1-1-1992

On reducing the coding-delay and computational complexity in an innovations-assisted linear predictive speech coder

Sunil Satyamurti
Iowa State University

On reducing the coding-delay and computational complexity in an innovations-assisted linear predictive speech coder

by

Sunil Satyamurti

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Electrical Engineering and Computer Engineering
Major: Electrical Engineering

Iowa State University
Ames, Iowa
1992

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Dr. T. V. Ramabadran for his guidance and encouragement without which this project wouldn't have been possible. I would also like to thank Dr. Venkatagiri and Mr. Sridhar Nath who found time, inspite of their busy schedule, to do the subjectve tests on the speech coder.

CHAPTER 1. INTRODUCTION

Speech coding at low bit-rates has been of great research interest in the past decade. This need for low bit-rate coding was triggered by the increase in information exchange over various communication networks. Aided by the rapid development in hardware technology and digital signal processing techniques, several speech coding techniques which offer high degree of compressibility with minimal loss in quality were developed for applications like cellular communication, satellite communication, voice paging, voice mail and voice response.

Digital Coding of Speech

Speech signals when in digital form offer several advantages over analog signals. The major advantage is that digital signals are less sensitive to transmission noise but of course need a higher bandwidth for transmission. Digital signals are also easy to regenerate and store. They can also be error-protected, encrypted and multiplexed. Of course, all these advantages do not come for free. Digitizing the speech signals introduces some coding distortion. So efficient ways of coding which would keep the distortion minimal for a given coding rate are necessary. This coding rate is measured by the number of bits transmitted per second or simply bit-rate. Distortion is measured in two ways.

(i) An objective signal-to-noise ratio (SNR). This is defined as the ratio of the energy of the actual speech signal and the energy in the error after encoding/decoding the speech signal.

(ii) A subjective measure such as *mean opinion score* (MOS). This involves formal listening tests by several trained personnel who rate the perceptual quality on a scale from 1 to 5 (1 corresponds to poor and 5 corresponds to excellent).

Over the years several coding methods have been developed to keep the bit-rate low but at the same time maintain quality of speech so that a given bandwidth can be effectively shared. This becomes very important in applications like cellular communication, satellite communication, voice storage, voice response and voice mail where as many number of users should be fit into the channel (or storage medium) to keep the cost low. These coders can be broadly classified as waveform coders and vocoders. The waveform coders try to reproduce the input speech waveform as closely as possible. They are usually of high quality, simple in structure, but do not offer high compression rates. The other group of coders, vocoders, make use of a signal-analysis procedure and extract significant features from the speech signal and use them to synthesize the speech signal. The quality of speech produced in these coders is not good. The spoken speech sound can be identified but the speaker usually cannot be identified. These coders are relatively complex but they offer high compression of speech (2.4-4.8Kbps).

Coding Schemes to Reduce Bit-Rate

As mentioned above coders can be either waveform coders or Vocoders. A brief discussion of these types is given below.

Waveform Coders

Waveform coders are simple in design and offer high quality speech. Pulse code modulation (PCM) is an example of a waveform coder where the input speech signal is sampled, quantized and transmitted. The quantizer is designed in such a way that lower amplitude signals are given finer quantization resolution than higher amplitude signals. At the receiver this process is just reversed and we get the speech signal back. This offers high quality speech at 64Kbps. Differential PCM (DPCM) uses a linear predictor to remove the redundancy in the speech signal by predicting the present sample as a linear combination of the past samples. The difference between the actual and the predicted value of the speech signal, also called the prediction residual, is quantized and transmitted. Fewer number of bits are enough for this since the prediction residual will be of lesser energy. If the parameters of this linear predictor are updated with time then it is called an adaptive DPCM (ADPCM). This can achieve a bit-rate of 32Kbps when maintaining the same quality as a PCM. When quantized to achieve a bit-rate lower than 32Kbps the performance falls rapidly.

Vocoders

Vocoders do not try to match the speech waveform. Instead they extract parameters of a model which would represent the actual speech and transmit them. The design of a vocoder is based on the human speech production system. The human speech production system comprises the glottis which provides the excitation and the vocal tract which modulates the excitation to produce different sounds. The vocoder is based on this system. Vocoders can provide very low bit-rate (2.4-4.8Kbps) transmission rates but the quality is very poor. These vocoders are also of high complexity.

Recently, coders which utilize the characteristics of both waveform coders and vocoders have been developed. These coders, called the hybrid coders, are capable of providing high-quality speech while still maintaining a low bit-rate of 8-16Kbps. These hybrid coders are based on the excitation-modulation model of the vocoder but they try to reproduce the actual waveform as closely as possible by transmitting some information about the excitation also. A variety of hybrid coders have been suggested and each one of them has its own way of providing information about the excitation. In the code excited linear predictive (CELP) coder [22] the information is provided as an entry from a carefully chosen codebook containing preselected, Gaussian white random sequences. In a multi-pulse linear predictive (MPLP) coder [23], it is a sequence of appropriately located and scaled impulses. In the thinned-out residual (TOR) method [24] the excitation is derived from the prediction error sequence by means of a thinning out procedure. These hybrid coders have a high complexity but due to the development of fast DSP chips and cheaper memory chips they are feasible now. Moreover, extensive research on CELP coders has helped it emerge as a standard in many applications.

Figure 1.1 gives a feel for different levels of coder complexity and the related quality of speech that can be attained [25]. The quality measure is based on MOS and unlike the objective quality measure it tends to saturate after a certain bit-rate is reached. The MOS scores are given as follows. A score of 5 indicates perfect quality. A score between 4 and 5 is referred to as toll quality and is required of Switched telephone networks or Public Switched Networks (PSN). A score between 3 and 4 is referred to as communication quality. On the basis of complexity the coders can be classified as simple, medium, and high complexity coders. PCM, which is

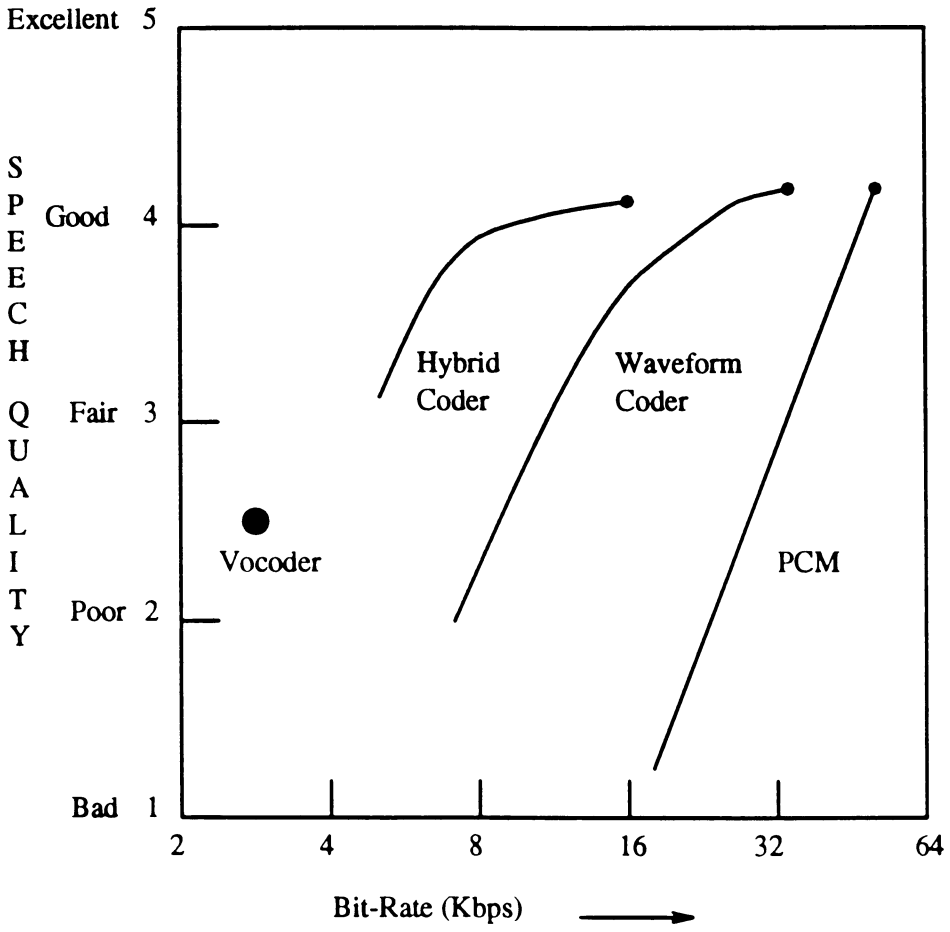


Figure 1.1: Quality of speech as a function of transmission rate

a simple coder achieves toll-quality at bit-rates of 56-64kbps and drops rapidly as the bit-rate is reduced. Other waveform coders, like DPCM and ADPCM, which are medium-complexity coders achieve toll-quality at bit-rates higher than 32kbps. Vocoders are high-complexity coders which work at very low bit-rates of 2.4-4.8Kbps but the speech quality attained is very poor and is referred to as synthetic-quality as the speaker cannot be identified. The high-complexity hybrid-coders combine the high quality potential of waveform coders with the compression efficiency of a vocoder. It can achieve toll-quality at as less as 16Kbps and between 8-16Kbps it

attains communication quality.

Another issue related to speech coders is the encoding delay. Encoding delay can be defined as the time interval between successive transmissions of speech data packets. This delay decides the kind of communication networks they can be used in. In public switched networks (PSNs) the allowable delay is less than 5ms. This is because in the presence of echoes, the perceptual degradation of the transmitted signal increases in severity as the round-trip transmission delay increases. This transmission delay depends on both the propagation delay and the encoding delay. Traditionally this problem was taken care of by echo cancellation circuits. The complexity of the echo cancellation circuitry increases with the increase in the round-trip delay. So by keeping the encoding delay to the minimum we can avoid the echo cancellation problem to some extent. The hybrid coders which have been mentioned here have a large coding delay, typically 20-40ms, which is due to a forward adaptation technique used for model parameter estimation and hence they cannot be used with PSNs in spite of their high quality speech and low bit-rates. This led to the development of low bit-rate low-delay coders. Here the model parameter adaptations are done in a backward fashion, by using the synthesized speech instead of actual speech and hence avoid the large coding delay. The coding delay is also dependent on the complexity of the coder and so the coder has to be kept as simple as possible. The low-delay vector excitation coding (LD-VXC) [26], low-delay backward adaptive CELP [27], low-delay codec based on Trellis coded quantization [28] and low-delay coders based on tree codes [29] are some examples of hybrid coders which have a coding delay less than 5ms. The proposed coding scheme will belong to this class of coders and is discussed in the following sections.

Motivation

Recently, a new approach to speech coding was introduced. This coding scheme was based on the Kalman estimation algorithm. In this technique, speech was considered to be a piece-wise stationary random process and modeled as the output of a two stage time varying all-pole filter excited by a white noise sequence. The parameters of the model are obtained by linear predictive analysis of the original speech. In state space representation, the states of the filter correspond to the actual speech samples at different instants. In this approach, the Kalman estimator algorithm is used to form measurements as a carefully chosen linear combination of the actual speech samples which are then used to optimally estimate the states of the filter. Estimates of these measurements are also obtained within the Kalman estimator algorithm and the difference between the actual measurements and their estimates, called the innovations, are quantized and transmitted at regular but sparse intervals. The parameters of the model and an estimate of the excitation variance are also quantized and transmitted. At the decoder the quantized innovations, the parameters and the excitation variance estimate are used to reconstruct the speech signal. This coder is referred to as the Innovations-Assisted Linear Predictive Coder (IALPC). This coder can be thought of as belonging to the class of hybrid coders. The performance of IALPC is good but it is highly complex and like other hybrid coders the encoding delay is high (32ms). So its use in PSNs is not possible. This has motivated us to develop a coder which will be of lower complexity and have a low coding delay (less than 5ms).

Proposed Coding Scheme

In the proposed coding scheme, the major limitations of the IALPC, namely high complexity and high coding delay, are overcome but of course with some reduction in the synthesized speech quality. The coding delay is reduced by employing backward adaptive techniques for the model parameter estimation and excitation variance estimation. In this technique, the parameters are computed from synthesized speech rather than actual speech and so we need not buffer the actual speech. Other than reducing the coding delay, it also helps us avoid transmission of the model parameters since they can be derived from the synthesized speech at the decoder itself. The complexity of the coder is also reduced by using a reduced Kalman state vector size and also by using alternate measurement techniques.

This thesis is organized as follows. Chapter 2 gives an introduction to the human speech production process, different speech sounds, and also an all-pole model representation of the speech production system. Linear prediction and methods of solving the linear prediction equation are also discussed. Chapter 3 deals with different adaptation techniques, namely forward adaptation and backward adaptation. Different windowing techniques and their usefulness in obtaining the autocorrelation values are discussed in Chapter 4. Special emphasis is given to recursive windowing technique since it is used in our proposed model to obtain the autocorrelation values. Kalman estimation algorithm is presented in Chapter 5 and is followed by a presentation of its application to speech coding in Chapter 6. The forward adaptive model and its limitations are also discussed. Chapter 7 gives a brief description of the proposed model and also deals with some key design issues like order of the prediction model, size of the Kalman state vector, quantizer design, and choice of a

measurement vector. Implementation details, results and conclusions are presented in Chapter 8.

CHAPTER 2. THE LINEAR PREDICTION MODEL

In this chapter, linear prediction theory which forms the basis of our speech prediction model is discussed. We will talk about different speech sounds that are produced by the human speech production system and how linear prediction can be used to model this system. Methods of solving the linear prediction equations are also dealt with. The presentation will be brief since linear prediction is a well understood and widely used concept.

Linear prediction can be defined as a method in which each sample of a signal is predicted as a linear combination of its past samples. The use of linear prediction for the estimation of future values of a signal from the past is not a new concept. It dates back to almost the mid twentieth century [1]. Linear prediction finds application in several areas like adaptive filtering, parameter estimation and system identification. The success of linear prediction for speech coding is due to its ability to provide us with a parametric description of a random process with Gaussian statistics such as speech. Linear Predictive Coding (LPC) is the only technique used now to achieve low bit rates of 4800 bps for synthesizing speech. Apart from its predictive nature, it also provides us with a good model of the vocal tract. This helps us to understand the working of the human speech production system for theoretical and practical purposes. Further, the parametric representation can also be used for recognition

purposes [2].

Linear prediction can also be viewed as a method to build an all-pole model for a signal which makes the prediction error almost white with a small variance. To see this consider a discrete-time signal $s(k)$ which is represented as the output of a system with difference equation

$$s(k) = \sum_{j=1}^{q} b_j \epsilon(k-j) - \sum_{m=1}^{p} a_m s(k-m) + \epsilon(k)$$

where b_j, $1 \leq j \leq q$ and a_m, $1 \leq m \leq p$ are the coefficients of the system. Here the signal $s(k)$ is expressed as a linear combination of its past outputs and present and past inputs $\epsilon(k)$. The transfer function of this linear prediction equation can be represented as

$$H(z) = \frac{1 + \sum_{j=1}^{q} b_j z^{-q}}{1 - \sum_{m=1}^{p} a_m z^{-m}}. \tag{2.1}$$

This is the most general system and is referred to as the ARMA (auto regressive moving average model) or the pole-zero model [4]. There are two variations of this model. If the present output depends only on the present and past inputs then the model does not have any poles and its transfer function will be

$$1 + \sum_{j=1}^{q} b_j z^{-q}$$

which is Eqn. 2.1 with the denominator equal to 1, and it is called an MA (Moving Average) model or all-zero model. If the present output depends only on the past outputs and the present input then it will not have any zeros and its transfer function will be

$$\frac{1}{1 + \sum_{m=1}^{p} a_m z^{-m}}$$

which is Eqn. 2.1 with the numerator equal to 1. This is called the AR (Auto-Regressive) model or the all-pole model. In modeling speech the all-pole model is favored because of several reasons which will become evident as the chapter progresses. Linear prediction is a method for obtaining an AR model of a signal.

The human speech production system mainly consists of the glottis where excitation is produced, and the vocal tract which modulates the excitation to produce different speech sounds. Speech signals can be voiced or unvoiced. Voiced sounds are produced by exciting the vocal tract with short periodic pulses [3]. The vocal tract is excited with noise like turbulent air to generate unvoiced sounds. Linear prediction can be used to build a model for the speech production system. Such a model is shown in Figure 2.1. The filter is obtained using linear prediction technique on speech segments. The excitation, $\delta(k)$, is chosen to be periodic pulses or white noise depending on whether the speech is voiced or unvoiced. $s(k)$ is the output speech.

A good understanding of speech sounds is essential in choosing an LP model. Speech can also be classified as vowels, nasals, stops, and fricatives according to acoustic phonetics. Voiced speech signals are nearly periodic for short durations (10-40ms). These voiced speech signals can be adequately represented by an all-pole filter model. The nasals, stops, and fricatives (unvoiced) need a pole-zero model to represent the vocal tract. The zeros in the transfer function of the model for nasals and unvoiced sounds lie within the unit circle in the z-plane [3].

