving these two simultaneous equations gives the optimum values of $G_{1i}$ and $G_{2i}$ for the cluster of vectors $N_i$ as

$$G_{1i} = \frac{\left(\sum_{m \in N_i} C_{\alpha m}\right)\left(\sum_{m \in N_i} \hat{\sigma}_m^2 \xi_{km}\right) - \left(\sum_{m \in N_i} \hat{\sigma}_m C_{km}\right)\left(\sum_{m \in N_i} \hat{\sigma}_m Y_{\alpha km}\right)}{\left(\sum_{m \in N_i} \xi_{\alpha m}\right)\left(\sum_{m \in N_i} \hat{\sigma}_m^2 \xi_{km}\right) - \left(\sum_{m \in N_i} \hat{\sigma}_m Y_{\alpha km}\right)^2} \tag{7.14}$$

and

$$G_{2i} = \frac{\left(\sum_{m \in N_i} \hat{\sigma}_m C_{km}\right)\left(\sum_{m \in N_i} \xi_{\alpha m}\right) - \left(\sum_{m \in N_i} C_{\alpha m}\right)\left(\sum_{m \in N_i} \hat{\sigma}_m Y_{\alpha km}\right)}{\left(\sum_{m \in N_i} \xi_{\alpha m}\right)\left(\sum_{m \in N_i} \hat{\sigma}_m^2 \xi_{km}\right) - \left(\sum_{m \in N_i} \hat{\sigma}_m Y_{\alpha km}\right)^2}. \tag{7.15}$$

Using Equations 7.14 and 7.15 we performed a closed-loop training of the vector quantizer gain codebook along with the fixed shape codebook similarly to the training of the shape and single gain codebooks described in Section 6.5. However we found a similar problem to that which we encountered when training scalar codebooks for $G_1$ and $G_2$ in Section 4.2.5. Specifically although almost all values of $G_1$ have magnitudes less than 2, a few values have very high magnitudes. This leads to a few levels in the trained vector quantizers having very high values, and being very rarely used. Following an in-depth investigation into this phenomenon we solved the problem by excluding all vectors for which the magnitude of $G_1$ was greater than 2, or the magnitude of $G_2$ was greater than 5, from the training sequence. This approach solved the problems of the trained gain codebooks having some very high and very rarely used levels.

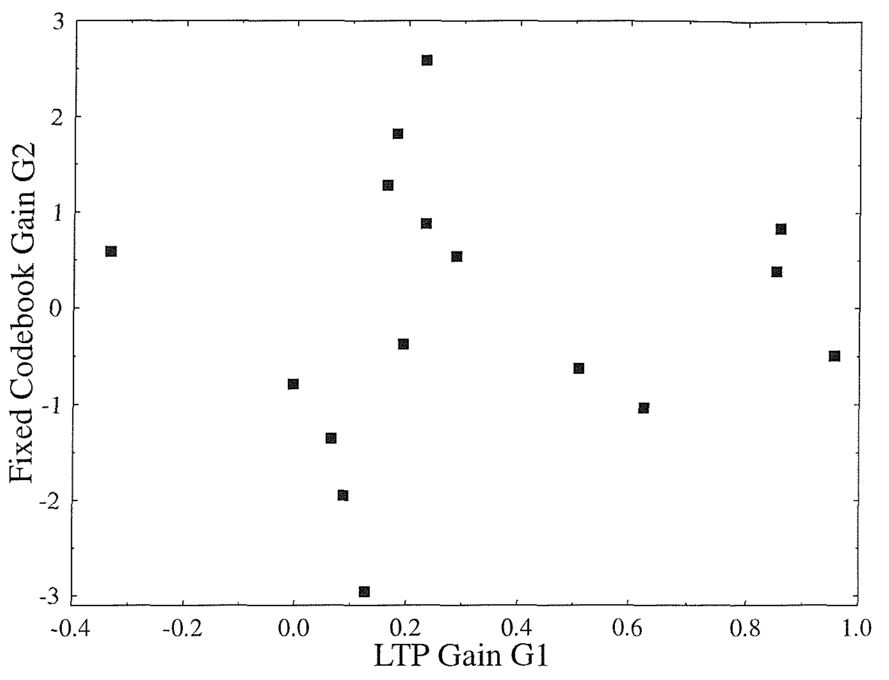

Figure 7.5: Values of $G_1$ and $G_2$ in the 4 Bit Gain Quantizer

We trained vector quantizers for the two gains using 4, 5, 6 and 7 bits. The values of the 4 bit trained vector quantizer for $G_1$ and $G_2$ are shown in Figure 7.5. It can be seen that when $G_1$ is close to zero the values of $G_2$ have a wide range of values between -3 and +3, but when the speech is voiced and $G_1$ is high the fixed codebook contribution to the excitation is less significant, and the quantized values of $G_2$ are closer to zero.

Our trained joint gain codebooks are searched as follows. For each fixed codebook entry $k$ the optimum gain codebook entry is found by trying each pair of gain values in Equation 7.8 to test which level maximises $T_{\alpha k}$ and hence minimises the weighted error energy. The segmental SNR of our Scheme Three codec with a trained 7 bit shape codebook and trained 4,5,6 and 7 bit joint $G1/G_2$ vector quantizers is shown in Table 7.2. The segmental SNRs in this table should be compared with the value of 14.29 dB obtained for the Scheme One codec with a 3 bit scalar quantizer for $G_2$ and 3 tap backward adapted LTP.

| Gain Codebook Bits | Segmental SNR (dB) |
|---|---|
| 4 Bits | 14.81 |
| 5 Bits | 15.71 |
| 6 Bits | 16.54 |
| 7 Bits | 17.08 |

Table 7.2: Performance of the Scheme Three Codecs

It can be seen from Table 7.2 that the joint $G_1/G_2$ gain codebooks give a steady increase in the performance of the codec as the size of the gain codebook is increased. In the next section we describe the use of backward adaptive voiced/unvoiced switched codebooks to further improve the performance of our codec.

### 7.4.3 Voiced/Unvoiced Switched Codebooks

In Section 6.5 we discussed using different codebooks for voiced and unvoiced segments of speech, and using a backward adaptive voicing decision to select which codebooks to use. However we found that in the case of a codec with fully backward adaptive LTP no significant improvement in the codec's performance was achieved by using switched codebook excitation. In this section we discuss using a similar switching arrangement in conjunction with our Scheme Three codec described above.

The backward adaptive voiced/unvoiced switching is based on the voiced/unvoiced switching used in the postfilter employed in the G728 codec [65]. In our codec the switch uses the normalised autocorrelation value of the past reconstructed speech signal $\hat{s}(n)$ at the delay $\alpha$ which is used by the adaptive codebook. This normalised autocorrelation value $\beta_\alpha$ is given by

$$\beta_\alpha = \frac{\sum_{n=-100}^{-1} \hat{s}(n)\hat{s}(n-\alpha)}{\sum_{n=-100}^{-1} \hat{s}^2(n-\alpha)}, \tag{7.16}$$

and when it is greater than a set threshold the speech is classified as voiced; otherwise the speech is classified as unvoiced. In our codec, as in the G728 postfilter, the threshold is set to 0.6.

Figure 7.6 shows a segment of the original speech and the normalised autocorrelation value $\beta_\alpha$ calculated from the reconstructed speech of our 8 kbits/s codec. To aid the clarity of this graph the values of $\beta_\alpha$ have been limited to lie between 0.05 and 0.95. It can be seen that the condition $\beta_\alpha > 0.6$ gives a good indication of whether the speech is voiced or unvoiced.

