Speech Coder using Line Spectral Frequencies of Cascaded Second Order Predictors

by

Visala Namburu

Thesis submitted to the Faculty of
The Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

APPROVED

Dr. A. A. (Louis) Beex, Chairman

Dr. Brian D. Woerner

Dr. William T. Baumann

November 2001
Blacksburg, VA

KEYWORDS: Speech Coding, Linear Prediction, Cascaded Second Order Predictors, Line Spectral Frequencies, Vector Quantization.

Speech Coder using Line Spectral Frequencies of Cascaded Second Order Predictors

by

Visala Namburu
Dr. A. A. (Louis) Beex, Chairman

The Bradley Department of Electrical and Computer Engineering

(ABSTRACT)

A major objective in speech coding is to represent speech with as few bits as possible. Usual transmission parameters include auto regressive parameters, pitch parameters, excitation signals and excitation gains. The pitch predictor makes these coders sensitive to channel errors. Aiming for robustness to channel errors, we do not use pitch prediction and compensate for its lack with a better representation of the excitation signal. We propose a new speech coding approach, Vector Sum Excited Cascaded Linear Prediction (VSECLP), based on code excited linear prediction.

We implement forward linear prediction using five cascaded second order sections - parameterized in terms of line spectral frequency - in place of the conventional tenth order filter. The line spectral frequency parameters estimated by the Direct Line Spectral Frequency (DLSF) adaptation algorithm are closer to the true values than those estimated by the Cascaded Recursive Least Squares - Subsection algorithm. A simplified version of DLSF is proposed to further reduce computational complexity.

Split vector quantization is used to quantize the line spectral frequency parameters and vector sum codebooks to quantize the excitation signals. The effect on reconstructed speech quality and transmission rate, of an increased number of bits and differently split combinations, is analyzed by testing VSECLP on the TIMIT database. The quantization of the excitation vectors using the discrete cosine transform resulted in segmental signal to noise ratio of 4 dB at 20.95 kbps, whereas the same quality was obtained at 9.6 kbps using vector sum codebooks.

Acknowledgements

I would like to thank Dr. A. A. (Louis) Beex, my academic and research advisor, for the guidance and support he provided throughout the duration of this project. His suggestions and directions helped me to improve my engineering and technical writing skills. I am also very grateful to Dr. Brian D. Woerner and Dr. William T. Baumann for serving on my committee and reviewing this work.

I am deeply indebted to my parents Tirumala Rao and Devi Tripura Sundari, without whose support I wouldn't have come here and studied in this wonderful University. I am also very thankful to my beloved husband Ramprasad and my family members for their continuous support and encouragement.

I would like to acknowledge my colleagues at the DSPRL, Takeshi Ikuma and Krishnaraj Varma for the help they provided throughout the course of this work.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AR	**AutoRegressive**
CELP	**Code-Excited Linear Prediction**
CRLS-SA	**Cascaded RLS with Subsection Adaptation**
DCT	**Discrete Cosine Transform**
FSVQ	**Full Search VQ**
LBG	**Linde-Buzo-Gray**
LP	**Linear Prediction**
LPC	**Linear Predictive Coding**
LSF	**Line Spectral Frequency**
LSF_2	**Line Spectral Frequency of a Second Order Section**
MLPC	**Multipulse LPC**
MSE	**Mean Squared Error**
MSVQ	**Multi – Stage VQ**
PEF	**Prediction Error Filter**
RELP	**Residual Excited LP**
RLS	**Recursive Least Squares**
SD	**Spectral Distortion**
SD_{rms}	**Root-Mean Square SD**
SD_{fw}	**Frequency Weighted SD**
SE	**Squared Error**
SGVQ	**Shape - Gain VQ**
SNR	**Signal-to-Noise Ratio**
SQ	**Scalar Quantization**
SSNR	**Segmental SNR**
SVQ	**Split Vector Quantization**
TSVQ	**Tree Search VQ**
VSELP	**Vector Sum Excited LP**
VQ	**Vector Quantization**
WMSE	**Weighted Mean Square Error**

Chapter 1 Introduction

The motivation for understanding the mechanism of speech production lies in the fact that speech is the human being's primary means of communication. Speech sounds are characterized in terms of the position and movement of the vocal-tract articulators, variation in their time waveform characteristics and frequency domain properties such as formant locations and bandwidth. The invention of the telephone by A. G. Bell brought a significant advance in human communication and opened a variety of research areas in speech [28].

Speech coding is a process in which the analog speech waveform is converted into digital form. A major objective in speech coding is to represent the digital representation of the speech signal with as few bits as possible. This operation is termed *speech compression*. Speech compression reduces the bit rate required to transmit digitized speech over a communication channel. The digitized speech can be stored in magnetic and optical media like tapes and magnetic discs. At the receiving end the digitized speech is converted back to the analog domain. Thus, the level of signal distortion introduced by the digital conversion process greatly affects the quality of reconstructed speech. The quality of reconstructed speech and the bit-rate are most important factors that determine the overall efficiency of speech coding. Implementation complexity and delay are also important factors to be considered in designing speech coders. Some speech coding applications are mobile satellite communications, cellular mobile radio, voice/data multiplexers for public and private networks, rural telephone, radio carrier system universal cordless telephones and interactive PC software. Advances in algorithmic techniques for speech coding continue to emerge as programmable signal processor chips and digital signal processing techniques make more efficient compression of data possible.

Speech coders are broadly divided into waveform coders and voice coders (vocoders). In waveform coding, the temporal and/or spectral characteristics of the speech signals are directly encoded. On the other hand, in voice coding, the speech signal is represented by an all-pole model of the vocal system. The AR parameters, also called LPC parameters are estimated from frames of speech, coded and transmitted over the channel. Vocoders achieve more bandwidth compression

than that achieved by waveform coders. Vocoders include *the cepstral (homomorphic) vocoder*, *the phase vocoder*, *the formant vocoder* and *the linear predictive vocoder*.

Our research concentrates on the linear predictive vocoders. The *Residual Excited Linear Prediction (RELP) Vocoder*, *Multipulse LPC (MLPC) Vocoder*, *Code-Excited Linear Predictive (CELP) Vocoder* are a few of the linear predictive vocoders. In RELP vocoders, the residual signal is generated by inverse filtering of the speech signal. The residual signal has a flat spectrum compared to the speech signal. Most of the important frequency components are present in the frequency range below 1000 Hz. Though it is ideal to transmit the entire residual signal to the decoder, this results in a high bit-rate. In order to reduce the bit-rate, the residual signal is low-pass filtered at cut-off frequency 1000 Hz. The output of the filter is decimated and transformed to the frequency domain via DFT. The magnitude and phase of the frequency components are coded and transmitted to the channel. At the receiver the high frequency components of the residual signal are generated by passing the signal through a full-wave rectifier and flattening the resulting spectrum by filtering. However, generating the high frequency components at the decoder results in a crude approximation of the high frequencies [28].

Atal and Remde proposed MLPC [29], an analysis-by-synthesis method that results in a better excitation method compared to that in RELP. In MLPC, the residual signal is passed through a perceptual weighting filter. The LPC filter is excited with multi-pulse excitation, which consists of a short sequence of pulses whose amplitude and locations are chosen to minimize the energy in the perceptually weighted residual signal. Singhal and Atal [30] observed that for voiced speech the MLPC shows a significant correlation from one pitch to the next and added a long-term correlation filter, also called pitch predictor, in cascade with the LPC filter. This approach further reduced the perceptually weighted error.

The CELP vocoder is also based on an analysis-by-synthesis method; here the excitation sequence is selected from a codebook of zero-mean Gaussian sequences [31]. CELP consists of a long-term pitch (LTP) filter in cascade with a short-term predictor (STP). The LTP is used to generate the pitch periodicity in voiced speech. The STP, an all pole filter typically with 10-12 coefficients, is used to generate the spectral envelope of the speech signal. The excitation sequence, selected from the codebook every 5ms, is used to excite the pitch filter in cascade with the LPC filter. The pitch parameters are updated every 5 ms whereas the LPC parameters are updated every 10-20 ms.

The inclusion of LTP makes the system very sensitive to channel errors. In a low-delay CELP implemented by Chen [6], the pitch predictor used in the conventional forward adaptive CELP is eliminated and instead a high order (around 50) backward LPC driven STP was included. The LPC

parameters are updated more frequently, typically every 2.5 ms, by performing LPC analysis on the previously quantized speech. The excitation vector is chosen from an excitation codebook every 0.625 ms. In this implementation, a logarithmic gain predictor is used to update the excitation gain along with a 10^{th} order perceptual weighting filter. Both the logarithmic gain predictor and the order of the perceptual weighting filter are updated once every 0.625 ms. With the high order backward LPC analysis, the coder becomes more robust to channel errors. No side information is required for high orders. By replacing the LTP with a high order STP, the algorithm becomes less speech specific. Hence, it is capable of handling non-voice signals. However, estimating the LPC parameters using a 50^{th} order filter requires a large number of correlation computations, resulting in high computational complexity. With high order filters, the chances of ill-conditioning are also increased [5].

The *Vector Sum Excited Linear Prediction (VSELP)* vocoder described by Gerson and Jasiuk [23] is another variation of the conventional CELP vocoder. The LPC coefficients represented by reflection coefficients are estimated every 20ms and are quantized using scalar quantization. In VSELP there are three excitation sources, one from the LTP filter and one from each of two *vector sum excitation (VSE)* codebooks. The excitation sequences are selected sequentially from each of the codebooks. Next, the excitation gains are optimized using a joint optimization technique and then vector quantized. The speech energy for every 20 ms segment is scalar quantized and transmitted.

In all the methods discussed so far, we see a variety of techniques used to improve the speech quality with different trade-off factors, for example, bit-rate and quality, bit-rate and delay, LTP and backward STP. This inspired us to come-up with a new speech coder from a different perspective.

In this thesis, we propose a new speech coder based on CELP. We term it Vector Sum Excited Cascade Linear Prediction (VSECLP) coding. In the proposed coder, in order to take advantage of not using LTP, we do not use a pitch predictor. We intend to compensate for the loss of LTP information with a better representation of the excitation signal. In the absence of LTP, the excitation signal consists of more speech information than would otherwise be the case. This requires a larger excitation codebook to represent the excitation signal efficiently. However, larger codebooks increase the optimization and search complexity. To overcome these problems, we use vector sum codebooks for the excitation signals.

Another important aspect of VSECLP is that we implement forward linear prediction using five cascaded 2^{nd} order LSF sections in place of the conventional 10^{th} order AR filter. The LSF parameters are more directly representative of the formant frequencies in speech and are more suitable for quantization, as compared to LPC coefficients. One of the methods to compute the LSF parameters

is to estimate the AR parameters first, then convert the AR parameters to LSF parameters. Several procedures, as implemented in the *auto-correlation, covariance,* and *recursive least squares* algorithms, can be used to provide estimates for the AR coefficients of the speech model. We consider the Recursive Least Squares (RLS) algorithm to estimate the AR coefficients. However, the RLS algorithm requires computation of the inverse of the auto-correlation matrix of the input data, resulting in computational complexity on the order of the square of the order of the filter [1]. The Cascaded RLS with Subsection Adaptation (CRLS-SA) algorithm [4] for adapting the AR filter coefficients associated with the cascaded model is one of the techniques, based on the RLS method, which significantly reduces the required computational effort relative to the RLS algorithm. The CRLS-SA takes advantage of the fact that, for inverse filtering applications, the gradients of each section in the cascade are almost uncorrelated with the gradients in other sections. In CRLS-SA the gradient auto-correlation matrix is assumed to be block diagonal, involving only 2 x 2 gradient auto-correlation matrices. This assumption reduces the computational complexity of RLS. The CRLS-SA can also be modified to adapt the 2nd order LSF parameters denoted by LSF_2 directly, and this method is termed Direct-LSF Subsection Adaptation (DLSF) using CRLS-SA [4]. We propose a simplified version of the DLSF algorithm that further reduces the computational complexity in DLSF.

The remainder of the thesis is organized as follows. Chapter 2 describes the RLS and CRLS-SA algorithms. Chapter 3 focuses on the DLSF algorithm. We show experimentally for an AR process that the estimated LSF_2 using DLSF are closer to the true LSF_2 than the LSF_2 obtained from the AR parameters estimated by CRLS-SA. Chapter 4 highlights SVQ and shape-gain VQ. Chapter 5 covers the quantization of LSF_2 using SVQ(2,4,4) and SVQ(2,2,2,4) codebooks. Both of these codebooks are compared in terms of the quality and bit-rate for the resulting speech coder based on the TIMIT database. The TIMIT corpus is exclusively designed to provide speech data for the acquisition of acoustic-phonetic knowledge. The quantization of excitation signals using the *discrete cosine transform* (DCT) and VSE are analyzed. Finally, the performance of the speech coder is evaluated on TIMIT speech data. Chapter 6 provides conclusions and suggestions for further research.

Chapter 2 Linear Prediction of Speech

In practical speech coding scenarios, speech is usually sampled at 8 kHz. Speech has at most four recognizable formant frequencies when it is sampled at 8 kHz. Thus, we attempt to model speech with an AR process of at least 8^{th} order. Most speech coding applications, use 10^{th} order AR filters. Speech is usually considered to be stationary within a 16 to 32 ms interval, leading to a corresponding analysis interval of 128 to 256 samples [5].

Linear Predictive Coding (LPC) is one of the analysis techniques commonly used in speech coding, as discussed in Section 2.1. Several procedures, as implemented in the *auto-correlation, covariance, recursive least squares* algorithms, can be used to provide estimates for the AR coefficients of the speech model. The Recursive Least Squares (RLS) based technique is used in the proposed speech coder. We describe the RLS algorithm in Section 2.2. However, the RLS algorithm requires computations of the inverse of the auto-correlation matrix of the input data, resulting in computational complexity on the order of the square of the order of the filter [1].

In speech the formants are widely separated in frequency, they occur in the neighborhood of [400 1000 1600 2400] Hz, and consequently the corresponding pairs of poles are dominant in only certain frequency bands. This nature of speech makes it feasible to model speech as if it were generated by cascaded AR sections of lower (here, second) order [4]. The Cascaded RLS with Subsection Adaptation (CRLS-SA) algorithm [4] for adapting the AR filter coefficients associated with the cascaded model is one of the techniques, based on the RLS method, which significantly reduces the required computational effort relative to the RLS algorithm as explained in Section 2.3. The CRLS-SA takes advantage of the fact that, for inverse filtering applications, the gradients of each section in the cascade are almost uncorrelated with the gradients in other sections. In CRLS-SA the gradient auto-correlation matrix is assumed to be block diagonal, involving only 2×2 gradient auto-correlation matrices. This assumption reduces the computational complexity of RLS.

2.1 Linear Prediction

We assume that the speech signal y_n is an N^{th} order AR process represented by

$$y_n = \sum_{k=1}^{N} a_{n,k} y_{n-k} + u_n \tag{2-1}$$

where u_n is the white noise excitation of the AR model, with its coefficient vector $\tilde{\mathbf{a}}_n$, given by

$$\begin{aligned} \tilde{\mathbf{a}}_n &= \begin{bmatrix} 1 & -a_{n,1} & -a_{n,2} & \dots & -a_{n,N} \end{bmatrix}^T \\ &= \begin{bmatrix} 1 & -\mathbf{a}_n^T \end{bmatrix}^T \end{aligned} \tag{2-2}$$

$$\mathbf{a}_n = \begin{bmatrix} a_{n,1} & a_{n,2} & \dots & a_{n,N} \end{bmatrix}^T \tag{2-3}$$

and y_n is the speech sample. The subscript *n*, in all the parameters represent time index. Let $\mathbf{y}_n$ be a speech input vector with past values given by

$$\mathbf{y}_n = \begin{bmatrix} y_{n-1} & y_{n-2} & \dots & y_{n-N} \end{bmatrix}^T \tag{2-4}$$

Linear Prediction (LP) is a method of predicting the unknown signal from its past. Say y_n is the output of an unknown system with some unknown input u_n. Thus, from (2-1) given the past outputs, the output y_n is estimated. Let $\hat{y}_n$, be the estimated value of y_n at time *n*, such that

$$\hat{y}_n = \sum_{k=1}^{N} \hat{a}_{n,k} y_{n-k} \tag{2-5}$$

where

$$\hat{\mathbf{a}}_n = \begin{bmatrix} \hat{a}_{n,1} & \hat{a}_{n,2} & \dots & \hat{a}_{n,N} \end{bmatrix}^T \tag{2-6}$$

are the estimated LPC coefficients defining $\hat{A}_n(z)$:

$$\hat{A}_n(z) = \sum_{k=1}^{N} \hat{a}_{n,k} z^{-k} \,. \tag{2-7}$$

Since y_n is estimated based on known knowledge at time (*n*-1), this operation is termed *forward linear prediction* and depicted in Figure 2-1. The *'forward prediction error f_e'* is the difference between the input sample y_n and its predicted value $\hat{y}_n$, and formulated as

$$f_e = e_n = y_n - \hat{y}_n \tag{2-8}$$

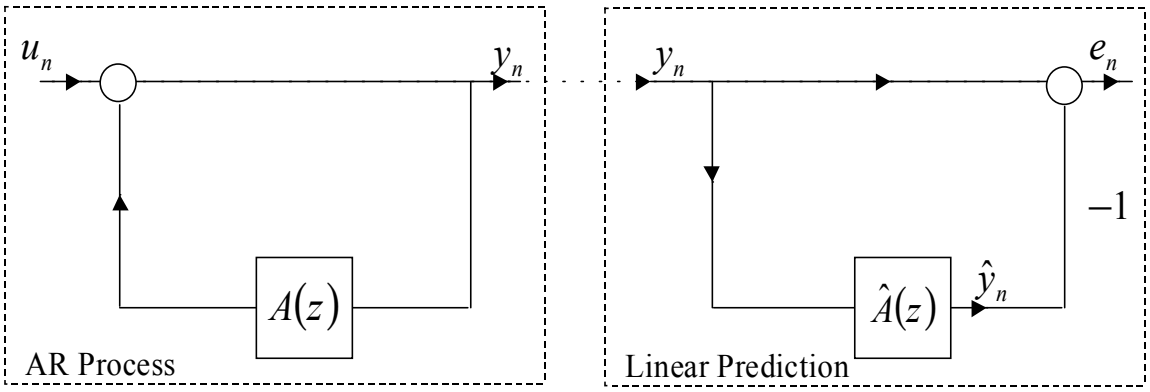

Figure 2-1 AR Process and Linear Prediction.

In other words, the linear prediction process finds an estimate of $A_n(z)$, $\hat{A}_n(z)$, by way of filtering y_n with the inverse filter $\left[1 - \hat{A}_n(z)\right]$ and minimizing the error according to some criterion. The operation of prediction error filtering applied to a stationary process $\{y_n\}$ is termed as *analysis* process. Hereby, we define an N^{th} prediction error filter, also termed analysis filter $H_a(z)$ as

$$H_a(z) = \left[1 - \hat{A}_n(z)\right] \tag{2-9}$$

where $\hat{A}_n(z)$ is as defined in (2-7). In theory, as the order of the prediction error filter increases, the correlation between input samples is reduced and eventually when the order becomes high enough, the output process consists of a sequence of uncorrelated samples [1]. A sequence of uncorrelated

random variables is called *white-noise*. A white-noise process denoted by $\{w_n\}$ has zero-mean and variance σ_w^2, such that

$$\mathrm{E}\left[w_i w_j^*\right] = \begin{cases} \sigma_w^2, & i = j \\ 0, & i \neq j \end{cases} \tag{2-10}$$

The process of converting a correlated input into white-noise output is termed as *whitening process*. When the output becomes white, the input process $\{y_n\}$ can be represented by the analysis filter coefficients and the prediction error power $P_M = \sigma_w^2$.

The auto-regressive modeling of the stationary process $\{y_n\}$, is termed as *synthesis* process. The analysis process and synthesis process are complementary to each other. In other words with white noise process $\{w_n\}$ of zero-mean and variance σ_w^2 at the input, an inverse analysis filter produces the stationary process $\{y_n\}$. The inverse analysis filter is termed as synthesis filter $H_s(z)$

$$H_s(z) = \left[1 - \hat{A}_n(z)\right]^{-1} \tag{2-11}$$

The analysis filter is an all-zero filter with an impulse response of finite duration. Conversely, the synthesis filter is an all-pole filter with an impulse response of infinite duration. The zeros of analysis filter lie inside the unit-circle and are located at exactly the same position as the poles of the synthesis filter. This ensures that the analysis filter and the inverse analysis filter or synthesis filter are both stable. Thus, the filters exhibit minimum-phase property.

In summary, given an AR speech model of order *N*, LPC analysis minimizes the residual error e_n, resulting in the filter $\left[1 - \hat{A}_n(z)\right]^{-1}$ given by

$$e_n = y_n - \left(\hat{a}_{n,1} y_{n-1} + \hat{a}_{n,2} y_{n-2} + \ldots + \hat{a}_{n,n} y_0\right) \tag{2-12}$$

where $\left[\hat{a}_{n,1} \quad \hat{a}_{n,2} \quad \ldots \quad \hat{a}_{n,N}\right]$ are the estimated LPC parameters.

2.1.1 Linear Prediction in Cascaded Structure

The RLS algorithm is one of the popular algorithms implemented to estimate the LPC parameters. However, the RLS requires computations of the inverse of the auto-correlation matrix of the input data. Using matrix inversion lemma, the direct computation of the inverse of the auto-correlation matrix can be eliminated, but the computational complexity is still on the order of the square of the order of the filter. This computational complexity can be reduced by implementing linear prediction in a cascaded structure.

Linear prediction in a cascaded structure was proposed by Jackson and Wood [2]. Here, the N^{th} order AR filter is represented by a cascade of M, 2^{nd} order sections. Each 2^{nd} order section results in a 2×2 auto-correlation matrix. The inverse of a 2×2 matrix can be readily and explicitly expressed in terms of its elements. Consequently the matrix inversion problem becomes simple. Moreover, it is easier to find root locations using a cascade structure than when using the Direct Form (DF) structure.

Forward linear prediction using a cascade of M 2^{nd} order sections is shown in Figure 2-2. From

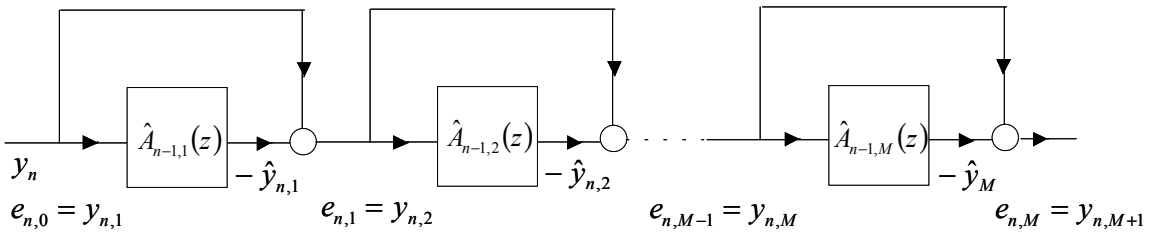

Figure 2-2 Linear Prediction in Cascade Structure.

Figure 2-2 the output of each section can be written as

$$\begin{aligned} e_{n,k} &= y_{n,k} - \hat{y}_{n,k} \\ &= y_{n,k} - \hat{\mathbf{a}}_{n,k}^T \mathbf{y}_{n,k} \\ &= y_{n,k} - \sum_{l=1}^{2} \hat{a}_{n,k,l} y_{n-l,k} \end{aligned} \tag{2-13}$$

where $y_{n,k}$, is the input to the k^{th} section with coefficients $\hat{\mathbf{a}}_{n,k}$ at time n

$$\hat{\mathbf{a}}_{n,k} = \begin{bmatrix} \hat{a}_{n,k,1} & \hat{a}_{n,k,2} \end{bmatrix}^T \tag{2-14}$$

defining $\hat{A}_{n,k}(z)$

$$\hat{A}_{n,k}(z) = \sum_{l=1}^{2} \hat{a}_{n,k,l} z^{-l} \tag{2-15}$$

Thus, the overall cascaded filter is represented by

$$\left(1 - \hat{A}_{n,1}(z)\right)\ \left(1 - \hat{A}_{n,2}(z)\right) \ldots \left(1 - \hat{A}_{n,M}(z)\right) = \prod_{k=1}^{M} \left[1 - \hat{A}_{n,k}(z)\right] \tag{2-16}$$

The cascaded analysis filter and synthesis filter are defined as

$$\begin{aligned} H_a(z) &= \prod_{k=1}^{M} \left[1 - \hat{A}_{n,k}(z)\right] \\ H_s(z) &= \left[H_a(z)\right]^{-1} = \left[\prod_{k=1}^{M} \left[1 - \hat{A}_{n,k}(z)\right]\right]^{-1} \end{aligned} \tag{2-17}$$

The prediction error of the k^{th} section $e_{n,k}$ is the input to the following section

$$y_{n,k+1} = e_{n,k} \tag{2-18}$$

and is given by

$$\begin{aligned} y_{n,k+1} &= e_{n,k} \\ &= \left[1 - \hat{A}_{n,k}(z)\right]\{y_{n,k}\} \\ &= y_{n,k} - \hat{y}_{n,k} \\ &= y_{n,k} - \hat{a}_{n,k,1} y_{n-1,k} - \hat{a}_{n,k,2} y_{n-2,k} \end{aligned} \tag{2-19}$$

The final prediction error $e_{n,M}$ is expressed as

$$
\begin{aligned}
e_{n,M} &= \left[\prod_{i=1}^{M}\left[1-\hat{A}_{n,i}(z)\right]\right]e_{n,0} \\
&= \left[\prod_{i=1}^{M}\left[1-\hat{A}_{n,i}(z)\right]\right]y_{n,1}
\end{aligned}
\tag{2-20}
$$

The gradient of each section is defined as the negative of the derivative of the final output error, defined in (2-20), with respect to the coefficients for that section. The individual gradients for the k^{th} section are therefore

$$
\begin{aligned}
\psi_{n,k,l} &= -\frac{\partial e_{n.M}}{\partial \hat{a}_{k.l}} = z^{-1}\left[\prod_{\substack{i=1 \\ i\neq k}}^{M}\left[1-\hat{A}_{n,i}(z)\right]\right]y_n \\
&= \frac{z^{-1}}{1-\hat{A}_{n,k}(z)}\left[\prod_{i=1}^{M}\left[1-\hat{A}_{n,i}(z)\right]\right]y_n \\
&= \frac{z^{-1}}{1-\hat{A}_{n,k}(z)}e_{n,M}
\end{aligned}
\tag{2-21}
$$

where $l = 1,2$. The cascaded linear prediction and the corresponding gradient computation are depicted in Figure 2-3.

2.2 The Recursive Least Squares (RLS) Algorithm

Given known initial conditions, the recursive least-squares method updates the old estimate of the model based on the available data, minimizing the cost function F_n [1], defined as

$$
F_n = \sum_{i=1}^{n}\lambda^{n-i}\left|e_i\right|^2; \qquad 0 << \lambda \leq 1 \tag{2-22}
$$

where λ is the forgetting factor, and e_i is the prediction error as defined in (2-8). The estimated AR coefficient vector $\hat{\mathbf{a}}_n$ and the corresponding AR filter are as defined in (2-6) and (2-7) respectively. Minimizing (2-22) results in the Yule-Walker or normal equations [3]:

$$
\mathbf{R}_n\hat{\mathbf{a}}_n = \mathbf{r}_{\mathbf{y}d_n} \tag{2-23}
$$

where $\mathbf{R}_n$ is the $N x N$ auto-correlation matrix of the input AR process y_n in (2-4), defined as

$$\mathbf{R}_n = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{y}_i \mathbf{y}_i^H \tag{2-24}$$

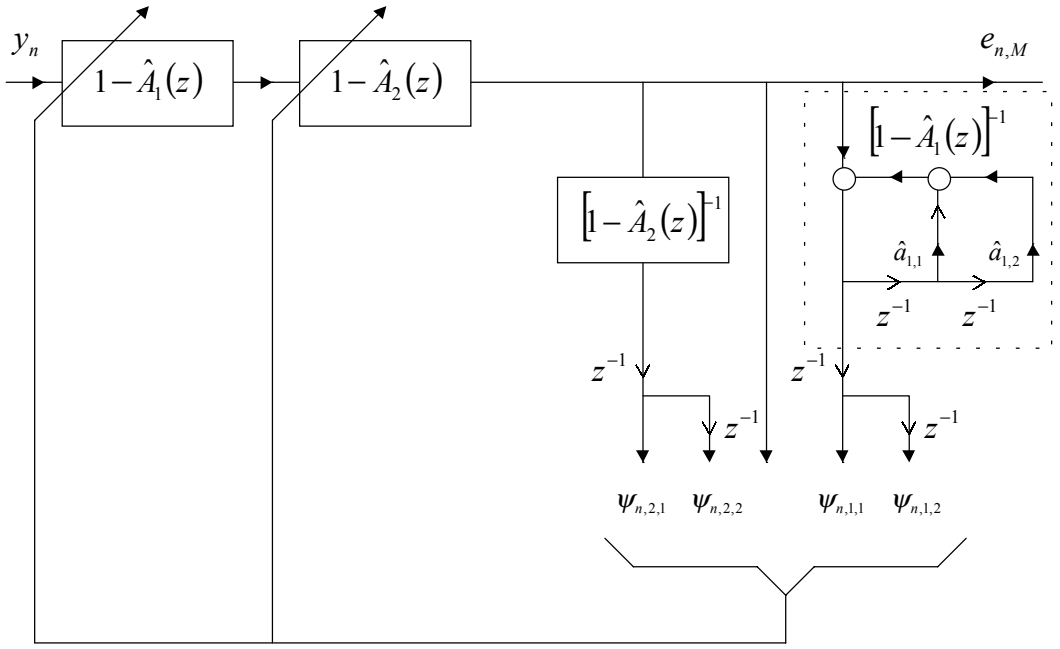

Figure 2-3 Cascade Linear Prediction and Gradient Computation.