The transfer function of the system can be chosen in many ways based on these differing speech sounds. For non-nasal voiced speech sounds, the transfer function of the linear filter can be represented by an all-pole model. Whereas for unvoiced

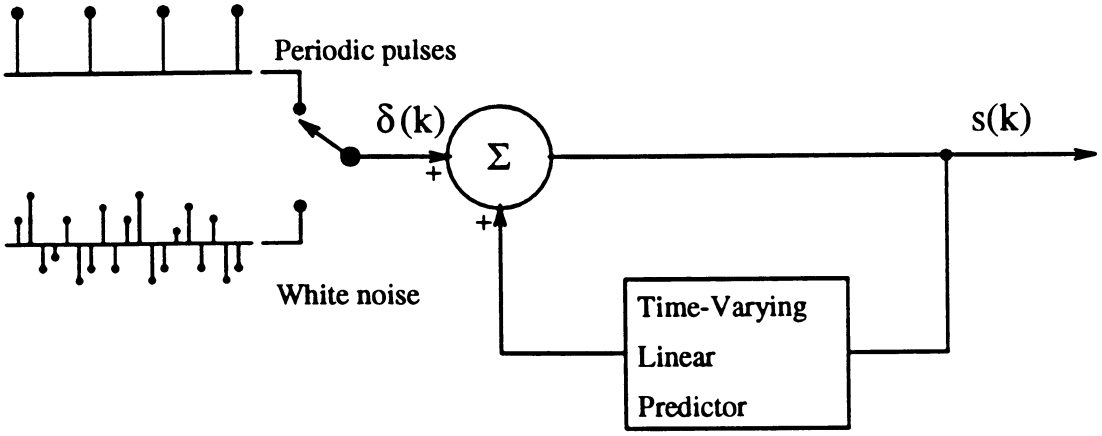


Figure 2.1: Linear Prediction Model of Vocal Tract

or nasal speech sounds, the filter transfer function has both poles and zeros. Since the zeros of this filter are all within the unit circle in the z plane it can be approximated with an all-pole model with the desired accuracy [3]. Moreover, if we choose an all-pole model the coefficients a_m can be computed from a set of linear equations. The speech production model can thus be represented by an all-pole filter with its transfer function as

$$F(z) = \frac{1}{1 + \sum_{m=1}^{p} a_m z^{-m}}. \tag{2.2}$$

This all-pole filter has p poles which are either real or occur in complex conjugate pairs. For the filter to be stable, these poles should lie inside the unit circle. The output of the filter at the k^{th} instant can then be written as,

$$s(k) = -\sum_{m=1}^{p} a_m s(k-m) + e(k). \tag{2.3}$$

Here the speech sample at the time instant k is formed as a linear combination of the past p samples and $e(k)$ is the k^{th} sample of the excitation. The parameters

a_m's are called the predictor coefficients and they are responsible for the filtering action. The vocal tract in a human speech production system varies according to the speech sounds spoken and so the coefficients of the all-pole model must be updated periodically to match the varying transfer function of the vocal tract. The following sections explain the method of obtaining the coefficients of the all-pole model.

The Linear Prediction Equations

Let a zero-mean signal $s(k)$ be estimated as a linear combination of its past p values. The prediction can be either 'one-step prediction' i.e., speech sample $s(k)$ is predicted from its immediate past samples or 'n-step prediction' where we predict $s(k-n-1)$ from the samples $s(k-1), s(k-2), \ldots, s(k-p)$. In this section we will deal with only one-step prediction. The prediction equation can be written as

$$\hat{s}(k) = -\sum_{m=1}^{p} a_m s(k-m) \tag{2.4}$$

where, $\hat{s}(k)$ is the predicted speech at time t_k. The z-transform notation of this prediction filter is given by

$$A(z) = -\sum_{m=1}^{p} a_m z^{-m}. \tag{2.5}$$

This predicted speech involves some error as the prediction is not perfect and so the actual speech at time t_k can be written as

$$s(k) = -\sum_{m=1}^{p} a_m s(k-m) - e(k). \tag{2.6}$$

The error in prediction is given by

$$e(k) = s(k) + \sum_{m=1}^{p} a_m s(k-m) = \sum_{m=0}^{p} a_m s(k-m) \tag{2.7}$$

where $a_0 = 1$. To obtain the predictor coefficients (a_m's) we minimize the mean squared error $E = \langle \epsilon^2(k) \rangle$, where $\langle . \rangle$ represents the expectation operator. This is done by using the *orthogonality* principle. Orthogonality principle states that the desired coefficients are obtained by making error orthogonal to the samples $s(k-1), s(k-2), \cdots, s(k-p)$ (Two random variables f and g are orthogonal if $\langle f\, g \rangle = 0$). Hence we have,

$$\langle s(k-m)\epsilon(k) \rangle = 0 \qquad \text{for } m = 1, 2, \ldots, p \tag{2.8}$$

$$\langle s(k-m) \sum_{i=0}^{p} a_m s(k-i) \rangle = 0. \tag{2.9}$$

Interchanging the order of the summing and the averaging and replacing ensemble average by time average we get,

$$\sum_{m=0}^{p} a_m \sum_{k} s(k-m)s(k-i) = 0. \qquad m = 1, 2, \ldots, p \tag{2.10}$$

The predictor coefficients can be computed from this equation.

Using the orthogonality principle the resulting minimum error can be expressed as

$$E = \langle \epsilon^2(k) \rangle = \langle s(k)\epsilon(k) \rangle \tag{2.11}$$

$$= \langle s(k) \sum_{m=0}^{p} a_m s(k-m) \rangle \tag{2.12}$$

where $a_0 = 1$. Interchanging the order of the summing and the averaging and replacing ensemble average by time average we get,

$$E = \sum_{m=0}^{p} a_m \sum_{k} s(k-m)s(k). \tag{2.13}$$

The limits on k in the summation depends on the number of points over which we minimize the squared error. There are two ways of doing this:

1. We can minimize the error over all time. In this case, limits of k are $-\infty$ and ∞. Then

$$\sum_{m=0}^{p} a_m R_{m-j} = 0, \qquad j = 1, 2, \cdots, p \text{ and} \tag{2.14}$$

$$E = \sum_{m=0}^{p} a_m R_m \tag{2.15}$$

where, the *autocorrelation* values are given by

$$R_m = \sum_{k=-\infty}^{\infty} s(k)s(k-m). \tag{2.16}$$

These autocorrelation values are usually computed from windowed speech, eg. Hamming window or a recursive window. The method of windowing speech and obtaining the autocorrelation coefficients will be discussed in detail in the next chapter. Since we have only a finite number of points, K, to operate on, the speech is suitably windowed, i.e., $s(k) = 0$ for k outside the range $(0, K-1)$. Then Eqn. 2.16 becomes

$$R_m = \sum_{k=m}^{K-1} s(k)s(k-m). \tag{2.17}$$

This is called the autocorrelation method.

2. We can also minimize the error over K points of the signal. In this case the limits of k will be 0 and $K-1$ and the equation becomes,

$$\sum_{m=0}^{p} a_m c_{mj} = 0 \qquad j = 1, 2, \cdots, p \tag{2.18}$$

and

$$E = \sum_{m=0}^{p} a_m c_{i0} \tag{2.19}$$

where

$$c_{mj} = \sum_{k=0}^{K-1} s(k-m)s(k-j). \tag{2.20}$$

To compute c_{00} to c_{pp} we need $K + p$ sample points of the signal. This method is called the *covariance* method [1].

In speech processing, the autocorrelation method gives better results for fricatives and the covariance method for periodic sounds. The autocorrelation method is computationally cheaper and it also leads to a number of interesting theoretical insights.

Solution to Linear Prediction Equations (Autocorrelation method)

Eqn. 2.14 can be written as follows:

$$\begin{aligned}
R_{-1}a_0 - R_0a_1 - R_1a_2 - \cdots - R_{p-1}a_p &= 0 \\
R_{-2}a_0 - R_{-1}a_1 - R_0a_2 - \cdots - R_{p-2}a_p &= 0 \\
R_{-3}a_0 - R_{-2}a_1 - R_{-1}a_2 - \cdots - R_{p-3}a_p &= 0 \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \quad &\vdots \\
R_{-p}a_0 - R_{1-p}a_1 - R_{2-p}a_2 - \ldots - R_0a_p &= 0
\end{aligned} \tag{2.21}$$

From Eqn. 2.17 it is clear that $R_{-i} = R_i$, and using $a_0 = 1$, we can rewrite Eqn. 2.21 we get

$$\begin{bmatrix} R_0 & R_1 & R_2 & \cdots & R_{p-1} \\ R_1 & R_0 & R_1 & \cdots & R_{p-2} \\ R_2 & R_1 & R_0 & \cdots & R_{p-3} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{p-1} & R_{p-2} & R_{p-3} & \cdots & R_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_p \end{bmatrix}. \tag{2.22}$$

The matrix in this equation is symmetric and Toeplitz. This equation can be solved using a recursive technique instead of the standard technique of matrix inversion. This recursion is called the Levinson-Durbin algorithm and is computationally very efficient [1].

Levinson-Durbin Algorithm

The recursion proceeds in steps and the predictor of order i is determined from the predictor of order $i-1$. The recursion is done until the required order of the predictor is obtained.

The recursion starts with an initial predictor order of $i = 0$.

1. For $i = 0$, $E_0 = R_0$
2. For step i, $i = 1, \ldots, p$

   a. $k_i = \frac{-1}{E_{i-1}} \sum_{j=0}^{i-1} a_j^{(i-1)} R_{i-j}$

   b. $a_i^{(i)} = k_i$

c. For $m = 1 \, to \, i-1$

$$a_m^{(i)} = a_m^{(i-1)} + k_i a_{i-m}^{(i-1)}$$

d. $E_i = E_{i-1}(1 - k_i^2)$

3. If $i \neq p$, increment i by 1 and goto step 2. If $i = p$ then end recursion.

At the start of the recursion E_0 is equal to the zeroth autocorrelation coefficient. Then for each recursion which goes on till the required order is reached, the k's which are called the reflection coefficients or PARCOR (PARtial CORrelation) coefficients are calculated for every order. The predictor coefficients for order i are calculated recursively from the reflection coefficient of order i and the predictor coefficients of order $i-1$. Superscripts of the a's represent the iteration number or order of the predictor. The quantity E_i, called the minimum prediction error for order i, either reduces or remains constant as the order of the predictor is increased. The recursion is stopped when the desired order p is reached. The predictor order is dependent on the rate at which the speech signals have been sampled. A predictor order between 10 and 16 is sufficient for speech to obtain a good prediction. If the linear prediction model matches the spectrum of the speech segment well then the prediction error will be white and have a small variance. The degree of match can be expressed in terms of prediction gain which is defined as

$$10 \log \frac{\sum_{k=0}^{K-1} s^2(k)}{\sum_{k=0}^{K-1} [s(k) - \sum_{m=1}^{p} s(k-m)]^2}$$

and its unit is *decibel* (dB). The higher the prediction gain the better the prediction

model. The predictor coefficients (a_m's) are updated for every short segment of speech so as to obtain a good prediction. In the case of a forward adaptation speech coder the a_m's are computed using the original speech signal and in the case of backward adaptation scheme they are computed from previously quantized speech. The forward adaptation and backward adaptation techniques are discussed in greater detail in the next chapter.

Pitch Prediction

We have already seen that the speech production system can be modeled as an all-pole filter excited by a white noise sequence. If we take the difference between the actual speech and its predicted value we should get an error sequence which is be white. The filter which would give this error sequence is called the prediction error filter and from Eqn. 2.7 the error sequence can be written as

$$\epsilon(k) = s(k) - \sum_{m=1}^{p} a_m s(k-m) \tag{2.23}$$

and the prediction error filter transfer function can be written as

$$H(z) = 1 - \sum_{m=1}^{p} a_m z^{-m} = \frac{1}{F(z)} \tag{2.24}$$

Because of this structure, where the prediction error filter is the inverse of the all-pole filter, it is also called an inverse filter and the process of passing actual speech through this filter to obtain the error sequence is called inverse filtering. The output of this filter is called 'inverse filtered speech' which is nothing but the error sequence.

The inverse filtered speech is almost white and has very small sample to sample correlation for short intervals but it is still correlated with samples in the adjacent pitch periods. This correlation can be removed with a knowledge of the pitch period

and a long-term predictor. This long-term predictor (or pitch predictor) assumes the form

$$P(z) = \beta_1 z^{-M+1} + \beta_2 z^{-M} + \beta_3 z^{-M-1}$$

where the delay M is the estimate of the pitch and β_q's are the coefficients of the pitch predictor. Numerous techniques have been suggested for pitch estimation [17], auto-correlation method and amplitude magnitude difference function being the popular methods used with speech coders. The coefficients β_q's are determined by minimizing the mean-squared error [18]. By minimizing the mean squared error we end up with a set of linear equations as given below:

$$\begin{bmatrix} r(M,M) & r(M,M+1) & r(M,M-2) \\ r(M+1,M) & r(M+1,M+1) & r(M+1,M-2) \\ r(M-2,M) & r(M-2,M-1) & r(M-2,M-2) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} r(0,M) \\ r(0,M+1) \\ r(0,M-2) \end{bmatrix} \quad (2.$$

where

$$r(i,j) = \sum_{k=0}^{N-1} \epsilon(k-i)\epsilon(k-j). \qquad 0 \le i,j < 3 \qquad (2.26)$$

This equation is solved to obtain the pitch predictor parameters.

Pitch prediction is very helpful in achieving additional prediction gain especially in forward adaptive speech coders where exact pitch estimation is possible. Prediction gain due to pitch prediction is usually between 5-7dB in forward adaptive coders [18] and 1-1.5dB in the case of backward adaptive coders [19].

CHAPTER 3. PREDICTOR ADAPTATION TECHNIQUES

In this chapter, we discuss methods of updating or adapting the all-pole model which is necessary to match the changes in the shape of the vocal tract for different speech sounds. Adaptive linear prediction is an extension of the linear predictive technique discussed in the previous chapter. The speech data is divided into short segments, typically of 10 to 20ms duration, and the predictor coefficients are computed on a segment by segment basis. The predictor coefficients computed may or may not be transmitted to the receiver depending upon the adaptation scheme used. At the receiver a synthesis filter reconstructs the speech using these predictor coefficients. The adaptation techniques frequently used with speech coding are forward adaptation and backward adaptation.

There are some important differences between forward and backward adaptation. First, forward adaptation can use clean speech to obtain the predictor coefficients of the all-pole model whereas backward adaptation has to use synthesized speech which alone is available at the decoder. Second, in forward adaptation the predictor coefficients computed for a particular frame of speech can be applied to the same frame whereas in backward adaptation the predictor coefficients computed from the previous frame have to be applied to the present frame of speech. A detailed description of the two methods is given below.

Forward Adaptation

In forward adaptation, the predictor coefficients are computed from actual speech samples. This requires buffering of the speech samples. In Figure 3.1, K input samples are buffered and the samples are released after p predictor coefficients are optimally computed from this segment. This segment is usually windowed with a Hamming window [4]. The predictor order is chosen such that an adequate prediction gain is obtained. The choice of the window length K depends on various factors:

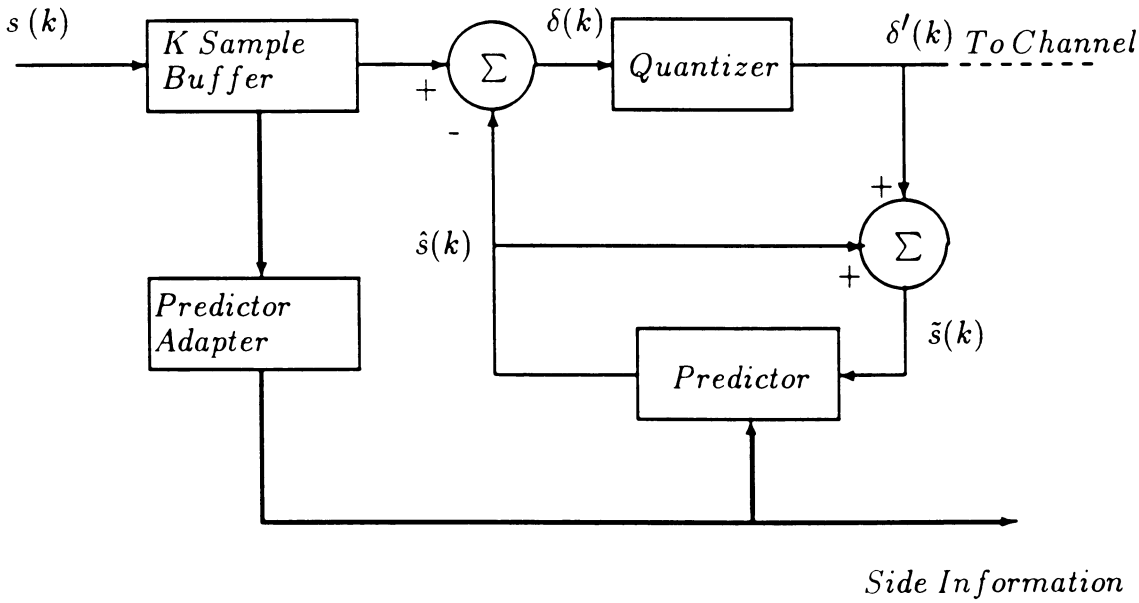


Figure 3.1: Generic Forward Adaptive Speech Coder

(i) The rate at which the predictor coefficients can be updated and transmitted to achieve required bit rate.

(ii) The time interval over which the input speech signal is stationary and also adequate for the optimization algorithm to be reliable.

Speech can be considered to be stationary over a time interval of 10 to 40ms and usually a K corresponding to a buffer size of 10 to 20 ms is adequate. A predictor order of 10 is required to ensure good prediction for a sampling rate of 8kHz [8].

In a typical forward adaptive coder, as in Figure 3.1, the input speech is buffered and the predictor coefficients are computed optimally. This information is updated every 10 to 20 ms in the predictor. A linear combination of the past quantized samples (or synthesized speech $\tilde{s}(k)$) serves as the predicted value $\hat{s}(k)$ which is subtracted from the current sample $s(k)$ to generate the residual $\delta(k)$. The residuals are quantized and transmitted. Side information about the predictor coefficients are also transmitted.

The inherent problem in this type of forward adaptive coder is the large coding delay which depends on the size of the buffer. This is objectionable in certain communication networks like the public switched networks. This coding delay can be avoided by the use of backward adaptation technique.

Backward Adaptation

The disadvantages of the forward adaptation technique, namely the coding delay, data buffering and extra channel capacity needed to transmit the side information are avoided here. This is possible because optimal estimation of the predictor coefficients is possible with quantized samples also [5].

