1

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

✓ S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

**An Advanced Speech Coder Based On A
Rate-Distortion Theory Framework**

by

Wilfrid Paul LeBlanc, B.A.Sc.

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
Master of Engineering

Ottawa-Carleton Institute for Electrical Engineering,
Faculty of Engineering,
Department of Systems and Computer Engineering,
Carleton University,
Ottawa, Ontario, Canada

March 15, 1988

© Wilfrid Paul LeBlanc, 1988

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-40644-5

The undersigned hereby recommends to the Faculty of Graduate
Studies and Research acceptance of the thesis,

"A Speech Coder Based on a
Rate-Distortion Theory Framework"

submitted by Wilfrid Paul LeBlanc, B.A.Sc., in partial fulfilment
of the requirements for the degree of Master of Engineering.

Thesis Supervisor

Chairman,
Department of Systems and
Computer Engineering

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
April 16, 1988

Abstract

A speech coder operating at a bit rate of between 6 and 8 kbps is developed in this study. The speech coder is based on a modified version of Code-Excited Linear Prediction (CELP), and is referred to as KELP. The speech coder has excellent performance with relatively high computational complexity and large memory requirements. Vector Quantization (VQ) is used to encode the short term spectral parameters, and the excitation (residual) sequence. The computational complexity has been reduced dramatically over the original design proposed by Atal and Schroeder, with only a small degradation in performance. A real time speech coder could be implemented (based on KELP) using one of the growing number of signal processors available on the market. The algorithms were simulated on a microcomputer and both objective and informal speech intelligibility tests were performed.

Acknowledgements

I would like to thank Professor S.A. Mahmoud for all his guidance and assistance during the course of this thesis. His suggestions and comments were always greatly appreciated.

I would also like to thank NSERC for the Postgraduate Scholarship which made this thesis financially possible.

Contents

1	Introduction	1
1.1	Digital Speech Coding	1
1.2	Thesis Objectives	4
1.3	Thesis Organization	5
2	Introduction To Speech Coding	7
2.1	Digital Communications Model	7
2.2	Speech Production Model	12
2.3	Evaluating Speech Coders	20
2.4	Linear Prediction of Speech	21
2.4.1	Basic Formulation and Solution	21
2.4.2	The Itakura-Saito Distortion Measure	31
2.4.3	The Inverse Filter Matching Principle	39
2.4.4	Alternate Representations of the Model Filter	41
3	Speech Coding Algorithms	46
3.1	Medium to High Bit Rate Speech Coders	46
3.1.1	Differential Pulse Code Modulation	46
3.1.2	Adaptive Predictive Coding	53
3.2	Very Low Bit Rate Speech Coders	55
3.3	Low Bit Rate Speech Coders	59
3.3.1	Multipulse Linear Predictive Coding (MPLPC)	59
3.3.2	Vector Adaptive Predictive Coding (VAPC)	64
3.3.3	Code-Excited Linear Prediction (CELP)	70
4	Vector Quantization	74
4.1	Introduction to Vector Quantization	74

4.2	Codebook Design	84
4.3	Computational and Storage Costs of VQ	89
4.4	Tree Searched VQ	93
4.5	Cascaded VQ	104
4.6	Product Code VQ	106
4.7	Gain Shape VQ	107
4.8	Vector Quantization of the LPC Model Filter	108
5	Optimization Of CELP	121
5.1	CELP Complexity	121
5.2	Reduced Complexity CELP	123
5.3	Codebook Design	135
5.4	Pitch Parameter Determination	138
5.5	Gain Quantization	146
5.6	Development of The Speech Coder	149
6	Summary and Suggestions For Future Research	154
6.1	Summary	154
6.2	Suggestions For Future Research	156
	References	158
	Appendices	164
	A The Autocorrelation Method	164
	B The Covariance Method	166

List of Figures

2.1	Communications Model	8
2.2	Distortion-Rate Function	10
2.3	Speech Production Model	14
2.4	Speech Analysis/Synthesis	15
2.5	Simplified Synthesis Structure	19
2.6	Speech Spectra/ Model Filter Response	28
2.7	Speech Segment	29
2.8	Speech Residual	30
2.9	Identification-Quantization	32
3.1	DPCM	47
3.2	D*PCM	49
3.3	D*PCM, Two Equivalent Noise Models	50
3.4	DPCM-NFC	51
3.5	APC-NFC	54
3.6	Voiced Speech Segment	56
3.7	Residual after Short Term Prediction	57
3.8	Residual after Pitch Prediction	58
3.9	MPLPC Search Procedure	61
3.10	Noise Weighting Filter	62
3.11	Resulting Error Spectrum	63
3.12	Simplified MPLPC Search Procedure	65
3.13	MPLPC Search Procedure Utilizing a Codebook	66
3.14	VAPC	68
3.15	VAPC, Analysis by Synthesis	69
3.16	VAPC, Enhanced Analysis by Synthesis	71
3.17	CELP	72

4.1	High Sample-Sample Correlation	75
4.2	Joint PDF	77
4.3	Joint PDF: Decorrelated	78
4.4	Joint PDF: Non-linear Dependencies	79
4.5	Joint PDF: Vector Quantization	80
4.6	Algorithmic Description of VQ Design	86
4.7	Splitting the Codebook	94
4.8	Codebook Design: Stage 1	96
4.9	Codebook Design: Stage 2	97
4.10	Codebook Design: Stage 3	98
4.11	Binary Tree Search	99
4.12	Non-Uniform Binary Tree Search	101
4.13	2-3-4 Tree Search	103
4.14	Two Stage Cascaded Vector Quantization	105
4.15	Model Filter Cascaded VQ	112
4.16	Distortion Performance Inside The Training Sequence	116
4.17	Distortion Performance Outside The Training Sequence	117
4.18	Speech Spectra, Optimum Model Spectra ($p = 20$)	118
4.19	Speech/Model Spectra, 20 bit product code ($p = 20$)	119
4.20	Speech/Model Spectra, 40 bit LSP ($p = 10$)	120
5.1	CELP, Location of $W(z)$ Moved	125
5.2	CELP, Separate Zero-Input, Zero-State Responses	126
5.3	KELP, Move Location of Codebook	129
5.4	KELP Synthesis Structure	130
5.5	Spectrogram, Voiced Segment, $1/A(z)$	132
5.6	Spectrogram, Voiced Segment, $1/A(z/\gamma)$	133
5.7	KELP, Fixed Weighting Filter	134
5.8	The Long Term Model Filter, $1/B(z)$	139
5.9	Implementing The Long Term Inverse Filter, $B(z)$	141
5.10	Decision Regions, Log Scale	148
5.11	Decision Regions, Non-Log Scale	148

List of Tables

2.1	Classification of Speech Coders	22
4.1	Distortion Measures and Centroids	90
5.1	Performance of The Various Configurations	153

Glossary of Acronyms

APC	- adaptive predictive coding
CELP	- code-excited linear prediction
Codec	- encoder/decoder
DPCM	- differential pulse code modulation
DRT	- diagnostic rhyme test
FFT	- fast (discrete) fourier transform
FLOPS	- floating point operations per second
IS-distortion	- Itakura-Saito distortion
I-distortion	- gain optimized IS-distortion or Itakura distortion
kbps	- kilo-bits per second
KELP	- Modified CELP
kHz	- kilo-hertz, 1000 cycles per second
LPC	- linear predictive coding
MOS	- mean opinion score
MPLPC	- multipulse linear predictive coding
MSAT	- mobile satellite experiment
MSE	- mean squared error

- NFC - noise feedback coding
- NWSNR - noise weighted signal to noise ratio
- PCM - pulse code modulation
- PDF - probability distribution function
- REL P - residually excited linear prediction
- SBC - subband coding
- SFM - spectral flatness measure
- SNR - signal to (quantization) noise ratio
- SNR_{SEG} - segmental signal to noise ratio
- UHF - ultra high frequency
- VAPC - vector adaptive predictive coding
- VHF - very high frequency
- VLSI - very large scale integrated circuit
- Vocoder - voice coder
- VQ - vector quantization/ quantizer
- VXC - vector excitation coding

Glossary of Symbols

$\mathbf{a}^T$	- bold font indicates a vector quantity, superscript T indicates vector or matrix transpose
$E[\cdot]$	- the expectation operator
$s(t)$	- continuous time speech signal
$\tilde{s}(t)$	- continuous time synthesized speech signal
s_n	- sampled (possibly windowed) speech signal
$\tilde{s}_n$	- sampled windowed speech signal
$\hat{s}_n$	- sampled, synthesized speech signal
$R(D)$	- rate-distortion function
$D(R)$	- distortion-rate function
H	- source entropy, ($R(0)$)
R	- the data rate
T	- symbol period (R^{-1})
M	- pitch period
N	- the analysis window size (short term predictor)
N_p	- the buffer size (short term predictor), the amount of overlap is just $N_p - N$
p	- the short term predictor order

$A(z)$	- the inverse (short term) model filter
$B(z)$	- the inverse (long term) model filter
$1/A(z)$	- the (short term) model filter
$P_s(z)$	- short term predictor, prediction based on the spectral envelope ($A(z) = 1 - P_s(z)$)
$P_l(z)$	- long term predictor, prediction based on the spectral fine structure ($B(z) = 1 - P_l(z)$)
q	- the number of pitch predictor coefficients is $2q + 1$
b_k	- the long term predictor parameters
a_k	- the predictor parameters (short term model filter)
k_i	- the reflection coefficients
ω_i	- the line spectral (radian) frequencies (LSFs)
ω	- angular frequency
d_n	- residual after short term prediction
e_n	- residual after both long and short term prediction
v_n	- the innovation (excitation) sequence
G	- the gain
$\hat{e}_n$	- synthesized residual (Gv_n)
$\hat{d}_n$	- synthesized residual after long term model filter
σ_x^2	- variance of x
γ_x^2	- spectral flatness measure
$\phi_{xx}(k)$	- autocorrelation function
$\Phi_{xx}(w)$	- Z -transform of $\phi_{xx}(k)$ evaluated on the unit circle $= X(e^{j\omega}) ^2$

K - vector dimension, codebook dimension

~~L~~ - codebook size, number of codebook levels

Chapter 1

Introduction

1.1 Digital Speech Coding

The emergence of several communications services and applications requiring integrated voice and data transmission have generated a great demand for a high quality digital voice coder able to utilize the bandwidth available in common analog circuits.

There are numerous applications and reasons for a near toll quality low bit rate digital voice coder, such as:

- high quality voice mail (store and forward);
- high quality (real time) voice codec;
- high security voice codec;
- low cost, high reliability digital implementation.

The first two items can be implemented by analog means whereas the latter items necessitate an all digital realization. In fact, the success of

the U.S. government LPC10 standard is evidence that the trend to an all digital *high* security voice coder is indeed required.

For mobile radio and mobile satellite (MSAT) applications a low bit rate voice coder could permit the multiplexing of several voice channels utilizing the currently available 25 kHz (VHF/UHF channel spacing) standards, or a single channel on the 5 kHz MSAT channel spacing. Regardless, it is sufficiently clear that a high quality, low bit rate, real time voice coder is required.

Until recently, the requirement of high quality digital voice forced one to use Adaptive Predictive Coding (APC), Residually Excited Linear Prediction (RELP), Subband Coding (SBC), and so on, with bit rates in the upper upper-medium range of 12-16 kbps (12000-16000 bits per second). However, the second requirement (narrow bandwidth), forced one to utilize the LPC vocoder which is based on the familiar voiced-unvoiced classification of speech which leads to poor quality synthetic voice. The LPC10 standard is based on the LPC Vocoder.

The first set of techniques (APC, RELP, SBC), strive for perfect reproduction of the speech signal and thus a good measure of performance is signal to noise ratio or segmental signal to noise ratio (to be defined in Section 2.3). With Vocoder techniques, a simple model of speech production is utilized whereby the vocal tract parameters are estimated, and voiced/unvoiced classifications are made in an attempt to duplicate the lungs and vocal cords (source excitation) and the vocal tract (linear filter). It will be shown that fundamental limitation of the Pitch Excited Model is due to the poor speech model, and that poor speech results even when infinitely fine quantization of the vocal tract parameters and source excitation parameters is used. For these reasons, signal to noise ratios are not meaningful when evaluating Vocoders, and subjective measures such as the

mean opinion score (MOS), and the diagnostic rhyme test (DRT) are used.

With more efficient source coding techniques which approach the theoretical limits predicted by Rate-Distortion Theory, bit rates in the 8-10 kbps range using the above techniques (APC, SBC, RELP) has become possible with comparable voice quality to the original higher bit rate designs. The complexity, however, has increased substantially. For bit rates in the 6-8 kbps range, (the bit rate required for this study), a growing number of techniques are becoming available such as Multi-Pulse Linear Predictive Coding (MPLPC), Vector Excitation Coding (VXC), Vector Adaptive Predictive Coding (VAPC), and Code-Excited Linear Predictive Coding (CELP). These techniques all require powerful (and computationally expensive) source coding techniques such as Tree Coding, Trellis Coding, or Vector Quantization to approach these low bit rates. These techniques often apply *Analysis by Synthesis* to encode a residual speech vector. In Analysis by Synthesis, synthetic speech vectors are constructed by exciting the synthesis filter by residual speech vectors (innovations) chosen from a codebook. The index of the codebook which produces the minimum distortion between the synthetic speech and the input speech is transmitted (along with the synthesis filter) to the receiver. Since the whole codebook must be searched, the transmitter is much more complex than the receiver.

In this study, we will briefly review Vocoder Techniques and Adaptive Predictive Coding (APC), and discuss in greater detail MPLPC, CELP, VXC, and VAPC. The computational complexity of these relatively new techniques is often unwieldy, requiring several hundred million floating point operations per second. Reduction of this complexity represents a major goal of the research reported here. This is achieved by developing a new structure with reduced complexity while maintaining excellent perceptual quality.

These relatively new techniques strive for perfect facsimile reproduction and a perceptually important objective noise measure is usually minimized.

1.2 Thesis Objectives

The main objective of this thesis, as indicated previously, is to design and simulate a speech coder operating between 6 and 8 kbps. The real time implementation is *not* a goal of the present work and may not be possible until faster signal processing devices or a suitable VLSI implementation is developed. The speech coder is based on Code-Excited Linear Prediction which, as the name implies, determines the optimum excitation sequence (using analysis by synthesis) which excites a linear filter designed using linear predictive coding techniques.

To develop the coder we require:

- a relatively simple open loop APC system incorporating a pitch predictor and a short term predictor based on the spectral envelope;
- Vector Quantization of the LPC pitch predictor coefficients, gain, and the LPC short term predictor coefficients;
- Vector Quantization of the residual using analysis by synthesis and utilizing a frequency weighted, perceptually important error criterion;
- Complexity reduction techniques for the above items, to enable the coder to be implemented in real time at some future date.

The coder will be implemented on a micro-computer and informal listening tests, as well as signal to noise ratio tests will be used to judge its performance.

A preliminary literature search indicated that few speech coding techniques are available at these low bit rates (between 6 and 8 kbps). A survey of these techniques is presented in Chapter 3. In fact, all applicable techniques available at such low bit rates are essentially the same, and thus, poor performance due to channel errors will have to be compensated for by increased complexity, and hence improved performance in the channel coder. Improved modulation and coding techniques, (such as Trellis Coding) may be required to give better bit error rate performance if the response of the coder to a relatively high bit error rate is poor.

However, during the design phase, source coding techniques which behave relatively poorly in the presence of channel errors will be avoided if possible.

1.3 Thesis Organization

The thesis is organized into four main chapters as well as an introductory chapter, and a concluding chapter.

In Chapter 2, we review briefly the digital communications model and theoretical aspects of data compression and communication. Linear Predictive Coding (LPC) techniques are also reviewed. Chapter 3 delves into several source coding techniques and speech coders; from very low bit rate speech coders (Vocoders) to relatively high bit rate coders (DPCM). Although we are ultimately interested in a speech coder operating between 6 and 8 kbps, we can borrow various techniques which enhance the overall performance from these above mentioned speech coders. In the last section of Chapter 3, we consider further the most promising speech coders for our application and choose a suitable speech coder based on our specific requirements.

In Chapter 4, Vector Quantization (VQ) is reviewed. This technique is used extensively throughout this thesis. Techniques to reduce the complexity of VQ are also investigated. VQ of the LPC model filter is investigated at some length.

Chapter 5 addresses the complexity issue in the original design, and introduces techniques to reduce drastically the overall complexity while maintaining acceptable performance levels. Techniques to encode the various parameters are investigated further. Performance of the various configurations are tested using both objective measures and informal listening tests.

Chapter 6 contains the concluding remarks as well as suggested directions for future research to enhance the performance or to obtain a lower bit rate.

Chapter 2

Introduction To Speech Coding

2.1 Digital Communications Model

The foundations of digital communications were laid out by C.E. Shannon some forty years ago [1] in which he states:

"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point."

Shannon showed that the problem could be divided into two separate, distinct problems as shown in Figure 2.1.

The first problem concerns the mapping from a possibly infinite ensemble of $s(t)$ onto a finite ensemble m_k , and the associated inverse mapping. We may denote the transformation as:

$$m_k = T[s(t)], \quad \text{for } t \in (t_k, t_k + \Delta t) \quad (2.1)$$

and the inverse transformation as:

Figure 2.1: Communications Model

$$\hat{s}(t) = T^{-1}[\hat{m}_k] \quad (2.2)$$

Note that for an error free channel $m_k = \hat{m}_k$, but $s(t) \neq \hat{s}(t)$ in general. It is desirable to minimize the average distortion between $s(t)$ and $\hat{s}(t)$ in the absence of channel errors. That is, we wish to minimize

$$E[d(s(t), T^{-1}[T[s(t)]])] \quad (2.3)$$

where $d(x, y)$ is the distortion between x and y , and $E(\cdot)$ is the expectation operator.

This leads to the rate-distortion function [2], $R(D)$ which is a lower bound on the rate required for a given distortion. Clearly, for a discrete source the minimum rate for no distortion is the source entropy H . Thus,

$$R(0) = H \quad (2.4)$$

The rate-distortion function is a generalization of the concept of entropy. $R(D)$ is a monotonically decreasing function of the distortion, and possibly infinite at zero distortion for a continuous (amplitude or time) source. It can be shown [2] that the rate-distortion function for discrete time continuous amplitude white gaussian noise is an upper bound for all discrete sources. For a given variance, samples from a gaussian pdf are the most random and unpredictable and thus contain the most information.

The problem of data compression/source coding is usually approached from one of two angles, namely:

- minimize the distortion for a given rate;
- minimize the rate for a given distortion.

Thus, the inverse of the rate-distortion function (the distortion-rate function, $D(R)$) is often required. The distortion-rate function for a unit

Figure 2.2: Distortion-Rate Function

variance white gaussian noise process is shown in Figure 2.2. This curve acts as a lower bound for all sources with equal variance.

Referring again to Figure 2.1 we note that the channel encoder maps one of M message waveforms m_k onto one of N transmitted waveforms x_k , where $N \geq M$. In his classic paper, Shannon showed that for properly chosen x_k , bandwidth B , and adequate signal to noise ratio, the error rate approaches zero as N approaches infinity if the data rate R is less than the channel capacity C . However, the decoding delay also approaches infinity as N does.

It is clear that the channel decoder must base its decision on the entire transmitted waveform, and is inherently sub-optimal if it does not. Similarly, the source encoder should observe $s(t)$ over a long period before the decision as to which m_k is to be transmitted. It is interesting to note the similarities between the channel decoder and the source encoder which both map a continuous waveform onto a single discrete waveform (out of a finite ensemble) while trying to minimize a distortion criterion. It was inevitable that the techniques of block coding, trellis coding, and tree coding used in the channel encoder and decoder would find their way into the source encoder and decoder, (vector quantization, tree and trellis coding). This is not entirely surprising since speech, like convolutional codes, can be modeled as a Markov Process.

Although many conceptual similarities exist between the channel decoder and the source encoder, the source encoder *always*¹ benefits from a delayed decision (even for white signals) whereas this is not always the case for a channel decoder (i.e. Phase Shift Keying in the absence of coding). That is, as a larger analysis window (Δt in Equation 2.1) is used, the rate-distortion bound can be approached for certain source coding techniques

¹ Except for a few trivial sources.

(i.e. VQ). A more thorough explanation is given in Section 4, which deals with Vector Quantization (VQ).

The problem of channel coding/decoding has been discussed at some length in the literature, and the performance of coding is now approaching the limits imposed by Information Theory. For speech, it is not entirely clear what the rate-distortion function is, and in fact, not easy to compute. However, the $R(D)$ function for certain Markov Processes are fairly well defined and the performance of source coding techniques can be compared to these lower bounds. The bounds are very valuable, especially when they are approached, since it informs us to repartition our time so as to devote more time to reducing the complexity of our source coder and less time on techniques to reduce the bit rate (or distortion) further.

Thus, if we can model speech by a Markov Process this will give us insight into what limits we can possibly attain, and what source coding techniques are likely to succeed. However, the time varying spectral properties of speech severely complicates this approach.

In the next section, we obtain a speech production model which will aid us in obtaining applicable source coding techniques. That is, techniques which will perform well with the chosen model.

2.2 Speech Production Model

The speech production model proposed for this thesis and in common use in most low and medium bit rate speech coders is shown in Figure 2.3, where

$$B(z) = 1 - P_l(z) = 1 - \sum_{k=-q}^q b_k z^{-(M+k)} \quad (2.5)$$

$$A(z) = 1 - P_s(z) = 1 - \sum_{k=1}^p a_k z^{-k} \quad (2.6)$$

The predictor $P_l(z)$ is called the long term predictor, or the predictor based on the spectral fine structure. $P_s(z)$ is the short term predictor, or the predictor based on the spectral envelope. M is the pitch period, $2q + 1$ is the number of pitch predictor coefficients, $M + q$ is the pitch predictor order, and p is the order of the short term predictor. Note that the long term predictor has a relatively high order ($M + q$), but many coefficients are zero. The filters $1/A(z)$ and $1/B(z)$ are referred to as the short and long term model filters, respectively. $A(z)$ and $B(z)$ are commonly called the inverse short and long term filters respectively.