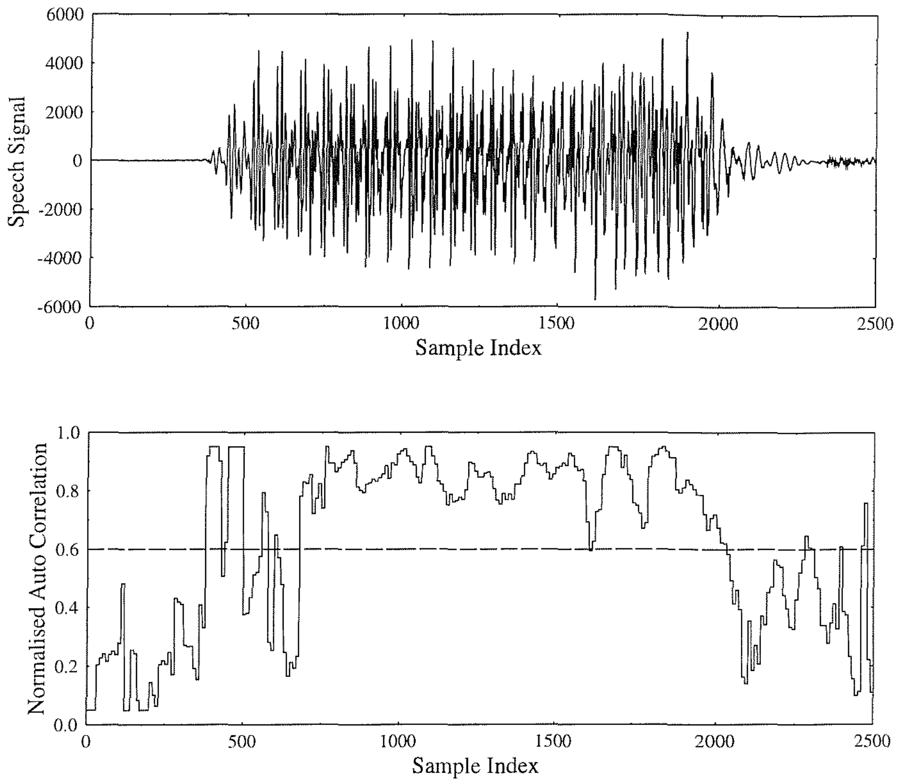

Figure 7.6: Normalised Autocorrelation Value $\beta_\alpha$ During Voiced and Unvoiced Speech

The backward adaptive voicing decision described above was incorporated into our Scheme Three codec shown in Figure 7.4 to produce a new coding arrangement which we referred to as "Scheme Four". Shape and joint gain codebooks were trained as described earlier for both the voiced and unvoiced modes of operation in a vector length 10 codec. The quantized values of $G_1$ and $G_2$ in both the 4 bit voiced and unvoiced codebooks are shown in Figure 7.7. It can be seen that similarly to Figure 7.5 when $G_1$ is high the range of values of $G_2$ is more limited than when $G_1$ is close to zero. Furthermore, as expected, the voiced codebook has a group of quantizer levels with $G_1$ close to one, whereas the values of the LTP gain in the unvoiced codebook are closer to zero.

The results we achieved with seven bit shape codebooks and joint gain codebooks of various sizes are shown in Table 7.3. It can be seen by comparing this to Table 7.2 that the voiced/unvoiced switching gives an improvement in the codec's performance of about 0.25 dB for the 4 and the 5 bit gain quantizers, and a smaller improvement

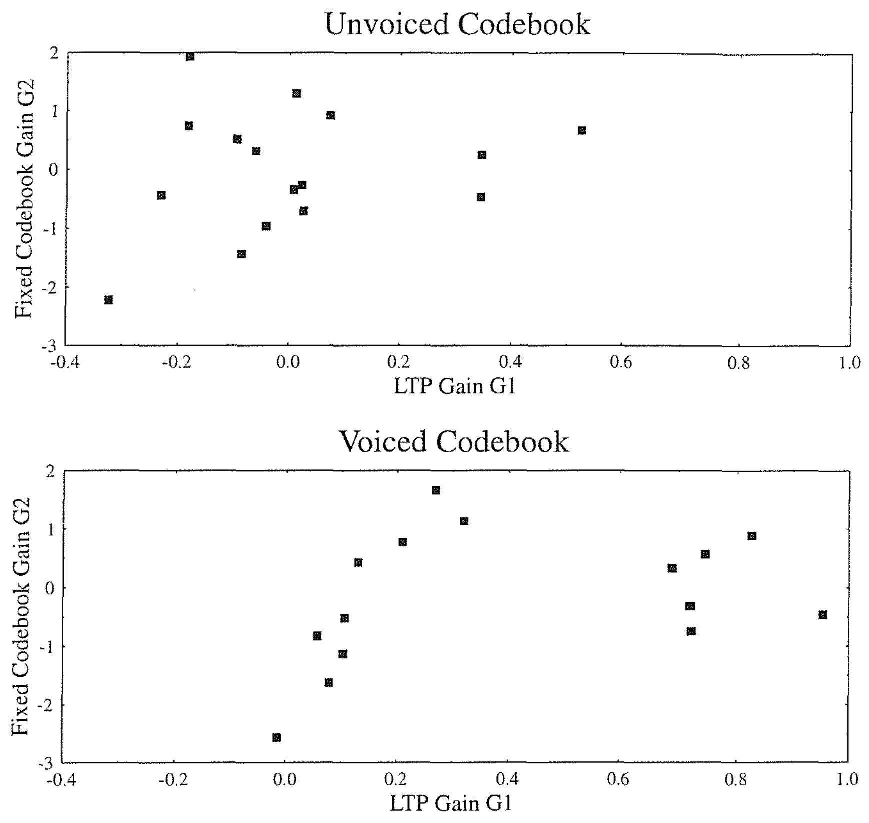

Figure 7.7: Values of $G_1$ and $G_2$ in the 4 Bit Voiced and Unvoiced Gain Quantizers

for the 6 and 7 bit gain quantizers.

## 7.5 Low Delay Codecs at 4-8 kbits/s

In the previous three sections we have considered the improvements that can be achieved in our vector size 10 codec by increasing the size of the shape and gain codebooks, and by using forward adaption of the short term predictor coefficients and

| Gain Codebook Bits | Segmental SNR (dB) |
|---|---|
| 4 Bits | 15.03 |
| 5 Bits | 15.92 |
| 6 Bits | 16.56 |
| 7 Bits | 17.12 |

Table 7.3: Performance of the Scheme Four Codecs

| | Synthesis Filter | Long Term Predictor | Shape C.B. | Gain C.B. | Extra Bits | Δ Segmental SNR |
|---|---|---|---|---|---|---|
| Scheme One | Backward | 3 Tap | 7 Bits | 3 Bits | 0 | 0 dB |
| | Adapted | Backward | 7 Bits | 4 Bits | 1 | +0.95 dB |
| | $p$=20 | Adapted | 8 Bits | 3 Bits | 1 | +1.04 dB |
| | | | 7 Bits | 5 Bits | 2 | +1.33 dB |
| | | | 8 Bits | 4 Bits | 2 | +1.72 dB |
| | | | 9 Bits | 3 Bits | 2 | +1.83 dB |
| Scheme Two | Forward Adapted $p$=10 | 3 Tap Backward Adapted | 7 Bits | 3 Bits | $\approx 2.4$ | $\leq +0.82$dB |
| Scheme Three | Backward | Forward | 7 Bits | 4 Bits | 1 | +0.52 dB |
| | Adapted | Adapted | 7 Bits | 5 Bits | 2 | +1.42 dB |
| | $p$=20 | | 7 Bits | 6 Bits | 3 | +2.25 dB |
| | | | 7 Bits | 7 Bits | 4 | +2.79 dB |
| Scheme Four | Backward | Switched | 7 Bits | 4 Bits | 1 | +0.74 dB |
| | Adapted | Forward | 7 Bits | 5 Bits | 2 | +1.63 dB |
| | $p$=20 | Adapted | 7 Bits | 6 Bits | 3 | +2.27 dB |
| | | | 7 Bits | 7 Bits | 4 | +2.83 dB |

Table 7.4: Improvements Obtained Using Schemes One to Four

the long term predictor gain. The improvements obtained by these schemes are summarised in Table 7.4, which shows the various gains in the codec's segmental SNR against the number of extra bits used to represent each ten sample vector.

