A Low Delay 16 kbit/sec Coder for Speech Signals

by

Vasu Iyengar
B.Eng. (Electrical)

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

Department of Electrical Engineering
McGill University
Montréal, Canada
August, 1987

Abstract

This thesis studies a low delay speech coder operating at 16 kbits/sec. The coding algorithm is a delayed decision tree-coding scheme using the multipath (M, L) tree search algorithm. Two different adaptive synthesis filter configurations are used for mapping the innovations code (excitation sequences) to the output or reconstruction code. The first configuration uses a short-term or formant synthesis filter which reconstructs the speech spectral envelope. The fine structure of the speech spectrum is contained in the innovations sequence in this case. The second configuration consists of a cascade of a long-term or pitch synthesis filter and a formant synthesis filter. The pitch filter first reconstructs the spectral fine structure, and the formant synthesis filter inserts the spectral envelope. Backward adaptation of both the pitch and formant synthesis filters results in no side information requirements for the transmission of adaptation information. Noise feedback encoder configurations are employed to allow for the use of a frequency weighted error measure. The innovations tree is populated using random numbers with a Laplacian distribution, and includes the effect of a backward adaptive gain.

Results of both objective and formal subjective testing of the encoding algorithm are presented. At an encoding rate of 16 kbits/sec, and an encoding delay of 1 ms, the algorithm yields a subjective quality equivalent to 7 bits/sample log-PCM. The encoding algorithm is suitable for use in digital links having low encoding delay constraints, such as the switched telephone network. Recommendations for future studies are given.

Sommaire

Ce mémoire étudie un codeur de parole à delai court et opérant à une vitesse de 16 kbits/sec. L'algorithme de codage est un schème de codage d'arbre à décision reportée utilisant l'algorithme multi-chemin (M, L) de recherche dans un arbre. Deux types de configuration de filtre de synthèse adaptif ont été utilisés pour appliquer les séquences d'excitation au code de reconstruction en sortie. Le première configuration utilise un filtre de synthèse de formant reconstruisant l'enveloppe du spectre vocal. La structure fine du spectre vocal est dans ce cas contenue dans la séquence d'excitation. Le deuxième configuration consiste en une cascade formée d'un filtre de synthèse de périodicité et d'un filtre de synthèse de formant. Le filtre de périodicité reconstruit d'abord le structure fine du spectre, le filtre de formant insère l'enveloppe spectral. L'adaptation causal des filtres de synthèse de périodicité et de formant implique qu'il n'est pas nécessaire de transmettre d'information latérale. Des configurations de codeur à retour de bruit sont employées pour permettre l'utilisation d'une mesure pondérée d'erreur de fréquence. L'arbre d'excitation est aléatoirement peuplé en utilisant des nombres générés selon une distribution de Laplace. Un gain adaptif est utilisé dans l'arbre d'excitation.

Les résultats des tests objectifs est subjectifs formels de l'algorithme de codage sont présentés. A une vitesse de codage de 16 kbits/sec et un délai d'une micro seconde, l'algorithme produit une qualité subjective équivalente à 7 bits/échantillon log-PCM. L'algorithme de codage convient pour utilisation dans les liens digitaux ayant une contrainte de delai de codage faible, tel que les réseaux téléphoniques commutés. Finalement on offre des recommendations au sujet de recherches futures possibles.

Acknowledgments

I would like to sincerely thank my supervisor Dr. Peter Kabal for his invaluable guidance during this study. I am also very grateful for his financial assistance. A special thanks goes to Mr. D. J. Millar for his help and advice. The lab facilities provided by INRS-Télécommunications is greatly appreciated.

I owe a very special thanks to my parents and my brother Seshadri and his family, for their continuous support and encouragement throughout my studies.

The following friends also deserve mention for providing the much needed companionship during the course of this work; (in alphabetical order), Chung Cheung Chu, John (Yiannos) Michaelides, Vitalice Oduol, and Ravi Ramachandran.

Table of Contents

List of Figures

Chapter 1 Introduction

Digital Coding refers to the process of representing an *analog signal* (continuous in time and amplitude), by a *digital signal* (discrete both in time and amplitude). The digital signals are usually in the form of a stream of binary digits or bits.

Digital representation of analog signals offers many advantages [1]. The most important and obvious of these are (1) regenerative amplification for transmission over long distances and (2) ease of encryption for secure communications. Regenerative amplification allows better use of noisy channels, provided enough repeaters are stationed between the source and the destination. During each regeneration, the digital signal can be stripped of noise and interference introduced during the transmission. If regeneration is done before the channel noise and interference become too severe, performance can be made virtually independent of distance. In analog systems, however, noise and other impairments accumulate with distance. Also, analog repeaters exhibit some non-linear behaviour, the effects of this being accumulated at every repeater. The advantages of digital coding for speech signals have, over the years, prompted the integration of digital techniques into telephone networks. As a result, the last two decades have witnessed a great deal of activity in *speech coding.*

Digital coding of analog waveforms entails some amount of coding distortion. Coding distortion, in general, decreases with increasing coder bit rate. However, high bit rate signals require a higher transmission bandwidth for reliable transmission,

and this implies higher transmission costs. The goal of all speech coding algorithms, therefore, is to represent speech with high quality (low distortion), yet at low encoding bit rates.

Speech researchers have over the years distinguished between four grades of quality (1) *Commentary* or *Broadcast quality*, (0–7000 Hz bandwidth) which is wide bandwidth speech with no perceptible noise, (2) *Toll quality* or *Telephone quality*, (0–3400 Hz bandwidth), which is narrow bandwidth speech as heard over the switched telephone network, (3) *Communications quality*, which is characterized by high intelligibility but with perceptible amounts of noise and distortion, and (4) *Synthetic quality*, which is highly intelligible, but which also tends to sound buzzy and unnatural, and lacks speaker identifiability. The work in this thesis is concerned with coders that yield toll quality or near toll quality speech, for use over the switched telephone network.

A starting point in any speech coding process is that of time discretization (sampling). According to the sampling theorem, any band-limited analog signal can be sampled uniformly without loss of information provided the sampling rate is at least equal to the *Nyquist rate*, (twice the highest frequency component in the original analog signal). Although speech signals have an energy which falls off rapidly with frequency, low pass filtering to preserve the perceptually important frequencies ensures that the signal is essentially band-limited. For example, speech transmitted over the telephone network is first band-limited to 3400 Hz, and then sampled using a conservative rate of 8000 Hz. Uniform time sampling at a sampling rate greater than or equal to the Nyquist rate is an information preserving operation, and the original band-limited analog signal can be recovered by low pass filtering the sampled signal. To obtain a digital representation, the information contained in the sampled speech signal has to be quantized. Speech coders are broadly classified into two categories according to how the sampled speech information is quantized. The two coder

types are (1) *Voice coders* or *vocoders*, and (2) *Waveform coders.*

Vocoders can be described in terms of a discrete time model for sampled speech signals. The model consists of a recursive digital filter driven by either white noise or a periodic pulse train [2]. The type of excitation and the parameters of the filter are determined in an analysis phase. The relevant parameters are then transmitted to the receiver where they are used to synthesize the output speech. Since no simple parametric model for the filter and the excitation can possibly take into account all the complexities of the human speech production process, the output speech usually has a 'buzzy' and synthetic quality. The synthetic quality of vocoder speech is also in part due to the automatic parameter estimation algorithms. Better quality speech can be obtained by *manual fine tuning* of the filter and excitation parameters. Vocoders generally operate at rates below 4.8 kbits/sec, and yield synthetic quality speech, and certainly do not meet toll quality standards yet. The coder considered in this thesis is therefore not of the vocoding type.

Waveform coders, unlike vocoders, attempt to track the actual time variations of the input speech. Waveform coders generally operate at rates above 9.6 kbits/sec, and achieve qualities ranging from communications quality to broadcast quality. Hence, almost all toll quality coders are waveform coders. The conventional method of waveform coding is *Pulse Code Modulation* (PCM), with the so-called μ-law and A-law companding schemes [3],[4]. Figure 1.1 illustrates the operation of the PCM coder. The PCM coder illustrates the two basic processes involved in waveform coding, *sampling* and *amplitude quantization.* Uniform time sampling is followed by amplitude quantization to one of a finite set of amplitudes. Usually, speech sampled at 8000 Hz is quantized to one of 256 amplitude levels giving a coding rate of 64 kbits/sec. After sampling, the analog input is compressed using the μ-law or A-law schemes. The final step in the encoding process is uniform quantization of the sampled and compressed signal. At the receiver, the output of the inverse quantizer

is expanded and low pass filtered to yield the output speech signal. Companding together with uniform quantization is equivalent to nonuniform quantization with a logarithmic characteristic, and yields a higher dynamic range and better idle-channel noise performance than uniform PCM schemes. Log-PCM is a low complexity coding scheme with essentially zero encoding delay, and was first standardized about 20 years ago. Spurred on by the decreasing cost of hardware, researchers have looked to more complex coding schemes that provide equivalent quality speech at lower rates. The complexity of a coder is determined by its signal processing and memory requirements. The following paragraphs review the evolution of some of these coding schemes, leading up to the coder studied in this thesis.

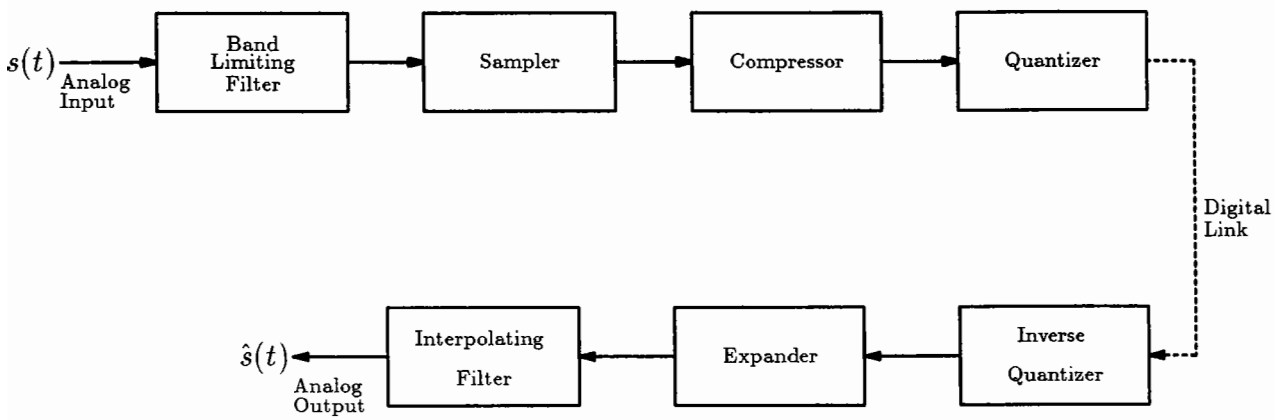


Fig. 1.1 Log-PCM Coder

It is well known that speech signals sampled at the Nyquist rate exhibit significant correlation between successive samples. In PCM, each sample is coded independently of all other samples. This method of coding is inefficient since the correlation between samples is ignored, and requires coding at rates of 64 kbits/sec to provide telephone quality speech. A class of coders called *Differential Pulse Code Modulation* (DPCM) coders utilize this redundancy to achieve reductions in coding rate over PCM [4]. A DPCM coder is shown in Fig. 1.2. It includes the use of a predictor in addition to a quantizer. The predictor exploits redundancy in the input stream through time

domain operations. The input to the quantizer is a prediction error formed by the difference between the current sample and the output of the predictor. Because of the correlation between successive samples, the prediction error sequence will have a smaller dynamic range, or variance, than the sampled speech signal. The quantizer can therefore have fewer levels in a DPCM coder than in a PCM coder without suffering an increase in quantization error variance.

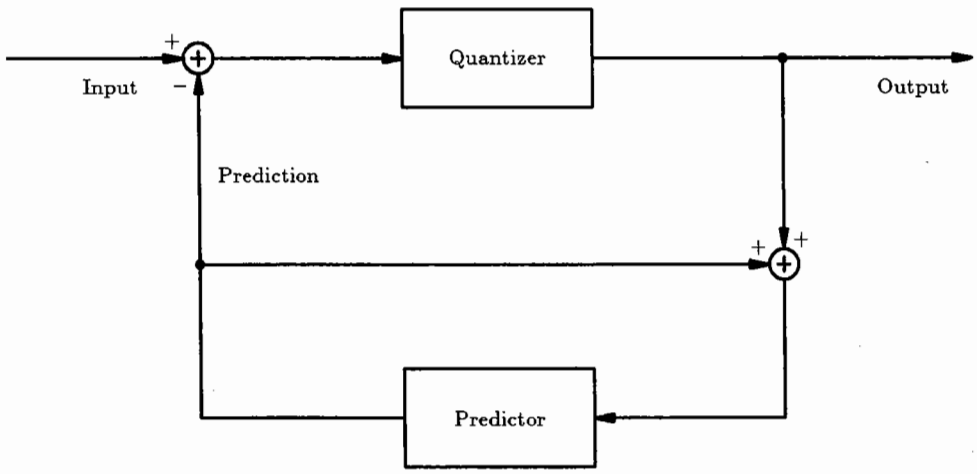


Fig. 1.2 Differential Encoder

Speech is a quasi-stationary source whose short time behaviour is stationary, but whose modes of stationarity change slowly with time. In DPCM coders, a fixed quantizer and fixed predictor are used. The design of these coders is based on long-term statistics of the input signal. The performance of differential encoders can be further improved by adapting the predictor and quantizer to match the short-term modes of stationarity of the input speech, and the dynamic range of the prediction error sequence respectively. Such coders are called *Adaptive Differential Pulse Code Modulation* (ADPCM) coders [4]. ADPCM coders can yield toll quality speech (equivalent to 64 kbits/sec log-PCM) at rates of 32 kbits/sec, a saving of 1:2 over conventional log-PCM techniques [5]. Indeed, the CCITT formally approved an ADPCM coder algorithm as an international standard at its October 1984 plenary session [6].

The CCITT ADPCM standard, while providing toll quality speech at half the bit rate of the PCM standard, also observes several important coder constraints imposed by existing telecommunications networks. One very severe constraint for terrestrial networks is that of echo tolerance of telephone links. Telephone links provide unsatisfactory performance in the presence of large round trip delays due to the problem of disturbing echo effects. Echoes are generated at the interface between 4-wire and 2-wire lines due to impedance mismatches at the hybrid interface. The disturbing effects of echoes can be reduced by introducing artificial losses in both directions of the 4-wire link. The amount of echo loss required for satisfactory performance increases with the round trip delay. Both the encoding delay and the propagation time contribute to the round trip delay. Since echo suppression affects the received signal also, the amount of echo suppression loss used has to be limited, and this in turn sets a limit on the maximum allowable round trip delay. Also end-to-end links may not be entirely digital, necessitating several tandem coding/decoding processes in cascade. Transmission across digital links may also involve several stages of digital transcodings to and from 64 kbit/sec log-PCM. In consideration of all these factors, it is in general necessary to limit single stage encoding delays to a maximum of 1 to 2 ms per direction [7]. This limit on processing delay should be taken into account when designing a new coding algorithm. A second property of the CCITT coding algorithm is that of backward adaptation of the predictor and quantizer, i.e., the predictor and quantizer are updated using information contained in the past quantized output signal. Since this signal is available at the receiver, it can keep track of the evolution of the predictor and quantizer without relying on side information. The transmission of side information necessitates more complex framing schemes for correct multiplexing of the quantizer output and side information bit streams, thus increasing overall coder complexity.

The aim of this study is to obtain a 16 kbits/sec toll quality coder that has

(1) an encoding delay no greater than 2 ms, and (2) no side information requirements. The performance of an ADPCM coder (based on the CCITT algorithm) at 16 kbits/sec and 24 kbits/sec is, unfortunately, poor. Specifically, at 16 kbits/sec, the output is considerably distorted and has high amounts of quantization noise. This is to be expected since the predictor and its update algorithms are hindered by poor quantization effects. The quantization effects in turn are affected by poor predictor performance. Hence at low rates, it has been necessary to use forward adaptation schemes for adapting the predictor. This involves operation on a block of input speech for determining the optimum predictor coefficients [3]. The main disadvantages of this are (1) encoding delays of the order of 10–20 ms due to data buffering to calculate the optimal predictor coefficients for the block, and (2) transmission of side information for parameters and for maintaining coder/decoder frame alignment. Hence, coders employing forward adaptation schemes are unsuitable for general use over terrestrial telephone networks. Some other refinement of the basic ADPCM scheme is required to enable it to perform adequately at low rates, observing at the same time the constraints of (1) low encoding delay and (2) no side information. A major advance towards improving the performance of waveform coders at low bit rates came with the use of multipath tree search algorithms with differential waveform coders. Various developments along these lines are discussed next.

A characteristic of differential coders is that the possible quantized output sequences are arranged in the form of a *tree code.* Encoding in conventional schemes then proceeds by a *single path search* of this tree to find the best output sequence. This has been identified as being a clear shortcoming of conventional DPCM and ADPCM. A major refinement to differential coders has been that of employing *Delayed Decision* with such coders. These schemes are called delayed decision DPCM or delayed decision ADPCM as the case may be. Differential encoders with delayed decision, as the term implies, involves some encoding delay. However, unlike forward

adaptive differential coders, where encoding delay is utilized for efficient redundancy removal from the input, delayed decision coders utilize encoding delay to provide the capability of a *multipath search* through the code tree, thus making more efficient use of the tree code. Delayed decision is a feature that leads to efficient coding of redundant as well as non-redundant inputs. The delays are usually of the order of a few samples, and can be kept within network echo delay constraints.

Delayed decision applied to a DPCM coder was first studied by Anderson and Bodie [8]. A computationally efficient multipath search algorithm, called the (M, L) algorithm, was used with a fixed predictor and quantizer together with a squared error distortion measure to give more than 4 dB improvement in signal-to-noise ratio over conventional single path searched DPCM schemes at a rate of 16 kbits/sec. Jayant and Christensen [9] applied delayed decision using the (M, L)-algorithm to a differential coder with backward adaptive quantization and fixed prediction. Although delayed decision coding in the above studies provided gains both in terms of measurable signal-to-noise ratio gains and in terms of perceived speech quality over conventional differential coders at a rate of 16 kbits/sec, the speech output quality was still reported to be characterized by easily perceived quantization noise. This is due to two main shortcomings. First, the encoding algorithm utilizes fixed and not adaptive prediction. Second, the tree code is a deterministic tree code. Much can be gained with the use of so-called *stochastic* tree codes as will be shown in this thesis work. Delayed decision ADPCM with forward adaptive prediction have been studied in [10] and [11]. The use of a stochastic code with forward adaptation was reported in [12]. However as mentioned previously, forward adaptation schemes are unsuitable for terrestrial telephone networks.

This thesis studies a delayed decision tree coder employing backward adaptive quantization and prediction. A stochastically populated innovations tree that includes the effect of a backward adaptive gain is used. Both a short-term or formant predictor

and a long-term or pitch predictor are used. Both types of predictors are backward adaptive. Generalized noise feedback encoder configurations [13] are utilized to permit the use of a subjectively meaningful frequency weighted error measure. The use of a pitch predictor in a tree coding application, and the adaptation of the pitch predictor using backward adaptive algorithms is new. Backward adaptation of the formant predictor in tree coding, and the use of a backward adaptive gain with a stochastic innovations tree is also new.

Results show that with an 8-sample (1 ms with an 8 kHz sampling rate) encoding delay, and a coding rate of 16 kbits/sec, speech quality equivalent to 7 bits/sample log-PCM is achieved.

1.1 Scope and organization of Thesis

This thesis is organized into five chapters. Chapter 2 reviews the conventional ADPCM coder, and the generalized noise feedback coding scheme of Atal [13], with both short-term and long-term predictors. The adaptation algorithms used for the predictors in this work are described. Some backward gain adaptation schemes used with conventional quantizers are presented.

In Chapter 3, descriptions are given of various tree codes associated with conventional DPCM and ADPCM schemes. The multipath (M, L) search algorithm is briefly reviewed. Deterministic and stochastic tree codes are described. The conventional quantizer gain adaptation schemes are extended to backward gain adaptation with stochastic innovations code trees. A historical summary of past work in tree coding relevant to this thesis work, in given.