Since the pitch period M is usually in the range 20–160 samples (2.5–20 msec), the long term model filter can accurately model the spectrally fine structure in the speech spectrum. The short term model filter has a relatively low order ($p \approx 10$ –20), and thus models the speech spectral envelope.

In general, the coefficients a_k and b_k are time varying and are typically updated every 10–25 milli-seconds. We immediately note that the model is an all pole model which considerably simplifies the computation of the predictor parameters, (the a_k s and b_k s).

For zero distortion with respect to the sampled speech, the speech samples can be filtered through the short term ($A(z)$) and long term ($B(z)$) inverse filters to obtain the error signal (residual) e_n . Furthermore, with $Gv_n = e_n$, the reconstructed speech is identical (zero distortion) to the sampled speech ($\hat{s}_n = s_n$).

We term this model exact, since with infinitely fine quantization of the gain, residual, and model filters, exact reproduction of the sampled speech can be accomplished. Compare this with the familiar pitch excited model [6] in which poor speech quality is realized with fine quantization of the parameters.

Figure 2.3: Speech Production Model

Figure 2.4: Speech Analysis/Synthesis

Note that interchanging the order of the short term predictor in the receiver without doing the same in the transmitter is invalid since these filters are, in general, time varying.

A fundamental result of linear prediction analysis is that in choosing $H(z) = A(z)B(z)$ to minimize the average power in e_n is equivalent to a spectral match of $G/|H(e^{j\omega})|^2$ to $|S(e^{j\omega})|^2$. Thus, e_n is relatively white, and in fact approximately gaussian [26]. Since e_n is white, and has a lower dynamic range than the s_n it can be quantized more efficiently than s_n . This intuitive notion can be more aptly proven using results from rate-distortion theory.

For an uncorrelated (and thus independent) gaussian source, x , with variance σ_x^2 , and using a mean squared error distortion measure (mse), we have [2]

$$D_G(R) = 2^{-2R} \sigma_x^2 \quad (2.7)$$

Thus, for signals with smaller dynamic range, fewer bits are required for a given distortion. For a correlated (gaussian) source with variance σ_x^2

$$D_G(R) = 2^{-2R} \sigma_x^2 \gamma_x^2 \quad (2.8)$$

where γ_x^2 is the spectral flatness measure given by

$$\gamma_x^2 = \frac{\exp(\frac{1}{2\pi} \int_{-\pi}^{\pi} \ln(\Phi_{xx}(\omega)) d\omega)}{\frac{1}{2\pi} \int_{-\pi}^{\pi} \Phi_{xx}(\omega) d\omega} \quad (2.9)$$

where $\Phi_{xx}(\omega)$ is the Z-transform of $\phi_{xx}(k)$ ($= E[x_n \dot{x}_{n+k}]$) evaluated on the unit circle [4].

Since $0 \leq \gamma_x^2 \leq 1$, the gain (in dB) in coding a correlated source over an uncorrelated source (with the same variance, and at the same rate) is $-10 \log(\gamma_x^2)$. It must be remembered, however, that this gain is due to exploiting the memory of the source, and is the maximum gain which can be realized.

Vector quantization can be shown to be optimum, in terms of approaching the rate-distortion bound as the vector dimension approaches infinity. However, it is more popular to use linear prediction to whiten the residual, and to quantize the residual signal in a relatively small number of bits. This is effectively a product code; seemingly independent parameters (gain, pitch, spectral model, and residual) are quantized independently. Although product codes are inherently suboptimal, far less complex systems can be realized than with direct vector quantization of the input speech waveform. The input speech samples are typically organized into blocks of samples (dimension N) and processed independently. The filter parameters are typically updated for each new vector, or block, of samples.

The method in which the residual signal e_n is quantized into the excitation signal v_n and the rate and manner in which the filter parameters are updated is dependent on the type of speech coder employed.

In an APC Coder the residual signal is simply quantized by a scalar, possibly adaptive, possibly nonuniform quantizer. Noise feedback to achieve noise spectral shaping may also be utilized.

In MPLPC (Multipulse Linear Predictive Coding) the residual is quantized into a signal with a minimal number of pulses in an analysis window (typically 8–10 nonzero pulses in a 10 msec frame). CELP attempts to find the optimum v_n from a large codebook of gaussian numbers, (typically 1024, 40 dimensional vectors). At an 8 kHz sampling rate this implies an analysis window of 5 msec for finding the optimum excitation sequence.

The pitch period M , is typically in the range 2.5–20 msec (or 20–160 samples). Thus, with CELP the pitch filter can be replaced (approximately) by the zero input response of the pitch filter, and the memory updated after each analysis frame, since the pitch period is typically larger than an analysis window.

Therefore, we arrive at the simplified synthesis structure shown in Figure 2.5. Thus, a vector of samples $v_n = (v_n(0), v_n(1), \dots, v_n(K-1))$ excites the short term linear predictor with its memory set to zero at the start of the analysis frame. At the end of the analysis frame the filter memory in the two filters is updated by passing v_n through the filter. Note that this model is equivalent to that in Figure 2.4 if the pitch period (in samples) is greater than $K + q$ where K is the vector dimension and $2q + 1$ is the number of pitch predictor taps. Thus, by forcing the pitch period into this range, we can maintain an exact model. The techniques we utilize to calculate the pitch filter parameters will allow us to perform this constrained search for the pitch period. Utilizing this model of speech production will allow us to make dramatic simplifications to the CELP structure as will be shown in Chapter 5.

Furthermore, we note that the model (Figure 2.3) is very general and encompasses the pitch excited model, the multipulse model, and other models as well. Also, unlike the pitch excited model, it is an exact model whereby with the unquantized parameters (short and long term model filters, gain, and residual) we can exactly synthesize the sampled speech. The pitch excited model, with its voiced/unvoiced classification and simple excitation model, is not an exact model since the original sampled speech can not be perfectly reconstructed given the unquantized parameters. Thus, even without quantization of the parameters, poor speech quality can result.

As a closing note, we emphasize that the coders we will study use one of the above models (Figure 2.5, or Figure 2.4) and only differ in the determination and quantization of the residual. These codecs are generically referred to as *Residually Excited Coders*, and are inherently suboptimal due to their product code nature.

Figure 2.5: Simplified Synthesis Structure

2.3 Evaluating Speech Coders

It is desirable to have an objective measure of signal to noise ratio which corresponds closely with the subjective quality of the reconstructed speech. That is, we require an objective measure which correlates well with the hearing mechanism of the human ear.

Clearly, a perceptual or subjective measure is most important in judging a speech coding algorithm but typically very time consuming. Objective measures are usually easy to compute but not always subjectively meaningful.

The most common objective measurement is signal to quantization noise ratio, (in dB) given by

$$\text{SNR} = 10 \log(\sigma_s^2 / \sigma_n^2) \quad (2.10)$$

where $\sigma_s^2 = E[s_n^2]$ and $\sigma_n^2 = E[(\hat{s}_n - s_n)^2]$.

The main problem with this measure is that speech segments in which the coder performs well (small quantization noise) tend to outweigh the segments in which the coder has relatively poor performance. Thus, an often better performance measure is the segmental signal to noise ratio defined by

$$\text{SNR}_{\text{SEG}} = E[\text{SNR}(m)] \quad (2.11)$$

where $\text{SNR}(m)$ is the above defined signal to noise ratio (in dB) over m samples. Usually, m corresponds to between 10 and 20 msec.

Objective measures attempt to quantify the performance of a speech coder in terms of a single number which relates to the subjective performance. The ultimate performance, however, depends on how the human auditory system perceives the quantization noise. In any speech coder

design, both informal listening tests and objective performance measures should be utilized in the development process.

A particularly suitable objective measure for this study is the noise weighted signal to quantization noise ratio defined by

$$\text{NWSNR} = 10 \log(\sigma_s^2 / \sigma_r^2) \quad (2.12)$$

where σ_r^2 is the variance of the spectrally weighted quantization noise. The weighting filter is given by

$$W(z) = \frac{A(z)}{A(z/\gamma)} \quad (2.13)$$

where $A(z)$ is the short term inverse filter and $\gamma \approx 0.73$. The weighting filter is discussed in greater detail in Chapters 3 and 5.

One of the most common subjective measures is the mean opinion score (MOS) in which test subjects rate the coder performance on a scale from 1 to 5. Tests are usually performed over a wide range of (male and female) speakers, test phrases, and over a wide range of listeners to obtain statistically meaningful results. Another common measure, often used for testing Vocoders, is the Diagnostic Rhyme Test (DRT) which is a word intelligibility score. For more information on subjective and objective performance measures see [5, Appendices E and F].

In most speech coding applications, we classify the speech coders (with respect to their bit rates) as shown in Table 2.1.

In this study, therefore, we are interested in a low rate codec (6–8 kbps).

2.4 Linear Prediction of Speech

2.4.1 Basic Formulation and Solution

As discussed in Section 2.2, the inverse filter $A(z)$ is chosen to minimize the mean squared prediction error, (or minimize the energy in the residual

Terminology	Bit rate (kbps)
high rate/wideband	> 16
medium rate/ band	8-16
low rate/narrow band	1-8
very low rate	< 1

Table 2.1: Classification of Speech Codecs

signal). Recall

$$\begin{aligned} A(z) &= 1 - P_s(z) \\ &= 1 - \sum_{k=1}^p a_k z^{-k} \end{aligned} \quad (2.14)$$

and

$$\begin{aligned} B(z) &= 1 - P_l(z) \\ &= 1 - \sum_{k=-q}^q b_k z^{-(M+k)} \end{aligned} \quad (2.15)$$

The long term predictor ($P_l(z)$) tries to minimize the mean squared prediction error after pitch prediction, (given the pitch period M). From Figure 2.4, we have

$$D(z) = S(z)A(z) \quad (2.16)$$

Therefore

$$d_n = s_n - \sum_{k=1}^p a_k s_{n-k} \quad (2.17)$$

Now, we wish to minimize d_n^2 over n samples, where D_p is the residual energy of the p^{th} order optimum predictor

$$\begin{aligned} D_p &= \sum_n d_n^2 \\ &= \sum_n \left(s_n - \sum_{k=1}^p a_k s_{n-k} \right)^2 \\ &= \sum_n s_n^2 - 2 \sum_n s_n \sum_{k=1}^p a_k s_{n-k} + \sum_n \sum_{k=1}^p a_k s_{n-k} \sum_{j=1}^p a_j s_{n-j} \end{aligned} \quad (2.18)$$

Minimizing with respect to the a_i for $1 \leq i \leq p$ yields

$$\begin{aligned} \frac{\partial D_p}{\partial a_i} &= -2 \sum_n s_n s_{n-i} + 2 \sum_n \sum_{k=1}^p a_k s_{n-k} s_{n-i} \\ &= 0 \end{aligned}$$

Therefore

$$\sum_n s_n s_{n-i} = \sum_{k=1}^p a_k \sum_n s_{n-k} s_{n-i} \quad 1 \leq i \leq p \quad (2.19)$$

Combining Equations 2.18 and 2.19 the minimum residual energy is

$$\begin{aligned} D_p &= \sum_n s_n^2 - \sum_n s_n \sum_{k=1}^p a_k s_{n-k} \\ &= \sum_n s_n^2 - \sum_{k=1}^p a_k \sum_n s_n s_{n-k} \end{aligned} \quad (2.20)$$

Since speech is only locally stationary over approximately 10–25 msec, the range of summation (n) must reflect this property. There are two primary methods of analysis, the Autocorrelation Method and the Covariance Method. With the first method of analysis, (the Autocorrelation Method) a time domain window is applied to the speech signal such that

$$\tilde{s}_n = 0, \quad \text{for } N_p \leq n < 0$$

where $\tilde{s}_n$ is the windowed speech signal.

Therefore, we define

$$\begin{aligned} r(i) &= \sum_n \tilde{s}_n \tilde{s}_{n-i} \\ &= \sum_{n=0}^{N_p-1-|i|} \tilde{s}_n \tilde{s}_{n+|i|} \end{aligned}$$

Rewriting Equation 2.19 we get

$$\sum_{k=1}^p a_k r(i-k) = r(i) \quad 1 \leq i \leq p$$

or in matrix notation

$$\mathbf{R}\mathbf{a} = \mathbf{r} \quad (2.21)$$

where

$$\begin{aligned} R_{ij} &= r(i-j) \\ \mathbf{r}^T &= (r(1), r(2), \dots, r(p)) \end{aligned}$$

and

$$\mathbf{a}^T = (a(1), a(2), \dots, a(p))$$

Thus, the residual energy² is given by (using Equations 2.18, 2.20 and 2.21)

$$D_p = r(0) - 2\mathbf{a}^T \mathbf{r} + \mathbf{a}^T \mathbf{R} \mathbf{a} \quad (2.22)$$

$$= r(0) - \mathbf{a}^T \mathbf{r} \quad (2.23)$$

$$= r(0) - \mathbf{a}^T \mathbf{R} \mathbf{a} \quad (2.24)$$

Note that Equation 2.22 yields the residual energy for any (optimum or sub-optimum) predictor $\mathbf{a}$, and Equations 2.23 and 2.24 apply to the optimum predictor only.

The solution for the predictor coefficients (the solution of the matrix equation) can be efficiently accomplished using the Levinson-Durbin algorithm, (see Appendix A). With the Autocorrelation method, the speech samples are broken up into N_p dimensional vectors, windowed, and the predictor coefficients are computed using the above algorithm.

If we utilize p past values and N new speech samples, we can write

$$\begin{aligned} \phi_{ik} &= \sum_n s_{n-k} s_{n-i} \\ &= \sum_{n=-k}^{N-k-1} s_n s_{n+k-i} \end{aligned} \quad (2.25)$$

and by rewriting Equation 2.19 we get

²Note that this is not the actual residual energy since the speech is weighted with a window before computing the autocorrelation coefficients, and overlap may be used, (see page 26).

$$\sum_{k=1}^p a_k \phi_{ik} = \phi_{i0} \quad 1 \leq i \leq p$$

or

$$\Phi \mathbf{a} = \boldsymbol{\phi} \quad (2.26)$$

The minimum residual energy, from Equation 2.18, is just

$$\begin{aligned} D_p &= \phi_{00} - 2\boldsymbol{\phi}^T \mathbf{a} + \mathbf{a}^T \Phi \mathbf{a} \\ &= \phi_{00} - \boldsymbol{\phi}^T \mathbf{a} \end{aligned}$$

Equation 2.26 can be efficiently computed using Cholesky Decomposition (Appendix B). This above method is called the Covariance Solution.

Although the names (Autocorrelation and Covariance) are somewhat inappropriate in the manner in which they are used, these two names have remained popular for historical reasons.

It can be shown that the Autocorrelation Method always results in a stable solution for the predictor coefficients whereas the covariance solution has no such guarantee. However, a modification to the covariance solution which can guarantee the stability of the resulting model filter has been developed [25] and is also discussed in Appendix B.

Overlap is often utilized to obtain a better estimate of the autocorrelation vector ($\mathbf{r}$). For example, an analysis window of N_p samples may be used and only N new samples ($N \leq N_p$) are shifted into the buffer before a new model filter is computed. A long buffer length is desirable to obtain a good spectral estimate, but a short buffer length is required due to the time varying spectral properties of speech. Now, the spectral estimate is of the N central samples in the buffer (those samples starting at location $(N_p - N)/2$). Thus, if we wished to compute the residual, we would filter the unwindowed central N samples in the buffer through the filter $A(z)$. For these reasons, the residual energy as given by Equation 2.24 is not

equal to the residual energy of the rectangularly windowed speech filtered through $A(z)$ due to the (non-rectangular) windowing and the overlap. The percentage overlap is just $100(N_p - N)/N$.

A representative choice of parameters (for the Autocorrelation or Covariance solution) is

$$N_p = 160 \quad (20 \text{ msec at a } 8 \text{ kHz sampling rate})$$

$$N = 80 \quad (10 \text{ msec at a } 8 \text{ kHz sampling rate})$$

$$p = 20$$

As an example we show the spectra of the model (synthesis) filter ($1/A(z)$) and the input speech spectrum for a voiced segment of speech, (Figure 2.6). The speech spectra were obtained by applying a Hamming Window [4] to the 160 sample buffer and computing a 1024 point FFT (fast fourier transform) on the result. The Autocorrelation method was used (again a Hamming Window was used to window the input speech) to obtain the predictor parameters. The model filter magnitude response was again computed using a 1024 point FFT. The spectral match is quite good and tends to fit the peaks of the spectra very well [6]. The sharp nulls in the spectra are due to the pitch information. In fact

$$M = \frac{1}{\Delta f} f,$$

where Δf is the average distance between successive nulls.

The original speech segment, and the residual after short term prediction (filtering through $A(z)$) is shown in Figures 2.7 and 2.8. The pitch information is clearly visible.

It has been mentioned above that linear prediction analysis is equivalent to a spectral matching formulation [6]. However, due to the sharp cutoff of the antialiasing and reconstruction filters (at frequencies above approximately 3.3 kHz) the information contained in this band is not important. That is, the linear prediction filter should not "work too hard" to maintain a

Figure 2.6: Speech Spectra/ Model Filter Response

Figure 2.7: Speech Segment

Figure 2.8: Speech Residual

spectral match in this frequency band. Furthermore, the large attenuation due to the antialiasing filter can lead to artificially high power gains (sum of the squares of the predictor coefficients) and possible instabilities in the predictor. Ideally, we could use a brick wall filter, with a cutoff of 4 kHz for a sampling frequency of 8 kHz. Since the existence of so called "brick wall" filters is highly suspect, a different method must be devised to "fill in" the missing frequencies. To fill in the missing frequencies one could simply pass the sampled speech through a filter which was the digital equivalent of the inverse of the analog antialiasing filter. Alternatively, one could pass white gaussian noise through the inverse of the antialiasing filter and add this to the sampled speech signal. On a dB scale this is almost equivalent to adding white gaussian noise directly to the speech signal, which in turn is equivalent to adding a small number to the diagonal of the Autocorrelation (or Covariance) matrix. The variance of the white noise is proportional to the prediction gain in the absence of the correction.

The details of this *high frequency correction* are discussed in [25,26].

2.4.2 The Itakura-Saito Distortion Measure

It must be emphasized that the model filters ($1/A(z)$ and $1/B(z)$) must be quantized before transmission to the receiver. Thus, in order to obtain a suitable model filter we usually proceed through a two step process as shown in Figure 2.9.

The Z -transform of the input speech spectrum is $S(z)$ ³, the optimum model filter (order p) is $\sqrt{\alpha_p}/A_p(z)$, the optimum quantized model filter with the gain unquantized is $\sqrt{\alpha}/A(z)$, and the optimum quantized model filter (with the gain also quantized) is $\sigma/A(z)$. The optimality criterion

³Note that a window is applied to the speech signal. Thus, strictly speaking, this is $\tilde{S}(z)$.

Figure 2.9: Identification-Quantization

depends on the distortion measure chosen.

The Itakura-Saito distortion measure (IS-distortion), or the Gain Optimized Itakura-Saito distortion (Itakura, or I-distortion) provides a tractable means to calculate the quantized model filter [42,43,15]. We will observe that the I-distortion is a useful distortion measure to use when designing a Vector Quantizer for the model filter. We will also discover that the IS-distortion for the analysis stage is the same as the Autocorrelation Method. Thus, we are using the same distortion measure to quantize our model filter as we did to obtain our optimum model filter (identification). Thus, the IS-distortion offers mathematical tractability, a sensible method to quantize the filter parameters, and is a subjectively meaningful distortion measure. Furthermore, the IS-distortion obeys a form of the triangle inequality; nothing is lost in splitting up the search for a model filter into a two step process.

The Z -transform of the residual signal after inverse filtering (with $A(z)$) is just

$$D(z) = Z[d_n]$$

and the residual energy is (from Parseval's Relations)

$$\sum d_n^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} D(e^{j\omega}) D^*(e^{j\omega}) d\omega \quad (2.27)$$

The gain α is defined by

$$\alpha = \frac{1}{2\pi} \int_{-\pi}^{\pi} |S(e^{j\omega})|^2 |A(e^{j\omega})|^2 d\omega \quad (2.28)$$

which is just the residual energy for the model filter $A(z)$. Recall that σ is the quantized version of $\sqrt{\alpha}$, and α_p is the gain for the optimum model filter (of order p). Thus

$$\alpha_p = \frac{1}{2\pi} \int_{-\pi}^{\pi} |S(e^{j\omega})|^2 |A_p(e^{j\omega})|^2 d\omega \quad (2.29)$$

Furthermore, in the limit as $p \rightarrow \infty$, we obtain

$$\begin{aligned}\alpha_\infty &= \lim_{p \rightarrow \infty} \alpha_p \\ &= \exp \left[\frac{1}{2\pi} \int_{-\pi}^{\pi} \ln(|S(e^{j\omega})|^2) d\omega \right]\end{aligned}\quad (2.30)$$

The term α_∞ is the lower limit on the prediction gain, and is sometimes called the one-step prediction error, or the gain of $|S(e^{j\omega})|^2$. Note that

$$|S(e^{j\omega})|^2 = \lim_{p \rightarrow \infty} \frac{\alpha_p}{|A_p(e^{j\omega})|^2} \quad (2.31)$$

since with an infinitely long predictor we can exactly match the input speech spectrum, and that [6, Section 2.5], (providing $A(z)$ is stable)

$$\int_{-\pi}^{\pi} \ln(|A(e^{j\omega})|) d\omega = 0 \quad (2.32)$$

Furthermore

$$\begin{aligned}\ln(\alpha_\infty) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln(|S(e^{j\omega})|^2 |A_\infty(e^{j\omega})|^2) d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln(|S(e^{j\omega})|^2) d\omega\end{aligned}\quad (2.33)$$

which is equivalent to Equation 2.30. Equation 2.32 implies that the log spectrum of the model filter is zero mean, (providing $A(z)$ is stable).