Figure 3.2 shows a generic backward adaptive coder. The predictor coefficients are computed from the synthesized speech $\tilde{s}(k)$. The samples are usually windowed by a recursive window. Hence the samples need not be buffered thereby reducing coding delay considerably. Also as synthesized speech samples are used to compute the

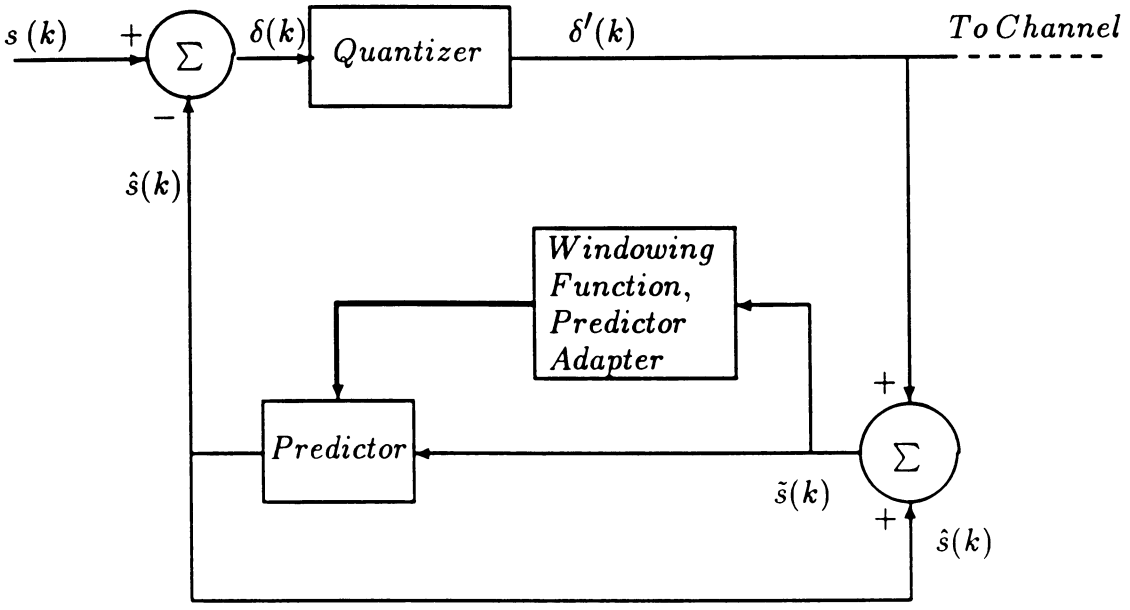


Figure 3.2: Generic Backward Adaptive Speech Coder

predictor coefficients, side information about the predictor need not be transmitted. This is possible because synthesized speech $\tilde{s}(k)$ is available at both encoder and decoder. Moreover, the predictor coefficients can be updated as frequently as required without increasing the bit rate. The residual $\delta(k)$ is obtained as a difference between actual speech sample at time instant k and the sample predicted using synthesized speech samples. As no buffering of the samples is done the residuals can be quantized and transmitted instantaneously.

The performance of a backward adaptive model is very close to a forward adaptive model [5]. It can provide prediction gains only 1 dB short of a forward predictive model for one-step prediction [Chapter 2 Section 2]. However, at low bit rates, when we use two-step prediction the prediction gain is poor especially at the beginning

of every pitch period and so robust adaptive quantizers are required to encode the residuals to compensate for the reduced prediction gains.

CHAPTER 4. WINDOWING TECHNIQUES

In computing the predictor coefficients, the speech signals are blocked into short segments (10-20ms) during which time speech can be considered stationary. This process of dividing the speech samples into segments is called windowing. In this chapter, we shall look at how the windowing of speech samples can be accomplished and how the autocorrelation values are computed from the windowed speech.

Windows that are used in relation to speech prediction are of two types, recursive and nonrecursive [9]. Nonrecursive windows are generally used when forward adaptation techniques are employed for model parameter estimation and recursive windows are used with backward adaptation techniques.

The effect of windowing can be explained as follows. Let $h_d(k)$ be the desired response of a system and $h(k)$ the actual response of the system given by

$$h(k) = h_d(k)w(k)$$

where $w(k)$ is a finite duration window. From the windowing theorem [6] we can write

$$H(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\theta})W(e^{j(\omega-\theta)})d\theta$$

where $H(e^{j\omega})$, the frequency response of the actual system, is the periodic convolution of the desired ideal frequency response with the Fourier transform of the window [6]. So windowing the data results in an alteration of the system response.

As linear prediction of speech is a process of spectrum matching the choice of the windowing technique used is quite important in obtaining a good spectral representation. The non-recursive and recursive windows are discussed in the following sections.

Nonrecursive Windows

Some of the common nonrecursive windows are the rectangular window and the Hamming window. The two windows can be mathematically represented as

Rectangular

$$w(k) = \begin{cases} 1, & 0 \leq k \leq K-1 \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

Hamming

$$w(k) = \begin{cases} 0.54 - 0.46\cos 2\pi k/K, & 0 \leq k \leq K-1 \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

where K is the length of the window. A simple example of a desired (or ideal) response of a system (Low-Pass Filter) is shown in Figure 4.2. The frequency response of a rectangular window and Hamming window (21 points) are shown in Figure 4.1. The frequency response of the window [10], or simply the window response, is dominated by two features. There is a large hump around $\omega = 0$ called the main lobe and the rest of the window response called the side lobes. The frequency response of the actual system using the rectangular window and the Hamming window are shown in Figure 4.2. It can be seen that the ripple in the frequency response of the actual system has a greater amplitude for the rectangular window than the Hamming window. But since the main lobe of the rectangular window is narrower than that

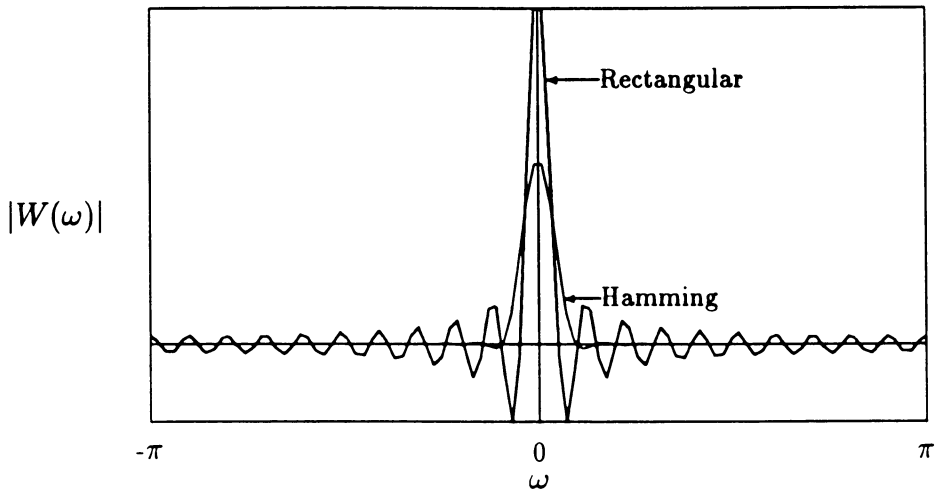


Figure 4.1: Frequency Response of Windowing Function

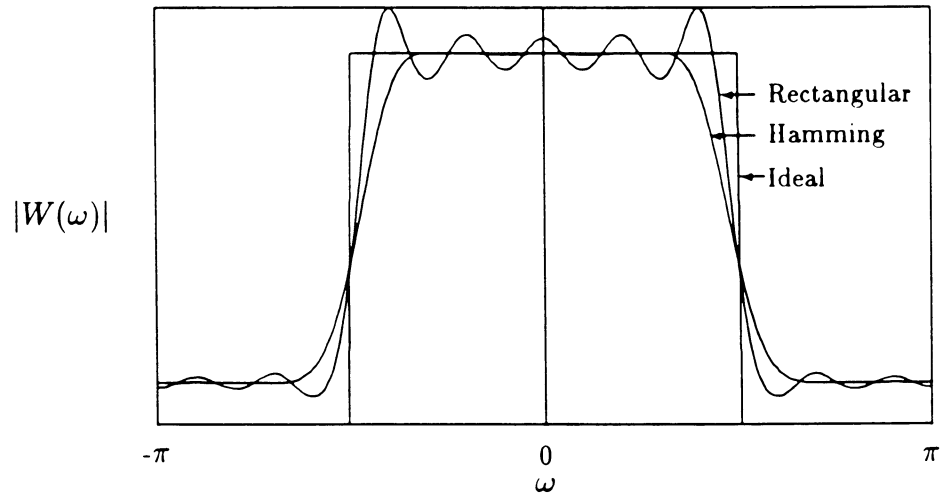


Figure 4.2: Frequency Response of a Low-Pass Filter

of the Hamming window, the width of the transition regions of the system frequency response are narrower too. If the window size is increased the width of the main lobe decreases for both the windows but the amplitude of the side lobes remain the same. So with the Hamming window technique we can achieve a better spectral matching by increasing the window length whereas a larger rectangular window will not have an improvement. In speech prediction, the predictor coefficients are more sensitive to the ripple than the width of the transition regions and so Hamming window is a better choice.

The autocorrelation estimation from windowed speech is done as follows. The input speech signal is first divided into segments. This is done by applying the window function to the segment which can be mathematically represented as

$$s_w(k,n) = s(k - nK)w(k) \qquad \text{for } k = 0, 1, \ldots, K-1 \qquad (4.3)$$

where $s_w(k,n)$ is the windowed speech, n specifies the n^{th} window or segment, and k is the index of the speech sample within the current segment. The autocorrelation values are given by Eqn. 2.17, with $s(k)$ replaced by $s_w(k,n)$.

Nonrecursive windows are suited for forward adaptive speech coders as the predictor coefficients are computed using buffered speech. The problem with nonrecursive windows is that (of course, it is an inherent problem of the forward adaptation techniques) it needs a lot of memory to buffer the speech samples and thereby increases the coding delay [5]. This problem is eliminated in the recursive windowing technique which is used with backward adaptation.

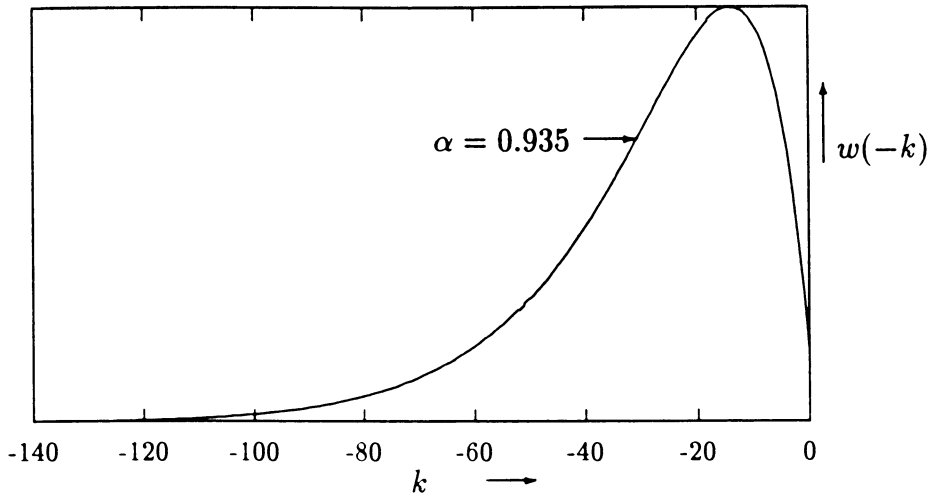


Figure 4.3: Impulse Response of a 2^{nd} Order IIR filter with Two Real Poles at $z = \alpha$

Recursive Window

Recursive window or infinite window is mainly used with backward adaptation techniques [9]. They use all the past data samples in the autocorrelation estimation. These windows cannot usually adapt to the rapid changes in the model. This necessitates the need to update the model parameters more often. The recursive window is actually the impulse response of an infinite impulse response filter (IIR) filter which has its poles within the unit circle. So these windows are unconditionally stable. Even though the window is of infinite duration, its magnitude is very small outside the desired region, which is controlled by the poles of the filter, and so meets the requirement of a finite window. The time reversed impulse response of an IIR filter is shown in Figure 4.3. The method of computing the autocorrelation values is described below.

Autocorrelation Estimation Using a Recursive Window

Let $\tilde{s}(n)$ be the synthesized speech at time instant t_n and let $w(k)$ be the windowing function [11]. Windowed speech is given by

$$\tilde{s}_w(n+k) = \tilde{s}(n+k)w(-k) \tag{4.4}$$

where the windowing function is such that $w(k) = 0$ for $k < 0$ and $w(k)$ is indexed backwards in time as we are using the past synthesized samples in the computation of autocorrelation values. The m^{th} autocorrelation value is then given by

$$R_m = \sum_{k=-\infty}^{\infty} \tilde{s}_w(n-k)\tilde{s}_w(n-k-m). \tag{4.5}$$

Substituting Eqn 4.4 in Eqn 4.5, we get

$$R_m = \sum_{k=-\infty}^{\infty} \tilde{s}(n-k)\tilde{s}(n-k-m)w(-k)w(-k-m). \tag{4.6}$$

Define

$$W(k,m) = w(k)w(k-m) \text{ and} \tag{4.7}$$

$$S(n-k,m) = \tilde{s}(n-k)\tilde{s}(n-k-m). \tag{4.8}$$

Eqn. 4.6 can now be written as

$$R_m = \sum_{k=-\infty}^{\infty} S(n-k,m)\,W(-k,m). \tag{4.9}$$

In this equation the m^{th} autocorrelation value is expressed as the convolution of $S(n-k,m)$ and $W(k,m)$. Since $W(k,m)$ is the product of the two windows $w(k)$ and $w(k-m)$, the z-transform of $W(k,m)$, namely $W_m(z)$, can be obtained as the convolution of the z-transforms of these two windows. The window is chosen as the

impulse response of an IIR filter with a transfer function of $H(z)$. Now $W_m(z)$ can be written as

$$\frac{1}{2\pi j}\oint H(\nu)H(z/\nu)\nu^{-m-1}\,d\nu. \tag{4.10}$$

The IIR filter is chosen as a 2^{nd} order filter with two poles at $z = \alpha$. The transfer function of the 2^{nd} order filter can be written as

$$H(z) = \frac{1}{(1-\alpha z^{-1})^2}. \tag{4.11}$$

Substituting for $H(z)$ in Eqn. 4.10 we get

$$W_m(z) = \frac{1}{2\pi j}\oint \frac{\nu^{m-1}}{(1-\alpha\nu^{-1})^2(1-\alpha\nu/z)^2}\,d\nu. \tag{4.12}$$

Solving this equation using Cauchy residue theorem [6] and we obtain

$$W_m(z) = \frac{b(0,m) - b(1,m)z^{-1}}{1 - a(1,m)z^{-1} - a(2,m)z^{-2} - a(3,m)z^{-3}} \tag{4.13}$$

where

$$b(0,m) = (m-1)\alpha^m \tag{4.14}$$

$$b(1,m) = -(m-1)\alpha^{m-2} \tag{4.15}$$

$$a(1,m) = 3\alpha^2 \tag{4.16}$$

$$a(2,m) = -3\alpha^4 \text{ and} \tag{4.17}$$

$$a(3,m) = \alpha^6. \tag{4.18}$$

This recursive algorithm can be directly implemented or the filter can be factorized into a cascade of 3 first order all-pole filters and an all-zero filter and then implemented. The z-transform of the recursive algorithm can now be represented as

$$W_m(z) = \frac{1}{(1-\alpha^2 z^{-1})}\frac{1}{(1-\alpha^2 z^{-1})}\frac{1}{(1-\alpha^2 z^{-1})}[(m+1)\alpha^m-(m-1)\alpha^{m+2}z^{-1}].(4.$$

Eqn. 4.19 can be implemented as shown in Figure 4.4 to compute the autocorrelation values. The input to the algorithm is the past quantized speech which is obtained as the output of the Kalman estimator explained in the next chapter. The product of the current synthesized sample and its past values are computed and then passed to the linear filter. The output of the cascade of the three all-pole filters is computed for every sample. The multiplies in the all-zero portion are done only when the autocorrelation values are needed.

Advantages

The use of the recursive windowing technique has the following advantages:

(i) The structure of the filter is very simple and so can be easily implemented in hardware.

(ii) The amount of storage memory required is considerably reduced.

(iii) The computations are distributed over all the samples.

(iv) The performance is as good as the Hamming window technique.

(v) This is well suited for backward adaptive coders which dont buffer the speech samples so as to reduce the coding delay.

The autocorrelation values obtained are then used in Eqn. 2.22 [Chapter 2] to obtain the predictor coefficients as explained earlier. In the next chapter, we will

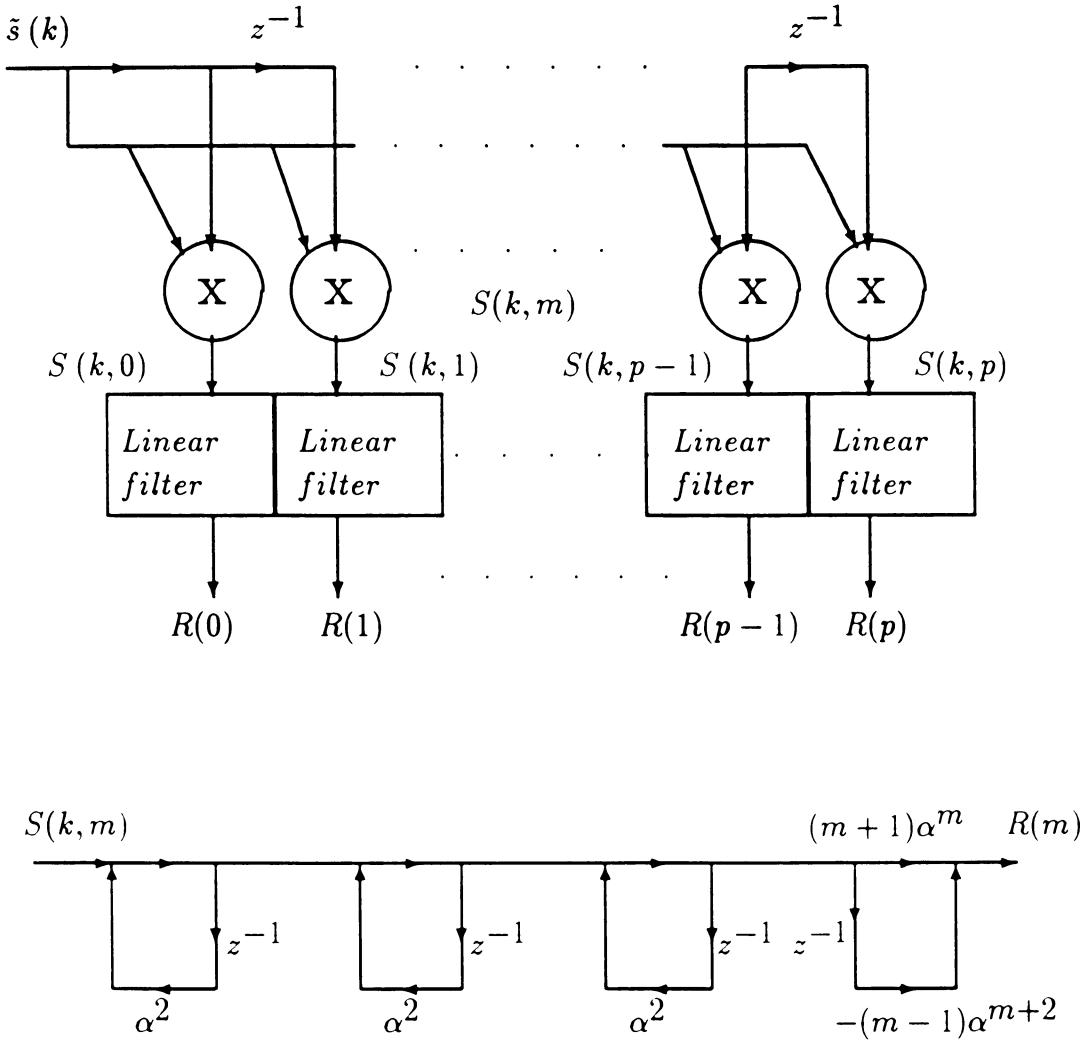


Figure 4.4: Recursive Windowing Algorithm

discuss some concepts of the Kalman estimator and how they can be used in speech coding applications.