Chapter 4 gives a detailed description of the encoding algorithms studied. Both objective test results and results of subjective listening tests are presented.

Chapter 5 concludes the work and proposes recommendations for future research.

Chapter 2 Conventional Differential Coders

This chapter provides a review of conventional Differential Encoding schemes. These coders provide significant coding gains over PCM systems by exploiting redundancies in sampled speech signals. The conventional Feedback Around the Quantizer configuration [14] is presented, together with the Generalized Predictive Coder configuration with adjustable noise spectrum [13].

This thesis studies the use of delayed decision using the Generalized Predictive Coder configuration. Such an encoder configuration allows for adaptive control of the output noise spectrum.

Finally, the adaptation schemes used in this study for the predictor and quantizer which constitute part of any differential encoding scheme are briefly described. Such adaptivity is necessary in order to account for the time varying nature of speech signals.

2.1 Conventional ADPCM System

The basic ADPCM system is shown in Fig. 2.1. The main components of the system are the *predictor* and the *quantizer*. Redundancy removal from the input speech samples is achieved by subtracting from each input sample $s(n)$, a predicted sample $\tilde{s}(n)$. Predictors that remove near-sample redundancies from the input have

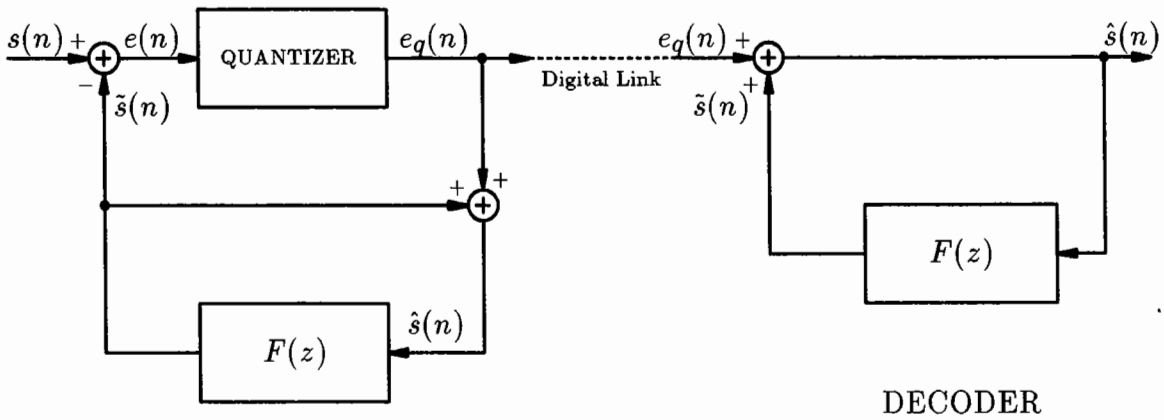


Fig. 2.1 Conventional ADPCM Coder

the system function of

$$F(z) = \sum_{i=1}^{P} a_i z^{-i}, \tag{2.1}$$

where P is the order of the predictor. Such predictors are based on the modelling of the acoustic resonances of the vocal tract by an all pole synthesis filter. The vocal tract resonances are called formants, and hence such predictors will be referred to henceforth as *formant predictors.* The prediction error formed as $(s(n) - \tilde{s}(n))$ is then quantized and transmitted over the channel. Simultaneously, this quantized value is summed with the predicted value to yield a quantized output sample $\hat{s}(n)$. The encoder configuration therefore also incorporates a local decoder. Since the input to the predictor is the reconstructed output $\hat{s}(n)$, the predicted signal $\tilde{s}(n)$ is given by

$$\tilde{s}(n) = \sum_{i=1}^{P} a_i \hat{s}(n-i) \quad . \tag{2.2}$$

The equations describing the operation of the system are given below.

$$\begin{aligned} e(n) &= s(n) - \tilde{s}(n) \\ e_q(n) &= e(n) + q(n) \end{aligned} \tag{2.3}$$

Assuming that there are no channel errors, the decoder will form a reconstructed signal $\hat{s}(n)$ given by

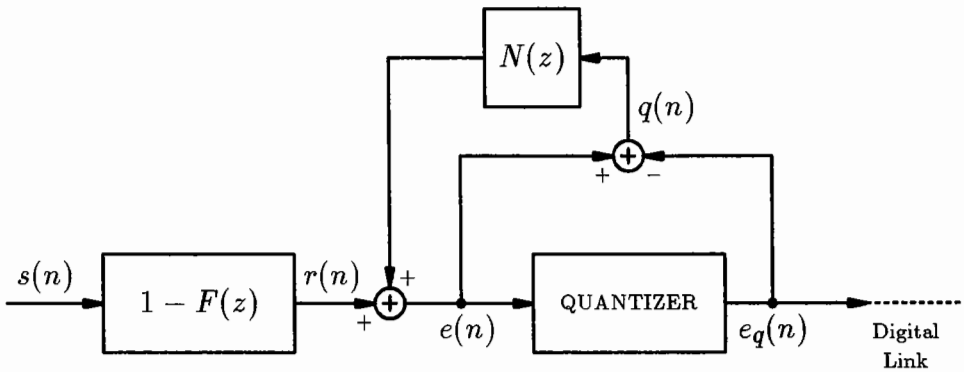
$$\begin{aligned}\hat{s}(n) =& e_q(n) + \tilde{s}(n) \\ =& e(n) + \tilde{s}(n) + q(n) \\ =& s(n) + q(n).\end{aligned} \tag{2.4}$$

The reconstruction error given by $\hat{s}(n) - s(n)$ is equal to the quantization error $q(n)$. For a given number of quantizer levels, the quantization error variance tends to be proportional to the variance of the quantizer input. Since the prediction error sequence $e(n)$ has a smaller variance than the input $s(n)$, differential coders will provide better performance than PCM systems, for the same number of quantizer levels. The main objective in the design of the predictor is therefore that of maximizing the *prediction gain*, the prediction gain being the ratio of the power in the input signal $s(n)$ to the power in the prediction error signal $e(n)$. The quantization error for a multi-level quantizer with finely spaced levels is approximately *white*, i.e., has a fairly flat power spectrum. Since the quantization error is equal to the reconstruction error, the latter also has a white spectrum under the given assumptions. The Generalized Predictive Coder configuration allows more control over the shape of the reconstruction error spectrum, and is presented in the next section.

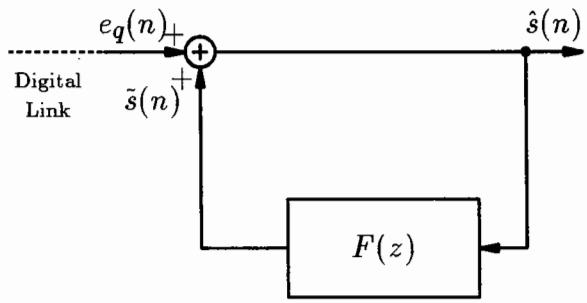
2.2 Generalized Predictive Coder Configuration

The block diagram of a Generalized Predictive Coder is shown in Fig. 2.2. In the conventional ADPCM coder described in the previous section, the output noise is approximately white. A white spectrum is, however, not perceptually good, especially if the noise power is high. The shape of the noise spectrum in relation to the speech spectrum is important from the point of view of perceived distortion in the output speech. Noise in the formant regions is partially or totally masked by the speech

signal, since the speech power is high in the formant regions. The perceived noise in the output speech therefore comes from noise in those frequency ranges where the signal level is low. A configuration that allows for adaptive adjustment of the noise spectrum in relation to the speech spectrum is the Generalized Predictive Coder configuration [13].



ENCODER



DECODER

Fig. 2.2 Generalized Predictive Coder

In Fig. 2.2 $F(z)$ and $N(z)$ are given by

$$F(z) = \sum_{i=1}^{P} a_i z^{-i}$$
$$N(z) = \sum_{i=1}^{P} b_i z^{-i} \qquad (2.5)$$

The quantizer input $e(n)$ is given by

$$e(n) = s(n) - \sum_{i=1}^{P} a_i s(n-i) + \sum_{i=1}^{P} b_i q(n-i) \tag{2.6}$$

The output of the decoder is given by

$$\begin{aligned} \hat{s}(n) =& e_q(n) + \sum_{i=1}^{P} a_i \hat{s}(n-i) \\ =& e(n) - q(n) + \sum_{i=1}^{P} a_i \hat{s}(n-i) \\ =& s(n) - \sum_{i=1}^{P} a_i s(n-i) + \sum_{i=1}^{P} b_i q(n-i) - q(n) + \sum_{i=1}^{P} a_i \hat{s}(n-i) \quad . \end{aligned} \tag{2.7}$$

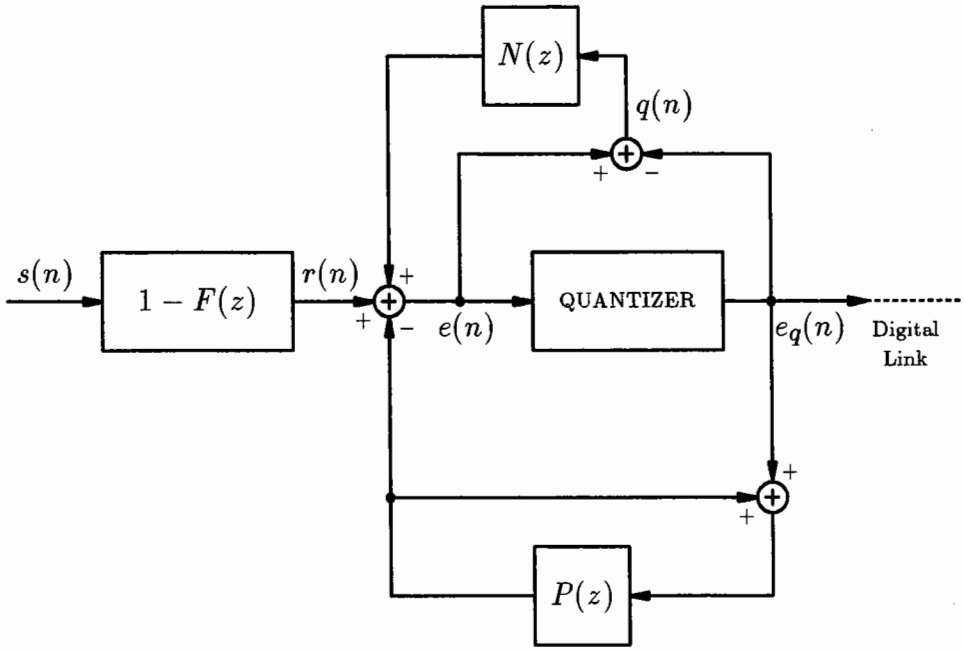
Taking z-transforms of both sides yields

$$\begin{aligned} \hat{S}(z) =& S(z) - S(z)F(z) + N(z)Q(z) - Q(z) + \hat{S}(z)F(z) \\ S(z) - \hat{S}(z) =& Q(z)\frac{1 - N(z)}{1 - F(z)} \end{aligned} \tag{2.8}$$

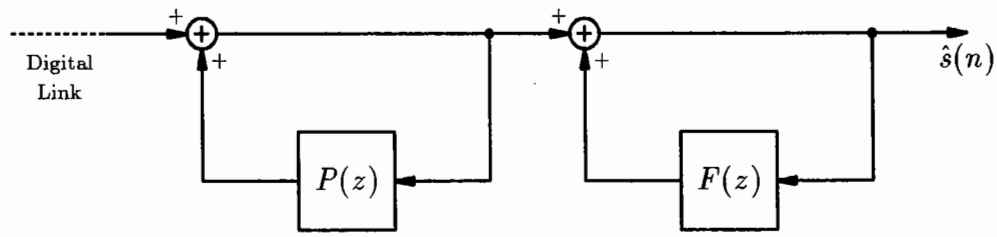
The spectrum of the reconstruction error is $S(z) - \hat{S}(z)$. $Q(z)$ is the spectrum of the quantization error, and under the usual assumptions of white noise, is equal to a constant. The shape of the reconstruction error can be controlled by choosing $N(z)$ appropriately. It is usual to choose $N(z)$ as a bandwidth expanded version of $F(z)$, i.e., $N(z) = F(\frac{z}{\mu})$, where $0 \leq \mu \leq 1$. The value of μ is usually chosen to be between 0.75 and 0.9. A value of $\mu = 1$ gives a white reconstruction error spectrum, while $\mu = 0$ gives an error spectrum which has the same shape as the signal spectrum. An intermediate value of μ has the effect of decreasing the noise power in the valleys (regions between the formants) of the speech spectral envelope, and increasing the noise power in the formant regions. This decreases the perceptual effect of noise in the output speech. Note that the above method of controlling the output noise spectrum relies on the quantizer noise spectrum being white.

2.3 Pitch Prediction

The block diagram of a Generalized Predictive Coder with a formant predictor as well as a *long term* or *pitch predictor* is shown in Fig. 2.3.



ENCODER



DECODER

Fig. 2.3 Generalized Predictive Coder with Pitch Prediction

The use of pitch prediction is motivated by the fact that voiced speech segments exhibit considerable similarity between adjacent pitch periods. Voiced speech is produced by excitation of the vocal tract by rhythmic glottal excitation. Voiced speech therefore tends to be quasi-periodic in nature, the period being equal to the pitch period, i.e., the time interval between adjacent glottal pulses. Formant predictors only remove redundancies in the input speech that are due to the vocal tract shape, and

hence, formant predicted residual signals will contain pitch pulses for voiced speech. Between pitch pulses, the formant residual is noise-like in nature. The formant residual during voiced segments still contains redundant information in the form of pitch pulses, which can be effectively removed with the use of pitch prediction. The overall residual can therefore be quantized more easily since its variance is further reduced through pitch prediction.

A third order pitch predictor has the system function of

$$P(z) = \beta_1 z^{-M_p+1} + \beta_2 z^{-M_p} + \beta_3 z^{-M_p-1}, \tag{2.9}$$

where M_p is the pitch period in samples. Since the sampling frequency is fixed and is in general unrelated to the pitch period, the pitch period may not be an integral number of samples. A third order predictor interpolates between adjacent samples and gives higher correlation from one period to the next than the individual samples.

Adaptation of pitch predictors is necessary, since both the pitch lag and the predictor coefficients have to be fine tuned to the analysis segment. Pitch prediction is conventionally achieved with forward adaptation. The use of backward adaptation is considered in this work. The following sections consider adaptation schemes used in this work for the formant and pitch predictors.

2.4 Formant Predictor

Since the signal spectrum is time varying, the coefficients of the filter $F(z)$ must be time varying also, to obtain a small prediction error variance. Predictor update algorithms are broadly classified into two categories, forward and backward adaptation algorithms. Forward adaptation schemes were not considered in this work since they involve large encoding delays, and also require extra channel capacity for the transmission of adaptation information.

Backward adaptation algorithms make use of information contained in the past quantized data to update the predictor. The predictor can therefore be updated at every sampling instant. This forms a framework for the use of recursive or sequential update algorithms [15]. Most recursive update algorithms are based on the method of steepest descent. The mean square prediction error is in general a quadratic function of the predictor coefficients. The error surface is therefore a bowl-shaped surface. Algorithms based on the method of steepest descent adjust the predictor coefficients by continually seeking the bottom of this bowl shaped surface.

2.4.1 Update Algorithm

The update algorithm used for the predictor in this work is the Adaptive Lattice Algorithm [16]. This algorithm is briefly described below.

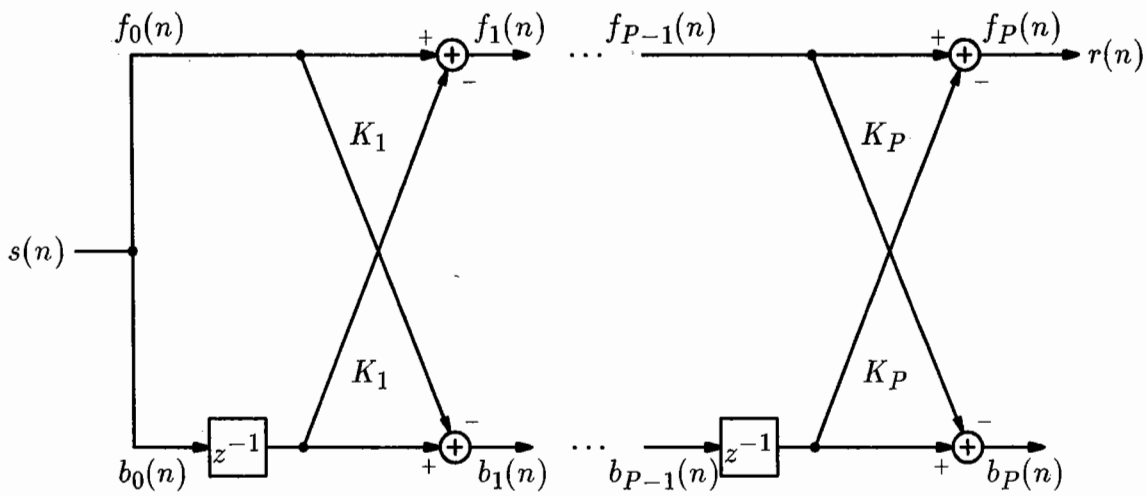


Fig. 2.4 Lattice Filter of Order P

The prediction error filter in Fig. 2.2 is given by

$$\begin{aligned} A(z) &= 1 - F(z) \\ &= 1 - \sum_{i=1}^{P} a_i z^{-i} \end{aligned} \tag{2.10}$$

This transfer function can be implemented as a lattice filter shown in Fig. 2.4. The reflection (or partial correlation) coefficients K_m have a unique relationship with the predictor coefficients a_i. Given K_m, $m = 1,\ldots,P$, the set a_i, $i = 1,\ldots,P$ are computed recursively from the following relations:

$$a_m^{(m)} = K_m \tag{2.11.a}$$

$$a_j^{(m)} = a_j^{(m-1)} + K_m a_{m-j}^{(m-1)}, \quad 1 \leq j \leq m-1. \tag{2.11.b}$$

Equations (2.11) are computed recursively for $m = 1, 2, \ldots, P$. The coefficients $a_j^{(m)}$ are the coefficients for the corresponding m^{th} order predictor.

Since the set of reflection coefficients K_m of the lattice and the coefficients a_i of the corresponding transversal filter have a one-to-one relationship, both implementations are entirely equivalent for *time invariant conditions.* In the time varying case however, and in particular the case where the filter $A(z)$ is updated at every sampling instant, the two implementations (lattice and transversal) are *not equivalent*, due to differing initial conditions. The adaptive lattice algorithm is an update scheme for the reflection coefficients of a lattice filter that can be used to realize the transfer function $A(z)$. In the actual implementation of the transfer function $A(z)$ however, one can use a lattice implementation directly, or use Eqs. (2.11) to convert the reflection coefficients to the corresponding transversal tap coefficients towards a direct form realization. Thus while the underlying update algorithm is an adaptive lattice, both lattice and transversal implementations can be used, although these are not entirely equivalent in theory.