The parameter $\mathbf{r}_{\mathbf{y}d_n}$ in (2-23) is the $M \times 1$ cross-correlation between the input signal y_n and the desired signal d_n, defined as

$$\mathbf{r}_{\mathbf{y}d_n} = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{y}_i d_i^* \tag{2-25}$$

For this application, the desired response d_n equals the input sample y_n.

$$d_n = y_n \tag{2-26}$$

Equation (2-24) can be recursively written as

$$\begin{aligned}\mathbf{R}_n &= \lambda\left[\sum_{i=1}^{n-1}\lambda^{n-i-1}\mathbf{y}_i\mathbf{y}_i^H\right] + \mathbf{y}_i\mathbf{y}_i^H \\ &= \lambda\mathbf{R}_{n-1} + \mathbf{y}_n\mathbf{y}_n^H\end{aligned} \tag{2-27}$$

where $\mathbf{R}_{n-1}$ is the correlation matrix at time (n-1). Thus, $\mathbf{R}_{n-1}$ is updated with the factor $\mathbf{y}_n\mathbf{y}_n^H$ at time n. Similarly, (2-25) can be written as follows

$$\mathbf{r}_{\mathbf{y}d_n} = \lambda\mathbf{r}_{\mathbf{y}d_{n-1}} + \mathbf{y}_n d_n^* \tag{2-28}$$

The filter coefficients $\hat{\mathbf{a}}_n$ are updated for each y_n. From (2-23) the updated coefficient estimates can be obtained as follows

$$\hat{\mathbf{a}}_n = \mathbf{R}_n^{-1}\mathbf{r}_{\mathbf{y}d_n} \tag{2-29}$$

By using the matrix inversion lemma, $\mathbf{R}_n^{-1}$ can be written as

$$\mathbf{R}_n^{-1} = \lambda^{-1}\mathbf{R}_{n-1}^{-1} - \frac{\lambda^{-2}\mathbf{R}_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^H\mathbf{R}_{n-1}^{-1}}{1+\lambda^{-1}\mathbf{y}_n^H\mathbf{R}_{n-1}^{-1}\mathbf{y}_n} \tag{2-30}$$

Let $\mathbf{P}_n$ be the $N\,x\,N$ inverse correlation matrix such that

$$\mathbf{P}_n = \mathbf{R}_n^{-1} \tag{2-31}$$

and $\boldsymbol{\kappa}_n$ be the $N\,x\,1$ gain vector defined as

$$\boldsymbol{\kappa}_n = \frac{\lambda^{-1}\mathbf{R}_{n-1}^{-1}\mathbf{y}_n}{1+\lambda^{-1}\mathbf{y}_n^H\mathbf{R}_{n-1}^{-1}\mathbf{y}_n} \tag{2-32}$$

Using these definitions (2-30) and (2-32) can be written as

$$\mathbf{P}_n = \lambda^{-1}\mathbf{P}_{n-1} - \lambda^{-1}\boldsymbol{\kappa}_n\mathbf{y}_n^H\mathbf{P}_{n-1} \tag{2-33}$$

$$\begin{aligned}\boldsymbol{\kappa}_n &= \lambda^{-1}\mathbf{P}_{n-1}\mathbf{y}_n - \lambda^{-1}\boldsymbol{\kappa}_n\mathbf{y}_n^H\mathbf{P}_{n-1}\mathbf{y}_n \\ &= \left[\lambda^{-1}\mathbf{P}_{n-1} - \lambda^{-1}\boldsymbol{\kappa}_n\mathbf{y}_n^H\mathbf{P}_{n-1}\right]\mathbf{y}_n\end{aligned} \tag{2-34}$$

The expression inside the brackets on the right-hand side of (2-34), equals $\mathbf{P}_n$ as defined in (2-33). Thus the gain vector $\boldsymbol{\kappa}_n$ in (2-34) simplifies to

$$\boldsymbol{\kappa}_n = \mathbf{P}_n\mathbf{y}_n \tag{2-35}$$

The coefficient vector $\hat{\mathbf{a}}_n$, from (2-29), is updated as follows, using (2-28), (2-31), so that

$$\begin{aligned}\hat{\mathbf{a}}_n &= \mathbf{R}_n^{-1}\mathbf{r}_{\mathbf{y}d\,n} \\ &= \mathbf{P}_n\mathbf{r}_{\mathbf{y}d\,n} \\ &= \lambda\mathbf{P}_n\mathbf{r}_{\mathbf{y}d\,n-1} + \mathbf{P}_n\mathbf{y}_n d_n^*\end{aligned} \tag{2-36}$$

Using (2-35) and substituting for $\mathbf{P}_n$ as defined in (2-33), in the first term on the right-hand side of (2-36) yields

$$\begin{aligned}\hat{\mathbf{a}}_n &= \mathbf{P}_{n-1}\mathbf{r}_{\mathbf{y}d\,n-1} - \boldsymbol{\kappa}_n\mathbf{y}_n^H\mathbf{P}_{n-1}\mathbf{r}_{\mathbf{y}d\,n-1} + \mathbf{P}_n\mathbf{y}_n d_n^* \\ &= \mathbf{R}_{n-1}^{-1}\mathbf{r}_{\mathbf{y}d\,n-1} - \boldsymbol{\kappa}_n\mathbf{y}_n^H\mathbf{R}_{n-1}^{-1}\mathbf{r}_{\mathbf{y}d\,n-1} + \mathbf{P}_n\mathbf{y}_n d_n^* \\ &= \hat{\mathbf{a}}_{n-1} - \boldsymbol{\kappa}_n\mathbf{y}_n^H\hat{\mathbf{a}}_{n-1} + \mathbf{P}_n\mathbf{y}_n d_n^* \\ &= \hat{\mathbf{a}}_{n-1} + \boldsymbol{\kappa}_n\left[d_n^* - \mathbf{y}_n^H\hat{\mathbf{a}}_{n-1}\right] \\ &= \hat{\mathbf{a}}_{n-1} + \boldsymbol{\kappa}_n\alpha_n^*\end{aligned} \tag{2-37}$$

where α_n^* is the a priori estimation error defined as

$$\alpha_n^* = d_n^* - \mathbf{y}_n^H\hat{\mathbf{a}}_{n-1} \tag{2-38}$$

Thus, the filter coefficients are updated recursively with each new sample of input data.

2.3 Cascade RLS – Subsection Adaptation Algorithm

As described in Section 2.1, the analysis process and synthesis process are complementary to each other. With a stationary process $\{y_n\}$ at the input, the output of the analysis filter tends to be white noise according to (2-1), provided the AR estimates converge to the true values. Correspondingly, the final output $e_{n,M}$ of the cascaded analysis filter (2-17) is white noise after convergence.

$$e_{n,M} = \prod_{i=1}^{M} \left[1 - \hat{A}_i(z)\right] y_n \tag{2-39}$$

From (2-39), we note that the gradient computation in (2-21) is the same as performing a whitening process except that the poles representing the k^{th} section are not whitened. Thus the gradient of the k^{th} section depends mainly on the poles of the k^{th} section, rendering the gradient correlation matrix $\hat{\mathbf{R}}_{\psi}$ of the form of a nearly block diagonal matrix [4]. Taking this concept into account, the Cascaded RLS – Subsection Adaptation Algorithm (CRLS-SA) [4] for adapting the AR filter coefficients based on RLS [1] explicitly assumes that the off-diagonal components of $\mathbf{R}_{\psi}$ are negligible. Therefore

$$\hat{\mathbf{R}}_{\psi} = diag\left\{\hat{\mathbf{R}}_{\psi\, n,1}, \quad \hat{\mathbf{R}}_{\psi\, n,2}, \quad \ldots \quad \hat{\mathbf{R}}_{\psi\, n,M}\right\} \tag{2-40}$$

where $\hat{\mathbf{R}}_{\psi\, n,k}$ is an estimate of the gradient auto-correlation matrix of the k^{th} section. As a result of assuming a block-diagonal nature for $\hat{\mathbf{R}}_{\psi}$, each section of the cascaded structure can be adapted independently of the other sections. From (2-21), the gradient of the final section is obtained as the input to that section. Thus, the gradient of the final section does not need to be computed separately. The CRLS-SA structure is shown in Figure 2-4.

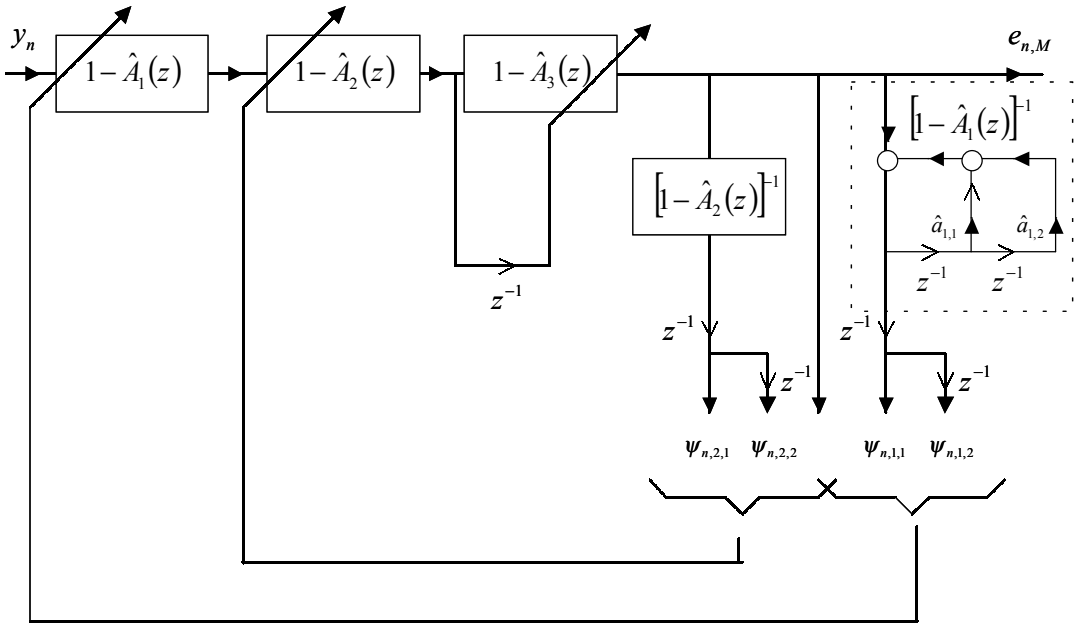

Figure 2-4 CRLS-SA and its Gradient Computation.

2.3.1 Description of CRLS-SA

The CRLS-SA adaptation is based on RLS. The cascaded filter consists of M 2nd order sections. Let

$$k = \{1 \quad 2 \quad \ldots \quad M\} \tag{2-41}$$

For each section k the inverse auto-correlation matrix, $\mathbf{P}_{n,k}$ is initialized with

$$\mathbf{P}_{0,k} = \delta^{-1}\mathbf{I} \tag{2-42}$$

where δ is a small positive constant, and $\mathbf{I}$ is the 2×2 identity matrix. Speech has most of its energy in the lower frequencies, as illustrated in Figure 2-5. Most of the spectral information is present in the lower formants. Since the exact energy associated with the formant frequencies is unknown, we distribute all the poles uniformly on the unit circle, and assign the lowest frequency pole pair (with the highest energy) to the first section and the highest frequency pole pair (with the lowest energy) to the last section in the cascaded filter.

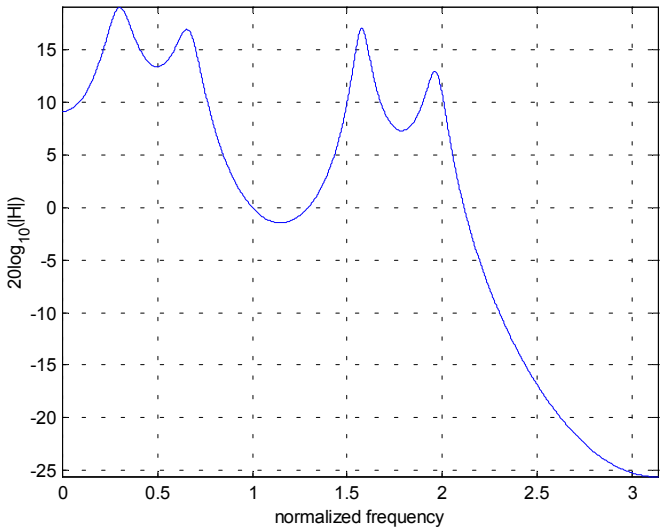

Figure 2-5 Frequency Response of AR model.

The output of the final section, $e_{n,M}$, is given by (2-20). The gradient of each section is computed from (2-21). The gain vector of each k^{th} section at time n is defined as

$$\boldsymbol{\kappa}_{n,k} = \frac{\mathbf{P}_{n-1,k}\boldsymbol{\psi}_{\mathbf{n,k}}}{\lambda + \boldsymbol{\psi}_{n,k}^{H}\mathbf{P}_{n-1,k}\boldsymbol{\psi}_{n,k}} \tag{2-43}$$

where $\boldsymbol{\psi}_{n,k}$, the gradient vector of the k^{th} section of order 2×1, is defined as

$$\boldsymbol{\psi}_{n,k} = \begin{bmatrix} \psi_{n,k,1} & \psi_{n,k,2} \end{bmatrix}^{T} \tag{2-44}$$

The coefficients of the k^{th} section are updated independently of other sections as defined in (2-37)

$$\hat{\mathbf{a}}_{n,k} = \hat{\mathbf{a}}_{n-1,k} + \boldsymbol{\kappa}_{n,k} e_{n,M} \tag{2-45}$$

The output of the k^{th} section forms the input to the $(k+1)^{\text{st}}$ section and is also the desired signal for the $(k+1)^{\text{st}}$ section.

$$e_{n,k} = d_{n,k} - \hat{\mathbf{a}}_{n-1,k}^{H}\mathbf{y}_{n,k} \tag{2-46}$$

$$d_{n+1,k} = e_{n,k} \tag{2-47}$$

Finally, from (2-33), $\mathbf{P}_{n,k}$ is updated according to

$$\mathbf{P}_{n,k} = \lambda^{-1}\left(\mathbf{P}_{n-1,k} - \boldsymbol{\kappa}_{n,k}\boldsymbol{\psi}_{n,k}^{H}\mathbf{P}_{n-1,k}\right) \tag{2-48}$$

The RLS algorithm does not guarantee a stable model. Several methods to check the stability of the estimated model exist in the literature [13]. Here, with each new sample of input data at time *n*, we check if the estimated model coefficient $\hat{a}_{n,k,2}$ satisfies [12]

$$\left|\hat{a}_{n,k,2}\right| < 1 \tag{2-49}$$

For stability, in addition, $\hat{a}_{n,k,1}$ must satisfy

$$\left|\hat{a}_{n,k,1}\right| < 1 - \hat{a}_{n,k,2} \tag{2-50}$$

If the estimated filter does not satisfy the above conditions, we first restrict $\hat{a}_{n,k,2}$ to

$$\hat{a}_{n,k,2} = 0.95\,\mathrm{sgn}\left(\hat{a}_{n,k,2}\right) \tag{2-51}$$

to satisfy (2-49) and then $\hat{a}_{n,k,1}$ is restricted to

$$\hat{a}_{n,k,1} = \mathrm{sgn}\left(\hat{a}_{n,k,1}\right)\left[1 - \hat{a}_{n,k,2}\right] \tag{2-52}$$

to ensure a stable model (applying the above repeatedly, if necessary). The CRLS-SA algorithm is summarized in Table 2-1.

Table 2-1 CRLS – SA Algorithm.

For $n = [1, \quad 2, \quad \ldots, \quad \ldots]$, with *n* being the sample index, compute

$$e_{n,M} = \left[\prod_{i=1}^{M}\left[1 - \hat{A}_{n-1,i}(z)\right]\right] e_{n,0} \tag{2-53a}$$

For $k = [1, \quad 2, \quad \ldots, \quad M]$, with *M* being the final section,

$$d_{n,k} = e_{n,k-1} \tag{2-53b}$$

$$\boldsymbol{\psi}_{n,k,l} = z^{-1}\left[1 - \hat{\mathbf{A}}_{n,k}(z)\right]^{-1}\{e_{n,M}\} \tag{2-53c}$$

$$\psi_{n,k,2} = \psi_{n-1,k,1} \tag{2-53d}$$

$$\boldsymbol{\psi}_{n,k} = \left[\psi_{n,k,1} \quad \psi_{n,k,2}\right]^T \tag{2-53e}$$

$$\boldsymbol{\pi}_{n,k} = \boldsymbol{\psi}_{n,k}^H \mathbf{P}_{n-1,k} \tag{2-53f}$$

$$\boldsymbol{\kappa}_{n,k} = \frac{\mathbf{P}_{n-1,k}\boldsymbol{\psi}_{\mathbf{n,k}}}{\lambda + \boldsymbol{\pi}_{n,k}\boldsymbol{\psi}_{n,k}} \tag{2-53g}$$

$$e_{n,k} = d_{n,k} - \hat{\mathbf{a}}_{n-1,k}^H \boldsymbol{\psi}_{n,k} \tag{2-53h}$$

$$\hat{\mathbf{a}}_{n,k} = \hat{\mathbf{a}}_{n-1,k} + \boldsymbol{\kappa}_{n,k} e_{n,M} \tag{2-53i}$$

$$\mathbf{P}_{n,k} = \frac{1}{\lambda}\left(\mathbf{P}_{n-1,k} - \boldsymbol{\kappa}_{n,k}\boldsymbol{\pi}_{n,k}\right) \tag{2-53j}$$

Chapter 3 Line Spectral Frequencies

After the LPC parameters are estimated, say for a frame of speech, they are to be coded and transmitted to the communication channel. The LPC are characterized by large dynamic range and would require 8-10 bits per coefficient for accurate representation [28]. Relatively small changes in the representation of the LPC parameters result in a large change in the pole locations of the filter model. The LPC parameters donot easily recognized correspondence to stable filters. The latter can lead to filter instability problems upon receiving LPC parameters corrupted during transmission. For these reasons, the LPC are rarely used directly in the communication channel. Instead, an equivalent representation of the LPC parameters with better quantization and interpolation properties is used. Some of the alternate representations of the LPC parameters that represent the same spectral information, are reflection coefficients, log area ratios and line spectral frequencies. Among all these representations, we choose the Line-Spectral Frequency (LSF) representation of the LPC parameters. The LSF exhibit ordering and distortion independence properties as described in Section 3.1. These properties enable us to represent the high frequencies associated with less energy with fewer bits.

One of the methods to compute the LSF parameters is to estimate the LPC parameters using CRLS-SA as described in Section 2.3, then to convert the estimated LPC parameters to LSF parameters. The LPC to LSF transformation is described in Section 3.1. The LSF parameters can be adapted directly using the Direct-LSF (DLSF) algorithm, based on CRLS-SA algorithm. The direct LSF adaptation and Direct LSF synthesis are described in Section 3.2 and Section 3.3 respectively. The CRLS-SA and DLSF estimation techniques are compared in Section 3.4 for a given AR_{10} process. The results show that the LSF parameters obtained with DLSF are closer to the true values than the LSF parameters obtained with CRLS-SA.

3.1 LPC to LSF Transformation

Let the analysis filter $H_a(z)$ as defined in (2-17) be a stable and even ordered filter [5]. Here onwards, the subscript n representing time is dropped for simplicity. The $H_a(z)$ filter can be represented with an even symmetric and an odd symmetric filter such that

$$H_a(z) = 1 - \hat{A}(z) = \frac{1}{2}[P(z) + Q(z)] \tag{3-1}$$

where $P(z)$ and $Q(z)$ are the even and odd symmetric filters respectively, defined as

$$\begin{aligned} P(z) &= H_a(z) - z^{-(N+1)} H_a(z^{-1}) \\ &= 1 - (\hat{a}_1 + \hat{a}_N) z^{-1} - \ldots \quad - (\hat{a}_N + \hat{a}_1) z^{-N} + z^{-(N+1)} \end{aligned} \tag{3-2}$$

$$\begin{aligned} Q(z) &= H_a(z) + z^{-(N+1)} H_a(z^{-1}) \\ &= 1 - (\hat{a}_1 - \hat{a}_N) z^{-1} - \ldots \quad - (\hat{a}_N - \hat{a}_1) z^{-N} - z^{-(N+1)} \end{aligned} \tag{3-3}$$

$P(z)$ and $Q(z)$ have one root at $z = -1$ and $z = 1$ respectively, thus $P(z)$ and $Q(z)$ can be written as that

$$\begin{aligned} P'(z) &= \frac{P(z)}{(1+z)} \\ &= p_0 z^N + p_1 z^{(N-1)} + \ldots + p_k z^{(N-k)} + \ldots + p_N \end{aligned} \tag{3-4}$$

$$\begin{aligned} Q'(z) &= \frac{Q(z)}{(1-z)} \\ &= q_0 z^N + q_1 z^{(N-1)} + \ldots + q_k z^{(N-k)} + \ldots + q_N \end{aligned} \tag{3-5}$$

where

$$\begin{aligned} p_0 &= 1 \\ q_0 &= 1 \\ p_k &= -(\hat{a}_k + \hat{a}_{N+1-k}) - p_{k-1}, \; for \quad k = 1, \quad \ldots, \quad N \\ q_k &= (-\hat{a}_k + \hat{a}_{N+1-k}) + q_{k-1}, \; for \quad k = 1, \quad \ldots, \quad N \end{aligned} \tag{3-6}$$

Substituting (3-6) in (3-4) and (3-5) the polynomials $P'(z)$ and $Q'(z)$ turn out to be symmetric such that

$$P'(z) = p_0 z^N + p_1 z^{(N-1)} + \ldots + p_k z^{(N-k)} + \ldots + p_1 z + p_0$$
$$Q'(z) = q_0 z^N + q_1 z^{(N-1)} + \ldots + q_k z^{(N-k)} + \ldots + q_1 z + q_0 \tag{3-7}$$

Using the symmetry property, $P'(z)$ and $Q'(z)$ can be expressed as a set of cosine functions as follows:

$$\begin{aligned} P'(z) &= p_0 z^N + p_1 z^{(N-1)} + \ldots + p_k z^{(N-k)} + \ldots p_1 z + p_0 \\ &= 2z^{N/2}\left[p_o\left(\frac{z^{N/2} + z^{-N/2}}{2} \right) + p_1\left(\frac{z^{(N-2)/2} + z^{-(N-2)/2}}{2} \right) + \quad \ldots \quad + \quad \frac{1}{2} p_{N/2} \right] \end{aligned} \tag{3-8}$$

Substituting $z = e^{j\omega}$ in (3-8)

$$\begin{aligned} P'(\omega) &= 2e^{j\omega N/2}\left[p_0\left(\frac{e^{j\omega N/2} + e^{-j\omega N/2}}{2} \right) + p_1\left(\frac{e^{j\omega(N-2)/2} + e^{-j\omega(N-2)/2}}{2} \right) + \ldots \quad + \frac{1}{2} p_{N/2} \right] \\ &= 2e^{j\omega N/2}\left[p_0 \cos(\omega N/2) \quad + p_1 \cos(\omega(N-2)/2) \quad + \ldots \quad + \frac{1}{2} p_{N/2} \right] \end{aligned} \tag{3-9}$$

Similarly,

$$Q'(\omega) = 2e^{j\omega N/2}\left[q_o \cos(\omega N/2) \quad + q_1 \cos(\omega(N-2)/2) + \quad \ldots \quad + \quad \frac{1}{2} q_{N/2} \right] \tag{3-10}$$

Thus, the polynomials $P'(\omega)$ and $Q'(\omega)$ can be simplified to a polynomial of order $N/2$ along with a $2e^{j\omega N/2}$ factor. For the k^{th} second order section (3-4) and (3-5) reduce to

$$P'_k(z) = 1 + p_k z^{-1} + z^{-2}$$
$$Q'_k(z) = 1 + q_k z^{-1} + z^{-2} \tag{3-11}$$

where $P'_k(z)$ and $Q'_k(z)$ are the reduced 2nd order odd and even symmetric filters for the k^{th} section and $[p_k \quad q_k]$ are given by

$$\begin{aligned} p_k &= -\left(1 + \hat{a}_{k,1} + \hat{a}_{k,2}\right) \\ q_k &= \left(1 - \hat{a}_{k,1} + \hat{a}_{k,2}\right) \end{aligned} \tag{3-12}$$

We refer to LSF parameters of 2nd order section by LSF_2 and the corresponding $[p \quad q]$ parameters by LSF_2 parameters. A second order filter with complex conjugate poles can be represented by

$$B(z) = 1 - 2r\cos\theta \; z^{-1} + r^2 z^{-2} \tag{3-13}$$

where r is the radius and θ the angular position of the upper half plane (UHP) pole. Comparing even and odd symmetric filters as defined in (3-11) to (3-13),

$$r = 1 \tag{3-14}$$

and the LSF_2 of $P'_k(z)$ and $Q'_k(z)$ are

$$\begin{aligned} \theta_{p,k} &= \cos^{-1}\left(-p_k \big/ 2\right) \\ \theta_{q,k} &= \cos^{-1}\left(-q_k \big/ 2\right) \end{aligned} \tag{3-15}$$

respectively. In other words, the LSF_2 of $P'_k(z)$ and $Q'_k(z)$ lie on the unit circle. From (3-15), the LSF_2 can be directly obtained from p_k and q_k without further computation, e.g. by table-lookup. From (3-12), the LPC parameters can be written in terms of LSF_2 parameters as

$$\begin{aligned} \hat{a}_{k,1} &= -\frac{p_k + q_k}{2} \\ \hat{a}_{k,2} &= -1 - \frac{p_k - q_k}{2} \end{aligned} \tag{3-16}$$

Comparing an analysis filter $H_a(z)$ as defined in (2-17) of order 2 with (3-13), the frequency of the pole ω_k is given by

$$\omega_k = \cos^{-1}\left(\frac{\hat{a}_{k,1}}{2\sqrt{-\hat{a}_{k,2}}}\right)$$
$$r_k^2 = -\hat{a}_{k,2} \tag{3-17}$$

Let $z_k = re^{j\omega_k}$ be the pole of the k^{th} section, then from p_k and q_k the formant frequency f_k and its 3 dB bandwidth BW_3 [20] are

$$f_k = \frac{\omega_k}{2\pi}$$
$$BW = \log_e\left(\frac{r}{\pi}\right) \tag{3-18}$$
$$BW_3 = \frac{-BW}{2\pi}$$

From (3-16), substituting for $\lfloor \hat{a}_{k,1} \quad \hat{a}_{k,2} \rfloor$ in (3-17) and (3-18), f_k and its 3 dB bandwidth BW_3 can be directly computed from LSF_2 parameters.

The LSF also exhibit *ordering* and *interlacing properties* i.e. the roots of $P'(z)$ and $Q'(z)$ alternate on the unit circle: $0 \le \theta_{p,1} < \theta_{q,1} < \theta_{p,2} < \theta_{q,2} \ldots \le \pi$. By the method of contradiction the ordering property of LSF_2 for a k^{th} second order section is proved as follows. Assume that

$$\theta_{q,k} \le \theta_{p,k} \tag{3-19}$$

Using (3-19), and substituting (3-12) in (3-15) yields

$$\begin{gathered}
\cos^{-1}\left(-q_k/2\right) \le \cos^{-1}\left(-p_k/2\right) \\
\cos^{-1}\left(-\frac{\left(1-\hat{a}_{k,1}+\hat{a}_{k,2}\right)}{2}\right) \le \cos^{-1}\left(\frac{\left(1+\hat{a}_{k,1}+\hat{a}_{k,2}\right)}{2}\right) \\
\left(-\frac{\left(1-\hat{a}_{k,1}+\hat{a}_{k,2}\right)}{2}\right) \ge \left(\frac{\left(1+\hat{a}_{k,1}+\hat{a}_{k,2}\right)}{2}\right) \\
2 \le -2\hat{a}_{k,2}
\end{gathered} \tag{3-20}$$

Consequently,

$$1 \le -\hat{a}_{k,2} \tag{3-21}$$

which contradicts the stability conditions in (2-49). So the assumption made in (3-19) is not correct, conversely,

$$\theta_{p,k} < \theta_{q,k} \tag{3-22}$$

Using (3-15), the inequality in (3-22) can be written in terms of LSF_2 parameters as

$$-\frac{p_k}{2} > -\frac{q_k}{2} \tag{3-23}$$

Similarly, for higher orders, the LSF also follow an ordering property, with the LSF of the even symmetric filter $P'(z)$ defined in (3-4) interlaced with the LSF of the odd symmetric filter $Q'(z)$ defined (3-5).

Summarizing LSF Properties

The LSF θ_i, i.e. the angular positions of the roots of $P'(z)$ and $Q'(z)$, satisfy the following properties:

1. Using the symmetric property $P'(z)$ and $Q'(z)$ are simplified to a polynomial of order $N/2$ along with a $2e^{j\omega N/2}$ factor as defined in (3-9) and (3-10) respectively.
2. All the roots of $P'(z)$ and $Q'(z)$ lie on the unit circle.
3. The LSF parameters exhibit an ordering and interlacing property, i.e. the roots of $P'(z)$ and $Q'(z)$ alternate on the unit circle: $0 \le \theta_{p,1} < \theta_{q,1} < \theta_{p,2} < \theta_{q,2} \ldots \le \pi$.
4. The LSF have inter-frame correlation $\xi = \{\phi_{i,k}\}$ and intra-frame correlation $\Omega = \{\phi_{i,j}\}$ properties [5] defined as

$$\phi_{i,k} = \theta_{n,i} \times \theta_{n-k,i}, \quad i = 1,2,\ldots,N, \qquad k = 1,2,\ldots,N \tag{3-24}$$

$$\phi_{i,j} = \theta_{n,i} \times \theta_{n,j}, \quad i = 1,2,\ldots,N, \qquad j = 1,2,\ldots,N \tag{3-25}$$

where θ_n is the LSF vector defined as

$$\theta_n = \begin{bmatrix} \theta_{n,p,1} & \theta_{n,q,1} & \cdots & \theta_{n,q,N-1} & \theta_{n,p,N} & \theta_{n,q,N} \end{bmatrix} \tag{3-26}$$

The LSF parameters also exhibit the distortion independence property [15] [16]. The distortion independence property means that any change in an LSF parameter will not produce effects globally, it will only affect the frequency spectrum close to it. Using this property, the LSF parameters at higher frequencies can be represented with fewer bits, as the human ear is not very sensitive to higher frequencies. Consequently, the bit-rate can be reduced.

The ordering property [5] of the LSF parameters are used in error control applications to detect and partially correct some transmission errors. For example, when a data packet is lost during transmission, making use of the LSF properties, the data in the lost packet can be estimated by interpolation from knowledge of the previous and succeeding data packets. Consequently, the use of LSF parameters has been found to be more practical than using LPC coefficients.

3.2 Direct LSF_2 Adaptation

Substituting (3-4) and (3-5) in (3-1), each second order section in the cascaded structure can be represented by the corresponding even symmetric filter $P'(z)$ and odd symmetric filter $Q'(z)$ as

$$1 - \hat{A}_k(z) = \frac{1}{2}\left[\left(1 + z^{-1}\right)P'_k(z) + \left(1 - z^{-1}\right)Q'_k(z)\right] \tag{3-27}$$

Using (2-16), the cascaded filter can thus be represented as follows

$$1 - \hat{A}(z) = \prod_{k=1}^{M}\left\{\frac{1}{2}\left[\left(1 + z^{-1}\right)P'_k(z) + \left(1 - z^{-1}\right)Q'_k(z)\right]\right\} \tag{3-28}$$

The direct LSF_2 analysis and adaptation for the k^{th} section is illustrated in Figure 3-1.

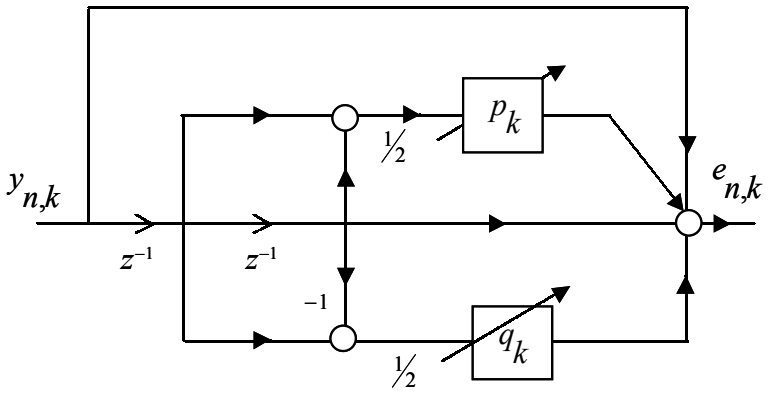

Figure 3-1 Direct LSF_2 Analysis and Adaptation Section $\left(1-\hat{A}_k(z)\right)$.