Finally, now that the preliminary definitions are completed, we can define the Itakura-Saito distortion measure which will enable us to compute the optimum quantized model filter (in terms of the IS-distortion) shown in Figure 2.9. The measure is denoted by d_{IS} where

$$d_{IS}(|S|^2, |\sigma/A|^2) = \frac{1}{2\pi} \int_{-\pi}^{\pi} [|\sigma A/\sigma|^2 - \ln(|\sigma A/\sigma|^2) - 1] d\omega \quad (2.34)$$

and where $|S|^2$ and $|\sigma/A|^2$ are used to denote the speech energy spectral density ($|S(e^{j\omega})|^2$) and the model filter magnitude response ($|\sigma/A(e^{j\omega})|^2$) respectively.

Utilizing Equations 2.28, 2.33 and 2.34 we can write

$$d_{IS}(|S|^2, |\sigma/A|^2) = \alpha/\sigma^2 - \ln(\alpha_\infty/\sigma^2) - 1 \quad (2.35)$$

The IS-distortion has a wealth of useful properties. Minimization of Equation 2.35 in the identification step (Figure 2.9) is equivalent to minimizing α which in turn leads to the familiar Autocorrelation Method. Secondly

$$d_{IS}(|S|^2, |\sigma/A|^2) = d_{IS}(|S|^2, |\sqrt{\alpha_p}/A_p|^2) + d_{IS}(|\sqrt{\alpha_p}/A_p|^2, |\sigma/A|^2) \quad (2.36)$$

Thus, the process of splitting the compression into a two step process does not entail a loss in optimality (in terms of the IS-distortion). That is, minimization of Equation 2.36 can be accomplished by minimizing

$$d_{IS}(|S|^2, |\sqrt{\alpha_p}/A_p|^2) \quad (\text{identification})$$

followed by minimization of

$$d_{IS}(|\sqrt{\alpha_p}/A_p|^2, |\sigma/A|^2) \quad (\text{quantization})$$

The identification stage minimizes

$$d_{IS}(|S|^2, |\sqrt{\alpha_p}/A_p|^2) = \ln(\alpha_p/\alpha_\infty) \quad (2.37)$$

The term α_∞ depends only on the speech frame and not on the model filter. Thus, minimizing Equation 2.37 is equivalent to minimizing

$$\begin{aligned} \alpha_p &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |S|^2 |A|^2 d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} S S^* \left[1 - \sum_{k=1}^p a_k e^{j\omega k} \right] \left[1 - \sum_{i=1}^p a_i e^{-j\omega i} \right] d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} S S^* d\omega - \frac{1}{2\pi} \sum_{i=1}^p a_i \int_{-\pi}^{\pi} S S^* e^{-j\omega i} d\omega \\ &\quad - \frac{1}{2\pi} \sum_{k=1}^p a_k \int_{-\pi}^{\pi} S S^* e^{j\omega k} d\omega \end{aligned}$$

$$\begin{aligned}
& + \frac{1}{2\pi} \sum_{i=1}^p \sum_{k=1}^p a_i a_k \int_{-\pi}^{\pi} S S^* e^{-j\omega(i-k)} d\omega \\
& = r(0) - 2 \sum_{k=1}^p a_k r(k) + \sum_{k=1}^p \sum_{i=1}^p a_k a_i r(i-k) \quad (2.38)
\end{aligned}$$

$$= r(0) - 2\mathbf{a}^T \mathbf{r} + \mathbf{a}^T \mathbf{R} \mathbf{a} \quad (2.39)$$

which is identical to Equation 2.22. Therefore, minimizing Equation 2.39 is equivalent to the Autocorrelation Method, ([15]).

The quantization step minimizes

$$d_{IS}(|\sqrt{\alpha_p}/A_p|^2, |\sigma/A|^2) = \alpha/\sigma^2 - \ln(\alpha_p/\sigma^2) - 1 \quad (2.40)$$

Combining Equations 2.35, 2.37, and 2.40 results in Equation 2.36. Note that

$$\begin{aligned}
\alpha &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |S|^2 |A|^2 d\omega \\
&= \frac{1}{2\pi} \int_{-\pi}^{\pi} |\sqrt{\alpha_p}/A_p|^2 |A|^2 d\omega \quad (2.41)
\end{aligned}$$

from the correlation matching property of the Autocorrelation Method [6, Chapter 2]. That is, the autocorrelation of the impulse response of the optimum (unquantized) model filter ($r_p(k)$) matches the signal autocorrelation at $2p+1$ points

$$r_p(k) = r(k) \quad \text{for } 0 \leq |k| \leq p$$

For purposes of computation, Equation 2.35 can be rearranged to give

$$d_{IS}(|S|^2, |\sigma/A|^2) + \ln(\alpha_\infty) + 1 = \alpha/\sigma^2 + \ln(\sigma^2) \quad (2.42)$$

Minimization of Equation 2.35 is equivalent to minimizing Equation 2.42 since α_∞ depends only on the speech frame, and not on the model filter.

The Gain Optimized Itakura-Saito Distortion— or simply the Itakura Distortion (I-distortion) measure is defined by

$$\begin{aligned} d_I(|S|^2, |A|^2) &= d_{IS}(|S|^2, |\sqrt{\alpha}/A|^2) \\ &= \ln(\alpha/\alpha_\infty) \end{aligned} \quad (2.43)$$

It can easily be shown that

$$d_{IS}(|S|^2, |\sigma/A|^2) = d_{IS}(|S|^2, |\sqrt{\alpha}/A|^2) + d_{IS}(\alpha, \sigma^2) \quad (2.44)$$

This demonstrates the important *Gain Separation* wherein the best model filter is first chosen to minimize the gain optimized IS-distortion and the gain is then quantized by minimizing $d_{IS}(\alpha, \sigma^2)$, where

$$d_{IS}(\alpha, \sigma^2) = \alpha/\sigma^2 - \ln(\alpha/\sigma^2) - 1 \quad (2.45)$$

A useful approximation to the IS-distortion is arrived at by noting that for small distortions

$$\begin{aligned} |SA/\sigma|^2 &= u = e^{\ln(u)} \\ &= 1 + \ln(u) + \frac{1}{2}[\ln(u)]^2 + \dots \end{aligned}$$

Thus

$$d_{IS}(|S|^2, |\sigma/A|^2) \approx \frac{1}{4\pi} \int_{-\pi}^{\pi} [\ln(|S|^2) - \ln(|\sigma/A|^2)]^2 d\omega \quad (2.46)$$

This is just one half the mean square log spectral error. Thus, for small distortions, the IS-distortion is approximately a spectral matching technique, (using a mse-log distortion). This is due to the fact that the IS-distortion is extremely closely related to the Autocorrelation Method which can also be thought of as a spectral matching technique.

A final, and probably the most useful property of the gain optimized IS-distortion is that Equation 2.43 can be minimized by minimizing α since α_∞

is a constant which depends only on the speech frame. From Equations 2.39 and 2.22, we may write (with $a_0 = -1$)

$$\begin{aligned}
 \alpha &= r(0) - 2\mathbf{a}^T \mathbf{r} + \mathbf{a}^T \mathbf{R} \mathbf{a} \\
 &= \sum_{j=0}^p \sum_{k=0}^p a_j a_k r(k) \\
 &= \sum_{k=-p}^p r(k) r_a(k) \\
 &= r(0) r_a(0) + 2 \sum_{k=1}^p r(k) r_a(k)
 \end{aligned} \tag{2.47}$$

where

$$r_a(k) = \sum_{j=0}^{p-|k|} a_j a_{j+|k|} \quad \text{with } a_0 = -1$$

Furthermore, if we have a codebook (determined somehow), of representative a_k s, (and thus $r_a(k)$ s), we can search through the codebook to find the codevector which minimizes Equation 2.47. Thus, we have reduced the identification *and* quantization steps into a single process (forgetting, momentarily, about the gain term), which is optimum in terms of choosing the minimum I-distortion model in the codebook. The gain term can also be chosen from a codebook utilizing the distortion in Equation 2.45. Note that even though the best model (and gain) is chosen from the codebooks, this two step selection is not optimum for a given rate constraint since the codebook is forced into a certain form, [15,42].

For the IS-distortion, one may minimize Equation 2.42, where α is computed as above. In this case we jointly encode the model filter and the gain term.

We have momentarily sidestepped the obviously important codebook design process. This is the topic of Chapter 4 entitled Vector Quantization.

2.4.3 The Inverse Filter Matching Principle

Suppose we obtain, somehow, a large codebook of representative model filters

$$A^{(k)}(z) \quad \text{for } 0 \leq k < L$$

To determine the optimum model filter out of the codebook based on the Inverse Filter Matching (IFM) principle we simply filter the input speech vector through each model filter, and choose the filter which results in minimum residual energy, [24].

Suppose we have p past speech values (the memory inherent in the p^{th} order predictors), and N new values. The filtered speech, through model filter $A^{(k)}(z)$ is just

$$\begin{aligned} d_n &= s_n - \sum_{j=1}^p a_j^{(k)} s_{n-j} \\ &= - \sum_{j=0}^p a_j^{(k)} s_{n-j} \quad \text{where } a_0 = -1 \end{aligned}$$

The residual energy (over N samples) is

$$\begin{aligned} D^{(k)} &= \sum_{n=0}^{N-1} d_n^2 \\ &= \sum_{n=0}^{N-1} \sum_{i=0}^p \sum_{j=0}^p a_i^{(k)} a_j^{(k)} s_{n-i} s_{n-j} \\ &= \sum_{i=0}^p \sum_{j=0}^p a_i^{(k)} a_j^{(k)} \sum_{n=0}^{N-1} s_{n-i} s_{n-j} \end{aligned}$$

Thus, the residual energy for the k^{th} codevector is

$$D^{(k)} = \sum_{i=0}^p \sum_{j=0}^p a_i^{(k)} a_j^{(k)} \phi_{ij} \quad (2.48)$$

$$= \phi_{00} - 2\phi^T a^{(k)} + a^{T(k)} \Phi a^{(k)} \quad (2.49)$$

where ϕ_{ij} are the elements of the covariance matrix in the Covariance Solution, (see Section 2.4.1, Equation 2.25), and $\mathbf{a}^{(k)}$ is the k^{th} vector of predictor parameters from the codebook where

$$\mathbf{a}^{T(k)} = (a_1^{(k)}, a_2^{(k)}, \dots, p^{(k)})$$

The computational complexity of Equation 2.48 is quite costly since it must be computed over the whole codebook. Given a codebook size L , predictor order p , window size N , and a sampling rate f_s , the computational complexity C in FLOPS (floating point operations per second) is approximately

$$C \approx L f_s p^2 / N \quad \text{FLOPS}$$

Note that this does not include the computation of the covariance matrix. For typical parameters ($L = 2^{10}$, $p = 20$, $f_s = 8$ kHz, and $N = 80$), we get

$$\begin{aligned} C &\approx 2^{10} \cdot 20^2 \cdot 8000 / 80 \quad \text{FLOPS} \\ &\approx 40 \text{ MFLOPS} \end{aligned}$$

This is approximately a factor of p more complex than the distortion computation based on the I-distortion, although a slightly lower complexity can be realized by exploiting the symmetry properties of the matrix Φ . Note that the I-distortion and IFM principle are closely related through Equations 2.48 and 2.38, since both attempt to minimize a (slightly different) residual energy.

The IFM principle is interesting in that it is the VQ equivalent of the covariance solution. Furthermore, it is conceptually very simple since Cholesky Decomposition is not required. For a hardware solution, the IFM principle has many attractive properties due to its simplicity.

We noted previously that the IS-distortion is intimately related to the Autocorrelation Method. The Covariance Solution and the IFM principle

are closely related as well. We have thus gained a much better intuitive notion about these seemingly different techniques.

2.4.4 Alternate Representations of the Model Filter

The most familiar and conceptually simple parameterization of the LPC model filter is the predictor coefficients. We have

$$A(z) = 1 - \sum_{k=1}^p a_k z^{-k} \quad (2.50)$$

The predictor coefficients are the a_k s. the predictor coefficient vector is

$$\mathbf{a}^T = (a_1, a_2, \dots, a_p) \quad (2.51)$$

An alternate representation discussed in Appendix A are the reflection coefficients. A one-to-one correspondence exists between the reflection coefficients and the predictor coefficients. The reflection coefficients offer a simple test for stability and have better quantization properties than the predictor coefficients. For example, although the Autocorrelation Method is guaranteed to produce a stable filter, under quantization, some of the poles may move outside the unit circle in the Z -plane. The reflection coefficient vector is

$$\mathbf{k}^T = (k_1, k_2, k_3, \dots, k_p) \quad (2.52)$$

The resulting filter is stable if and only if

$$|k_i| < 1 \quad \text{for } 1 \leq i \leq p$$

The Log-Area-Ratios (LARs) are defined by

$$\begin{aligned} c_i &= \frac{1}{2} \ln \left(\frac{1+k_i}{1-k_i} \right) \\ &= \tanh^{-1}(k_i) \end{aligned} \quad (2.53)$$

Uniform quantization of the LARs using a mse distortion measure corresponds (approximately) to minimum spectral error. Usually, the LARs are quantized using a fixed, (or possibly adaptive) bit assignment or quantized jointly using a vector quantizer, and a mean squared error distortion measure. Quantization and properties of the reflection coefficients and LARs are discussed at length in [5,6] and the references contained therein.

Yet another parameter set which offers improved quantization properties over the LARs and predictor coefficients are the Line-Spectral-Frequencies, or Line-Spectrum-Pairs (LSFs or LSPs), [52,53,54,55].

With the LSP formulation a pair of polynomials is used to represent the LPC (polynomial) filter. In particular, with $A(z)$ given by Equation 2.50, the two LSP polynomials are

$$F_1(z) = A(z) + z^{-(p+1)}A(z^{-1}) \quad (2.54)$$

$$F_2(z) = A(z) - z^{-(p+1)}A(z^{-1}) \quad (2.55)$$

Important properties of the LSP polynomials are:

1. All zeros of $F_1(z)$ and $F_2(z)$ are on the unit circle in the Z -plane;
2. The zeros of $F_1(z)$ and $F_2(z)$ are interlaced;
3. Stability is ensured under quantization of the zeros of $F_1(z)$ and $F_2(z)$ as long as properties 1 and 2 are adhered to. That is, any ordered set of LSFs can be used to construct a stable LPC filter.

Since the zeros (z_i) are on the unit circle they can be expressed as $e^{j\omega_i}$. Since the predictor coefficients are real, if z_i is a complex zero of F_1 or F_2 then so is z_i^* . Furthermore, $F_1(z)$ and $F_2(z)$ are symmetric and antisymmetric respectively. That is, given

$$F_1(z) = f_{10} + f_{11}z^{-1} + \dots + f_{1(p+1)}z^{-(p+1)}$$

$$F_2(z) = f_{20} + f_{21}z^{-1} + \cdots + f_{2(p+1)}z^{-(p+1)}$$

then

$$f_{1i} = f_{1(p+1-i)}, \quad f_{2i} = -f_{2(p+1-i)}$$

The properties of symmetric and antisymmetric polynomials can be exploited to show that for p even

$$F_1(z) = (1 + z^{-1})G_1(z), \quad F_2(z) = (1 - z^{-1})G_2(z)$$

and for p odd

$$F_1(z) = G_1(z), \quad F_2(z) = (1 - z^{-2})G_2(z)$$

Thus, since there are $2p$ total zeros of G_1 and G_2 , and for each zero we have a corresponding complex conjugate zero, we may parameterize the LSP polynomial in a p coefficient vector

$$\omega = (\omega_1, \omega_2, \dots, \omega_p)$$

where ω_i are the angular positions of the zeros of G_1 or G_2 in the upper half Z -plane. The ω_i are called the Line-Spectral-Frequencies (LSFs). The zeros are given by

$$z_i = \exp(j\omega_i) \quad 0 < \omega_i < \pi$$

Note that since $+1$ is a zero of $F_2(z)$ for even or odd p $\exp(j\omega_1)$ is always a root of $F_1(z)$ due to the interlacing property. Furthermore

ω_i is an LSF of G_1 iff i is odd

ω_i is an LSF of G_2 iff i is even

The numerical search for the zeros of G_1 and G_2 can be enhanced using the interlacing property and the unit circle property, [55,52]. In [52], the

discrete cosine transform was used to calculate the zeros of $G_1(z)$ and $G_2(z)$. In [55], Chebyshev polynomials were utilized to compute the zeros of G_1 and G_2 ; thus eliminating the requirement of a trigonometric evaluation.

Although the numerical search for the LSFs is an interesting problem, it is beyond the scope of this report. The interested reader is referred to the above cited references.

It was shown in [54], that quantization of the LSPs was approximately 25% more efficient than quantization of the LARs. They determined, based on a DRT score, (see Section 2.3), that the same level of intelligibility was realized with 33 bit quantization of the LSPs as compared to 41 bit quantization of the LARs. Their research was carried out based on a 2400 bps vocoder, which is not in line with our research. They also performed 12 bit Vector Quantization of the LPC model filter based on LSFs. The performance (based on the DRT) was approximately 1.5 points lower than 41 bit LAR scalar quantization.

Based on the literature survey we appear to have a number of possibilities to quantize the LPC model filter. We choose to utilize the techniques of vector quantization (VQ) due to its superior performance over scalar quantization. The techniques we will consider include VQ based on:

- the I-distortion;
- the IS-distortion;
- the LSFs;
- the LARs;
- Inverse Filter Matching.

Furthermore, it appears that 10 bit quantization is inadequate to represent the LPC model filter. Thus, a greater bit rate may be required to

represent the spectrum sufficiently, [54].

In Section 4.8 we investigate the quantization properties of the model filter and choose a suitable quantization method.

Chapter 3

Speech Coding Algorithms

3.1 Medium to High Bit Rate Speech Coders

3.1.1 Differential Pulse Code Modulation

Differential PCM systems typically operating in the high bit rate region (> 16 kbps) exploit long term speech statistics utilizing a fixed, low order predictor. Noise feedback coding (NFC), or a closed loop predictor, are typically used to obtain a desired noise spectrum.

A simple closed-loop DPCM system is shown in Figure 3.1. From Figure 3.1 we have

$$d_n = s_n - \tilde{s}_n$$

$$\hat{d}_n = \hat{s}_n - \tilde{s}_n$$

The quantizer quantization noise is

$$\begin{aligned} q_n &= d_n - \hat{d}_n \\ &= (s_n - \tilde{s}_n) - (\hat{s}_n - \tilde{s}_n) \\ &= s_n - \hat{s}_n \end{aligned}$$

Figure 3.1: DPCM

which is just the reconstructed quantization noise ($r_n = s_n - \hat{s}_n$). Thus, for DPCM

$$\Phi_{rr}(\omega) = \Phi_{qq}(\omega) \quad (3.1)$$

where $\Phi_{xx}(\omega)$ is the power spectral density of x . That is, the Z -transform of the autocorrelation of x evaluated on the unit circle ($z = e^{j\omega}$).

Thus for DPCM, the signal quantization noise is just the residual quantization noise. If the energy in the difference signal, d_n , is less than the energy in the sampled speech signal s_n , it can be quantized using a scalar quantizer more efficiently. Since d_n is relatively white, the quantization noise is also relatively white (spectrally flat).

An open loop DPCM system, which is usually referred to as D*PCM, is shown in Figure 3.2. Again, if we can assume that the quantization noise is white, which is not, at all, a bad assumption for a high SNR quantizer, then the signal quantization noise (r_n) has the same spectral envelope as $1/A(z) = 1/(1 - P_s(z))$. This is clearly shown in Figure 3.3, where $q_n = d_n - \hat{d}_n$. From the lower figure, it is apparent that the signal quantization noise has a spectral envelope of $1/A(z)$. In general, for D*PCM

$$\Phi_{rr}(\omega) = \Phi_{qq}(\omega) \left| \frac{1}{A(z)} \right|^2 \quad (3.2)$$

To obtain more control over the quantization noise spectrum a popular approach is to use an open loop DPCM system with noise feedback as shown in Figure 3.4. It can be shown, [5], that the power spectrum of the reconstructed quantization noise is just

$$\Phi_{rr}(\omega) = \Phi_{qq}(\omega) \left| \frac{1 - F(e^{j\omega})}{1 - P_s(e^{j\omega})} \right|^2 \quad (3.3)$$

Note that with $F(z) = P_s(z)$ we obtain DPCM, and with $F(z) = 0$ we obtain D*PCM. Thus, this structure is quite general.

Figure 3.2: D*PCM

Figure 3.3: DPCM, Two Equivalent Noise Models

Figure 3.4: DPCM-NFC

To minimize Equation 3.3 given a flat quantization noise spectrum (i.e. $\Phi_{qq}(\omega) = \text{constant}$) would maximize the coder SNR. Assuming $1 - P_s(z)$ is fixed, we must minimize

$$\int_{-\pi}^{\pi} |1 - F(e^{j\omega})|^2 / |1 - P_s(e^{j\omega})|^2 d\omega \quad (3.4)$$

with the criterion that

$$\int_{-\pi}^{\pi} \log(|1 - F(e^{j\omega})|^2) d\omega = 0 \quad (3.5)$$

Equation 3.5 must hold for a stable predictor as was discussed in Section 2.4.2. Using the technique of Lagrange Multipliers, it is simple to show that to maximize SNR

$$F(e^{j\omega}) = P_s(e^{j\omega})$$

which is just DPCM.

However, in terms of subjective speech quality, a different noise spectrum is usually desired, but always at a corresponding *decrease* in SNR. A good coverage of Noise Feedback Coding is available in [5,25,26] and the references contained therein.

Although the various forms of DPCM are not too important for the current study, the concepts of Noise Feedback Coding are invaluable. Many questions, have been left unanswered, such as

- how do we choose the fixed predictor $P_s(z)$;
- how do we design a suitable quantizer;
- what is a good noise weighting filter $F(z)$;
- does the performance degrade significantly when errors are introduced into the channel?

Many of these questions will be answered in the following sections, and many are answered in the above cited references.

Since only one parameter need be quantized (d_n), a DPCM system is quite simple to design and implement, both from systems and hardware points of view. However, the performance is inadequate for this study, due mainly to the non-adaptive predictor. In the next sections we begin to discuss more suitable speech coders.