From Fig. 2.4, the following relations hold at the first stage and at each succeeding stage of the lattice.

$$\begin{aligned} f_0(n) &= b_0(n) = s(n) \\ f_{m+1}(n) &= f_m(n) - K_{m+1} b_m(n-1) \\ b_{m+1}(n) &= -K_{m+1} f_m(n) + b_m(n-1) \end{aligned} \tag{2.12}$$

where $s(n)$ is the input speech signal. The value $r(n)$ is the prediction error or residual signal at the final stage of the lattice. It is given by

$$r(n) = f_P(n) = f_{P-1}(n) - K_P b_{P-1}(n-1) \tag{2.13}$$

The input speech signal is quasi-stationary. Therefore, the reflection coefficients K_m must vary with time to track the modes of stationarity of the input $s(n)$. The reflection coefficients K_m are therefore a function of time n, and shown explicitly by writing $K_m(n)$, $m = 1, 2, \ldots, P$. From the knowledge of all the quantities at time n given in Eq. (2.12), we need to compute the reflection coefficients $K_m(n+1)$, at time $n+1$. In the update method used, the reflection coefficients are updated on a stage by stage basis. The update method is based on the minimization of a weighted error of the form

$$E_m(n) = \sum_{k=-\infty}^{n} w(n-k) e_m^2(k) \ , \tag{2.14}$$

where $e_m^2(k)$ is a weighted sum of forward and backward residual energies given by

$$e_m^2(k) = (1-\gamma) f_m^2(k) + \gamma b_m^2(k), \qquad 0 \le \gamma \le 1, \tag{2.15}$$

and $w(n)$ is a causal window function. Substituting Eq. (2.15) in Eq. (2.14) yields

$$E_m(n) = \sum_{k=-\infty}^{n} w(n-k)[(1-\gamma) f_m^2(k) + \gamma b_m^2(k)] \tag{2.16}$$

Substituting for $f_m^2(k)$ and $b_m^2(k)$ in Eq. (2.16) yields

$$\begin{aligned} E_m(n) = & \sum_{k=-\infty}^{n} w(n-k)(1-\gamma)[f_{m-1}^2(k) \\ & \qquad - 2K_m(n) f_{m-1}(k) b_{m-1}(k-1) + K_m^2(n) b_{m-1}^2(k-1)] \\ & + \sum_{k=-\infty}^{n} w(n-k)(\gamma)[K_m^2(n) f_{m-1}^2(k) \\ & \qquad - 2K_m(n) b_{m-1}(k-1) f_{m-1}(k) + b_{m-1}^2(k-1)] \end{aligned} \tag{2.17}$$

Minimizing $E_m(n)$ with respect to $K_m(n)$ yields the update $K_m(n+1)$. The update $K_m(n+1)$ is given by

$$K_m(n+1) = \frac{\sum_{k=-\infty}^{n} w(n-k) f_{m-1}(k) b_{m-1}(k-1)}{\sum_{k=-\infty}^{n} w(n-k)[\gamma f_{m-1}^2(k) + (1-\gamma) b_{m-1}^2(k-1)]} = \frac{C_m(n)}{D_m(n)} \tag{2.18}$$

The sufficient conditions for stability of the synthesis filter, $\frac{1}{1-A(z)}$ are that (1) $\gamma = 0.5$ and (2) $w(n) \geq 0$ for $n \geq 0$ [16]. The factor γ in Eq. (2.15) determines the mix between forward and backward residuals. Having $\gamma = 0.5$ corresponds to an equal mix of forward and backward residual energies. It is desirable to minimize only the forward residual, but this implies having $\gamma = 0$ which does not guarantee stability of the resulting synthesis filter. The parameter γ will be referred to henceforth as the *lattice stability constant.*

2.4.2 Choice of Window and Effect on Computation

The window $w(n)$ is used to form a weighted sum of a function of the forward and backward residual energies at each stage of the lattice, i.e., the window $w(n)$ weights the residual energy into the past. The reflection coefficient for a particular stage is then updated for the next time instant by minimizing the weighted sum in Eq. (2.14) with respect to the reflection coefficient at that stage. The immediate past contains information concerning the current mode of stationarity of the input speech. Therefore, the shape of the window $w(n)$ should be such that the residual energy over the immediate past is weighted more than the residual energy over the more distant past. This ensures that the predictor evolves in accordance with the changing modes of stationarity of the input speech. The time frame over which the error is minimized

is also an important design factor. A time frame that is too long results in averaging over two or more different modes of stationarity, whereas a time frame that is too short will not contain enough information about the input speech. Two important considerations in choosing a window are therefore the *shape* of the window, and the *effective length* of the window. A third important consideration in choosing a window is the effect on the computational complexity of the update algorithm, as will be seen in the next paragraph.

The general update procedure is now given. At each time instant n, the algorithm has to maintain in memory the following, (1) $K_m(n)$, $m = 1, \ldots, P$, (2) $f_m(k)$, $m = 1, \ldots, P$, and $k < n$, and (3) $b_m(k)$, $m = 1, \ldots, P$, and $k \leq n-1$. The input $s(n)$ to the lattice is the sequence of past quantized output speech data that is available at both the encoder and the decoder. In response to the latest output $s(n)$, Eqs. (2.12) are used to compute $f_m(n)$ and $b_m(n)$ for $m = 1, \ldots, P$, P being the filter order. The sums in the numerator and denominator of Eq. (2.18) are the calculated for $m = 1, \ldots, P$. The updated reflection coefficients are then calculated and kept for the next time instant. The forward and backward residuals $f_m(n)$ and $b_m(n)$ are also kept for the next time instant. Although the sums in the numerator and denominator are shown to be infinite, one could use finite length windows. In that case, the forward and backward residual sequences at each stage have to be maintained in memory, the length of these sequences being equal to the length of the window. Note also that for each reflection coefficient update, a large number of multiplication operations are involved in evaluating the the numerator and denominator of Eq. (2.18). If a rectangular window is used, these multiplications are avoided, but a relatively large amount of memory is still required to maintain the forward and backward residual sequences at each stage. A way of avoiding such high computational complexity in the update procedure is to use windows $w(n)$ that can be considered as the impulse response of a causal finite order recursive digital filter. The use of such windows leads

to the possibility of obtaining recursive update equations for $C_m(n)$ and $D_m(n)$ in Eq. (2.18) as explained below.

In general if $W(z)$ is of the form

$$W(z) = \frac{1 - \sum_{i=1}^{N_z} \alpha_i z^{-i}}{1 - \sum_{i=1}^{N} \beta_i z^{-i}}, \tag{2.19}$$

then $C_m(n)$ satisfies the following recursive relationship,

$$\begin{aligned} C_m(n) = \sum_{i=1}^{N} \beta_i C_m(n-i) - \sum_{i=1}^{N_z} \alpha_i f_{m-1}(n-i) b_{m-1}(n-1-i) \\ + f_{m-1}(n) b_{m-1}(n-1), \end{aligned} \tag{2.20}$$

and $D_m(n)$ satisfies,

$$\begin{aligned} D_m(n) = \sum_{i=1}^{N} \beta_i D_m(n-i) - \sum_{i=1}^{N_z} \alpha_i [\gamma f_{m-1}^2(n-i) + (1-\gamma) b_{m-1}^2(n-1-i)] \\ + [\gamma f_{m-1}^2(n) + (1-\gamma) b_{m-1}^2(n-1)]. \end{aligned} \tag{2.21}$$

A simple window is the one-pole or exponential window given by

$$W(z) = \frac{1}{1 - \beta z^{-1}}. \tag{2.22}$$

The update equations for $C_m(n)$ and $D_m(n)$ are then given by

$$\begin{aligned} C_m(n) &= \beta C_m(n-1) + f_{m-1}(n) b_{m-1}(n-1) \\ D_m(n) &= \beta D_m(n-1) + [\gamma f_{m-1}^2(n) + (1-\gamma) b_{m-1}^2(n-1)] \end{aligned} \tag{2.23}$$

Such a recursive update offers considerable savings in computation and memory requirements of the update algorithm. The equivalent window response $w(n)$ is given by

$$w(n) = \begin{cases} \beta^n & \text{for } n \geq 0 \\ 0 & \text{for } n < 0 \ , \end{cases} \tag{2.24}$$

and the parameter β controls the effective length of the window.

The simplified update algorithm that results with the use of a recursive window is now given. The algorithm maintains the following in memory, (1) $K_m(n)$, $m =$

$1,\ldots,P$, (2) $C_m(n-i)$, $i=1,\ldots,N$, $D_m(n-i)$, $i=1,\ldots,N$, and for $m=1,\ldots,P$, (3) $f_{m-1}(n-i)$, $i=1,\ldots,N_z$, $b_{m-1}(n-i)$, $i=1,\ldots,N_z$, and for $m=1,\ldots,P$. For each value of m, Eqs. (2.12) are used to calculate $f_m(n)$ and $b_m(n)$. The quantities $C_m(n)$ and $D_m(n)$ are then updated according to Eq. (2.20) and Eq. (2.21). The updated reflection coefficient $K_m(n+1)$ is given by Eq. (2.18). The reflection coefficients and the backward residual $b_m(n)$ are then saved for the next time instant.

2.5 Pitch Predictor Update Algorithm

Unlike recursive adaptation schemes used for formant predictors, the pitch predictor adaptation scheme is basically a non-recursive procedure operating on past quantized formant residual samples.

A 3-tap pitch predictor is given by

$$P(z) = \beta_1 z^{-M_p+1} + \beta_2 z^{-M_p} + \beta_3 z^{-M_p-1}, \tag{2.25}$$

was used exclusively in this work. The update is based on the covariance formulation of linear prediction [2]. Minimizing the mean square error over a frame of length N samples results in the following system of linear equations to be solved,

$$\sum_{n=1}^{N} r(n)r(n-M_p+2-i) = \sum_{j=1}^{3} \beta_j \sum_{n=1}^{N} r(n-M_p+2-i)r(n-M_p+2-j), \tag{2.26}$$

for $i=1,2,3$. The sequence $r(n)$ is a formant predicted residual signal. This above equation can be written in matrix form as

$$\begin{bmatrix} \phi(M_p-1,M_p-1) & \phi(M_p-1,M_p) & \phi(M_p-1,M_p+1) \\ \phi(M_p,M_p-1) & \phi(M_p,M_p) & \phi(M_p,M_p+1) \\ \phi(M_p+1,M_p-1) & \phi(M_p+1,M_p) & \phi(M_p+1,M_p+1) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} \phi(0,M_p-1) \\ \phi(0,M_p) \\ \phi(0,M_p+1) \end{bmatrix} \tag{2.27}$$

where $\phi(i,j)$ is given by

$$\phi(i,j) = \sum_{n=1}^{N} r(n-i)r(n-j)\ . \tag{2.28}$$

This in turn can be written compactly as $\mathbf{\Phi\beta} = \boldsymbol{\alpha}$.

Given data consisting of the past quantized formant residual or the actual formant residual, an estimate of the pitch period M_p is first obtained, and this value of M_p is used in conjunction with Eq. (2.27) to obtain the corresponding set of predictor coefficients β_i. In a backward adaptation scheme, the sequence $r(n)$ would correspond to a quantized formant residual signal.

The lag estimate is obtained in this work using a computationally inexpensive method described in [17]. This method is briefly described below. The mean squared prediction error ε^2 is given by

$$\varepsilon^2 = \phi(0,0) - \boldsymbol{\alpha}^T \mathbf{\Phi} \boldsymbol{\alpha}. \tag{2.29}$$

The pitch lag M_p is chosen so as to maximize $\boldsymbol{\alpha}^T \mathbf{\Phi} \boldsymbol{\alpha}$. If the off-diagonal terms in the matrix of Eq. 2.27 are neglected, $\boldsymbol{\alpha}^T \mathbf{\Phi} \boldsymbol{\alpha}$ is approximately given by

$$\boldsymbol{\alpha}^T \mathbf{\Phi} \boldsymbol{\alpha} \approx \sum_{m=M_p-1}^{M_p+1} \frac{\phi^2(0,m)}{\phi(m,m)}. \tag{2.30}$$

The pitch lag M_p is chosen so as to maximize Eq. (2.30). Neglecting the off diagonal terms is justified since $r(n)$ is the formant predicted residual, and therefore has small near sample correlations. Note that this method of estimating the lag is appropriate only when the lag is estimated from the formant predicted residual.

If the pitch predictor is backward adapted, the frame over which the mean square prediction error is minimized does not correspond to the frame over which the pitch predictor is applied. Backward adapted pitch predictors perform poorly in transition regions where the pitch lag is changing rapidly. This is because the pitch lag and coefficients are too finely tuned to the analysis frame. Due to encoding delay constraints, the encoding algorithm is constrained to use backward adaptation. The solution considered here to reduce some of the adverse effects of backward pitch predictor adaptation is that of '*softening*' the pitch predictor. Softening the pitch predictor

amounts to making the predictor less finely tuned to the analysis frame. This can be done by adding uncorrelated white noise to the signal $r(n)$ or its quantized version and using this perturbed signal to solve for the pitch predictor coefficients. This approach is equivalent to adding a noise term to the diagonal elements of the covariance matrix $\mathbf{\Phi}$, and is the approach used here. Thus the diagonal elements $\phi(i,i)$ of the matrix $\mathbf{\Phi}$ are replaced by $(1+\alpha)\phi(i,i)$. This perturbed matrix is used in Eq. (2.27) to solve for the pitch predictor coefficients.

2.6 Quantizer Gain Update Algorithm

For a source whose statistics do not vary with time, there are two basic issues involved in the design of a quantizer for the source. These are (1) the *characteristic* and (2) the *dynamic range* of the quantizer. The quantizer characteristic is determined from the distribution function of the source, and the dynamic range is determined from variance of the source. When quantizing a source with time varying statistics, both the quantizer characteristic and dynamic range have to adapt to match the source statistics. Most differential encoders use quantizers with a uniform step size characteristic. Thus differential encoders with adaptive quantizers adapt the dynamic range or step size of a uniform quantizer to match the local variations in dynamic range of the prediction residual. Just as with predictor updates, both forward and backward adaptations of the step size can be used, but only backward adaptation schemes are considered in this work. Backward adaptation schemes usually take the form of a recursive variance estimator.

An adaptive quantizer can be thought of as one that normalizes its input with a variable gain, and quantizes the normalized value with a fixed quantizer. A well known backward adaptation scheme is that of *adaptive quantization with a one word memory*, also known as a *Jayant adaptive quantizer* [18]. In this scheme, the normalizing

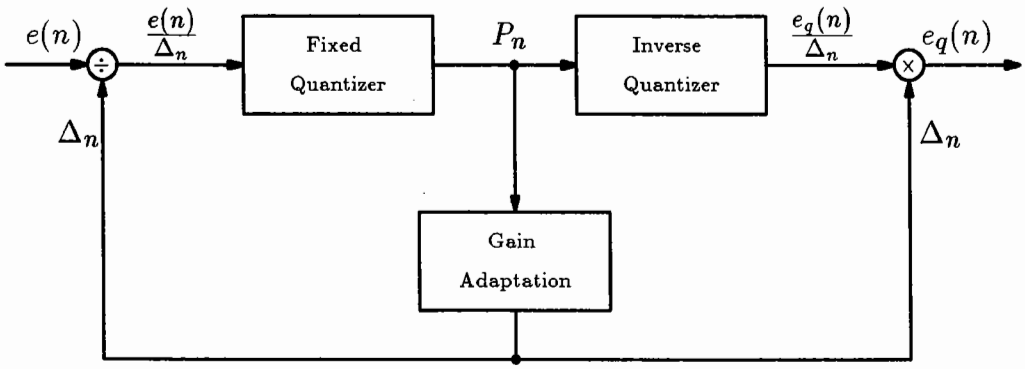


Fig. 2.5 Backward Adaptive Quantizer

variable of the quantizer is updated by multiplying by a multiplier which depends on the quantizer codeword or level chosen at the previous sampling instant. A Jayant adaptive quantizer is shown in Fig. 2.5. The variable gain at time n, Δ_n, is given by

$$\Delta_n = \Delta_{n-1} \cdot M(P_{n-1}), \tag{2.31}$$

where P_{n-1} is the quantizer output codeword at time $n-1$, and $M(\cdot)$ is the gain multiplier. The outer levels of the quantizer are assigned multiplier values greater than one, while the inner values are assigned values less than one. The quantizer range thus tends to track the dynamic range of the input.

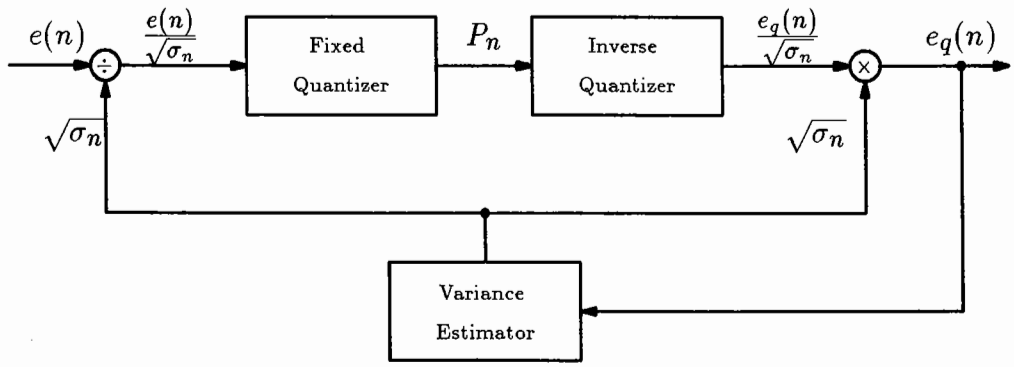


Fig. 2.6 Variance Estimating Quantizer

An alternative to the above gain adaptation scheme is that of the *variance estimating quantizer* shown in Fig. 2.6. In this scheme, the input is normalized by a

variance estimate of the input. This variance estimate is obtained using a weighted sum of the square of the past quantized outputs, $e_q^2(n)$. It is shown in [19] that there exists a Jayant Adaptive Quantizer which is equivalent to an exponential average based Variance Estimating Quantizer.

The exponential based variance estimate is updated using the following equation,

$$\sigma_{n+1} = \delta\sigma_n + (1-\delta)e_q^2(n) \quad 0 < \delta < 1. \tag{2.32}$$

The parameter δ controls the effective memory of the estimator. If the output levels of the fixed quantizer are given by $f(P_n)$, then an equivalent Jayant adaptive quantizer has a fixed quantizer with the same output levels $f(P_n)$, and multipliers assigned to these levels given by

$$M^2(P_n) = (1-\delta)f^2(P_n) + \delta \tag{2.33}$$

The above adaptive quantization schemes deal with instantaneous quantization of individual samples using a quantizer with a variable step size. However, the ideas dealing with step size adaptation will be extended later to *backward gain adaptation* with stochastic and deterministic innovations tree codes in the next chapter.

Chapter 3 Delayed Decision Encoding

Delayed Decision Coding or *Multipath Search Coding* are encoding techniques that employ encoding delay to provide a multipath search capability. The use of delayed decision in the coding of a source can provide encoding performance that is closer to the rate distortion bound for that source than the performance of a zero memory quantizer [20]. This is true even for a source that is independent and identically distributed (i.i.d). Delayed decision also provides a framework for source encoding at rates less than 1 bit per sample, i.e., at fractional bit rates per sample. (The lowest possible rate that can be achieved with instantaneous quantization is one bit per sample). Delayed decision coding techniques are broadly classified into three categories, the so-called *codebook*, multipath *tree* and multipath *trellis* coding algorithms [11]. Codebook coding is also known as *vector quantization.* In vector quantization, a block of samples are encoded together by choosing one of a set of output blocks that best matches the input block. The delay involved is therefore equal to the waiting time for gathering a block of samples of a given size for subsequent quantization. In tree and trellis coders, the possible quantized output sequences are graphically arranged in the form of tree and trellis structures respectively.

The class of delayed decision coders relevant to this study are tree coders. Differential encoding schemes and PCM schemes which have formed the basis of waveform coding of speech signals, offer candidate output sequences which can be graphically

listed in the form of a code tree. Subsequent sections will describe various tree codes associated with PCM and differential encoding schemes. Among differential encoding schemes, a distinction can be made between *deterministic tree codes* and *stochastic tree codes.* Deterministic codes are generated according to a fixed rule, and have more modest memory requirements than stochastic tree codes, although the latter class of codes give better performance.