In this table the Scheme One codec (see Section 7.2) is the vector size 10 codec developed in Chapter 6, with three tap backward adapted LTP and a 20 tap backward adapted short term predictor. The table shows the gains in the segmental SNR of the codec that are achieved by adding one or two extra bits to the shape or the scalar gain codebooks.

The Scheme Two codec (see Section 7.3 ) also uses 3 tap backward adapted LTP, but uses forward adaption to determine the short term synthesis filter coefficients. Using these coefficients without quantization gives an improvement in the codecs segmental SNR of 0.82 dB, which would be reduced if quantization were applied. In reference [66], where forward adaption is used for the LPC parameters, 19 bits are used to quantize a set of LSFs for every 80 sample frame; this quantization scheme would require us to use about 2.4 extra bits per 10 sample vector.

The Scheme Three codec (see Section 7.4 ) uses backward adaption to determine the short term predictor coefficients and the long term predictor delay. However forward adaption is used to find the LTP gain, which is jointly determined along with the fixed codebook index and gain. The LTP gain and the fixed codebook gain are jointly vector quantized using 4, 5, 6 or 7 bit quantizers, which implies using between 1 and 4 extra bits per 10 sample vector.

Finally the Scheme Four codec uses the same coding strategy as the Scheme Three codec, but also implements a backward adapted switch between specially trained shape and vector gain codebooks for the voiced and unvoiced segments of speech.

It is clear from Table 7.4 that, for our vector size 10 codec, using extra bits to allow forward adaption of the synthesis filter parameters is the least efficient way of using these extra bits. If we were to use two extra bits the largest gain in the codec's segmental SNR is given if we simply use the Scheme One codec and increase the size of the shape codebook by 2 bits. This gain is almost matched if we allocate one extra bit to both the shape and gain codebooks in the Scheme One codec, and this would increase the codebook search complexity less dramatically than allocating both extra bits to the shape codebook.

In order to give a fair comparison between the different coding schemes at bit rates between 4 and 8 kbits/s we tested the Scheme One, Scheme Three and Scheme Four codecs using 8 bit shape codebooks, 4 bit gain codebooks and vector sizes of 12, 15, 18 and 24 samples. This gave three different codecs at 8, 6.4, 5.3 and 4 kbits/s. Note that as the vector size of the codecs increase, their complexity also increases. Methods of reducing this complexity are possible [77], but have not been studied in our work. The segmental SNRs of our three 4-8 kbits/s codecs against their bit rates is shown in Figure 7.8.

Several observations can be made from this graph. At 8 kbits/s, as expected from the results in Table 7.4, the Scheme One codec gives the best quality reconstructed speech, with a segmental SNR of 14.55 dB. However as the vector size is increased and hence the bit rate reduced it is the Scheme One codec whose performance is most badly affected. At 6.4 kbits/s and 5.3 kbits/s all three codecs give very similar segmental SNRs, but at 4 kbits/s the Scheme One codec is clearly worse than the other codecs, which use forward adaption of the LTP gain. This indicates that although the three tap backward adapted LTP is very effective at 8 kbits/s and above, it is less effective as the bit rate is reduced. Furthermore the backward adaptive LTP scheme is more prone to channel error propagation.

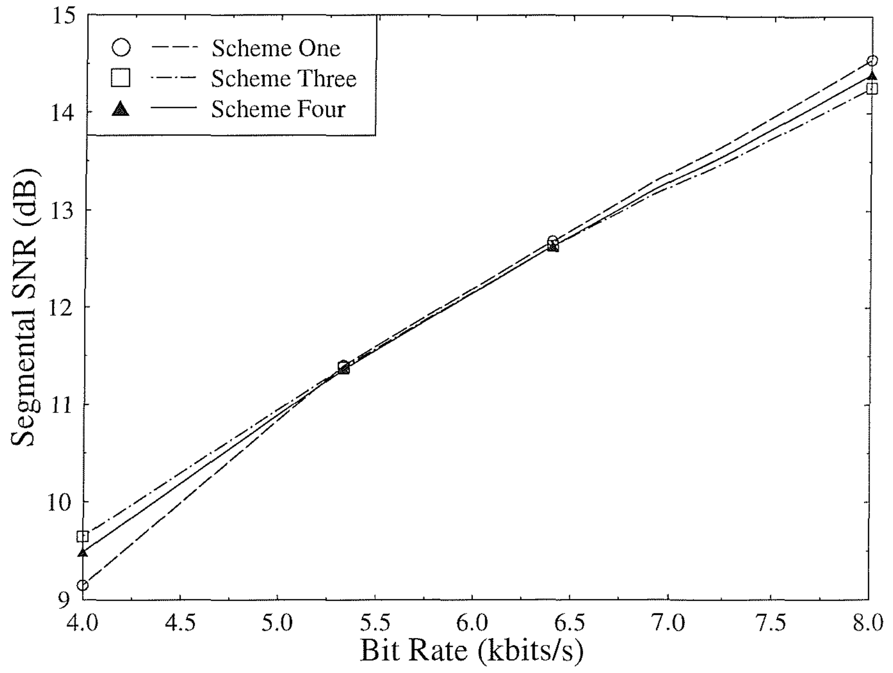

Figure 7.8: Performance of Schemes One Three and Four Codecs at 4-8 kbits/s

Similarly, as indicated in Table 7.4, the backward adaptive switching between specially trained voiced and unvoiced gain and shape codebooks improves the performance of our Scheme Four codec at 8 kbits/s so that it gives a higher segmental SNR than the Scheme Three codec. However as the bit rate is reduced the gain due to this codebook switching is eroded, and at 4 kbits/s the Scheme Four codec gives a lower segmental SNR than the Scheme Three codec. This is due to inaccuracies in the backward adaptive voicing decisions at the lower bit rates. Figure 7.9 shows the same segment of speech as was shown in Figure 7.6, and the normalised autocorrelation value $\beta_\alpha$ calculated from the reconstructed speech of our Scheme Four codec at 4 kbits/s. It can be seen that the condition $\beta_\alpha > 0.6$ no longer gives a good indication of the voicing of the speech. Again for clarity of display the values of $\beta_\alpha$ have been limited to between 0.05 and 0.95 in this figure.