3.1.2 Adaptive Predictive Coding

Adaptive Predictive Coding (APC) is very similar to the DPCM type systems we encountered in the previous section. However, the predictor $P_s(z)$ is now time varying, and a long term pitch predictor is usually embedded. An APC-NFC system is shown in Figure 3.5. The predictors $P_s(z)$ and $P_l(z)$ are discussed in Section 2.2, and are usually designed to minimize the open loop prediction residual (Figure 2.4).

Quantized versions of the short and long term predictor parameters, the quantized residual, and any quantizer adaptation information are transmitted to the receiver. The short term predictor ($P_s(z)$) and the long term predictor ($P_l(z)$) are updated every 10–20 msec. The pitch period M is typically between 20 and 160 samples, (2.5–20 msec at a 8 kHz sampling rate). The computation of the various parameters is detailed in [5,25,26].

APC capitalizes on the time varying spectral properties (formants) and the pitch information by periodically updating $A(z)$ and $B(z)$ respectively.

As an example we consider the voiced section of speech shown in Figure 3.6. The residual after short term prediction is shown in Figure 3.7. It is evident that considerable redundancy exists in the residual waveform. The sharp periodic impulses are the "pitch pulses". After open loop pitch prediction (filtering by $B(z)$) we obtain the signal shown in Figure 3.8. The

Figure 3.5: APC-NFC

pitch redundancy has appeared to have been filtered out. Note that in Figure 3.5 the input to the pitch predictor is the quantized residual. Thus, care must be exercised to ensure the stability of the pitch filter. Furthermore, the open loop prediction gains may far exceed the closed loop prediction gains.

APC typically suffers in low bit rate applications due to the scalar quantizer. That is, scalar quantization of the residual at low bit rates (< 1 bits/sample) is at best difficult, and poor speech quality results.

It was observed [25], that severe center clipping of the residual had little effect on the perceptual reconstructed speech quality. This undoubtedly led Atal and Remde [27] to devise the Multipulse Model (see Section 3.3.1).

3.2 Very Low Bit Rate Speech Coders

Vocoder¹ techniques are based on the model of speech production shown in Figure 2.3 (with $B(z) = 1$ and) where $v(n)$ is either unit variance white gaussian noise, or a series of impulses separated by the pitch period. The predictor $P_s(z)$ is determined using one of the previously discussed techniques, and the pitch period determination and voiced/unvoiced classification can use any one of many techniques, [3]. Notice that the reliability of voiced/unvoiced classifications and pitch period estimation are essential since unlike APC, no residual is transmitted. Vocoders are important in very low bit rate applications (< 2400 bps) but have very poor subjective and objective performance. In fact, the poor performance (subjective speech quality) of conventional LPC Vocoders was a major reason to perform this study. However, since Vocoders must transmit the vocal tract model ($P_s(z)$) and gain (G) very efficiently, we may borrow from the tech-

¹A contraction of Voice Coder.

Figure 3.6: Voiced Speech Segment

Figure 3.7: Residual after Short Term Prediction

Figure 3.8: Residual after Pitch Prediction

niques developed for Vocoders, (since we will require quantization of $P_s(z)$).

Considering the model, it is surprising Vocoders work at all. However, they are just another form of a residually-excited speech coder, with a strange method of quantizing the residual. Vocoders also typically behave poorly in the presence of channel errors. It is this model, albeit poor, which hinders the upper limit on the performance of Vocoders. Many speech frames cannot be modeled accurately as voiced or unvoiced. Furthermore, the model is an inexact model, which upper bounds the ultimate performance.

Due to the poor quality synthetic voice, Vocoders are not applicable to this study, and will not be discussed further.

3.3 Low Bit Rate Speech Coders

3.3.1 Multipulse Linear Predictive Coding (MPLPC)

We mentioned in Section 3.1.2 that it was observed severe center clipping of the residual in APC had little effect on the performance. Thus, only a few pulses per analysis interval are required to represent the residual. For voiced segments of speech, it was found that one pulse per pitch period was inadequate, and leads to poor quality LPC Vocoders. Thus, more than one pulse per pitch period may lead to a more natural sounding speech coder.

It was found that only 8 pulses per 10 msec interval was necessary to represent the residual accurately. Although MPLPC can be approached from both angles, a generalization of the LPC Vocoder or APC, MPLPC has its own, distinct, search algorithm for the best pulse amplitudes and locations.

The MPLPC search procedure is shown in Figure 3.9. The encoder (analysis) is based on the synthesis method (decoder). This technique is

termed analysis by synthesis, and is typically very computationally intensive. The synthesis filter ($1/A(z)$) is designed using the usual linear prediction techniques. The perceptual weighting filter, which has the same purpose as the noise feedback filter in APC, is typically given by

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

Note that for $\gamma = 1$, $W(z) = 1$, and a white quantization noise spectrum results. The quantization noise has a spectral envelope which is proportional to $|1/W(e^{j\omega})|^2$. The interesting range of γ is $0 \leq \gamma \leq 1$. Typically $\gamma \approx 0.73$. The parameter γ expands the zeros of the polynomial $A(z)$. We have

$$\begin{aligned} A(z) &= 1 - \sum_{k=1}^p a_k z^{-k} \\ A(z/\gamma) &= 1 - \sum_{k=1}^p a_k \gamma^k z^{-k} \end{aligned} \quad (3.7)$$

From the properties of the Z -transform [4], if h_n is the impulse response of $A(z)$, then the impulse response of $A(z/\gamma)$ is just $\gamma^n h_n$. Clearly, for $\gamma \leq 1$, (and $\gamma > 0$), the resulting weighting filter is stable. A plot of the frequency response of the noise weighting filter ($|1/A(e^{j\omega}/\gamma)|^2$), the spectral envelope ($|1/A(e^{j\omega})|^2$) for $\gamma = 0.73$ (all computed as in Section 2.4.1) is shown in Figure 3.10. The resulting error spectrum, ($|1/W(e^{j\omega})|^2$), is shown in Figure 3.11, for $\gamma = 0.73$ and for $\gamma = 1$. Note that the noise is enhanced in the formant (spectral peak) regions and suppressed in the spectral valleys in going from $\gamma = 1$ to $\gamma = 0.73$. Thus, an attempt is made to provide a relatively good SNR in all frequency bands, and *not* just in the area of the formant frequencies as is provided with a flat error spectrum. It again must be emphasized that a flat error spectrum maximizes the SNR, and the weighting filter above attempts to minimize a perceptual distortion.

Figure 3.9: MPLPC Search Procedure

Figure 3.10: Noise Weighting Filter

Figure 3.11: Resulting Error Spectrum

The closed loop search (Figure 3.9) is conducted by calculating the optimum pulse positions and amplitudes in a sequential manner. We have, for example, a block of speech N samples long, and wish to quantize the residual such that it has m pulses in each block, with pulses at locations $n_1, n_2, \dots, n_m$ with amplitudes $\alpha_1, \alpha_2, \dots, \alpha_m$. The procedure to calculate the optimum pulse location and amplitudes leads to a set of non-linear equations, [27,29]. Thus, the pulse locations and amplitudes are obtained in a sequential manner, which leads to a tractable solution. Often, the previous pulse amplitudes (but not locations) are re-optimized after each iteration [29]. The search is usually restructured by moving the location of the perceptual weighting filter as shown in Figure 3.12. A pitch predictor can also be incorporated into the search procedure to improve the performance [29]. The bit rate for MPLPC is typically in the 8–10 kbps range.

Rather than using the search procedure above, one could simply utilize a codebook of representative excitation sequences. The optimum excitation sequence is the codevector which minimizes the weighted mean squared error, as shown in Figure 3.13. The determination of the optimum codebook is a problem in Vector Quantization, and not easy to compute, although the computation of suboptimal (but useful) codebooks is tractable. Furthermore, the structure of the search in Figure 3.13 is identical to the search procedure used in Code-Excited Linear Prediction (CELP). CELP is the topic of Sections 3.3.3 and Chapter 5.

3.3.2 Vector Adaptive Predictive Coding (VAPC)

Vector APC is interesting, if only from a historical standpoint, since it was one of the first waveform coders to use Vector Quantization (VQ) successfully. VAPC is the VQ counterpart of APC: It consists of a vector linear

Figure 3.12: Simplified MPLPC Search Procedure

Figure 3.13: MPLPC Search Procedure Utilizing a Codebook

predictor (VP), and an embedded vector quantizer as shown in Figure 3.14. The vector linear predictor (a generality of the scalar counterpart) predicts future vectors as linear weighted sums of past vectors. Embedding a vector quantizer into a conventional source coding technique allows one to reap the benefits of both source coding techniques, while maintaining manageable complexity. The VQ can exploit the non-linear dependencies in the residual, as well as the pdf shape, (Chapter 4). Also, a VQ can operate at rates of 1 bit/sample and below; the point at which scalar quantizers tend to perform poorly.

The vector linear predictor is chosen adaptively from a codebook of L fixed vector linear predictors by a frame classifier which bases its decision on the speech statistics. The design of the frame classifier, predictor, and vector quantizer is described in [45]. VAPC obtains relatively good performance at bit rates in the 8–10 kbps range. The weak link in the VAPC structure is the vector linear predictor which has a small vector dimension and a small codebook size. Scalar linear predictors have better properties and performance since future predicted values are only *one* sample in the future.

The VAPC structure can also utilize a codebook of stored residual vectors, and use analysis by synthesis, as shown in Figure 3.15. This structure has many benefits, and only one major weakness, the computational complexity. The optimum codevector is found by an exhaustive search over the entire codebook of residual vectors. The codebook index of both the vector linear predictor, and the residual codebook is transmitted to the receiver. Note that since we are using analysis by synthesis we can replace the vector linear predictor with a simple scalar linear predictor. The redundancy removal properties of scalar linear predictors is better, and the quantization properties of the model filter is well understood. The incorporation of a

Figure 3.14: VAPC

Figure 3.15: VAPC, Analysis by Synthesis

noise weighting filter can also aid the performance. Rearranging Figure 3.15 with these changes we obtain Figure 3.16.

This structure, after the inclusion of a pitch predictor, is often referred to as Vector Excitation Coding (VXC). VXC is identical to CELP.² CELP is briefly discussed in the following section and is discussed in greater detail in Chapter 5.

3.3.3 Code-Excited Linear Prediction (CELP)

While studying APC, Atal and Schroeder [26,31] determined that the residual after both short and long term prediction is essentially white gaussian. They observed, [31], that by utilizing a codebook populated by white gaussian noise, and using analysis by synthesis, the residual could be encoded at a bit rate of only 2 kbps while maintaining excellent speech quality. However, they did not encode the short or long term predictor parameters or the gain term.

The search procedure is shown in Figure 3.17. The short and long term predictor coefficients are determined using standard LPC techniques, (see Section 2.4.1). The optimum excitation sequence is determined using an exhaustive search over the entire codebook. Thus, the codevector which minimizes the noise weighted mean squared error between the synthetic speech and natural speech is chosen. The noise weighting filter $W(z)$ weights the error signal appropriately and is based on how the human auditory system is believed to work. Typically

$$W(z) = \frac{A(z)}{A(z/\gamma)} \quad (3.8)$$

where $\gamma \approx 0.73$. The gain term G is calculated by minimizing the mean squared error between s_n and $\hat{s}_n$ for the particular codevector.

²Although historically they were developed along different lines.

Figure 3.16: VAPC, Enhanced Analysis by Synthesis

Figure 3.17: CELP

Atal and Schroeder used a 40 dimensional codebook, and 2^{10} possible excitation sequences. The codevectors were samples of unit variance white gaussian noise. For speech sampled at 8 kHz, this implies a bit rate of only 2 kbps for transmitting the residual (the codebook excitation sequence) assuming an equal probability of choosing each excitation sequence. For unequal and known probabilities, a lower entropy results, and some form of entropy coding could be used to reduce the bit rate further. It was determined, [31], that by using a codebook consisting of unit variance random gaussian numbers, or by constructing the codebook from random residual vectors, equivalent performance was obtained.

It was noted in [31] that the codebook designed in either manner was inherently sub-optimal, and that the codevectors should be optimally placed on the surface of an N -dimensional hypersphere, (with $N = 40$). This is just a problem in coding theory. Thus, if our requirement was to minimize the mean squared error given a uniform distribution of residual vectors over the surface of the sphere, we could simply choose a suitable rate $1/4$ (binary) code. Due to the noise weighting, and non-uniform distribution of the residual vectors, even a *perfect* rate $1/4$ code would not be optimum.³ However, a good code would be expected to work well, since a random gaussian code performs well, and the memory requirements would be minimal since only binary vectors would need to be stored.

The codevectors should be so placed as to minimize the noise *weighted* mean squared error. This is just a problem in Vector Quantization which is the topic of the next chapter. The CELP codebook design process is investigated further in Chapter 5.

³A perfect rate $1/4$, 40-dimensional code is not even known to exist [7]

Chapter 4

Vector Quantization

4.1 Introduction to Vector Quantization

In Section 2.1 we mentioned that in a digital communications system the transmitter could be separated into two functional entities; the source coder and the channel encoder. We also mentioned that the source coder will benefit from a delayed decision whereby blocks of data are analysed before making the quantization decision. Below, we give an intuitive feel for the reasons for these improvements.

Consider quantizing a signal x_k , where x_k exhibits a high sample to sample correlation. A simple scalar quantizer can exploit the pdf (probability distribution function) shape by assigning more bits to regions which are more likely, (i. e. a μ -law codec). A representative scatter plot of x_{2k} versus x_{2k+1} and the associated (scalar) quantization intervals are shown in Figure 4.1.

Clearly, bits are wasted in regions where the joint probability density function between x_{2k} and x_{2k+1} has low probability. Furthermore, with scalar quantization the cell shapes are confined to be rectangles and arbitrary cell

Figure 4.1: High Sample-Sample Correlation

shapes are not allowed.

Thus far, we have determined that quantization in higher dimensions can exploit pdf shape, linear correlation, and the cell shape. However, linear correlation can alternatively be exploited by using a rotation to obtain a new set of variables ($\hat{x}_k$) which are uncorrelated. For example, consider the joint pdf in Figure 4.2. By utilizing a suitable rotation, we can decorrelate x_{2k} and x_{2k+1} efficiently by utilizing the pdf shape and the linear dependency (see Figure 4.3). However, consider quantizing a signal with the joint pdf shown in Figure 4.4. Now, although x_{2k+1} and x_{2k} are uncorrelated (linearly independent), there exists a great deal of non-linear dependency which can be exploited by properly designing the quantization cells. An example of a possible set of cell shapes is shown in Figure 4.5. These cell shapes are not necessarily optimal, but undoubtedly would offer a performance advantage over a simple scalar quantizer. Clearly, quantizing in two (or more) dimensions (Vector Quantization) is much more efficient, (in terms of minimizing the distortion for a given rate or in terms of minimizing the bit rate for a given distortion), than scalar quantization.

Thus, Vector Quantization (Block, Joint Quantization, Codebook Coding) is the joint quantization of a block of parameters which takes advantage of the four interrelated properties [8]:

1. pdf shape;
2. linear dependency (correlation);
3. non-linear dependency;
4. vector dimensionality, (cell shape).

Property four implies that more efficient quantizers are possible by increasing the vector dimension. In fact, even for independent data, the

Figure 4.2: Joint PDF

Figure 4.3: Joint PDF: Decorrelated

Figure 4.4: Joint PDF: Non-linear Dependencies

Figure 4.5: Joint PDF: Vector Quantization

rate-distortion bound may *only* be reached as the vector dimension approaches infinity. This is a justification of the property that in higher dimensions, more efficient cell shapes can be utilized to approach the rate-distortion bound.

It must be emphasized that vector quantization works well with both scalar and vector sources. A vector source is a source in which a vector of parameters of dimension K is generated every T seconds. An example of a vector source is the set of predictor coefficients which are generated from LPC analysis. A scalar source is the degenerate case of a vector source with $K = 1$. Clearly, we may group N of these vectors together to obtain a new vector of dimension $\hat{K} = KN$ and perform vector quantization on this new vector of dimension $\hat{K}$. Conversely, we may split up the vector of dimension K into two smaller vectors with dimension K_1 and K_2 and use two vector quantizers to quantize the smaller vectors, (providing we have a suitable distortion measure).

By grouping vectors together, we will obtain a more efficient quantizer (closer to the rate-distortion bound), at the expense of an increase in complexity. By splitting up the vector, the performance decreases, but the complexity decreases as well.

It should be becoming clear, that a Vector Quantizer is simply a method of assigning an input vector (x_i) to a particular cell (from a usually finite ensemble L) in K -dimensional space. To each cell a minimum distortion centroid is assigned which is the reconstruction vector (or output vector) y_i for an input vector which is in cell i . The centroid is usually chosen to minimize the distortion over cell C_i .

Thus, we say x is quantized as y or

$$y = Q[x] \quad (4.1)$$

where x and y are both column vectors.

2 of/de 2

Typically, (although not essential)¹, $\mathbf{y}$ takes on a finite set of values from a codebook $\mathcal{Y}$. That is

$$\mathcal{Y} = \{\mathbf{y}_i, 0 \leq i < L\} \quad (4.2)$$

L is the size of the codebook, (an L -level codebook), and the $\mathbf{y}_i$ s are the K -dimensional codevectors (templates).

To design the codebook $\mathcal{Y}$, we partition the K -dimensional space of the random vector $\mathbf{x}$ into L regions or cells $\{C_i, 0 \leq i < L\}$, and associate with each cell C_i , a vector $\mathbf{y}_i$. The quantizer then assigns the codevector $\mathbf{x}$ to $\mathbf{y}_i$ if $\mathbf{x}$ is in C_i . That is

$$Q(\mathbf{x}) = \mathbf{y}_i \quad \text{if } \mathbf{x} \in C_i \quad (4.3)$$

The cell boundaries are constructed using a nearest neighbour calculation. Thus

$$Q(\mathbf{x}) = \mathbf{y}_i \quad \text{iff } d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j) \quad (4.4)$$

for $j \neq i, 0 \leq j < L$

Thus, $\mathbf{y}_i$ is the closest codevector (in terms of the distortion $d(\mathbf{x}, \mathbf{y})$) to $\mathbf{x}$. The average distortion is

$$D = \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{n=1}^M d(\mathbf{x}_n, \hat{\mathbf{x}}_n) \quad (4.5)$$

¹Note that even though the number of levels in the codebook may be infinite the entropy could be finite. That is, $p_i \log_2 p_i$ is small if p_i is small.

Thus, the entropy

$$H(\mathbf{y}) = - \sum_{i=1}^{\infty} p_i \log_2 p_i$$

may be finite.

However, the codebook size must be finite if the quantizer is to be implementable.

where $\mathbf{x}_n$ and $\hat{\mathbf{x}}_n$ are the n^{th} input vector and reconstruction vector respectively, out of the ensemble M . As a notational convenience, we let $\mathbf{y}$, or $\hat{\mathbf{x}}$ denote the quantized value of $\mathbf{x}$. Thus, $\hat{\mathbf{x}}_n$ is the n^{th} reconstruction vector, and is an element from the codebook $\mathcal{Y}$.

If $\mathbf{x}$ is stationary and ergodic then

$$\begin{aligned} D &= E[d(\mathbf{x}, \mathbf{y}_i)] \\ &= \sum_{i=0}^{L-1} P(\mathbf{x} \in C_i) E[d(\mathbf{x}, \mathbf{y}_i) | \mathbf{x} \in C_i] \\ &= \sum_{i=0}^{L-1} P(\mathbf{x} \in C_i) \int_{C_i} d(\mathbf{x}, \mathbf{y}_i) p(\mathbf{x}) d\mathbf{x} \end{aligned} \quad (4.6)$$

where $p(\mathbf{x})$ is the multidimensional pdf of $\mathbf{x}$, and $P(\mathbf{x} \in C_i)$ is the discrete probability that $\mathbf{x}$ is in C_i . Each vector $\mathbf{y}_i$ is encoded into a codeword of binary digits, m_i , with length B_i bits. In general, the different codewords have different lengths. We define

$$B = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=1}^N B_n \quad \text{bits/vector} \quad (4.7)$$

The transmission rate is

$$R = 1/T = BF_c \quad \text{bits/sec} \quad (4.8)$$

where F_c is the number of codewords (or vectors) per second.

The number of bits per dimension, or the number of bits per vector element is

$$R_d = B/K \quad (4.9)$$

A K -dimensional vector quantizer is said to be optimal if it minimizes the average distortion D in Equation 4.6. Combining Equations 4.1 and 4.6 we obtain (for the K -dimensional quantizer Q^*)

$$D_{Q^*} = E[d(\mathbf{x}, Q^*[\mathbf{x}])] \quad (4.10)$$

Thus, Q^* is optimal if and only if

$$D_{Q^*} \leq D_Q \quad (4.11)$$

for all quantizers Q , (of dimension K).

Note that there are two criteria of optimality, optimality of the codebook, and optimality of the codebook search for the optimum codevector. The former criterion, as we have said, is realized by minimizing Equation 4.10 and the latter by using the minimum distortion search (over the whole codebook)² described by Equation 4.4.

In the next section, we utilize these two criteria to obtain an iterative design procedure based on a training sequence of representative data vectors ($\mathbf{x}_n$).

4.2 Codebook Design

By utilizing Equation 4.4 and 4.11, it is possible to design a vector quantizer based on either a known probability density ($p(\mathbf{x})$), or on a sufficiently rich training sequence of representative data. We will review the procedure for the design based on the training sequence, since this is most applicable to our research. For a summary of both methods see [14].

Suppose we are given a training sequence

$$\mathcal{T} = \{\mathbf{x}_n, 0 \leq n < M\}, \quad M \gg L$$

and an initial codebook $\mathcal{Y}^{(0)}$.

$$\mathcal{Y}^{(0)} = \{\mathbf{y}_i^{(0)}, 0 \leq i < L\},$$

²An exhaustive search is not necessary if the codebook is well structured (by luck, or by design).