3.1 Tree Coders

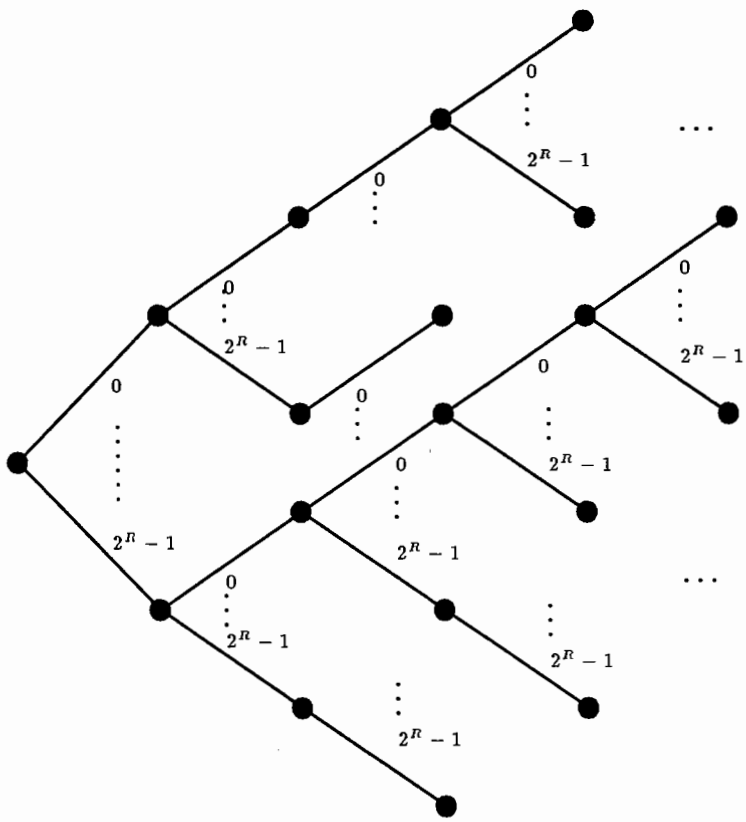


Fig. 3.1 Tree Structure of Branching Factor 2^R

In tree coders, the output sequences possess a particular graphical structure called a *tree structure.* The purpose of the next paragraph is to establish some common terminology regarding tree structures.

A tree structure is shown in Fig. 3.1. The tree consists of *branches* and *nodes*. A fixed number of branches emanate from each node, and each of these branches terminates in nodes. No two branches terminate on the same node. The number of branches emanating from each node is called the *branching factor*. The tree shown in Fig. 3.1 has a branching factor of 2^R. A tree with a branching factor of two is called a *binary tree*. The initial node of the tree is called the *root*. The *depth* of a set of nodes is the number of branches traversed along the path from the root to any one of that set of nodes. A given tree structure is specified by its branching factor.

In tree codes, each of the branches emanating from a node is assigned a unique branch number. This branch number assignment is then consistently applied throughout the tree as shown in Fig. 3.1. In general, each node of the tree is associated with β output values. In this work, interest was centered only on the case where $\beta = 1$, i.e., one output value is assigned to each node. Each depth of the tree then corresponds to a sampling instant. Thus, depth 1 corresponds to sampling instant 1, depth 2 to sampling instant 2 and so on. The values assigned to the set of nodes at a particular depth correspond to the possible output values at the time instant associated with that depth. A particular output sequence is specified by tracing along a particular path of the tree. This path is uniquely specified by the sequence of branch numbers encountered along that path. This sequence of branch numbers is called the *path map*. The encoding rate in bits per sample is given by $\frac{\beta}{log_2 b}$, where b is the branching factor of the tree structure. Knowledge of the code tree and a path map enables the decoder to uniquely decode a particular output sequence. Note that corresponding to a depth d, there are b^d possible output sequences starting from the root.

Any tree coding scheme involves two basic issues. The first is the choice of an effective code tree, and the second involves the choice of a search algorithm to search through the tree for the output sequence that best matches the input. The choice of a code tree deals with the issue of how the nodes of the tree are to be populated with

output values. A effective tree code is one that offers good (typical) candidate output sequences that are typical of the input. Among search algorithms, a distinction can be made between *single path searches* and *multipath searches.* Single path searches proceed through the code tree along one line of decisions, whereas multipath searches consider several paths in parallel, and choose among them at a later time. The fact that a multipath search can and should yield better performance than single path searches is seen from the following argument. Suppose we have two sampling instants t_1 and t_2, with $t_1 < t_2$. Consider the optimum path map sequences up to time instants t_1 and t_2 given by consideration of the input sequence up to time t_1 and t_2 respectively. Denote these path map sequences by P_1 and P_2. The path map P_2 up to time t_1 need not in general be the same as P_1. Single path searches neglect this possibility by making irreversible instantaneous decisions about the best path map. Multipath searches therefore yield better performance than single path searches.

Given a tree code, what is the optimum multipath search scheme? If the total length of the input sequence to be encoded is L samples, then the best search scheme would consider all possible output sequences of length L, i.e., the encoder performs an exhaustive search of all possible output sequences. If the branching factor of the tree is b, then there are b^L possible output sequences of length L. A two second speech segment consists of 16000 samples (assuming an 8 kHz sampling rate). With $b = 2$, the exhaustive search algorithm has to consider 2^{16000} output sequences and choose the best one, an impossibly complex and unnecessary procedure in practice. Also, the choice of L is limited in practice by an encoding delay constraint. One therefore has to consider suboptimal (non-exhaustive) but less computationally expensive multipath search schemes. A highly efficient multipath search scheme is the (M, L) algorithm. Although well described in the literature, a description of the (M, L) algorithm is given in the next section for the sake of completeness.

3.2 (M,L) Algorithm

The (M, L) algorithm is controlled by two parameters, M and L. Since this search algorithm is a multipath search algorithm, the algorithm keeps several paths of equal length in contention at any stage in the encoding process. The maximum number of paths kept is equal to M. The length of these paths is equal to L. The paths saved by the (M, L) algorithm have the property that at any given stage they stem from a single node *at most L time samples back.* The path leading up to this node represents a path through the tree that has already been decided upon. The branch number sequence for this path can therefore be transmitted to the receiver.

The encoding process is as follows. Each of the saved paths is first extended to the nodes corresponding to the next sampling instant. The cumulative errors for each of the paths are then calculated, and the extended path with the lowest error is identified. This lowest error path will extend from a single node L time samples back. The branch number for this node is then transmitted. (This corresponds to the incremental mode of operation where each search involving sequences of length $L+1$ is followed by the release of one branch number). Among the other extended paths, a distinction is made between *valid paths* and *invalid paths.* Valid paths are those that extend from the chosen node L time samples back, and invalid paths are those that do not. Among the valid paths, at most M lowest error paths are kept and saved for the next stage. This preserves the basic property of the retained paths at any stage in the encoding process. In general the number of valid paths may be less than M.

3.3 Examples of Tree Codes

This section describes the tree codes associated with some waveform coders. These tree codes can be classified into two categories, *deterministically populated* and *stochastically populated* tree codes. The tree codes associated with conventional

waveform coders such as PCM, DPCM, and ADPCM, are examples of deterministic tree codes. These tree codes have a branching factor equal to 2^R where R is the number of bits per sample used to encode the input.

3.3.1 PCM Tree Codes

Consider the case where a source is quantized on a sample by sample basis using a quantizer of some given characteristic. Suppose that the quantizer is non-adaptive, its characteristics having been matched to the source *a priori*, based on the statistics of the source. If the encoding rate is R bits per sample, then there are 2^R possible output reconstruction values at each time instant. Therefore, the possible output reconstruction sequences can be listed on a code tree with a branching factor of 2^R as in Fig. 3.1. The values assigned to the nodes are the possible output reconstruction values of the quantizer. In conventional PCM schemes, having arrived at a given node in the course of the path selection process, a choice is made among the 2^R branches emanating from that node given the input sample at that time instant. Instantaneous and irreversible decisions are therefore made at each sampling instant. This corresponds to a single path search of the PCM code tree. The search for the optimum sequence is carried out along only a single line of decisions. The conventional PCM tree code is an example of a deterministic tree code. This code is known to both the encoder and the decoder. Given the transmitted path map, the decoder can form the reconstructed output sequence.

3.3.2 Differential Encoder Tree Codes

A differential encoder consists of two main parts, a quantizer and a predictor. The quantizer in a differential encoder quantizes a prediction residual formed as the difference between an input sample and its predicted value. The quantized residual

sequence is then used to drive a synthesis filter to produce an output sequence. With differential encoders, it is useful, for reasons of clarity, to make a distinction between the *quantized residual* or *innovations code tree* and the *output* or *reconstruction code tree*. The innovations code tree is a graphical listing of the possible quantized residual sequences in time. The fact that the possible quantized residual sequences are arranged in the form of a tree is most easily seen, due to the similarity of such codes with PCM tree codes. The reconstruction code tree is obtained by passing each of the quantized residual sequences of the innovations code tree through the decoder filter. The nodes of the reconstruction code tree are then populated with the output values of the decoder filter. The decoder filter may be either fixed, backward adaptive, or forward adaptive. In all three cases, there is a one to one correspondence between the innovations code tree and the reconstruction code tree. With a backward adaptive synthesis filter, the reconstruction code tree is completely specified by the filter order, the innovations code tree, the initial conditions of the filter at the starting time of the filter, and finally the update algorithms of the parameters of the filter. With a forward adaptive filter, the reconstruction code tree is completely specified by the innovations code tree, and the side information giving the values of the synthesis filter parameters for each frame.

3.3.2.1 Innovations Tree Codes

Innovations code trees can be broadly classified into two categories, the *deterministic* and so-called *stochastically populated* trees. The innovations code tree and the decoder filter at the receiver (whether adaptive or fixed) uniquely determines the reconstruction code tree. Reconstruction code trees are therefore classified as being deterministic or stochastic depending on whether the corresponding innovations code tree is deterministic or stochastic. In both the deterministic and stochastic tree cases, the decoder receives a digital sequence corresponding to the path map sequence. Hav-

ing received the path map sequence, a table look up is then employed to determine the quantized residual sequence. This innovations sequence is then fed into a recursive synthesis filter to produce an output sequence. If the encoding rate is R bits per sample, a quantized residual sample can only take on one of 2^R values in a deterministic innovations tree. In the stochastic tree case however, this restriction is not imposed. Stochastic codes are therefore less restricted in providing good candidate output sequences. The nodes of a stochastic innovations tree are populated as follows. One starts off with a dictionary of size 2^N containing 2^N numbers. Each node in the tree is associated with a unique path map or branch number sequence from the root up to that particular node. The N least significant bits of the path map are used as an index into the dictionary. The node is then populated with the number from the dictionary associated with that index. The tree nodes are therefore populated with values from the dictionary. Such code trees are said to be stochastically populated because the dictionary is usually populated with random numbers with a certain distribution. For proper decoding of the quantized residual sequence at the receiver, identical copies of the dictionary must be stored at both the transmitter and the receiver. Thus although stochastic codes are richer than deterministic codes, they have larger storage requirements.

Gain Adaptation The innovations tree discussed above are examples of fixed gain trees. In both the deterministic and stochastic cases, it is also possible to have an adaptive gain. This adaptive gain can be either forward or backward adaptive. Only backward adaptive schemes are considered here.

A useful adaptive gain strategy in the deterministic case is the Jayant Adaptive Gain strategy. In this case, the extended nodes emanating from a given node are all assigned an adaptive gain value G. The 2^R quantizer output values are all multiplied by this gain G. The *modified alphabet* is then used to populate the extended nodes just as in the fixed gain case. Each of the extended nodes is associated with a branch

number, the branch numbers being associated with multiplier values. The extended nodes are then assigned different gain values given by the previous gain G multiplied by the respective multiplier values of the extended nodes. At any given stage, each of the paths of the deterministic tree are associated with different gain values. The multiplier values assigned to the branch numbers depend on the addressed values from the fixed alphabet. Small values are assigned multiplier values with a magnitude less than one, while large values are assigned multiplier values with magnitude greater than one.

With stochastic trees, the above gain adaptation strategies have to be modified. The fixed values addressed are all taken from a dictionary of size 2^N, where N could be as large as ten. It is not feasible to assign multipliers for each of the 2^N numbers, since this effectively doubles storage, and also involves the complication of how to assign the multipliers. The following modification can therefore be made. The effective amplitude range of the 2^N dictionary values is split up into several sub-ranges. Each of these subsections is then assigned a multiplier value. If a node has a dictionary address D, then that node is populated with the dictionary number corresponding to address D multiplied by the adaptive gain value G. This gain is then updated by the multiplier value assigned to the amplitude sub-section occupied by the dictionary number with address D.

An alternative strategy in the stochastic case is to update the gain based on an exponentially averaged variance estimate as in a variance estimating quantizer (see Chapter 2). The dictionary value assigned to a node is multiplied by the node gain G to yield an innovations sample e. The node gain is then updated according to

$$\hat{G}^2 = \delta G^2 + (1-\delta)e^2, \quad 0 < \delta < 1, \tag{3.1}$$

where $\hat{G}$ is the new gain value, and δ is a parameter that controls the effective length of the exponential window.

3.4 Brief Historical Review

Tree coding with a multipath search was first studied by Anderson and Bodie [8]. A deterministic innovations tree with a fixed gain was used together with a fixed synthesis filter. It was noted that a code tree optimal for a single path search was not necessarily optimal for a multipath search. Hence various "smoothing" techniques were used to modify the code for use with a multipath search.

Jayant and Christensen [9] studied the effects of multipath searching on code trees having a deterministic innovations code with a backward adaptive gain, and a fixed synthesis filter. Gains of 1.5 to 3 dB over a single path search were reported.

Wilson and Husain [10] studied multipath searching with a deterministic innovations tree with a forward adaptive gain, and a forward adaptive synthesis filter. The use of a fixed noise shaping filter was also studied.

Studies with a stochastic tree were reported in [12]. Both the gain and the synthesis filter were forward adaptive. The use of a stochastic trellis was studied in [11], and was found to give significant gains over the use of a deterministic trellis.

3.5 Discussion and Summary

The previous sections have described the various types of innovations and reconstruction code trees encountered in practice. The following is a summary of the properties of differential encoder code trees.

DPCM coders have a fixed predictor and quantizer. The innovations code tree is therefore a deterministic tree with a fixed gain. The mapping from the innovations tree to the reconstruction tree is done through a fixed synthesis filter.

Backward adaptive ADPCM coders have a backward adaptive predictor and quantizer. The innovations tree in this case is deterministic with an adaptive gain. A

backward adaptive synthesis filter maps the innovations tree to the reconstruction tree.

The encoding algorithm studied in this work is described in the next chapter, and uses the ideas presented in this chapter. Briefly stated, the coding scheme is a delayed decision scheme using the multipath (M, L) algorithm, and operating on a reconstruction code trees defined by a stochastically populated innovations code tree with a backward adaptive gain, and backward adaptive recursive synthesis filters.

Chapter 4 Encoding Algorithm and Computer Simulation Results

This chapter gives a detailed description of the encoding algorithm studied in this thesis. The relevant background material has been covered in the previous chapters. The encoding bit rate of the system is 16 kbits/sec, (2 bits per sample with an 8 kHz sampling rate). The encoding algorithm was simulated on a VAX 8600 computer using FORTRAN. All arithmetic operations were done using floating point arithmetic.

Both the objective and the subjective performance of the coder were analysed using six test utterances. Details of these utterances are given in the appendix. The objective measures include segmental signal-to-noise ratios plotted versus time, and also segmental signal-to-noise ratios averaged over the whole sentence based on non-overlapping 16 ms blocks. Subjective testing was carried out by conducting preference tests between sentences coded with the tree coding algorithm and sentences coded with various bit rates of log-PCM.

4.1 Description of Encoding Algorithm

Recall from the previous chapter that the reconstruction code trees for differential encoders are uniquely defined by the innovations tree and the details of operation of the decoder filter. Given the input to be encoded, the encoding proceeds by a single

path or multipath search of the reconstruction code tree. Different encoder configurations such as the conventional feedback around the quantizer configuration and the Generalized Predictor Configuration merely reflect the use of different error criteria in searching through the reconstruction code tree, irrespective of the search scheme used. The reconstruction code tree is not determined by the encoder configuration, but by the decoder configuration and the innovations tree.

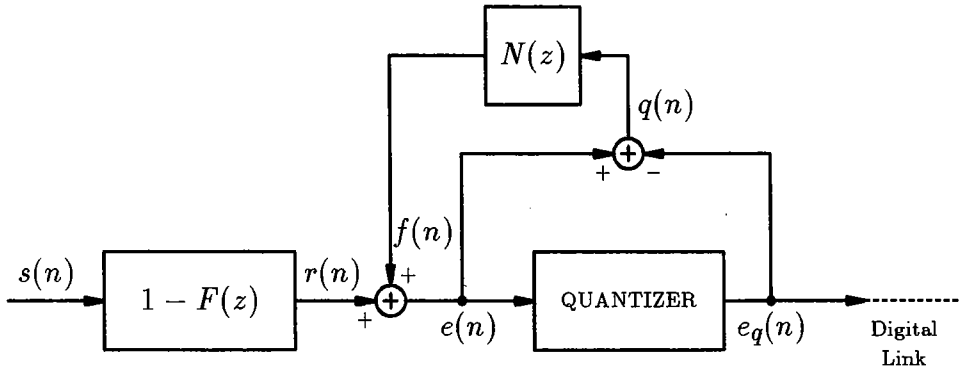
A stochastically populated innovations tree is used in this study. The tree was populated from a dictionary containing random numbers from a Laplacian pseudo-random number generator. The multipath (M, L) algorithm was then employed with the Generalized Predictive Coder configuration. This configuration allows the use of a frequency weighted error measure in choosing among various paths of the reconstruction code tree.

Two types of output codes are studied. These output codes result from the use of two different decoder configurations. The first configuration consists of an all-pole formant synthesis filter, and the second consists of a cascade of a pitch synthesis filter and a formant synthesis filter. A multipath search using the (M, L) algorithm, of these two codes, is studied.

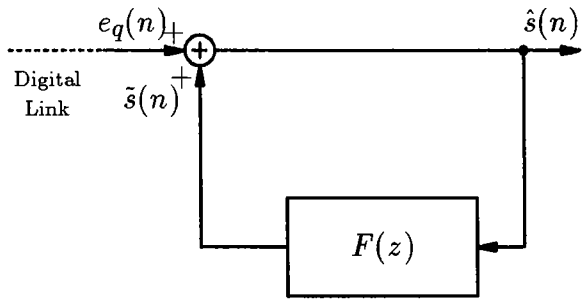
The following section describes the sequence of operations to be performed while proceeding along a single line of decisions or a single path of the code tree in question. It is then relatively straightforward to extend the coding algorithm to the multipath search case. With a multipath search, several paths are followed in parallel, the sequence of operations to be performed while proceeding along each of these paths being identical to the single path search case.

4.1.1 Single Path Search Algorithm

The encoder and decoder configurations of the Generalized Predictive Coder are shown in Fig. 4.1. The system functions of $F(z)$ and $N(z)$ are as given in Chapter 2.



ENCODER



DECODER

Fig. 4.1 Generalized Predictive Coder

A single path search with this encoder configuration is now described.

In response to an input sample $s(n)$, the prediction error filter $1 - F(z)$ forms a prediction residual $r(n)$. This residual $r(n)$ is then added to the output $f(n)$ of the noise feedback filter $N(z)$, to form the quantizer input $e(n)$. The value $f(n)$ is formed as a linear combination of the past quantization errors. The result of this addition, $e(n)$, is then quantized. Since the quantization rate is 2 bits/sample, the quantizer offers 4 possible candidate output samples, of which the one which minimizes the square of the quantization error is chosen. The following paragraph explains how the candidate quantized residual samples are obtained.

With a single path search, the chosen path map (the branch number sequence

that has already been decided upon) extends up to depth $n-1$ of both the innovations and reconstruction code trees, since there is no delayed decision involved. This path is associated with an adaptive gain value G for the innovations tree. The candidate quantized residual samples at time instant n are obtained by extending the node at depth $n-1$ of the existing path of the innovations tree, and populating the extended nodes as explained in Chapter 3. Since the branching factor of the code trees is four, there will be four extended nodes. The chosen path map up to time $n-1$ is simply the sequence of bits corresponding to the sequence of branch numbers along that path. Each extended node is therefore associated with a different path map starting from the root. The path map up to a particular extended node is obtained by shifting the chosen path map bit sequence left by two bits, and appending the branch number of the extended node. The dictionary address for each of these extended nodes is then given by the N least significant bits of the corresponding path map (assuming a dictionary size of 2^N), and the addressed dictionary numbers are multiplied by the adaptive gain G. These numbers are then used to populate the extended nodes of the innovations tree.