In listening tests we found that all three codecs gave near toll quality speech at 8 kbits/s, with differences between the codecs being difficult to distinguish. However at

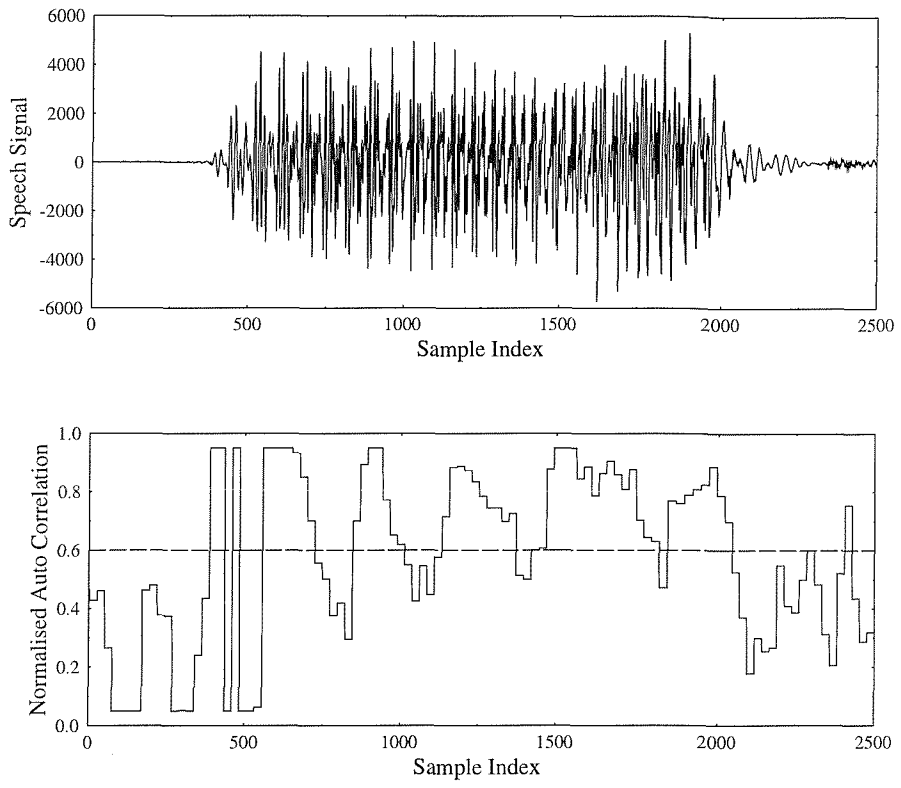

Figure 7.9: Normalised Autocorrelation Value $\beta_\alpha$ During Voiced and Unvoiced Speech

4 kbits/s the Scheme Two codec sounded clearly better than the Scheme One codec, and gave reconstructed speech of communications quality.

## 7.6 A Low Delay ACELP Codec

In this final section of our work on Low Delay CELP codecs operating between 4 and 8 kbits/s we implemented a low delay version of our Algebraic CELP (ACELP) codec which was described in Chapter 3. We developed a series of low delay codecs with a frame size of 40 samples or 5 ms, and hence a total delay of about 15 ms, and with various bit rates between 5 and 6 kbits/s. All of these codecs use backward adaption with the recursive windowing function described in Section 6.2.1 to determine the coefficients for the synthesis filter, which has an order $p = 20$. Furthermore they use the same weighting filter, which was described in Section 6.2.3, as our other low delay codecs. However apart from this they have a structure similar to the codecs described

| Pulse Number i | Amplitude | Possible Position $m_i$ |
|---|---|---|
| 0 | +1 | 1,6,11,16,21,26,31,36 |
| 1 | -1 | 2,7,12,17,22,27,32,37 |
| 2 | +1 | 3,8,13,18,23,28,33,38 |
| 3 | -1 | 4,9,14,19,24,29,34,39 |

Table 7.5: Pulse Amplitudes and Positions for the 12 bit ACELP Codebook

in Chapter 3. An adaptive codebook is used to represent the long term periodicities of the speech, with possible delays taking all integer values between 20 and 147 and being represented using 7 bits. As described in Chapter 3 the best delay is calculated once per 40 sample vector within the Analysis-by-Synthesis loop at the encoder, and then transmitted to the decoder.

Initially we used the 12 bit ACELP fixed codebook structure shown in Table 3.2 which is repeated here in Table 7.5. Each 40 sample vector has a fixed codebook signal given by 4 non-zero pulses of amplitude +1 or -1, whose possible positions are shown in Table 7.5. Each pulse position is encoded with 3 bits giving a 12 bit codebook. As explained in Section 3.6, the pulse positions can be found using a series of four nested loops, leading to a very efficient codebook search algorithm [78, 1].

In our first low delay ACELP codec, which we refer to as Codec A, we used the same 3 and 5 bit scalar quantizers as were used in the codecs in Chapter 3 to quantize the adaptive and fixed codebook gains $G_1$ and $G_2$. This meant that 12 bits were required to represent the fixed codebook index, 7 bits for the adaptive codebook index and a total of 8 bits to quantize the two codebook gains. This gave a total of 27 bits to represent each 40 sample vector, giving a bit rate for this codec of 5.4 kbits/s. We found that this codec gave an average segmental SNR of 10.20 dB, which should be compared to the average segmental SNRs for the same speech files of 9.83 dB and 11.42 dB for our 4.7 kbits/s and 7.1 kbits/s forward adaptive ACELP codecs described in Chapter 3. All of these codecs have a similar level of complexity, but the backward adaptive 5.4 kbits/s ACELP codec has a frame size of only 5 ms, compared to the frame sizes of 20 and 30 ms for the 7.1 and 4.7 kbits/s forward adaptive systems. Furthermore it can be seen from Figure 7.10 that, upon interpolating the segmental SNRs between the two forward adaptive ACELP codecs, the backward adaptive ACELP codec at 5.4 kbits/s gives a very similar level of performance to the forward adaptive codecs. In this figure we have marked the segmental SNRs of the two

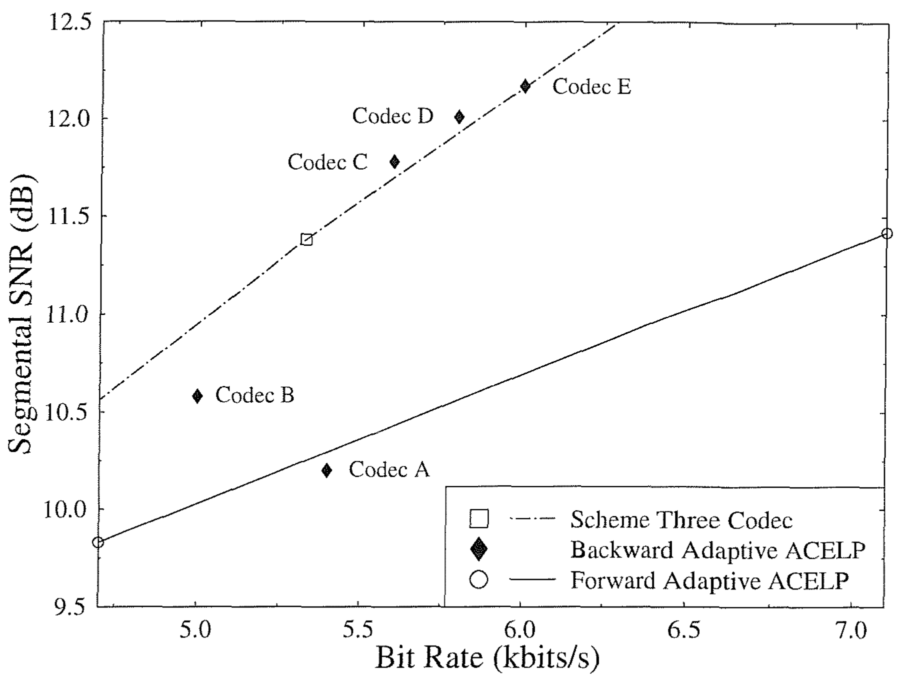

Figure 7.10: Performance of Low Delay ACELP codecs

forward adaptive ACELP codecs with circles, and the segmental SNR of our low delay ACELP codec at 5.4 kbits/s with a black diamond. Also marked with diamonds are the segmental SNRs and bit rates of other backward adaptive ACELP codecs which will be described later. For comparison the performance of the Scheme Three low delay codec, described in Section 7.5 and copied from Figure 7.8, is also shown.