CHAPTER 5. THE KALMAN ESTIMATOR

In the proposed speech coding scheme, the Kalman estimator acts as an analysis filter at the encoder and produces the necessary information to be encoded and transmitted to the decoder. At the decoder, the Kalman estimator acts as a synthesis filter and generates synthesized speech using the information received from the encoder. So knowledge of the Kalman estimator concepts is essential in understanding the speech coding scheme. In this chapter, we discuss the method of representing a random process using state space concepts and also present the Kalman filter recursive equations.

The Kalman filter, proposed by R. E. Kalman, provides a recursive solution to the least squares problem[15]. It has found numerous application areas because of its features, namely, (i) its mathematical formulation is described in terms of *state space concepts*, (ii) it uses vector modeling of random processes, and (iii) it uses recursive processing of noisy measurement data for prediction, smoothing and filtering. Kalman filter can be used for a wide range of applications which include adaptive equalization, statistical modeling, parameter estimation, image restoration, etc. Moreover, with the advent of special-purpose digital signal processors which are ideally suited to implement the Kalman filter its application in digital signal processing has increased.

Kalman filter algorithm exists for continuous processes as well as discrete time processes. In both the process being estimated is considered to be the result of passing white noise through a linear system model. The process forms the state vector and is recovered using measurements which are linear combinations of the state variables. The process need not be necessarily stationary as the model can vary with time. A detailed description of the discrete Kalman filter in given in the next section.

The Discrete Kalman Filter

Let the input stochastic process be modeled as

$$\mathbf{x}_{k-1} = \mathbf{\Phi}_k \mathbf{x}_k - \mathbf{w}_k \tag{5.1}$$

and the observation (measurement) be modeled as,

$$z_k = \mathbf{H}_k \mathbf{x}_k - \nu_k \tag{5.2}$$

where, $\mathbf{x}_k$ - (nx1) process state vector at time t_k,

$\mathbf{\Phi}_k$ - (nxn) state transition matrix at time t_k relating $\mathbf{x}_k$ to $\mathbf{x}_{k-1}$,

$\mathbf{w}_k$- (nx1) process noise vector at time t_k,

z_k- scalar measurement at time t_k,

$\mathbf{H}_k$- (1xn) measurement vector at time t_k relating $\mathbf{x}_k$ to z_k, and

ν_k- scalar measurement noise at time t_k.

The process noise and the measurement noise are assumed to be zero mean, white, and uncorrelated with each other and also with the initial state vector $\mathbf{x_0}$. Mathematically these assumptions can be expressed as,

$$E[\mathbf{w}_k \mathbf{w}_i^T] = \mathbf{Q}_k \delta_{ki}, \tag{5.3}$$

$$E[\nu_k \nu_i] = R_k \delta_{ki}, \tag{5.4}$$

$$E[\mathbf{w}_k \nu_i] = \mathbf{0}, \text{for all } k \text{ and } i \tag{5.5}$$

$$\begin{aligned} E[\mathbf{w}_k \mathbf{x}_0^T] &= \mathbf{0}, \text{ for all } k \text{ and} \\ E[\nu_k \mathbf{x}_0^T] &= \mathbf{0}, \text{ for all } k \end{aligned} \tag{5.6}$$

where δ_{ki} is the Kronecker delta function. Eqn. 5.1 is called the time update equation which provides the linear relationship between the state vectors at time t_{k-1} and t_k. Eqn. 5.2 is called the measurement equation where the measurement z_k is formed as an inner product of the state vector and the measurement vector $\mathbf{H}_k$ plus some noise ν_k called the measurement noise which arises due to improper measurements. These measurements are used to refine or filter the time updated states to get an optimal estimate of the actual process. This phase is also called the measurement update phase.

The Kalman Filter Equations

An initial estimate of the process state vector is assumed to be known at some point of time t_k. The estimate is assumed on the basis of all our knowledge about the process prior to time t_k. This initial estimate, called *a priori* estimate or one-step predicted estimate, is denoted by $\hat{\mathbf{x}}_k^-$, where the 'hat' specifies that it is an estimate and the 'super minus' indicates that it is based on all our knowledge prior to t_k. So the error in this *a priori* estimate can be written as

$$\mathbf{e}_k^- = \mathbf{x}_k - \hat{\mathbf{x}}^- \tag{5.7}$$

and the associated error covariance matrix, called the *a priori* error covariance matrix, can be defined as

$$\mathbf{P}_k^- = E[\mathbf{e}_k^- \mathbf{e}_k^{-T}]. \tag{5.8}$$

The *a priori* estimate $\hat{\mathbf{x}}_k^-$ along with the measurement z_k is used to refine the estimate and form the *a posteriori* estimate, also called the measurement updated estimate or filtered estimate, denoted by $\hat{\mathbf{x}}_k$. The error in the *a posteriori* estimate can be written as

$$\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k \tag{5.9}$$

and its associated error covariance matrix $\mathbf{P}_k$, called the *a posteriori* error covariance matrix, can be written as

$$\mathbf{P}_k = E[\mathbf{e}_k \mathbf{e}_k^T]. \tag{5.10}$$

The trace of this $\mathbf{P}_k$ matrix has to be reduced to obtain optimal estimates. These lead to a set of recursive equations and they are collectively called the Kalman recursive algorithm[15].

The Kalman Recursive Algorithm

The recursive equations are split into *analysis equations* and *synthesis equations* so as to explain later how they can used in speech coding.

Analysis Equations

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{\Phi}_k \hat{\mathbf{x}}_k \tag{5.11}$$

$$\mathbf{P}_{k+1}^- = \mathbf{\Phi}_k \mathbf{P}_k \mathbf{\Phi}_k^T + \mathbf{Q}_k \tag{5.12}$$

$$\xi_k = z_k - \mathbf{H}_k \hat{\mathbf{x}}_k^- \tag{5.13}$$

Synthesis Equations

$$\Lambda_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + R_k \tag{5.14}$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \Lambda_k^{-1} \tag{5.15}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \xi_k \tag{5.16}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \tag{5.17}$$

The scalar quantity ξ_k which is the difference between the measurement z_k and its estimated value $\mathbf{H}_k \hat{\mathbf{x}}_k^-$ is called the innovation and its variance is Λ_k. $\mathbf{K}_k$ is called the Kalman gain. This Kalman gain $\mathbf{K}_k$ has been chosen such that it minimizes the trace of $\mathbf{P}_k$ and hence minimize the mean square estimation error. The state vector , $\mathbf{x}_k$, can be recursively estimated if we have knowledge of the transition matrix, which describes the system, the process and the measurement variances and the initial values of $\mathbf{x}_k$ and $\mathbf{P}_k^-$ at time instant $k = 0$. The best possible estimation would be when the trace of $\mathbf{P}_k$ is made zero. This is possible if the measurement is noiseless ($R_k = 0$) but is seldom true in real applications as the measurements are noisy.

In the forthcoming chapters, we discuss how the Kalman filter can be used in the proposed speech coding, methods of choosing a measurement vector for optimal estimation and also present implementation details of the speech coder.

CHAPTER 6. APPLICATION TO SPEECH CODING

In Chapter 2 we pointed out that speech can be considered as piece-wise stationary and modeled as the output of an all-pole filter driven by Gaussian white noise. The Kalman estimator (see Chapter 5) also assumes that a random process can be modeled as the output of a linear system driven by white noise. In this chapter, we describe how the LP model and the Kalman estimator algorithms are applied to speech coding.

As seen in Chapter 5, the Kalman estimator works as a predictor and corrector combination. It first predicts the next state and then optimally corrects it using the measurements. Since the LP analysis provides a model which predicts the next speech sample by minimizing the prediction error, the LP model can be used to describe the linear system in the Kalman estimator. Also in LP analysis, the residual variance is obtained by inverse filtering the input speech. This residual variance can be used to provide an estimate of the variance of the white noise input to the Kalman estimator. Since speech is stationary only for short segments (10-20ms) the parameters of the linear system in the Kalman estimator have to be updated at least every 20ms so as to obtain optimal estimates. This is achieved by doing the LP analysis every 20ms or lesser using either forward adaptive or backward adaptive technique [see Chapter 3].

In the Kalman estimator problem, the system is represented in state-space for-

mulation in terms of the process equation given by

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{w}_k \tag{6.1}$$

where,

$\mathbf{x}_k$ - (nx1) process state vector at time t_k,

$\mathbf{\Phi}_k$ - (nxn) state transition matrix at time t_k relating $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$, and

$\mathbf{w}_k$- (nx1) process noise vector at time t_k.

The system model arising from LP analysis of speech is characterized by just a short term predictor or a combination of short term and long term predictor. For simplicity, let us assume that only a short term predictor of order p is used. The speech sample at time instant t_k can then be represented as

$$s_k = \hat{s}_k - e_k = -\sum_{m=1}^{p} a_m s(k-m) - e_k \tag{6.2}$$

where a_m's are the parameters of the all-pole filter. Suppose the state vector at time t_k consists of the present sample and past $p-1$ samples as in Eqn. 6.3

$$\mathbf{x}_k = \begin{bmatrix} s_k \\ s_{k-1} \\ s_{k-2} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ s_{k-p+1} \end{bmatrix}_{(p\text{x}1)} \tag{6.3}$$

then using Eqn. 6.2 we can identify the state-transition matrix $\mathbf{\Phi}_k$ as

$$\mathbf{\Phi_k} = \begin{bmatrix} -a_1 & -a_2 & . & . & . & -a_m & . & . & -a_p \\ 1 & 0 & 0 & 0 & 0 & . & . & . & 0 \\ 0 & 1 & 0 & 0 & 0 & . & . & . & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & . & . & 1 & 0 \end{bmatrix}_{(p \text{x} p)} \tag{6.4}$$

and the white noise input $\mathbf{w}_k$ as

$$\mathbf{w}_k = \begin{bmatrix} 1 \\ 0 \\ 0 \\ . \\ . \\ . \\ . \\ 0 \end{bmatrix}_{(p \text{x} 1)} \epsilon_k. \tag{6.5}$$

The measurements are formed by $\mathbf{H}_k$ operating on $\mathbf{x}_k$, i.e. as a linear combination of current and past $p-1$ speech samples. The selection of the measurement vector $\mathbf{H}_k$ is crucial in obtaining optimal estimates and will be discussed in the next section. To reduce the number of bits transmitted, these measurements are taken at sparse but regular intervals and the innovations obtained from them [Eqn. 5.13] are quantized

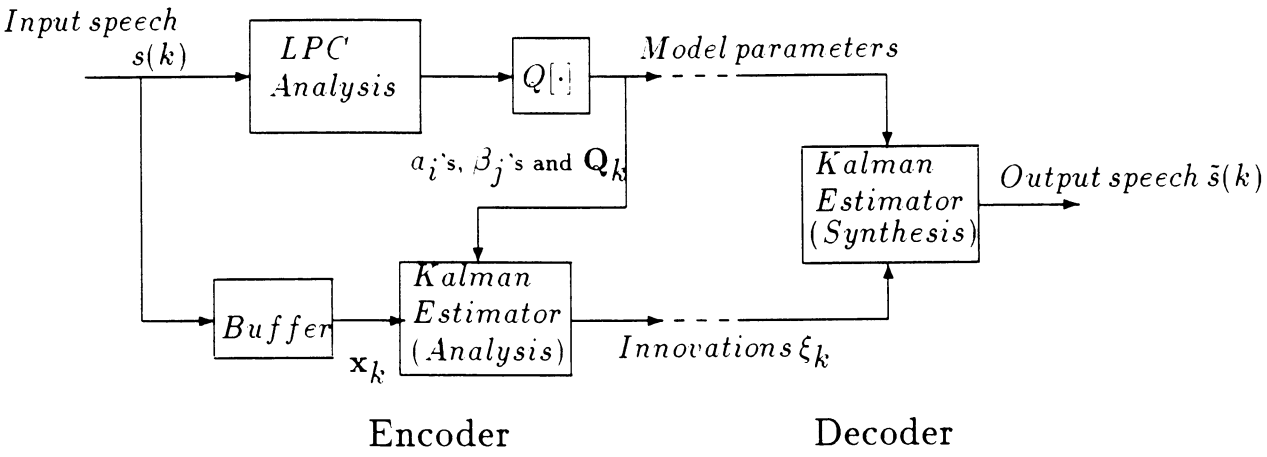


Figure 6.1: Block diagram of Kalman estimator based forward adaptive coder

and transmitted. The process noise variance $\mathbf{Q}_k$ is obtained by inverse filtering the synthesized speech and estimating its variance. The measurement noise variance R_k is obtained as a function of the innovation variance [Eqn. 5.14] and the quantizer design.

Previous Work on Speech Coders Using the Kalman Estimator

Speech coders utilizing the concepts of Kalman estimator have been studied earlier[13]-[16]. A block diagram of such a coder is shown in Figure 6.1. The model used here is forward adaptive and consists of both a short-term predictor and a pitch predictor.

The input speech is blocked (10-20ms) and windowed with a Hamming window. LP analysis is then done and the short-term predictor parameters (a_m's) are obtained using the *stabilized covariance* method with high frequency correction[13]. The input speech is inverse filtered using the short-term predictor alone and the pitch (M) and

pitch predictor coefficients (β_j's) are computed from this. The excitation variance is computed from the residuals obtained by inverse filtering this signal again using the long-term predictor. The transfer function of the prediction filter with both the short-term and the long-term filter can be written as

$$A(z) = \sum_{m=1}^{p} a_m z^{-m} + \sum_{j=M}^{M+q-1} \beta_{j+1-M}(z^{-j} - \sum_{i=1}^{p} a_k^{(j)} z^{-j-i}). \qquad (6.6)$$

The first row of the state transition matrix in equation 6.4 is replaced in this case by the coefficients of z^{-i} of Eqn. 6.6 and size n of the state vector is $M+p+q-1$. The size of the state vector is dependent on the maximum pitch and so it is usually of a high order (e.g., 128). The measurements are formed as a linear combination of the input speech in the Kalman filter. The measurement vector $\mathbf{H}_k$ is selected optimally in a way similar to the selection of the Kalman gain vector to minimize the trace of the *a posteriori* error covariance matrix $\mathbf{P}_k$. This means that the Kalman estimator has to be executed even at the encoder. Finally, the innovations which are formed as a difference of the measurements and their estimates ($\mathbf{H}_k \hat{\mathbf{x}}_k^-$) are quantized using a non-uniform quantizer and transmitted at regular but sparse intervals (L). The short-term predictor coefficients, pitch, pitch predictor coefficients, and the excitation variance are also quantized and transmitted. At the decoder, the Kalman estimator which acts as a synthesis filter makes use of the model parameters and the quantized innovations to synthesize speech. Since the innovations play an important role in the synthesis of speech, this coder is called an Innovations-Assisted Linear Predictive Coder (IALPC). Moreover, the coder is a forward adaptive IALPC as the model parameters are obtained by forward adaptation technique (see Chapter 3). The optimal measurement selection procedure is discussed next.

Selection of a Measurement Model

The measurements are obtained as linear combinations of the present and past speech samples [Eqn. 5.2]. These measurements are taken at regular but sparse intervals (L) so as reduce the number of bits required to code them. So the *a priori* estimates $\hat{\mathbf{x}}_k^-$ are L-step predicted and not one-step predicted. This affects the performance of the speech coder especially when used in backward adaptive speech coders. The selection of the measurement vector is therefore an important factor in obtaining good performance since better measurements can compensate for bad prediction. The states of the speech process, i.e., the past speech samples, are available at the encoder. So we can form any linear combination of the states to obtain a measurement every time one is required. This would provide us with better estimates than if we use a fixed measurement vector (a fixed linear combination) for all measurements[13]. The method of selecting an optimal measurement model which is used with the forward adaptive model is discussed below.

Optimal Measurement Vector We use the same minimization criterion as used in the Kalman filter algorithm to arrive at an optimum Kalman gain vector to obtain the optimum measurement vector also. This measurement vector is chosen to minimize the trace of the *a posteriori* error covariance matrix $\mathbf{P}_k$. Given the *a priori* estimate $\hat{\mathbf{x}}_k^-$ and the *a priori* error covariance matrix $\mathbf{P}_k^-$ we have to obtain a $\mathbf{H}_k$ which will minimize the trace of $\mathbf{P}_k$. Combining Eqns. 5.12 and 5.17 we can represent $\mathbf{P}_k$ as

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{P}_k^- \mathbf{H}_k{}^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k{}^T + R_k)^{-1} \mathbf{H}_k \mathbf{P}_k^- . \tag{6.7}$$

The trace of $\mathbf{P}_k$ can now be written as

$$tr[\mathbf{P}_k] = tr[\mathbf{P}_k^-] - tr[\mathbf{P}_k^-\mathbf{H}_k{}^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k{}^T + R_k)^{-1}\mathbf{H}_k\mathbf{P}_k^-]. \tag{6.8}$$

Instead of minimizing $tr[\mathbf{P}_k]$ with respect to $\mathbf{H}_k$ we can maximize the quantity $tr[\mathbf{P}_k^-\mathbf{H}_k{}^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k{}^T + R_k)^{-1}\mathbf{H}_k\mathbf{P}_k^-]$. Since $\mathbf{P}_k^-$ is symmetric and $(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k{}^T + R_k)$ is a scalar the quantity to be maximized can be written as

$$\frac{tr[(\mathbf{H}_k\mathbf{P}_k^-)^T(\mathbf{P}_k^-\mathbf{H}_k)]}{\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k{}^T + R_k} = \frac{(\mathbf{H}_k\mathbf{P}_k^-)(\mathbf{P}_k^-\mathbf{H}_k{}^T)}{\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k{}^T + R_k}.$$

The measurement noise variance R_k is dependent on the quantizer design and the innovation variance Λ_k. So it can be represented as $\delta(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k{}^T)$ where δ is a function of the number of quantization levels of the quantizer. Making use of this dependence of R_k on the innovation variance the quantity to be maximized can be modified to

$$\frac{1}{1+\delta}\left(\frac{\mathbf{H}_k\mathbf{P}_k^-\mathbf{P}_k^-\mathbf{H}_k{}^T}{\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k{}^T}\right). \tag{6.9}$$

This quantity is maximized if $\mathbf{H}_k$ is chosen as the eigenvector of $\mathbf{P}_k^-$ corresponding to the largest eigenvalue. The measurement vector thus selected is used in forming the scalar measurements which are used to correct the predicted estimates and obtain optimal estimates.