That is, the training sequence has M vectors, the n^{th} one being $\mathbf{x}_n$, and the initial codebook has centroids $\mathbf{y}_i^{(0)}$ corresponding to cell C_i . A subset M_i of the codevectors (from the training sequence) will be in cell C_i . The new codebook elements (codevectors) are computed by calculating the centroid of the training vectors which belong to cell C_i , utilizing the minimum distortion search. That is

$$\mathbf{y}_i^{(1)} = \text{cent}(C_i) \quad 0 \leq i < L \quad (4.12)$$

The process is repeated until the average distortion over the whole training sequence stabilizes. That is

$$D^{(k)} = \frac{1}{M} \sum_{n=0}^{M-1} d(\mathbf{x}_n, \hat{\mathbf{x}}_n^{(k)}) \quad (4.13)$$

where $\hat{\mathbf{x}}_n^{(k)}$ is the n^{th} reconstruction vector at the k^{th} iteration. The iteration is terminated (at iteration k) when

$$\frac{D^{(k)} - D^{(k-1)}}{D^{(k)}} \leq \epsilon \quad (4.14)$$

where ϵ is some small number (typically $\epsilon \approx 10^{-3}$).

Thus, we only require a distortion measure, a centroid calculation, an initial codebook, and a sufficiently rich training sequence to design a vector quantizer. A random codebook³ is often used as the initial codebook. An algorithmic description of VQ design is shown in Figure 4.6.

This algorithm is called K -means or the LBG algorithm after [14]. The distortion is guaranteed to decrease (or at worst stay the same) with each iteration. However, only locally optimal codebooks are guaranteed with this algorithm [11,14]. This is a rather moot point since our training sequence is only a small portion of the ensemble. Thus, even if we had an optimum

³A random codebook is simply a random choice of L distinct vectors from the training sequence.

VQ Design:

1. choose by some method an initial set of codevectors

$$\mathcal{Y}^{(0)} = \{\mathbf{y}_i^{(0)}, 0 \leq i \leq L-1\},$$

$$\text{set } D^{(-1)} = 0$$

$$\text{set } m = 0.$$

2. classify the set of training vectors $\mathcal{T} = \{\mathbf{x}_n, 0 \leq n < M\}$ into the cells $C_i^{(m)}$ utilizing the nearest neighbour rule

$$\mathbf{x}_n \in C_i^{(m)} \text{ iff } d(\mathbf{x}_n, \mathbf{y}_i^{(m)}) \leq d(\mathbf{x}_n, \mathbf{y}_j^{(m)})$$

for all $j \neq i$, and all $n, 0 \leq n < M$.

3. update the codevector of every cluster by computing the new centroid of the training vectors in each cell

$$\mathbf{y}_i^{(m+1)} = \text{cent}(C_i^{(m)}) \quad 0 \leq i \leq L-1$$

4. compute the total distortion over the whole training sequence

$$D^{(m)} = \sum_{n=0}^{M-1} d(\mathbf{x}_n, \hat{\mathbf{x}}_n^{(m)})$$

where $\hat{\mathbf{x}}_n^{(m)}$ is given by the nearest neighbour rule above.

if $(D^{(m)} - D^{(m-1)})/D^{(m)} \leq \epsilon$ terminate

else increment m , and goto step 2.

Figure 4.6: Algorithmic Description of VQ Design

codebook for our training sequence, it would not necessarily be optimum for data outside the training sequence. Thus, the training sequence should be large enough to obtain good performance on data outside the training sequence (typically $M \approx 20 \cdot L$).

We will consider a number of distortion measures, the mean squared error, the IS-distortion, the I-distortion, and the Inverse Filter Matching (IFM) principle.

The mean squared error (mse) distortion is

$$d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}) \quad (4.15)$$

and the centroid, assuming M_i training vectors in cell C_i , is given by

$$\begin{aligned} \mathbf{y}_i &= \text{cent}(C_i) \\ &= \frac{1}{M_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}_n \end{aligned} \quad (4.16)$$

which is simply the sample mean. It can easily be shown that the sample mean minimizes the distortion over the training sequence vectors in cell C_i .

For the IS-distortion measure

$$d_{IS}(|S|^2, |A/\sigma|^2) = \frac{1}{2\pi} \int_{-\pi}^{\pi} (|SA/\sigma|^2 - \ln(|SA/\sigma|^2) - 1) d\omega$$

The centroid is computed by minimizing the expected distortion D_i over the M_i training vectors in cell C_i . Thus

$$\begin{aligned} D_i &= \frac{1}{M_i} \sum_k d_{IS}(|S^{(k)}|^2, |A/\sigma|^2) \\ &= \frac{1}{2M_i\pi} \int_{-\pi}^{\pi} |A/\sigma|^2 \sum_k |S^{(k)}|^2 d\omega - \frac{1}{M_i} \sum_k \ln(\alpha_{\infty}^{(k)}/\sigma^2) - 1 \end{aligned} \quad (4.17)$$

where $|S^{(k)}|^2$ is the k^{th} spectra in cell C_i , $\alpha_{\infty}^{(k)}$ is the gain of $|S^{(k)}|^2$, A and σ^2 are the model and gain centroids for cell C_i .

Let $\bar{\alpha}_{\infty}$ be the gain of the mean speech spectrum. That is, the gain of

$$\frac{1}{M_i} \sum_k |S^{(k)}|^2 = \overline{|S|^2}$$

Thus

$$\begin{aligned} D_i &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |A/\sigma|^2 \overline{|S|^2} d\omega - \ln(\bar{\alpha}_\infty/\sigma^2) - 1 - \frac{1}{M_i} \sum_k \ln(\alpha_\infty^{(k)}/\bar{\alpha}_\infty) \\ &= d_{IS}(\overline{|S|^2}, |\sigma/A|^2) - \frac{1}{M_i} \sum_k \ln(\alpha_\infty^{(k)}/\bar{\alpha}_\infty) \end{aligned}$$

The second term depends only on the spectra of the training vectors and not on the model spectra. Thus, the centroid $|\sigma/A|^2$, is given by minimizing

$$d_{IS}(\overline{|S|^2}, |\sigma/A|^2) \quad (4.18)$$

This is equivalent to utilizing the Autocorrelation Method on the mean spectra, or simply using the Autocorrelation Method on the arithmetic mean of the autocorrelation vectors, $(\overline{r^{(k)}})$, in cell C_i of the training sequence.

For the Gain Optimized IS-distortion, (the I-distortion), the centroid is computed by minimizing

$$\begin{aligned} D_i &= \frac{1}{M_i} \sum_k d_{IS}(|S^{(k)}|^2, |\sqrt{\alpha}/A|^2) \\ &= \frac{1}{M_i} \sum_k \ln(\alpha^{(k)}/\alpha_\infty^{(k)}) \end{aligned}$$

If $\alpha^{(k)}/\alpha_\infty^{(k)} \approx 1$, then the arithmetic mean is a good approximation to the geometric mean. However, $\alpha_p^{(k)}$ is much easier to compute than $\alpha_\infty^{(k)}$. Furthermore

$$D_i = \frac{1}{M_i} \sum_k \ln(\alpha^{(k)}/\alpha_p^{(k)}) + \frac{1}{M_i} \sum_k \ln(\alpha_p^{(k)}/\alpha_\infty^{(k)})$$

The second term depends only on the spectra of the training vectors and the model spectra but *not* on the centroid A . Also, since $\alpha^{(k)}/\alpha_p^{(k)} \approx 1$ we may utilize the arithmetic mean to upper bound the geometric mean. Thus

$$D_1 \approx \frac{1}{M_1} \sum_k \alpha^{(k)} / \alpha_p^{(k)} \quad (4.19)$$

The minimization of Equation 4.19 is equivalent to minimizing

$$d_{IS}(\overline{|S|^2}, |\sqrt{\alpha}/A|^2) \quad (4.20)$$

where

$$\overline{|S|^2} = \frac{1}{M_1} \sum_k |S^{(k)}|^2 / \alpha_p^{(k)}$$

This is equivalent to averaging the weighted spectra, (normalized spectra), or averaging the normalized autocorrelation vectors, $(\mathbf{r}^{(k)} / \alpha_p^{(k)})$, and computing the model filter (the centroid) using the autocorrelation method.

The IFM distortion is given by d_{IFM} where

$$d_{IFM}(\Phi, \phi, \mathbf{a}) = \phi_{00} - 2\mathbf{a}^T \phi + \mathbf{a}^T \Phi \mathbf{a} \quad (4.21)$$

The centroid is calculated by minimizing Equation 4.21 over cell C_1 . That is we minimize

$$\begin{aligned} D_1 &= \sum_{k \in C_1} \phi_{00}^{(k)} - 2\bar{\mathbf{a}}^T \phi^{(k)} + \bar{\mathbf{a}}^T \Phi^{(k)} \bar{\mathbf{a}} \\ &= \sum_{k \in C_1} \phi_{00}^{(k)} - 2\bar{\mathbf{a}}^T \left(\sum_{k \in C_1} \phi^{(k)} \right) + \bar{\mathbf{a}}^T \left(\sum_{k \in C_1} \Phi^{(k)} \right) \bar{\mathbf{a}} \end{aligned}$$

The centroid calculation is simply computed by Choleski Decomposition on the mean covariance matrix and vector in each cell.

In summary, the four distortion measures, and their corresponding centroid calculations are shown in Table 4.1.

4.3 Computational and Storage Costs of VQ

When addressing the complexity issue of VQ one is immediately astounded by the huge computational and memory burden imposed by vector quantization, as compared to scalar quantization. In this section we calculate the

Distortion Measure	Centroid Computation
mse, Equation 4.15	$\sum_{k \in C_i} x^{(k)}$
IS-distortion, Equation 2.34	$\sum_{k \in C_i} r^{(k)}$
I-distortion, Equation 2.43	$\sum_{k \in C_i} r^{(k)} / \alpha_p^{(k)}$
IFM-distortion, Equation 4.21	$\sum_{k \in C_i} \Phi^{(k)}, \sum_{k \in C_i} \phi^{(k)}$

Table 4.1: Distortion Measures and Centroids

complexity of VQ (computational and memory) under certain simplifying assumptions. In the following sections we detail methods to reduce this complexity.

Assume we have a scalar source emitting scalars at a rate of f_s samples per second. We group the scalars into blocks of size K , and encode each vector using B bits. Thus, we are encoding each sample using $B/K = R_d$ (Equation 4.9) bits. Thus, the codebook size is just

$$L = 2^B \quad (4.22)$$

Assuming each distortion computation requires K multiply adds, which is reasonable for both the mse-distortion and I-distortion, the computational complexity⁴ for an exhaustive search over the whole codebook is just

$$C = KL \text{ FLOP/vector} \quad (4.23)$$

$$= KLf_s/K \text{ FLOPS}$$

$$= f_s L \text{ FLOPS} \quad (4.24)$$

$$= f_s 2^B \text{ FLOPS} \quad (4.25)$$

The memory requirement (assuming K storage locations per vector) is

$$\mathcal{M} = KL \quad (4.26)$$

$$= K2^B \quad (4.27)$$

Thus, the computational complexity, (C), is proportional to the sampling rate and exponential in the number of bits per vector (B). Alternatively, using Equations 4.9 and 4.25, the complexity is exponential in the number of bits per dimension *and* in the vector dimension, (and proportional to the sampling rate). Similarly, the memory complexity is proportional to the vector dimension and exponential in the number of bits per

⁴A FLOP is a FLoating point OPeration; typically a multiply and an addition. FLOPS are FLoating point OPerations per Second.

vector (B). One obvious method to decrease the complexity is to decrease the vector dimension. However, the performance penalty is usually severe.

As an example, assume we have

$$f_s = 8 \text{ kHz}$$

$$K = 10$$

$$R_d = 1$$

Therefore

$$B = 10, \quad \text{and} \quad L = 2^{10}$$

and the bit rate for the source is

$$f_s R_d = 8 \text{ kbps}$$

The computational complexity is

$$\begin{aligned} C &= f_s L \\ &= 8 \cdot 2^{10} \text{ KFLOPS} \\ &\approx 8 \text{ MFLOPS} \end{aligned}$$

and the memory complexity is

$$\begin{aligned} \mathcal{M} &= KL \\ &= 10 \cdot 2^{10} \text{ WORDS} \\ &\approx 10 \text{ KWORDS} \end{aligned}$$

In the following sections we discuss methods to reduce this immense computational burden; often at the expense of an increase in memory requirements.

4.4 Tree Searched VQ

In Section 4.3 we determined that the costs of VQ (computational and memory requirements) are exponential in the number of bits per vector. We primarily want to reduce the computational complexity due to the exhaustive search of the codebook by performing a non-exhaustive search over a portion of the codebook. It is simplest to think in terms of an mse distortion measure, but the results below also apply to other distortion measures.

Suppose, for example, that we could locate a K -dimensional hyperplane which "split" the codebook (of size L) into two halves, (each with $L/2$ codevectors). We would then need only two distortion computations to determine which half of the codebook to use for the subsequent full search over the smaller ($L/2$) codebook. Figure 4.7 diagrammatically explains this proposed method, (with $K = 2$ and $L = 8$). Note that for this method to succeed, line $a-b$ must be straight and split the codebook into two halves. Due to the irregular shapes of the cell boundaries, the existence of a K -dimensional hyperplane which splits the codebook along the cell boundaries is very improbable. Thus, we impose regularity in the codebook during the codebook design process, which results in a suboptimal codebook but a much reduced search complexity. That is, we are forcing the codebook into a certain form which, in general, leads to a suboptimal codebook.

To design this well structured codebook we first design a codebook with $L = 2$ based on the training sequence of data. With this codebook, we can split the training sequence into two halves, using the minimum distortion search. We then design two more ($L = 2$) codebooks based on these two training sequences. We can then use these codebooks to split the two training sequences. Thus, we have four training sequences now, and the codebook design process continues until the desired number of codebook

Figure 4.7: Splitting the Codebook

entries is reached. The design of a codebook with $L = 8$ and $K = 2$ is shown diagrammatically in Figures 4.8, 4.9, and 4.10. Thus, the K -means algorithm, with $L = 2$, is used at each step to design a codebook for a portion of the training sequence. The search procedure consists of a binary tree search over the various codebooks as shown in Figure 4.11. Recall, v_0 and v_1 are the centroids of the codebook after using the K -means algorithm with $L = 2$. The vectors v_2 and v_3 are the centroids of the codebook designed using that portion of the codebook which was "closer" to v_0 than v_1 , (that is, after the codebook was split into two).

With input vector x , the search for the codevector $y = \hat{x}$ is accomplished by traversing the branches with minimum distortion until the bottom of the tree is reached. Thus, the intermediate centroids are required in the codebook design, and in the codebook search process, resulting in increased memory requirements.

Note that the search for the codevector is optimal, a better codevector in the *codebook* cannot be found. Nevertheless, the codebook is suboptimal. That is, Equation 4.4 is minimized over the whole codebook during the binary search since the codebook is, by virtue of the design process, well structured. Furthermore, the structure we impose on the codebook, results in the codebook suboptimality.

The computational cost of the binary tree search is substantially lower than the full search. In fact

$$\begin{aligned} C &= 2KB \text{ FLOP/vector} \\ &= 2Bf, \text{ FLOPS} \end{aligned}$$

and

$$M = 2K(L - 2)$$

although slightly lower complexity can be realized with certain distortion

Figure 4.8: Codebook Design: Stage 1

Figure 4.9: Codebook Design: Stage 2

Figure 4.10: Codebook Design: Stage 3

Figure 4.11: Binary Tree Search

measures [11]. Note that the computational complexity has decreased enormously (it is no longer exponential in B), although the memory complexity has nearly doubled.

Using our old example, ($f_s = 8$ kHz, $K = B = 10$, $R_d = 1$, $L = 2^{10}$) we get

$$\begin{aligned} C &= 2 \cdot 10 \cdot 8000 \text{ FLOPS} \\ &= 160 \text{ KFLOPS} \\ M &= 2 \cdot 10 \cdot (2^{10} - 2) \text{ WORDS} \\ &\approx 20 \text{ KWORDS} \end{aligned}$$

In this example the computational complexity has decreased by a factor of 50 while the memory requirements have roughly doubled.

The search procedures in the above cases are termed uniform binary since each node in the tree has exactly two (binary) branches and the tree consists of exactly B intermediate nodes (uniform), as the tree is traversed from top to bottom. Rather than dividing the tree uniformly at each node (in the codebook design process) one could divide only the node which contributes the highest average distortion. That is, in the training phase at each point in the subdivision process, the total distortion contributed by each node is examined and the node which contributes the largest distortion is subdivided next. Contrast this method (binary non-uniform) with the binary uniform tree whereby *every* node is subdivided at each stage. A non-uniform binary tree is shown in Figure 4.12, for $L = 9$. The non-uniform binary trees may have a codebook size which is not a power of two. This does not offer any incentive to use non-binary trees since the codevector index is usually encoded into B binary digits. However, non-uniform binary has an advantage (in terms of minimizing average distortion) over uniform binary, although the search is more complicated and perhaps longer.

Figure 4.12: Non-Uniform Binary Tree Search

Non-binary, and in fact non-binary non-uniform, trees are possible and often give better performance at a slight increase in computational complexity. For example, instead of splitting a node into two sub-nodes (as in the binary case) one could split each node into four, five, or any reasonable number of nodes. For example, a 2-3-4 tree search vector quantizer is shown in Figure 4.13. The codebook size is given by the product of the number of branches emanating from each node as one descends through the tree. For example, a 2-3-4 tree searched vector quantizer has $2 \cdot 3 \cdot 4 = 24$ codebook elements.

Consider once again our standard example, ($f_s = 8$ kHz, $K = B = 10$, $R_d = 1$, $L = 2^{10}$), and a 32-32 uniform search,⁵

$$\begin{aligned} C &= K(32 + 32) \text{ FLOP/vector} \\ &= f_s(32 + 32) \text{ FLOPS} \\ &\approx 500 \text{ KFLOPS} \\ \mathcal{M} &= K(32 + 32 \cdot 32) \text{ WORDS} \\ &\approx 10 \text{ KWORDS} \end{aligned}$$

Here we have a factor of 16 less computationally with only a very slight increase in memory requirements (10,560 WORDS instead of 10,240).

Note that the distortion performance of the non-binary tree searches are clearly inferior to the full search codes. However, they offer memory advantages over binary tree searched codes with only a slight computational expense. Furthermore, the non-binary codes typically offer better distortion performance over the binary codes since the cell shapes are less restrictive.

⁵Note that $L = 1024 = 32 \cdot 32$

Figure 4.13: 2-3-4 Tree Search

4.5 Cascaded VQ

The major advantage of binary tree searched VQ over full search VQ is a large decrease in the number of distortion computations required to determine the (suboptimal) codevector. The memory requirements have, however, increased. Cascaded (or Multistage) VQ is a method whereby both computations and memory requirements may be decreased. Cascaded VQ consists of a number of VQ stages, each stage operating on the residual of the previous stage as shown in Figure 4.14.

In Figure 4.14 the two stage cascaded VQ utilizes B_1 bits in quantizing x and B_2 bits in quantizing u for a total of

$$B = B_1 + B_2$$

$$L_1 = 2^{B_1}$$

$$L_2 = 2^{B_2}$$

The quantized value is just

$$Q(x) = z + A^{-1}w$$

The matrix A and its purpose is discussed in [11]. The complexity of cascaded VQ is just

$$C = f_s(L_1 + L_2) \text{ FLOPS}$$

$$M = K(L_1 + L_2) \text{ WORDS}$$

Additional memory is also required for the $K \times K$ matrix A .

With $B_1 = B_2 = 5$ the computational complexity is the same as the 32-32 tree searched VQ although the memory requirements are greatly reduced. However, the distortion performance of Cascaded VQ is much inferior to full searched and tree searched VQ, [11]. There is a certain amount of similarity between Cascaded VQ and Tree Searched VQ. For

Figure 4.14: Two Stage Cascaded Vector Quantization

example, the first stage of the 32-32 tree search, and the first stage of the cascaded (5-5) VQ are identical, (the codebooks and the search technique). However, the tree searched VQ has another codebook (of size 32) for each node in the tree, and the cascaded VQ has only one codebook (the second stage). Thus, tree searched VQ can be thought of as cascaded VQ with numerous second stage codebooks, and the one chosen is conditional upon the first search result. Since there is a different codebook for every initial search result, the differencing operation (see Figure 4.14) is not necessary. Alternatively, cascaded VQ can be thought of as tree search VQ, with all groups of branches past the second node equal.

4.6 Product Code VQ

A product code is nothing more than splitting up a vector (of length K) into two (or possibly more) component vectors of length K_1 and K_2 where $K_1 + K_2 = K$. Clearly, for a scalar source, this is nothing more than using a smaller vector dimension. For truly vector sources this eases computational burden and memory requirements substantially while decreasing the overall distortion performance. For some distortion measures (the I-distortion) the vector can not be split up into a number of component vectors and thus product code VQ is not relevant. However, some authors use the term "product code" fairly loosely, and include Cascaded VQ as a product code.

Product Code VQ is extensively discussed in [11,17,18,19]. A particularly interesting type of product code is termed a Gain Shape Vector Quantizer (GSVQ), and is the subject of the next section.

4.7 Gain Shape VQ

A Gain Shape Vector Quantizer (GSVQ) is a product code vector quantizer whereby the component vectors are all normalized and the gain (normalization factor) and shape (normalized vector) are quantized independently. A VQ based on the gain optimized Itakura-Saito distortion is, in fact, a GSVQ.

Utilizing the mse distortion, consider a vector

$$\mathbf{v}^T = (v_0, v_1, \dots, v_{K-1})$$

then

$$\sigma^2 = \mathbf{v}^T \mathbf{v}$$

and the normalized vector is

$$\begin{aligned} \mathbf{x}^T &= (x_0, x_1, \dots, x_{K-1}) \\ &= (v_0/\sigma, v_1/\sigma, \dots, v_{K-1}/\sigma) \end{aligned}$$

Now, by quantizing $\mathbf{x}$ using a VQ, and scalar quantizing σ , we obtain a GSVQ. The VQ must be designed to optimally place the codevectors on the surface of an N -dimensional sphere. Thus, the GSVQ design algorithm and centroid calculations are slightly different.