It can be seen from Figure 7.10 that although the 5.4 kbits/s low delay backward adaptive ACELP codec described above gives a similar performance in terms of speech quality to the higher delay forward adaptive ACELP codecs, it performs significantly worse than the Scheme Three codec of Table 7.4, which uses a shorter vector size and a trained shape codebook. We therefore attempted to improve the performance of our low delay ACELP codec by introducing vector quantization and joint determination of the two codebook gains $G_1$ and $G_2$. Note that similar vector quantization and joint determination of these gains was used in the Scheme Three and Scheme Four codecs described in Section 7.5. We also re-introduced the backward adaption of the fixed codebook gain $G_2$ which was used in our other low delay codecs and is described in Section 6.2.2. We replaced the 3 and 5 bit scalar quantizers for $G_1$ and $G_2$ with a 6 bit joint vector quantizer for these gains, which resulted in a total of 25 bits being used to represent each 40 sample vector and therefore gave us a 5 kbits/s codec. We refer to this as Codec B. The joint 6 bit vector quantizer for the gains was trained as described in Section 7.4.2. A joint codebook search procedure was used so that for each fixed codebook index $k$ the joint gain codebook was searched to find the gain codebook index which minimised the weighted error for that fixed codebook index. The best shape and gain codebook indices are therefore determined together. This codebook search procedure results in a large increase in the complexity of the codec, but also significantly increases the performance of the codec.

We found that our 5 kbits/s Codec B, using joint vector quantization of $G_1$ and $G_2$ and backward adaption of $G_2$, gave an average segmental SNR of 10.58 dB. This is higher than the segmental SNR of the codec with scalar gain quantization, ie Codec A, despite codec B having a lower bit rate. The performance of this Codec B is marked with a diamond in Figure 7.10, which shows that it falls between the segmental SNRs of the ACELP codecs with scalar gain quantization and the Scheme Three codecs.

Next we replaced the 12 bit algebraic codebook detailed in Table 7.5 with the 15 bit codebook recommended in the 8 kbits/s ACELP codec [66, 76] submitted as a candidate for CCITT standardisation. This codebook also uses only 4 non-zero pulses per 40 sample vector, but the pulses have a greater range of potential positions they can take within the vector. Furthermore, rather than all the pulses having a fixed sign, one of them can be either +1 or -1. This 15 bit algebraic codebook structure is shown in Table 7.6. The first two pulses can take any one of ten positions each, and their positions are encoded with 7 bits. The third and fourth pulses can take one of eleven positions each, and their positions are also encoded with 7 bits. Notice that the last possible position for these two pulses falls outside the vector boundary. When this position is chosen by the codebook search it indicates that the pulse is missing and that the vector has only three or even two non-zero pulses. Finally one bit is used to encode the sign of the fourth pulse. This results in a total of 15 bits being used to represent the output from the codebook.

This 15 bit codebook structure increases the bit rate of the codec by 600 bits per second, and gives a significant increase in the performance of the codec. We trained

| Pulse Number i | Amplitude | Possible Position $m_i$ |
|---|---|---|
| 0 | +1 | 0,4,8,12,16,20,24,28,32,36 |
| 1 | -1 | 1,5,9,13,17,21,25,29,33,37 |
| 2 | +1 | 2,6,10,14,18,22,26,30,34,38,(42) |
| 3 | +1 or -1 | 3,7,11,15,19,23,27,31,35,39,(43) |

Table 7.6: Pulse Amplitudes and Positions for the 15 bit ACELP Codebook

a 6 bit joint gain vector quantizer for use with this 15 bit codebook, and found that the average segmental SNR of the resulting 5.6 kbits/s codec, referred to as Codec C, was 11.78 dB. This is marked by another diamond on Figure 7.10. It can be seen from the figure that the use of the 15 bit codebook has resulted in a codec with very similar performance to the Scheme Three codec.

Note that the use of a 15 bit algebraic codebook increases the complexity of the codec by approximately a factor of 8 in comparison to the 12 bit codebook used in Codec B. Together with the use of a fairly large vector quantizer, which is searched for every fixed codebook index, to represent the two gains $G_1$ and $G_2$ this results in an extremely complex codec. However in [78] a focused search procedure for an algebraic codebook is described, and it is reported in [76] that only 4 % of the 15 bit codebook described above needs to be searched to give a " performance equivalent to that of the full search". Also after the adaptive codebook delay $\alpha$ is found, the value for the adaptive codebook gain which would be used if $G_1$ and $G_2$ were sequentially determined can be calculated. This sequential LTP gain value is given in Equation 7.1 as $G_1 = C_\alpha/\xi_\alpha$, where the symbols $C_\alpha$ and $\xi_\alpha$ have their usual meanings. This gain value could then be used to limit the portion of the gain vector codebook that is searched to only those entries which have a $G_1$ value close to $C_\alpha/\xi_\alpha$. With the use of this technique and a focused algebraic codebook search the codec complexity can be significantly reduced.

Next we investigated the performance of our low delay ACELP codec with a 15 bit shape codebook at two slightly higher bit rates. First the size of the vector quantizer for the two gains was increased from 6 to 7 bits, resulting in the 5.8 kbits/s Codec D which gave an average segmental SNR of 12.01 dB. Finally as described in reference [66] an extra sign bit was introduced to represent the global sign of the algebraic codebook entry. This bit is effectively used to represent the sign of $G_2$, and therefore doubled the resolution of the quantization of $G_2$. The sign of $G_2$ is determined by

| | Algebraic Codebook | Gain Quantization | Bit Rate (kbits/s) | Segmental SNR |
|---|---|---|---|---|
| Codec A | 12 Bit | 3+5 Bit Scalar | 5.4 | 10.20 dB |
| Codec B | 12 Bit | 6 Bit Vector | 5 | 10.58 dB |
| Codec C | 15 Bit | 6 Bit Vector | 5.6 | 11.78 dB |
| Codec D | 15 Bit | 7 Bit Vector | 5.8 | 12.01 dB |
| Codec E | 15 Bit | 7+1 Bit Vector | 6 | 12.17 dB |

Table 7.7: Performance and Structure of Low Delay ACELP Codecs

testing the sign of $C_k$ for each codebook entry. This, along with the 15 bit algebraic codebook and the 7 bit joint gain codebook, gave a bit rate of 6 kbits/s. We found that the resulting codec, referred to as Codec E, gave an average segmental SNR of 12.17 dB. The segmental SNRs of both these higher bit rate codecs are marked in Figure 7.10 with diamonds. It can be seen that increasing the bit rate of the codec as described above gives an almost linear increase in the segmental SNR of the codecs using a 15 bit algorithmic codebook.