This forward adaptive coder is computationally highly complex because updating of the error covariance matrix in the Kalman estimator takes considerable time. This makes it difficult to implement the coder in real-time. Another problem is the large coding delay which is inherent in forward adaptive coders[5]. This is due to fact that 10-20ms of speech has to be buffered to compute the model parameters. This

delay causes problems in applications where echo cancellation is done, e.g., in Public Switched Networks (PSN).

CHAPTER 7. PROPOSED CODER

In the previous chapters, we discussed about the speech production model, state-space representation of speech, and how Kalman estimator is used to implement a forward adaptive LALPC. In this chapter, we will put forth the limitations of the forward adaptive LALPC and give a block diagram of the proposed coder which overcomes some of these limitations followed by some discussions on the effect of various parameters on the coder performance.

Motivation

The need to reduce the coding delay and also the complexity of the forward adaptive LALPC motivated us to look at solutions to the problem. In this work, two problems are addressed:

(i) reducing the coding delay to less than 5ms and,

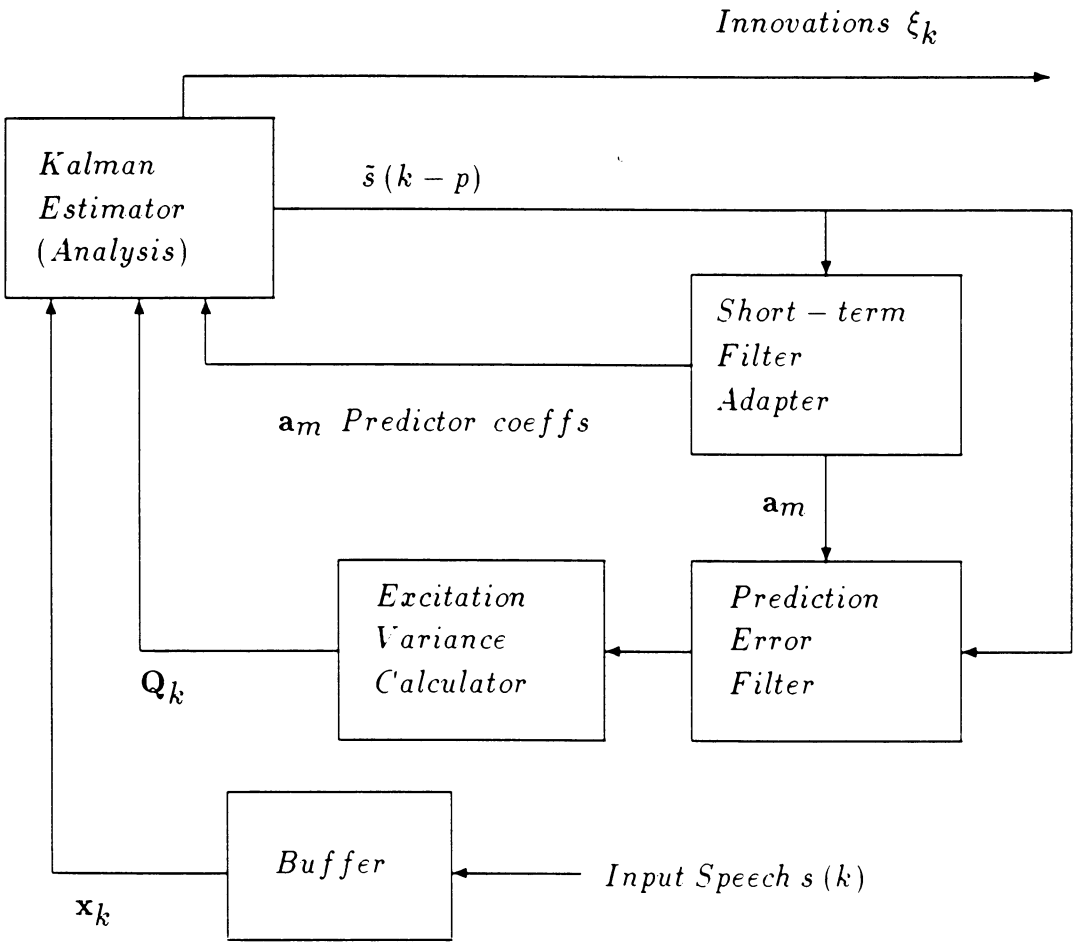
(ii) reducing the complexity of the coder to enable real-time implementation.

To tackle the coding delay problem, we used a backward adaptive linear predictor along with a recursive windowing technique for autocorrelation value calculation which eliminates the need for buffering speech samples. The computational complexity of the coder has been reduced in various stages which include the linear predictor and the Kalman estimator. The computational complexity of the error covariance

matrix update in the Kalman estimator has been reduced by using a dimensionally smaller error covariance matrix. Also the choice of a sub-optimal measurement vector reduced the complexity further. The algorithmic delay has been restricted to 1.25ms by using a vector size of 10 samples. In the next section we provide a description of the proposed coder along with its block diagram. This is followed by discussions related to the choice of the predictor order, choice of the recursive window size, size of the Kalman state vector, and the choice of the measurement vector.

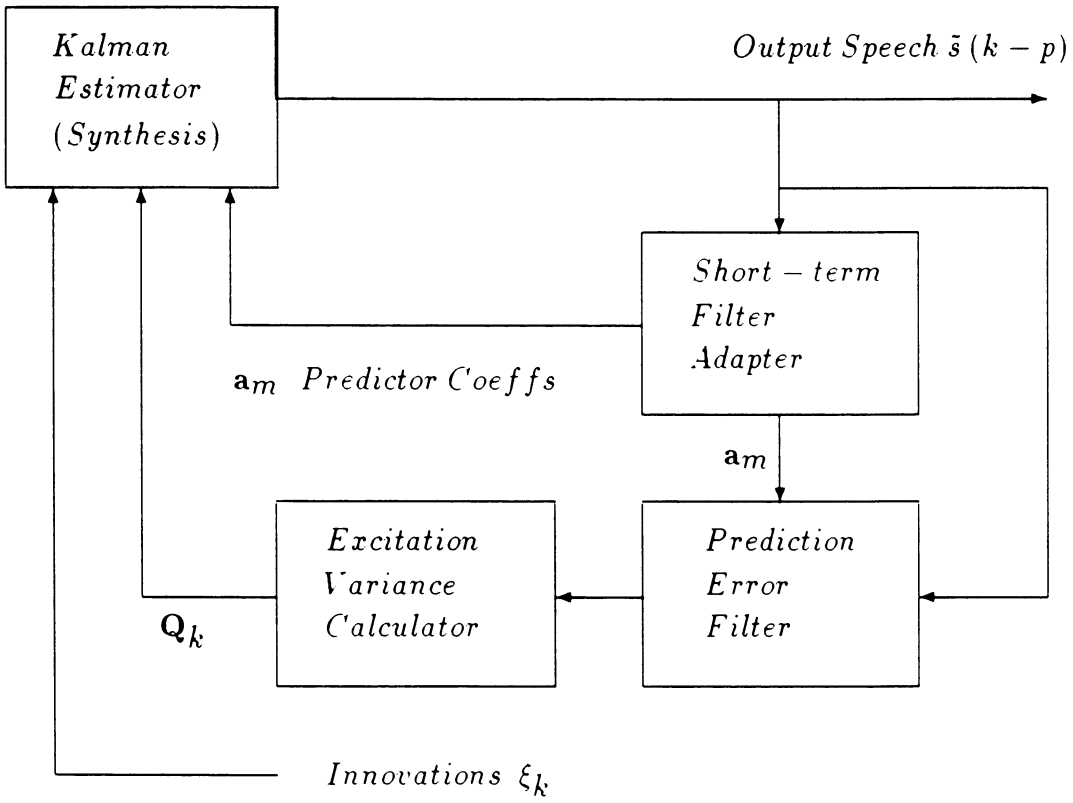
Block Diagrams

The encoder and decoder block diagrams of the proposed coding scheme are shown in Figures 7.1 and 7.2 respectively. At the encoder the input speech is buffered and the state vector $\mathbf{x}_k$ (10 samples) is formed. This is used in forming the measurements in the Kalman estimator which acts as an analysis filter at the encoder. The state transition matrix of the Kalman estimator is defined by the coefficients of the all-pole filter which are computed using a backward adaptive technique in the short-term filter adapter block. The short-term filter adapter makes use of the synthesized speech output of the Kalman estimator to compute the autocorrelation values using a recursive windowing technique as described in Chapter 4. The predictor coefficients are then computed using Levinson-Durbin algorithm (see Chapter 2 Sec. 2). Once the predictor coefficients are obtained the synthesized speech is inverse filtered in the 'prediction error filter' block to obtain the prediction error sequence. The average energy in one error vector (10 samples) is computed to get an estimate of the process noise variance $\mathbf{Q}_k$. These parameters are used in the Kalman estimator to produce the innovations at regular but sparse intervals (L) and also to synthesize speech. The



Encoder

Figure 7.1: Block diagram of Encoder



Decoder

Figure 7.2: Block diagram of Decoder

innovations are then quantized with an adaptive non-uniform quantizer and transmitted to the decoder. As the model parameters are computed from synthesized speech which is available at the decoder also they dont have to be transmitted to the decoder. The decoder has a similar structure like the encoder except that the Kalman estimator in the decoder acts as a synthesis filter. The quantized innovations received along with the model parameters that are computed at the decoder are used to reconstruct the speech signal.

Design of the Proposed Coder

In this section, issues such as (i) selection of an optimum model order which will give a reasonably good prediction at the same time keep the computational complexity low and (ii) modified designs to quantize the innovations so as to improve the objective and subjective quality of the synthesized speech, are discussed. In describing the design procedure, we will have a need to illustrate the influence of different design parameters. This will be done based on Averaged SEGmental Signal-to-Noise Ratio (SEGSNR) values using a speech database. These SEGSNR values are obtained by calculating the signal-to-noise ratio in dB for nonoverlapping blocks 16ms in duration, and averaging all the SNR values omitting those blocks which have energies 40dB less than the average energy of the speech sentence. The speech database consists of six sentences and the details of these sentences are shown in table 7.1. Also plots for one particular vowel /æ/ as in 'Cats' from the sentence "Cats and dogs each hate the other" has been chosen to show the effect of different parameters on the speech quality where necessary.

Table 7.1: Details of representative sentences

Sen Id	Sentence	Sex	Approx. pitch (Hz)	Number of samples
F1	The pipe began to rust while new	F	190	22424
F2	The ripe taste of cheese improves with age	F	220	27054
F3	Open the crate but dont break the glass	F	220	25050
M1	Cats and Dogs each hate the other	M	120	23046
M2	Open the crate but dont break the glass	M	140	23547
M3	Thieves who rob friends deserve jail	M	130	23453

Structure of the Linear Predictor

The choice of the linear predictor model is crucial since the complexity of the coder depends on it. The possible structures are

(i) A short-term filter of a small order ($p \leq 16$)

(ii) A short-term filter of small order and a long-term filter

(iii) A short-term filter of large order (say 50)

If we use structure (ii) or (iii) then we will have to find ways of keeping the size of the error covariance matrix small so as to reduce the computational complexity of its update. If we use structure (ii) like the one which is used in the forward adaptive IALPC [see Chapter 6] then the size of the error covariance matrix will be of a very large size (approx. 128) and so the update will be computationally very intensive. To avoid this we have to use the long-term filter first and then the short-term filter so that the estimation process is limited to the short-term filter. This means that the Kalman estimator will estimate the modified excitation for the long-term filter. The reconstructed speech can be obtained from this by passing it through a filter whose transfer function is $\frac{1}{1-P(z)}$ where $P(z)$ is the transfer function of the long-term predictor [see Chapter 2]. The problems with this structure are: (a) If we use

the long-term filter first it removes some of the short-term correlation also and so the prediction gain of the short-term filter suffers. The overall prediction gain obtained is much less than what could be achieved with a model which has the short-term filter first and then the long-term filter. (b) The use of backward adaptation technique for the model parameter estimation results in a great reduction in the prediction gain with this structure. So using a long-term predictor is not going to help. The only choices we have are to use just a short-term predictor of small order or large order if it provides a significant increase in the prediction gain for L-step prediction which is necessary to keep the bit-rate low. But the complexity of the coder is dependent on the size of the error covariance matrix as discussed earlier and it is proportional to n^2, where n is the size of the state vector. So if we use a 50^{th} order model the computation time is going to increase drastically. When a 50^{th} order model was tried with a state vector also of size 50, there was one more problem apart from the increased complexity. The output synthesized speech was available only after a delay of 50 samples to the short-term filter adapter for processing because it had to be shifted through the *a posteriori* state vector $\hat{\mathbf{x}}_k$. This large delay resulted in very poor prediction gains since the model parameters did not adequately describe the present segment of speech. To avoid this a different technique was used. In this technique, a 50^{th} order model was used for prediction whereas for the error covariance update a 10^{th} order model was used. Both these models were obtained from synthesized speech. With this technique, prediction gains and coder performance were obtained without any quantization of the measurements and will be discussed later in this section.

Figure 7.3 shows one-step prediction gains for various orders of the all-pole model

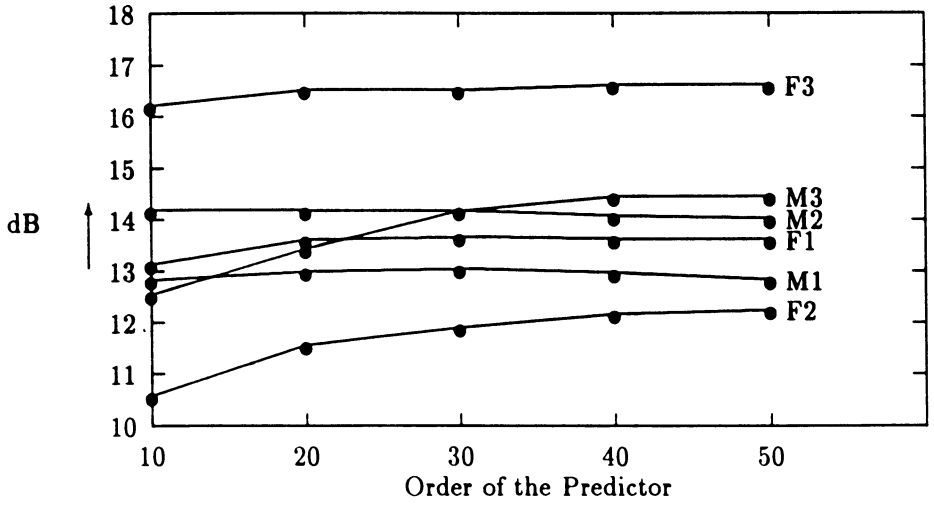


Figure 7.3: Effect of model order on prediction gain (Backward adaptation)

in the proposed coder. Though the average increase in prediction gain for the six sentences is only 0.7dB, the gains for two sentences, namely F2 and M3, show an increase of almost 1.5dB. This led us to investigate how two-step prediction would affect the prediction gains for different model orders. While using a two-step prediction model, the Kalman estimator is started off with a sub-optimal measurement vector and then changed to an optimal measurement vector, the reason for which will be discussed later in this chapter.

Figure 7.4 shows comparison of prediction gains for order $p = 10$ and $p = 50$ for two-step prediction. The prediction gains showed a definite increase for most of the sentences, except M1, with prediction model order 50.

Figure 7.5 shows results of overall performance of the coder for prediction orders 10 and 50. It is clear that the overall performance of the coder with prediction order 50 suffers eventhough it has a higher prediction gain. This may be due to the

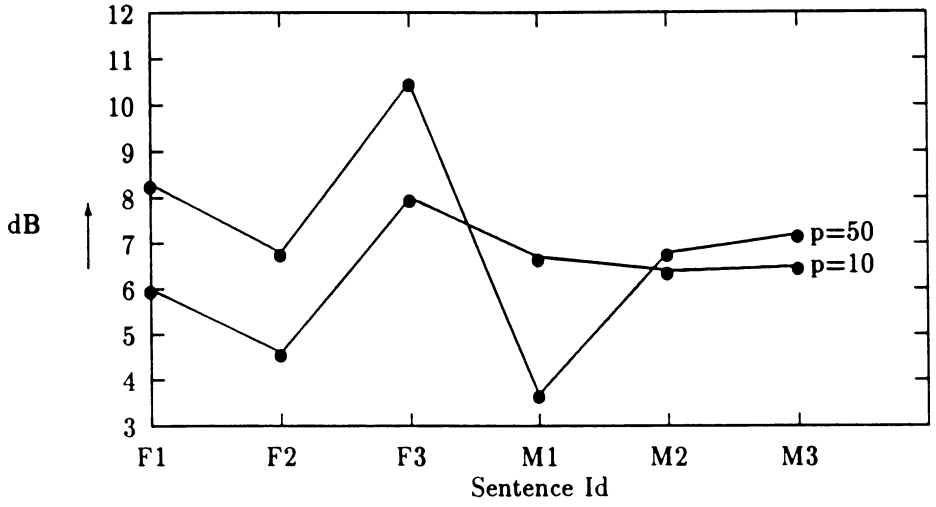


Figure 7.4: Effect of two-step prediction on prediction gain (No quantization of measurements, Optimum $\mathbf{H}_k$)

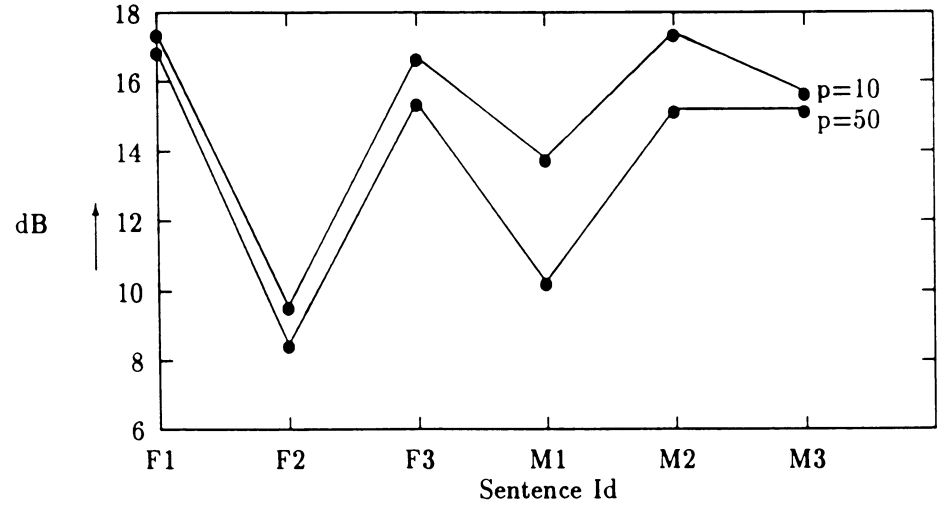


Figure 7.5: Coder performance for predictor order 10 and 50. (L=2, No quantization of measurements, Optimum $\mathbf{H}_k$ vector)

measurement model used which was chosen to reduce the error covariance matrix size. These results made us decide in favor of a coder with a model order $p = 10$ and also in keeping the state vector size equal to the model order. The model parameters are computed using backward adaptation technique and updated every 10 samples (1.25ms). From this point onwards, all our discussions will involve a coder with model order and state vector of size 10 and error covariance matrices of size (10x10) unless otherwise specified.