The values populating the extended nodes of the *reconstruction* tree are obtained by driving the synthesis filter with the corresponding samples populating the extended nodes of the innovations tree. The extended node with the lowest error is chosen, according to some well defined distortion measure. If a squared error distortion measure is used, the error for each extended node is given by the square of the difference between the input sample $s(n)$ and the value populating the extended node of the reconstruction code tree. The use of a squared error (unweighted) distortion measure amounts to having the noise feedback filter $N(z)$ equal to $F(z)$. If a frequency weighted error measure is used, a squared error distortion measure is applied to the output of a noise weighting filter $W(z)$. In this case, the noise feedback filter $N(z)$ is a bandwidth expanded version of $F(z)$. With both types of error measures, choosing

the lowest error extended node of the reconstruction tree corresponds to choosing the lowest error extended node of the innovations tree according to a *squared error* criterion, provided $N(z)$ is chosen according to the desired error measure. There is therefore no need to calculate the values populating all the extended nodes of the reconstruction tree with the aim of determining the node with the lowest error. Hence, the fact that a reconstruction tree is searched is only implicit in the coding algorithm.

Once the extended node with the lowest error is identified, the corresponding branch number is transmitted to the receiver, i.e., an instantaneous irreversible decision is made. Various quantities are then updated before proceeding to the next stage. The adaptive gain G can be adapted using one of the methods described in Chapter 3. In this work, the gain was adapted using a variance estimate of the innovations samples along the existing innovations path, based on an exponential average. Suppose the chosen extended node is populated by an innovations sample e_q. The value e_q is given by the dictionary number for the chosen node multiplied by the path gain G. The updated path gain $\hat{G}$ is the given by

$$\hat{G}^2 = \delta G^2 + (1-\delta)e_q^2, \tag{4.1}$$

where $0 \leq \delta \leq 1$. The memory of the two filters $1 - F(z)$ and $N(z)$, and the path map are then updated. Updating the path map merely consists of replacing the previous path map by the path map of the chosen extended node. The updated path map therefore extends up to depth n of both the innovations and reconstruction code trees, once the input sample $s(n)$ is quantized. From the quantized value $e_q(n)$ of $e(n)$, a local decoder forms the quantized output sample $\hat{s}(n)$. This sample populates the chosen extended node of the reconstruction code tree. (The value $e_q(n)$ is the value that populates the chosen extended node of the innovations tree). The past quantized output samples $\hat{s}(n)$ are then used via the Adaptive Lattice Algorithm to update

the reflection coefficients of the lattice filter equivalent to $1 - F(z)$. The updated reflection coefficients are then converted to transversal tap values using Eqs. (2.11). A transversal implementation results in reduced computational complexity over the lattice implementation, even though there is an overhead involved in converting the reflection coefficients to transversal tap values. Also, implementation of the noise feedback filter $N(z)$ in the reflection coefficient domain is not possible. Further, experimental evidence shows that both transversal and lattice implementations give similar performance. The filter $N(z)$ is also updated accordingly.

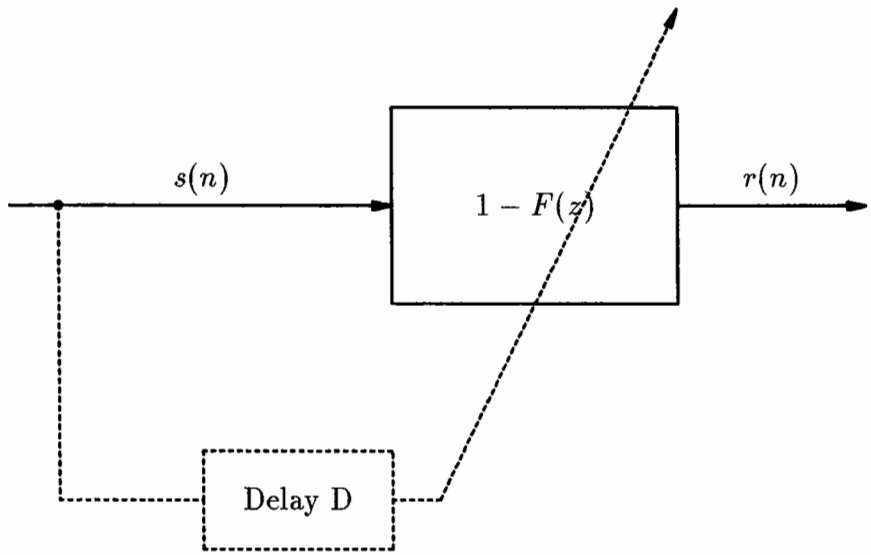


Fig. 4.2 Configuration to study the effect of a delayed update on prediction gain

Recall from Chapter 2 that the reflection coefficients are updated based on the minimization of an averaged error criterion, the error criterion being a function of the forward and backward residual energies. The update algorithm is explained in detail in Chapter 2. If the update is performed at each sampling instant using the most recent output sample $\hat{s}(n)$, the window is aligned with and includes the immediate past. Instead of using the most recent output, the update can also be done using the output a finite number of samples back, i.e., the predictor evolves via a *delayed*

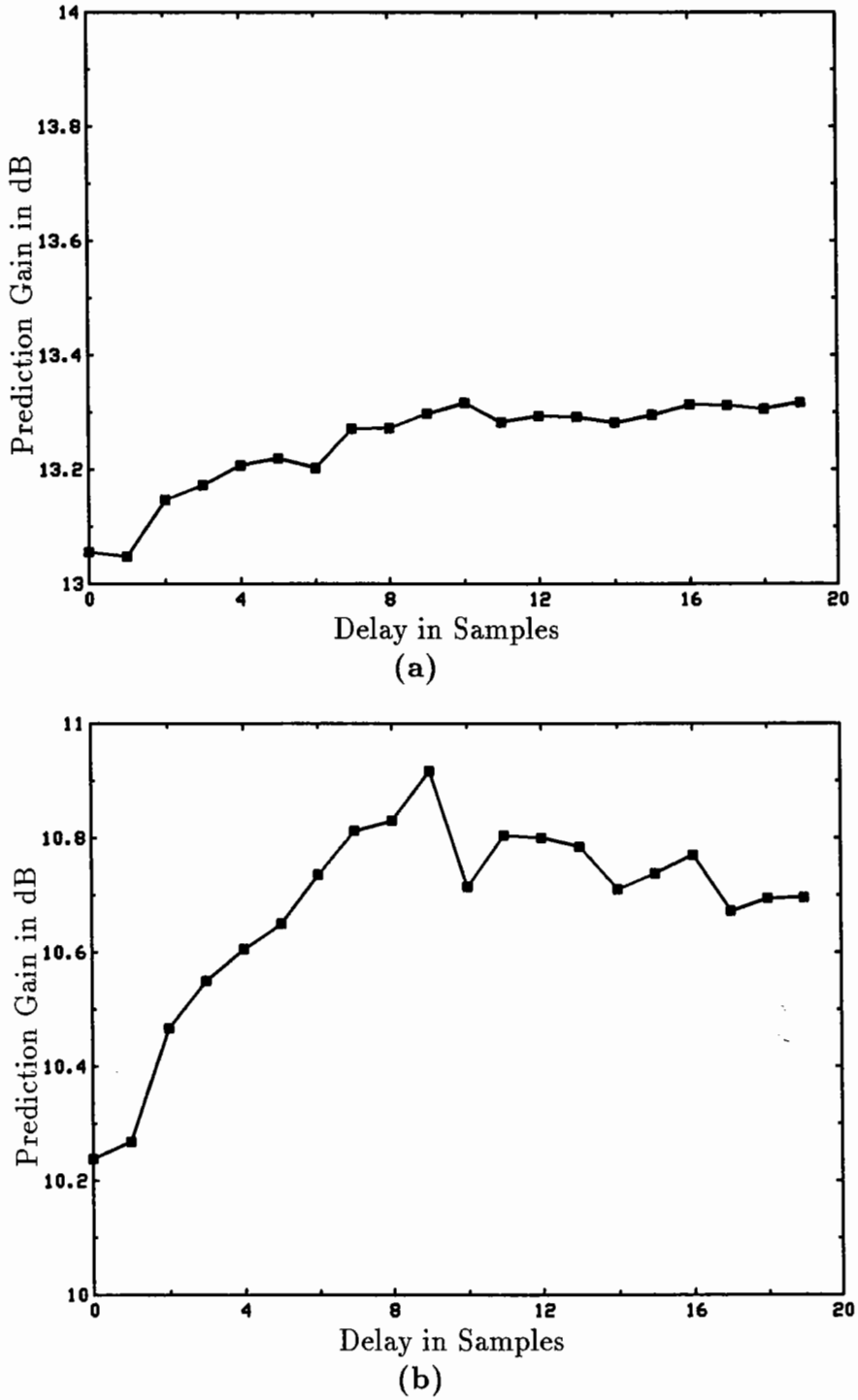


Fig. 4.3 Variation of prediction gain with delayed update for two different sentences.

update. The effect on the prediction gain of a delayed update was investigated using the configuration shown in Fig. 4.2. The adaptive lattice was used with an exponential or one-pole window, with the pole factor β equal to 0.986 and the predictor order equal to 8. The prediction error filter is updated using a delayed version of the input signal. Figure 4.3 shows the graph of prediction gain versus delay for two different sentences. Note that the maximum prediction gain is not obtained with a delay of zero as might be expected. It is interesting to note that the maximum prediction gain is obtained with a non-zero delay. A delay of about 8 samples works well. The use of a delayed update will be seen to reduce computational complexity in a delayed decision scheme.

4.1.2 Multipath Search Algorithm

With the single path search, the encoding algorithm has to keep track of various quantities in memory. These are the memories required for the adaptive lattice, memories for the filter $F(z)$ and $N(z)$, the adaptive gain for the path followed along the innovations tree, and finally the path map.

In a multipath search algorithm, the above is true for each of the paths that are being considered in parallel. In addition, the cumulative errors for each of the paths have to be tracked. The cumulative error for each extended path is the sum of the quantization errors at each node along that path. Since all the paths stem from a single node N_s a finite number of time samples back, the cumulative error for each extended path is given by the sum of cumulative error up to node N_s and the cumulative error from node N_s up to the final node of the extended path. Since only the relative errors between the various paths is important, only the cumulative errors from node N_s need be considered.

The multipath search algorithm is as follows. Each of the saved paths is first extended, and the quantized residual values for the extended nodes are found. These

values are given by the gain value for a particular path multiplied by the dictionary values for each of the nodes extended from the final node of that particular path. In response to an input sample $s(n)$, a filter output $r^j(n)$ is found for each of the saved paths. The superscript j is used to signify that the particular quantity related to the j^{th} saved path. Note that each of the saved paths is associated with a different set of transversal tap coefficients a_i^j if there is no delay involved in the update, since the filters evolve differently along different paths. Next the outputs of the noise feedback filters $f^j(n)$ are found for each path. Each of the saved paths is associated with a noise feedback filter which is a bandwidth expanded version of $F^j(z)$. The quantizer input for each extended node emanating from the saved path j is then given by the $r^j(n) + f^j(n)$, and denoted by $e^j(n)$. The quantizer error for each extended node emanating from the j^{th} saved path is given by the square of the difference between the innovations sample populating the extended node and the quantizer input $e^j(n)$. The new cumulative error for each of the extended paths is given by the previous cumulative error plus the square of the quantizer error. (Note that if there are M saved paths, there will be $4M$ extended paths). Of all the extended paths, the one with the lowest cumulative error is then identified. As explained in Chapter 3, this path will stem from a single node L time samples back, due to the property of the paths saved by the (M, L) algorithm. The branch number for this node is transmitted to the receiver. Of the remaining extended paths, the best M valid paths are kept for the next stage, and the rest of the paths are discarded. (Note that the number of valid paths could be less than M). The memories for the retained paths are then updated for the next stage, as discussed in the next paragraph.

The innovations path gains are updated according to Eq. (4.1). The path maps for each of the retained paths are updated as in the single path case. If the filters are updated without a delayed update, then the filter coefficients for each path are updated using the sample populating the final node of the new saved path of the

reconstruction tree, i.e., the most recent output sample along that path. If there are more than M valid paths, M sets of filters will have to be updated. Thus, a multipath search without a delayed filter update involves the use of memory for each path to track the evolution of filters along that path, and also extra computation to update the filters along these paths. Use of a delayed update can reduce computational complexity of the search as explained next.

A delayed update along any path amounts to updating the filters for that path by a reconstruction sample a fixed number of time samples back along that path. (Updating without a delay amounts to using the most recent reconstruction sample along that path). By the property of the paths saved by the (M, L) algorithm, all the saved paths stem from a single released node L time samples back. This node is common to all the saved paths. If the synthesis filters are updated with a delay of L samples, then the filters for each of the saved paths evolve in an identical way. Updating with a delay of L samples amounts to updating with the most recently released output sample. All the saved paths are therefore associated with a single filter $F(z)$ which evolves via a delayed update of L samples. Furthermore, nothing is lost by way of prediction gain, as seen from the plots presented earlier (see Fig. 4.3). Note however that the use of a delayed update necessitates the use of buffers to maintain synchronization. In decoding the reconstruction sample from the released innovations sample, the synthesis filter should be the inverse of the prediction error filter that was used L samples back. Therefore, L sets of predictor coefficients have to be kept in memory and updated at each time instant. If L is less than M, the use of a delayed update also results in a reduction in memory requirements.

4.1.3 Multipath Search with Pitch Prediction

The multipath search with pitch prediction is very similar to the multipath search with formant prediction. The block diagram of a Generalized Predictive Coder with

pitch prediction is shown in Fig. 4.4. The decoder configuration consists of a pitch synthesis filter in cascade with a formant synthesis filter.

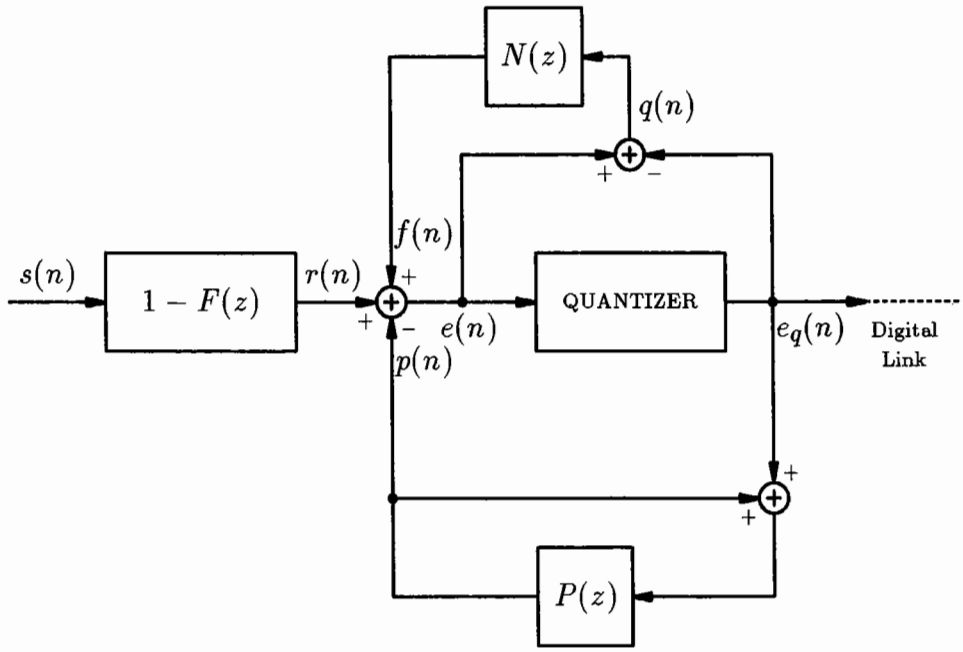
In response to an input sample $s(n)$, a formant residual $r^j(n)$ and a pitch prediction $p^j(n)$ is found for each of the saved paths. Again in this case, the superscript j associated with a quantity signifies that it is associated with the j^{th} saved path. The pitch prediction is a linear combination of the past quantized formant residual samples. Since the pitch lags are constrained to be in the range of 20 to 120 samples, and an L value of 20 or greater was not used, the quantized residual samples which form the pitch prediction all correspond to already released samples.

The quantizer input for each saved path is formed as $r^j(n) + f^j(n) - p^j(n)$, and denoted by $e^j(n)$. The cumulative errors for each extended path are then formed as in the previous section. The lowest cumulative error path is identified, and the corresponding branch number L time samples back is released. The best M valid paths are then kept for the next stage.

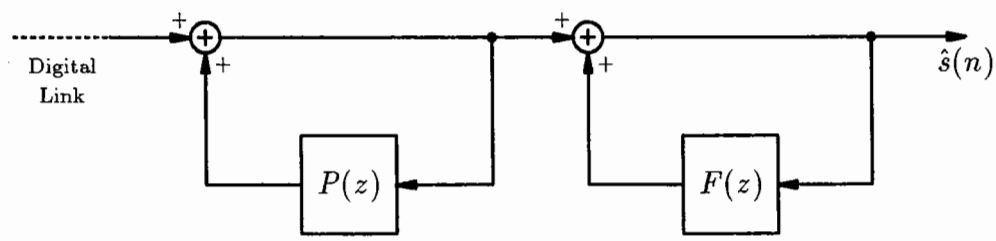
A local decoder forms the released output sample L time samples back. The decoder in this case consists of a pitch synthesis filter followed by a formant synthesis filter. The formant filter is then updated using the released output sample. The pitch predictor is updated using the past released quantized residual samples, i.e., using the output of the pitch synthesis filter. The update method is as explained in Chapter 2. Buffering is also required in this case for the pitch filter in order to maintain synchronization.

4.2 Initialization

With the (M, L) algorithm, a certain number of paths are retained at each stage. At the start of the encoder operation, the coding algorithm requires the existence of a certain number of saved paths. The saved paths should reflect the 'rest state' of



ENCODER



DECODER

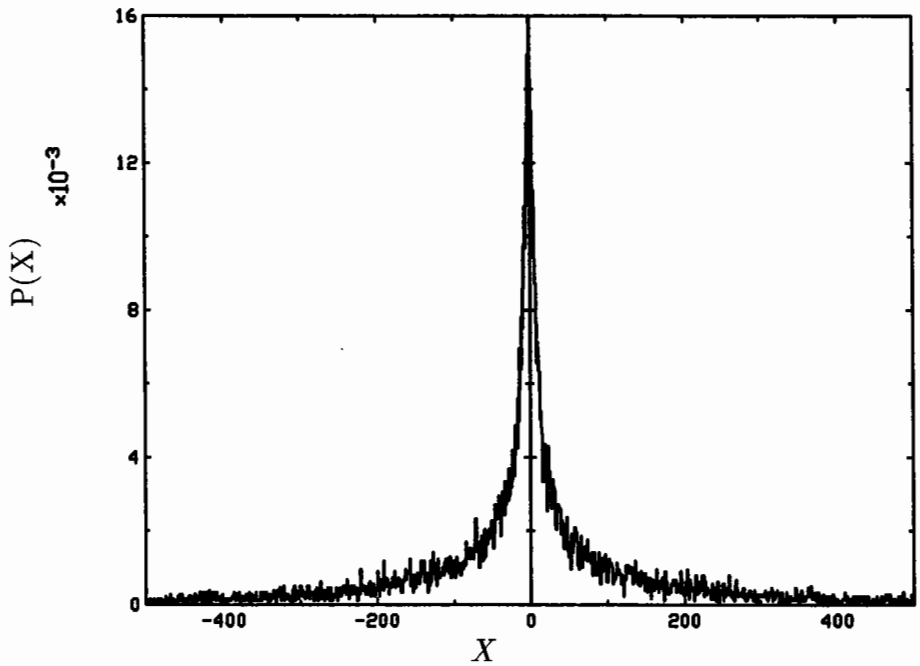
Fig. 4.4 Generalized Predictive Coder with Pitch Prediction

the coder before start up. The most obvious choice for the initial saved paths is to have one saved innovations path of length L samples, whose nodes are all populated with samples of value zero. The initial gain value for this path can assigned any reasonable value. The initial gain value was not found to be critical in this work. The filter coefficients for the initial path are also set to zero. Note that the initial innovations path and the gain value are assumed to be known to the decoder.