3.3 Direct LSF_2 Synthesis

The LSF_2 can be used directly for synthesis as follows.

$$Y_k(z)=\frac{E_k(z)}{1-\hat{A}_k(z)} \tag{3-29}$$

Substituting (3-1) in denominator of (3-29), we have

$$\begin{aligned} Y_k(z) &= \frac{E_k(z)}{1-\hat{A}_k(z)} \\ &= \frac{E_k(z)}{0.5\left(P_k(z)+Q_k(z)\right)} \\ &= \frac{2E_k(z)}{\left(1+z^{-1}\right)\left(1+p_k z^{-1}+z^{-2}\right)+\left(1-z^{-1}\right)\left(1+q_k z^{-1}+z^{-2}\right)} \end{aligned} \tag{3-30}$$

Multiplying out and taking inverse z-transforms, we get

$$\begin{aligned} y_k &= 0.5\left\{2e_{n,k}-y_{n-1,k}\left(p_k+q_k\right)-y_{n-2,k}\left(2+p_k-q_k\right)\right\} \\ &= e_{n,k}-0.5\,y_{n-1,k}\left(p_k+q_k\right)-0.5\,y_{n-2,k}\left(p_k-q_k\right)-y_{n-2,k} \end{aligned} \tag{3-31}$$

From (3-31), the LSF_2 parameters can be used directly for synthesis. The gradients $\varphi_{n,k,p}$ and $\varphi_{n,k,q}$ for adapting $P'_k(z)$ and $Q'_k(z)$, as illustrated in Figure 3-2, are

$$\begin{aligned}
\varphi_{n,k,p} &= -\frac{\partial e_{n,M}}{\partial p_k} \\
&= -0.5\left(1+z^{-1}\right)z^{-1}\prod_{\substack{i=1\\ i\neq k}}^{M}\left\{0.5\left[\left(1+z^{-1}\right)P'_i(z)+\left(1-z^{-1}\right)Q'_i(z)\right]\right\}y_n \\
&= \frac{-0.5\left(1+z^{-1}\right)z^{-1}}{0.5\left[\left(1+z^{-1}\right)P'_k(z)+\left(1-z^{-1}\right)Q'_k(z)\right]}\prod_{i=1}^{M}\left\{0.5\left[\left(1+z^{-1}\right)P'_i(z)+\left(1-z^{-1}\right)Q'_i(z)\right]\right\}y_n \\
&= -\left(1+z^{-1}\right)z^{-1}\frac{e_{n,M}}{2\left(1-\hat{A}_k(z)\right)}
\end{aligned} \tag{3-32}$$

similarly,

$$\varphi_{n,k,q} = -\left(1-z^{-1}\right)z^{-1}\frac{e_{n,M}}{2\left(1-\hat{A}_k(z)\right)} \tag{3-33}$$

From (3-31), we note that the LSF_2 parameters can be adapted with the scaling factor –0.5. We therefore define the gradient vector $\varphi_{n,k}$ and LSF_2 parameter vector $\mathbf{f}_{n,k}$ as

$$\varphi_{n,k} = \begin{bmatrix}\varphi_{n,k,p} & \varphi_{n,k,q}\end{bmatrix}^T \tag{3-34}$$

$$\mathbf{f}_{n,k} = \begin{bmatrix}-0.5p_{n,k} & -0.5q_{n,k}\end{bmatrix}^T \tag{3-35}$$

Let us define

$$\begin{aligned}
\tilde{p}_{n,k} &= -0.5p_{n,k} \\
\tilde{q}_{n,k} &= -0.5q_{n,k}
\end{aligned} \tag{3-36}$$

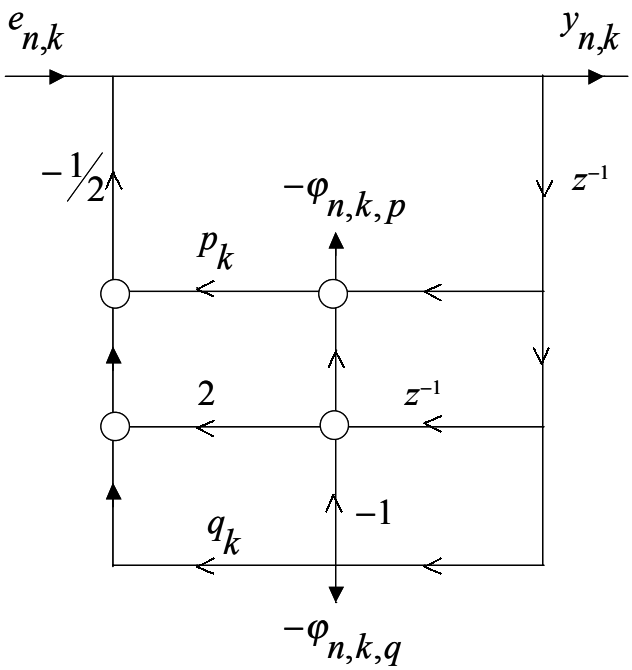

Figure 3-2 Direct LSF$_2$ Synthesis Section $\left(1-\hat{A}_k(z)\right)$, using $\left[p_k \quad q_k\right]$ Parameters.

and consequently the $\mathbf{f}_{n,k}$ redefined as

$$\mathbf{f}_{n,k} = \left[\tilde{p}_{n,k} \quad \tilde{q}_{n,k}\right]^T \tag{3-37}$$

Rewriting (3-12) in terms of $\mathbf{f}_{n,k}$, we have

$$\begin{aligned} \tilde{p}_k &= 0.5\left(1+\hat{a}_{k,1}+\hat{a}_{k,2}\right) \\ \tilde{q}_k &= -0.5\left(1-\hat{a}_{k,1}+\hat{a}_{k,2}\right) \end{aligned} \tag{3-38}$$

Consequently (3-16) and (3-27) are equivalent to

$$\begin{aligned} \hat{a}_{k,1} &= \tilde{p}_k + \tilde{q}_k \\ \hat{a}_{k,2} &= -1 + \tilde{p}_k - \tilde{q}_k \end{aligned} \tag{3-39}$$

$$\begin{aligned} 1-\hat{A}_k(z) &= 1-(\tilde{p}_k+\tilde{q}_k)z^{-1}-(-1+\tilde{p}_k-\tilde{q}_k)z^{-2} \\ &= 1-\hat{F}_k(z) \end{aligned} \tag{3-40}$$

The location of poles $[r_{k1} \quad r_{k2}]$ of (3-40)

$$[r_{k1} \quad r_{k2}] = \frac{(\tilde{p}_k+\tilde{q}_k) \pm \sqrt{(\tilde{p}_k+\tilde{q}_k)^2+4(1+\tilde{p}_k-\tilde{q}_k)}}{2} \tag{3-41}$$

Using (3-39), rewriting (3-17) in terms of $[\tilde{p}_k \quad \tilde{q}_k]$

$$\begin{aligned} \omega_k &= \cos^{-1}\left(\frac{\tilde{p}_k+\tilde{q}_k}{2\sqrt{1-\tilde{p}_k+\tilde{q}_k}}\right) \\ r_k^2 &= 1-\tilde{p}_k+\tilde{q}_k \end{aligned} \tag{3-42}$$

Substituting (3-42) in (3-18), the f_k and its 3-dB bandwidth BW_3 are redefined as

$$\begin{aligned} f_k &= \frac{\omega_k}{2\pi} \\ &= \frac{1}{2\pi}\cos^{-1}\left(\frac{\hat{a}_{k,1}}{2\sqrt{-\hat{a}_{k,2}}}\right) \\ &= \frac{1}{2\pi}\cos^{-1}\left(\frac{\tilde{p}_k+\tilde{q}_k}{2\sqrt{1-\tilde{p}_k+\tilde{q}_k}}\right) \end{aligned} \tag{3-43}$$

$$\begin{aligned} BW &= \log_e\left(\frac{r}{\pi}\right) \\ &= \log_e\left(\frac{\sqrt{-\hat{a}_{k,2}}}{\pi}\right) \\ &= \log_e\left(\frac{\sqrt{(1-\tilde{p}_k+\tilde{q}_k)}}{\pi}\right) \\ BW_3 &= \frac{-BW}{2\pi} \end{aligned} \tag{3-44}$$

The close relation between LSF_2 parameters and formant frequencies, in (3-43), not only reduces the computational complexity, it also enables the LSF_2 parameters to converge faster to the true values as compared to those obtained from CRLS-SA, as will be explained and illustrated in Section 3.4. The direct LSF_2 synthesis section illustrated in Figure 3-2 is modified to adapt the LSF_2 parameter vector $\mathbf{f}_{n,k}$ in (3-37), as depicted in Figure 3-3. The DLSF algorithm is summarized in Table 3-1.

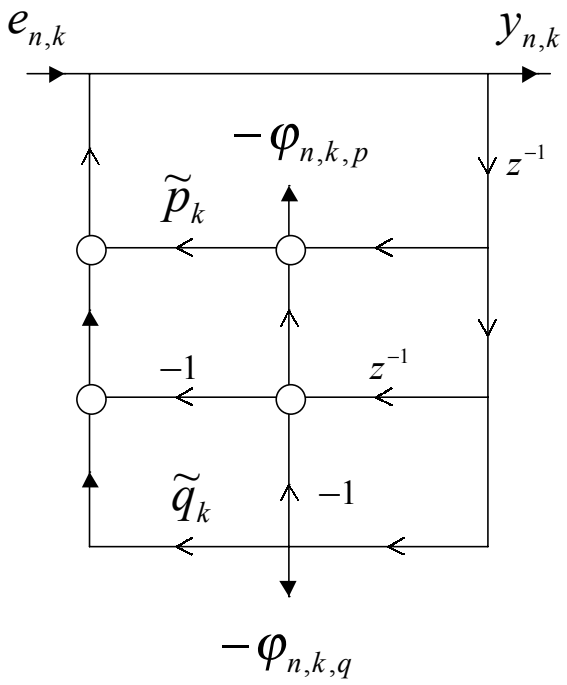

Figure 3-3 Direct LSF_2 Synthesis Section $\left(1-\hat{A}_k(z)\right)$ using $[\tilde{p}_k \quad \tilde{q}_k]$ Parameters.

Table 3-1 DLSF Algorithm using CRLS – SA.

For $k = 1,2,\ldots M$,

$$d_{n,k} = y_{n,k} \tag{3-45a}$$

$$\mathbf{y}_{n-1,k} = \begin{bmatrix} y_{n-1,k} & y_{n-2,k} \end{bmatrix}^T \tag{3-45b}$$

$$\varphi_{n,k,p} = -\left(1+z^{-1}\right)z^{-1}\frac{e_{n,M}}{2\left(1+2\hat{F}_{n,k}(z)\right)} \tag{3-45c}$$

$$\varphi_{n,k,q} = -\left(1-z^{-1}\right)z^{-1}\frac{e_{n,M}}{2\left(1+2\hat{F}_{n,k}(z)\right)} \tag{3-45d}$$

$$\varphi_{n,k} = \begin{bmatrix} \varphi_{n,k,p} & \varphi_{n,k,q} \end{bmatrix}^T \tag{3-45e}$$

$$\boldsymbol{\pi}_{n,k} = \varphi^H{}_{n,k}\mathbf{P}_{n-1,k} \tag{3-45f}$$

$$\boldsymbol{\kappa}_{n,k} = \frac{\mathbf{P}_{n-1,k}\varphi_{n,k}}{\lambda + \boldsymbol{\pi}_{n,k}\varphi_{n,k}} \tag{3-45g}$$

$$e_{n,k} = y_{n,k+1} = d_{n,k} + 2\hat{\mathbf{f}}^H_{n-1,k}\mathbf{y}_{n-1,k} \tag{3-45h}$$

$$\hat{\mathbf{f}}_{n,k} = \hat{\mathbf{f}}_{n-1,k} + \boldsymbol{\kappa}_{n,k}e^*_{n,M} \tag{3-45i}$$

$$\mathbf{P}_{n,k} = \lambda^{-1}\mathbf{P}_{n-1,k} - \lambda^{-1}\boldsymbol{\kappa}_{n,k}\varphi^T_{n,k}\mathbf{P}_{n-1,k} \tag{3-45j}$$

3.4 Comparing CRLS-SA and DLSF Adaptation Techniques

The AR process of 10th order, with LSF_{10} parameters shown in Table 3-2, is used to verify and compare the performance of LSF_2 estimation using the CRLS-SA and DLSF algorithms. The results are reflected in Table 3-2. In order to facilitate comparison of the LSF_2 results with the original LSF_{10}, the LSF_2 cascade results are converted to the corresponding AR(2) using (3-39). The AR(2) are convolved together to give AR(10) coefficients, and then the corresponding implicit LSF_{10} estimates are computed.

Table 3-2 Statistical Performance Comparison between DLSF and CRLS-SA approaches.

Method		DLSF	CRLS-SA
LSF root	Actual	Mean / Std. Dev. / MSE (10^{-3})	Mean/Std.Dev./MSE (10^{-3})
P1	0.0445	0.04472 / 0.00253 / 0.0228	0.04536 / 0.00577 / 0.0172
P2	0.1026	0.10331 / 0.00231 / 0.0280	0.10841 / 0.00309 / 0.0177
P3	0.2276	0.23550 / 0.00294 / 0.0580	0.23829 / 0.00289 / 0.1023
P4	0.2711	0.28326 / 0.00605 / 0.1517	0.30441 / 0.00229 / 0.5058
P5	0.3219	0.33533 / 0.01118 / 0.1307	0.37705 / 0.00299 / 0.8933
Q1	0.0671	0.06664 / 0.00194 / 0.0404	0.06637 / 0.00453 / 0.0291
Q2	0.1241	0.12730 / 0.00470 / 0.0692	0.14728 / 0.00470 / 0.1147
Q3	0.2505	0.25096 / 0.00161 / 0.0185	0.25261 / 0.00149 / 0.0070
Q4	0.3088	0.31067 / 0.00299 / 0.0354	0.31956 / 0.00212 / 0.0125
Q5	0.3763	0.41048 / 0.00958 / 0.0654	0.40958 / 0.00591 / 0.1683

For this comparison, an AR process of length 2400 samples is generated with a white noise input to the AR(10) filter characterized by the LSF parameters tabulated in Table 3-2. The AR process realization is divided into 100 non-overlapping, concatenating blocks, each of length 240 samples. The CRLS-SA estimated AR_2 coefficients are convolved together to produce AR_{10} coefficients. The corresponding LSF_{10} are obtained by finding the roots of $P'(z)$ and $Q'(z)$ as defined in (3-4) and (3-5). Also, the LSF_2 are directly estimated using DLSF, the LSF_2 converted to AR_2 polynomials. The AR_2 coefficients are convolved together to produce AR_{10} coefficients and the corresponding LSF_{10} computed from (3-4) and (3-5). The mean, the standard deviation of the LSF_{10} estimates, obtained from both methods are shown in Table 3-2, along with the corresponding true values. We define the mean square error MSE measure between each of the true LSF_{10} parameter $R_k, 1 \le k \le 10$ and estimated LSF_{10} parameter $\hat{R}_k, 1 \le k \le 10$, obtained over I blocks as

$$MSE_k = \frac{1}{I}\sum_{i=1}^{I}\left(R_{k,i} - \hat{R}_{k,i}\right)^2 \tag{3-46}$$

where LSF_{10} vector R is defined as

$$R = [\text{P1} \quad \text{P2} \quad \ldots \quad \text{P5} \quad \text{Q1} \quad \text{Q2} \quad \ldots \quad \text{Q5}]^T \tag{3-47}$$

The MSE between each of the true LSF_{10} parameter $R_k, 1 \le k \le 10$ and estimated LSF_{10} parameter $\hat{R}_k, 1 \le k \le 10$, obtained over 100 blocks is tabulated in Table 3-2. We observe that the mean of the DLSF estimates are closer to the true values than the means of the CRLS-SA estimates. Moreover, the DLSF estimates yield a smaller MSE and a larger variance (for most of the LSF_{10} estimates) compared to the CRLS-SA estimates, i.e. the DLSF estimates are less biased comparatively. The coefficient trajectories obtained by DLSF and CRLS-SA are illustrated in Figure 3-4 and Figure 3-5 respectively. From Figure 3-4 and Figure 3-5, we see that the DLSF estimated LSF_{10} are closer to the true values than those obtained with CRLS-SA.

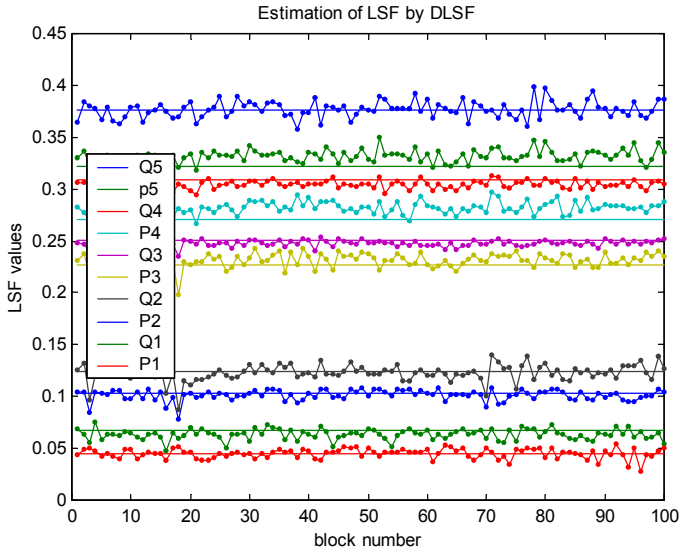

Figure 3-4 Coefficient Trajectories obtained by DLSF.

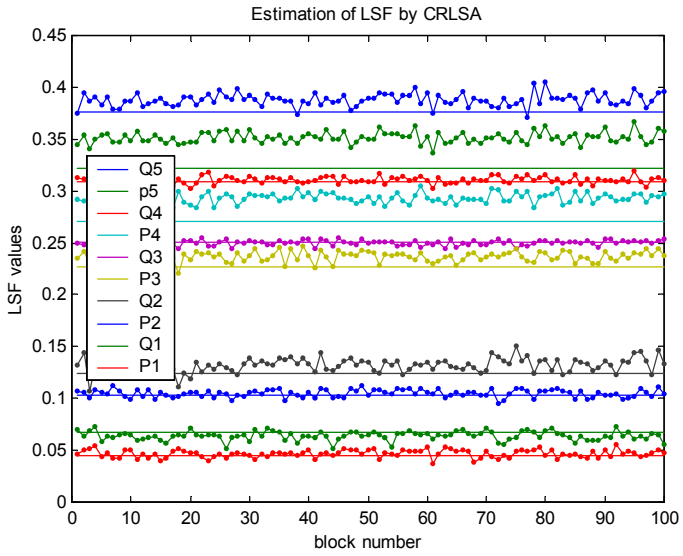

Figure 3-5 Coefficient Trajectories obtained by CRLS-SA.

Note that in *MSE* we compared true LSF_{10} and estimated LSF_{10} parameters. In order to see the performance of CRLS-SA and DLSF for LSF_2 parameters itself (as opposed to converting LSF_2 to AR(2), then to AR(10) and to LSF_{10}) we use the coefficient distance LSF_dist measure [14] defined as

$$LSF\_dist = 10\log_{10}\left(\sum_{i=1}^{N}\left(\mathbf{f}_{N_i}-\hat{\mathbf{f}}_{N_i}\right)^2\right) \tag{3-48}$$

where $\mathbf{f}_N$ and $\hat{\mathbf{f}}_N$ are the true and estimated LSF_2 parameter vectors for an N^{th} order AR process modeled using five 2nd order AR sections,

$$\mathbf{f}_N = \begin{bmatrix}\tilde{p}_1 & \tilde{q}_1 & \tilde{p}_2 & \tilde{q}_2 & \cdots & \cdots & \tilde{p}_{N/2} & \tilde{q}_{N/2}\end{bmatrix}^T \tag{3-49}$$

$$\hat{\mathbf{f}}_N = \begin{bmatrix}\hat{\tilde{p}}_1 & \hat{\tilde{q}}_1 & \hat{\tilde{p}}_2 & \hat{\tilde{q}}_2 & \cdots & \cdots & \hat{\tilde{p}}_{N/2} & \hat{\tilde{q}}_{N/2}\end{bmatrix}^T \tag{3-50}$$

Let us define

$$\hat{\tilde{\mathbf{p}}} = \begin{bmatrix} \hat{\tilde{p}}_1 & \hat{\tilde{p}}_2 & \dots & \hat{\tilde{p}}_{N/2} \end{bmatrix}^T \tag{3-51}$$

$$\hat{\tilde{\mathbf{q}}} = \begin{bmatrix} \hat{\tilde{q}}_1 & \hat{\tilde{q}}_2 & \dots & \hat{\tilde{q}}_{N/2} \end{bmatrix}^T \tag{3-52}$$

The mean of the coefficient distances obtained using DLSF and CRLS-SA for the given AR_{10} process, over 100 blocks, is -28.0310 dB and -22.1713 dB respectively. Thus, the DLSF yields -5.8597 dB better than CRLS-SA in terms of the coefficient distances measure. The distribution of the estimated poles obtained by using DLSF and CRLS-SA, for each block of the given AR_{10} process, is illustrated in Figure 3-6 and Figure 3-7 respectively. Recall that the farther the poles from the unit circle, the more is the noise associated with them. One important observation from Figure 3-6 is that the poles estimated by DLSF adaptation moves farther towards the true pole at radius 0.6, than the CRLS-SA adapted estimated poles. In other words, simplified DLSF catches the poles associated with noise better than CRLS-SA.

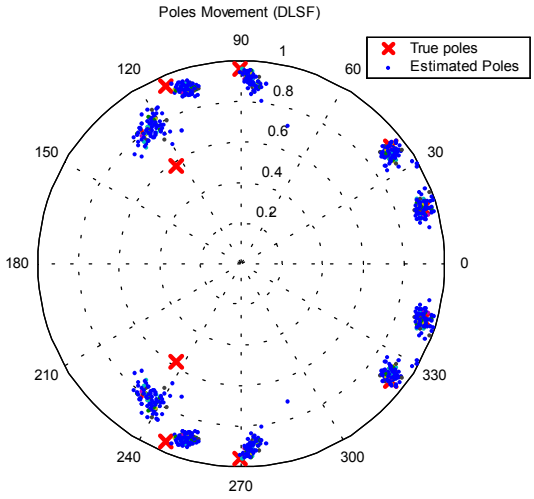

Figure 3-6 Pole Estimate Distribution using DLSF.

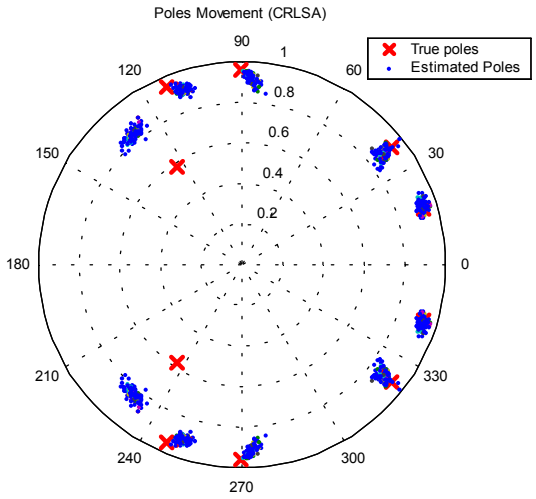

Figure 3-7 Pole Estimate Distribution using CRLS-SA.

In Figure 3-8 and Figure 3-9 $\hat{\tilde{\mathbf{p}}}$ in (3-51) and $\hat{\tilde{\mathbf{q}}}$ (3-52) obtained by simplified DLSF and CRLS - SA are compared after removal of the true values.

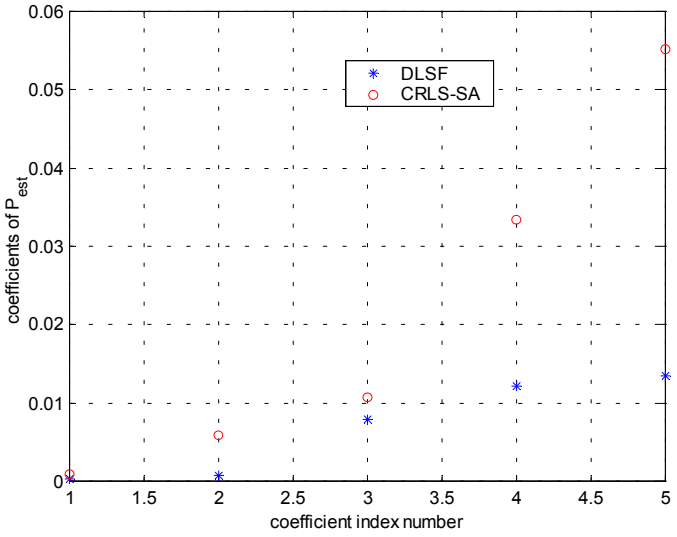

Figure 3-8 Mean of the Estimated $\hat{\tilde{\mathbf{p}}}$ Parameter Vector from DLSF and CRLS-SA with the True Values Removed.

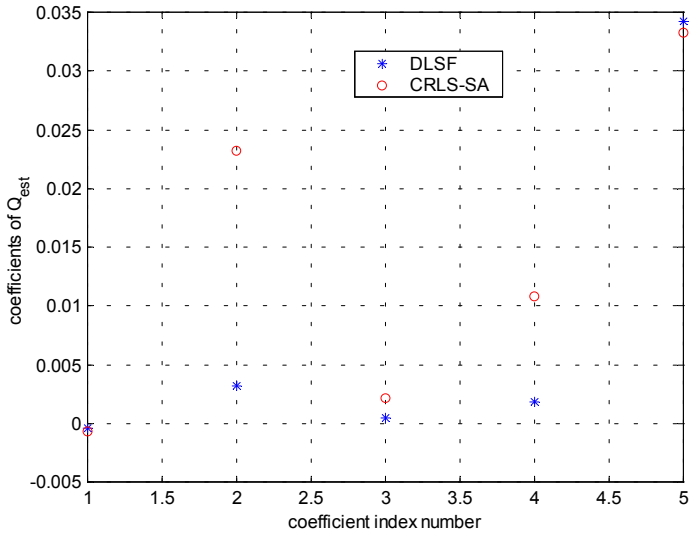

Figure 3-9 Mean of the Estimated $\hat{\tilde{\mathbf{q}}}$ Parameter Vector from DLSF and CRLS-SA with the True Values Removed.

From Figure 3-8 and Figure 3-9, the LSF_2 parameters estimated by DLSF are closer to the zero axis (representing the true values) than those estimated by CRLS-SA. This behavior comes from the difference in the adaptation parameters and filter structures of the methods. The choice of a filter structure affects the operation of the overall algorithm [1]. Representing the analysis filter $H_a(z)$ in (2-17) as a combination of even and odd symmetric filter structures in (3-1) provides an efficient mechanism to adapt the LSF_2 parameters and help adapt to the true values. We could also see the close relation between LSF_2 parameters and the formant frequencies in (3-43). It was shown earlier [4] that the DLSF estimated LSF_{10}, yield relatively unbiased estimates, whereas the auto-correlation method yields more biased estimates, for the same speech-like AR_{10} process for which the true LSF parameters are given in Table 3-2. Based on the results of this analysis, we subsequently use the DLSF algorithm to estimate the filter coefficients, in the form of the second-order-section LSF parameters $\tilde{p}_k$ and $\tilde{q}_k$.

Chapter 4 Vector Quantization

Quantization of transmission parameters reduces the needed transmission rate or bandwidth of the communication channel. In the speech coder proposed here, the transmission parameters include the LSF parameters, the excitation signal and the excitation gain. All the parameters are to be quantized with as few bits as possible, retaining the transparency quality. The transparency quality is obtained if the quantized signal does not introduce any audible distortion. It is achieved when the following requirements are simultaneously satisfied [10][11],

- The average spectral distortion (SD) is less than 1 dB.
- Fewer than 2% of the codebook vectors have SD greater than 2 dB.
- No the codebook vectors with SD greater than 4 dB exist.

We consider the frequency weighted spectral distortion measure [11] SD_{fw} to measure the quality of the codebooks. The SD_{fw} is defined as

$$SD_{fw} = \sqrt{\frac{1}{W_0}\sum_{f=0}^{4000}\left|W_B(f)\right|^2\left(10\log_{10}\left(\frac{\left|A_q\left(e^{j2\pi f}\right)\right|^2}{\left|A\left(e^{j2\pi f}\right)\right|^2}\right)\right)^2} \tag{4-1}$$

where W_0 is the sum of weighting factors

$$W_0 = \sum_{f=0}^{4000} W_B(f) \tag{4-2}$$

$W_B(f)$ is the Bark weighting factor given by

$$W_B(f) = \frac{1}{25+75\left(1+1.4\left(\frac{f}{1000}\right)^2\right)^{0.69}} \tag{4-3}$$

In practice, a 256-point FFT is used to approximate the spectrum of $A(z)$ and $A_q(z)$.

Transparent quantization of LPC parameters means that the reconstructed speech produced quantized LSF is audibly indistinguishable from the original speech produced from the un-quantized LSF. It can be achieved using 32-34 bits/frame using scalar quantization (SQ) techniques, while vector quantization (VQ) using 24-26 bits/frame retains similar quality [22]. The VQ approaches include full-search VQ (FSVQ), split VQ (SVQ), multi-stage VQ (MSVQ), shape-gain VQ (SGVQ), and tree-search VQ (TSVQ). A FSVQ requires a large codebook to meet the transparency requirements, thus leading to an increase in the search complexity. However, using a structured codebook such as a tree-search codebook, split codebook and multi-stage codebook the search complexity can be reduced remarkably [11].

We describe the vector quantization, codebook structure and the *generalized Lloyd algorithm* for codebook design in Section 4.1. The basic concepts behind the SVQ and the SGVQ are described in Section 4.2 and Section 4.3 respectively.

4.1 Vector Quantization

A vector quantizer (VQ) quantizes a block of input data as a single vector, thus VQ is multidimensional, unlike the uni-dimensional scalar quantizer. A VQ produces less distortion than a scalar quantizer for the same number of bits [7]. VQ exploits linear and non-linear dependence among the vectors to be quantized. VQ allows different cell shapes, like hexagons, to fill the region $\Re^K$. This is unlike in SQ, where the region $\Re^K$ is filled with rectangular cells. For example, let $\aleph$ be a plane as shown in Figure 4-1(a). Figure 4-1(b) and Figure 4-1(c) depict the rectangular and the hexagonal partitions of $\aleph$ respectively. For the same number of quantization cells, hexagonal cell shapes result in a lower worst-case error for a statistical error criterion such as Euclidean distance than rectangular cells, provided the edge effects are negligible [7]. The advantage of VQ is that it allows different cell shapes like hexagons that fill the region $\Re^K$ more efficiently than the rectangular cells in allowed in SQ.