Estimation of Process Noise Variance

The process noise variance $\mathbf{Q}_k$ is also estimated in a backward adaptive fashion from synthesized speech. This estimation is collectively done by the 'Prediction error filter' and the 'Excitation variance calculator' blocks in figures 7.1 and 7.2. First, the synthesized speech is passed through the prediction error filter whose transfer function is given by Eqn. 2.24 and the error sequence is generated. The length of the error sequence generated is equal to the size of the state vector. The average squared energy of the error sequence is computed to give an estimate of $\mathbf{Q}_k$ and is updated every 10 samples (1.25ms).

Selection of a Measurement Model

The method of selecting an optimum measurement vector was described earlier in Chapter 6 for the forward adaptive model. The computation of this optimum measurement vector requires $n^2 + 2n$ multiplies and $n^2 - n$ adds. A sub-optimum measurement vector can reduce this complexity. The use of optimum measurement vector leads to another interesting problem. To start with, the output of the Kalman

estimator is just zeros and the predictor coefficients are all zeros, except a_0, which is 1. This results in a $\mathbf{P}_k^-$ whose diagonal elements are zeros except the first two diagonal elements which are non-zero because of the process noise variance $\mathbf{Q}_k$ which is given a non-zero initial value as we do not have any *a priori* estimate of its variance. When we find an optimum measurement vector using this $\mathbf{P}_k^-$, we obtain a $\mathbf{H}_k$ of the form $[1,0,0,\cdots,0]$ and also a Kalman gain vector of the form $\mathbf{K}_k^T = [1,0,0,\cdots,0]$. The estimates are corrected only when a measurement is available. So the corrected estimates are alternately zeros.i.e., the samples have values only when a measurement is incident and zero otherwise. This when run through the recursive window algorithm for model parameter estimation, results in predictor parameters which are also alternately zeros. This again results in a $\mathbf{H}_k$ vector of the form $[1,0,0,\ldots,0]$ and the whole process repeats leading to alternate samples of the synthesized speech being zeros. This problem can be avoided by selecting a sub-optimum measurement vector at least in the beginning.

Sub-Optimal Measurement Vector A measurement vector which is sub-optimal but depends on $\mathbf{P}_k^-$ so as to still make a choice based on the error in the *a priori* estimates was sought. This $\mathbf{H}_k$ was obtained as the normalized diagonal entries of $\mathbf{P}_k^-$. The diagonal elements are normalized with the first diagonal element, which is the error variance of the latest sample. The computation required for this sub-optimal measurement vector is just n multiplies. Also the 'alternate zero' problem in the synthesized speech was solved. This is because the first two diagonal elements of $\mathbf{P}_k^-$ are non-zero to start with due to the process noise variance $\mathbf{Q}_k$. So the $\mathbf{H}_k$ vector is of the form $[1,1,0,\cdots,0]$ and the Kalman gain vector will also be of the

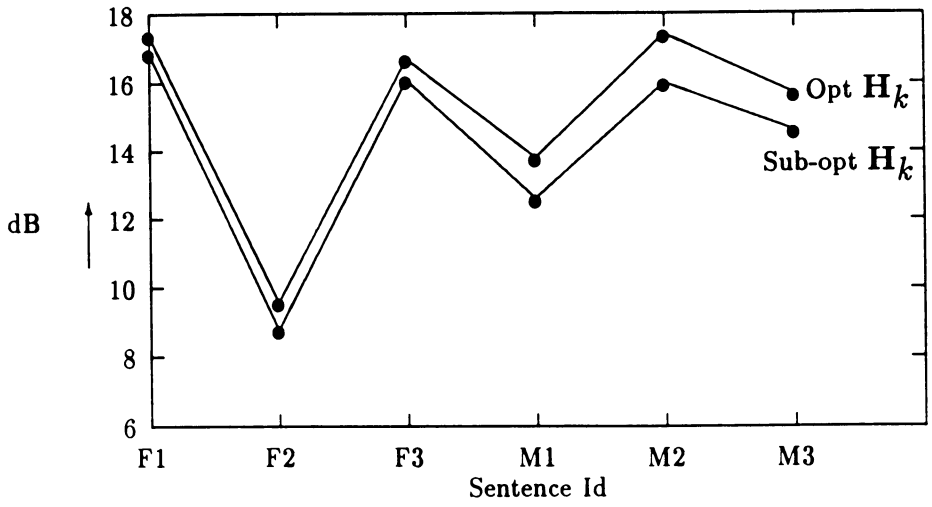


Figure 7.6: Effect of measurement vector on coder performance (L=2, No quantization of measurements)

form $\mathbf{K}_k^T = [1, 1, 0, \cdots, 0]$ initially. This results in the correction of estimates even when there is no corresponding measurement. Therefore, the synthesized samples no longer have alternate zeros and we obtain proper estimates of the model parameters. Even when the coder uses the optimal $\mathbf{H}_k$ vector, it is initiated with the sub-optimal $\mathbf{H}_k$ and then switched to optimal $\mathbf{H}_k$ vector after a certain time (100 samples in our implementation) which is chosen arbitrarily. Performance of the coder with optimal and sub-optimal $\mathbf{H}_k$ vectors are shown in Figure 7.6. There is only 0.8dB SEGSNR drop on an average for the six sentences when sub-optimal $\mathbf{H}_k$ is used.

Quantization of Innovations

Once the measurement vector is obtained the scalar measurements z_k which are used to correct the estimates are computed as linear combinations of the actual

speech samples. These measurements have to be transmitted to the decoder and to keep the bit-rate low we need to quantize the measurements. Instead of quantizing the measurements directly we can also quantize the innovations ξ_k [see Eqn. 5.13]. The advantage in quantizing the innovations is that it is of lesser energy than the measurements and so fewer number of bits are enough to quantize them. The measurements can be easily recovered from the innovations. The measurements and innovations are assumed to have a Gaussian distribution since they are obtained as linear combinations of the actual speech samples. So innovations are quantized with a Gaussian non-uniform quantizer [5] which makes use of the innovation variance (Λ_k) and the measurement mean ($\mathbf{H}_k\hat{\mathbf{x}}_k^-$) for the design. Since only the innovations are transmitted to the decoder, the number of bits used to quantize them will decide the bit-rate of the coder. For example, the innovations are quantized with a 4-bit Gaussian non-uniform quantizer and transmitted at regular but sparse intervals of $L = 2$ to achieve a bit-rate of 16Kbps.

Figure 7.7 shows the actual speech (thin lines) and the synthesis error (thick lines) waveforms for a short segment of speech. This waveform corresponds to the vowel /æ/ as in 'Cat'. It can be seen that the synthesis error is very large at the beginning of every peak.

The prediction gain cannot be improved any further as discussed earlier in this chapter. So the quantized measurements and its related quantities like the measurement mean ($\mathbf{H}_k\hat{\mathbf{x}}_k^-$) and innovation variance (Λ_k) were studied to give some clue about how to keep the quantization noise small in the measurements. Figure 7.8 shows a plot of the measurements with (thick lines) and without (thin lines) quantization. The measurements surely suffer because of the quantization.

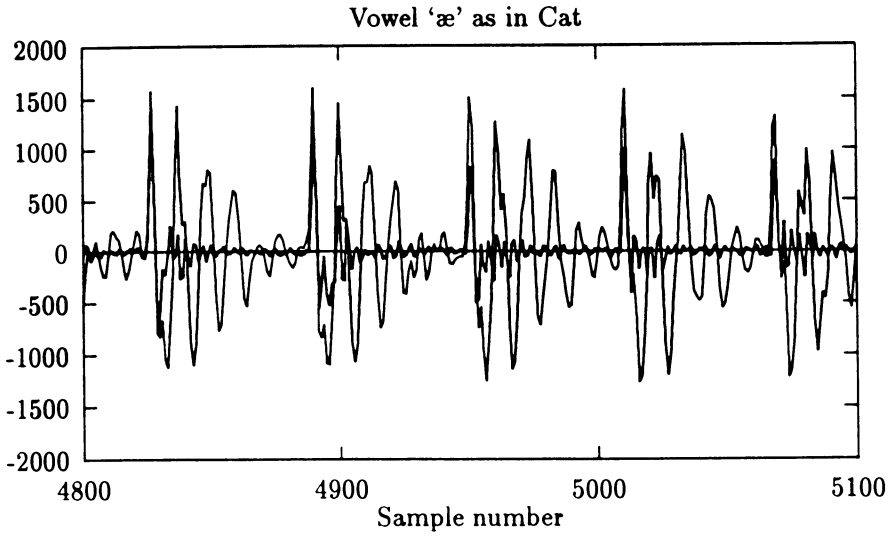


Figure 7.7: Effect of 4 bit quantization on synthesized speech (L=2, Optimum $\mathbf{H}_k$)

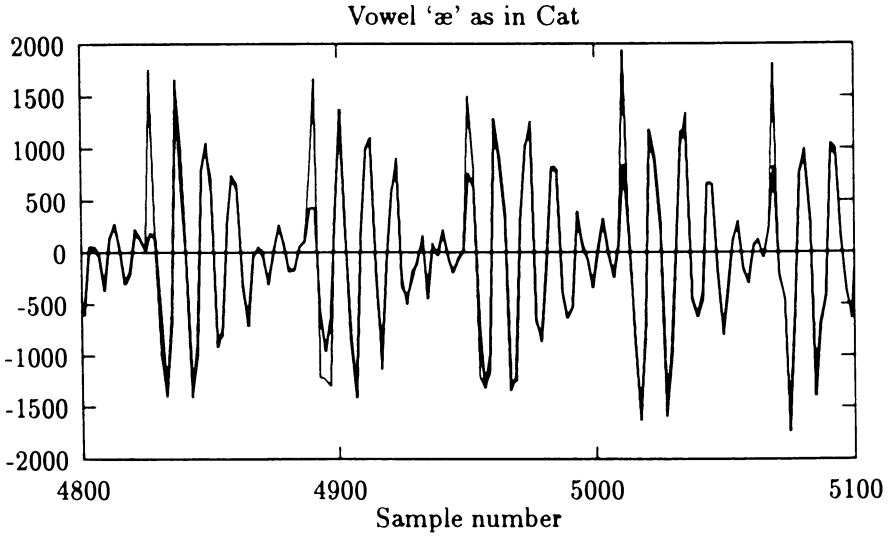


Figure 7.8: Effect of 4-bit quantization on measurements (L=2, Optimum measurement vector)

Figure 7.9 shows a plot of the measurement mean when the innovations are not quantized. The measurement mean is obtained from the past speech samples and so when there is a sudden peak after a small signal its value is still small. This is a problem introduced by backward adaptation.

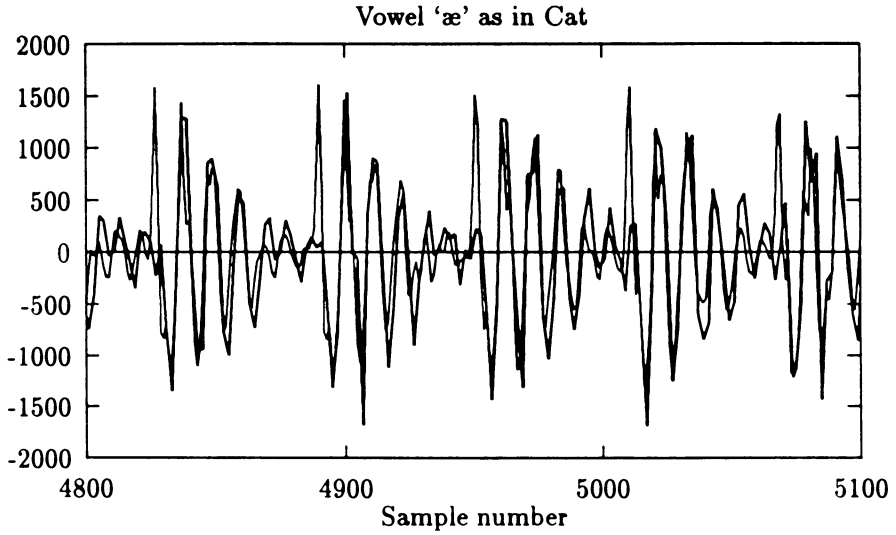


Figure 7.9: Effect of backward adaptation on measurement mean (L=2, No quantization of measurements, Optimum $\mathbf{H}_k$)

The innovation variance (Λ_k) with (thick lines) and without (thin lines) quantization is shown in Figure 7.10. The innovation variance is definitely reduced drastically because of quantization effects.

Since the measurement variance and measurement mean are small, the quantized measurement is clipped. This type of noise introduced in the measurement is called *overload* noise [5] and it occurs when the input signal to the quantizer exceeds the the maximum value that can be handled by the quantizer. One solution to this problem would be to increase the innovation variance somehow so that the step sizes of the

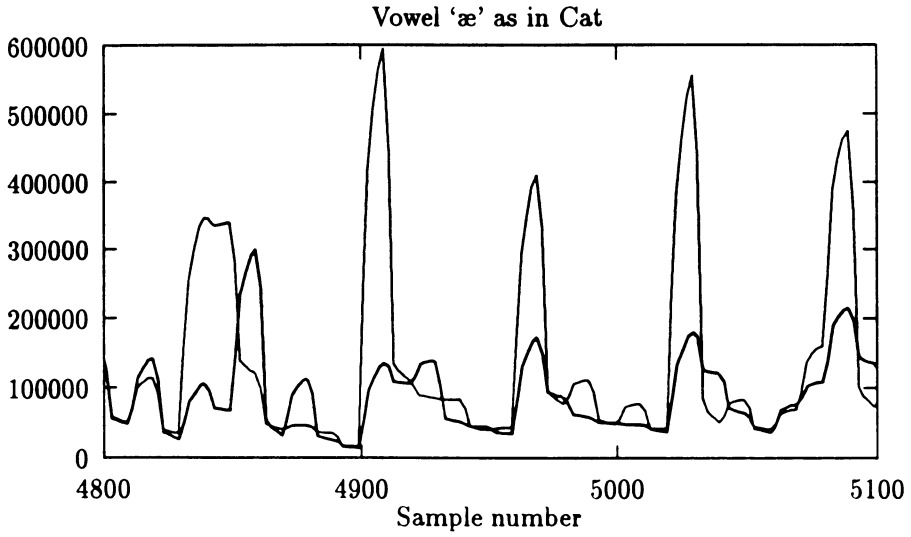


Figure 7.10: Effect of 4-bit non-adaptive quantization on measurement variance (L=2, Optimum $\mathbf{H}_k$)

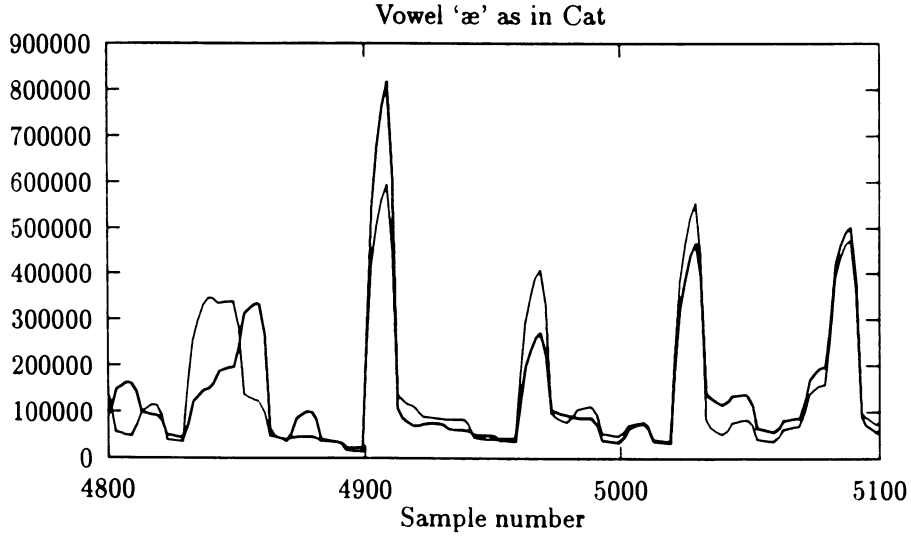


Figure 7.11: Effect of 4-bit adaptive quantization on measurement variance (L=2, Optimum $\mathbf{H}_k$)

quantizer are larger and thus accommodate higher amplitude signals also. A simple solution was provided by increasing the innovation variance 4 times. An average increase of 1.5dB in SEGSNR was observed in the coder performance. To improve the measurements further, an adaptive quantizer [5] [20] was used to quantize the innovations in addition to scaling the innovation variance up 4 times. Adaptive quantization is described below. The discussion will be very brief as it is a well understood concept.