The characteristics of our low delay ACELP codecs are summarised in Table 7.7. We found that our 6 kbits/s low delay ACELP Codec E gave reconstructed speech of good communications quality. This codec is very similar to the 8 kbits/s low delay ACELP codec described in [66, 76] which is being considered for CCITT standardisation. The main difference is that the codec in [66] uses forward adaption of the synthesis filter parameters, and as a result has a delay of more than double our codec as well as a bit rate 2 kbits/s higher. However for the cost of these disadvantages the forward adaption used for the LPC parameters in [66] results in both a lower vulnerability to channel errors, and a higher speech quality. Nonetheless our 6 kbits/s codec provides a useful alternative, when a lower speech quality is acceptable and a higher channel quality can be guaranteed.

## 7.7 Conclusion

In this chapter we have described several low delay coding schemes operating between 4 and 8 kbits/s. Codecs using both forward and backward adaption of the long term filter have been considered, but all the codecs use backward adaption of the short term synthesis filter and so have frame sizes of at most 5 ms. Both relatively small trained shape codebooks and large algebraic codebooks were used. We found that the resulting codecs offered a range of reconstructed speech qualities between communications quality at 4 kbits/s to almost toll quality at 8 kbits/s.

# Chapter 8

# Conclusions and Suggestions for Further Work

## 8.1 Summary and Conclusions

In this thesis we have studied the coding of narrow-band speech, which has been sampled at 8 kHz, at bit rates between 4 and 16 kbits/s. The work has concentrated on codecs using the Code Excited Linear Prediction (CELP) [2] algorithm. These codecs use an adaptive linear synthesis filter to model the vocal tract of the speaker, and an Analysis-by-Synthesis (AbS) structure to determine the excitation signal which is fed to the synthesis filter to produce the reconstructed speech. The excitation signal is vector quantized typically using two codebooks. One of these codebooks is adaptive, and is used to model the long term periodicities which are present in voiced speech due to the periodic opening and closing of the vocal cords. The other codebook is very large, typically 1024 entries or more, and due to the complexity of the AbS search of such a large codebook special structures are used to simplify the search. One such structure is the algebraic codebook structure [79], and we describe two low complexity Algebraic CELP (ACELP) codecs operating at 4.7 and 7.1 kbits/s. The 4.7 kbits/s codec gives an average segmental SNR of 9.83 dB, and produces reconstructed speech of good communications quality. The 7.1 kbits/s codec gives an average segmental SNR of 11.42 dB, and produces speech which is noticeably more transparent than the lower rate codec.

An important aspect of CELP codecs is the AbS structure which is used to determine the excitation signal for the synthesis filter in such a way that the perceptual error between the original speech and the reconstructed speech is minimised. We investigated several ways of extending this AbS loop in our 4.7 kbits/s ACELP codec so that the signals from the two codebooks are optimized together. Several techniques of various complexities, and giving different improvements in the reconstructed speech quality, were proposed. We found that it was possible to improve the segmental SNR of our codec by 0.5 dB, and give a noticeable improvement in its perceptual quality, but at the cost of increasing the codec complexity by 40 %. We also studied methods of including the determination of the synthesis filter parameters in the AbS loop by updating these parameters once the excitation signal which is fed to the filter is known. However we found that although it was possible to produce a significant improvement in the codec's performance using such update schemes, similar improvements could also be achieved using the common techniques of bandwidth expansion and Line Spectral Frequency (LSF) interpolation.

A very important application of low bit rate codecs is in digital mobile telephony. Such codecs are used over the extremely hostile mobile radio channel, and so must be reasonably robust to errors introduced by the channel between the encoder and the decoder. We studied in detail this aspect of our ACELP codecs, and ways of making the transmitted parameters of these codecs less susceptible to errors. We also proposed a new method of measuring the sensitivity of various bits within a codec's bit stream to errors. This new method attempts to take proper account of the long memory of some codec parameters to errors, and allows speech and channel codecs to be finely matched in order to achieve as high a system performance as possible.

An important part of CELP codecs is the linear predictive synthesis filter which is employed to model the effects of the vocal tract of the speaker [4]. This filter is adaptive, and we have studied codecs which use forward adaption of the filter, as well as other codecs which use backward adaption. The ACELP codecs above are examples of forward adaptive systems. Such codecs need to buffer a frame of input speech at the encoder, typically 20 to 30 ms long, and this leads to codecs with a total delay of the order of 60 to 90 ms. Such a high delay can be unacceptable in some applications, for example in two way communication systems where echoes are present. Thus codecs with a lower delay are also important, and such codecs tend to use backward adaption of the synthesis filter parameters in order to avoid the need to buffer a long frame of speech at the encoder. We have studied in detail the recently standardised CCITT 16 kbits/s low delay CELP codec [3], and proposed an extension to this codec to allow it to operate as a variable rate codec between 8 and 16 kbits/s. At 8 kbits/s if the delay of the codec is kept constant at less than 2 ms then an average segmental SNR of 12.56 dB is achieved. If the delay of the codec is increased as its bit rate is decreased then at 8 kbits/s it has a delay of less than 4 ms, and gives an average segmental SNR of 14.29 dB.

We conclude this thesis with a study of low delay codecs operating between 4 and 8 kbits/s. Several coding schemes are introduced, and we show that it is possible to achieve a segmental SNR of 9.65 dB at 4 kbits/s with a total delay of less than 10 ms. In listening tests this variable rate codec gives communications quality speech at 4 kbits/s, which rises to close to toll quality speech at 8 kbits/s. We also propose a backward adaptive ACELP codec with a delay of about 15 ms. At 6 kbits/s this codec gives reconstructed speech of good communications quality with an average segmental SNR of 12.17 dB.

## 8.2 Suggestions for Further Work

The low delay codecs we described in Chapters 6 and 7 offer reasonable speech quality at bit rates between 4 and 16 kbits/s. However one area in which backward adaptive systems traditionally suffer is their vulnerability to channel errors, and this aspect of our codecs needs to be investigated. Work is currently progressing in this area. Also as the bit rate of these codecs is reduced towards 4 kbits/s the complexity of the search of the trained codebooks used in some of the codecs increases. Methods of reducing this complexity need to be investigated before the codecs would be practical for real time systems.

Despite hectic activity in speech coding research for the the past ten years there are still many problems which remain unsolved, and areas in which further work could yield good results. One such area is the integration of speech codecs into communications systems so that the speech codec, channel codec and modulation schemes are optimised together in order to provide the best possible system performance. Another useful area for further research is in our perception of the quality of coded speech signals. Noise weighting filters are already employed in CELP codecs, and lead to a significant improvement in the quality of these codecs. However although these weighting filters go some way towards modelling the human ear's perception of speech, much work remains to be done.

# Bibliography

[1] Raymond Steele, *Mobile Radio Communications.* Pentech Press, London, 1992.

[2] Manfred R. Schroeder and Bishnu S. Atal, "Code Excited Linear Prediction (CELP): High-Quality Speech at Very Low Bit Rates," *Proc. ICASSP*, pp. 937–940, 1985.

[3] Juin-Hwey Chen, Richard V. Cox, Yen-Chun Lin, Nikil Jayant and Melvin J. Melchner, "A Low-Delay CELP Coder for the CCITT 16 kb/s Speech Coding Standard," *IEEE Journal on Selected Areas in Communications*, pp. 830–849, June 1991.

[4] L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals.* Prentice-Hall, 1978.

[5] Shihua Wang, Andrew Sekey and Allen Gersho, "An Objective Measure for Predicting Subjective Quality of Speech Coders," *Journal on Selected Areas in Communications*, pp. 819–829, 1992.

[6] Douglas O'Shaughnessy, *Speech Communication, Human and Machine.* Addison-Wesley Publishing Company, 1987.