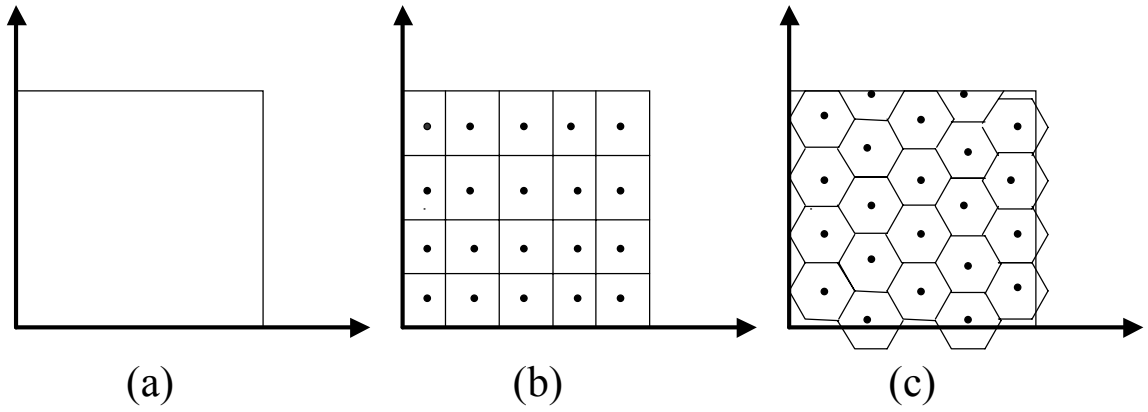

Figure 4-1 Two Dimensional Quantization a) Plane, ℵ b) Rectangular Partition c) Hexagonal Partition.

Let $\mathbf{y}$ be an input vector of length K defined as

$$\mathbf{y} = [y_0 \quad y_1 \quad \dots \quad y_{K-1}]^T \tag{4-4}$$

VQ quantizes $\mathbf{y}$ to $\hat{\mathbf{y}}$,

$$\hat{\mathbf{y}} = Q(\mathbf{y}) \tag{4-5}$$

where $Q(\ )$ is the vector quantization operation.

The vector $\hat{\mathbf{y}}$ is chosen from a set of L code words $\mathbf{C} = \mathbf{c}_i,\ 0 \le i \le L-1$ such that the nearest neighbor rule

$$d(\mathbf{y}, \mathbf{c}_i) \le d(\mathbf{y}, \mathbf{c}_k), \quad 0 \le i < k \le L-1, i \ne k \tag{4-6}$$

is satisfied. The parameter $d(\mathbf{x}, \mathbf{z})$ denotes a distortion measure between $\mathbf{x}$ and $\mathbf{z}$, $\mathbf{x}$ and $\mathbf{z}$ being column vectors of length K each. The set of vectors $\mathbf{C}$, called the codebook, consists of L code vectors and is represented as

$$\mathbf{C} = [\mathbf{c}_0 \quad \mathbf{c}_1 \quad \dots \quad \mathbf{c}_{L-1}] \tag{4-7}$$

where $\mathbf{c}_i$ is the i^{th} code vector of length K, defined as

$$\mathbf{c}_i = \begin{bmatrix} c_{0,i} & c_{1,i} & \dots & c_{K-1,i} \end{bmatrix}^T \tag{4-8}$$

Thus, a vector quantizer Q of dimension $K \, x \, L$ maps a vector in the K dimensional Euclidean subspace $\Re^K$, to the finite codebook $\mathbf{C}$ of size $K \, x \, L$ and chooses the code vector that is closest to the input vector.

$$Q : \Re^K \rightarrow \mathbf{C} \tag{4-9}$$

The resolution rate r is the number of bits per vector component used to represent the input vector,

$$r = \left(\log_2 L\right) \Big/ K = B \Big/ K \tag{4-10}$$

where B is the number of bits needed to address the code words in $\mathbf{C}$. VQ achieves fractional values of resolution, as defined in (4-10), essential for low-bit rate applications [7].

A distortion measure $d(\mathbf{y}, \hat{\mathbf{y}})$ associated with quantizing any input vector $\mathbf{y}$ to $\hat{\mathbf{y}}$ can be used to evaluate the performance of a system. A quantizer is good if the average distortion is small. The most common distance measures are the squared error (SE) distance measure denoted by $d_{SE}(\mathbf{y}, \hat{\mathbf{y}})$, the mean-square error (MSE) distance measure denoted by $d_{MSE}(\mathbf{y}, \hat{\mathbf{y}})$, and the weighted mean square error ($WMSE$) distance measure denoted by $d_{WMSE}(\mathbf{y}, \hat{\mathbf{y}})$. These distance measures are respectively defined as follows

$$\begin{aligned} SE = d_{SE}(\mathbf{y}, \hat{\mathbf{y}}) &= (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) \\ &= \sum_{k=1}^{K} (y_k - \hat{y}_k)^2 \end{aligned} \tag{4-11}$$

$$MSE = d_{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{K} (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) \tag{4-12}$$

$$WMSE = d_{WMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{K} (\mathbf{y} - \hat{\mathbf{y}})^T \mathbf{W} (\mathbf{y} - \hat{\mathbf{y}}) \tag{4-13}$$

where $\mathbf{W}$ is a symmetric and positive weighting matrix of size $K \times K$ [7], $\mathbf{y}$ and $\hat{\mathbf{y}}$ are column vectors each of the same length. The *WMSE* measure includes the *MSE* measure for $\mathbf{W} = \mathbf{I}$, the identity matrix. In speech and image compression applications, a matrix $\mathbf{W}$ that depends explicitly on the input vector $\mathbf{y}$ to be quantized, is chosen to obtain perceptually motivated distortion measures [7]. For example, let $\mathbf{W}(\mathbf{y})$ be $\|\mathbf{y}\|^{-2}\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and $\|\mathbf{y}\| > 0$, then (4-13) turns out to be the ratio of noise energy to signal energy,

$$d_{WMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\|\mathbf{y} - \hat{\mathbf{y}}\|^2}{\|\mathbf{y}\|^2} \tag{4-14}$$

For a given noise energy $\|\mathbf{y} - \hat{\mathbf{y}}\|^2$, $d_{WMSE}(\mathbf{y}, \hat{\mathbf{y}})$ is higher when $\mathbf{y}$ is small than when $\mathbf{y}$ is large.

4.1.1 Codebook Structure

The codebook $\mathbf{C}$ of dimension $K \times L$, is obtained by training with a large number of input vectors; these are also called the training vectors. The training data is partitioned into L Voronoi regions or cells R_i in the K dimensional subspace $\Re^K$ such that there exists no overlap region [7],

$$\bigcup_{i=1}^{L} R_i = \Re^K; \quad R_i \bigcap R_j = [\ ], \ for \quad i \neq j \tag{4-15}$$

The centroid $\mathbf{c}_i$ of the Voronoi region R_i is the code word representing R_i. The Voronoi region R_i [10][17] associated with a code vector $\mathbf{c}_i$ is defined as

$$R_i = \left[\mathbf{y} \subset \Re^K : \|\mathbf{y} - \mathbf{c}_i\| < \|\mathbf{y} - \mathbf{c}_k\|; \quad 0 \leq i \leq L-1, i \neq k\right] \tag{4-16}$$

VQ quantizes $\mathbf{y}$ to $\mathbf{c}_i$ if $\mathbf{y} \in R_i$.

4.1.2 Codebook Initializations

A codebook can be initialized in different ways. The initialization of the codebook affects the overall performance of the resulting codebook. An initial codebook can be obtained by randomly

selecting code vectors from the training database, for example by considering the first L vectors from the training database. Any other codebook representing the variety of test vectors is also reasonable.

4.1.3 Codebook Training

An optimum quantizer chooses the code word that results in minimum distortion,

$$Q(\mathbf{y}) = \mathbf{c}_i = \hat{\mathbf{y}}_i; \quad iff\ d(\mathbf{y},\mathbf{c}_i) \le d(\mathbf{y},\mathbf{c}_k), 0 \le i < k \le L-1 \tag{4-17}$$

for which the codebook should be trained with enough data such that each code vector $\mathbf{c}_i$ is optimized to give minimum average distortion D_i in cell R_i [5], defined in (4-16)

$$\begin{aligned} D_i &= \mathrm{E}\{[d(\mathbf{y},\mathbf{c}_i)/\mathbf{y} \in R_i]\} \\ &= \int_{y \in \mathbf{C}_i} d(\mathbf{y},\mathbf{c}_i) p(\mathbf{y}) d\mathbf{y} \end{aligned} \tag{4-18}$$

where $p(\mathbf{y})$ is the probability density function of vectors that result in the quantized vector $\hat{\mathbf{y}}_i$ lying in cell R_i.

4.1.4 The Generalized Lloyd Algorithm

The Generalized Lloyd Algorithm is an iterative clustering algorithm popular for codebook design [7]. The algorithm can be described as follows

1. To form the initial codebook $\mathbf{C}$, choose L arbitrary code vectors $\mathbf{c}_i$, $i = 0, 1, \ldots, L-1$ from the training database.
2. Based on the nearest neighbor rule, as defined in (4-6),
   a. Assign training vector $\mathbf{y}_n$, $n = 0, 1, \ldots, P-1$ to cell R_{i^*} if

$$i^* = \arg\min_i [d(\mathbf{y}_n,\mathbf{c}_i)] \quad i = 0, 1, \ldots L-1$$

   where P is the total number of training vectors in the training database.
   b. If $\mathbf{y}_n$ is nearest to two or more code vectors then it is assigned to the cell with lowest index.

$$d(\mathbf{y}_n, \mathbf{c}_i) = d(\mathbf{y}_n, \mathbf{c}_j); i \neq j, i < j \;\; then \;\; \mathbf{y}_n \in R_i$$

3. Update the code vector $\mathbf{c}_i$, by taking the mean of all the vectors assigned to cell R_i

$$\mathbf{c}_i = \frac{1}{M_i} \sum_{k=1}^{M_i} \mathbf{y}_k \quad \forall \, \mathbf{y}_k \in R_i$$

where M_i is the total number of training vectors $\mathbf{y}_k \in R_i$.

4. Compute the average distortion

$$D = \frac{1}{P} \sum_{n=0}^{P-1} d(\mathbf{y}_n, Q(\mathbf{y}_n))$$

where $\mathbf{y}_n$ is n^{th} training vector and P is the total number of training vectors such that

$$P = \sum_{i=0}^{L-1} M_i$$

In order to increase the likelihood of the best codebook obtained, we train the codebook number of times (5-or more times), each time we perform operations from step 2 to step 4. The codebook with smallest D is chosen as the final codebook. The spectral distortion measure SD_{fw}, defined in (4-1), is evaluated for each code vector in the final codebook, to verify if the codebook meets the transparency requirements. If the codebook does not meet the transparency requirements, another codebook with a different initialization is trained.

4.2 Split VQ

A *split* VQ (SVQ) is used to code the LSF parameters, as will be described in Section 5.2. In split VQ (SVQ), the input vector $\mathbf{y}$ as defined in (4-4) is split into two or more sub-vectors. The initial codebook is split accordingly. Depending on the application, a FSVQ or sub-optimum search VQ is performed for each sub-codebook. In SVQ, the input vector can be split into any number of sub-vectors and the size of each sub-codebook can also be varied to obtain better overall performance.

The search and storage complexity in a SVQ is very low compared to that of a FSVQ. For example, the input vector $\mathbf{y}$, of length 10 say, is divided into two sub-vectors (5,5) which are encoded with 8 and 9 bits respectively. This results in two codebooks, of dimension 256 x 5 and 512 x 5 respectively. Comparatively, a 17-bit FSVQ needs a codebook of dimension 2^{17} x 10. Thus, SVQ is computationally simpler than FSVQ and requires relatively less memory [19]. However, the performance of SVQ is lower than that of FSVQ in terms of the quality of quantized vector $\hat{\mathbf{y}}$.

4.3 Shape – Gain VQ

A *shape-gain vector quantizer* (SGVQ) is used to code the excitation, as will be described in Section 5.2. In SGVQ, the vector to be quantized is represented by a *gain* and a *shape* vector. The gain value is the root mean-square value of the vector components. The input vector normalized by gain value represents the *shape* vector. The basic idea of a SGVQ is that the same shape can recur with different gain values [7]. So shape vectors can be quantized independently of the gain value. The gain g of a vector $\mathbf{y}$ as defined in (4-4) is a scalar quantity and is scalar quantized. It is defined as the Euclidean norm of the vector and is given by

$$g = g(\mathbf{y}) = \|\mathbf{y}\| = \sqrt{\sum_{i=0}^{N-1} y_i^2} \tag{4-19}$$

The shape $\mathbf{s}$ of the vector $\mathbf{y}$ is defined as

$$\mathbf{s} = \mathbf{s}(\mathbf{y}) = \frac{\mathbf{y}}{g(\mathbf{y})} \tag{4-20}$$

such that

$$\|\mathbf{s}\| = 1 \tag{4-21}$$

Thus, the vector $\mathbf{y}$ can be represented as

$$\begin{aligned} \mathbf{y} &= g(\mathbf{y})\mathbf{s}(\mathbf{y}) \\ &= g\mathbf{s} \end{aligned} \tag{4-22}$$

The shape vector $\mathbf{y}$ is vector quantized. The quantized version of $\mathbf{y}$ denoted as $\hat{\mathbf{y}}$ is defined as

$$\hat{\mathbf{y}} = \hat{g}\hat{\mathbf{s}} \tag{4-23}$$

where $\hat{g}$ and $\hat{\mathbf{s}}$ are the quantized versions of gain and shape respectively.

Let $\mathbf{C}_S$ and $\mathbf{C}_g$ be the shape and gain codebooks of sizes N_s and N_g respectively. The encoding process is done in two steps. Based on the squared error distortion measure defined in (4-11),

$$\begin{aligned} d(\mathbf{y}, \hat{g}\hat{\mathbf{s}}) &= \|\mathbf{y} - \hat{g}\hat{\mathbf{s}}\|^2 \\ &= \|\mathbf{y}\| + \hat{g}^2 - 2\hat{g}(\mathbf{y}^t\hat{\mathbf{s}}) \end{aligned} \tag{4-24}$$

First the best shape vector that maximizes the $\mathbf{y}^t\hat{\mathbf{s}}$ term in (4-24) is selected from the shape codebook $\mathbf{C}_S$. Secondly, for a given $\hat{\mathbf{s}}$, the $\hat{g}$ is chosen which minimizes the distortion d defined in (4-24). Thus, $\hat{\mathbf{y}}$ as defined in (4-23) is obtained.

Chapter 5 Quantization of Transmission Parameters

Based on the ordering and distortion independence properties of LSF_2 as described in Section 3.1, SVQ is chosen for the quantization of LSF_2. The characteristics of the speech in the TIMIT database are described in Section 5.1. We considered two initialization methods: random initialization (RI) and selective random initialization (SRI). The efficiency of the two initializations is compared in terms of frequency weighted spectral distortion measure. In order to see the effect on reconstructed speech and transmission rate with increased number of bits and different split we choose SVQ (2,2,2,4)SRI with a (9-8-8-9) bit configuration and SVQ(2,4,4)SRI with a (9-9-9) bit configuration to quantize LSF_2. Both the codebooks are tested on TIMIT testing database. The results are summarized in Section 5.2. The operation and performance of Discrete Cosine Transform (DCT) based speech coder are detailed in Section 5.3.

5.1 Source of Speech

Speech data from the TIMIT database [25] is used to analyze the performance of the designed encoder. The TIMIT corpus is exclusively designed to provide speech data for the acquisition of acoustic-phonetic knowledge. The TIMIT data is used for the development of automatic speech recognition systems. TIMIT contains speech sampled at 16 kHz. Data contains 6300 sentences, spoken by 630 speakers, both male and female, from 8 dialect regions in the USA. The dialect regions are given in Table 5-1. The sentences include dialect sentences, phonetically compact sentences and phonetically diverse sentences.

The TIMIT database is divided into two sections: *a training database* and *a testing database*. About 73% of the total data is used to train the codebook and the remaining data is used to test the optimized codebook. The training and testing databases have no speech or speaker in common. The TIMIT training database contains speech from 462 speakers, both male and female from all 8 dialect regions and is 189 min long. The TIMIT test database is further categorized as *core testing database* and

Table 5-1 Dialect Regions Considered in TIMIT Database.

Dialect Region Index	Dialect Region
DR1	New England
DR2	Northern
DR3	North Midland
DR4	South Midland
DR5	Southern
DR6	New York City
DR7	Western
DR8	Army Brat

complete testing database. The core testing database contains speech of 9.78 min duration, spoken by 24 speakers, 2 male speakers and 1 female speaker from each of the 8 dialect regions. Each speaker read a different set of 8 sentences. The selected texts contain at least one occurrence of each phoneme. The core testing database is appended with speech from 144 speakers from all the dialect regions. The complete testing database contains speech of 63 min duration. The core testing database is the minimum recommended testing database, where as the complete testing database is used for extensive testing. More information is found in National Institute of Standards and Technology (NIST) publications (NTIS# PB91-100354). We use the complete testing database to test our codebooks.

In practical speech coding scenarios, speech is usually sampled at 8 kHz and modeled using 10^{th} order AR filters [5]. TIMIT contains speech sampled at 16 kHz. Speech from the TIMIT training database is lowpass filtered at 3.2 kHz and down-sampled by a factor of 2. The resultant speech at the 8 kHz sampling frequency is used in all our experiments. Speech is considered to be stationary within a 16 to 32 ms interval, leading to a corresponding analysis interval of 128 to 256 samples [5]. We choose a 20 ms interval corresponding to L=160 samples as our analysis frame. Thus we have I=567,000 frames in the training database and J=189,100 frames in the testing database.

5.2 Quantization of LSF Parameters

Based on the results obtained in Section 3.4, we use the DLSF algorithm to estimate the filter coefficients, in the form of the LSF_2 parameters $\tilde{p}_k$ and $\tilde{q}_k$ as defined in (3-38). Since most of the speech energy is concentrated in the lower frequencies, the lowest frequency, associated with most of the speech energy, is assigned to the first section and the highest frequency is assigned to the last section. In other words, we order the formant frequencies such that

$$f_1 < f_2 < f_3 < f_4 < f_5 \tag{5-1}$$

where f_k are the formant frequencies of the k^{th} section as defined in (3-43), associated with the P polynomial in (3-2) and the Q polynomial as defined in (3-3) frequencies. The lowest frequency f_1 is assigned to the first section, f_2 to the second section, and so on, until f_5 is assigned to the last section in the cascaded filter. Recall, we are adapting, the LSF_2 parameters and not the formant frequencies. Substituting (3-43) in (5-1), we have

$$\begin{gathered}\frac{1}{2\pi}\cos^{-1}\left(\frac{\widetilde{p}_1+\widetilde{q}_1}{2\sqrt{1-\widetilde{p}_1+\widetilde{q}_1}}\right)<\frac{1}{2\pi}\cos^{-1}\left(\frac{\widetilde{p}_2+\widetilde{q}_2}{2\sqrt{1-\widetilde{p}_2+\widetilde{q}_2}}\right)<\ldots<\frac{1}{2\pi}\cos^{-1}\left(\frac{\widetilde{p}_5+\widetilde{q}_5}{2\sqrt{1-\widetilde{p}_5+\widetilde{q}_5}}\right) \\ \Downarrow \\ \left(\frac{\widetilde{p}_1+\widetilde{q}_1}{2\sqrt{1-\widetilde{p}_1+\widetilde{q}_1}}\right)>\left(\frac{\widetilde{p}_2+\widetilde{q}_2}{2\sqrt{1-\widetilde{p}_2+\widetilde{q}_2}}\right)>\ldots>\left(\frac{\widetilde{p}_5+\widetilde{q}_5}{2\sqrt{1-\widetilde{p}_5+\widetilde{q}_5}}\right)\end{gathered} \tag{5-2}$$

Let us rearrange the LSF_2 parameter *pairs* $[\widetilde{p}_k \quad \widetilde{q}_k]$ in descending order of $\widetilde{p}_k$

$$\widetilde{p}_1 > \widetilde{p}_2 > \widetilde{p}_3 > \widetilde{p}_4 > \widetilde{p}_5 \tag{5-3}$$

In order to see the validity of ordering LSF_2 parameter according to (5-3) satisfies the ordering of the formant frequencies as per (5-1), we give an example for a pair of $[\widetilde{p}_k \quad \widetilde{q}_k]$, $k=1,2$ set as tabulated in Table 5-2. The formant frequency as defined in (3-43) and BW_3 as defined in (3-44) are calculated and tabulated in Table 5-2.

Case1: The distance between $[\widetilde{p}_1 \quad \widetilde{q}_1]$, pair is increased from 0.1932 to 0.6746. This increased f_1 by 0.0619, with an increase of 0.0726 dB in BW_3. The results are tabulated in Table 5-2. The formant frequency f_1' of the new $[\widetilde{p}_1 \quad \widetilde{q}_1]$ is closer to that of the $[\widetilde{p}_2 \quad \widetilde{q}_2]$ by 0.0619. Thus, by increasing the BW_3 of $[\widetilde{p}_1 \quad \widetilde{q}_1]$ and making f_1 and f_2 closer, the condition in (5-3) is still valid.

Table 5-2 Ordering of $[\widetilde{p}_k \quad \widetilde{q}_k]$ in Case 1.

LSF Parameter	Actual	Formant Frequency/ BW3	Case 1	Case1 Formant Frequency/ BW3
$\widetilde{p}_1$	0.1592	0.2386 / 0.1989	0.1592	0.3005 / 0.2715
$\widetilde{q}_1$	-0.0304		-0.5154	
$\widetilde{p}_2$	-0.3447	0.3298 / 0.1980	-0.3447	0.3298 / 0.1980
$\widetilde{q}_2$	-0.5254		-0.5254	

Case2: The distance between $[\widetilde{p}_1 \quad \widetilde{q}_1]$, is further increased by 0.1896. This increased f_1 by 0.1526, with an increase of 0.1471 dB in BW_3. The results are tabulated in Table 5-3. Also note that the formant frequency f_1'' of the new $[\widetilde{p}_1 \quad \widetilde{q}_1]$ is closer to that for $[\widetilde{p}_2 \quad \widetilde{q}_2]$ by 0.0614 compared to that in Case 1. However, this time the condition in (5-1) is not satisfied, though the condition in (5-3) is satisfied.

Table 5-3 Ordering of $[\widetilde{p}_k \quad \widetilde{q}_k]$ in Case 2.

LSF Parameter	Actual	Formant Frequency/ BW3	Case 2	Formant Frequency/ BW3
$\widetilde{p}_1$	0.1592	0.2386 / 0.1989	0.1592	0.3912 / 0.3460
$\widetilde{q}_1$	-0.0304		-0.7132	
$\widetilde{p}_2$	-0.3447	0.3298 / 0.1980	-0.3447	0.3298 / 0.1980
$\widetilde{q}_2$	-0.5254		-0.5254	

This experiment suggests that ordering LSF_2 parameters according to (5-3) does not necessarily ensure the inequality in (5-1). As a next step, we computed the format frequencies of the LSF_2 parameters of speech segments obtained using simplified DLSF. For each segment we verified the validity of (5-3). We found that by arranging the LSF_2 parameters according to (5-3) satisfied the inequality in (5-1). Thus, for the LSF_2 parameters of speech, arranging the LSF_2 parameters according to (5-3) satisfied the inequality in (5-1). With this observation, we arrange the LSF_2 parameters according to (5-3) and assign the $[\widetilde{p}_1 \quad \widetilde{q}_1]$ pair to the first section, the $[\widetilde{p}_2 \quad \widetilde{q}_2]$ pair to the second section, and so on, until the $[\widetilde{p}_5 \quad \widetilde{q}_5]$ pair is assigned to the last section in the cascaded filter. This saves the computation of finding the actual formant frequency for each section. Note that $\widetilde{q}_k$ is not rearranged separately; it remains linked with its original pair parameter $\widetilde{p}_k$. This ensures that

each LSF_2 parameter pair $[\tilde{p}_k \quad \tilde{q}_k]$ represents a set of complex conjugate poles. The LSF_2, for each frame of length L, are obtained for all frames in the testing and training databases. For each LSF_2 parameter vector, the $[\tilde{p} \quad \tilde{q}]$ pairs are ordered such that (5-3) is satisfied.

After we estimate the LSF_2 parameters, the next step is to quantize the LSF_2 before transmission. The quantization of LSF_2 is done by employing either scalar quantization (SQ) or vector quantization (VQ). Uni-dimensional scalar quantization is computationally much simpler than multi-dimensional vector quantization. However, when the samples to be quantized are correlated, vector quantization significantly outperforms scalar quantization and the cost of complexity associated with VQ becomes tolerable. In other words, as the samples of the vector to be quantized become more independent, the simple scalar quantizer yields almost the same performance as the more complex vector quantizer [7]. The *LSF* in a frame are correlated with those in the same frame as well as with the *LSF* in the neighboring frames. These properties, termed inter-frame and intra-frame properties as described in Section 3.1, allow us to choose VQ to quantize *LSF*. It was reported [22] that the transparent quantization of LPC parameters can be achieved using 32-34 bits/frame using SQ techniques, while VQ using 24-26 bits/frame achieves similar quality. The distortion independence property of *LSF* described in Section 3.1, enables us to take advantage of SVQ for quantizing *LSF*. In SVQ the input vector is split into two or more sub-vectors. Any distortion produced in quantizing a sub-vector has little effect on the overall performance due to the distortion independence property of *LSF*. Moreover, SVQ is computationally simpler and requires less memory in comparison to FSVQ, as described in Section 4.2. Making use of the ordering property of *LSF*, a sub-optimum search is made on the sub-codebooks as described later in this section. This further simplifies the search complexity.

We consider two types of initializations of the LSF_2 codebook: *random initialization* (RI) and *selective random initialization* (SRI). In RI, to represent the initial codebook, we randomly choose the LSF_2 vectors from the training database. The codebook with random initialization is split into three sub-codebooks, namely CB1, CB2, and CB3. We use CB1 for LSF_2 section 1, CB2 for LSF_2 sections 2 and 3, and CB3 for LSF_2 sections 4 and 5, as illustrated in Figure 5-1. We denote this codebook as SVQ (2,4,4)RI. Since we already arranged the LSF_2 parameter vectors to satisfy (5-3), the $[\tilde{p} \quad \tilde{q}]$ with highest $\tilde{p}$ gets assigned to the first section, and the $[\tilde{p} \quad \tilde{q}]$ pair with lowest $\tilde{p}$ to the last section. Each sub-codebook is assigned 9 bits, so that we use a total of 27 bits for quantizing all LSF_2.

Similarly, we partition the estimated LSF_2 for the frames in the training database base into three partitions T1, T2 and T3.

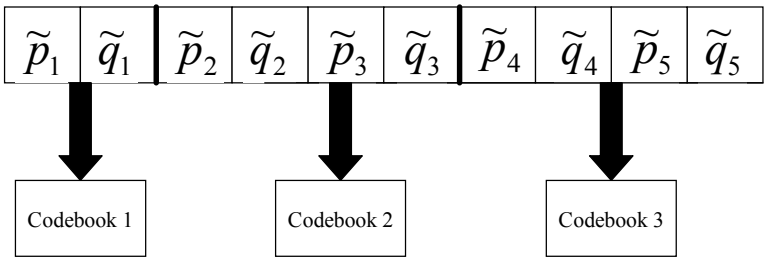

Figure 5-1 SVQ (2,4,4).

The codebook CB1 is trained with the vectors in the T1 database partition, CB2 with the T2 partition and CB3 with the T3 partition. The Generalized Lloyd Algorithm is used for training, and described in Section 4.1. In SRI, the codebook is initialized such that it explicitly represents both male and female speakers from all the dialect regions in the training database. We denote the SVQ(2,4,4) with SRI SVQ(2,4,4)SRI and is obtained in the same lines as we obtained SVQ(2,4,4)RI.

The SVQ search procedure is as follows. The LSF_2 parameters of the frames in the testing database are quantized using the obtained codebooks CB1, CB2 and CB3. Let the quantized LSF_2 be denoted by LSF_{q_2}. In order to ensure the ordering property of the quantized LSF_2 parameters a sub-optimum search is made on the sub-codebooks. A full search is made on CB1 and the best codevector, the one that results in minimum MSE between $[\tilde{p}_1 \quad \tilde{q}_1]$ and $[\hat{p}_1 \quad \hat{q}_1]$ as defined in (4-12), is chosen. Next, in order to quantize the second sub-vector, only the codevectors in CB2 are searched whose first element is less than $\hat{p}_1$. Similarly sub-optimum searches are made on the successive codebooks.

The SD using SD_{fw} as defined in (4-1) is evaluated for the obtained codebook. Secondly, the quality of the obtained codebook is assessed in terms of the signal-to-noise ratio SNR and the segmental signal-to noise ratio SNR_{seg} of the reconstructed speech. The signal-to-noise ratio SNR is defined as

$$SNR = 10\log_{10}\left[\sum_{n=m_j-N+1}^{m_j} \frac{s_n^2}{[s_n - \hat{s}_n]^2}\right] \tag{5-4}$$

where s_n are the components of the un-quantized speech segment $\mathbf{s}$ of length LJ, defined as

$$\mathbf{s} = \begin{bmatrix} s_0 & s_1 & \dots & s_{LJ-1} \end{bmatrix}^T \tag{5-5}$$

and $\hat{s}_n$ are the components of the quantized speech segment $\hat{\mathbf{s}}$ of length LJ, defined as

$$\hat{\mathbf{s}} = \begin{bmatrix} \hat{s}_0 & \hat{s}_1 & \dots & \hat{s}_{LJ-1} \end{bmatrix}^T \tag{5-6}$$

The indices m_0, m_1, … m_{J-1} in (5-4) are the end-times for the J frames, with each being a multiple of L=160. The segmental signal-to-noise ratio SNR_{seg}, a frame-based measure, is formulated as

$$SNR_{seg} = \frac{1}{M}\sum_{j=0}^{J-1} 10\log_{10}\left[\sum_{n=m_j-N+1}^{m_j} \frac{s_n^2}{\left[s_n - \hat{s}_n\right]^2}\right] \tag{5-7}$$

The SNR measure defined in (5-4) is computed for each j^{th} frame, denoted by $SSNR_j$ is defined as

$$SSNR_j = 10\log_{10}\left[\sum_{n=(j-1)L+1}^{jL} \frac{s_n^2}{\left[s_n - \hat{s}_n\right]^2}\right] \tag{5-8}$$

The mean of all these $SSNR_j$ gives SNR_{seg}. In SNR_{seg}, loud and soft portions of speech are weighted equally. However, silence frames result in a large negative SNR_{seg} value. This problem can be overcome by removing the silent frames from the SNR_{seg} calculation, or by replacing all SNR_{seg} values below a threshold value, say 0 dB, with the threshold value. On the other hand, all SNR_{seg} above an upper threshold value are reset to the threshold value. Normally, 35 dB is chosen as an upper threshold value, because the human ear cannot significantly distinguish SNR_{seg} greater than 35 dB from SNR_{seg} of 35 dB [28]. Using the two threshold values mitigates the influence of the (relatively) few frames that do not contribute significantly to the final SNR_{seg}. The SD_{fw} as defined in (4-1) for the LSF_2 vectors in the testing database is evaluated using SVQ(2,4,4)RI and SVQ(2,4,4)SRI

and the corresponding results are tabulated in Table 5-4 and Table 5-5 respectively. We see that both SVQ(2,4,4)RI and SVQ(2,4,4)SRI meets the transparency requirements described in Section 4.1.