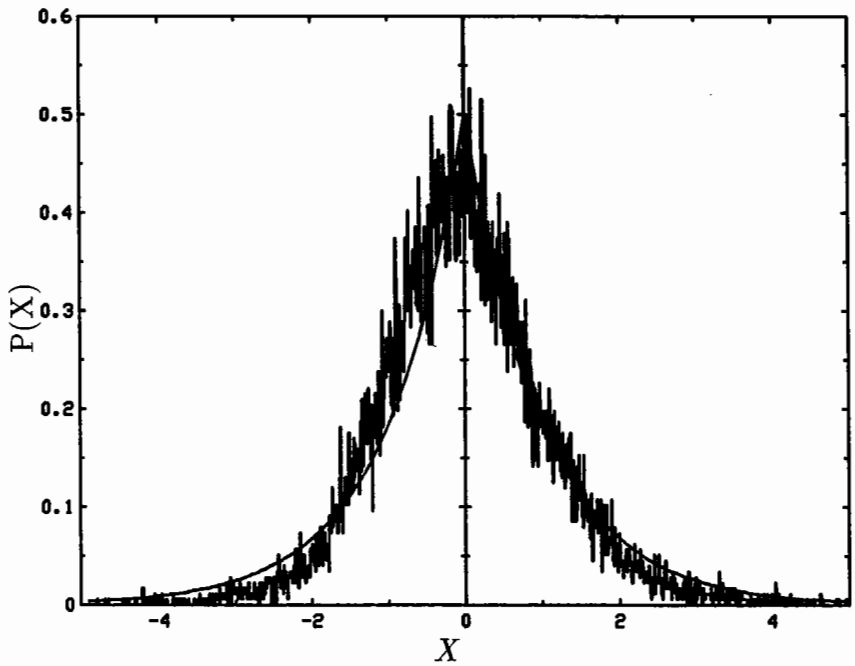
4.3 Population of the Dictionary

In the course of the encoding process, the values extracted from the dictionary are multiplied by an adaptive gain and then used to populate the nodes of the innovations tree. The values populating the nodes of the innovations tree should in some way reflect the statistics of residual or prediction error samples encountered in practice. The statistics that one may consider are those of long-term average distribution, and also statistical dependence between samples. Accounting for the correlations between residual samples is difficult in practice. For unvoiced speech, the residual samples are invariably noise-like, with little or no correlation between samples. Populating the dictionary with variates having the same long-term distribution is therefore quite adequate. However with voiced speech, the residual signal contains pitch pulses if only a formant predictor is used. The residual signal in this case cannot be considered to be noise-like, due to the presence of pitch pulses.

With the use of pitch prediction, the pitch pulses can be removed, resulting in an overall noise-like residual signal. Effective pitch prediction, however, necessitates the use of an adaptation strategy. With the use of a backward adaptation strategy for the pitch predictor, not all the pitch pulses can be removed since the frame over which the coefficient analysis is done is not the frame over which the pitch prediction is carried out. With backward adaptation, pitch predictors are too finely tuned to the analysis frame to fully compensate for rapid changes in pitch lags in transition regions. A backward adaptive pitch prediction scheme is used here. The innovations tree will therefore be populated with values which account for the long-term average distribution of formant predicted residual signals encountered in practice. A long-term averaged histogram of formant predicted residual samples was obtained using several speech sentences. The histogram is shown in Fig. 4.5a. The long-term distribution can be seen to be approximately Laplacian in nature. Since the dictionary values are



(a) Long-term residual density



(b) Long-term gain normalized density — solid line shows the Laplacian Density Function

Fig. 4.5 Long-term Histograms of Formant Residual Samples

multiplied by a gain before being used to populate the nodes of the innovations tree, the dictionary values should have the same distribution as gain normalized residual samples. Fig. 4.5b shows the long-term averaged distribution of gain normalized residual samples. (Normalization was done by dividing each residual sample by a variance estimate of past residuals). The distribution of gain normalized residual samples is seen to be approximately Laplacian in nature. The dictionary values in this study are therefore random numbers from a Laplacian random number generator.

4.4 Objective Test Results

The objective test results of the coding algorithm are given in this section. The objective results are given in terms of plots of segmental signal-to-noise ratio (segSNR) versus time, and in terms of averaged segSNR values, the average being taken over a whole sentence. In the former, the signal-to-noise ratio in decibels (dB) is calculated for successive overlapping blocks of 100 samples (12.5 ms blocks), thus yielding a graphical display of the time variations in signal to noise ratio. Averaged segSNR values are obtained by calculating the signal-to-noise ratios in dB for non-overlapping blocks 16 ms in duration, and then averaging the SNR values over all the blocks in a sentence. Thus,

$$\text{segSNR} = \frac{1}{N} \sum_{i} \text{SNR}_i, \tag{4.2}$$

where SNR_i is the signal-to-noise ratio for the i^{th} block in dB, there being N such blocks. Such an objective measure is a more realistic indication of the performance of an algorithm than an SNR value taken over a whole sentence since it takes into account certain regions in a sentence where the signal-to-noise ratios are high.

Before presenting the results, the various parameters and factors controlling the performance of the algorithm will be reviewed, in order to establish the notation.

For the formant prediction error filter and the formant synthesis filter, the relevant parameters are the predictor order P, and details controlling the update of the predictor. The factors controlling the update are the window used in the adaptive lattice, and the stability constant γ of the adaptive lattice. With an exponential window, the pole factor β is fixed at 0.986. A later section considers the use of a new class of windows. The stability constant γ of the lattice was fixed at 0.5 to ensure stability of the synthesis filter.

The factors controlling the pitch prediction process are the order of the pitch predictor, and the analysis method used to solve for the coefficients and the pitch lag. The update rate of the pitch predictor is also a relevant parameter.

The innovations tree is determined by the size of the dictionary, the distribution of the variates populating the dictionary, and the effective length of the exponential window used to obtain the variance estimate for the gain adaptation. The effective length of the window is controlled by the value of δ in Eq. (4.1). It was found experimentally that a δ value of 0.86 gave the best results. Laplacian random numbers were used to populate the dictionary. The dictionary size N_D also determines the nature of the innovations code tree, and was fixed at 4096.

Finally, the multipath search is controlled by the values of M (the number of paths kept in contention at each stage) and L (the length of these paths).

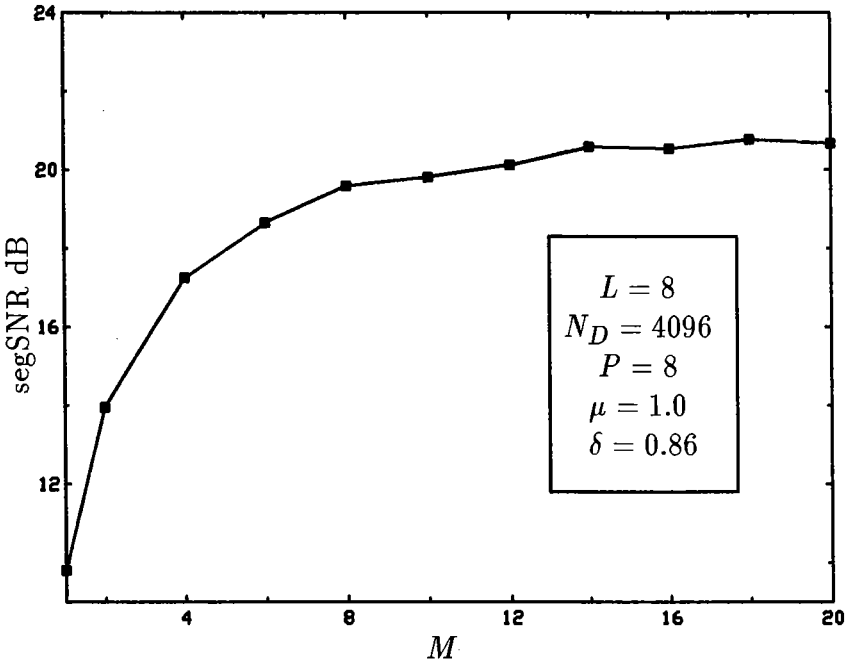
4.4.1 Performance with M

This section shows the performance of the system with M. The parameter M is the number of paths kept in contention at any stage in the encoding process. The two plots in Fig. 4.6 show the averaged segSNR values for two sentences, CATF8 and CATM8. The value of L in both cases is fixed at 8. Other sentences show very similar behaviour. The figures show that the segSNR values increase rapidly with M

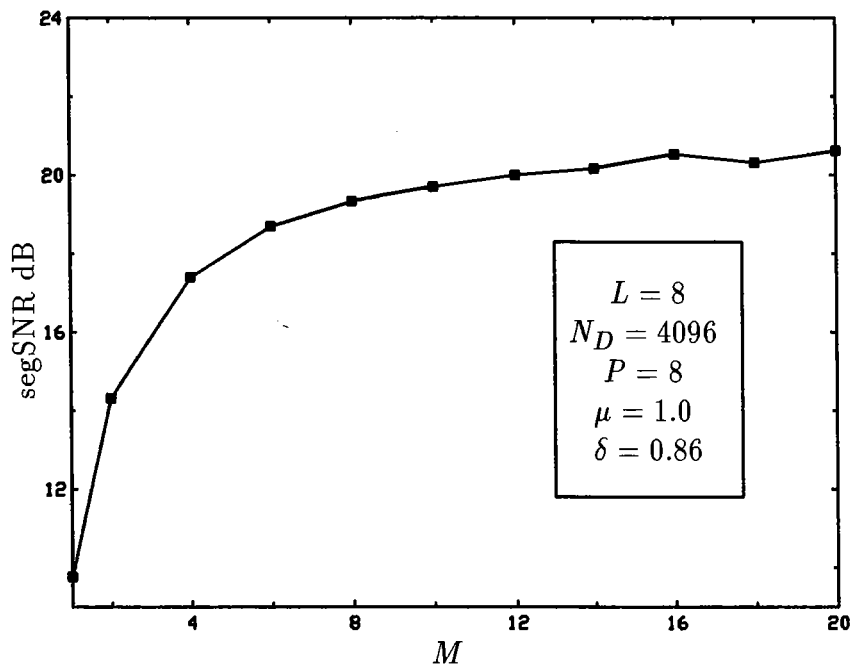
at first, and then finally saturate with M to an almost constant value. Saturation is achieved with M equal to about 16. Note that the branching factor of the code tree is equal to four, and that with L equal to eight, there are 4^8 paths available, of which a maximum of M are considered at any stage. Saturation in performance with M is therefore attained with a value of M that is very much less than the total number of paths available. This is in keeping with the results of multipath search using the (M, L) algorithm of differential encoder code trees [8][9][10][11].

In the work of Anderson and Bodie [8], and Jayant and Christensen [9], it was found that most of the performance from the multipath search is obtained with M equal to about 4. Furthermore, it was found in [8] that good codes required a larger value of M for saturation than poorer codes, i.e., good codes demand a larger M to find the better paths of the code tree. This explains why saturation in performance with M occurs at a large value of $M = 16$, in the present work. The stochastic code studied here is a much 'richer' code than the deterministic code inherent in conventional forward and backward adaptive differential encoders.

Jayant and Christensen [9], have shown that the use of gain adaptation with a deterministic innovations code tree yields a performance gain that is independent of the gain obtained with the use of a multipath search. Chan and Anderson [21] have shown that adaptation of the predictor and quantizer yields an increase in performance that is independent of the increase in performance obtained through a multipath search. With a single path search, increases in signal-to-noise ratio were obtained with predictor adaptation and with quantizer step size adaptation. The use of a multipath search in all these cases produced a further increase in signal-to-noise ratio. Gains obtained through adaptation are complementary to the gains obtained through a multipath search. The above observations can be summed up by saying that with deterministic codes, inclusion of gain adaptation with the innovations tree, and predictor adaptation results in a reconstruction tree code that gives improved performance with



(a)



(b)

Fig. 4.6 Performance of the system with M

both single path and multipath searches. An obvious question that now arises is whether the use of a gain adaptive stochastic innovations tree provides an independent source of gain as compared with a gain adaptive deterministic innovations code tree. This question can be answered in the negative by observing that the segSNR value obtained for M=1 is several dB below that obtained with conventional single path ADPCM with a deterministic tree. Use of a stochastic innovations tree therefore yields a reconstruction code that does not perform well with a single path search.

Figures 4.7 and 4.8 show spectrograms of some original and coded sentences and plots of signal-to-noise ratio. These were obtained with $M = 16$ and $L = 8$. Note that the formant and pitch structures are well preserved in the coded signal. Signal-to-noise ratios of 10–15 dB are attained in fricative segments.

4.4.2 Performance with L

This section investigates the performance of the system with L. The parameter L is the length of the paths considered in the multipath search. Figure 4.9 shows plots of segSNR with L for fixed M, ($M = 16$). A saturating trend in performance with L is seen. The value of L at which saturation occurs is somewhat related to the predictor order as seen from Fig. 4.9a and Fig. 4.9b. With an eighth order predictor, performance is seen to saturate at L equal to about 10. With a third order predictor, performance saturates at a lower value of L, this time at L equal to about 6.

The value of L does not seem to be critical, provided it is high enough to account for the predictor order, although the dependence on the saturation point with predictor order is not very strong.

4.4.3 Performance with Predictor order

Figure 4.10 shows objective performance with predictor order. The performance

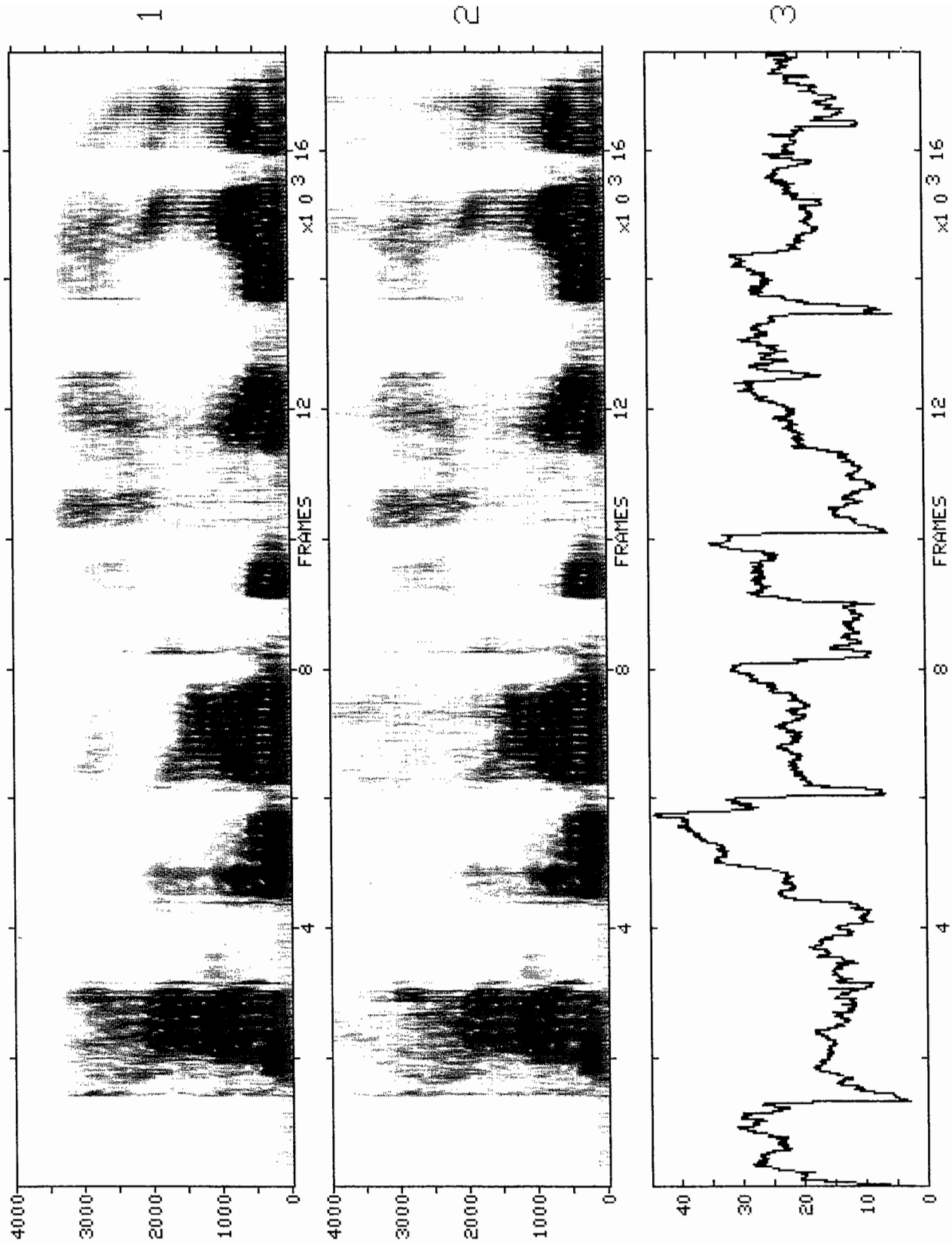


Fig. 4.7 (1) Spectrogram of original sentence (CATF8), (2) Spectrogram of coded sentence, (3) Plot of segSNR in dB.