To encode a vector $\mathbf{x}$, we must minimize (using the mse)

$$\begin{aligned} \epsilon &= \sum_{k=0}^{K-1} (\sigma x_k - y_k^{(i)})^2 \quad i = 0, 1, \dots, L-1 \\ &= \sigma^2 \sum_{k=0}^{K-1} x_k^2 - 2\sigma \sum_{k=0}^{K-1} x_k y_k^{(i)} + \sum_{k=0}^{K-1} (y_k^{(i)})^2 \end{aligned}$$

where the codebook

$$\mathcal{Y} = (\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(L-1)})$$

The first and last term in Equation 4.28 are independent of the codeword ($\sum y_k^2 = 1$), and thus can be omitted in the search procedure. Furthermore, σ depends only on the vector being quantized and not on the current codevector.

Hence, we must maximize

$$\begin{aligned}\epsilon &= \sum_{k=0}^{K-1} x_k y_k^{(i)} \\ &= \mathbf{x}^T \mathbf{y}^{(i)} \quad \forall i\end{aligned}$$

Thus, the nearest neighbour distortion computation is simply a correlation, (or inner product). The centroid calculation is computed by normalizing the average of the sum of all training vectors in each cell, [17,18,19]. Thus, we can design a GSVQ, based on the mse, using the same technique as in Section 4.2 using the above centroid calculation and nearest neighbour search procedure.

4.8 Vector Quantization of the LPC Model Filter

In Section 2.4.4 we examined a number of different parameter transformations suitable for use with Vector Quantization. Section 2.4.2 discussed the Itakura-Saito distortion measure and the gain optimized IS-distortion measure, and how these methods could be used to implement a VQ. Section 2.4.3 discussed the Inverse Filter Matching principle and its use in a VQ approach. The codebook design approach was investigated in Section 4.2.

We are primarily interested in five LPC model filter quantization methods:

1. VQ of the LARs;
2. VQ of the LSFs;
3. VQ based on IFM;
4. VQ based on the IS-distortion;
5. VQ based on the I-distortion.

The LARs were devised to obtain uniform spectral sensitivity to perturbations of the reflection coefficients [6]. Thus, uniform quantization of the LARs, or vector quantization of the LARs using an mse distortion measure, results in (approximately) uniform spectral sensitivity. The IS-distortion, or the I-distortion is a spectral matching formulation and is much less conceptually complex than computing LARs.⁶ Also, as discussed previously, the I-distortion utilizes the same distortion measure in the identification procedure, as it does during the quantization procedure.

The advantage of the Line-Spectral-Frequencies, (besides the obvious stability criterion), is the ability to allocate certain LSFs more bits in accordance with their perceptual importance. However, a distortion measure, for the LSFs, which would be suitable for VQ is not at all apparent. In [54] a crude 12 bit VQ based on the LSFs was constructed, but a suitable system was not realized. However, they maintain that scalar quantization of the LSFs offer improved performance over scalar quantization of the LARs. They also point out that 10 bit VQ of the model filter is inadequate, even for Vocoders. One could, however, use cascaded VQ or a product code on

⁶Recall that for the I-distortion we need only compute the autocorrelation vector and one distortion computation for each member of the codebook. To compute the quantized LARs, we require the Levinson-Durbin Algorithm, a few logarithms, and one distortion computation for each member of the codebook.

the LSFs to obtain a more suitable system. The interleaving property must always hold, however, which would severely complicate the construction of a product code, or cascaded VQ.

The I-distortion quantizes the model filter in one step. Thus, spectral properties (i.e. high order reflection coefficients or high order predictor parameters) which are not too important in terms of the I-distortion are encoded in very few bits. Spectral properties which are judged to be perceptually more important could be weighted by weighting the input speech with an appropriate spectral function.

The above related studies were concerned with Vocoders, in which no attempt is made to maximize signal to noise ratio, (or even weighted signal to noise ratio). In our study, (with CELP), a weighted signal to noise ratio is maximized. Thus, if we attempt to maximize some inconsistent *ad hoc* SNR measure to obtain the model filter, the effects will be counterbalanced by the residual codebook search to minimize the weighted mean squared error. It would be sensible to design the model filter in order to minimize the weighted mean squared error *after* the codebook search. However, this is not an easy task. Alternatively, we could try to minimize the mean squared residual energy (after filtering through $A(z)$). However, this is exactly what the I-distortion or IFM principle attempts. We have previously observed that the IFM principle is very computationally complex and approximately equal to the I-distortion.

The I-distortion offers certain advantages over the IS-distortion since the gain of the codec (G) is to be computed by other means, (see Section 3.3.3 and Chapter 5).

We shall consider two quantization procedures: cascaded VQ based on the I-distortion; and product code VQ based on the LSPs.

Due to the nature of the I-distortion, splitting up the model into a

product code form, using standard techniques, is not possible. Cascaded VQ, as shown in Figure 4.15, is expected to perform much better than a single 10-bit codebook.⁷ The predictor, $A_1(z)A_2(z)$ can be constructed by convolving the two predictors.

That is

$$\begin{aligned} A_1(z) &= 1 - \sum_{k=1}^p a_{1k} z^{-k} \\ A_2(z) &= 1 - \sum_{k=1}^p a_{2k} z^{-k} \end{aligned}$$

Thus

$$\begin{aligned} A(z) &= A_1(z)A_2(z) \\ &= \left(1 - \sum_{k=1}^p a_{1k} z^{-k}\right) \left(1 - \sum_{k=1}^p a_{2k} z^{-k}\right) \\ &= 1 - \sum_{k=1}^{2p} a_k z^{-k} \end{aligned}$$

It can be shown, by algebra, that

$$a_k = - \sum_{j=0}^p a_{2j} a_{1(k-j)} \quad \text{where } a_{20} = a_{10} = -1$$

To decrease the order of the resulting predictor we could simply drop the p higher order predictor parameters. However, this would not ensure the stability of the resulting model filter. Alternatively, we transform the predictor parameters into reflection coefficients and drop the p higher order reflection coefficients; this ensures the stability of the resulting filter. The reflection coefficients are then transformed back into predictor parameters.

The performance of the various techniques were tested based on a speech database consisting of twelve speakers. Each utterance was twenty seconds

⁷ Many authors may call this a product code, as mentioned in Section 4.6.

Figure 4.15: Model Filter Cascaded VQ

in duration, well balanced, with seven male speakers and five female speakers. The segments were recorded off local radio stations, and consisted of speech with low background noise. The recordings were played back through an eighth order maximally flat low pass antialiasing filter⁸ and followed by 12-bit linear analog to digital conversion⁹. The speech was then digitally highpass filtered with a first order highpass filter with $f_c \approx 100$ Hz. Thus, with a window size of 10 msec, we obtained 24,000 autocorrelation vectors in which to train our vector quantizers.

The computations were performed on a SUN-3 with a floating point accelerator installed (fpa). To design a full search 10 bit ($L = 1024$) VQ based on the I-distortion required approximately 12 hours of stand alone computing time.

Figure 4.16 shows the distortion performance of full search codes, inside the training sequence, based on the I-distortion for codebook sizes between 2 and 1024. The figure shows the average I-distortion between the quantized model filter and the optimum model filter

$$D_{avg} = \frac{1}{M} \sum_{k=0}^{M-1} \ln(\alpha_p^{(k)} / \alpha^{(k)})$$

The average distortion for a 32-32 tree searched code, full search random codes, a 5-5 and 10-10 product code, and a 10-10 product code with 32-32 tree search of each stage is also shown in Figure 4.16. the random codebooks were simply designed by randomly choosing L vectors from the training sequence.

The performance is slightly biased, since the quantizer is tested based on the same data used to design the codebooks. That is, the performance was tested *inside the training sequence*. The distortion performance based on

⁸Krohn-Hite model 3202 with two fourth order ($f_c = 3.3$ kHz) sections in series

⁹Data Translation Model DT2828

data outside of the training sequence, a twenty second segment consisting of two speakers (male and female), is shown in Figure 4.17. Note that the I-distortion is approximately the natural logarithm of the prediction gain. In our computation, however, we are referencing to the optimum p^{th} order model. Thus, to obtain the average difference in the prediction gain (ΔG_p in dB) between the optimum model filter and the quantized model filter we compute

$$\Delta G_p = 10 D_{\text{avg}} / \ln(10)$$

The 32-32 tree searched codes perform remarkably well at a computational complexity well below the full search codes, and at a slightly larger memory complexity. Note that the performance inside and outside of the training sequence is markedly different. The performance inside the training sequence for a random codebook is better than full search codes outside the training sequence. The difference could be made smaller, in fact arbitrarily small, by increasing the size of the training sequence. However, disk space limits and computation time limit the size of the training sequence.

In Figure 4.18 the spectrum of a voiced segment of speech¹⁰, and the optimum model filter¹¹ of order $p = 20$ is shown. Figure 4.19 shows the same spectra, but now quantized with a 10-10 product code. Each stage utilized a 32-32 tree search.

As a second technique we choose vector quantization of each Line Spectrum Pair (LSP). We quantize each LSP ($p/2$ in total) in a B , level vector quantizer. We found that by using $p = 10$ and 256 levels for each VQ, good performance was maintained with respect to the unquantized case. With $p = 10$ this requires 40 bits to quantize the model filter. This would require 4 kbps for a window size of 80 samples. However, we determined that if we

¹⁰Using a 160 sample speech buffer, hamming windowed, and a 1024 point FFT

¹¹Again computed with a 1024 point FFT.

quantize the model filter every 160 samples and linear interpolate the LSPs between frames, very good performance was obtained. No unstable frames (due to LSP quantization) were observed over 240 seconds of the speech data base. Thus, we require a total rate of only 2 kbps for the short term model filter. Figure 4.20 shows the same speech spectra (as above) and the quantized model filter using the LSPs.

The average gain over a 20 second segment of speech using LSP quantization and the 10-10 product code based on the I-distortion is 12.6 dB and 12.0 dB respectively. The LSP quantization offers better performance at a somewhat higher rate.

The 40 LSP product code is adequate for our purposes. Although the performance of the 10-10 product code outside training sequence is fairly poor, we believe that the structure is sound, and a better training sequence could improve the performance for data outside the training sequence. The data rate for the model filter ($A(z)$) with a window size of 160 samples is thus 2 kbps. Recall that we update the model filter every 80 sample by linear interpolation between adjacent frames.

Figure 4.16: Distortion Performance Inside The Training Sequence

Figure 4.17: Distortion Performance Outside The Training Sequence

Figure 4.18: Speech Spectra, Optimum Model Spectra ($p = 20$)

Figure 4.19: Speech/Model Spectra, 20 bit product code ($p = 20$)

Figure 4.20: Speech/Model Spectra, 40 bit LSP ($p = 10$)

Chapter 5

Optimization Of CELP

5.1 CELP Complexity

Although the reconstructed speech quality is excellent, the computational complexity of CELP is enormous. For the original CELP design (Section 3.3.3) we have the following parameters:

- codebook size: $L = 2^{10}$;
- codebook dimension: $K = 40$;
- long term predictor ($P_l(z)$) with $2q + 1$ coefficients, ($q = 1$);
- short term predictor ($P_s(z)$) with p coefficients, ($p = 20$);
- a noise weighting filter $W(z) = A(z)/A(z/\gamma)$, $\gamma = 0.73$.

The complexity in FLOPV (floating point operations per codebook vector) is:

- $K(2q + 1)$, filtering through the long term predictor;
- Kp , filtering through the short term predictor;

- $2Kp$, weighting filter;
- K , magnitude square.

Thus, the complexity (computational) with f_s/K vectors per second and codebook size L is:

$$\begin{aligned} C &= KL(3p + 2q + 2)f_s/K \text{ FLOPS} \\ &\approx 3pLf_s \text{ FLOPS} \end{aligned}$$

For $p = 20$, $f_s = 8 \text{ kHz}$, and $L = 2^{10}$ we get

$$C \approx 500 \text{ MFLOPS}$$

Clearly, this has no hope of a real time implementation in the near future, given the present structure.

The majority of the computational complexity is due to the filtering through the noise weighting filter and the short term predictor. Also, techniques to reduce the search complexity (a binary tree search for example) would greatly reduce the computational load. Note that the complexity C is the complexity of the search for the optimum excitation vector and does not encompass the computation of the short and long term predictors. However, these computations are insignificant compared to the search complexity, with the present structure.

We note that a complexity of approximately 16 MFLOPS is attainable using currently available DSP (digital signal processing) devices. The TMS32030 DSP can, for example, operate at a maximum throughput of 16 MFLOPS,¹ with an instruction rate of 16 MIPS, (Mega-instructions per second). The throughput of 16 MFLOPS is an upper limit and does not include the memory operations, and other logical instructions which

¹Note that Texas Instruments, the supplier for the TMS32030, has a slightly different definition of a FLOP.

would undoubtedly be necessary in a real time implementation. Some floating point operations (for example divides) are typically quite cumbersome. Conversely, vector operations (inner products) execute at the full rate (16 MFLOPS).

Experience has shown, that a factor of 5-10 in overhead is required to maintain a real time implementation, depending on the quantity of divides, memory operations and so on.

Including the overhead, a complexity of 1-3 MFLOPS is implementable using currently available DSP devices. A less complex search procedure is required which will permit a real-time implementation. A simple codebook structure, suitable for a VLSI implementation or a commercially available DSP device, should be maintained in the simplified search structure.

In the next section, we detail methods which attempt to obtain a simplified search procedure with the above goals in mind.

5.2 Reduced Complexity CELP

Factors of twenty (or so) in the computational complexity can be realized with very little or no degradation as outlined in [33,32]. However, we require factors of about 500 if we wish to implement the algorithm utilizing currently available DSP devices. In this section we modify the original CELP structure to arrive at two structures which greatly reduce the complexity and are amenable to fast search, non-exhaustive codebook searches.

One must be extremely careful, however, when modifying the structure of Figure 3.17 since time invariance in the synthesis filters does not hold. Nevertheless, there are a number of extremely simple techniques to reduce the complexity of the CELP search procedure.

First, we note that we may move the location of $W(z)$ (given by Equa-

tion 3.8) as shown in Figure 5.1.

We note that there are two components to $\hat{w}_n$, the zero input response and the zero state response. The former is due to the inherent memory in the short and long term predictors and can be subtracted out prior to beginning the exhaustive search. That is, the filter memory is due to the previous codevectors and is independent of the search for the current vector. Once this is accomplished, the filter memory can be set to zero with no loss in search optimality. Furthermore, if we can assume the pitch period is greater than the vector dimension ($M > K + q$), only the memory inherent in the pitch filter need be used. That is, the zero state response is equal to unity. Recall that significant long term prediction gains can be realized by using a delay which is an integral number of pitch periods, [5]. Thus, imposing this limit on the pitch period should not degrade the performance significantly.

Using these previous ideas and observations we obtain the following block diagram shown in Figure 5.2.

Note that interchanging the order of G and $1/A(z/\gamma)$ is now valid since we are dealing with the zero state response only. Now, we minimize the mean squared error between x_n and $G\hat{x}_n$. That is

$$\epsilon = \sum_{n=0}^{K-1} (x_n - G\hat{x}_n)^2 \quad (5.1)$$

To minimize, we compute

$$\begin{aligned} \frac{\partial \epsilon}{\partial G} &= 0 \\ &= 2 \sum_{n=0}^{K-1} \hat{x}_n (x_n - G\hat{x}_n) \end{aligned}$$

Thus

Figure 5.1: CELP, Location of $W(z)$ Moved

Figure 5.2: CELP, Separate Zero-Input, Zero-State Responses

$$G = \frac{\sum_{n=0}^{K-1} x_n \hat{x}_n}{\sum_{n=0}^{K-1} \hat{x}_n^2} \quad (5.2)$$

and

$$\epsilon = \sum_{n=0}^{K-1} x_n^2 - 2G \sum_{n=0}^{K-1} \hat{x}_n x_n + G^2 \sum_{n=0}^{K-1} \hat{x}_n^2 \quad (5.3)$$

$$= \sum_{n=0}^{K-1} x_n^2 - \frac{\left(\sum_{n=0}^{K-1} \hat{x}_n x_n \right)^2}{\sum_{n=0}^{K-1} \hat{x}_n^2} \quad (5.4)$$

The search for the optimum codevector from the L level codebook attempts to minimize the error between a block of samples of x_n and $\hat{x}_n$. Codevectors $v_n^{(l)}$ (for $l = 0$ to $L - 1$, where $v_n^{(l)}$ is the l^{th} codevector out of the codebook) are filtered through the weighted zero state synthesis filter ($1/A(z/\gamma)$), to obtain the weighted error signal $\hat{x}_n^{(l)}$. The difference between x_n and $G\hat{x}_n^{(l)}$ is minimized over a block of K samples, where G is computed as above (Equation 5.2). Once the optimum codevector is found, it is scaled by G and filtered through the two zero input filters, which have been reset to their initial memories at the beginning of the block. The search for the optimum codevector for the next block of samples then proceeds.

Instead of designing the codebook for the v_n , if we design it for the $\hat{x}_n$ we can place the codevectors on the surface of a K dimensional sphere. That is, we normalize the codevectors, such that

$$\sum_{n=0}^{K-1} \hat{x}_n^2 = 1 \quad (5.5)$$

Since the first term in Equation 5.4 is independent of the current codevector, minimization of Equation 5.4 is equivalent to maximizing

$$\frac{\left(\sum_{n=0}^{K-1} x_n \hat{x}_n\right)^2}{\sum_{n=0}^{K-1} \hat{x}_n^2}$$

However, since the codevectors are normalized, the optimum codevector is calculated by maximizing

$$G = \sum_{n=0}^{K-1} x_n \hat{x}_n^{(l)} \quad (5.6)$$

for $l = 0$ to $L - 1$.

The search procedure is outlined in Figure 5.3. Enough modifications have been made in order to give this structure a new name. We refer to this structure as KELP, for no reason in particular. Note that moving the codebook forces us into a new synthesis structure. At the receiver, we have a copy of the codebook. The synthesis filters are transmitted over the channel. Thus, to get the unweighted residual, we first scale the codevector by the gain, filter the codebook element with $A(z/\gamma)$ (zero state response only), and finally filter the unweighted residual through the two synthesis filters $1/B(z)$ and $1/A(z)$ (in respective order), as shown in Figure 5.4.

We immediately notice that by moving the codebook location that we have eliminated the filtering operations which must be performed on each codevector. The codebook structure is now just a Gain Shape Vector Quantizer (GSVQ), (Section 4.7). The question which remains is "will moving the codebook location degrade the performance?". The answer is a resounding *yes*, since $\hat{x}_n$ has a larger dynamic range than v_n and a time varying power spectrum. However, there is no formant or pitch structure in x_n since $A(z/\gamma)$ is relatively flat. The spectrogram for a segment of speech, that is the magnitude response of $1/A(z)$ at time intervals of 5 milli-seconds, is shown in Figure 5.5. The segment consists of a transition

Figure 5.3: KELP, Move Location of Codebook

Figure 5.4: KELP Synthesis Structure

from mildly voiced, to unvoiced, and back to voiced. Thus, there is considerable spectral (and time domain) variation. The spectrogram of $1/A(z/\gamma)$ is shown in Figure 5.6. Note that the latter spectrogram is relatively flat and consistent, thus implying that moving the codebook should not degrade the performance significantly. Furthermore, in [35] the noise weighting filter was modified such that

$$\hat{W}(z) = A(z)/C(z/\gamma) \quad (5.7)$$

where $C(z/\gamma)$ is a fixed filter based on the long term speech statistics. They determined that by using this fixed filter in the denominator that little or no degradation was incurred over the variable filter. By utilizing the fixed filter the original CELP structure can be modified into the KELP structure with absolutely no degradation. The codebook, in the KELP structure, could be designed by simply passing the original codebook through the fixed filter $(1/C(z/\gamma))$. The KELP structure with a fixed weighting filter is shown in Figure 5.7. The KELP structure, whether using a fixed or variable weighted synthesis filter, is expected to perform quite well. Note that the two search procedures, based on a fixed or variable weighted synthesizer, are identical, and have the same computational complexity.

The complexity of the search procedure in Figures 5.3 and 5.7 is given by

$$C = f_s L \text{ FLOPS}$$

The reduction in the complexity over the original CELP design is $3p$. The factor of 3 is due to moving the location of $W(z)$ and the factor of p is due to moving the location of the codebook.

Furthermore, with the search structured as in Figure 5.3 or Figure 5.7, we may use a non-exhaustive search procedure such as a binary, or non-binary tree search. Note that with the original CELP design a tree searched

Figure 5.5: Spectrogram, Voiced Segment, $1/A(z)$

Figure 5.6: Spectrogram, Voiced Segment, $1/A(z/\gamma)$

Figure 5.7: KELP, Fixed Weighting Filter

procedure is not so simple since the codevectors are filtered through a time varying filter before computing the mean squared error.

The complexity of the binary tree searched KELP is

$$\begin{aligned} C &= 2f_s \log_2(L) \\ &\approx 160 \text{ KFLOPS, with } L = 2^{10} \end{aligned}$$

which is a factor of approximately 3000 over the original CELP design, with a 100% increase in memory requirements.

A 32-32 tree search offers some advantages over a binary tree search, (less memory, lower distortion) with a complexity (with $L = 2^{10}$) given by

$$\begin{aligned} C &= 2f_s 32 \\ &\approx 500 \text{ KFLOPS} \end{aligned}$$

which is a factor of 1000 less computationally complex as compared to original CELP with only a 3% increase in memory requirements. This complexity, (0.5 MFLOPS) can be easily reached with currently available DSP devices. The majority of these DSP devices are designed to perform inner products (Equation 5.6) extremely fast which ensures that the real time implementation will be possible.

In the next section we discuss methods to design the codebook described by either Figure 5.3 or Figure 5.7.

5.3 Codebook Design

In the original CELP design, the codebook was constructed from samples of white gaussian noise, and from a codebook of random residual vectors, each of which provided equivalent perceptual quality. Although a random codebook approaches the rate-distortion bound as the vector dimension approaches infinity, the performance using relatively small codebooks has been

shown to behave quite poorly, and often worse than binary tree searched codebooks, [11]. A gaussian codebook was expected to work quite well, since experiments with APC had shown that the residual was approximately gaussian after short and long term prediction with $A(z)$ and $B(z)$ respectively [26,31].