Table 5-4 Spectral Distortion from Random Initialization with SVQ(2,4,4)RI.

Spectral Distortion	Mean (dB)	2 dB < Outlier < 4 dB %	Outlier > 4 dB %
SD_{fw}	0.127	0	0

Table 5-5 Spectral Distortion from Selective Random Initialization of SVQ(2,4,4)SRI.

Spectral Distortion	Mean (dB)	2 dB < Outlier < 4 dB %	Outlier > 4 dB %
SD_{fw}	0.0579	0	0

The mean of SD_{fw} obtained by SVQ(2,4,4)SRI is almost half to that obtained by SVQ(2,4,4)RI. Thus SVQ(2,4,4)SRI results in lower SD_{fw} than that obtained by SVQ(2,4,4)RI. Next, we evaluate the quality of the codebook for an AR process, in terms of SNR and SNR_{seg}. We consider an AR filter with poles as illustrated in Figure 5-2a and characterized by the LSF_2 parameters as tabulated in Table 5-6. An AR process $\mathbf{s}_n$ of length 16,000 is generated at the output of the AR filter from a white noise input. The AR process $\mathbf{s}_n$ is then divided into 100 frames of 160 samples each. The LSF_2 parameters for each frame are estimated using DLSF. The mean of the estimated LSF_2 is tabulated in Table 5-6. We observe from the tabulated results that the simplified DLSF estimated are close to the original LSF_2. The LSF_dist as defined in (3-48) between the given LSF_2 and the estimated LSF_2 is –25.2248 dB. Figure 5-2a illustrates the given and DLSF estimated LSF_2. Figure 5-2b illustrates the frequency response of the original and estimated AR filters. Then the estimated LSF_2 are quantized using SVQ(2,4,4)RI and SVQ(2,4,4)SRI and the results are tabulated in Table 5-6. Let LSF'_{q_2} and LSF''_{q_2} denote the quantized LSF_2 using SVQ(2,4,4)RI and SVQ(2,4,4)SRI respectively. Recall that we are modeling the process using five 2nd order sections. Substituting the LSF'_{q_2} parameters in (3-41), the location of poles for each of the 2nd order filter is obtained. Figure 5-3a and

Figure 5-3b illustrates the location of the poles of the cascaded structure characterized by LSF'_{q_2} and LSF''_{q_2} respectively. The LSF_dist defined in (3-48) between the estimated LSF_2 and LSF'_{q_2} is -15.6533 dB. The LSF_dist between the estimated LSF_2 and LSF''_{q_2} is -25.5099 dB. Thus, the LSF_dist obtained using SVQ(2,4,4)SRI is around 9.5 dB less than that obtained using SVQ(2,4,4)RI indicating that the selective random codebook is efficient than a random codebook. We also evaluate the performance of the RI codebook and the SRI codebook on original speech later in this section and then choose the best initialization method.

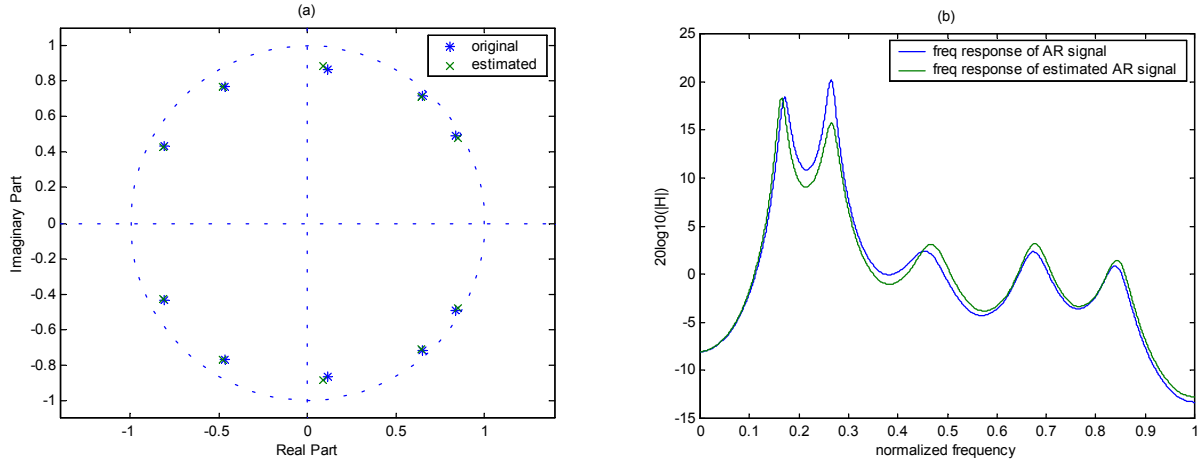

Figure 5-2 a) Original and DLSF-Estimated Poles b) Frequency Response of Original and Estimated AR Filters.

Next, we analyze the effect of quantization of the LSF_2 parameters on the original speech in three steps. We illustrate the results in all three steps for a speech frame $\mathbf{s}_{n,M}$ picked from the testing database. Figure 5-4a illustrates $\mathbf{s}_{n,M}$. Substituting the LSF_{q_2} parameters in (3-41), the location of poles for each of the 2nd order filter is obtained. Figure 5-4b illustrates the location of the estimated poles along with the original poles.

Table 5-6 Original and Estimated LSF_2 along with LSF_{q_2} obtained by SVQ(2,4,4)RI and SVQ(2,4,4)SRI.

LSF_2 Parameter	Original LSF_2	Estimated LSF_2 by DLSF	LSF_{q_2} by SVQ(2,4,4)RI	LSF_{q_2} by SVQ(2,4,4)SRI
$\widetilde{p}_1$	0.8650	0.8733	0.9411	0.8669
$\widetilde{p}_2$	0.6815	0.6829	0.7777	0.6846
$\widetilde{p}_3$	0.2275	0.1884	0.1596	0.1977
$\widetilde{p}_4$	-0.3758	-0.3919	-0.4107	-0.3917
$\widetilde{p}_5$	-0.7278	-0.7375	-0.7501	-0.7338
$\widetilde{q}_1$	0.8031	0.8178	0.9133	0.8133
$\widetilde{q}_2$	0.6221	0.5961	0.5498	0.6323
$\widetilde{q}_3$	-0.0050	-0.0163	-0.0442	-0.0499
$\widetilde{q}_4$	-0.5622	-0.5656	-0.5752	-0.5747
$\widetilde{q}_5$	-0.8865	-0.8908	-0.9029	-0.9020

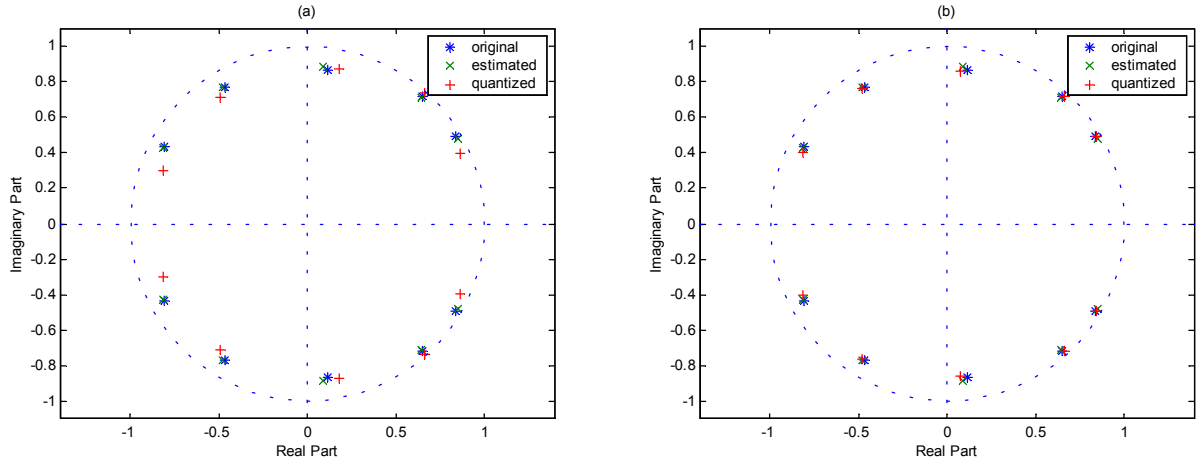

Figure 5-3 a) Original, DLSF- Estimated and SVQ(2,4,4)RI-Quantized Poles b) Original , DLSF- Estimated and SVQ(2,4,4)SRI-Quantized Poles.

In the first step, the reconstructed speech $\breve{\mathbf{s}}_{n,M}$ is obtained from un-quantized LSF_2. The procedure is described as follows. For every M^{th} frame of speech $\mathbf{s}_{n,M}$ in the testing database, the LSF_2 parameters are estimated. The residual signal $\mathbf{e}_{n,M}$ of the M^{th} frame is generated, by inverse filtering $\mathbf{s}_{n,M}$ with the estimated LSF_2 analysis filter. The reconstructed speech $\breve{\mathbf{s}}_{n,M}$ is obtained by filtering $\mathbf{e}_{n,M}$ with the un- quantized LSF_2 synthesis filter as illustrated in Figure 5-5. The procedure is repeated for all the speech frames in the complete testing database.

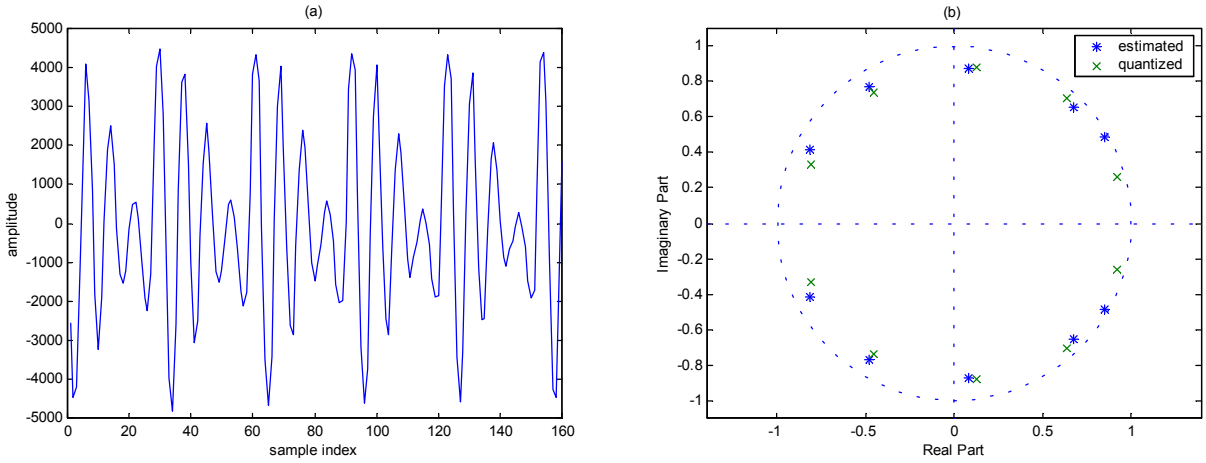

Figure 5-4 a) Speech Frame b) Estimated and Quantized LSF_2.

Each time the $SSNR_j$ between $\mathbf{s}_{n,M}$ and $\breve{\mathbf{s}}_{n,M}$, as defined in (5-8), is calculated. The mean of all $SSNR_j$ results in SNR_{seg} of 290.21 dB. This SNR_{seg} is expected when there is no quantization of parameters. From Figure 5-6, we see that $\mathbf{s}_{n,M}$ and $\breve{\mathbf{s}}_{n,M}$ are indistinguishable.

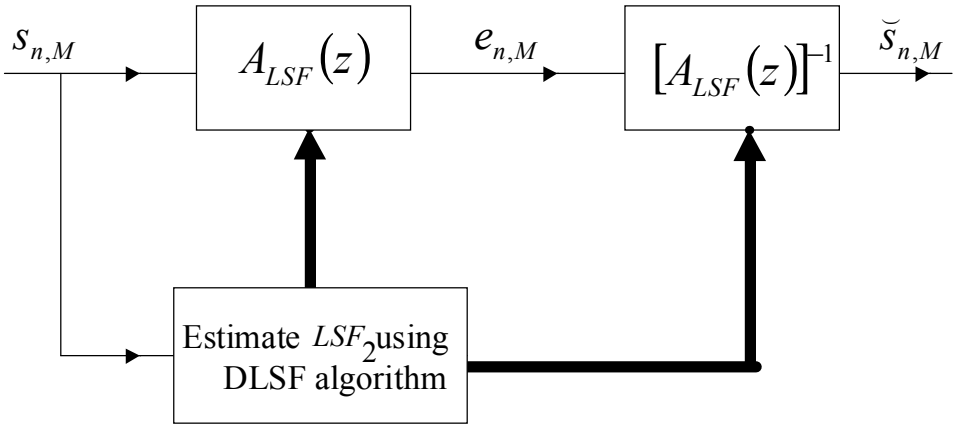

Figure 5-5 Analysis and Synthesis of Speech using Cascaded LSF_2.

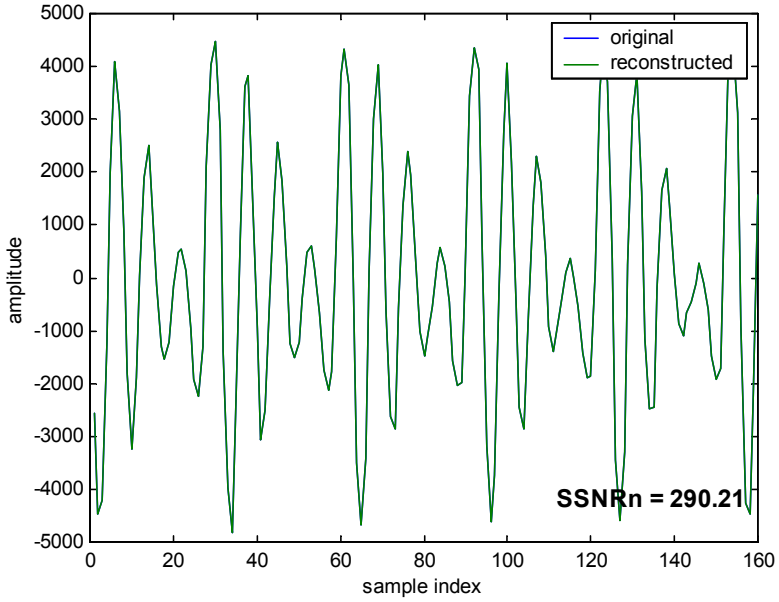

Figure 5-6 Original Speech $\mathbf{s}_{n,M}$ and Reconstructed Speech $\breve{\mathbf{s}}_{n,M}$ Frames without LSF_2 Quantization.

In the next step, we use LSF_{q_2} in the synthesis filter. The residual signal $\mathbf{e}_{n,M}$ of the M^{th} frame is generated, by inverse filtering $\mathbf{s}_{n,M}$ with the estimated LSF_2 analysis filter. Then LSF_2 is quantized using SVQ(2,4,4)RI to give LSF'_{q_2}. The reconstructed speech $\hat{\mathbf{s}}_{n,M}$ is obtained by filtering $\mathbf{e}_{n,M}$ with the LSF_{q_2} synthesis filter as illustrated in Figure 5-7.

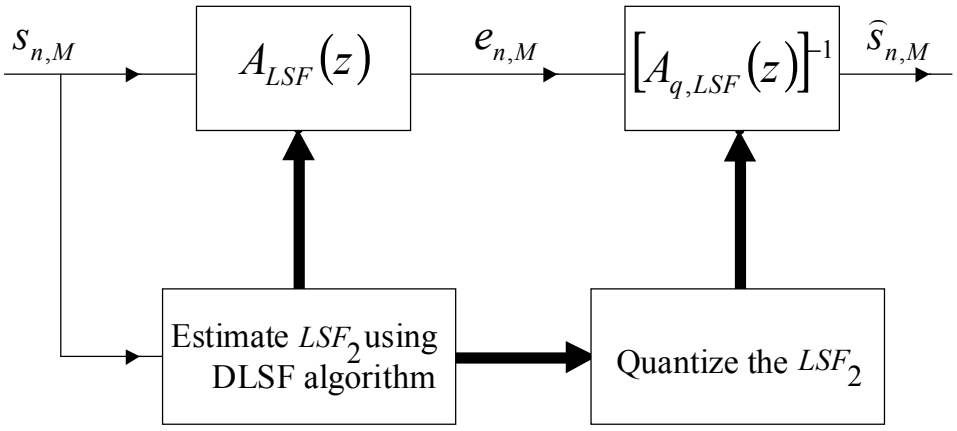

Figure 5-7 Analysis and Synthesis of Speech using Cascaded LSF_{q_2} at the Synthesis filter.

The LSF_dist defined in (3-48) between the estimated LSF_2 and LSF'_{q_2} is -12.6083 dB. The SNR_{seg} corresponding to the frames in the testing database is 7.5 dB. Figure 5-8a illustrates $\mathbf{s}_{n,M}$ and $\hat{\mathbf{s}}_{n,M}$ with $SSNR_j$ of 7.5 dB. The low $SSNR_j$ suggests that the random codebook is not optimum.

The LSF_2 of $\mathbf{s}_{n,M}$ are next quantized using SVQ(2,4,4)SRI to give LSF''_{q_2}. The above procedure is repeated with LSF''_{q_2}. Figure 5-8b illustrates $\mathbf{s}_{n,M}$ and $\hat{\mathbf{s}}_{n,M}$ with $SSNR_j$ of 17.85 dB. Figure 5-8c illustrates the zeros of the LSF_2 analysis filter and the poles of the LSF''_{q_2} synthesis filter. The *LSF_dist* between the estimated LSF_2 and LSF''_{q_2} is -26.1023 dB. So, the LSF_dist obtained using LSF''_{q_2} is 13.49 dB less than that obtained using LSF'_{q_2}, inidicating that the LSF''_{q_2} are closer to the estimated LSF_2. The increase in $SSNR_j$ comes from the selective random initialization of the codebook. Thus, choosing a SRI for a codebook, one that explicitly represents all sorts of speech to which the codebook will be applied later, is a better choice than RI. Henceforth we choose SRI to initialize our LSF_2 codebooks. The performance of the obtained codebook with SRI is evaluated over all the frames in the testing database. Figure 5-9 illustrates the $SSNR_j$ obtained over 500 consecutive frames for all the 7 consecutive iterations of codebook optimization. The average distortion factor D as defined in Section 4.1.4 is obtained over the entire training database in each of the iterations and is illustrated in Figure 5-10. In Figure 5-10 we see that the average distortion takes minimum value at 6th iteration with a value of 0.0092. So, we choose the codebook at the 6th iteration with the lowest D to quantize the LSF_2 parameters.

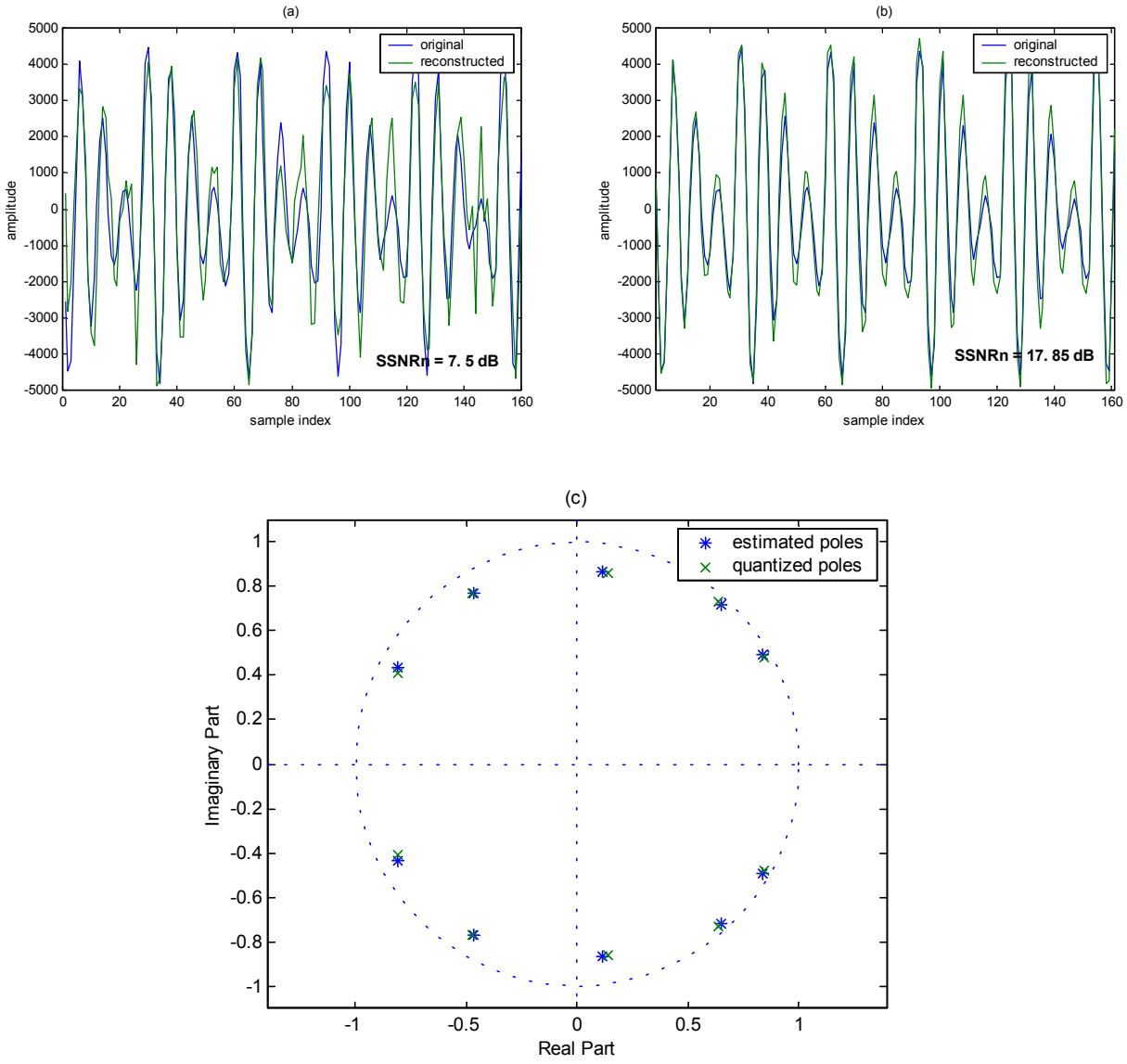

Figure 5-8 Original Speech $\mathbf{s}_{n,M}$ *and Reconstructed Speech* $\hat{\mathbf{s}}_{n,M}$ *with* LSF_{q_2} *at Synthesis Filter with a) SVQ(2,4,4)RI b) SVQ(2,4,4)SRI c) Estimated and SVQ(2,4,4)SRI-Quantized* LSF_2 *.*

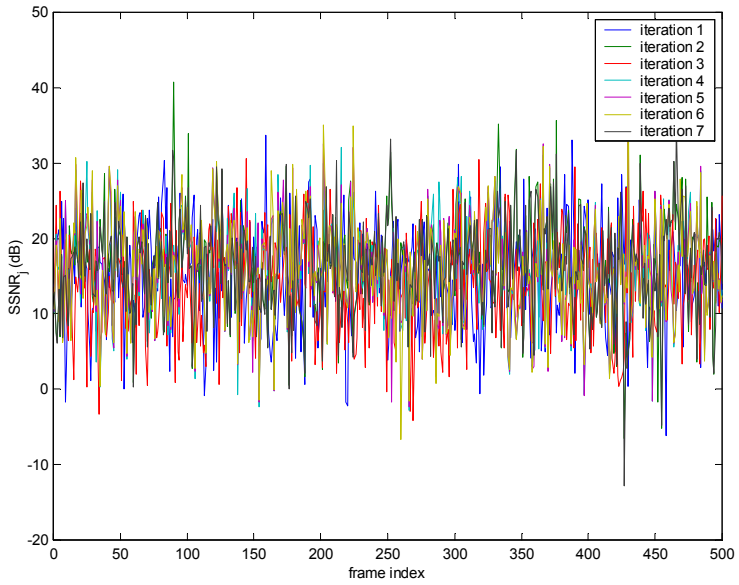

Figure 5-9 $SSNR_j$ Obtained Over 500 Consecutive Frames Using SVQ(2,4,4)SRI.

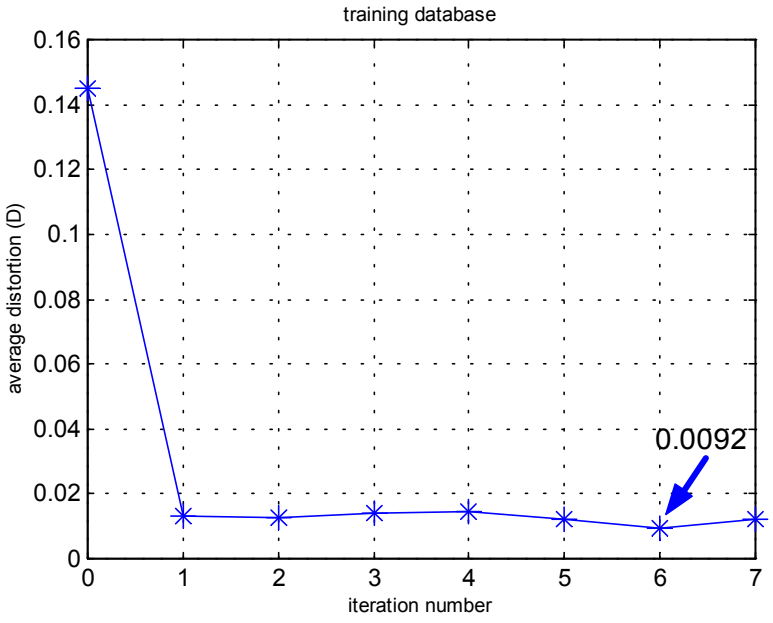

Figure 5-10 Average Distortion Factor D obtained using SVQ(2,4,4)SRI at each iteration.

Figure 5-11 illustrates the histogram of $SSNR_j$ for all the frames in the testing database. The resultant SNR_{seg} is 14.68 dB with a standard deviation of 6.68 dB. In order verify the performance of

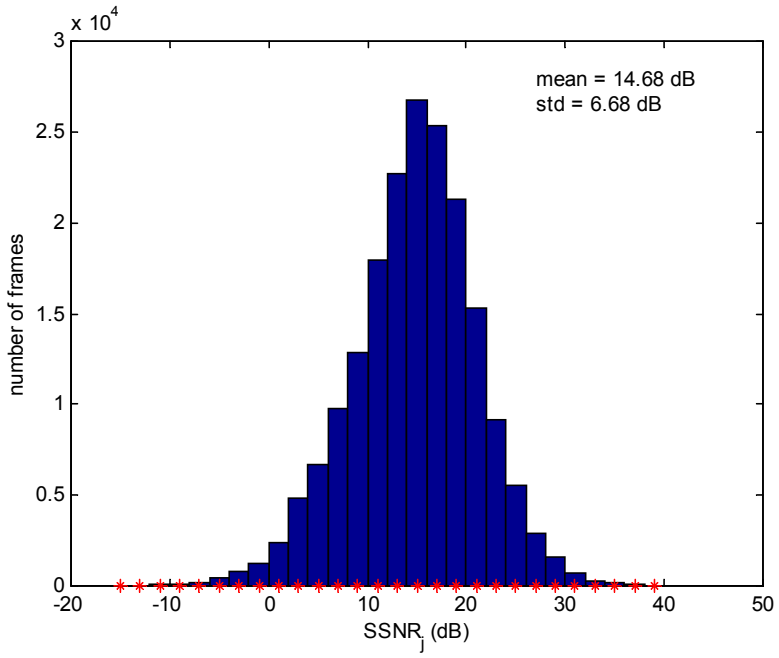

Figure 5-11 Histogram of $SSNR_j$ of Frames in the Testing Database Using SVQ(2,4,4)SRI.

the obtained codebook for female and male speech, we tested the frames corresponding to female and male speakers separately. Figure 5-12 illustrates the histogram of $SSNR_j$ for all the frames in the testing database corresponding to female speakers. The SNR_{seg} turns out to be 15.67 dB with a standard deviation of 6.07 dB. Figure 5-13 illustrates the histogram of $SSNR_j$ for all frames in the testing database corresponding to male speakers, and yielded a SNR_{seg} of 14.19 dB and standard deviation of 6.36 dB. Thus, the codebook performs similarly well for both male and female speech and is not biased towards either male or female speech.

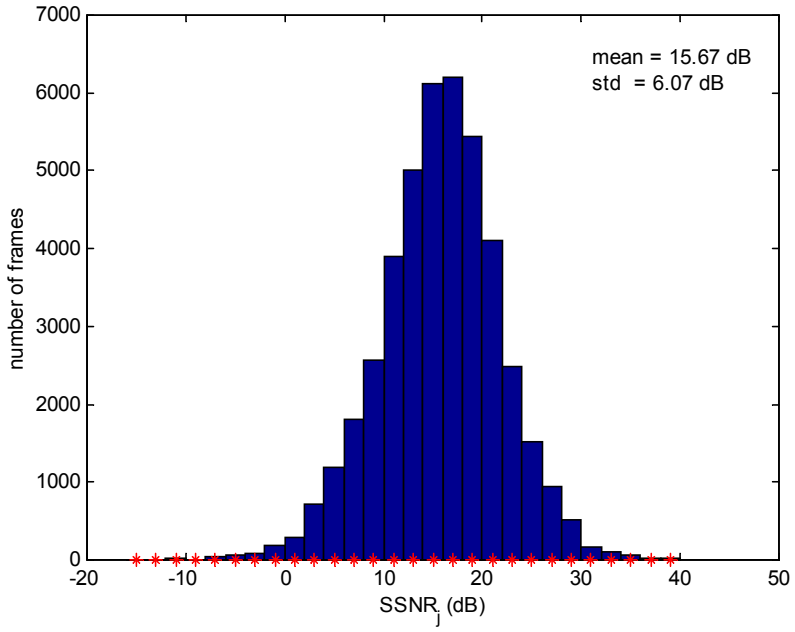

Figure 5-12 Histogram of $SSNR_j$ Corresponding to Female Speech Frames in the Testing Database Using SVQ(2,4,4)SRI.