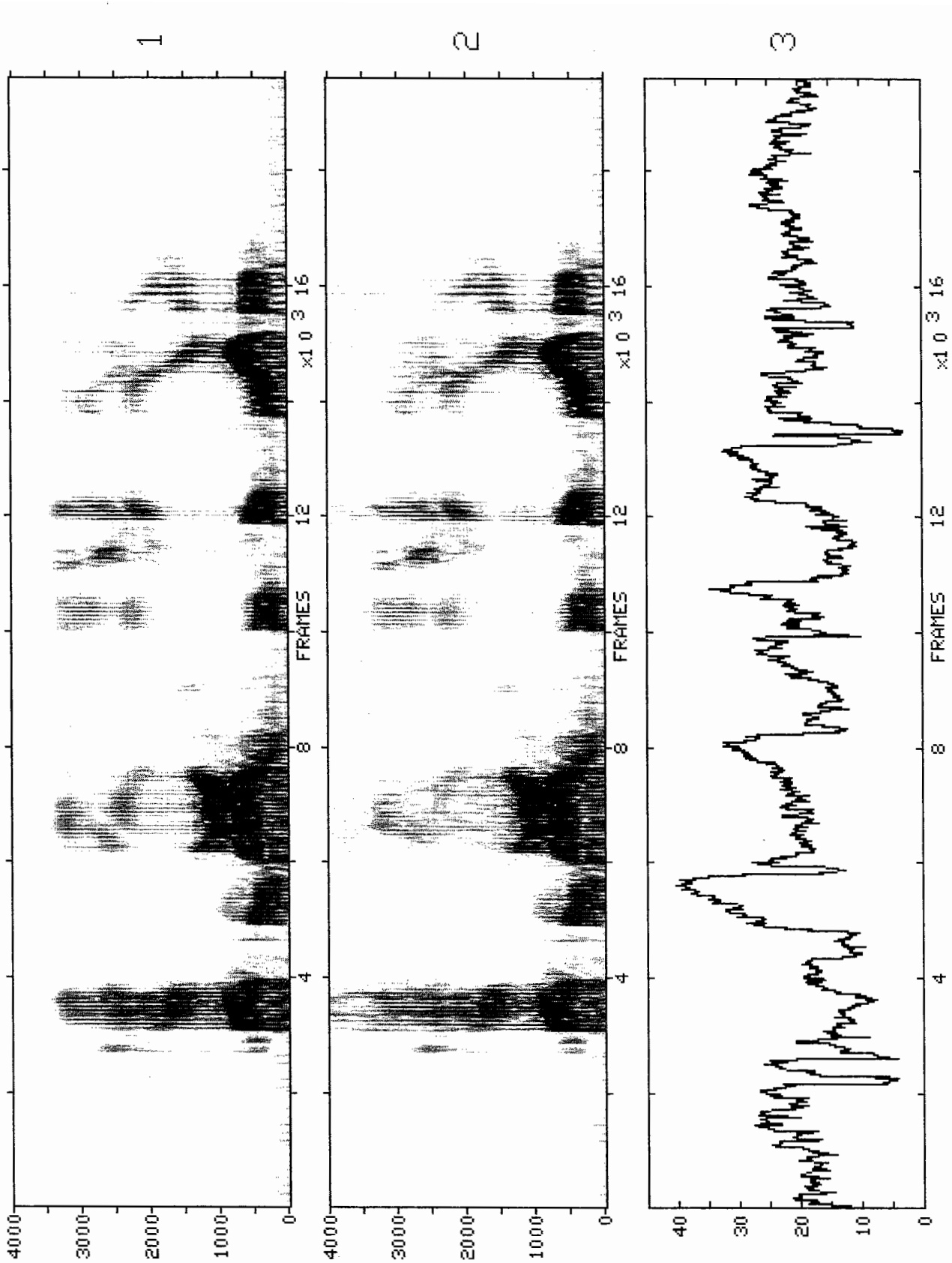
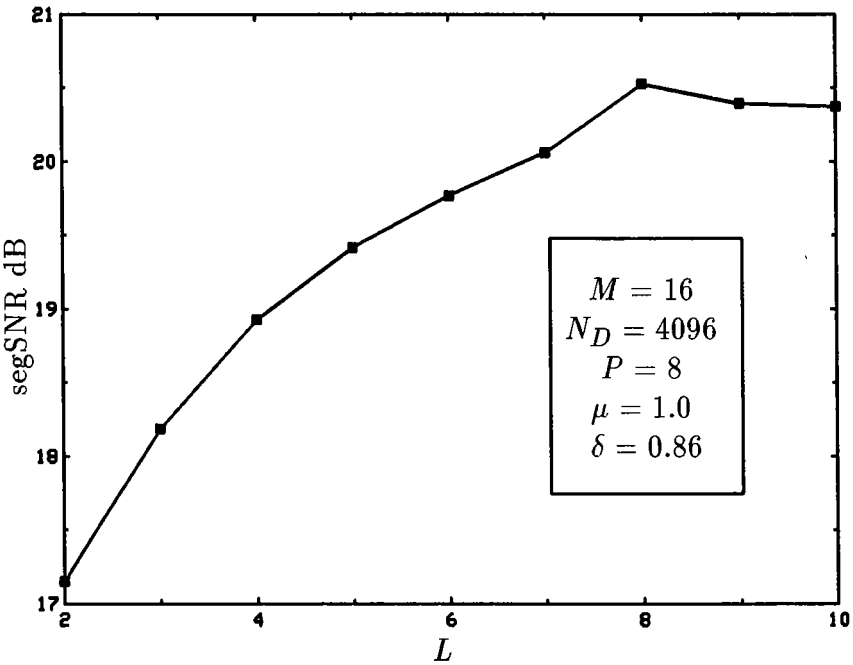
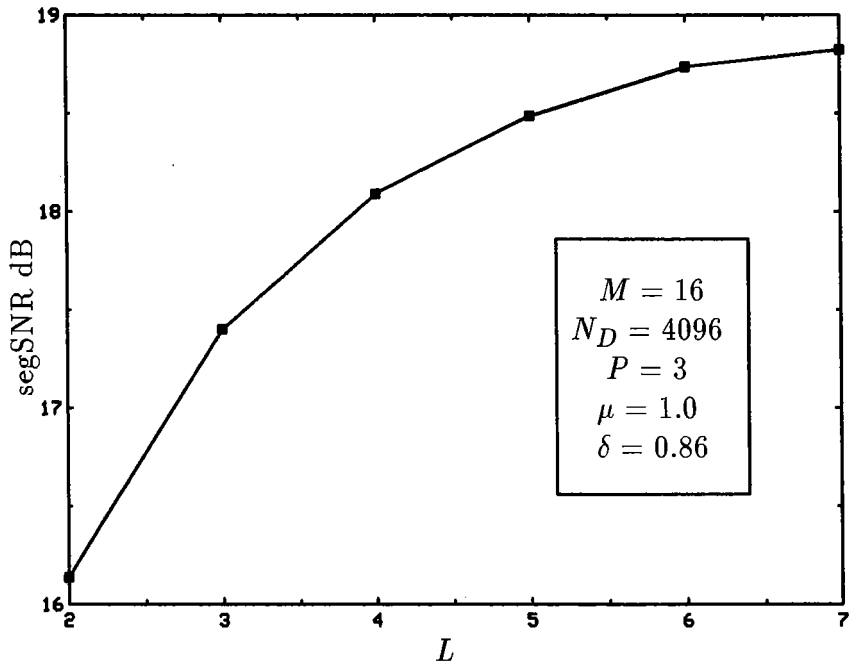


Fig. 4.8 (1) Spectrogram of original sentence (CATM8), (2) Spectrogram of coded sentence, (3) Plot of segSNR in dB.



(a) With an eighth order formant predictor



(b) With a third order formant predictor

Fig. 4.9 Performance of the system with L

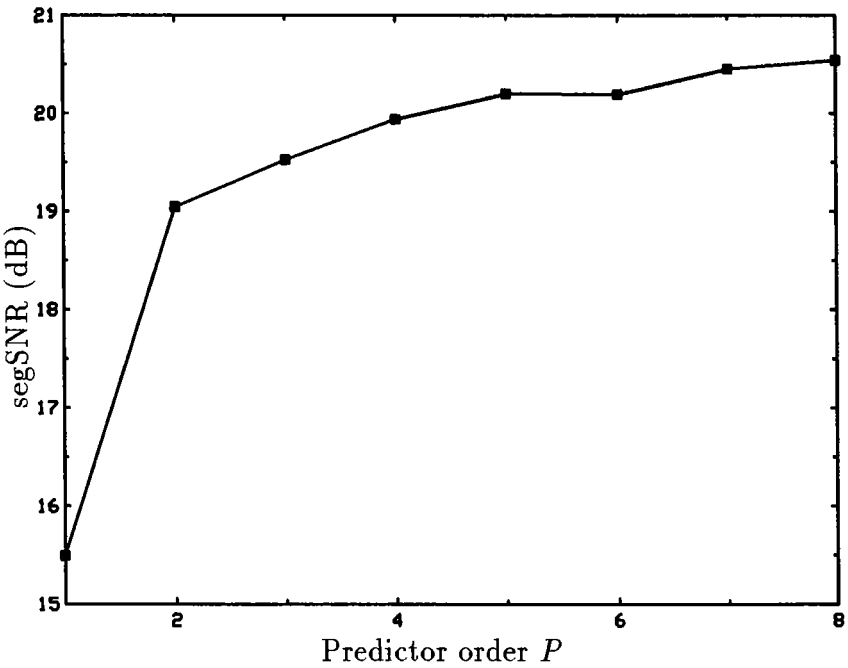


Fig. 4.10 Performance of system with predictor order

drops gradually with P and then falls off sharply at a value of P equal to one.

Subjectively, degradation in the output speech becomes more audible for values of P less than eight. No improvement in performance is obtained for values of P greater than eight.

4.5 Adaptive Lattice with Pole-Zero Windows

Recall from Chapter 2 that using windows corresponding to impulse responses of recursive digital filters leads to recursive update equations for the Adaptive Lattice. One-pole windows are simple and have the property of weighting the energy more over the immediate past than the distant past. However it does not have the property of uniformly weighting the energy over a narrow range of samples as would a Hamming window or a rectangular window. Two-pole windows are better approximations to a Hamming-type window than exponential windows. However, two-pole windows are not the best choice for the present application since the range of samples over

which the residual energy is minimized is considerably lagged from the time instant of application of the predictor. The predictor therefore will not adequately track the different stationarity modes of the input.

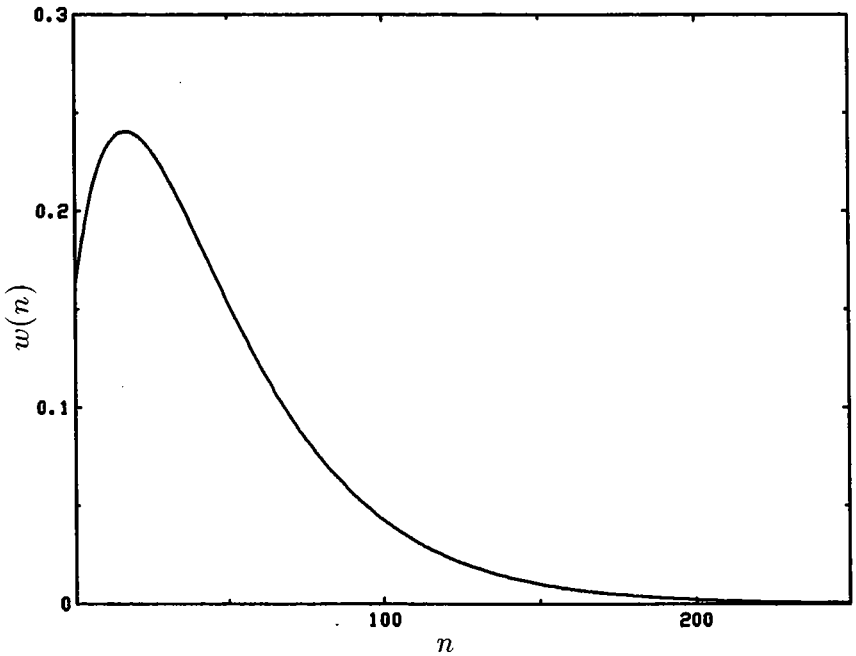


Fig. 4.11 Window obtained with $\beta_1 = 0.97, \beta_2 = 0.95$, and $a = 0.85$.

A class of windows that are impulse responses of filters having one zero and two poles were tried. The window $w(n)$ is obtained as the sum of two decaying exponential sequences as given below:

$$w(n) = (\beta_1)^n - a(\beta_2)^n \tag{4.3}$$

The values of β_1, β_2 and a were chosen so as to ensure that $w(n) \geq 0$ for $n \geq 0$. The z-transform of $w(n)$ is given by

$$W(z) = \frac{(1-a) + (a\beta_1 - \beta_2)z^{-1}}{1 - (\beta_1 + \beta_2)z^{-1} + \beta_1\beta_2 z^{-2}} \tag{4.4}$$

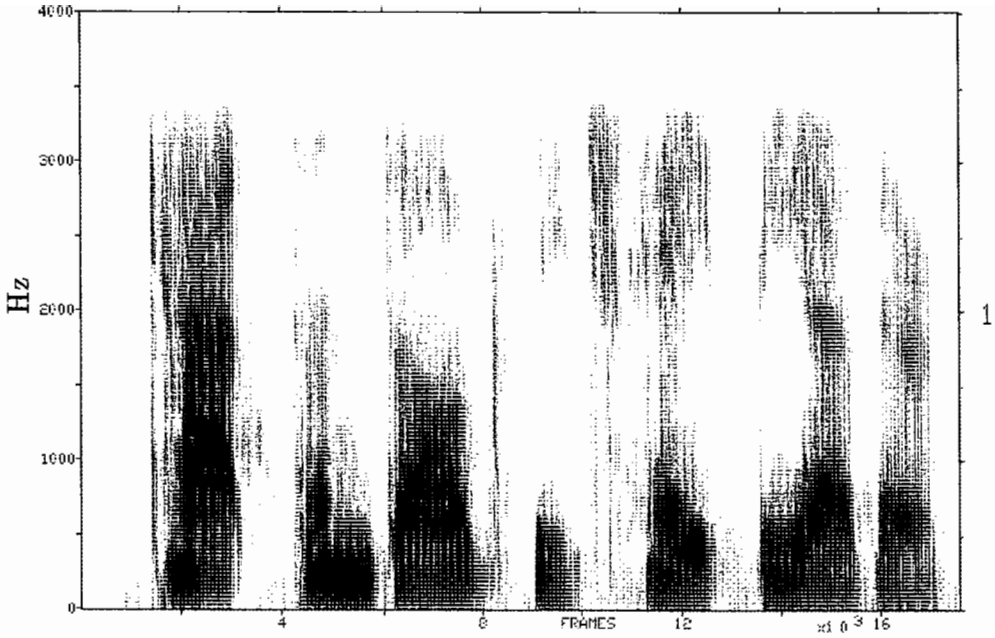
An example of such a window is shown in Fig. 4.11. By carefully controlling the parameters, a window shape that is intermediate to the one and two-pole-types is

obtained. For β_1 and β_2 equal to 0.97 and 0.95 respectively, choosing a close to zero results on a one-pole-type window, and choosing a close to one results in a two-pole-type window. Use of the window shown in Fig. 4.11 resulted in a 0.1–0.3 dB increase in signal-to-noise ratio, for various sentences. Subjectively, this window gives an output speech quality that is 'crisper' than that obtained with a one-pole window. However for simplicity, only a one-pole window was used in the compilation of test results.

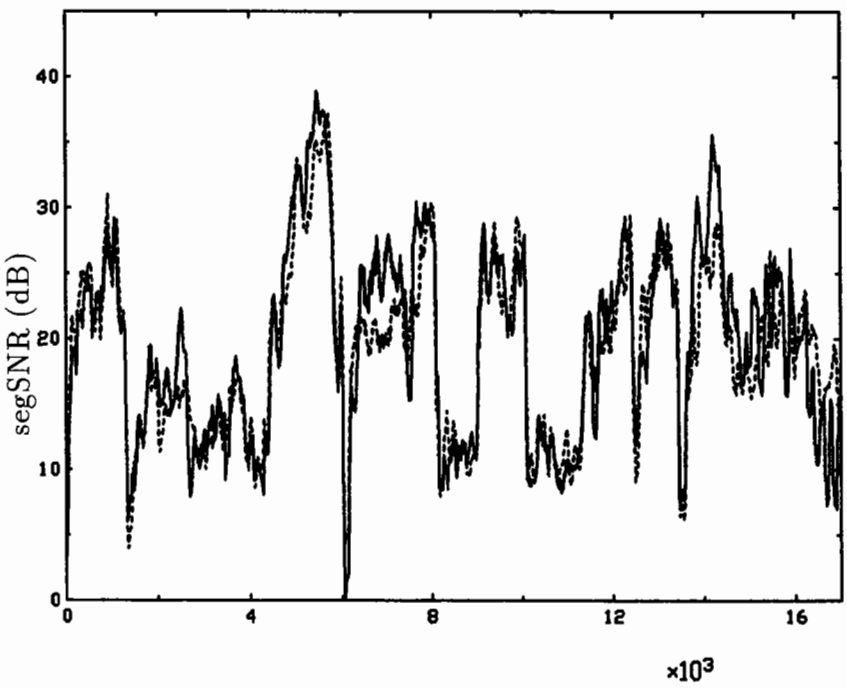
4.6 Results With a Backward Adaptive Pitch Predictor

This section presents results of a multipath search with an encoder configuration that incorporates a backward adaptive 3-tap pitch predictor. The pitch predictor is updated every 20 samples by analyzing the most recent released formant residual sequence. Updating the pitch predictor at every sample involves a great deal of computation, and does not give any improvement in performance over a slower update rate of 20 samples. The coefficients are calculated by minimizing the prediction error over a frame of samples. The best performance was obtained with a frame length of about 100 samples. The pitch lags are constrained to lie between 20 and 120 samples. This range is enough to account for a wide range of speakers. Details of the update method are given in Chapter 2.

Since the pitch predictor is backward adaptive, pitch prediction is not carried out on the analysis frame. Because of changing pitch lags during actual speech, and due to transitions from unvoiced to voiced speech, not all the pitch pulses in the formant residual are removed in practice. Pitch pulses are removed only in steady state voiced segments during which the pitch period is relatively constant. Fig. 4.12 shows a plot of segmental signal-to-noise ratio both with and without the use of pitch prediction. The accompanying spectrogram facilitates identification of voiced and unvoiced segments.



(a) Spectrogram of sentence CATF8



(b) The dashed curve shows segSNR without pitch prediction. The solid line shows the segSNR with pitch prediction and no noise addition.

Fig. 4.12 segSNR with Pitch Prediction

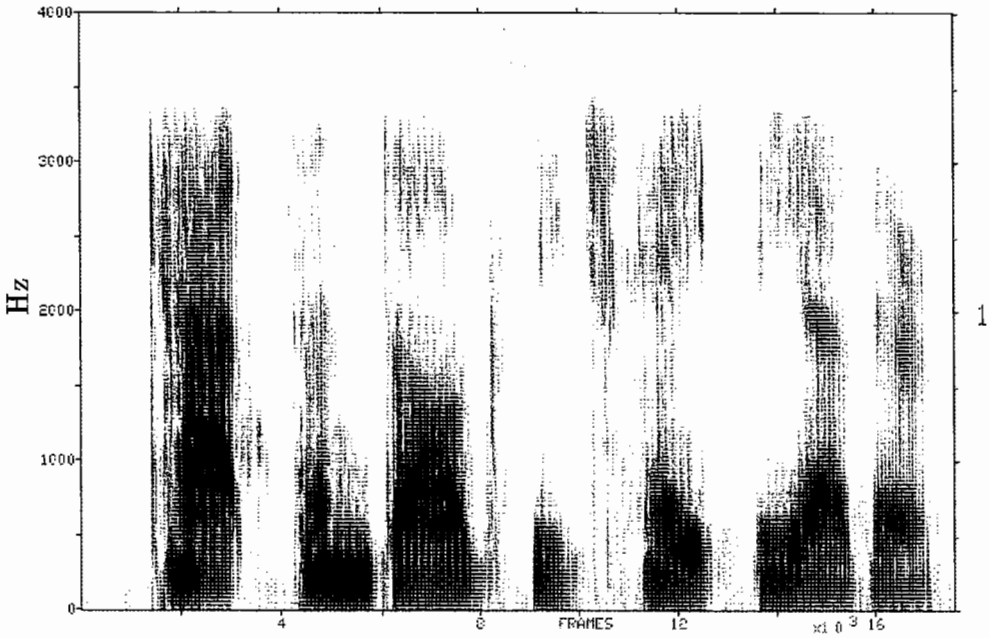
Note that the signal-to-noise ratio with the pitch predictor is increased during voiced segments and remains about the same during unvoiced segments.

Because pitch filtering is not done in the analysis frame, some extra pitch peaks are added to the formant residual in regions where the lag is changing rapidly. The negative effects of this can be lessened by *'softening'* the pitch predictor, i.e., by making it less finely tuned to the analysis frame. One way of accomplishing this is to modify the diagonal elements of the covariance matrix used to evaluate the predictor coefficients. In particular, each diagonal element $\phi(i,i)$ is replaced by $(1+\alpha)\phi(i,i)$, for $i = 1,2,3$. Solving for the coefficients using a covariance matrix perturbed in this way is equivalent to adding white noise to the formant residual and then evaluating its covariance matrix and solving for the pitch predictor coefficients. Solving for the coefficients in this way has the effect of 'softening' the pitch predictor and reducing some of the adverse effects of backward adaptation. A value of α equal to 0.01 gives good results. Too high a value results in degradation in the output speech. Fig. 4.13 shows plots of segmental SNR using a pitch predictor with and without noise addition. Noise addition is seen to improve the SNR even further.