Adaptive Quantization Adaptive quantizers are so called because they operate with a time varying step size depending on the input statistics. Quantizer adaptation to changing inputs are of three important types:

(i) Adaptation to the changing variance of the input

(ii) Adaptation to the changing input *pdf* (Probability Density Function)

(iii) Adaptation to a changing mean of the input

In our case we have a problem of sudden changes in the innovation variance. So we will briefly deal with type (i) here. Let us assume that the step-size of a uniform quantizer is $\Delta(k)$, where k refers to the time index of the input signal and let the number of quantizer levels be $2N$ (N each for positive and negative vlaues). The problem we have in hand is then to adapt this step-size $\Delta(k)$ to the changing input variance $\mathbf{Q}_k$. The step-size adaptation works as follows. The step-size is changed by a multiplicative quantity M_i, where $i = 1, 2, \ldots, N$ refers to the output level. If the previous input signal amplitude falls in the 'inner' levels, say for example in the first 4 levels of a 4-bit quantizer, then the step-size is probably large and the step-size is reduced by multiplying the variance of the input signal by a quantity less than 1

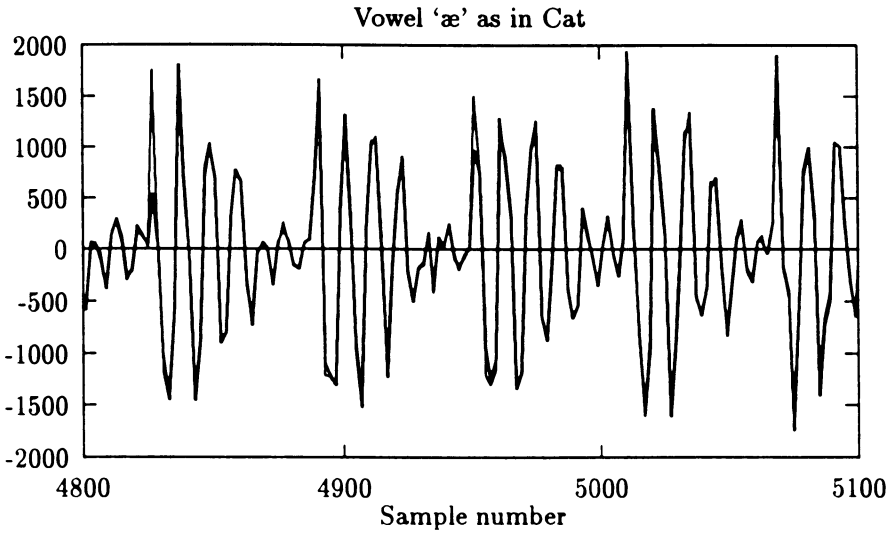


Figure 7.12: Effect of 4-bit adaptive quantization on measurements (L=2, Optimum $\mathbf{H}_k$)

for the next input sample. If the amplitude of the input sample falls in one of the 'outer' levels, for example in the top 4 levels, the step-size is probably small and so it is increased by multiplying the variance of the input signal by a quantity greater than 1. These multipliers were chosen through trial and error so as give the best performance possible and we chose the following multipliers for the 4-bit adaptive quantizer to quantize the innovations. $M_1 = 0.625, M_2 = 0.625, M_3 = 0.625, M_4 = 0.625, M_5 = 1.5, M_6 = 1.5, M_7 = 2$ and $M_8 = 2$.

Figure 7.11 shows how the measurement variance improves with the use of the adaptive quantizer. The estimate of the innovation variance seem to track the actual variance much better than with a non-adaptive quantizer. In the figure the thick lines show the innovation variance for different samples when the innovations are quantized with an adaptive quantizer and the thin lines are the innovation variances when the

innovations are not quantized. The adaptive quantizer designed as above provided almost 2dB increase in SEGSNR over the non-adaptive quantizer. Figure 7.12 shows the plot of measurements when the innovations are quantized with (thick lines) an adaptive quantizer and without (thin lines) any quantization.

Error Spectrum Shaping

Another issue related to speech coders is reconstruction error spectrum shaping. It has been found that the speech hearing mechanism works in such a way that we can have more noise in the formant regions than in the inter-formant regions. According to this experiment the subjective quality does not depend on reducing the squared error alone. The error spectrum should be distributed such that there is more noise in the formant regions and less in the inter-formant regions where the error is perceptually less noticeable. This redistribution of the error energy can be done by means of 'pre-filtering' actual speech and then 'post-filtering' the synthesized speech by means of appropriate filters and also minimizing 'weighted' error instead of actual error. This reconstruction error, which is the error between the synthesized and the actual speech, can be 'weighted' by passing the error through a filter whose frequency response is approximately the inverse of the speech spectrum. This will shape the reconstruction error as desired. One such filter is [7]

$$W(z) = \frac{1 + \sum_{m=1}^{p} a_m z^{-m}}{1 + \sum_{m=1}^{p} a_m \gamma^m z^{-m}} \tag{7.1}$$

where the parameter γ controls the shape of the spectrum. For example, a value of $\gamma = 1$ results in no perceptual weighting and $\gamma = 0$ makes the reconstruction error proportional to the envelope of the speech spectrum. A good choice for γ is 0.73 for speech signals sampled at 8kHz.

Perceptual weighting is done as follows in our speech coder. First, the input speech is weighted using the weighting filter $W(z)$ and the measurements are computed from this weighted speech. So $W(z)$ is used as the 'pre-filter'. We should also synthesize weighted speech and so the transfer function of the synthesis filter will be

$$F(z)\,W(z) = \frac{1}{1 + \sum_{m=1}^{p} a_m \gamma^m z^{-m}}. \tag{7.2}$$

From this equation we can deduce that weighted speech can be synthesized by replacing the predictor parameters a_m's by $\gamma^m a_m$'s in the state vector update equation [Eqn. 5.11] and the error covariance matrix update equation [Eqn. 5.12]. Once weighted synthesized speech is obtained by minimizing the 'weighted' error, it is deweighted by means of a 'post-filter' of the form $1/W(z)$. The parameters for the weighting filter $W(z)$ are usually computed from actual speech in most coders. But these coefficients have to be transmitted to the decoder if done this way. So to avoid increasing the bit-rate we use the same predictor coefficients that were computed from synthesized speech.

In figures 7.13 and 7.14 the thick lines represent the spectrum of error and the thin lines represent spectrum of actual speech. The error spectrum which is almost white in Figure 7.13 has been shaped in Figure 7.14 and it is seen to follow the shape of the actual speech spectrum.

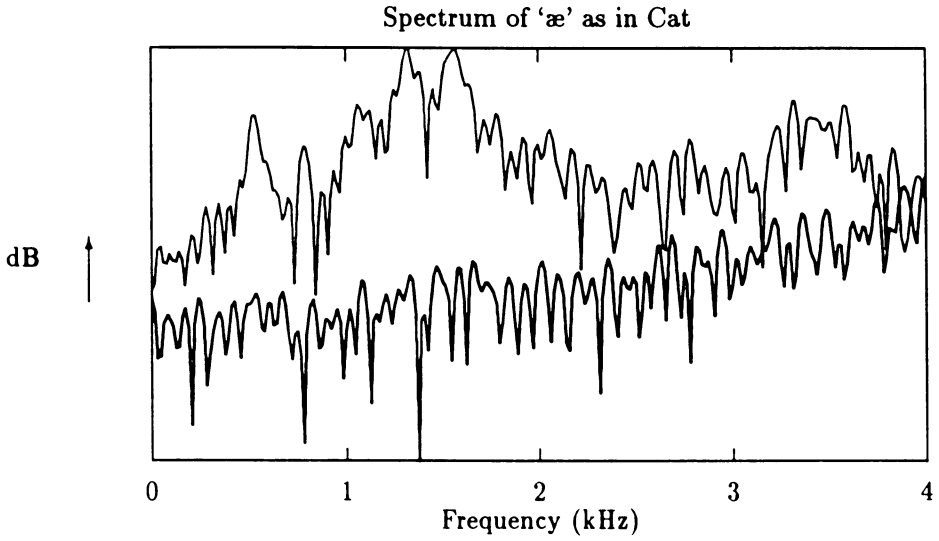


Figure 7.13: Spectrum of /æ/ as in Cats without error shaping ($\gamma = 1.0$)

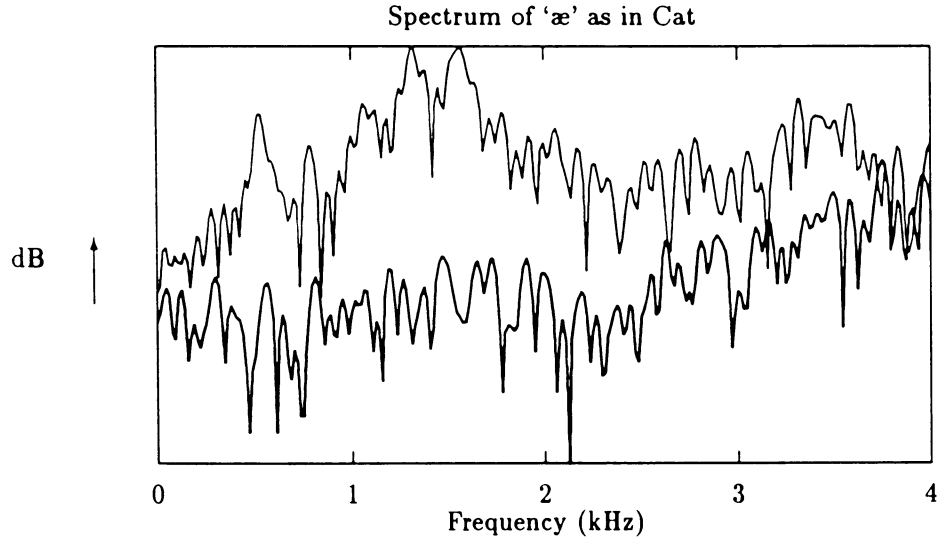


Figure 7.14: Spectrum of /æ/ as in Cats with error shaping($\gamma = 0.73$)

CHAPTER 8. IMPLEMENTATION AND RESULTS

In Chapter 6 we had discussed issues related to the design of different functional blocks of the proposed coder. In this chapter we will talk about the implementation and working of the coder and also present results for a 16kbps coder.

Implementation

In this implementation,

(i) the speech signals are sampled at 8kHz and so the interval between two samples is 0.125ms,

(ii) the all-pole model used is of order $p = 10$,

(iii) a group of 10 consecutive speech samples is called a 'vector' and in the Kalman estimator it will be called a 'state vector'; a sequence of 10 prediction error values will form an error vector,

(iv) since the state vector of size 10, the related error covariance matrix is of size 10x10,

(v) innovations produced by the Kalman estimator (analysis) are quantized with an adaptive Gaussian non-uniform quantizer and transmitted at regular but sparse interval of L=2 to the decoder,

(vi) the model parameters and the process noise variance are updated once every

vector, and

(vii) the error spectrum is shaped by using a weighting filter $W(z)$ given by Eqn. 7.1; the parameter γ which controls the shape of the error spectrum takes a value 0f 0.73.

The synthesized speech, which is the output of the Kalman estimator, is used in the computation of the model parameters and the process noise variance estimate. The synthesized speech samples are fed to the recursive windowing algorithm which computes the autocorrelation coefficients. These coefficients are computed only once every vector as we need to update the model parameters only once in 10 samples. The autocorrelation coefficients that are obtained are passed to the Levinson-Durbin algorithm which computes the parameters of the all-pole model. An error vector is obtained by passing synthesized speech through the prediction error filter. The average energy in the error vector is computed to give an estimate of the process noise variance $\mathbf{Q}_k$. This is also updated once every vector.

At the start of a speech sentence, the model parameters are all zeros and the estimate of $\mathbf{Q}_k$ will also be zero. If this is used in the error covariance update the $\mathbf{P}_k^-$ matrix will contain only zeros and so will result in a 0/0 operation in the calculation of $\mathbf{H}_k$. To avoid this problem $\mathbf{Q}_k$ is limited to a minimum value of 1. These parameters are now used by the Kalman estimator to analyze and synthesize speech. At the encoder, the input speech is buffered, the state vector $\mathbf{x}_k$ is formed, and given to the Kalman estimator (analysis). The state transition matrix $\mathbf{\Phi}_k$, whose elements are defined by the all-pole model parameters, is used to update the *a posteriori* state estimate and form the *a priori* (predicted) state estimate. A two-step prediction is done since the measurements are available only once in two samples. The state transition matrix is also used in updating the error covariance matrix. The estimate

of the process noise variance $\mathbf{Q}_k$ which is used in this update is obtained as explained earlier. The sub-optimal measurement vector $\mathbf{H}_k$ is obtained from $\mathbf{P}_k^-$ and used in the formation of the measurements.

The innovations are then formed as a difference between the measurements and their estimates and quantized with an adaptive Gaussian non-uniform quantizer. These innovations are then used to correct the *a priori* estimates and obtain the *a posteriori* estimates (filtered) of the state vector. The two samples at the bottom of the *a posteriori* state estimate vector are output as synthesized speech. To start with, the *a priori* and *a posteriori* state vectors are zeros and then are corrected as the measurements are given. Since the measurements correspond to the estimate on top of the state vector buffer and the output synthesized speech samples correspond to the bottom of the state vector there is an algorithmic delay of 10 samples (1.25ms). In our implementation we did not have a separate decoder since the function of the encoder and decoder are essentially the same except for the generation of the innovations at the encoder.

Results

As we have already dealt with the effect of different parameters like two-step prediction, model order, measurement vector, and quantizer design in Chapter 6 we will just present results for coder with an implementation as described earlier with and without quantization of innovations. Results are provided for performance of coder with and without error shaping. Both optimum and sub-optimum measurement vectors have been used in these tests. In all the experiments, sentences described in Table 7.1 were used and SEGSNR was used as an objective measure to quantify

the performance. The algorithm was also tested by subjective (informal hearing) measures.

We begin by showing how the innovations help in correcting the estimates. Figure 8.1 shows plots of the prediction gain, the *a posteriori* gain, and the actual output SEGSNR of the sentences. The average prediction gain is about 5dB. After one correction the average SEGSNR increases to almost 11dB. Because of the choice of the measurement vector the estimates are corrected till the synthesized speech samples come out of the Kalman estimator. This is the reason why we see a further improvement in the gain (shown by the Output SNR).

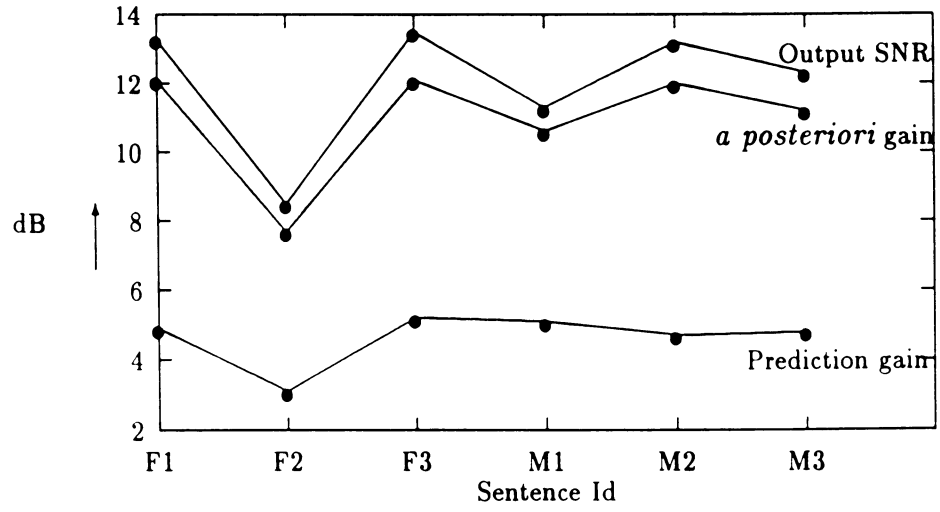


Figure 8.1: Improvement in estimates due to measurement vector (L=2, Optimum $\mathbf{H}_k$ vector)

Figure 8.2 shows plots of the actual speech, reconstructed speech, reconstruction error and the segmental SNRs for sentence M2. An optimum measurement vector

was used, the innovations were quantized with a 4-bit adaptive quantizer and no error shaping was done. The reconstruction is quite good for voiced speech sounds which is evident from the segmental SNRs and reconstructed speech. Sometimes the innovations fail to correct the estimates properly because of quantization effects explained in the previous chapter. This is the reason why we find some prominent spikes in the reconstruction error. The segmental SNRs show that the performance is between poor and medium for unvoiced sounds [low-amplitude signals in Figure 8.2].

Objective Quality

As mentioned earlier, SEGSNR is used as the objective quality measure. Tables 8.1 and 8.2 show results for the coder when optimum measurement vector was used. When the innovations were not quantized the average SEGSNR of the 6 sentences was 15.1dB without any error shaping and it dropped to 14.5dB on error shaping. When the innovations were quantized with 4-bits, the average SEGSNR dropped to 12dB without error shaping. With error shaping an average of 10.2dB was obtained.