[7] Jerry V. Tobias, *Foundations of Modern Auditory Theory.* Academic Press, 1970.

[8] Y. Be'ery, Z. Shpiro, T. Simchony, L. Shatz and J. Piasetzky, "An Efficient Variable-Bit-Rate Low-Delay CELP," in *Advances in Speech Coding* (Bishnu S. Atal, Vladimir Cuperman and Allen Gersho, ed.), pp. 257–266, Kluwer Academic Publishers, 1991.

[9] Nobuhiko Kitawaki, Masaaki Honda and Kenzo Itoh, "Speech-Quality Assessment Methods for Speech Coding Systems," *IEEE Communications Magazine*, pp. 26–33, 1984.

[10] Augustine H. Gray and John D. Markel, "Distance Measures for Speech Processing," *IEEE Transactions on Acoustics, Speech and Signal Processing*, pp. 224–231, Oct 1976.

[11] Nobuhiko Kitawaki, Hiromi Nagabuchi and Kenzo Itoh, "Objective Quality Evaluation for Low-Bit-Rate Speech Coding Systems," *Journal on Selected Areas in Communications*, pp. 242–248, 1988.

[12] M.R. Schroeder, B.S. Atal and J.L. Hall, "Optimizing Digital Speech Coders by Exploiting Masking Properties of the Human Ear," *Journal of the Acoustic Society of America*, pp. 1647–1652, 1979.

[13] D. Sen and W.H. Holmes, "PERCELP-Perceptually Enhanced Random Codebook Excited Linear Prediction," *Proc. IEEE Workshop on Speech Coding for Telecommunications*, pp. 101–102, 1993.

[14] D. Sen and W.H. Holmes, "Perceptual Enhancement of CELP Speech Coders," *Proc. ICASSP*, vol. 2, pp. 105–108, 1994.

[15] Jon E. Natvig, "Evaluation of Six Medium Bit-Rate Coders for the Pan-European Digital Mobile Radio System," *IEEE Journal on Selected Areas in Communications*, pp. 324–331, Feb 1988.

[16] N.S. Jayant and Peter Noll, *Digital Coding of Waveforms.* Prentice-Hall, 1984.

[17] "CCITT Recommendation G.721."

[18] R. Steele, *Delta Modulation Systems.* Halstad Press, New York, 1975.

[19] Raymond Steele, "Speech Codecs for Personal Communications," *IEEE Communications Magazine*, pp. 76–83, Nov 1993.

[20] Bishnu S. Atal and Joel R. Remde, "A New Model of LPC Excitation for Producing Natural-Sounding Speech at Low Bit Rates," *Proc. ICASSP*, pp. 614–617, 1982.

[21] P. Kroon and E.F. Deprettere, "Regular Pulse Excitation - A Novel Approach to Effective Efficient Multipulse Coding of Speech," *IEEE Trans. on Acoustics, Speech and Signal Processing*, pp. 1054–1063, 1986.

[22] Bishnu S. Atal and Manfred R. Schroeder, "Predictive Coding of Speech Signals and Subjective Error Criteria," *IEEE Transactions on Acoustics, Speech and Signal Processing*, pp. 247–254, June 1979.

[23] Sharad Singhal and Bishnu S. Atal, "Amplitude Optimization and Pitch Prediction in Multipulse Coders," *IEEE Trans. on Acoustics, Speech and Signal Processing*, pp. 317–327, Mar 1989.

[24] "Federal Standard 1016 – Telecommunications: Analog to Digital Conversion of Radio Voice by 4,800 Bits/Second Code Excited Linear Prediction (CELP)," February 14 1991.

[25] S. Wang and A. Gersho, "Phonetic Segmentation for Low Rate Speech Coding," in *Advances in Speech Coding* (Bishnu S. Atal, Vladimir Cuperman and Allen Gersho, ed.), pp. 257–266, Kluwer Academic Publishers, 1991.

[26] Peter Lupini, Hisham Hassanein and Vladimir Cuperman, "A 2.4 kbit/s CELP Speech Codec with Class-Dependent Structure," *Proc. ICASSP*, vol. 2, pp. 143–146, 1993.

[27] Daniel W. Griffin and Jae S. Lim, "Multiband Excitation Vocoder," *IEEE Trans. on Acoustics, Speech and Signal Processing*, pp. 1223–1235, Aug 1988.

[28] Masayuki Nishiguchi, Jun Matsumoto, Ryoji Wakatsuki and Shinobu Ono, "Vector Quantized MBE with Simplified V/UV Division at 3.0KBPS," *Proc. ICASSP*, pp. II–151–II–154, 1993.

[29] W. Bastiaan Kleijn, "Encoding Speech Using Prototype Waveforms," *IEEE Trans. on Acoustics, Speech and Signal Processing*, pp. 386–399, Oct 1993.

[30] T. Berger, *Rate Distortion Theory, A Mathematical Basis for Data Compression.* Prentice-Hall, 1971.

[31] Peter Noll and Rabiner Zelinski, "Bounds on Quantizer Performance in the Low Bit-Rate Region," *IEEE Transactions on Communications*, pp. 300–304, Feb 1978.

[32] Timothy Thorpe, "The Mean Squared Error Criterion: Its Effect On The Performance of Speech Coders," *Proc. ICASSP*, pp. 77–80, 1989.

[33] John O'Neal, "Bounds on Subjective Performance Measures for Source Encoding Systems," *IEEE Transactions on Information Theory*, pp. 224–231, May 1971.

[34] John Makhoul, Salim Roucos and Herbert Gish, "Vector Quantization in Speech Coding," *Proceedings of the IEEE*, pp. 1551–1588, Nov 1985.

[35] Masami Akamine and Kimio Miseki, "ARMA Model Based Speech Coding at 8kb/s," *Proc. ICASSP*, pp. 148–151, 1989.

[36] John Makhoul, "Linear Prediction: A Tutorial Review," *Proceedings of the IEEE*, pp. 561–580, April 1975.

[37] Stephen W. Lang and James H. McClellan, "A Simple Proof for the Stability of All-Pole Linear Prediction Models," *Proceedings of the IEEE*, pp. 860–861, May 1979.

[38] J.D. Markel and A.H. Gray, Jr., *Linear Prediction of Speech.* Springer-Verlag, 1976.

[39] F. Itakura, "Line Spectrum Representation of Linear Predictive Coefficients of Speech Signals," *Journal of the Acoustic Society of America, Vol 57*, p. S35, 1975.

[40] Frank K. Soong and Biing-Hwang Juang, "Line Spectrum Pair (LSP) and Speech Data Compression," *Proc. ICASSP*, pp. 1.10.1–1.10.4, 1984.

[41] Kuldip K. Paliwal and Bishnu S. Atal, "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame," *IEEE Transactions on Speech and Audio Processing, Vol 1,No 1*, pp. 3–14, Jan 1993.

[42] R.A. Salami, L. Hanzo and D.G. Appleby, "A Computationally Efficient CELP Codec With Stochastic Vector Quantization of LPC Parameters," *URSI Int. Symposium on Signals, Systems and Electronics*, pp. 140–143, 1989. Sept 18-20 1989, Erlangen, West Germany.

[43] J.P. Adoul, P. Mabilleau, M. Delprat and S. Morissette, "Fast CELP Coding Based on Algebraic Codes," *Proc. ICASSP*, pp. 1957–1960, April 1987.

[44] P. Kabal, J.L. Moncet and C.C. Chu, "Synthesis Filter Optimization and Coding: Applications to CELP," *Proc. ICASSP*, vol. 1, pp. 147–150, 1988.