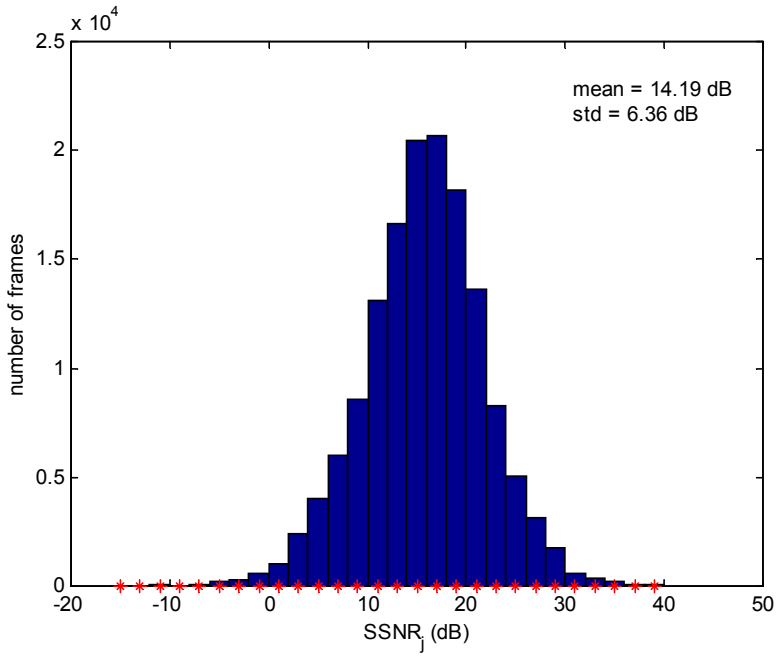

Figure 5-13 Histogram of $SSNR_j$ Corresponding to Male Speech Frames in the Testing Database Using SVQ(2,4,4)SRI.

Figure 5-14 depicts the SNR_{seg} obtained for all the frames in each of the dialect regions, along with the SNR_{seg} obtained for female and male speech in the corresponding dialect region. The lowest SNR_{seg} is 13.36 dB corresponding to the 6th dialect region and the highest SNR_{seg} of 15.2 dB corresponds to the 5th dialect region.

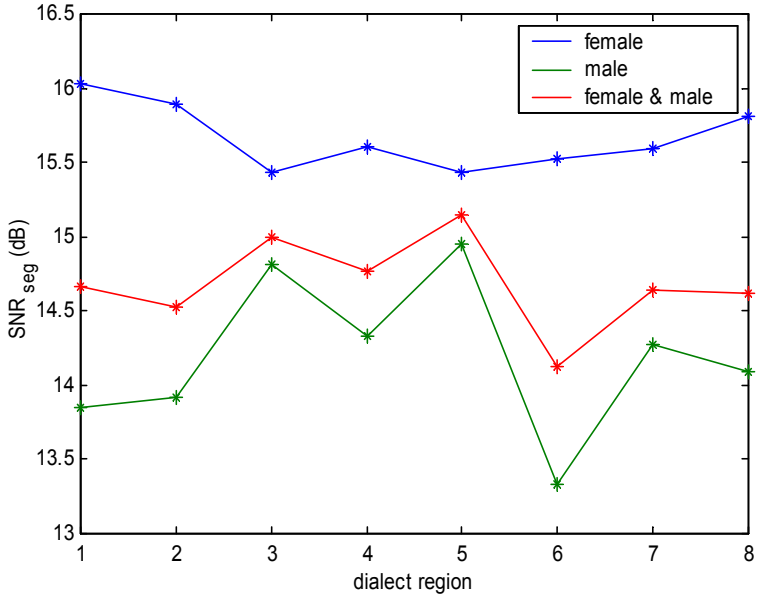

Figure 5-14 The SNR_{seg} Obtained in Each of the Dialect Regions along with the Corresponding SNR_{seg} for the Female and Male Speakers Using SVQ(2,4,4)SRI.

In order to see the effect on SNR_{seg} of an increase in the number of codebooks and bits, we split the LSF_2 into four sub-codebooks, namely CB1, CB2, CB3 and CB4. We use CB1 for the LSF_2 of section 1, CB2 for the LSF_2 of section 2, CB3 for the LSF_2 section 3, and CB4 for the LSF_2 of sections 4 and 5, as illustrated in Figure 5-15. We use SRI and denote this codebook as SVQ (2,2,2,4)SRI. We assign 9-bits each to CB1 and CB4 and 8- bits each to CB2 and CB3. Note that in SVQ(2,4,4)SRI described earlier, we used a (9-9-9) bit configuration. So, in SVQ(2,2,2,4)SRI with a (9-8-8-9) bit configuration we are dividing the CB2 of SVQ(2,4,4)SRI further, into two codebooks. By doing so we are allocating different codebooks to the second and third formant frequencies, the rest being the same as in SVQ(2,4,4)SRI with a (9,9,9) bit configuration. This split gives us a chance to see the consequences of using separate codebooks for the second and third formant frequencies.

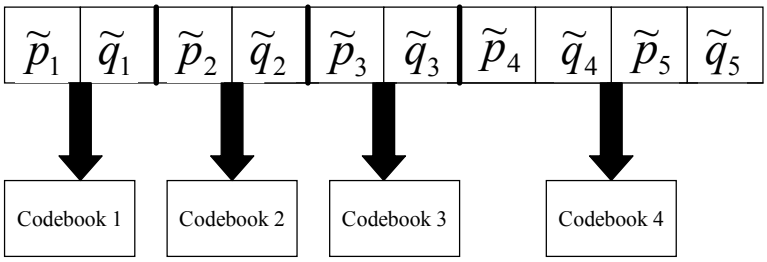

Figure 5-15 SVQ(2,2,2,4).

The codebooks CB1 and CB3 of SVQ(2,4,4)SRI at each iteration during the training process were retained. These codebooks correspond to the CB1 and CB4 codebooks of SVQ(2,2,2,4)SRI with a (9-8-8-9) bit configuration. We start with the initial CB2 of SVQ(2,4,4)SRI obtained by selective random initialization and split the codebook into two. The first half represents CB2 of section 2 and the second half represents CB3 of section 3 for SVQ(2,2,2,4)SRI. We train CB2 and CB3 of SVQ(2,2,2,4)SRI using the Generalized Lloyd Algorithm as described in Section 4.3. At each iteration, we obtain trained CB2 and CB3 codebooks. These codebooks combined with the original CB1 and CB4 codebooks of SVQ(2,4,4)SRI are used to quantize the LSF_2 frames in the testing database. The reconstructed speech $\hat{\mathbf{s}}_{n,M}$ is obtained as illustrated in Figure 5-7. The SD_{fw} as defined in (4-1) is evaluated for LSF_2 in the testing database. The results are tabulated in Table 5-7.

Table 5-7 Spectral Distortion from Selective Random Initialization Using SVQ(2,2,2,4).

Spectral Distortion	Mean (dB)	2 dB < Outlier < 4 dB %	Outlier > 4 dB %
SD_{fw}	0.0485	0	0

Figure 5-16 illustrates the $SSNR_j$ obtained over 500 consecutive frames for all the 7 iterations of SVQ(2,2,2,4)SRI codebook optimization. The average distortion factor D as defined in Section 4.1.4 is obtained over the entire training database in each of the iterations for SVQ(2,2,2,4)SRI with (9-8-8-9) bit configuration. Figure 5-17 illustrates the average distortion obtained using SVQ(2,2,2,4)SRI with (9-8-8-9) bit configuration along with that of SVQ(2,4,4)SRI with (9-9-9) bit configuration as was illustrated in Figure 5-10. In Figure 5-17 we see that the average distortion takes minimum value at 5th iteration with a value of 0.0078 which is smaller than the least distortion obtained using SVQ(2,4,4)SRI with (9-9-9) bit configuration. More over, we see that with SVQ(2,2,2,4)SRI with (9-8-8-9) bit

configuration the best codebook with least distortion is achieved at less number of iterations compared to that of SVQ(2,4,4)SRI with (9-9-9) bit configuration.

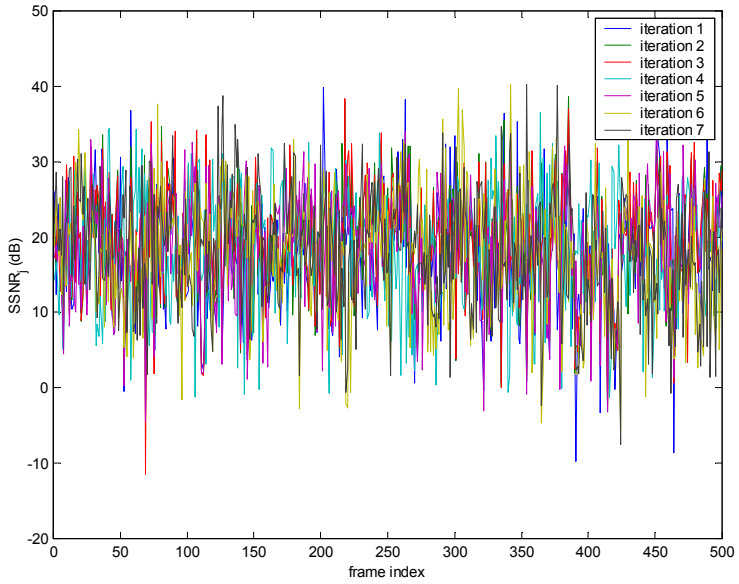

Figure 5-16 $SSNR_j$ Obtained Over 500 Consecutive Frames Using SVQ(2,2,2,4)SRI.

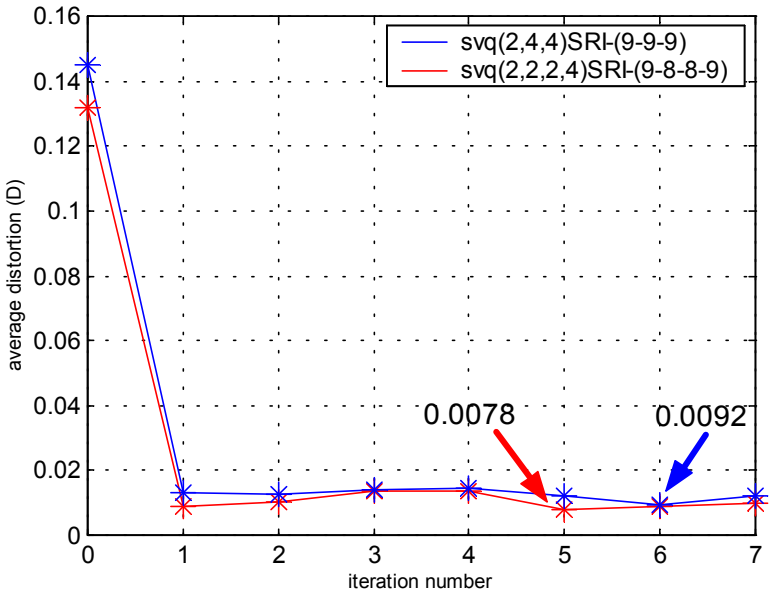

Figure 5-17 Average Distortion Factor D for the SVQ(2,4,4)RI with (9-9-9) Bit Configuration and SVQ(2,2,2,4)SRI with (9-8-8-9) Bit Configuration.

Figure 5-18 illustrates the histogram of $SSNR_j$ for all the frames in the testing database with mean at 17.95 dB and standard deviation of 7.29 dB. The SNR_{seg} takes value as low as -16 dB and as high as 45 dB. Figure 5-19 illustrates the histogram of $SSNR_j$ for all the frames corresponding to female speakers in the testing database. The SNR_{seg} is 18.68 dB with a standard deviation of 7.09 dB.

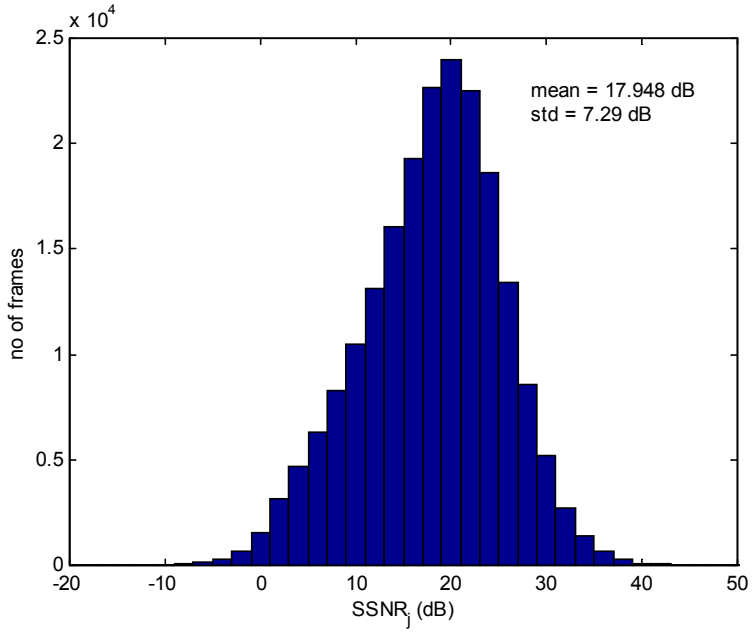

Figure 5-18 Histogram of $SSNR_j$ of Frames in the Testing Database using SVQ(2,2,2,4)SRI.

Figure 5-20 illustrates the histogram of $SSNR_j$ for all the frames corresponding to male speakers in the testing database. The resultant SNR_{seg} is 17.56 dB with a standard deviation of 7.37 dB.

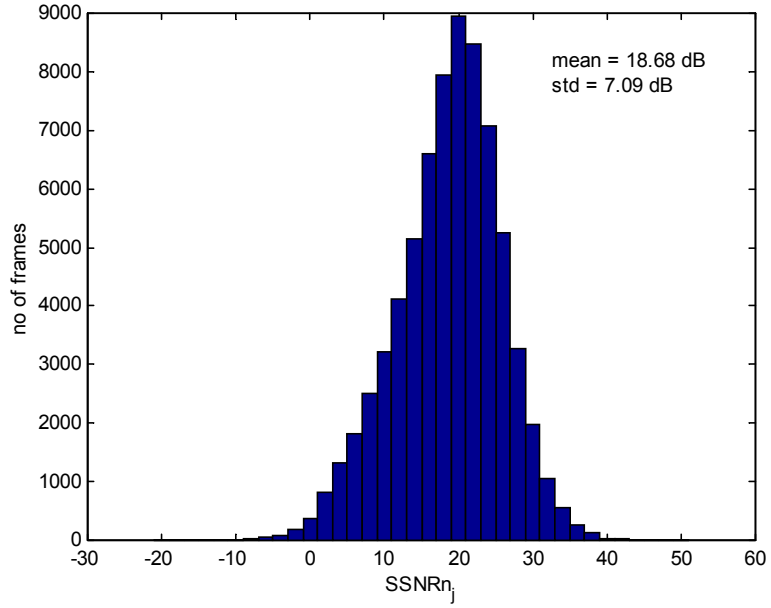

Figure 5-19 Histogram of $SSNR_j$ Corresponding to Female Speech Frames in the Testing Database Using SVQ(2,2,2,4)SRI.

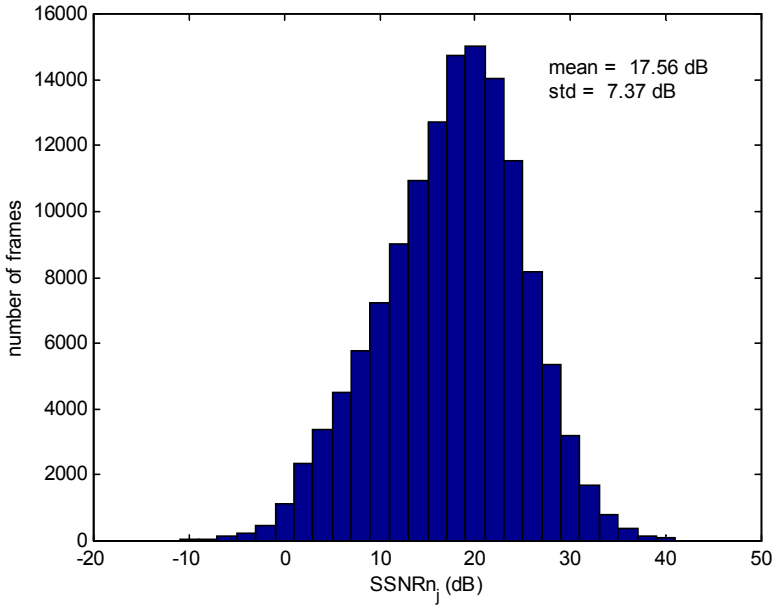

Figure 5-20 Histogram of $SSNR_j$ Corresponding to Male Speech Frames in the Testing Database using SVQ(2,2,2,4)SRI.

Figure 5-21 depicts the SNR_{seg} obtained for all the frames in each of the dialect regions, along with the SNR_{seg} obtained for female and male speech in the corresponding dialect region. The SNR_{seg} for all the frames in the testing database is 17.76 dB with the lowest SNR_{seg} of 17.2 dB corresponding to the 6th dialect region and the highest SNR_{seg} of 18.3 dB corresponding to the 5th dialect region. Thus, we end up with a SNR_{seg} of 17.76 dB using SVQ(2,2,2,4)SRI with a (9-8-8-9) bit configuration, whereas we found SNR_{seg} of 14.68 dB using SVQ(2,4,4)SRI with a (9-9-9) bit configuration. Adding 7 more bits produced a 3.08 dB gain in SNR_{seg}.

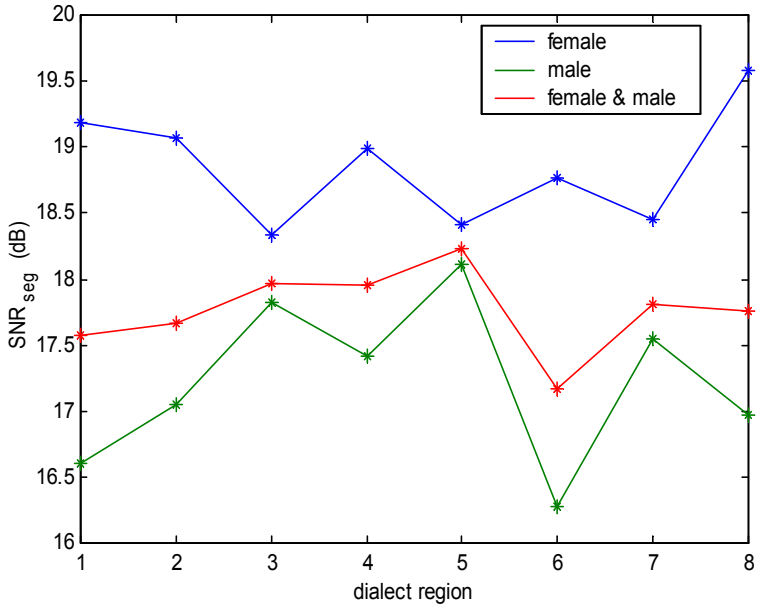

Figure 5-21 The SNR_{seg} Obtained in Each of the Dialect Regions using SVQ(2,2,2,4)SRI .

Finally, we use LSF_{q_2} in both the analysis and synthesis filters as illustrated in Figure 5-22. The residual signal $\tilde{\mathbf{e}}_{n,M}$ of the M^{th} frame is generated, by inverse filtering $\mathbf{s}_{n,M}$ with the LSF_{q_2} analysis filter. The reconstructed speech $\tilde{\mathbf{s}}_{n,M}$ is obtained by filtering $\tilde{\mathbf{e}}_{n,M}$ with the LSF_{q_2} synthesis filter. The procedure is repeated for all the speech frames in the testing database.

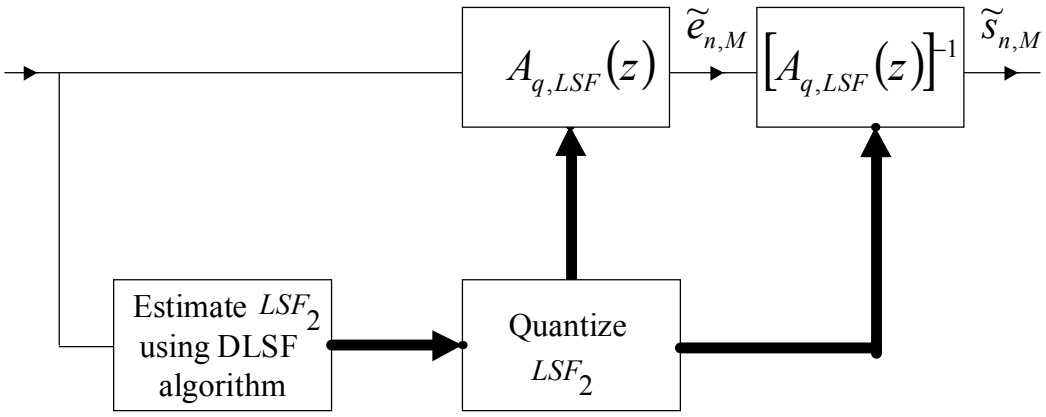

Figure 5-22 Generating Reconstructed Speech Using Cascaded LSF_{q_2} at Both Analysis and Synthesis Filters.

Each time the *SNR* between $\mathbf{s}_{n,M}$ and $\tilde{\mathbf{s}}_{n,M}$, as defined in (5-4), is calculated. The resultant SNR_{seg} of around 290 dB, which is equal to the SNR_{seg} we obtained in the first step, when there is no LSF_2 quantization. Figure 5-23 illustrates $\mathbf{s}_{n,M}$ and $\tilde{\mathbf{s}}_{n,M}$ with SNR_{seg} of 290.14 dB and the corresponding residual signal denoted by $\tilde{\mathbf{e}}_{n,M}$ is illustrated in Figure 5-25. The high SNR_{seg} resulted from using the same LSF_2 parameters in both the analysis and synthesis filters. Any distortion introduced by quantization of LSF_2 is compensated for by $\tilde{\mathbf{e}}_{n,M}$ irrespective of how well the LSF_2 parameters are quantized. For example, consider the same speech frame as depicted in Figure 5-4a, with estimated poles as illustrated in Figure 5-4b. Let LSF_2 be quantized to LSF'_{q_2}. The location of the poles of the cascaded filter with LSF'_{q_2} obtained by using (3-41) are depicted in Figure 5-24a). From Figure 5-24a we see that the quantized poles are quite off from the estimated poles, especially at the lower frequencies with most of the speech energy. Let $\tilde{\mathbf{e}}'_{n,M}$ be the residual signal obtained by inverse filtering $\mathbf{s}_{n,M}$ with the LSF'_{q_2} analysis filter as depicted in Figure 5-25. The reconstructed speech $\tilde{\mathbf{s}}'_{n,M}$ is obtained by filtering $\tilde{\mathbf{e}}'_{n,M}$ with the LSF'_{q_2} synthesis filter. Figure 5-24b) illustrates $\mathbf{s}_{n,M}$ and $\tilde{\mathbf{s}}'_{n,M}$ with SNR_{seg} denoted by SNR'_{seg} of 290.21 dB, which is the same SNR_{seg} we obtained between $\mathbf{s}_{n,M}$ and $\tilde{\mathbf{s}}_{n,M}$.

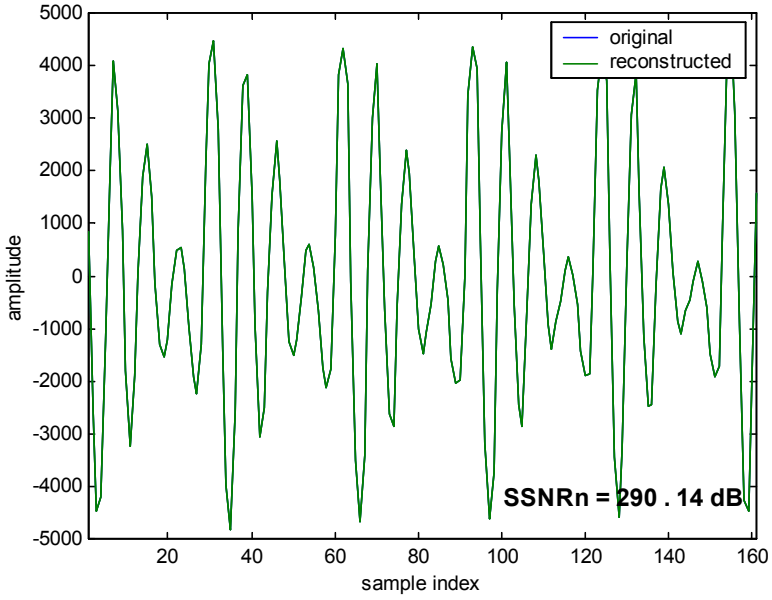

Figure 5-23 Original Speech $\mathbf{s}_{n,M}$ *and Reconstructed Speech* $\widetilde{\mathbf{s}}_{n,M}$ *Frames with* LSF_2 *Quantization.*

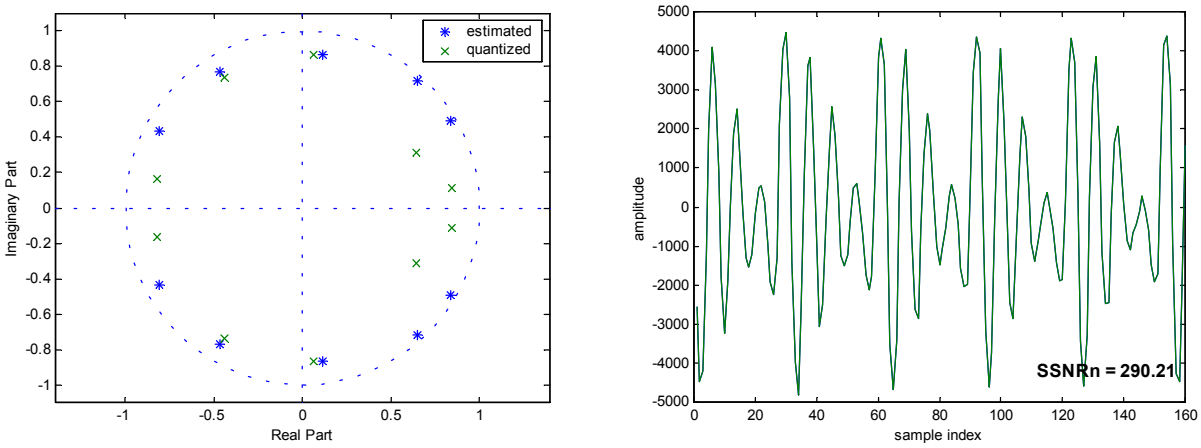

Figure 5-24 a) Estimated and Quantized Poles for $\mathbf{s}_{n,M}$ *b) Corresponding Original Speech* $\mathbf{s}_{n,M}$ *and Reconstructed Speech* $\widetilde{\mathbf{s}}_{n,M}$ *.*

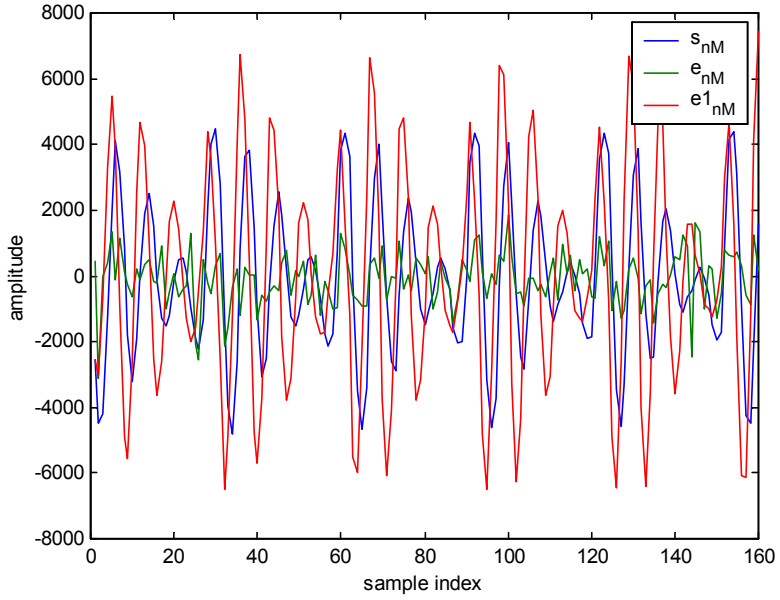

Figure 5-25 Original Signal $\mathbf{s}_{n,M}$ *along with the Residual Signals* $\tilde{\mathbf{e}}_{n,M}$ *and* $\tilde{\mathbf{e}}'_{n,M}$.

The important observation from Figure 5-25 is that $\tilde{\mathbf{e}}'_{n,M}$ is periodic and follows $\mathbf{s}_{n,M}$, and is therefore speech like, whereas $\tilde{\mathbf{e}}_{n,M}$ looks like noise and exhibits no periodicity. Although the SNR_{seg} is same in both the cases, we requires more bits, consequently larger codebooks, to code the speech like residual signal $\tilde{\mathbf{e}}'_{n,M}$ than to code the noise like residual signal $\tilde{\mathbf{e}}_{n,M}$. Thus, implementing the third method, does not mean that any codebook (whether it does or does not meet transparency requirements) can be used to quantize LSF_2. This experiment reveals the fact that there exists a connection between the LSF coding and the excitation coding, and both of them should be optimized simultaneously.

Of all three steps described, we choose to implement third step in our speech coder. In third step the effect of LSF_2 quantization is revealed in the excitation signal. The LSF_2 and excitation vectors can be optimized together to reduce the overall quantization effect on the reconstructed speech. Moreover, it is practical, since the LSF_{q_2} information is present at the encoder. So in the speech coder, we generate the excitation signal $\tilde{\mathbf{e}}_{n,M}$ by inverse filtering $\mathbf{s}_{n,M}$ with the LSF_{q_2} analysis filter. In the next section, we describe the quantization of $\tilde{\mathbf{e}}_{n,M}$.

5.3 Quantization of Excitation Signal

Quantization of the excitation vectors is another very important task in the speech coder. The excitation vectors, 160 samples long each, require a VQ codebook of size 160×2^{160}. The optimization, as well as a codebook search, for such huge codebooks is computationally very complex. In order to alleviate this problem, each excitation frame is split into 4 sub-frames of equal length. A *Shape-Gain* VQ (SG-VQ) is used to quantize the excitation sub-frames as described in Section 5.3.1. However, the optimization, as well as the codebook search, complexity problems still exist.

In order to reduce the computational complexity we choose the *Discrete Cosine Transform* (DCT) to quantize the excitation signals. The DCT is computationally simpler than VQ. The DCT is a fixed, signal-independent transform and alleviates the need for training of the codebook completely. The DCT is used to quantize the excitation vectors as described Section 5.3.2. The operation and performance results based on the quality of the reconstructed speech and bit-rate of the DCT based speech coder is described Section 5.3.2.