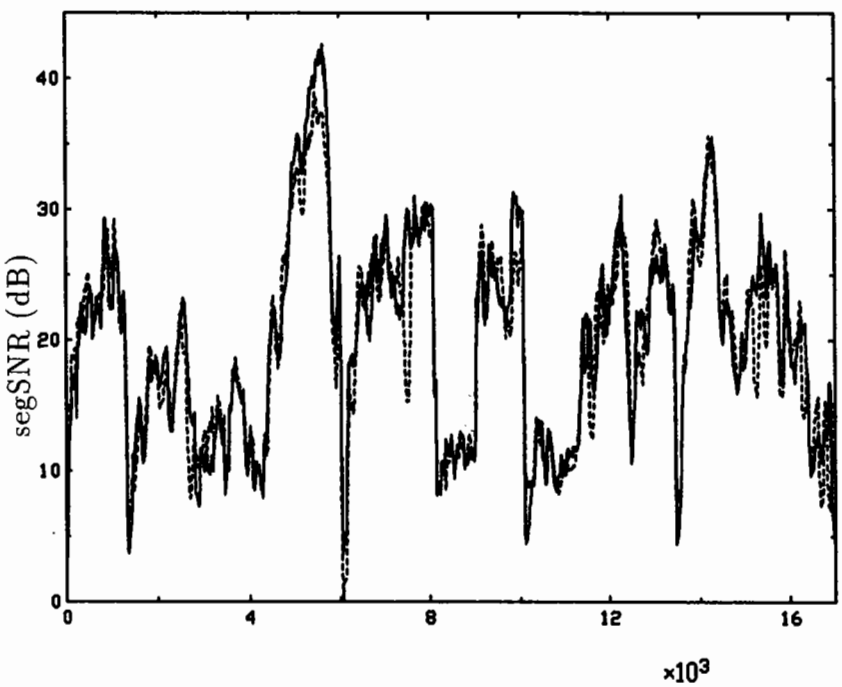
Pitch prediction together with the predictor softening approach yields up to 5 dB increase in signal-to-noise ratio in certain voiced segments as seen from Fig. 4.14 and Fig. 4.15. Subjectively, the use of pitch prediction produces a cleaner sounding coded speech signal. The reason for the improvement obtained with pitch prediction is that pitch prediction renders the overall prediction error signal more noise-like. The method used for populating the dictionary is more optimal in quantizing such a signal.

4.7 Subjective Test Results

A subjective test of the coder was carried out by conducting a preference test



(a) Spectrogram of sentence CATF8



(b) The dashed curve shows segSNR with pitch prediction but without noise addition. The solid line shows the segSNR with pitch prediction and noise addition.

Fig. 4.13 segSNR with Pitch Prediction and Noise Addition

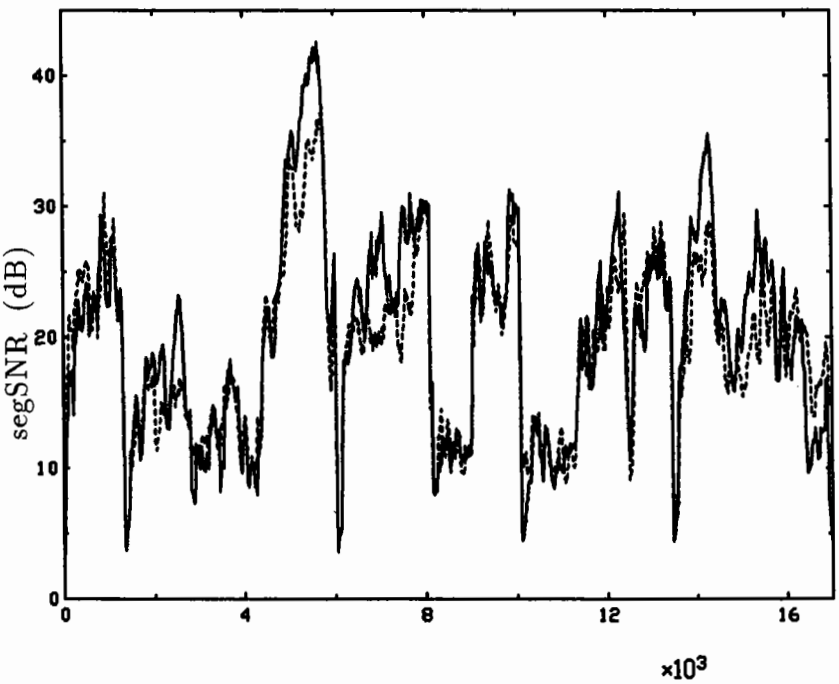
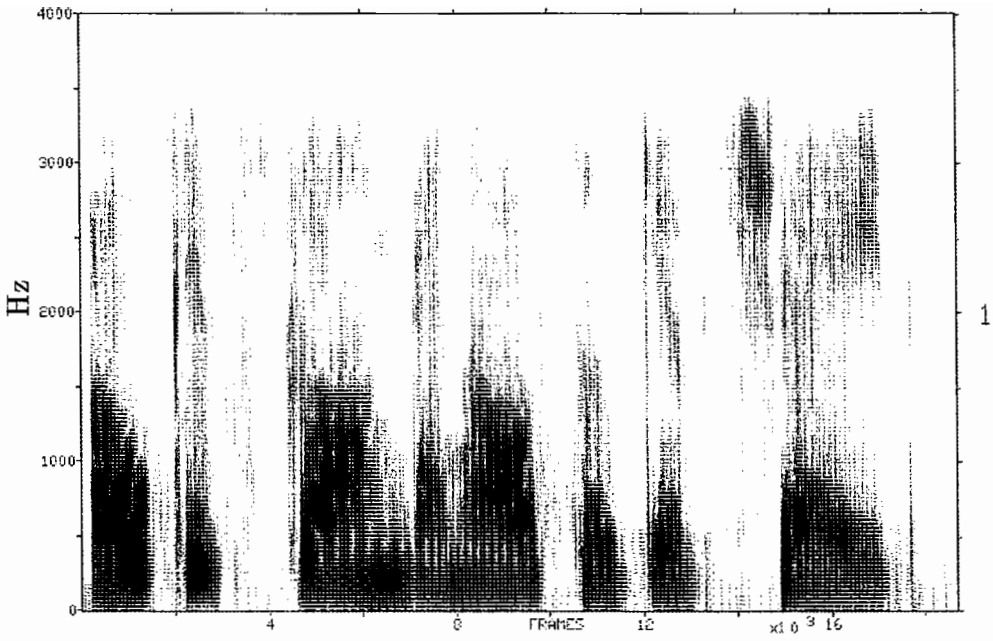
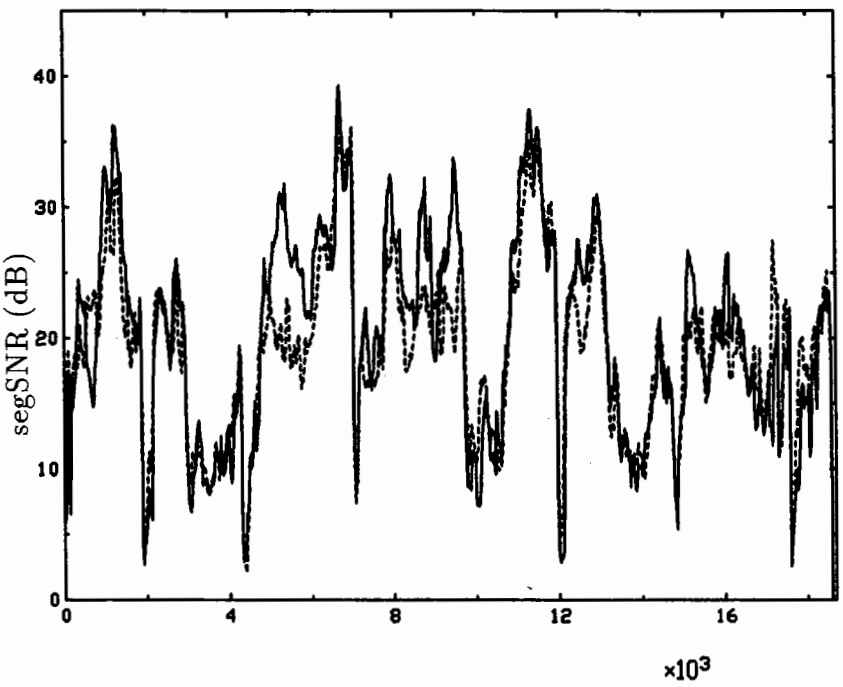


Fig. 4.14 The dashed curve shows segSNR without pitch prediction. The solid line shows the segSNR with pitch prediction and noise addition.

between tree coded files and log-PCM coded files of various bit rates. The sentences used were CATF8, CATM8, OAKF8, OAKM8, THVF8, and THVM8 (see Appendix A). The tests were conducted with high quality headphones in a soundproof acoustic chamber. The listeners consisted of mostly 'naive listeners' (students working in areas other than speech coding) and a few trained listeners (those working in the speech coding area). The 'naive listeners' were more inclined towards the tree coded speech files than 'trained listeners'. The following parameters were used for the tree coded sentences; pitch prediction with noise addition ($\alpha = 0.01$), noise shaping ($\mu = 0.85$), tree searching with $M = 16$ and $L = 8$, and finally an eighth order formant predictor using a one-pole window ($\beta = 0.986$). The tree coded sentences were compared with 5, 6, 7, and 8 bit/sample log-PCM coded sentences. The subjective test file consisted of pairs of sentences, a tree coded version and a log-PCM version. A preference test



(a) Spectrogram of sentence OAKF8



(b) The dashed curve shows segSNR without pitch prediction. The solid line shows the segSNR with pitch prediction and noise addition.

Fig. 4.15 segSNR with Pitch Prediction and Noise Addition for sentence OAKF8

between a tree coded sentence and say a 5 bit/sample log-PCM version was done by including two pairs in the test file of the tree coded version and the log-PCM version, in reverse order. This is done for each sentence, and with 5, 6, 7, and 8 bits/sample log-PCM coded sentences. The various test pairs were randomly ordered in the test file.

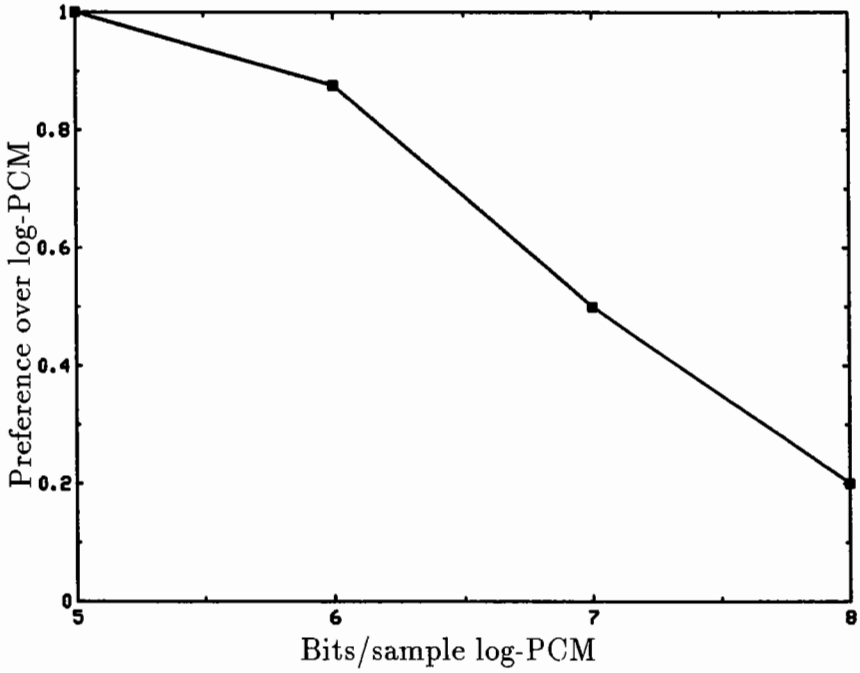


Fig. 4.16 Preference curve over log-PCM

Results of the subjective tests are shown in Fig. 4.16. The vertical axis shows the fraction of times that the tree coded sentences were preferred over the corresponding log-PCM coded sentences. For example, tree coded sentences were preferred over 5 bits/sample log-PCM coded sentences every time. The equal preference point is achieved at about 7 bit/sample log-PCM. One can therefore conclude that the tree coding scheme achieves a level of subjective quality equal to 7 bit/sample log-PCM.

Chapter 5 Conclusion and Recommendations For Future Research

The aim of this study was to address the problem of low delay toll quality coding of speech at a rate of 16 kbits/sec. The importance of digital encoding and in particular low-delay coding, was explained in Chapter 1.

Waveform coding has formed the basis for high quality digital coding of speech. Among waveform coders, the emphasis has been on predictive or differential encoding schemes. Adaptation of the quantizer and predictor in such schemes is vital in achieving high quality at low bit rates. However achieving high quality at a rate of 16 kbits/sec requires the use of forward adaptation schemes which introduce a large amount of encoding delay, this being in violation of the objectives. Backward adaptation schemes enable encoder operation with near zero encoding delay, but coder performance with such schemes at a low rate of 16 kbits/sec is poor.

The technique of multipath searching of differential encoder tree codes has played an important role in improving the performance of differential encoders at low bit rates, without introducing high encoding delays. Multipath searching introduces only a small amount of encoding delay, in the acceptable range of 1 to 2 ms. Previous work has centered on multipath searching with the (M, L) algorithm, of tree codes given

by a deterministic innovations tree, and either a fixed or forward adaptive formant synthesis filter. A description of various differential encoder tree codes together with a summary of previous work in tree coding was given in Chapter 3.

In this thesis, a stochastically populated innovations code tree was used. The tree is populated with random numbers having a Laplacian distribution, and further, included the effect of a backward adaptive gain along each path. The output code is then given as a mapping through synthesis filters, of the innovations code. A particular path of the output tree is obtained by passing the corresponding path of the innovations tree through the synthesis filter. Two synthesis filter configurations were considered. The first consisted of a backward adaptive all-pole formant synthesis filter. Although the filter is an adaptive lattice, its implementation was carried out in transversal form. The second configuration consisted of a cascade of a pitch synthesis filter and a formant synthesis filter. The pitch filter reconstructs the fine structure of the speech spectrum, and the formant filter inserts the speech spectral envelope. The pitch filter was backward adaptive, in that the estimate of the pitch lag, and the coefficients of the pitch filter were determined from the past quantized formant residual signal. Addition of a small perturbation term to the diagonal elements of the pitch covariance matrix in the calculation of the pitch predictor coefficients, was seen to reduce some of the adverse effects of backward adaptation of the pitch filter. The use of a pitch filter gives almost 5 dB improvement in signal-to-noise ratio in certain steady state voiced segments. With both types of synthesis filter configurations, encoder configurations that permit the use of a frequency weighted error measure were used to reduce subjective loudness of the output noise.

The output code tree was searched using the multipath (M, L) search algorithm. Most of the performance from the code is obtained with M and L values of 16 and 8 respectively. Segmental SNR values of about 20 dB were obtained with a squared error distortion measure. The tree code is one that gives good performance with a

multipath search, although performance with a single path search is very poor. With a single path search, segSNR values fall below 12 dB.

Subjective tests consisting of comparisons between log-PCM coded sentences and tree coded sentences were carried out. The tests show that listeners have an equal preference between the tree coded sentences and the corresponding 7 bit/sample log-PCM coded versions.

To conclude, subjective quality equivalent to 7 bit/sample log-PCM was obtained with an encoding delay of 1 ms (8 sample delay) together with a modest amount of tree searching (16 paths kept in contention at each stage), all at an encoding rate of 16 kbits/sec. The coding algorithm is in conformity with the constraints of (1) low encoding delay, (2) high quality, and (3) low to medium encoding bit rate — 16 kbits/sec.

5.1 Recommendations for Future Research

The dictionary was populated using random numbers with a Laplacian distribution. This method of population is optimal in coding the output of an i.i.d. Laplacian source. Since the Laplacian distribution only reflects the long-term distribution of residual samples, the method used for populating the dictionary is not optimal. Changing both the distribution and the gain adaptation strategy to take into account the behaviour of the residual during voiced segments and plosives, for example, might be looked into. The use of a complementary innovations code might also be investigated. Complementary code trees are pseudo-stochastic code trees, in which a set of extended nodes are populated with non-independent values. Complementary code trees therefore have a controlled amount of structure imbedded.

The encoding algorithm is suitable for use over the switched telephone network. However, full integration of the coding algorithm into the telephone network would

involve considerations of (1) transmission of voice band data signals, (2) tandeming, and (3) efficient transcoding to existing log-PCM techniques. The performance of the algorithm in the presence of channel errors should be investigated. Possible modifications of the algorithm to improve performance in the presence of non-ideal channel conditions should be looked into.

Appendix A. Details of Speech Data Base

The speech sentences were first low pass filtered to 5.5 kHz (1 dB down at 5 kHz and 40 dB down at 10 kHz) and then sampled at 20 kHz. The speech data files were obtained by first digitally filtering the 20 kHz sampled data and then changing the sampling rate to 8 kHz. The digital filter had a pass band between 0 and 3200 Hz, and a stop band between 3350 and 5000 Hz.

The speech sentences used are :

(1) CATF8 - "Cats and dogs each hate the other". (Female speaker)

(2) CATM8 - "Cats and dogs each hate the other". (Male speaker)

(3) OAKF8 - "Oak is strong and also gives shade" (Female speaker)

(4) OAKM8 - "Oak is strong and also gives shade" (Male speaker)

(5) THVF8 - "Thieves who rob friends deserve jail" (Female speaker)

(6) THVM8 - "Thieves who rob friends deserve jail" (Male speaker)

References

1. M. R. Aaron, "Digital Communications — The Silent (R)evolution", *IEEE Communications Magazine*, January, 1979.
2. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, 1978.
3. N. S. Jayant and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, 1984.
4. J. L. Flanagan, M. R. Schroeder, B. S. Atal, R. E. Crochiere, N. S. Jayant and J. M. Tribolet, "Speech coding", *IEEE Trans. Communications*, vol. COM-27, April 1979.
5. T. Nishitani, S. Aikoh, T.Araseki, K. Ozawa, and R. Maruta, "A 32 kbits/sec Toll Quality ADPCM Codec using Single Chip Signal Processor", *Proc. Int. Conf. Acoust., Speech, Signal Processing*, May, 1982.
6. W. R. Daumer, P. Mermelstein, X. Maitre, and I. Tokizawa, "Overview of the ADPCM Coding Algorithm", *IEEE Global Telecommunications Conference*, Nov. 1984.
7. X. Maitre and T. Aoyama, "Speech Coding Activities Within CCITT : Status and Trends", *Proc. Int. Conf. Acoust., Speech, Signal Processing*, April 1982.
8. J. B. Anderson, and J. B. Bodie, "Tree Encoding of Speech", *IEEE Transactions on Information Theory*, vol. IT-21, No. 4, July, 1975.
9. N. S. Jayant, and S. A. Christensen, "Tree Encoding of Speech Using the (M,L)-Algorithm and Adaptive Quantization", *IEEE Transactions on Communications*, vol. COM-26, No. 9, September, 1978.
10. S. G. Wilson, and S. Husain, "Adaptive Tree Encoding of Speech at 8000 Bits/s with a Frequency-Weighted Error Criterion", *IEEE Transactions on Communications*, vol. COM-27, No. 1, January, 1979.
11. 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.
12. T. Svendsen, "Tree Coding of the LPC Residual", *Proc. Int. Conf. Acoust., Speech, Signal Processing*, April 1984.
13. 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.
14. N. S. Jayant, "Digital Coding of Speech Waveforms: PCM, DPCM, and DM Quantizers", *Proceedings of the IEEE*, vol. 62, No. 5, May, 1974.
15. J. D. Gibson, "Adaptive Prediction in Speech Differential Encoding Systems", *Proceedings of the IEEE*, Vol. 68, No. 4, April, 1980.
16. J. I. Makhoul and L. K. Cosell, "Adaptive Lattice Analysis of Speech", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-29, No. 3, June, 1981.

17. R. P. Ramachandran, "Pitch Filtering in Adaptive Predictive Coding of Speech", *M. Eng Thesis, Department of Elec. Eng., McGill University*, March, 1986.

18. N. S. Jayant, "Adaptive Quantization with a One Word Memory", *Bell Syst. Tech. J.*, Vol. 52, Sept. 1973.

19. D. L. Cohn and J. L. Melsa, "The Relationship between an Adaptive Quantizer and a Variance Estimator", *IEEE Transactions on Information Theory*, November, 1975.

20. T. Berger, *Rate Distortion Theory, A Mathematical Basis for Data Compression*, Englewood Cliffs, N.J., Prentice-Hall 1971.

21. H. C. Chan and J. B. Anderson, "Adaptivity Versus Tree Searching in DPCM", *IEEE Transactions on Communications*, vol. COM-30, No. 5, May, 1982.