[45] Yoh'Ichi Tohkura, Fumitada Itakura and Shin'Ichiro Hashimoto, "Spectral Smoothing Technique in PARCOR Speech Analysis-Synthesis," *IEEE Trans. on Acoustics, Speech and Signal Processing*, pp. 587–596, 1978.

[46] Juin-Hwey Chen and Richard V.Cox, "Convergence and Numerical Stability of Backward-Adaptive LPC Predictor," *IEEE Workshop on Speech Coding for Telecommunications*, pp. 83–84, 1993.

[47] Sharad Singhal and Bishnu S. Atal, "Optimizing LPC Filter Parameters for Multi-Pulse Excitation," *Proc. ICASSP*, pp. 781–784, 1983.

[48] M. Fratti, G.A. Miani and G. Riccardi, "On The Effectiveness of Parameter Reoptimization in Multipulse Based Coders," *Proc. ICASSP*, vol. 1, pp. 73–76, 1992.

[49] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery, *Numerical Recipes in C.* Cambridge University Press, 1992.

[50] Gene H. Golub and Charles F. Van Loan, "An Analysis of the Total Least Squares Problem," *SIAM Journal of Numerical Analysis*, vol. 17, no. 6, pp. 883–890, 1980.

[51] MD. Anisur Rahham and Kai-Bor Yu, "Total Least Squares Approach for Frequency Estimation Using Linear Prediction," *IEEE Transactions on Acoustics, Speech and Signal Processing*, pp. 1440–1454, 1987.

[52] Ronald D. Degroat and Eric M. Dowling, "The Data Least Squares Problem and Channel Equalization," *IEEE Transactions on Signal Processing*, pp. 407–411, 1993.

[53] F.F. Tzeng, "Near-Optimum Linear Predictive Speech Coding," *IEEE Global Telecommunications Conference*, pp. 508.1.1–508.1.5, 1990.

[54] Mahesan Niranjan, "CELP Coding with Adaptive Output-Error Model Identification," *Proc. ICASSP*, pp. 225–228, 1990.

[55] J.P. Woodard and L. Hanzo, "Improvements to the Analysis-by-Synthesis Loop in CELP codecs," *Proceedings of the IEEE International Conference on Radio Receivers and Associated Systems*, pp. 114–118, September 1995.

[56] L. Hanzo, R. Salami, Prof R. Steele and P.M. Fortune, "Transmission of Digitally Encoded Speech at 1.2 Kbaud for PCN," *IEE Proceedings-I, Vol 139,No.4*, pp. 437–447, 1992.

[57] Richard V. Cox, W. Bastiaan Kleijn and Peter Kroon, "Robust CELP Coders for Noisy Backgrounds and Noisy Channels," *Proc. ICASSP*, pp. 739–742, 1989.

[58] Joseph Campbell, Vanoy Welch and Thomas Tremain, "An Expandable Error-protected 4800 bps CELP Coder (U.S. Federal Standard 4800 bps Voice Coder)," *Proc. ICASSP*, pp. 735–738, 1989.

[59] S.A. Atungsiri, A.M. Kondoz and B.G. Evans, "Error Control For Low-Bit-Rate Speech Communication Systems," *IEE Proceedings-I*, vol. 140, pp. 97–104, April 1993.

[60] Redwan Ali Salami, *Robust Low Bit Rate Analysis-by-Synthesis Predictive Speech Coding*. PhD thesis, University of Southampton, 1990.

[61] L.K. Ong, A.M. Kondoz and B.G. Evans, "Enhanced Channel Coding Using Source Criteria in Speech Coders," *IEE Proceedings-I, Vol 141,No.3*, pp. 191–196, June 1994.

[62] W.B. Kleijn, "Source-Dependent Channel Coding and its Application to CELP," in *Advances in Speech Coding* (Bishnu S. Atal, Vladimir Cuperman and Allen Gersho, ed.), pp. 257–266, Kluwer Academic Publishers, 1991.

[63] J. Woodard and L. Hanzo, "A Dual-Rate Algebraic CELP-based Speech Transceiver," *Proceeding of the IEEE Conference on Vehicular Technology*, vol. 3, pp. 1690–1694, June 1994.

[64] L. Hanzo and J.P. Woodard, "An Intelligent Multimode Voice Communications System for Indoor Communications." To be published in the IEEE Trans. on Vehicular Technology, Nov 1995.

[65] "Coding of Speech at 16 kbit/s Using Low-Delay Code Excited Linear Prediction." CCITT Recommendation G.728, 1992.

[66] Redwan Salami, Claude Laflamme, Jean-Pierre Adoul and Dominique Massaloux, "A Toll Quality 8 Kb/s Speech Codec for the Personal Communications System (PCS)," *IEEE Transactions on Vehicular Technology*, pp. 808–816, August 1994.

[67] "Terms of Reference of the Ad Hoc Group on 16 kbits/s Speech Coding (Annex 1 to Question 21/XV)." CCITT Study Group XVIII, June 1988.

[68] C. Hong, *Low Delay Switched Hybrid Vector Excited Linear Predictive Coding of Speech.* PhD thesis, National University of Singapore, 1994.

[69] Jian Zhang and Hong Shen Wang, "A Low Delay Speech Coding System at 4.8 Kb/s," *Proceedings of the IEEE International Conference on Communications Systems*, vol. 3, pp. 880–883, Nov 1994.

[70] Juin-Hwey Chen and Allen Gersho, "Gain-Adaptive Vector Quantization with Application to Speech Coding," *IEEE Transactions on Communications*, pp. 918–930, Sept 1987.

[71] T.P. Barnwell III, "Recursive Windowing for Generating Autocorrelation Coefficients for LPC analysis," *IEEE Transactions on Acoustics, Speech and Signal Processing*, pp. 1062–1066, Oct 1981.

[72] J.P. Woodard, J.M. Torrance and L. Hanzo, "A Low Delay Multimode Speech Terminal." To be published in the Proceedings of the IEEE Vehicular Technology Conference, April 28 - May 1 1996.

[73] Y. Linde, A. Buzo and R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE. Trans. Communications*, Jan 1980.

[74] Juin-Hwey Chen, "High-Quality 16 Kb/s Speech Coding with a One-Way Delay Less Than 2ms," *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, pp. 453–456, 1990.

[75] Juin-Hwey Chen and Allen Gersho, "Adaptive Postfiltering for Quality Enhancement of Coded Speech," *IEEE Transactions on Speech and Audio Processing*, pp. 59–71, Jan 1995.

[76] R. Salami, C. Laflamme and J-P. Adoul, "8 kbits/s ACELP Coding of Speech with 10 ms Speech Frame: A Candidate for CCITT Standardization," *Proc. ICASSP*, vol. 2, pp. 97–100, 1994.

[77] W. Bastiaan Kleijn, Daniel J. Krasinski and Richard H. Ketchum, "Fast Methods for the CELP Speech Coding Algorithm," *IEEE Trans. on Acoustics, Speech and Signal Processing*, pp. 1330–1342, August 1990.

[78] C. Laflamme, J-P. Adoul, R. Salami, S. Morissette and P. Mabilleau, "16 KBPS Wideband Speech Coding Technique Based on Algebraic CELP," *Proc. ICASSP*, pp. 13–16, 1991.

[79] C. Laflamme, J-P. Adoul, H.Y. Su and S. Morissette, "On Reducing the Complexity of Codebook Search in CELP Through The Use of Algebraic Codes," *Proc. ICASSP*, pp. 177–180, 1990.