5.3.1 Shape - Gain VQ

The SG-VQ described in Section 4.3 is used to quantize the excitation vectors. The excitation frames for the training database are obtained as described in Section 5.3. Each M^{th} excitation frame $\tilde{\mathbf{e}}_{n,M}$ of length $L \times 1$ is divided into K sub-frames $\tilde{\mathbf{e}}_{n,M_k}$, where L is the frame size of 160 samples,

$$\tilde{\mathbf{e}}_{n,M} = \begin{bmatrix} \tilde{\mathbf{e}}_{n,M_0}^T & \tilde{\mathbf{e}}_{n,M_1}^T & \cdots & \tilde{\mathbf{e}}_{n,M_{K-1}}^T \end{bmatrix}^T \tag{5-9}$$

and where each k^{th} sub-frame $\tilde{\mathbf{e}}_{n,M_k}$ is defined as

$$\tilde{\mathbf{e}}_{n,M_k} = \begin{bmatrix} \tilde{e}_{j,M} & \tilde{e}_{j+1,M} & \cdots & \tilde{e}_{j+N-1,M} \end{bmatrix}^T; \quad N = L/K,\ j = Nk \tag{5-10}$$

We divide $\tilde{\mathbf{e}}_{n,M}$ into four sub-frames of 40 samples each. Each sub-frame $\tilde{\mathbf{e}}_{n,M_k}$ is normalized to produce a shape vector and a gain value. The shape vectors and gain values obtained from the excitation vectors constitute the shape and gain database. The initial shape codebook of size 40×2^{40} is obtained by choosing vectors randomly from the shape database. The codebook is to be trained over the entire shape database. However, training a codebook of size 40×2^{40} is still computationally complex and the search procedure is also time-consuming. In order to reduce the computational

complexity we choose the *Discrete Cosine Transform* (DCT) to quantize the excitation signals as described in the next section.

5.3.2 Transform Coding

To overcome the training and search complexity of SG-VQ, we use *transform coding* for the excitation vectors. Transform coding, introduced by Kramer and Mathews in 1956 [20], is a simple and effective method to obtain good data reduction. In transform coding, the input vector $\mathbf{x}$ defined as

$$\mathbf{x} = [x_0 \quad x_1 \quad \ldots \quad x_{N-1}]^T \tag{5-11}$$

can be transformed into a new vector $\mathbf{y}$ of the same length using a suitable linear transformation. The coefficients of $\mathbf{y}$, called *transform coefficients*, are less correlated than the original samples. Most of the information may be concentrated in just a few of the transform coefficients [7]. The transformation matrix $\mathbf{T}$ of dimension $N \times N$ is given by

$$\begin{aligned} \mathbf{T} &= [\mathbf{t}_0 \quad \mathbf{t}_1 \quad \ldots \quad \mathbf{t}_{N-1}] \\ \mathbf{t}_i &= [t_{0,i} \quad t_{1,i} \quad \ldots \quad t_{N-1,i}]^T \end{aligned} \tag{5-12}$$

The transformation matrix $\mathbf{T}$ is an invertible matrix i.e., $\mathbf{T}^{-1}$ exists. The transformed vector $\mathbf{y}$ is given by

$$\mathbf{y} = \mathbf{T}\mathbf{x} \tag{5-13}$$

Depending on the application, either all or a few of the transform coefficients - containing most of the information are scalar quantized and transmitted over the channel. In the latter case, the quantized values along with the corresponding indices are transmitted and the rest of the transform coefficients are assumed to be zero. At the decoder end, the reconstructed signal $\hat{\mathbf{x}}$ is obtained as

$$\hat{\mathbf{x}} = \mathbf{T}^{-1}\hat{\mathbf{y}} \tag{5-14}$$

The choice of $\mathbf{T}$ is very important, as it determines the overall distortion between $\mathbf{x}$ and $\hat{\mathbf{x}}$. The transformation matrix $\mathbf{T}$ is said to be orthogonal, if

$$\mathbf{T}^t = \mathbf{T}^{-1} \tag{5-15}$$

that is

$$\sum_{n=0}^{N-1} t_{n,i} t_{n,j} = \begin{cases} \varsigma & i = j \\ 0 & i \neq j \end{cases} \tag{5-16}$$

The transformation $\mathbf{T}$ is orthonormal if

$$\sum_{n=0}^{N-1} t_{n,i} t_{n,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{5-17}$$

The orthonormal transformation $\mathbf{T}$ has the following properties [7]:

- The determinant of $\mathbf{T}$ is 1, i.e.,

  $$det(\mathbf{T}) = 1$$

  where $det(\ )$ is the determinant operation.

- Let $\mathbf{y}_1 = \mathbf{T}\mathbf{x}_1$ and $\mathbf{y}_2 = \mathbf{T}\mathbf{x}_2$, then

  $$\|\mathbf{y}_1 - \mathbf{y}_2\| = \|\mathbf{x}_1 - \mathbf{x}_2\|$$

  i.e. distances are preserved by the transformation. Let $\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$ and $\mathbf{y} = \mathbf{y}_1 - \mathbf{y}_2$ such that $\mathbf{y} = \mathbf{T}\mathbf{x}$,

  $$\|\mathbf{y}\|^2 = \|\mathbf{x}^* \mathbf{T}^* \mathbf{T}\mathbf{x}\| = \|\mathbf{x}\|^2$$
  $$\|\mathbf{y}_1 - \mathbf{y}_2\|^2 = \|\mathbf{x}_1 - \mathbf{x}_2\|^2.$$

- The transformation does not change the determinant of the auto-correlation matrix of the input vector $\mathbf{x}$. Say $\mathbf{y} = \mathbf{T}\mathbf{x}$, then

$$\mathbf{R}_{\mathbf{y}} = E\left[\mathbf{y}\mathbf{y}^t\right] = \mathbf{T}E\left[\mathbf{x}\mathbf{x}^t\right]\mathbf{T}^t = \mathbf{T}\mathbf{R}_x\mathbf{T}^t .$$

Taking determinants on both sides, we get

$$det\left(\mathbf{R}_{\mathbf{y}}\right) = det\left(\mathbf{T}\mathbf{R}_{\mathbf{x}}\mathbf{T}^t\right).$$

Using the determinant property

$$det\left(\mathbf{A}\mathbf{B}\right) = det\left(\mathbf{A}\right)det\left(\mathbf{B}\right)$$

where $\mathbf{A}$ and $\mathbf{B}$ are matrices and substituting $det\left(\mathbf{T}\right) = det\left(\mathbf{T}^t\right) = 1$, we have

$$\begin{aligned} det\left(\mathbf{R}_{\mathbf{y}}\right) &= det\left(\mathbf{T}\right)det\left(\mathbf{R}_{\mathbf{x}}\right)det\left(\mathbf{T}^t\right) \\ det\left(\mathbf{R}_{\mathbf{y}}\right) &= det\left(\mathbf{R}_{\mathbf{x}}\right) \end{aligned} .$$

Thus, the determinant of the auto-correlation matrix of the input vector $\mathbf{x}$ is the same as the determinant of the auto-correlation matrix of the transformed vector $\mathbf{y}$.

One of the most widely used transforms in speech and image compression applications is the *discrete-cosine transform* (DCT). The DCT is an orthogonal and signal independent transform. The DCT of a discrete function $\mathbf{x}$ defined in (5-11) is [26]

$$y_k = \frac{2c_k}{N}\sum_{i=0}^{N-1} x_i \cos\left[\frac{(2i+1)k\pi}{2N}\right]; \quad k = 0,\ 1,\ \ldots,\ N-1 \tag{5-18}$$

and the inverse transform is

$$x_i = \sum_{k=0}^{N-1} c_k y_k \cos\left[\frac{(2i+1)k\pi}{2N}\right]; \quad i = 0,\ 1,\ \ldots,\ N-1 \tag{5-19}$$

where

$$c_k = \frac{1}{\sqrt{2}} = 0.7071 \quad for \quad k = 0$$
$$= 1 \qquad\qquad for \quad k = 1, 2, \ldots, N-1 \tag{5-20}$$

Writing (5-18) in matrix form and substituting for c_k as defined in (5-20), we have

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{bmatrix} = \left(\frac{2}{N}\right) \begin{bmatrix} 0.7071 & 0.7071 & \cdots & 0.7071 \\ \cos\left[\frac{\pi}{2N}\right] & \cos\left[\frac{3\pi}{2N}\right] & \cdots & \cos\left[\frac{(2N-1)\pi}{2N}\right] \\ \cos\left[\frac{2\pi}{2N}\right] & \cos\left[\frac{6\pi}{2N}\right] & \cdots & \cos\left[\frac{2(2N-1)\pi}{2N}\right] \\ \vdots & \vdots & \cdots & \vdots \\ \cos\left[\frac{(N-1)\pi}{2N}\right] & \cos\left[\frac{3(N-1)\pi}{2N}\right] & \cdots & \cos\left[\frac{(2N-1)(N-1)\pi}{2N}\right] \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix} \tag{5-21}$$

equivalently

$$\mathbf{y} = \mathbf{M}_{DCT}\,\mathbf{x} \tag{5-22}$$

where $\mathbf{y}$ is the $N\,x\,1$ transformed vector , $\mathbf{M}_{DCT}$ is the $N\,x\,N$ DCT matrix.

We illustrate the quantization of excitation vectors using the DCT with an example. Let $\tilde{\mathbf{e}}_{n,M}$ be the excitation vector obtained by inverse filtering $\mathbf{s}_{n,M}$ depicted in Figure 5-4a. The vector $\tilde{\mathbf{e}}_{n,M}$ is divided into four sub-frames $\tilde{\mathbf{e}}_{n,M_k}$, $k = 0,1,\ldots,3$ as defined in (5-10). Each sub-vector $\tilde{\mathbf{e}}_{n,M_k}$ of length $40\,x\,1$ is transformed to $\mathbf{y}_{n,M_k}$ of length $40\,x\,1$ using the DCT matrix $\mathbf{M}_{DCT}$ of size $40\,x\,40$ such that

$$\mathbf{y}_{n,M_k} = \mathbf{M}_{DCT}\,\tilde{\mathbf{e}}_{n,M_k} \tag{5-23}$$

Figure 5-26a illustrates the transformed vector $\mathbf{y}_{n,M_1}$ corresponding to $\tilde{\mathbf{e}}_{n,M_1}$. Figure 5-26b illustrates the energy of $\mathbf{y}_{n,M_1}$ in the time domain. From Figure 5-26b, we see that most of the energy is present in only a few of the transformed coefficients. So, $\mathbf{y}_{n,M_1}$ can be approximated by the first few samples,

say G, associated with the highest energy and the rest is then assumed to be zero. Let the approximated $\mathbf{y}_{n,M_1}$ be $\widetilde{\mathbf{y}}_{n,M_1}$ as illustrated in Figure 5-26c. The gain values associated with $\widetilde{\mathbf{y}}_{n,M_1}$ are quantized using an 8-bit scalar quantizer. Let $\hat{\mathbf{y}}_{n,M_1}$ denote the quantized $\widetilde{\mathbf{y}}_{n,M_1}$. The quantized gain values along with the corresponding indices are transmitted over the channel. At the decoder end, the reconstructed excitation vector $\hat{\mathbf{e}}_{n,M}$ is obtained using the inverse DCT matrix $\mathbf{M}_{DCT}^{-1}$ such that

$$\hat{\mathbf{e}}_{n,M} = \left[\hat{\mathbf{e}}_{n,M_0}^T \quad \hat{\mathbf{e}}_{n,M_1}^T \quad \ldots \quad \hat{\mathbf{e}}_{n,M_{K-1}}^T\right]^T, \; k = 0,1,\ldots,3 \tag{5-24}$$

where

$$\hat{\mathbf{e}}_{n,M_k} = \mathbf{M}_{DCT}^{-1}\,\hat{\mathbf{y}}_{n,M_k} \tag{5-25}$$

The reconstructed speech $\hat{\mathbf{s}}_{n,M}$ is then obtained by filtering $\hat{\mathbf{e}}_{n,M}$ with the LSF_{q_2} synthesis filter. Note that $\mathbf{M}_{DCT}$ and $\mathbf{M}_{DCT}^{-1}$ are computed only once, since the matrices are signal independent. The approximation of $\mathbf{y}_{n,M_k}$ to $\widetilde{\mathbf{y}}_{n,M_k}$ reduces the bit-rate at the expense of the quality of the reconstructed signal $\hat{\mathbf{s}}_{n,M}$. The bit allocation for a DCT based speech coder is tabulated in Table 5-8. Since the length of each sub-frame $\widetilde{\mathbf{e}}_{n,M_k}$ is 40, to represent each index $i \in [1,2,\ldots,40]$ we need 6 bits.

Table 5-8 Bit Allocation for DCT Based Speech Coder.

parameter	bits/sub-frame	bits/frame
LSF_2 coefficients		27
excitation gain indices	$6G$	$24G$
Excitation gains	$8G$	$32G$
Total bits T	$14G$	$27+56G$

The bit-rate (bits/sec) denoted by R is defined as

$$R = {}^{T F_s}\!/_{L} \tag{5-26}$$

where T is the total number of bits used to represent the transmission parameters, L is the frame size in samples, and F_s is the sampling frequency in Hz. Here, L =160 samples, F_s =8000 Hz. The SNR_{seg} obtained for various values of G is tabulated in Table 5-9.

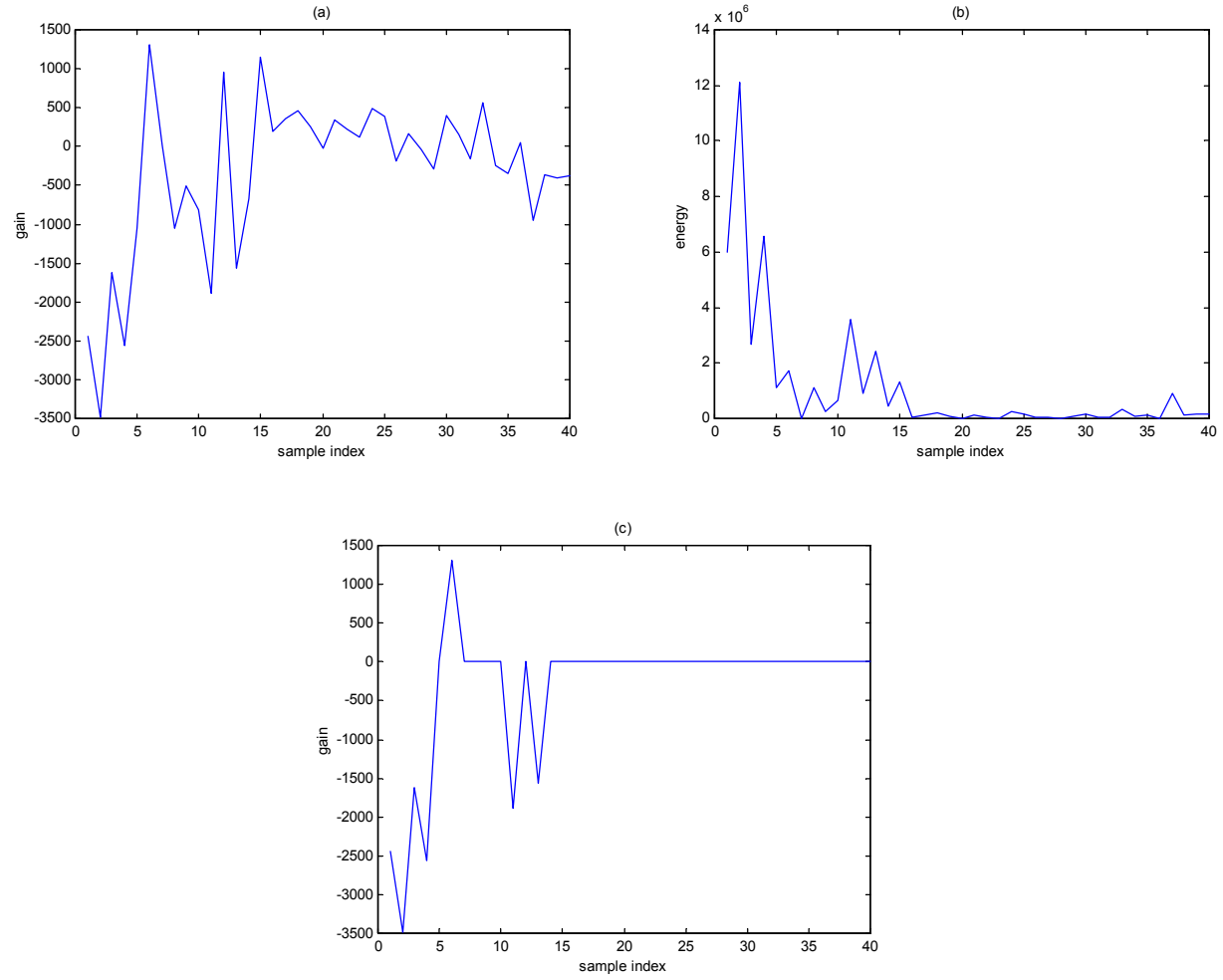

Figure 5-26 Transformed vector $\mathbf{y}_{n,M_1}$ representing a) gain b) energy in time domain c) $\tilde{\mathbf{y}}_{n,M_1}$.

Table 5-9 Performance Evaluation of DCT Based Speech Coder with changing G.

G	R kbps	$\approx SNR_{seg}$ dB
7	20.95	4 dB
10	29.35	6 dB
20	57.35	11 dB

In spite of reduced computational complexity, using the DCT for quantization of the excitation vectors resulted in high bit-rates. So, we are still looking for a computationally fast algorithm, with good performance at acceptable bit-rates. Transform coding is a form of scalar quantization, and removes the computational complexity of training larger VQ codebooks at the cost of an increase in the bit-rate. On the other hand, VQ exploits linear and non-linear dependence among the vectors to be quantized, but it is complex compared to transform coding. These conclusions lead to the use of *Vector Sum Codebooks* (VSC) to quantize the excitation signals in VSECLP as will be described in the next chapter.

Chapter 6 VSE Cascaded LP

The inclusion of LTP makes the system very sensitive to channel errors [5]. In order to take advange of not using the LTP, we do not use a LTP in our proposed speech coder, *Vector Sum Excited Cascade Linear Prediction (VSECLP)* coder. We intend to compensate for the loss of LTP information with a better representation of the excitation signal. Another important aspect of VSECLP is that we implement forward linear prediction using five cascaded 2^{nd} order LSF sections in place of the conventional 10^{th} order AR filter. The concept of vector sum codebooks is introduced in Section 6.1. The operation of VSECLP is described in Section 6.2. The VSECLP coder is tested on the TIMIT testing database comprising of 191,000 frames each of 20 ms duration. In Section 6.3, the performance of VSECLP coder in terms of quality and bit-rate is compared to that of the DCT based speech coder described in Section 5.3.

6.1 Vector Sum Codebooks

In VQ, the input vector $\mathbf{s}_n$ is quantized to $\hat{\mathbf{s}}_n$, where $\hat{\mathbf{s}}_n$ is the code vector - from a finite codebook - that gives the minimum mean square error between $\mathbf{s}_n$ and $\hat{\mathbf{s}}_n$. Thus, the performance of a quantizer depends on how well the code vectors represent the signal to be quantized. Increasing number of bits increases the number of code vectors. These code vectors can be further utilized for the better representation of the signal. However, adding one bit per code vector doubles the codebook size, resulting in an increase in the computational cost and storage requirements for the codebook search procedure. For example, the shape-gain codebook, described in Section 4.3, to quantize an excitation sub-frame is of size 40×2^{40}. Structured codebooks allow us to benefit from larger codebooks while maintaining tolerable computational as well as storage complexity. Some examples of structured codebooks are the tree-search codebook, the split codebook, the vector-sum codebook and the multi-stage codebook.

6.1.1 Vector Sum Excitation Codebooks

Gerson and Jasiuk introduced a search procedure [23] which reduces the excitation search complexity procedure. In *vector sum excitation codebook* (VSC), instead of generating the code vectors

independently of each other, either randomly or by a design procedure, they are generated from a much smaller set of basis vectors. In the VSC design procedure, B excitation basis vectors are generated. The code vectors are constructed by taking all possible binary linear combinations of those B basis vectors. This results in 2^B code vectors. The code vectors are arranged such that each code vector differs from its preceding code vector in only one bit position. This sequencing reduces the codebook search as described in Section 6.1.4.

6.1.2 Vector Sum Codebook Structure

The speech samples are divided into blocks of L samples each. Each block of data is called a frame. Let $\mathbf{s}_{n,M}$ denote the M^{th} speech frame. Then each frame of speech is divided into K sub-frames $\mathbf{s}_{n,M_k}$, $1 \le k \le K$ such that

$$\mathbf{s}_{n,M} = \begin{bmatrix} \mathbf{s}_{n,M_1}^T & \mathbf{s}_{n,M_2}^T & \cdots & \mathbf{s}_{n,M_k}^T \end{bmatrix}^T \tag{6-1}$$

where each sub-frame $\mathbf{s}_{n,M_k}$ is defined as

$$\mathbf{s}_{n,M_k} = \begin{bmatrix} \mathbf{s}_{j,M} & \mathbf{s}_{j+1,M} & \cdots & \mathbf{s}_{j+N-1,M} \end{bmatrix}^T; \quad N = L/K,\ j = Nk \tag{6-2}$$

Let $\mathbf{b}_m$ be the m^{th} basis vector from a given set of B basis vectors

$$\mathbf{b}_m = \begin{bmatrix} b_{0,m} & b_{1,m} & \cdots & b_{N-1,m} \end{bmatrix}^T \tag{6-3}$$

Then the i^{th} code vector $\mathbf{c}_i$, defined as

$$\mathbf{c}_i = \begin{bmatrix} c_{0,i} & c_{1,i} & \cdots & c_{N-1,i} \end{bmatrix}^T \tag{6-4}$$

is constructed as

$$c_{n,i} = \sum_{m=1}^{B} \theta_{im} b_{n,m} \tag{6-5}$$

where $0 \le i \le 2^B - 1$ and $0 \le n \le N-1$,

$$\begin{aligned} \theta_{im} &= +1 \quad \textit{if bit } m \textit{ of } i^{th} \textit{ codevector is } 1 \\ \theta_{im} &= -1 \quad \textit{if bit } m \textit{ of } i^{th} \textit{ codevector is } 0 \end{aligned} \tag{6-6}$$

The code words are arranged such that each code word differs from its successive code vector in only one bit position. This sequencing can be accomplished using a binary Gray code.

6.1.3 Excitation Codeword Selection

In order to perform the VSC searches, the zero state response $\tilde{\mathbf{b}}_m$ of each basis vector $\mathbf{b}_m$ to $H_s(z)$ as defined in (2-17) is obtained. Using (6-5), each filtered code vector $\tilde{\mathbf{c}}_i$ can be expressed as

$$\tilde{c}_{n,i} = \sum_{m=1}^{B} \theta_{im} \tilde{b}_{n,m} \tag{6-7}$$

The best excitation code vectors are searched for every sub-frame $\mathbf{s}_{n,M_k}$, $1 \le k \le K$. Let us define

$$v_i = \sum_{n=0}^{N-1} \tilde{c}_{n,i} p_n = \tilde{\mathbf{c}}_i^T \mathbf{p}_{n,M_k}, \quad 1 \le k \le 4 \tag{6-8}$$

$$g_i = \sum_{n=0}^{N-1} \left(\tilde{c}_{n,i}\right)^2 \tag{6-9}$$

where $\mathbf{p}_{n,M_k}$ is the input speech for the k^{th} sub-frame of M^{th} frame, with the zero-input response of $H_s(z)$ subtracted out. The code vector $\tilde{\mathbf{c}}_{i^*}$ that maximizes

$$i^* = \arg \min_i \left[\frac{(v_i)^2}{g_i} \right], \; 0 \le i \le 2^B - 1 \tag{6-10}$$

is chosen.

6.1.4 Reducing the VSC search complexity

Since, we have arranged the code vectors such that each code vector differs from its preceding code vector in only one bit position, the computation of (6-8) and (6-9) can be simplified as follows. Let us define,

$$\begin{aligned} r_m &= 2\sum_{n=0}^{N-1} \tilde{b}_{n,m} p_n \qquad 1 \le m \le B \\ d_{mj} &= 4\sum_{n=0}^{N-1} \tilde{b}_{n,m} \tilde{b}_{n,j} \qquad 1 \le m \le j \le B \end{aligned} \tag{6-11}$$

Using the above definitions, v_i in (6-8), and g_i in (6-9) are redefined as

$$v_i = 0.5\sum_{m=1}^{B} \theta_{im} r_m \tag{6-12}$$

$$g_i = 0.5\sum_{j=2}^{B}\sum_{m=1}^{j-1} \theta_{im}\theta_{ij} d_{mj} + 0.25\sum_{j=1}^{B} d_{jj} \tag{6-13}$$

Code word $\mathbf{c}_{i+1}$ differs from codeword $\mathbf{c}_i$ in only one bit position, say position k, such that $\theta_{ik} = -\theta_{(i+1)k}$ and $\theta_{il} = \theta_{(i+1)l}$ for $k \ne l$. Using this property, the parameters v_i in (6-12) and g_i in (6-13) can be return recursively as

$$\begin{aligned} v_{i+1} &= v_i + \theta_{(i+1)k} r_k \\ g_{i+1} &= g_i + \sum_{j=1}^{k-1} \theta_{(i+1)j}\theta_{(i+1)k} d_{jk} + \sum_{j=k+1}^{B} \theta_{(i+1)j}\theta_{(i+1)k} d_{kj} \end{aligned} \tag{6-14}$$

Computing v_i, $0 \le i \le 2^B - 1$ as defined in (6-8) requires $2^B N$ multiplications and $2^B(N-1)$ additions, whereas it requires only $N + 2^B - 1$ multiplications and $(N-1) + 2^B - 1$ additions. So, by organizing the codebook, such that each code vector differs from its preceding codevector in only one bit, a much simpler search can be used. Note that half of the code vectors in the codebook are

complementary code vectors. A code vector and its complementary code vector result in the same value for (6-10). So, only half of the code vectors are searched for the $\widetilde{\mathbf{c}}_i$ that maximizes (6-10). If the sign of v_i is positive then $\mathbf{c}_i$ is the best code vector, otherwise the code vector associated with the ones complement of i is chosen as the best code vector. After choosing the best excitation vector, the excitation gain is chosen from the excitation gain codebook, as will be described in Section 6.1.6.

6.1.5 Multiple VSC and Orthogonalization

The concept of VSC can be extended to multiple codebooks, say M codebooks and gains are jointly optimized at each stage. Initially, the first codebook is searched for the best filtered code vector, say $\widetilde{\mathbf{c}}_{I_1}$ where I_1 is represents the I^{th} code vector in first codebook. Then, the filtered code vectors in the second codebook are effectively orthonormalized to $\widetilde{\mathbf{c}}_{I_1}$. In practice, this task is accomplished by orthonormalizing the filtered basis vectors representing the second codebook to $\widetilde{\mathbf{c}}_{I_1}$. The filtered code vectors of second codebook are computed from the orthonormalized filtered basis vectors obtained using (6-7). Both these orthonormalizing methods are equivalent. However, orthonormalizing B basis vectors is computationally simpler than orthonormalizing 2^B code vectors. The orthonormalization process is accomplished by using Gram-Schmidt Orthogonalization procedure [5]. This ensures that the second codebook vectors do not cover the vector space that has already been covered by the first codebook vectors. The modified second codebook is now searched for the best code vector $\widetilde{\mathbf{c}}_{I_2}$ for the signal $\mathbf{p}_{n,M_k}$. Similarly, all the code vectors in the third stage are orthogonalized to both the code vectors $\widetilde{\mathbf{c}}_{I_1}$ and $\widetilde{\mathbf{c}}_{I_2}$ [24]. We illustrate the procedure of searching the excitation code vectors for a frame of speech in Section 6.3.

6.1.6 Excitation Gain Quantization

Say, there are two VSC, then the total quantized speech $\mathbf{p}_{n,M_k}$, denoted by $\hat{\mathbf{p}}_{n,M_k}$ of size $N \times 1$ is given by

$$\hat{\mathbf{p}}_{n,M_k} = \gamma_{1q}\,\widetilde{\mathbf{c}}_{I_1} + \gamma_{2_q}\,\widetilde{\mathbf{c}}_{I_2} \tag{6-15}$$

where $\widetilde{\mathbf{c}}_{I_1}$ and $\widetilde{\mathbf{c}}_{I_2}$ are the best excitation code vectors selected from the first and second codebooks respectively. The total excitation vector $\hat{\mathbf{e}}_{n,M_k}$ is given by

$$\hat{\mathbf{e}}_{n,M_k} = \gamma_{1q}\,\mathbf{c}_{I_1} + \gamma_{2_q}\,\mathbf{c}_{I_2} \tag{6-16}$$

where $\mathbf{c}_{I_1}$ is the excitation (unfiltered) code vector at index I in the first VSC. The parameter $\mathbf{c}_{I_2}$ in (6-16), is obtained as follows. Once the indices I_1 and I_2 are obtained, the basis vectors representing the second codebook are orthonormalized to $\mathbf{c}_{I_1}$ and then $\mathbf{c}_{I_2}$ is constructed explicitly from the now orthonormalized basis vectors using (6-7). The parameters γ_{1_q} and γ_{2_q} are the excitation gains associated with $\widetilde{\mathbf{c}}_{I_1}$ and $\widetilde{\mathbf{c}}_{I_2}$ respectively. Figure 6-1 depicts the excitation code word search procedure. In the multiple stage VSC, the best code vectors $\widetilde{\mathbf{c}}_{I_k}$, $k=1,2$ from each stage are computed first, then γ_{1_q} and γ_{2_q} pair that results in minimum MSE as defined in (4-12) between $\mathbf{p}_{n,M_k}$ and $\hat{\mathbf{p}}_{n,M_k}$ is selected from the excitation gain codebook. The excitation gain is also quantized for every sub-frame.

The quantization of excitation gains consists of two stages. In the first stage, the average speech energy E_i, of each i^{th} frame, defined as

$$E_i = \mathbf{s}_i^T \mathbf{s}_i \tag{6-17}$$

is scalar quantized. Let $\hat{E}_i$ denote the quantized version of the frame energy E_i. Then, the best excitation vectors and excitation gains for each sub-frame of the i^{th} frame are computed. For a sub-frame of the i^{th} frame, the energy in each of its best excitation vectors is given by

$$R_k = \mathbf{c}_{I_k}^T \mathbf{c}_{I_k} \quad k = 1, 2 \tag{6-18}$$

The residual energy RS in a given sub-frame of the i^{th} frame is a function of N, $\hat{E}_i$, and the prediction gain P_G of the cascaded synthesis filter. It can be approximated as [23]

$$RS = N\hat{E}_i P_G \tag{6-19}$$

The prediction gain P_G of the cascaded synthesis filter is given by

$$P_G = \prod_{i=1}^{M} P_{G_i} \tag{6-20}$$

where M is the number of SOS, P_{G_i} is the prediction gain of the i^{th} second order filter given by

$$P_{G_i} = \prod_{k=1}^{2} \left(1 - r_k^2\right) \tag{6-21}$$

where r_k is the k^{th} reflection coefficient of the i^{th} second order filter.