Tables 8.3 and 8.4 present results for the coder when the sub-optimum measurement vector was used. There was only an average drop of 0.8dB in SEGSNR when compared to the results obtained with the optimal measurement vector but the computational complexity was significantly reduced (to be discussed later). The objective quality of reconstructed speech deteriorates with error shaping. In speech coding, we are interested more in the perceptual quality than the objective quality. Plots of the spectrum with and without error shaping [Figures 8.4 and 8.3] show how the error spectrum is shaped. Without error shaping the error spectrum is almost

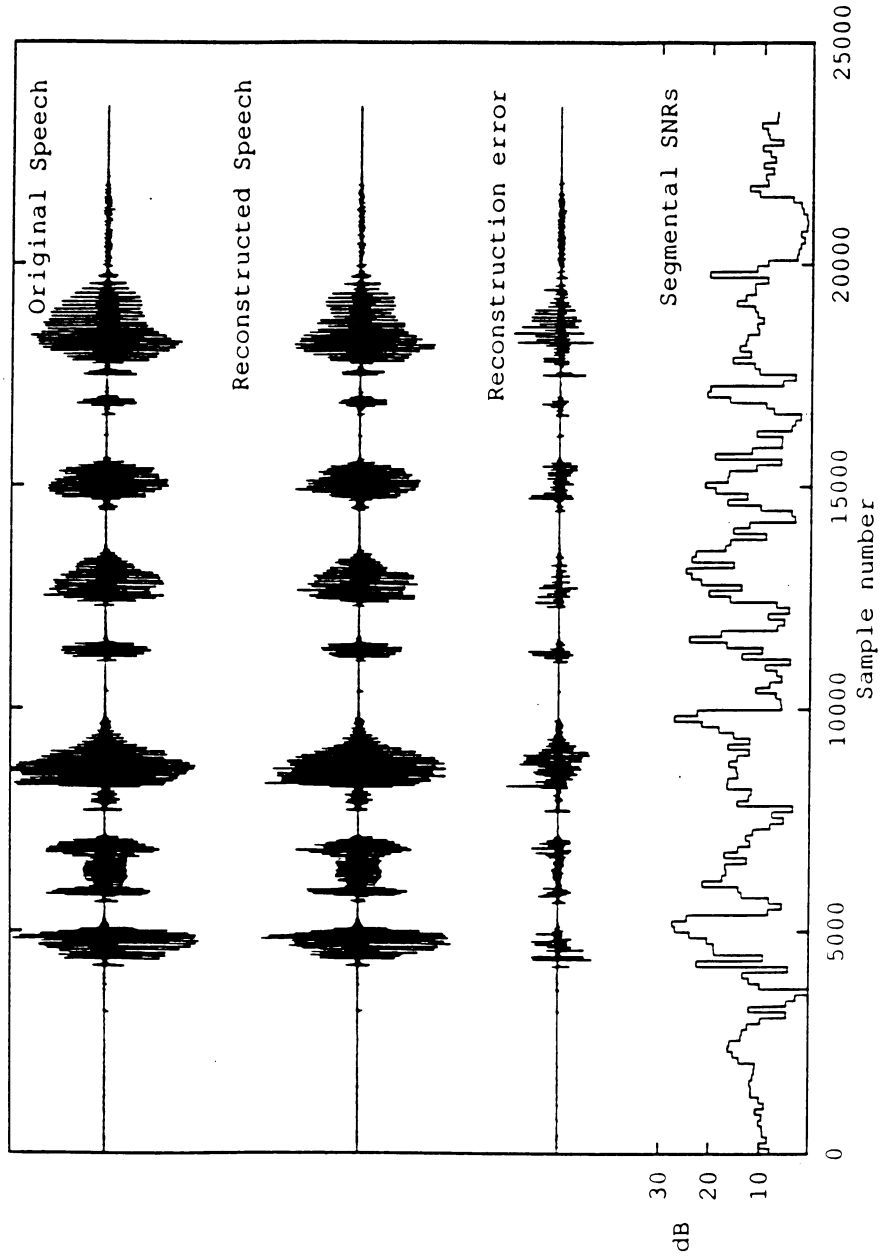


Figure 8.2: Illustration of Coder Performance

Table 8.1: Coder performance without quantization of measurements (L=2, Optimum $\mathbf{H}_k$ vector)

	SEGSNR (dB)	
Sen Id	Without error shaping	With error shaping $\gamma = 0.73$
F1	17.4	17.2
F2	9.6	8.0
F3	16.7	16.4
M1	13.8	13.5
M2	17.4	16.9
M3	15.7	15.1

Table 8.2: Performance of 16Kbps coder (L=2, 4-bit adaptive quantization of measurements, Optimum $\mathbf{H}_k$ vector)

	SEGSNR (dB)	
Sen Id	Without error shaping	With error shaping $\gamma = 0.73$
F1	13.3	11.0
F2	8.5	7.4
F3	13.5	11.0
M1	11.3	10.0
M2	13.2	11.2
M3	12.3	10.3

white but after error shaping it follows the spectrum of the actual speech. Informal listening tests (see next section) show that the perceptual speech quality has definitely improved after error shaping.

Subjective Quality

Subjective quality tests were done informally since a formal perceptual hearing test would involve time and money beyond the scope of this project. These tests were done with the help of two listeners. Listener 1 has prior experience in speech perception but listener 2 does not have any such experience. These two listeners compared the synthesized speech sentences (with and without error shaping) with 7bit, 6bit and 5bit μ-law quantized speech sentences. They graded the synthesized speech as better than (B), worse than (W) and equal to (E) the μ-law coded sentences. A B* indicates that the performance is slightly better and a W* that it is slightly worse.

Tables 8.5 and 8.6 present scores given by the two listeners. Sentence M2 has a quality which is close to a 7bit μ-law coded speech whereas M1 is close to 6bit μ-law coded speech. The subjective performance varies from sentence to sentence and the two listeners have different opinions about the quality which is evident from the results shown.. But both listeners agreed that reconstructed speech with error shaping was better than reconstructed speech without error shaping. They also mentioned that in most of the sentences the high frequency components were affected. The trained listener (listener 1) commented that intelligibility was quite good in the synthesized speech but speaker identification was affected to a small extent.

Table 8.3: Coder performance without quantization of measurements (L=2, sub-optimum $\mathbf{H}_k$ vector)

	SEGSNR (dB)	
Sen Id	Without error shaping	With error shaping $\gamma = 0.73$
F1	16.9	16.7
F2	8.8	9.0
F3	16.1	15.6
M1	12.6	12.5
M2	16.0	15.9
M3	14.6	14.6

Table 8.4: Performance of 16Kbps coder (L=2, 4-bit adaptive quantization of measurements, sub-optimum $\mathbf{H}_k$ vector)

	SEGSNR (dB)	
Sen Id	Without error shaping	With error shaping $\gamma = 0.73$
F1	12.7	10.3
F2	7.6	6.8
F3	13.1	10.8
M1	10.6	9.4
M2	12.3	10.6
M3	11.6	9.7

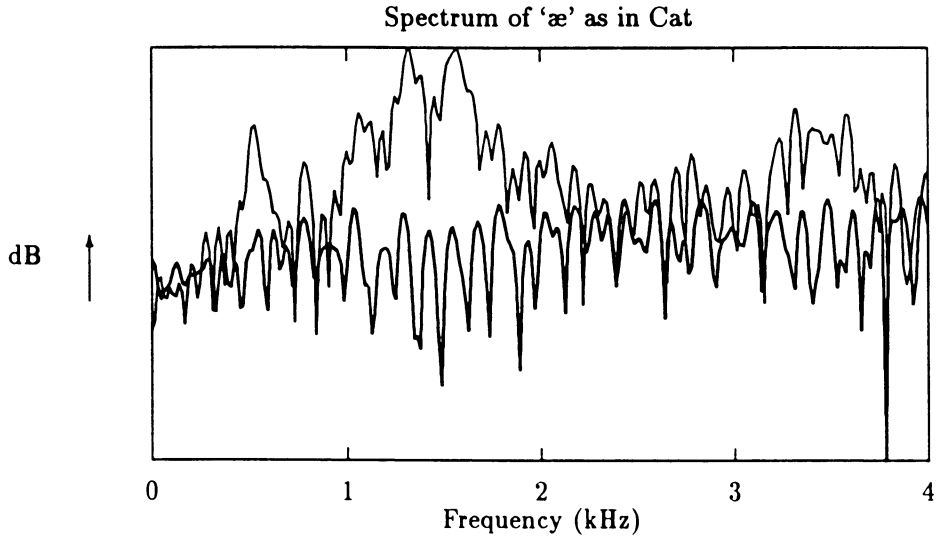


Figure 8.3: Spectrum of vowel /æ/ as in Cat without error shaping (L=2, 4-bit adaptive quantization of measurements, Optimum measurement vector, $\gamma = 1.0$)

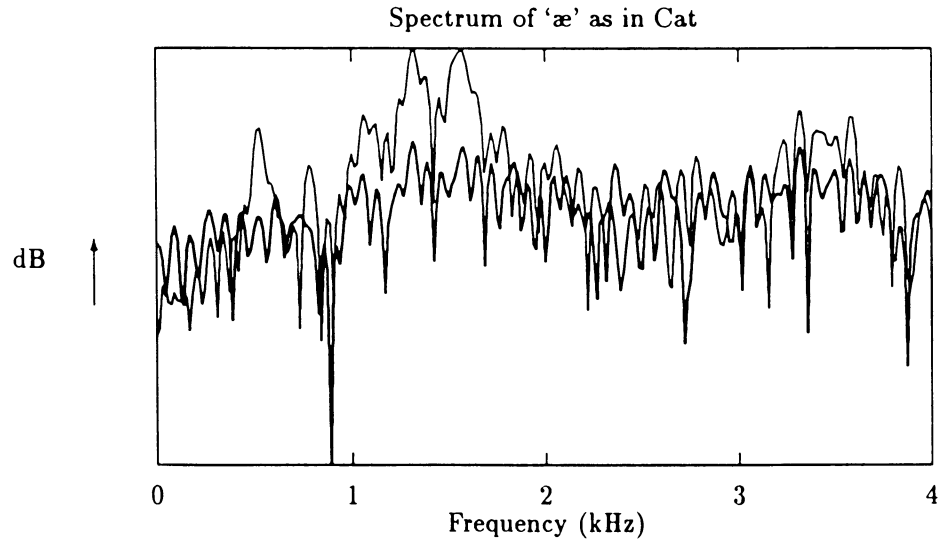


Figure 8.4: Spectrum of vowel /æ/ as in Cat with error shaping (L=2, 4-bit adaptive quantization of measurements, Optimum measurement vector, $\gamma = 0.73$)

Table 8.5: Subjective scores: Listener 1

Sen Id	7bit	6bit	5bit
F1 (Unweighted)	W	W*	W*
F1 (Weighted)	B	B	B*
F2 (Unweighted)	W	B	B*
F2 (Weighted)	W	W	E
F3 (Unweighted)	W	E	E
F3 (Weighted)	W	E	E
M1 (Unweighted)	W*	W*	W*
M1 (Weighted)	W*	E	E
M2 (Unweighted)	E	E	E
M2 (Weighted)	E	E	E
M3 (Unweighted)	W	W	W*
M3 (Weighted)	W	W	W*

Table 8.6: Subjective scores: Listener 2

Sen Id	7bit	6bit	5bit
F1 (Unweighted)	W	W	W
F1 (Weighted)	W	W	W
F2 (Unweighted)	W	W	W
F2 (Weighted)	W	W	W
F3 (Unweighted)	W	W	W
F3 (Weighted)	W	W	B
M1 (Unweighted)	W	E	B
M1 (Weighted)	W	W	W
M2 (Unweighted)	B	B	B
M2 (Weighted)	E	B	B
M3 (Unweighted)	W	W	W
M3 (Weighted)	W	W	W

Complexity of the Coder

The computational complexity of the coder as compared to the forward adaptive IALPC has been reduced in two steps.

(i) The complexity is reduced by using autocorrelation method instead of covariance method for model parameter estimation. The computation in autocorrelation method is proportional to p^2 whereas it is proportional to p^3 for the covariance method [21]. Moreover, with the use of a recursive window instead of the traditional Hamming window the number of computations required for the autocorrelation coefficient estimation is further reduced. In Hamming window method the number of multiplies needed is $3(pD - p/2)$ and for a recursive window it is $4n(p + 1) + 2p + 1$, where p is the order of the predictor, n is the frame length and D is the length of the window. In our implementation, p is 10, n is 10 and D is 300 samples. So we need 8985 multiplies if we use Hamming window technique and 461 multiplies if we use the recursive window which is an enormous reduction in complexity since we are updating the parameters every 10 samples. One added advantage in using a recursive window is that the number of memory locations needed is $6p+8$ rather than $n+3D/2$ for Hamming window. In actual our implementation it is 68 memory locations for recursive window technique and 460 for Hamming window technique.

(ii) The complexity is also reduced because of the reduced size of the Kalman state vector. The number of multiplies and adds needed by the Kalman estimator are $6.5n^2 + 9n$ and $6.5n^2 + 8n + 3$ respectively when an optimal measurement vector is used. So it is clear that the complexity is related to the size of the state vector (n) and the size of the error covariance matrices (nxn). Since n is of the order 128 [13] in the forward adaptive model and just 10 in the proposed backward adaptive coder

there is an enormous reduction is complexity. The complexity is further reduced by using the sub-optimal measurement vector. The optimal measurement vector needs n^2+n multiplies and n^2+n adds whereas the sub-optimal measurement vector needs only n multiplies. These reductions in computations make it feasible to implement the proposed coder in real-time.

Conclusions and Future Work

A backward adaptive innovations-assisted linear predictive coder was designed and implemented. By using a backward adaptive model the coding delay was reduced to be between (2.5-3.25ms) which is required in certain communication networks like the public switched networks (PSN). The computational complexity of the coder was also reduced a lot compared to the forward adaptive model and this makes it feasible to implement it real-time. The results are also quite encouraging to pursue further work on this type of coders. But because of the backward adaptation the prediction gain suffers and so the performance of the coder is as good as the forward adaptive model. The prediction gain does not increase with the increase in model order or with the inclusion of a pitch predictor. One way of increasing the quality could be by using a better measurement model. Instead of correcting the estimates only when the measurements are available, we could predict (in a backward fashion) what the measurement could be, by using a linear model, whenever a measurement is not available and use it to correct the estimates. This would increase the complexity of the coder but could probably improve the performance. A different adaptation technique could be used for the quantizer so that they adapt better to the sudden increase in magnitude of the innovations. This would help in eliminating the loss of

high frequency components.

There are several other issues open to investigation. The performance of the coder in the presence of channel errors could be studied. Since the complexity of the coder has been reduced, the algorithm could be implemented using DSP hardware and its performance in real-time could make an interesting study. Systolic array designs, which are better suited for hardware implementations, could be used for the Kalman filter.

BIBLIOGRAPHY

[1] T.W. Parsons, *Voice and Speech Processing*, McGraw Hill, New York, 1987.

[2] L. R. Rabiner, "On Creating Reference Templates for Speaker Independent Recognition of Isolated Words," *IEEE Trans. on ASSP.*, Vol. 26, pp. 34-42, Feb 1978.

[3] B. S. Atal and S. L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," *The Journal of the Acoustical Society of America*, pp. 637-655, Apr 1971.

[4] J. Makhoul, "Linear Prediction: A Tutorial Review," *Proceedings of the IEEE*, Vol. 63, pp. 561-580, Apr 1975.

[5] N. S. Jayant and P. Noll, *Digital Coding of Waveforms - Principles and Applications to Speech and Video*, Prentice-Hall, Englewood Cliffs, N.J., 1984.

[6] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

[7] B. S. Atal, "Predictive coding of speech at Low Bit Rates," *IEEE Trans. on Communications*, Vol. 30, pp. 600-614, Apr 1982.

[8] B. S. Atal and M. R. Schroeder, "Predictive Coding of Speech Signals and Subjective Error Criteria," *IEEE Trans. on ASSP.*, Vol. 27, pp. 247-254, Jun 1979.

[9] P. Strobach, *Linear Prediction Theory*, Springer - Verlag, New York, 1990.

[10] C. S. Williams, *Designing Digital Filters*, Prentice-Hall, Englewood Cliffs, N. J., 1986.

[11] T. P. Barnwell, "Recursive Windowing for Generating Autocorrelation Coefficients for LPC Analysis," *IEEE Trans. on ASSP*, Vol. 29, pp. 1062-1066, Oct 1981.

[12] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, N. J., 1991.

[13] T. V. Ramabadran and D. Sinha, "On the Selection of Measurements in Least-Squares Estimation," *Proc. IEEE Int. Conf. on Systems Eng.*, Dayton, Ohio, pp.221-226, Aug 1989.

[14] A. A. Giordano and F. M. Hsu, *Least Square Estimation with Applications to Digital Signal Processing*, John Wiley & Sons, New York, 1976.

[15] R. G. Brown, *Introduction to Random Signal Analysis and Kalman Filtering*, John Wiley & Sons, New York, 1983.

[16] T. V. Ramabadran and D. Sinha, "Speech Coding Using Least-Squares Estimation," *Advances in Speech Coding*, Eds. B. Atal, V. Cuperman, A. Gresho, Chapter 7.33, pp. 349-358, Kluwer Acadamic Publishers, Boston, 1991.

[17] L. R. Rabiner, M. J. Cheng, A. E. Rosenberg, C. A. McGonegal, "A Comparative Performance Study of Several Pitch Detection Algorithms," *IEEE Trans. on ASSP*, Vol. 24, Vol. 24, pp. 399-418, Oct 1976.

[18] R. P. Ramachandran and P. Kabal, "Pitch Prediction Filters in Speech Coding," *IEEE Trans. on ASSP*, Vol. 37, pp. 467-478, Apr 1989.

[19] R. Pettigrew and V. Cuperman, "Hybrid Backward Adaptive Pitch Prediction for Low-Delay Vector Excitation Coding," *Advances in Speech Coding*, Eds. B. Atal, V. Cuperman, A. Gresho, Chapter 1.6, pp. 57-66, Kluwer Acadamic Publishers, Boston, 1991.

[20] D. J. Goodman and A. Gersho, "Theory of an Adaptive Quantizer," *IEEE Trans. on Communications*, Vol. 22, pp. 1037-1045, Aug 1974.

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

[22] M. R. Schroeder and B. S. Atal, "Code -Excited Linear Prediction (CELP): High-Quality Speech at Very Low Bit Rates," *Proc. Int. Conf. on ASSP.*, pp. 937-940, April 1985.

[23] B. S. Atal and J. R. Remde, "A New Model of LPC Excitation for Producing Natural-Sounding Speech at Low Bit Rates," *Proc. Int. Conf. on Acoustics, Speech and Signal Processing.*, pp. 434-442, Oct 1977.

[24] A. Ichikawa, S. Takeda and Y. Asakawa, "A Speech Coding Method Using Thinned-Out Residual," *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 961-964, April 1985.

[25] N. Jayant, "High-Quality Coding of Telephone Speech and Wideband Audio," *Advances in Speech Signal Processing*, Eds. S. Furui and M. M. Sondhi, Chapter 1.3, pp. 85-108, M. Dekker Inc., New York, 1991.

[26] V. Cuperman, A. Gresho, R. Pettigrew, J. J. Shynk, J. Yao, "Backward Adaptive Configurations for Low-Delay Vector Excitation Coding," *Advances in Speech Coding*, Eds. B. Atal, V. Cuperman, A. Gresho, Chapter 2.2, pp. 13-23, Kluwer Acadamic Publishers, Boston, 1991.

[27] J. Chen, "A Robust Low-Delay CELP Speech Coder at 16 kb/s," *Advances in Speech Coding*, Eds. B. Atal, V. Cuperman, A. Gresho, Chapter 2.3, pp. 25-35, Kluwer Acadamic Publishers, Boston, 1991.

[28] M. W. Marcellin and T. R. Fischer, "A Trellis-Searched 16 kbit/sec Speech Coder With Low-Delay," *Advances in Speech Coding*, Eds. B. Atal, V. Cuperman, A. Gresho, Chapter 2.5, pp. 47-56, Kluwer Acadamic Publishers, Boston, 1991.

[29] V. Iyengar and P. Kabal, "A Low Delay 16kb/s Speech Coder," *IEEE Trans. on Signal Processing*, Vol. 39, pp. 1049-1057, May 1991.