Schroeder and Atal had commented, in [31], that the codebook should be designed to optimally place the codevectors on the surface of a K -dimensional hypersphere. For a mean squared error distortion (that is $\gamma = 1.0$) and uniform distribution of the residuals over the surface of the sphere a perfect binary code (K -dimensional, rate $\frac{K}{2}$) would be optimum in terms of minimizing the mean squared quantization error. However perfect codes are not so easy to find, the residuals are not necessarily distributed uniformly over the surface, and the noise weighting is not flat. Thus, a perfect code would not be necessarily optimum, even if it could be found. A perfect code would, however, be expected to perform better than the random gaussian codes, since the surface would be more uniformly mapped out by the perfect code. That is, since the minimum distance between all codevectors is the same for a perfect code, it would offer distortion improvements over the random gaussian code. The binary code would also offer decreased memory requirements since only binary codevectors would need to be stored. The filtering complexity would also be decreased since only adds and subtracts are required in the filtering operation of the binary codevectors. However, many popular signal processors perform multiplies as quickly as adds and subtracts and thus filtering with binary elements does not offer any advantage. However, for a VLSI implementation, the binary codevectors and the structure of Figure 5.2 could result in a manageable system in terms of memory and computational requirements, if fast binary multiply-adds could be implemented efficiently.

Superior performance, based on a training sequence of representative speech, could be realized using the techniques of vector quantization. Three techniques could be utilized to design the codebook based on the structure of Figure 5.3, (and also applicable to Figure 5.7).

The first proposed technique assumes (like original CELP) that the residual is essentially white gaussian. A data base is then constructed by passing random gaussian vectors through a large number of random weighting filters ($1/A(z/\gamma)$). Once a large number of vectors have been stored, a gain shape vector quantizer codebook is designed using these representative vectors.

The second technique uses no quantization in Figure 5.3 (that is $x_n = \hat{x}_n$) to generate a database of representative x_n s. A GSVQ is then designed based on this database. We note that this method is not optimum since x_n consists partly of past quantized residuals. That is, the database should change each time a new codebook is constructed.

The final codebook design technique utilizes an original codebook, designed using one of the above techniques, and the structure of Figure 5.3 to quantize a segment of speech. The x_n vectors are then utilized to update the codebook using VQ techniques (GSVQ). Note that this final method is a (closed loop) iterative method whereby a new database of representative codevectors is generated for each new codebook. Furthermore, it is quite computationally complex.

This final technique, along with the second technique (for the initial codebook), was utilized to design the codebooks.

5.4 Pitch Parameter Determination

We have not, thus far, discussed in detail the computation of the pitch predictor parameters. The structure we have developed (Figure 5.2 or Figure 5.3) lends itself to an efficient method of calculating the pitch parameters ($B(z)$).

Referring to Figure 5.1 we note that the zero state response of $1/B(z)$ does not depend on the current codevector in the search but only on the past quantized residuals since we force $M > K + q$. It was noted previously that this should not degrade the performance since good prediction gains can be made using a delay which is a multiple of the actual pitch period. The all pole long term model filter is given by

$$\frac{1}{B(z)} = \frac{1}{1 - P_l(z)}$$

where

$$P_l(z) = \sum_{k=-q}^q b_k z^{-(M+k)}$$

and has the structure shown in Figure 5.8, for a three coefficient predictor ($q = 1$). From the figure, it is clear why the zero state response is unity (for a duration of at least K) if the pitch period (M) is greater than $K + q$.

The pitch period M , and the predictor parameters (b_k s) are calculated by minimizing the mean squared *closed* loop prediction error (e_n). The zero input response of the long term model filter is just $\tilde{d}_n$, from Figure 5.2. Note that in the absence of quantization, the input (at the end of the search) to the zero input response long term model filter is just the residual e_n . That is, the filter memory is just d_n . Thus, in the absence of quantization, $\hat{d}_{n-(M+k)} = d_{n-(M+k)}$, for $M > K + q$, and the closed loop and open loop structure is equivalent, as it should be. In other words, the inverse

Figure 5.8: The Long Term Model Filter, $1/B(z)$

model filter ($B(z)$) can be implemented in any one of two ways as shown in Figure 5.9.

In the presence of quantization, the closed loop structure is used, and quantized residuals ($\hat{e}_n$) are used to update the zero input response of $1/B(z)$. Thus, the residual energy, e_n , should be minimized based on past quantized samples of $\tilde{d}_n$. That is, by computing the pitch period and predictor parameters based on d_n , and not $\tilde{d}_n$, we lose optimality since $\hat{e}_n$ is filtered through $1/B(z)$ in the synthesis structure.

Thus, we minimize the residual energy in e_n (E_q), for a predictor with $2q + 1$ non-zero coefficients

$$E_q = \sum_{n=0}^{K-1} (d_n - \tilde{d}_n)^2 \quad (5.8)$$

over one analysis interval of length K where

$$\tilde{d}_n = \sum_{k=-q}^q b_k \hat{d}_{n-(M+k)} \quad (5.9)$$

is the zero input response of $1/B(z)$.

Note that the buffer contains past quantized residuals ($\hat{d}_{n-(M+k)}$) and not just past zero input residuals ($\tilde{d}_{n-(M+k)}$) since the filter memory is continuously updated, at the end of every search.

Substitution of Equation 5.9 into Equation 5.8 yields

$$\begin{aligned} E_q &= \sum_{n=0}^{K-1} \left(d_n - \sum_{k=-q}^q b_k \hat{d}_{n-(M+k)} \right)^2 \\ &= \sum_{n=0}^{K-1} d_n^2 - 2 \sum_{k=-q}^q b_k \sum_{n=0}^{K-1} d_n \hat{d}_{n-(M+k)} \\ &\quad + \sum_{n=0}^{K-1} \left(\sum_{k=-q}^q b_k \hat{d}_{n-(M+k)} \right)^2 \\ &= \sum_{n=0}^{K-1} d_n^2 - 2 \sum_{k=-q}^q b_k \sum_{n=0}^{K-1} d_n \hat{d}_{n-(M+k)} \end{aligned} \quad (5.10)$$

Open Loop Structure

Closed Loop Structure

Figure 5.9: Implementing The Long Term Inverse Filter, $B(z)$

$$\begin{aligned}
& + \sum_{k=-q}^q b_k \sum_{j=-q}^q b_j \sum_{n=0}^{K-1} \hat{d}_{n-(M+k)} \hat{d}_{n-(M+j)} \\
& = D - 2\mathbf{b}^T \boldsymbol{\psi} + \mathbf{b}^T \boldsymbol{\Psi} \mathbf{b}
\end{aligned} \tag{5.11}$$

where

$$D = \sum_{n=0}^{K-1} d_n^2 \tag{5.12}$$

is the residual energy in d_n ,

$$\psi_i = \sum_{n=0}^{K-1} \hat{d}_n \hat{d}_{n-(M+i)} \quad -q \leq i \leq q \tag{5.13}$$

and

$$\Psi_{ij} = \sum_{n=0}^{K-1} \hat{d}_{n-(M+i)} \hat{d}_{n-(M+j)} \quad -q \leq i, j \leq q \tag{5.14}$$

Minimization of Equation 5.10 for the unknown pitch period and pitch predictor parameters leads to a non-linear solution for the $2(q+1)$ unknowns. Instead, the pitch period is optimized for a one coefficient predictor ($q=0$) and the linear equations (for the given M) are then calculated.

For $q=0$, we can rewrite Equation 5.10 as

$$E_0 = \sum_{n=0}^{K-1} d_n^2 - 2b_0 \sum_{n=0}^{K-1} d_n \hat{d}_{n-M} + b_0^2 \sum_{n=0}^{K-1} \hat{d}_{n-M}^2 \tag{5.15}$$

Minimization with respect to b_0 yields

$$\begin{aligned}
-\frac{1}{2} \frac{\partial E_0}{\partial b_0} &= 0 \\
&= \sum_{n=0}^{K-1} d_n \hat{d}_{n-M} - b_0 \sum_{n=0}^{K-1} \hat{d}_{n-M}^2
\end{aligned}$$

Therefore

$$\sum_{n=0}^{K-1} d_n \hat{d}_{n-M} = b_0 \sum_{n=0}^{K-1} \hat{d}_{n-M}^2 \tag{5.16}$$

Solving for the unknown b_0 leads to

$$b_0 = \frac{\sum_{n=0}^{K-1} d_n \hat{d}_{n-M}}{\sum_{n=0}^{K-1} \hat{d}_{n-M}^2} \quad (5.17)$$

Substitution of Equations 5.16 and 5.17 into Equation 5.15 yields

$$E_0 = \sum_{n=0}^{K-1} d_n^2 - \frac{\left(\sum_{n=0}^{K-1} d_n \hat{d}_{n-M} \right)^2}{\sum_{n=0}^{K-1} \hat{d}_{n-M}^2} \quad (5.18)$$

Since the first summation in Equation 5.18 does not depend on the pitch period M , the second term is maximized over all M of interest, to find the optimum pitch period ($q = 0$). Note that the divides are not really necessary since we really only need M and not the actual value of the division. That is

$$\frac{n_1}{d_1} > \frac{n_2}{d_2} \implies n_1 d_2 > n_2 d_1$$

Once the pitch period is calculated, we minimize Equation 5.10 for the given pitch period. We compute

$$\begin{aligned} -\frac{1}{2} \frac{\partial E_q}{\partial b_i} &= 0 \\ &= \sum_{n=0}^{K-1} d_n \hat{d}_{n-(M+i)} - \sum_{k=-q}^q b_k \sum_{n=0}^{K-1} \hat{d}_{n-(M+k)} \hat{d}_{n-(M+i)} \end{aligned}$$

for $-q \leq i \leq q$. Rearranging, we obtain

$$\sum_{n=0}^{K-1} d_n \hat{d}_{n-(M+i)} = \sum_{k=-q}^q b_k \sum_{n=0}^{K-1} \hat{d}_{n-(M+k)} \hat{d}_{n-(M+i)} \quad -q \leq i \leq q \quad (5.19)$$

or

$$\Psi \mathbf{b} = \boldsymbol{\psi} \quad (5.20)$$

The solution for the optimum predictor parameters $\mathbf{b}$ can be efficiently computed using Choleski Décomposition, described in Appendix B. For the optimum parameters we note that

$$\begin{aligned} E_q &= D - \mathbf{b}^T \boldsymbol{\psi} \\ &= D - \mathbf{b}^T \Psi \mathbf{b} \end{aligned} \quad (5.21)$$

but in general the residual energy is given by Equation 5.11. Equation 5.21 is the minimum prediction error over all predictors with $2q + 1$ non-zero coefficients, and pitch period M .

We require a distortion measure and a centroid calculation for the long term model filter for use in a vector quantization context.

Equation 5.11 is a suitable distortion measure, mean squared residual energy. Since D in Equation 5.11 does not depend on the model filter from the codebook, minimizing Equation 5.11 is equivalent to minimizing

$$d_{IFM} = -2\mathbf{b}^T \boldsymbol{\psi} + \mathbf{b}^T \Psi \mathbf{b} \quad (5.22)$$

which can be simplified, slightly, since Ψ is symmetric. The distortion measure is called d_{IFM} since it is based on finding the inverse filter which results in the minimum prediction residual. That is, it is an inverse filter match, similar to that discussed in Section 2.4.3 in the context of the short term model filter.

The centroid calculation is performed by minimizing the average distortion, $\bar{E}$ over all training vectors in cell C_i . We minimize

$$\begin{aligned} \bar{E} &= \sum_{k \in C_i} (D^{(k)} - 2\bar{\mathbf{b}}_i^T \boldsymbol{\psi}^{(k)} + \bar{\mathbf{b}}_i^T \Psi^{(k)} \bar{\mathbf{b}}_i) \\ &= \sum_{k \in C_i} D^{(k)} - 2\bar{\mathbf{b}}_i^T \left(\sum_{k \in C_i} \boldsymbol{\psi}^{(k)} \right) + \bar{\mathbf{b}}_i^T \left(\sum_{k \in C_i} \Psi^{(k)} \right) \bar{\mathbf{b}}_i \end{aligned}$$

where $\bar{b}_i$ is the centroid of cell C_i , and $\psi^{(k)}$ and $\Psi^{(k)}$ are the covariance vectors and matrices (respectively) of the k^{th} training sequence element in cell C_i . From Equation 5.10, the distortion is easily minimized by performing Choleski Decomposition using the average covariance vector and matrix in cell C_i . That is, $\bar{b}_i$ is the solution to

$$\bar{\Psi}_i \bar{b}_i = \bar{\psi}$$

where

$$\bar{\Psi}_i = \sum_{k \in C_i} \Psi^{(k)}$$

and

$$\bar{\psi} = \sum_{k \in C_i} \psi^{(k)}$$

The complexity of each distortion computation, (Equation 5.22) is

$$C = (2q + 1)^2 + 2(2q + 1)$$

The total complexity of the vector quantizer in FLOPS is just

$$C = \left[(2q + 1)^2 + 2(2q + 1) \right] Lf_s / K$$

$$\approx 0.77 \text{ MFLOPS}$$

for $q = 1$, $L = 256$, $K = 40$, and $f_s = 8 \text{ kHz}$.

The complexity does not include the computation of the covariance matrix and vector, or the pitch period. The above complexity can be reduced by using a tree search, and/ or exploiting properties of symmetric matrices in the computation of Equation 5.22. A smaller codebook would also speed computation at the expense of a decrease in performance. Since the pitch period is typically in the range of 2.5 to 20 msec, (or 20-160 samples), we require approximately seven bits of accuracy to quantize the pitch period. We also must restrict the pitch period to be greater than $K + q$. Thus, the range of values of M in Equation 5.18 is

$$K + q < M \leq K + q + 2^{B_M}$$

where B_M is the number of bits used to quantize the pitch period M . Assuming B_b bits to quantize the predictor parameters (the b_k s), the total bit rate for the long term model filter is just

$$R = B_M f_s / K + B_b f_s / K$$

Assuming an 8 bit codebook ($B_b = 8$, $q = 1$), 7 bit pitch period quantization, and a vector dimension of $K = 40$, then

$$41 < M < 170$$

and

$$R = 3 \text{ kbps}$$

In the next section we look at the final, and perhaps easiest parameter to quantize, the gain G .

5.5 Gain Quantization

The gain term is usually quantized on a log scale, or can be quantized using the IS-distortion between the optimum model gain and the quantized gain, (Equation 2.45). It can be shown that both methods are approximately equal.

Recall

$$d_{IS}(\alpha, \sigma^2) = \alpha / \sigma^2 - \ln(\alpha / \sigma^2) - 1$$

where $\sqrt{\alpha}$ is the optimum model gain of the quantized model filter and σ is the quantized value of α . In our current notation, we denote G as the

optimum gain and $\hat{G}$ as the quantized value. We note that

$$\begin{aligned}\alpha/\sigma^2 &= \exp(\ln(\alpha/\sigma^2)) \\ &= 1 + \ln(\alpha/\sigma^2) + \frac{1}{2} [\ln(\alpha/\sigma^2)]^2 + \dots\end{aligned}$$

For small distortions ($\alpha/\sigma^2 \approx 1$), the gain distortion is therefore

$$\begin{aligned}d_{IS}(\alpha, \sigma^2) &\approx \frac{1}{2} [\ln(\alpha) - \ln(\sigma^2)]^2 \\ &\approx 2 [\ln(\sqrt{\alpha}) - \ln(\sigma)]^2\end{aligned}$$

The gain optimized IS-distortion is therefore also quantizing the gain on an approximate log scale².

For these above reasons, quantizing on a log scale and using a mean squared error distortion measure seems appropriate.

A Lloyd-Max scalar quantizer is used to design the gain codebook³. Note that since the logarithmic function is monotonic, by storing the non-log decision boundaries *and* centroids the computation of logarithms is not necessary. Furthermore, for a scalar quantizer, a binary tree search is an optimum search procedure. For example, consider for a moment a simple 3-level uniform scalar gain quantizer, based on a log-mse distortion, with the decision regions and centroids shown in Figure 5.10. Assuming a base 10 logarithm, the non-log centroids and decision regions are shown in Figure 5.11.

To quantize the gain, on a log-scale, we would simply use the decision regions and centroids in the lower figure, (Figure 5.11). Quantization is performed on a log scale, but the requirements for logarithms has been eliminated, except in the design process. Note that the memory requirement is slightly higher since the centroids and cell boundaries *must* be

²Note that the base of the logarithm is immaterial.

³The Lloyd-Max scalar quantizer design algorithm is simply a VQ design algorithm (LBG) with a vector dimension of unity.

Figure 5.10: Decision Regions, Log Scale

Figure 5.11: Decision Regions, Non-Log Scale

stored. At the receiver, only the centroids are required. Note that this process is exactly equivalent to computing the log-gain and quantizing using logarithmic centroids.

The added memory requirement is insignificant since only five bits are typically needed to quantize the gain. Note that the gain must be recomputed after every codebook search. Thus, for a vector of dimension $K = 40$, we require 1 kbps for the transmission of the gain. For $K = 20$, 2 kbps is required.

Note that the gain terms are highly correlated from sample to sample. This is not utilized to decrease the bit rate further. This correlation could, however, be used in some manner to reduce the rate. In Section 6.2 we investigate this point further, with some degree of detail.

5.6 Development of The Speech Coder

In this section we use the modifications to the CELP structure developed in Section 5.2 to obtain a suitable voice codec. A few modifications are made to arrive at a suitable implementation.

We first required a full simulation of the original CELP structure. This will give us a benchmark on which to compare our modifications. Quantization of the long and short term model filters, and the gain term was not required. The Autocorrelation method (with $p = 20$, $N_p = 160$, $N = 80$, and a Hamming window) was used to obtain a suitable model filter, $(A(z))$. The above parameters were chosen to yield good prediction gains, (small residual error), over a long testing sequence. The parameters were also consistent with those chosen in the literature. The pitch predictor was discussed in Section 5.4. The pitch period was constrained to be in the range $K + q < M \leq K + q + 128$, for 7 bit pitch period quantization. The pitch

predictor structure we developed was believed to be superior to maximizing the open loop prediction residual. The predictor was optimized, given the structure we had developed. Instabilities in the pitch predictor with this design algorithm occur less frequently, and are believed to have less of an effect on the synthesized speech.

The codebook (for CELP) was constructed from white gaussian noise samples. We chose a vector dimension of $K = 40$, and a codebook size of $L = 2^{10} = 1024$.

The objective measure, consistent with our proposed codec, is the noise weighted signal to noise ratio. That is, the speech energy divided by the frequency weighted quantization noise energy.⁴ This measure is appropriate given the distortion measure we are aiming to minimize.

The simulation was developed on a SUN-3 operating under the UNIX operating system, with a floating point accelerator (fpa) installed.

The original CELP algorithm required approximately 8 hours of stand alone CPU time and obtained a noise weighted SNR (NWSNR) of 15.5 dB. The reconstructed speech quality was judged to be excellent. The difference in the perceptual quality between the original (sampled at 8 kHz and quantized using a 12 bit linear Analog to Digital Converter) and the reconstructed speech was hardly noticeable.

We re-emphasize the fact that the residual is encoded at 2 kbps since

$$R = \log_2(L) f_s / K$$

and that the gain, and model filters ($1/A(z)$ and $1/B(z)$) are left unquantized. We note that in any *practical* codec, the gain and model filters would have to be quantized.

As our first modification, we use the structure of Figure 5.3 and design the codebook as discussed in Section 5.3 using the closed loop design pro-

⁴The energy in $w_n - \hat{w}_n$ from Figure 5.1.

cess. We design a full search and a 32-32 tree search codebook. The full search design required approximately 12 hours of CPU time. A database of 240 seconds of speech (see Section 4.8) was used to design (train) the codebooks. The full search and tree searched KELP algorithm obtained NWSNRs of 15.3 and 14.8 dB respectively, on a test sequence *outside* of the training sequence. The test sequence consisted of a twenty second utterance, (10 seconds from a female speaker, and 10 seconds from a male speaker), and was the same segment in which the original CELP algorithm was tested. We immediately note that the CELP algorithm offers only slight advantages over the full search KELP algorithm. The 32-32 search obtains a performance penalty of only 0.80 dB (NWSNR) over the original CELP algorithm. The perceptual quality is still excellent. Thus, KELP with a 32-32 tree search, with a complexity of only 0.50 MFLOPS, is an excellent tradeoff between computational requirements and perceptual quality.

The gain and model filters have been left unquantized. However, the comparison between the original CELP and KELP is fair since the model parameters are computed in the same manner and left unquantized.

The long and short term model filters are quantized as discussed in Sections 5.4 and 4.8. The NWSNR after quantization of $A(z)$ (KELP, full search) decreased 1.1 dB to 14.2 dB. Note that the prediction order was decreased from $p = 20$ to $p = 10$. Quantization of $B(z)$ using an eight bit full search code results in a further decrease of 0.1 dB.

The gain is quantized on a log-mse scale to five bits. Gain quantization results in a loss of less than 0.1 dB in the NWSNR. After quantization of $A(z)$, $B(z)$, and the gain, the perceptual quality is still very good and just noticeably distorted.

The greatest degradation in speech quality and NWSNR was incurred upon the reduction in the order and the quantization of $A(z)$. Better quan-

tization techniques could be developed to maintain the excellent speech quality obtained without quantization of these parameters. However, the reconstructed speech quality is still very good.

A summary of the techniques tested, and the NWSNR for each is shown in Table 5.1, with the dash indicating an unquantized parameter.

Codec	Bitrate (kbps) for:				NWSNR (dB)
	v_n	G	$A(z)$	$B(z)$	
CELP	2.0	—	—	—	15.5
KELP (full search)	2.0	—	—	—	15.3
KELP (32-32 search)	2.0	—	—	—	14.8
KELP (full search)	2.0	—	2.0	—	14.2
KELP (full search)	2.0	1.0	2.0	3.0	14.1
KELP (32-32 search)	2.0	1.0	2.0	3.0	13.7

Table 5.1: Performance of The Various Configurations

Chapter 6

Summary and Suggestions For Future Research

6.1 Summary

In this study we have arrived at a structure for a very efficient and high quality speech codec. Vector Quantization was utilized to encode the short and long term predictors, the gain, and the residual excitation. Since the parameters are updated on a block by block basis, the structure is inherently suboptimal. However, this product code nature results in a system of manageable complexity.