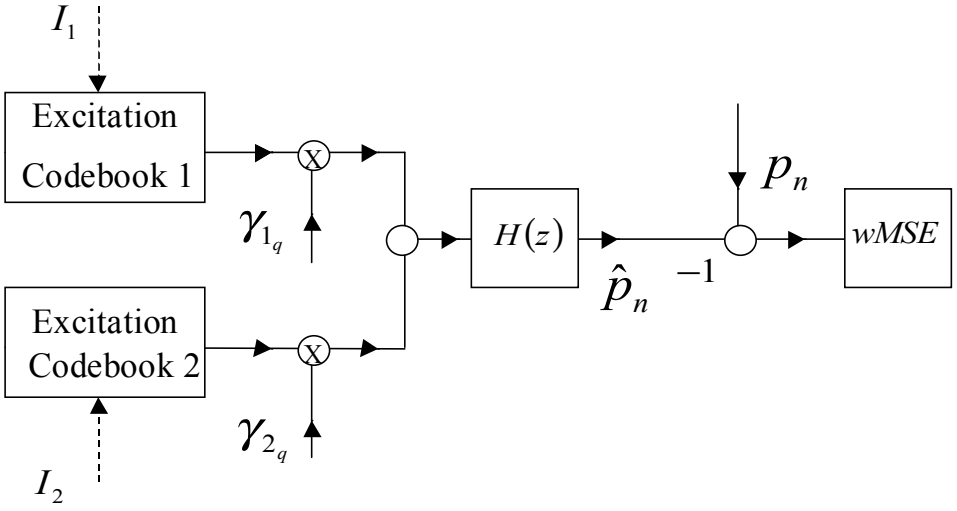

Figure 6-1 Excitation Codeword Search in Vector Sum Excitation Codebooks.

Let R be the approximate excitation energy of the total excitation signal in (6-16)

$$R = (GS)(RS) \tag{6-22}$$

where GS, called the energy offset, is a coded parameter which refines the estimated value of RS. The excitation gains γ_{1_q} and γ_{2_q} are defined as

$$\gamma_{1_q} = \sqrt{\frac{RS\ GS\ P_1}{R_1}}$$
$$\gamma_{2_q} = \sqrt{\frac{RS\ GS\ P_2}{R_2}} \tag{6-23}$$

The parameter $P_k, k = 1,2$ in (6-23) is defined as the approximate energy contribution of the code vector selected from the k^{th} codebook as a fraction of the total excitation energy at a sub-frame. Since, $P_k, k = 1,2$ is defined as a fraction of the total energy, the sum of the fractions is one i.e.,

$$0 \le P_k \le 1,\ \ k = 1, 2$$
$$P_1 + P_2 \le 1 \tag{6-24}$$

Using (6-24), (6-23) can be rewritten as

$$\gamma_{1_q} = \sqrt{\frac{RS\ GS\ P_1}{R_1}}$$
$$\gamma_{2_q} = \sqrt{\frac{RS\ GS\ (1-P_1)}{R_2}} \tag{6-25}$$

The parameters $GS - P_1 - P_2$ are coded using an 8 bit VQ. So, we choose the best $GS - P_1 - P_2$ code that results in minimum MSE as defined in (4-12) and then compute γ_{1_q} and γ_{2_q} from $GS - P_1 - P_2$ using (6-23).

6.2 VSECLP Coder

In Vector Sum Excited Cascade Linear Prediction (VSECLP) coding, we do not use a long term predictor (LTP). We intend to compensate for the loss of LTP information with a better representation of the excitation signal. In the absence of LTP, the excitation signal consists of more speech information than would otherwise be the case. This requires a larger excitation codebook to represent the excitation signal efficiently. However, larger codebooks increase the optimization and search complexity. To overcome these problems, we use vector sum codebooks for the excitation signals as described in Section 6.1. The excitation vectors and the corresponding excitation gains are

coded for each sub-frame of 5 ms duration. We use 3 VSC codebooks of size 11 $x\,2^{11}$ each. Thus, we are using 33 bits to represent the excitation signal per sub-frame. The excitation gains are coded using 8 bit $GS-P_1-P_2-P_3$ VQ. The speech energy is coded using 5 bit scalar quantizer.

The LSF_2 parameters for each analysis frame in the testing are estimated using the simplified DLSF as described in Section 3.3. We choose SVQ(2,4,4)SRI codebook with (9-9-9) bit configuration as obtained in Section 5.2 to quantize the LSF_2.

6.3 Simulation Results

The excitation vectors $\widetilde{\mathbf{e}}_{n,M}$ as defined in (5-9) for the testing database are obtained as illustrated in Figure 5-22 using SVQ(2,4,4)SRI. The $\widetilde{\mathbf{e}}_{n,M}$ is divided into 4 sub-frames $\widetilde{\mathbf{e}}_{n,M_k}$ as defined in (5-10). In VSC, each code vector is constructed as a binary linear combination of *B* basis vectors as described in Section 6.1. An excitation data matrix $\widetilde{\mathbf{E}}$ of size $N\,x\,V$ defined as

$$\widetilde{\mathbf{E}}=\begin{bmatrix} \widetilde{e}_{0,0} & \widetilde{e}_{0,1} & \cdots & \widetilde{e}_{0,V-1} \\ \widetilde{e}_{1,0} & \widetilde{e}_{1,1} & \cdots & \widetilde{e}_{1,V-1} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{e}_{N-1,0} & \widetilde{e}_{N-1,1} & \cdots & \widetilde{e}_{N-1,V-1} \end{bmatrix} \tag{6-26}$$

is formed by randomly picking V of the excitation sub-frames from the training database. The auto-correlation matrix $\mathbf{R}_{\widetilde{\mathbf{E}}}$ is given by

$$\mathbf{R}_{\widetilde{\mathbf{E}}}=\widetilde{\mathbf{E}}\widetilde{\mathbf{E}}^{T} \tag{6-27}$$

The eigenvalues and eigenvectors of $\mathbf{R}_{\widetilde{\mathbf{E}}}$ are calculated. Each eigenvalue represents the signal strength in the direction of the corresponding eigenvector. The eigenvectors associated with the *B* largest eigenvalues are chosen as the basis vectors [27]. A VSQ codebook is formed from the basis vectors as explained in Section 6.1.2. Similarly, each of the 3 VSC codebooks of size 11 x 2^{11} is obtained using 11 DKLT basis vectors. The excitation code vectors for each sub-frame and the corresponding gains are quantized as described in Section 6.1.6. The bit allocation for VSECLP is tabulated in Table 6-1. The bit-rate R as defined in (5-26) is 9.6 kbps.

Table 6-1 Bit Allocation VSECLP Speech Coder.

parameter	bits/sub-frame	Bits/frame
LSF_2 coefficients		27
speech energy		5
excitation codes	11+11+10	128
excitation gains	8	32
Total bits T	40	192

All the VSC codebooks are searched for every excitation sub-frame $\tilde{\mathbf{e}}_{n,M_k}$. The procedure for quantizing an excitation frame $\tilde{\mathbf{e}}_{n,M}$ using three VSC is described with an example. Figure 6-2 a) illustrates the same speech frame $\mathbf{s}_{n,M}$ in Figure5.4a) that was used illustrate the performance of SVQ. The dashed lines are the speech sub-frame $\mathbf{s}_{n,M_k}$ boundaries. For the first speech sub-frame $\mathbf{s}_{n,M_1}$ as depicted in Figure 6-2 b), substituting $\mathbf{p}_{n,M_1} = \mathbf{s}_{n,M_1}$ in (6-8), the first VSC is searched for the best-filtered vector $\tilde{\mathbf{c}}_{I_1}$ that maximizes (6-10). Then each of the filtered basis vectors representing second VSQ codebook to orthonormalized to $\tilde{\mathbf{c}}_{I_1}$. The filtered code vectors of second codebook are computed from the orthonormalized filtered basis vectors obtained using (6-7). In order to see if the resultant code vectors are orthonormalized to $\tilde{\mathbf{c}}_{I_1}$ or not, we verified the orthogonality condition as defined in (5-16) between each $\tilde{\mathbf{c}}_{I_2}, 0 \le i \le 2^B - 1$ and $\tilde{\mathbf{c}}_{I_1}$. They satisfy the orthogonality condition. Thus, by orthonormalizing the basis vectors to $\tilde{\mathbf{c}}_{I_1}$, the resultant code vectors are also orthonormalized to $\tilde{\mathbf{c}}_{I_1}$. After the second VSC codebook is searched for the best filtered vector $\tilde{\mathbf{c}}_{I_2}$, the filtered basis vectors representing the third codebook are orthonormalized to both $\tilde{\mathbf{c}}_{I_1}$ and $\tilde{\mathbf{c}}_{I_2}$. The third VSC obtained from the orthonormalized basis vectors is searched for the best filtered vector $\tilde{\mathbf{c}}_{I_3}$. After finding all the three filtered vectors $\tilde{\mathbf{c}}_{I_1}$, $\tilde{\mathbf{c}}_{I_2}$, and $\tilde{\mathbf{c}}_{I_3}$ the $GS-P1-P2-P3$ codebook is searched for best $GS-P1-P2-P3$ code that results in the minimum MSE between $\mathbf{p}_{n,M_k}$ and $\hat{\mathbf{p}}_{n,M_k}$. The quantized excitation vector $\hat{\mathbf{e}}_{n,M_1}$ is given by

$$\hat{\mathbf{e}}_{n,M_1} = \gamma_{1_q}\mathbf{c}_{I_1} + \gamma_{2_q}\mathbf{c}_{I_2} + \gamma_{3_q}\mathbf{c}_{I_3} \tag{6-28}$$

The vector $\hat{\mathbf{e}}_{n,M_1}$ is zero padded with 120 zeros to make it equal to the frame length. Let $\hat{\mathbf{z}}_{n,M_1}$ be the zero-padded version of $\hat{\mathbf{e}}_{n,M_1}$. Figure 6-2 c) and Figure 6-2 d) illustrates $\hat{\mathbf{e}}_{n,M_1}$ and $\hat{\mathbf{z}}_{n,M_1}$ respectively. The zero-state response of $H_s(z)$ for input $\hat{\mathbf{z}}_{n,M_1}$ be abbreviated by ZS_1 and symbolized as $\hat{\mathbf{s}}_{n,M_1}$.

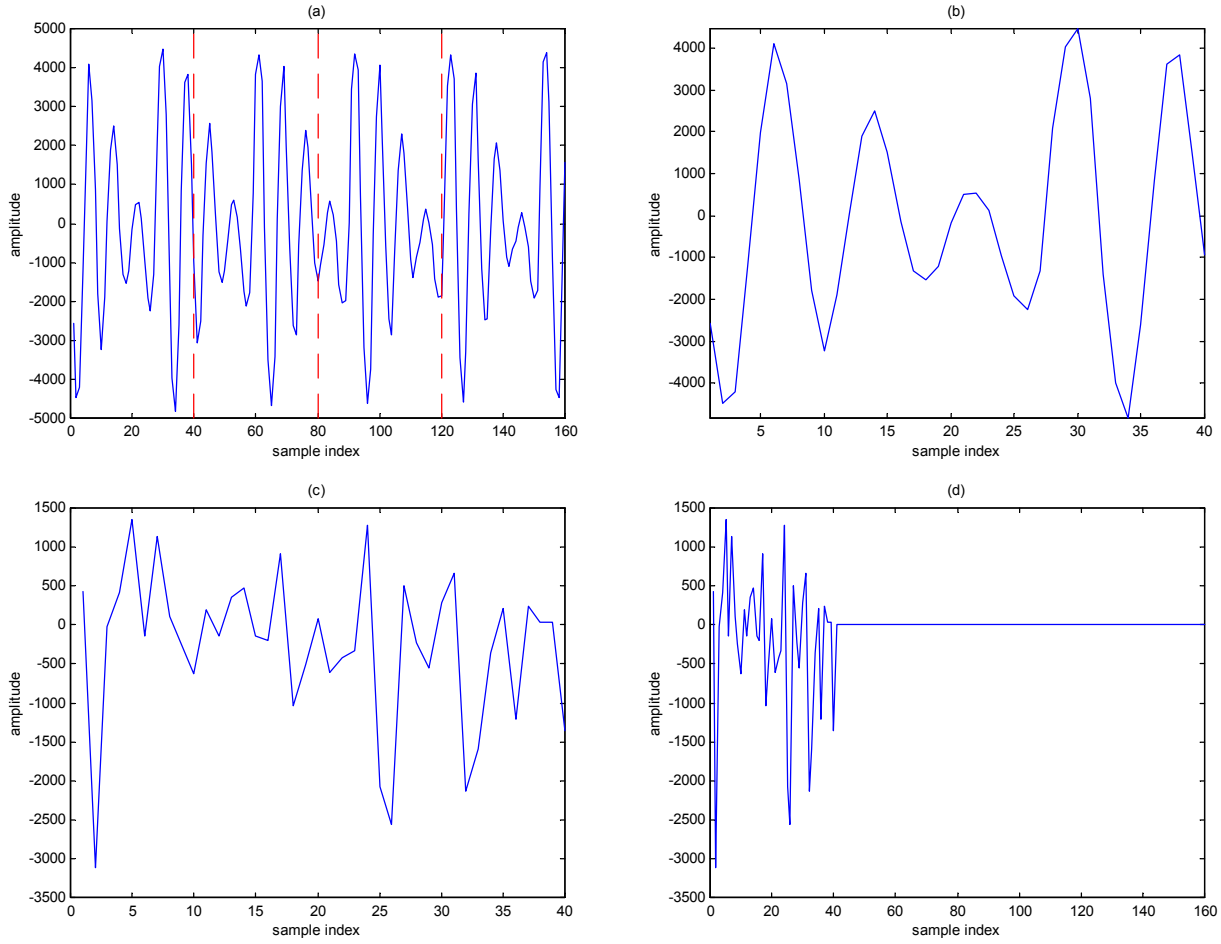

Figure 6-2 a) Speech Frame $\mathbf{s}_{n,M}$ b) First Speech Sub-Frame $\mathbf{s}_{n,M_1}$ c) Estimated Excitation for First Sub-Frame $\hat{\mathbf{e}}_{n,M_1}$ d) Zero-padded $\hat{\mathbf{e}}_{n,M_1}$ $\hat{\mathbf{z}}_{n,M_1}$.

The zero-state response illustrated in Figure 6-3 decays slowly and is almost zero (–0.001) at the end of the frame. The *SNR* defined in between $\mathbf{s}_{n,M_1}$ and $\hat{\mathbf{s}}_{n,M_1}$ in the first sub-frame is 28.1 dB. It can be observed from Figure 6-3 that the zero-state response also exists for samples 40-160, corresponding to the second, third and fourth sub-frames.

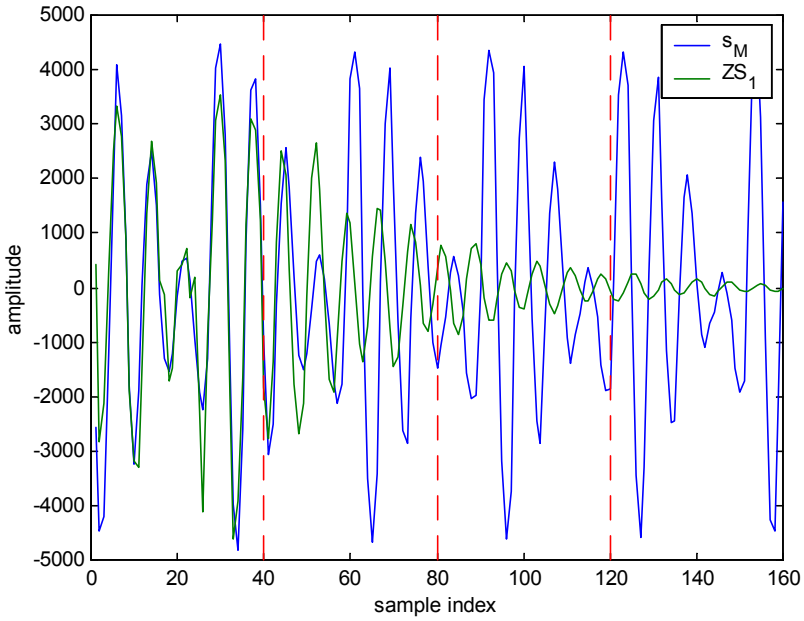

Figure 6-3 Zero-State Response of $H_s(z)$ for Input $\hat{\mathbf{z}}_{n,M_1}$.

This is indeed the zero-input response during the second, third and fourth sub-frames, since the input $\hat{\mathbf{z}}_{n,M_1}$ is evaluated only for the first sub-frame. In order to take the zero-input response into account, we subtract it from the corresponding speech sub-frame and evaluate the best excitation code vector for the difference speech signal. For example, for the second frame, we subtract the zero-input response during the second sub-frame corresponding to samples 40-79 of $\hat{\mathbf{s}}_{n,M_1}$ from $\mathbf{s}_{n,M_2}$, thus we have $\mathbf{p}_{n,M_2} = \mathbf{s}_{n,M_2} - \left(\hat{\mathbf{s}}_{n,M_1}\right)_{40}^{79}$. Substituting $\mathbf{p}_{n,M_2}$ in (6-8), the best filtered code vectors $\tilde{\mathbf{c}}_{I_i}, 1 \le i \le 3$ and the corresponding excitation estimate $\hat{\mathbf{e}}_{n,M_2}$ is calculated similar to that of $\hat{\mathbf{e}}_{n,M_1}$. The vector $\hat{\mathbf{e}}_{n,M_2}$ is zero padded with 120 zeros to make it equal to the frame length. Let $\hat{\mathbf{z}}_{n,M_2}$ be the zero-padded version of $\hat{\mathbf{e}}_{n,M_2}$. Again the zero-state response ZS_2 of $H_s(z)$ with input $\hat{\mathbf{z}}_{M_2}$ is obtained. Figure 6-4 illustrates the zero-state responses ZS_k, $1 \le k \le 4$ of $H_s(z)$ with input $\hat{\mathbf{z}}_{M_k}$, $1 \le k \le 4$. Note that each ZS_k of length 160, starts at the beginning of the corresponding sub-frame. From Figure 6-4, the zero-state responses ZS_1 and ZS_2 exists in the range 80-159. Similarly, for the third sub-frame, we subtract the zero-input response during the third frame corresponding to ZS_1 and ZS_2, i. e., $\mathbf{p}_{n,M_3} = \mathbf{s}_{n,M_3} - \left(\hat{\mathbf{s}}_{n,M_2}\right)_{40}^{79} - \left(\hat{\mathbf{s}}_{n,M_1}\right)_{80}^{119}$.

For the fourth sub-frame, we model $\mathbf{p}_{n,M_4} = \mathbf{s}_{n,M_3} - \left(\hat{\mathbf{s}}_{n,M_3}\right)_{40}^{79} - \left(\hat{\mathbf{s}}_{n,M_2}\right)_{80}^{119} - \left(\hat{\mathbf{s}}_{n,M_1}\right)_{120}^{159}$. The choice of appending $\hat{\mathbf{e}}_{n,M_k}$ with 120 zeros is to obtain the zero-input response of the following sub-frames. Experiments showed that the zero-input response during the k^{th} sub-frame contributed by the zero-state response of the preceding sub-frame ZS_{k-1} is significant compared to that of the other sub-frames. So, $\mathbf{p}_{n,M_k}$ can be approximated as

$$\mathbf{p}_{n,M_k} \approx \mathbf{s}_{n,M_k} - \hat{\mathbf{s}}_{n,M_{k-1}} \tag{6-29}$$

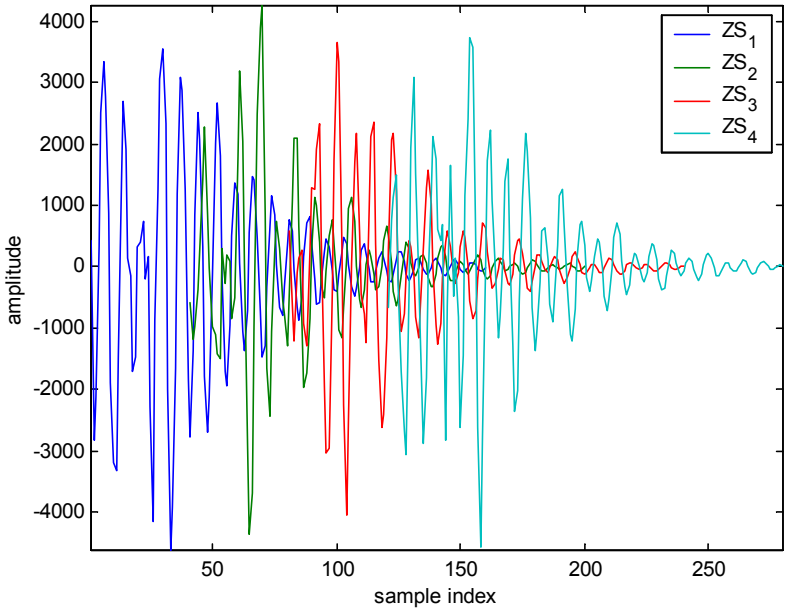

Figure 6-4 Zero-State Response ZS_k of $H_s(z)$ for input $\hat{\mathbf{z}}_{n,M_k}$.

The reconstructed M^{th} speech frame $\hat{\mathbf{s}}_{n,M}$ is

$$\hat{\mathbf{s}}_{n,M} = \left[\left(\hat{\mathbf{s}}_{n,M_1}^T\right)_1^{40} \quad \left(\hat{\mathbf{s}}_{n,M_2}^T\right)_1^{40} \quad \left(\hat{\mathbf{s}}_{n,M_3}^T\right)_1^{40} \quad \left(\hat{\mathbf{s}}_{n,M_4}^T\right)_1^{40}\right]^T \tag{6-30}$$

Figure 6-5 illustrates the original $\mathbf{s}_{n,M}$ and the reconstructed $\hat{\mathbf{s}}_{n,M}$ speech frames. The obtained SNR_{seg} is 7.85 dB.

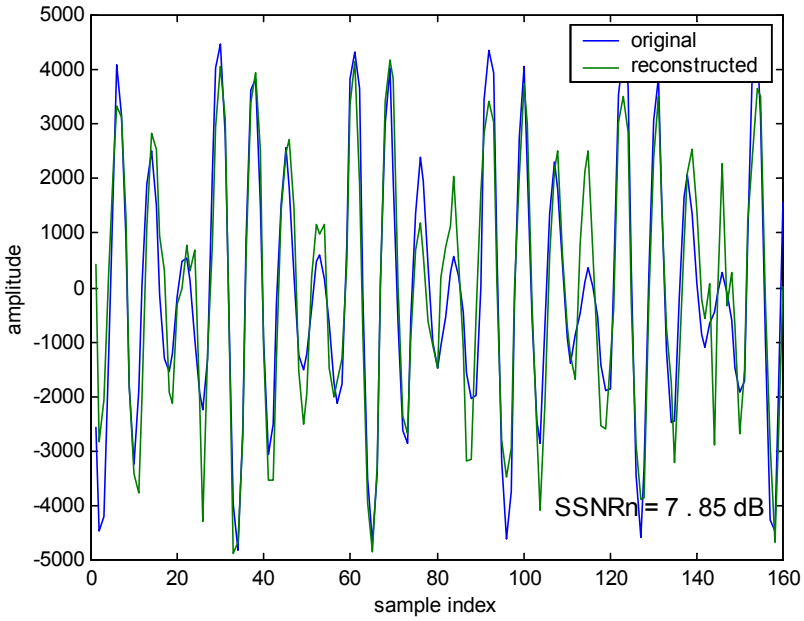

Figure 6-5 Original and Reconstructed Speech Frame.

We compared the performance of VSECLP with the DCT based speech coder using the TIMIT testing database. The DCT based speech coder resulted in SNR_{seg} of around 4 dB at 20.95 kbps, whereas the same SNR_{seg} was achieved at 9.6 kbps using VSECLP. Thus, the proposed speech coder produces the same performance as the DCT based speech coder, but at half bit-rate.

Chapter 7 Conclusions and Further Research

A new speech coder based on code excited linear prediction is proposed. The procedure is termed Vector Sum Excited Cascade Linear Prediction (VSECLP) coding. We did not implement any pitch prediction so that, consequently, each frame is transmitted independent from the rest of the frames. Thus, a dropped frame or corrupted frame will not affect the validity of other frames, and as a result the robustness to channel errors is increased. The implementation of the forward prediction with cascaded second order sections, avoids the matrix inversion problems encountered in higher order filters. The second order line spectral frequency (LSF_2) parameters estimated by the proposed simplified Direct Line Spectral Frequency (DLSF) adaptation algorithm are found to be closer to the true values than those estimated by the Cascaded Recursive Least Squares - Subsection algorithm. The simplified DLSF reduces the computational complexity of the DLSF algorithm by directly adapting the LSF_2 parameters modified by a scale factor.

Split vector quantization is used to quantize the LSF_2 parameters and vector sum codebooks are used to quantize the excitation vectors. The split vector quantization of the LSF_2 parameters with selective random initialization resulted in a segmental signal to noise ratio of 17.85 dB, whereas that with random initialization resulted in 7.85 dB. The use of selective random initialization proved to be better than random initialization by 10 dB. Using vector sum codebooks for quantization of the excitation signals reduced the optimization and search complexity as compared to that using shape-gain vector quantization. The quantization of the excitation vectors using the discrete cosine transform resulted in segmental signal to noise ratio of 4 dB at 20.95 kbps, whereas the same quality was obtained at 9.6 kbps using vector sum codebooks. In other words, VSECLP achieves the same quality as the discrete cosine transform based speech coder, but at half the bit-rate.

As an extension to this research, it would be interesting to see if a split vector quantization with different sub-codebook dimension and bit allocation would improve performance. In the absence of a pitch predictor, the excitation signal consists of more speech information than would otherwise be the case. A better representation of the excitation codebook is expected to improve the overall performance of the coder.

References

[1] S. Haykin, Adaptive Filter Theory, Englewood Cliffs, Prentice Hall, 1991.

[2] L.B Jackson and S.L. Wood, 'Linear Prediction in Cascade Form, IEEE Transactions on ASSP, vol. ASSP-26, pp. 519-528, December 1978.

[3] S. M. Kay, Modern Spectral Estimation, Prentice Hall, 1998.

[4] Gaguk Zakaria, Cascaded RLS with Subsection Adaptation, Ph.D. Dissertation, Department of Electrical and Computer Engineering, Virginia Tech, 2000.

[5] A. M. Kondoz, Digital Speech: Coding for Low Bit Rate Communications Systems, Willey, 1994.

[6] J. H. Chen, High-Quality 16KBPS Speech Coding with a One-way Delay Less Than 2 ms, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 1, pp. 453-456, 1990.

[7] Allen Gersho, Robert M Gray, Vector Quantization and Signal compression, Kulwer Academic, 1992.

[8] W. B. Klejin and J. Haagen, *Waveform Interpolation for Speech synthesis*, Speech Coding and Synthesis, W. B. Klejin and K. K. Paliwal editors, pp. 175-207, Elsevier, 1995.

[9] K. K. Paliwar and B. S. Atal, Advances in Speech Signal Processing, S. Furui and M. Sondhi editors, Markel Dekker, Inc., New York, 1992.

[10] R. P. Ramachandran et.al., A Two Codebook Formant for Robust Quantization of Line Spectra Frequencies, *IEEE Transactions on Speech and Audio Processing*, vol.3, no.3, pp. 157-167, May 1995.

[11] W. P. LeBlanc, B. Battacharya, S. A. Mahmoud, and V. Cuperman, Efficient Search and Design Procedure for Robust Multi-Sate VQ of LPC Parameters for 4kb/s Speech Coding, *IEEE Transactions on Speech and Audio Processing*, vol.1, no.4, pp. 373-385, October 1993.

[12] John G. Proakis, Dimitris and G. Manolakis, Digital Signal Processing – Principles, Algorithms, and Applications, Prentice Hall, 1996.

[13] John Makhoul, "Linear Prediction: A Tutoirial Review", *Proceedings of the IEEE*, vol. 63, No. 4, April 1975.

[14] G. Zakaria, and A. A. (Louis) Beex, "Cascade Recursive Least Squares with Subsection Adaptation for AR Parameter Estimation", Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, pp. 953 -956, 1998.

[15] F. K. Soong and B. H. Juang, "Line Spectrum Pair (LSP) and Speech Data Compression", IEEE Proceedings ICASSP, pp. 1.10.1-1.10.4, 1984.

[16] N. Sugamura and N. Farvardin, "Quantizer Design in LSP Speech Analysis-Synthesis", IEEE Journal on Selected Areas in Communications, vol. 6, no. 2, pp. 432-440, Feb.1988.

[17] P. Heldelin, P.Knagenhjelm, and M. Skoglund, Vector Quantization for Speech Transmission, Speech Coding and Synthesis, W. B. Klejin and K. K. Paliwal editors, pp. 311-345, Elsevier, 1995.

[18] J. S. Collura, A. McCree, and T.E. Tremain, Perceptually Based Distortion Measurements for Spectrum Quantization, 1995 IEEE Workshop on Speech Coding for Telecommunications, pp. 49-59, September 1995.

[19] Tianyi Jiang, Yongming Li, Hongyi Chen, A 1.44 kbps Vocoder Based on LSP, Proceedings of International Conference on Signal Processing, vol. 2, pp. 697-701.

[20] H. P. Kramer and M. V. Mathews, A Linear Coding for Transmitting a Set of Correlated Signals, IRE Transactions on Information Theory, IT-23, pp. 41-46, Sept. 1956.

[21] N. Ahmed, T. Natarjan, and K. R. Rao, Discrete Cosine Transform, IEEE Transactions on Computer, Jan. 1974, pp. 90-93.

[22] W. B. Klejin and J. Haagen, Quantization of LPC Parameters, Speech Coding and Synthesis, W. B. Klejin and K. K. Paliwal editors, pp. 433-466, Elsevier, 1995.

[23] Ira A. Gerson and Mark A. Jasiuk, Vector Sum Excited Linear Prediction (VSELP) Speech Coding at 8kbps, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 1, pp. 461-464, 1990.

[24] A. Nejat Ince, Digital Speech Processing, Speech Coding, Synthesis and Recognition, Kluwer Academic, 1992.

[25] The Darpa TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT), Training and Testing Data and Speech Header Software, NIST Speech Disc CD1-1.1, Oct. 1990.

[26] Wen-Hsiung Chen, C. Harrison Smith and S. C. Fralick, A Fast Computational Algorithm for the Discrete Cosine Transform, IEEE Transactions on Communications, vol. 25, No. 9, Sep. 1977.

[27] Charles W. Therrien, Discrete Random Signals and Statistical Signal Processing, Prentice Hall, 1992.

[28] John R. Deller, Jr, John G. Proakis, John H. L. Hansen, Discrete-Time Processing of Time Signals, Macmillan, 1993.

[29] B. S. Atal and J. R. Remde, A New Model of LPC Excitation for Producing Natural-Sounding Speech at Low Bit Rates, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 614 -617, May 1982.

[30] S. Singhal and B. S. Atal, Improving Performance of Multi-Pulse LPC Coders at Low Bit-rates, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 1, pp. 131-134, 1984.

[31] M. R. Schroeder and B. S. Atal, Code Excited Linear Prediction (CELP): High-Quality Speech at Very Low Bit-rates, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 937-940, 1985.

Vita

Visala Namburu was born in Chirala, India in 1977. She graduated with a Bachelor's degree in Electronics & Communications Engineering, from Andhra University, Visakapatnam, India in June 1999. She received her Masters degree in Electrical Engineering from Virginia Polytechnic Institute and State University, Blacksburg, Virginia in December 2001. Her research interests are primarily in the areas of digital signal processing and wireless communications.