We began with the Code-Excited Linear Prediction (CELP) structure and simplified it into a form which was amenable to fast, non-exhaustive search techniques. We obtained new methods to determine and quantize the long term model filter based on Cholesky Decomposition and the Inverse Filter Matching (IFM) principle. The predictor is computed based on past quantized values of the residual. A simple, and efficient method was devised to design the codebook based on the IFM principle which is applicable to

the IFM principle based on the short term predictor. Quantization of the gain was accomplished using a Lloyd-Max scalar quantizer and a standard (log-mse) distortion measure. We have obtained a complexity reduction by a factor of 1000 for the residual codebook search, while maintaining excellent speech quality and nearly the same memory requirements.

In quantizing the short term model filter we arrived at a new product code structure based on the Itakura distortion and Vector Quantization. It was determined that the performance outside the training sequence was substantially inferior to the performance obtained inside the training sequence. A larger training sequence was proposed to alleviate this problem to some degree. It is strongly believed, in retrospect, that twenty bit quantization is not adequate, with the chosen distortion measure. Weighting the input speech, to emphasize spectrally important regions, could enhance the performance somewhat. The LSP product code VQ performed well. The short term predictor was quantized every 20 msec, and recalculated every 10 msec by using linear interpolation between frames.

The reconstructed speech using the fully quantized version is not distortion free but has very good quality and is clearly intelligible. The reconstructed speech is not "synthetic sounding" as with LPC10 vocoders. The individual words are all easily recognizable with a small amount of background noise. It was believed that with better quantization of the model filter, excellent speech quality can be maintained at a bit rate of 8 kbps and lower.

In the next section we present suggestions for further research.

6.2 Suggestions For Future Research

As we have discussed earlier, the speech coder currently operates at a bit rate of 8 kbps. A number of enhancements could be made to lower the bit rate and/or improve the perceptual quality of the synthesized speech. After completing the above enhancements a suitable hardware implementation could be undertaken in VLSI or using a suitable DSP microchip.

The bit rate for the long term inverse filter ($B(z)$) is fairly large (3 kbps). The structure could be modified to reduce this rate while maintaining good quality. In the recent literature, the open loop gain is typically minimized in designing $B(z)$. This would lead to a lower bit rate requirement at a slight increase in the distortion.

It was also noted that the 20 bit cascaded VQ algorithm (to quantize the short term predictor) was also insufficient. By using a larger training sequence and/or an appropriate spectral weighting function improved performance could be realized. Furthermore, if memory is incorporated, such that future model filters depend on past model filters extremely efficient quantizers could result[41]. That is, since the model spectra change continuously and rather slowly, a differential type PCM system could be used to encode the LSP parameters, (or LARs). By utilizing this memory a more acceptable system could be realized. A differential system based on the I-distortion would be desirable due to the mathematical tractability of this distortion measure.

The gain also exhibits high sample to sample correlation. This correlation could also be used to decrease the bit rate further.

It should be emphasized, however, that the CELP structure is inherently suboptimal. It would be worthwhile to compare the performance of KELP (or CELP) to other systems of equal complexity. Investigation of the performance of tree codes, trellis codes, and finite state vector quantization

on the residual and on the original speech may lead to further improvement. Trellis codes appear to be promising since the coder states can serve in tracking the speech spectral properties. Furthermore, the M-algorithm can provide a greatly reduced computational complexity.

Testing and improvement of the codec over a wide range of speakers and in the presence of bit errors would also be necessary.

- A rather ambitious goal would be to implement the codec in VLSI or on a suitable DSP device.

Bibliography

- [1] C.E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, Chicago Illinois, 1949.
- [2] T. Berger, *Rate Distortion Theory*, Prentice Hall Inc., Englewood Cliffs, N.J., 1971.
- [3] L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice Hall Inc., Englewood Cliffs, N.J., 1978.
- [4] A. Oppenheim and R. Schafer, *Digital Signal Processing*, Prentice Hall Inc., Englewood Cliffs, N.J., 1975.
- [5] N.S. Jayant and P. Noll, *Digital Coding of Waveforms— With Applications to Speech and Video*, Prentice Hall Inc., Englewood Cliffs, N.J., 1984.
- [6] J.D. Markel and A.H. Gray Jr., *Linear Prediction of Speech*, Springer-Verlag, New York, New York, 1976.
- [7] J.G Proakis, *Digital Communications*, McGraw Hill, 1983.
- [8] J. Makhoul, "Linear Prediction: A Tutorial Review," *Proceedings of the IEEE*, Vol. 63, pages 561-580, April 1985.
- [9] N.S. Jayant, "Coding Speech at Low Bit Rates," *IEEE Spectrum*, August 1986.
- [10] J.L. Flanagan et. al., "Speech Coding," *IEEE Transactions on Communications*, Vol. COM-27, No. 4, April 1979.

- [11] J. Makhoul, S. Roucos, and H. Gish, "Vector Quantization in Speech Coding," *Proceedings of the IEEE*, Vol. 73, No. 11, November 1985.
- [12] R.M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, April 1984.
- [13] A. Gersho and V. Cuperman, "Vector Quantization— A Pattern Matching Technique For Speech Coding," *IEEE Communications Magazine*, December 1983.
- [14] Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, Vol. COM-28, No. 1, January 1980.
- [15] A. Buzo, A.H. Gray, and G. Rebolledo, "Speech Coding Based Upon Vector Quantization," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-28, No. 5, October 1980.
- [16] H. Abut, R.M. Gray, and G. Rebolledo, "Vector Quantization of Speech and Speech-Like Waveforms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-30, No. 3, June 1982.
- [17] M.J. Sabin and R.M. Gray, "Product Code Vector Quantizers for Waveform and Voice Coding," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-32, No. 3, June 1984.
- [18] M.J. Sabin and R.M. Gray, "Product Code Vector Quantizers for Speech Waveform Coding," *Globecom*, December 1982.
- [19] J. Chen and A. Gersho, "Gain Adaptive Vector Quantization for Medium Rate Speech Coding," *IEEE International Conference on Communications*, Chicago, June 1985.
- [20] R.M. Gray and H. Abut, "Full Search and Tree Searched Vector Quantization of Speech Waveforms," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Paris France, May 1982.
- [21] B.H. Juang and A.H. Gray Jr., "Multiple Stage Vector Quantization for Speech Coding," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Paris France, May 1982.

- [22] D.Y. Wong, B.H. Juang, and A.H. Gray, "An 800 BPS Vector Quantization LPC Vocoder," *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. ASSP-30, No. 5, October 1982.
- [23] B.H. Juang, D.Y. Wong, and A.H. Gray, "Distortion Performance of Vector Quantization for LPC Voice Coding," *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. ASSP-30, No. 2, April 1982.
- [24] Y. Matsuyama and R.M. Gray, "Voice Coding and Tree Encoding Speech Compression Systems Based Upon Inverse Filter Matching," *IEEE Transactions on Communications*, Vol. COM-30, No. 4, April 1982.
- [25] B.S. Atal and M.R. Schroeder, "Predictive Coding of Speech Signals and Subjective Error Criteria," *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. ASSP-27, No. 3, June 1979.
- [26] B.S. Atal, "Predictive Coding of Speech at Low Bit Rates," *IEEE Transactions on Communications*, Vol. COM-30, No. 4, April 1982.
- [27] B.S. Atal and J.R. Remde, "A New Model of LPC Excitation for Producing Natural Sounding Speech at Low Bit Rates," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Paris France, May 1982.
- [28] H. Koyama and A. Gersho, "Fully Vector Quantized Multipulse LPC at 4800 BPS," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [29] S. Singhal, "Reducing Computation in Optimal Amplitude Multipulse Coders," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [30] K. Ozawa and T. Araseki, "High Quality Multi-Pulse Speech Coder With Pitch Prediction," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.

- [31] M.R. Schroeder and B.S. Atal, "Code-Excited Linear Prediction (CELP): High Quality Speech at Very Low Bit Rates," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tampa Florida, March 1985.
- [32] I.M. Trancoso and B.S. Atal, "Efficient Procedures for Finding the Optimum Innovation in Stochastic Coders," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [33] G. Davidson and A. Gersho, "Complexity Reduction Methods for Vector Excitation Coding," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [34] B.S. Atal, "High Quality Speech at Low Bit Rates: Multipulse and Stochastically Excited Linear Predictive Coders," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [35] P. Kroon, R.J. Sluyter, and E.F. Deprettere, "A Low Complexity Regular Pulse Coding Scheme With a Reduced Transmission Delay," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [36] G. Ohyama, "A Novel Approach to Estimating Excitation Code in Code Excited Linear Prediction Coding," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [37] M. Copperi and D. Sereno, "CELP Coding for High Quality Speech at 8 kbps," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [38] L.A. Hernandez-Gomez et. al., "On the Behaviour of Reduced Complexity Code-Excited Linear Prediction (CELP)," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.

- [39] G. Davidson, M. Yong, and A. Gersho, "Real-Time Vector Excitation Coding of Speech at 4800 bps", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Dallas, Texas, 1987.
- [40] J. Chen and A. Gersho, "Real-Time Vector APC Speech Coding at 4800 bps with Adaptive Postfiltering" *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Dallas, Texas, 1987.
- [41] J. Chen, G. Davidson, and K. Zeger, "Speech Coding for the Mobile Satellite Experiment", *IEEE International Conference On Communications*, Seattle, Washington, 1987.
- [42] R.M. Gray et. al., "Distortion Measures for Speech Processing," *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. ASSP-28, No. 4, August 1980.
- [43] T. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. ASSP-23, No. 1, February 1975.
- [44] J.H. Chen and A. Gersho, "Vector Adaptive Predictive Coding of Speech at 9.6 kbps," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [45] V. Cuperman and A. Gersho, "Adaptive Differential Vector Coding of Speech," *Globecom*, December 1982.
- [46] E. Masgrau, J. Marino, and F. Vallverdu, "A Continuously Adaptive Vector Predictive Coder (AVPC) for Speech Encoding," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [47] L.C. Stewart, R.M. Gray, and Y. Linde, "The Design of Trellis Waveform Coders," *IEEE Transactions on Communications*, Vol. COM-30, No. 4, April 1982.
- [48] H.G. Fehn and P. Noll, "Multipath Search Coding of Stationary Signals with Applications to Speech", *IEEE Transactions on Communications*, Vol. COM-30, No. 4, April 1982.

- [49] B.H. Juang, "Design and Performance of Trellis Vector Quantizers for Speech Signals," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [50] M.O. Dunham and R.M. Gray, "An Algorithm for the Design of Labeled Transition Finite State Vector Quantizers," *IEEE Transactions on Communications*, Vol. COM-33, No. 1, January 1985.
- [51] V. Ramamoorthy and S. Abdul-Jabbar, "A Reduced Search Vector Quantizer for Speech," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tokyo Japan, 1986.
- [52] F. Soong and B. Juang, "Line Spectrum Pair (LSP) and Speech Data Compression," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1983.
- [53] G. Kang and L. Fransen, "Application of Line-Spectrum Pairs To Low-Bit-Rate Speech Encoders", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tampa Florida, 1985.
- [54] G. Kang and L. Fransen, "Experimentation with Synthesized Speech Generated From Line-Spectrum Pairs", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 4, April 1987.
- [55] P. Kabal and R. Ramachandran, "The Computation of Line Spectral Frequencies Using Chebyshev Polynomials", *IEEE Transactions On Acoustics Speech and Signal Processing*, Vol. ASSP-34, No. 6, December 1986.

Appendix A

The Autocorrelation Method

The Autocorrelation Method utilizes N new samples, $N_p - N$ old samples, and attempts to minimize the residual energy in the windowed speech vector. Typically, a hamming or hanning window (length N_p) is used to window the input speech vector. In Section 2.1 we showed that this leads to a matrix equation (Equation 2.21),

$$\mathbf{R}\mathbf{a} = \mathbf{r} \quad (\text{A.1})$$

where

$$\begin{aligned} R_{ij} &= r(|i - j|) \\ \mathbf{r}^T &= (r(1), r(2), \dots, r(p)) \\ \mathbf{a}^T &= (a_1, a_2, \dots, a_p) \end{aligned}$$

The residual energy (D_p), for a predictor of order p , is just,

$$D_p = r(0) - \mathbf{a}^T \mathbf{R} \mathbf{a} \quad (\text{A.2})$$

Note that this is the residual energy of the windowed speech buffer filtered through the predictor.

Levinson Recursion is used to solve the matrix equation, (Equation A.1). the algorithm computes the optimum predictors of order 1, 2, and so on until the optimum p^{th} order predictor is computed.

We let a_{mk} be the k^{th} coefficient of the m^{th} order optimum predictor. The algorithm is easily computed in two steps (see [7]):

- Step 1, Initialize

$$\begin{aligned}a_{11} &= r(1)/r(0) \\ D_0 &= r(0) \\ i &= 1\end{aligned}$$

- Step 2, Repeat until $i = P$

$$a_{ii} = \left[r(i) - \sum_{k=1}^{i-1} a_{i-1,k} r(i-k) \right] / D_{i-1} \quad (\text{A.3})$$

$$a_{i,k} = a_{(i-1)k} - a_{ii} a_{(i-1)(i-k)}, \quad 1 \leq k \leq i-1 \quad (\text{A.4})$$

$$D_i = (1 - a_{ii}^2) D_{i-1} \quad (\text{A.5})$$

$$i = i + 1 \quad (\text{A.6})$$

The p^{th} order optimum predictor is just:

$$a_{pi}, \quad 1 \leq i \leq p$$

The p intermediate coefficients, the a_{ii} s, are called the reflection coefficients (k_i). We may write (from Equation A.5):

$$D_p = D_0 \prod_{i=1}^p (1 - k_i^2)$$

We note that the predictor energy must always be greater than zero, which puts a restriction on the reflection coefficients. That is:

$$|k_i| \leq 1$$

for $1 \leq i \leq p$. This condition ensures the stability of the model filter, $(A(z))$.

The Lattice analysis and synthesis structure is based only on the reflection coefficients. For a development and discussion of the Lattice structure see [6].

Appendix B

The Covariance Method

Equations 2.25 and 2.26 utilize a block of N new samples and p previous samples to compute the optimum predictor of order p which minimizes the mean square prediction error over N samples. This method uses Cholesky Decomposition, or a stabilized version of Cholesky Decomposition to solve the matrix equation.

We have, from Equation 2.26

$$\Phi \mathbf{a} = \phi \quad (\text{B.1})$$

or

$$\sum_{k=1}^p a_k \phi_{ik} = \phi_{i0} \quad (\text{B.2})$$

for $1 \leq i \leq p$ where

$$\phi_{ik} = \sum_{n=-k}^{N-k-1} s_n s_{n+k-i} \quad (\text{B.3})$$

The predictor error, or residual energy for a predictor of order p is

$$\begin{aligned} D_p &= \sum_{n=0}^{N-1} s_n^2 - \sum_{k=1}^p a_k \sum_{n=0}^{N-1} s_n s_{n-k} \\ &= \phi_{00} - \mathbf{a}^T \phi \\ &= \phi_{00} - \mathbf{a}^T \Phi \mathbf{a} \end{aligned} \quad (\text{B.4})$$

If Φ is positive definite, then we can write

$$\Phi = LDU \quad (\text{B.5})$$

where L is a lower triangular matrix, D is a diagonal matrix, and U is an upper triangular matrix. Since Φ is symmetric, $\Phi = \Phi^T$. Thus

$$\begin{aligned} \Phi &= \Phi^T \\ &= LDU \\ &= (LDU)^T \\ &= U^T D L^T \\ &= VV^T \end{aligned} \quad (\text{B.6})$$

where

$$V = L\sqrt{D} \quad (\text{B.7})$$

Therefore, V is also a lower triangular matrix, and V^T is upper triangular.

For a fourth order predictor, for example, we have $\Phi a = \phi$, or

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} & \phi_{14} \\ \phi_{21} & \phi_{22} & \phi_{23} & \phi_{24} \\ \phi_{31} & \phi_{32} & \phi_{33} & \phi_{34} \\ \phi_{41} & \phi_{42} & \phi_{43} & \phi_{44} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} \phi_{10} \\ \phi_{20} \\ \phi_{30} \\ \phi_{40} \end{bmatrix}$$

and $\Phi = VV^T$

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} & \phi_{14} \\ \phi_{21} & \phi_{22} & \phi_{23} & \phi_{24} \\ \phi_{31} & \phi_{32} & \phi_{33} & \phi_{34} \\ \phi_{41} & \phi_{42} & \phi_{43} & \phi_{44} \end{bmatrix} = \begin{bmatrix} v_{11} & 0 & 0 & 0 \\ v_{21} & v_{22} & 0 & 0 \\ v_{31} & v_{32} & v_{33} & 0 \\ v_{41} & v_{42} & v_{43} & v_{44} \end{bmatrix} \begin{bmatrix} v_{11} & v_{21} & v_{31} & v_{41} \\ 0 & v_{22} & v_{32} & v_{42} \\ 0 & 0 & v_{33} & v_{43} \\ 0 & 0 & 0 & v_{44} \end{bmatrix}$$

From the example, we can see that in general

$$\begin{aligned} \phi_{ij} &= \sum_{k=1}^j v_{ik} v_{jk}, \quad 1 \leq j \leq i \\ &= v_{ij} v_{jj} + \sum_{k=1}^{j-1} v_{ik} v_{jk} \end{aligned} \quad (\text{B.8})$$

Thus

$$\phi_{jj} = v_{jj}^2 + \sum_{k=1}^{j-1} v_{jk}^2 \quad (\text{B.9})$$

or

$$v_{jj}^2 = \phi_{jj} - v_{jj}^2 + \sum_{k=1}^{j-1} v_{jk}^2 \quad (\text{B.10})$$

From Equation B.8 we can write

$$v_{ij} = \left[\phi_{ij} - \sum_{k=1}^{j-1} v_{ik} v_{jk} \right] / v_{jj} \quad (\text{B.11})$$

for $1 \leq j \leq i-1$.

Typically v_{11} is first computed, followed by v_{21} , v_{31} , v_{41} and so on. That is, the columns of v are computed in order, from top to bottom.

Now to solve

$$\begin{aligned} \Phi a &= \phi \\ &= VV^T a \end{aligned}$$

we solve

$$Vy = \phi$$

where

$$y = V^T a$$

and then solve

$$V^T a = y$$

From our previous fourth order example we have

$$\begin{bmatrix} v_{11} & 0 & 0 & 0 \\ v_{21} & v_{22} & 0 & 0 \\ v_{31} & v_{32} & v_{33} & 0 \\ v_{41} & v_{42} & v_{43} & v_{44} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \phi_{10} \\ \phi_{20} \\ \phi_{30} \\ \phi_{40} \end{bmatrix}$$

and

$$\begin{bmatrix} v_{11} & v_{21} & v_{31} & v_{41} \\ 0 & v_{22} & v_{32} & v_{42} \\ 0 & 0 & v_{33} & v_{43} \\ 0 & 0 & 0 & v_{44} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

The column vector $\mathbf{y}$ is given by

$$y_i = \left[\phi_{i0} - \sum_{j=1}^{i-1} v_{ij} y_j \right] / v_{ii} \quad (\text{B.12})$$

for $i = 1, 2, \dots, p$ and the column vector $\mathbf{a}$, which depends on the computed vector $\mathbf{y}$ is

$$a_i = \left[y_i - \sum_{j=i+1}^p v_{ji} a_j \right] / v_{ii} \quad (\text{B.13})$$

for $i = p, p-1, p-2, \dots, 1$.

Note that the vector $\mathbf{a}$ is computed in reverse order, and $\mathbf{y}$ is computed in natural (increasing) order. A minimum of p square roots, and p divides are required; typically $1/v_{ii}$ is calculated before Equations B.12 and B.13 are processed.

Note that the stability of the resulting predictor is not ensured. By modifying the Cholesky Decomposition we can ensure the stability of the resulting predictor.

We have, from Equation B.4

$$\begin{aligned} D_p &= \phi_{00} - \mathbf{a}^T \Phi \mathbf{a} \\ &= \phi_{00} - \mathbf{a}^T \mathbf{V} \mathbf{V}^T \mathbf{a} \\ &= \phi_{00} - \mathbf{y}^T \mathbf{y} \\ &= \phi_{00} - \sum_{i=1}^p y_i^2 \end{aligned}$$

Thus, we may write

$$D_p = D_{p-1} - y_p^2 \quad (\text{B.14})$$

where

$$D_{p-1} = \phi_{00} - \sum_{i=1}^{p-1} y_i^2$$

is the residual energy for the optimum predictor of order $p-1$.

Recall, from Appendix A, that for the reflection coefficients (k_i)

$$\begin{aligned} D_p &= r(0) \prod_{i=1}^p (1 - k_i^2) \\ &= D_{p-1} (1 - k_p^2) \end{aligned} \quad (\text{B.15})$$

where

$$D_{p-1} = r(0) \prod_{i=1}^{p-1} (1 - k_i^2)$$

is the residual energy for the optimum predictor of order $p - 1$. The autocorrelation at lag zero ($r(0)$) is just the speech energy, and is equal to ϕ_{00} . Thus, from Equations B.14 and B.15

$$\begin{aligned} D_i &= D_{i-1} - y_i^2 \\ &= D_{i-1} (1 - k_i^2) \end{aligned}$$

and simple algebra yields

$$k_i = y_i / \sqrt{D_{i-1}} \quad (\text{B.16})$$

Since the residual energy D_i must be greater than (or equal to) zero, from Equation B.14 we get

$$y_i^2 \leq D_{i-1}$$

Thus, $|k_i| \leq 1$, which results in a stable predictor. The resulting predictor is not optimum in terms of minimizing the mean squared prediction error—but it is stable.

END

09-08-88

FIN