Low-Delay Speech Coding at 16 kb/s and Below

by

Majid Foodeei

B. Eng.

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

Department of Electrical Engineering
McGill University
Montréal, Canada
May, 1991

Abstract

Development of *network quality* speech coders at 16 kb/s and below is an active research area. This thesis focuses on the study of low-delay Code Excited Linear Predictive (CELP) and tree coders. A 16 kb/s stochastic tree coder based on the (M,L) search algorithm suggested by Iyengar and Kabal and a low-delay CELP coder proposed by AT&T (CCITT 16 kb/s standardization candidate) are examined. The first goal is to compare and study the performance of the two coders. Second objective is to analyze the particular characteristics which make the two coders different from one another. The final goal is the improvement of the performance of the coders, particularly with a view of bringing down the bit rate below 16 kb/s.

When compared under similar conditions, the two coders showed comparable performance at 16 kb/s. The analysis of the components and particular characteristics of the tree and CELP coders provide new insight for future coders. Higher performance coder components such as prediction, gain adaptation, and residual signal quantization are needed. Issues in backward adaptive linear prediction analysis for both near and far-sample redundancy removal such as analysis methods, windowing, ill-conditioning, quantization noise effects and computational complexities are studied. Several new backward adaptive high-order methods show much better prediction gains than the previously reported ones. Other than a better high-order predictor for both coders, other suggestions to improve the performance of the coders include a new scheme of training of the excitation dictionary and better gain adaptation strategy for the tree coder. A hybrid "Tree-CELP" coder, taking the best components from the two archetypes is a good candidate to push coding rates below 16 kb/s.

Sommaire

Le codage du signal de parole de qualité "*network*" a fait récemment l'objet d'un effort de recherche considérable portant sur le développement de codeurs faible délai à 16 kb/s et moins. Ce mémoire traite du codage par "Code Excited Linear Predictive" (CELP) et du codage arborescent à faible délai. Un codeur arborescent basé sur l'algorithme de recherche (M,L) de Iyengar et Kabal et un codeur CELP à faible délai proposé par AT&T (candidat pour la standardisation à 16 kb/s au CCITT) sont utilisés. Nous poursuivons trois objectifs. Le premier est de comparer la performance des deux codeurs. Le second est d'analyser les caractéristiques qui les distinguent. Le but final est l'amélioration de la performance des codeurs, et en particulier, d'abaisser le débit sous le seuil de 16 kb/s.

Les deux codeurs donnent, dans les mêmes conditions, des performances similaires à 16 kb/s. L'analyse des composantes et des caractéristiques spécifiques aux codeurs arborescent et CELP facilitera le développement de nouveaux codeurs. Des éléments plus performants du codeur (prédiction, adaptation du gain, et quantification du signal résiduel) sont nécessaires. Plusieurs aspects de la prédiction linéaire par adaptation causale visant l'élimination des redondances sont étudiés: les méthodes d'analyse, le fenêtrage, le conditionnement, l'effet du bruit de quantification et la complexité des calculs. Plusieurs nouvelles méthodes causales d'ordre élevé presentent un gain de prédiction supérieur à celui obtenu avec les méthodes connues. L'amélioration du prédicteur n'est pas la seule suggestion: une nouvelle méthode d'entraînement du dictionnaire "d'excitation" et une meilleure stratégie d'adaptation du gain pour le codeur arborescent peuvent aussi améliorer la performance du codeur. Un codeur hybride "arborescent—CELP" possédant les meilleures caractéristiques des deux codeurs serait une bonne suggestion pour abaisser le débit sous le seuil de 16 kb/s.

Acknowledgements

My biggest debt of gratitude goes to my supervisor Dr. Peter Kabal for his guidance. Other people who had a significant impact on this thesis, both in motivating the ideas and the presentation, are Vasu Iyengar of Bell Northern Research (BNR) and Juin-Hwey Chen of AT&T. The major portion of the research was conducted at Institut National de la Recherche Scientifique (INRS) - Télécommunications. The laboratory and facilities provided by INRS were a great help to my research. The financial support provided by my supervisor from the National Science and Engineering Research Council (NSERC) and other research funds was appreciated.

I would like to give special thanks to my parents, my brother, and sisters for their consistent support as well as to Dana for her love and understanding. I am also grateful for the companionship provided by my many friends at INRS and McGill.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Digital representation of speech signals has many advantages. Ease of regenerative amplification for transmission over long distance and possibility of signal encryption are the most obvious ones. There is however the disadvantage of some digitization distortion. Lowering this distortion often means higher bit rates resulting in higher transmission bandwidth. Larger bandwidth has become available as a result of use of the new communication channels such as fiber optics. Nevertheless, for many reasons including the wide use of the old media, lower bit rate and bandwidth are still highly desirable. Therefore, the goal of reducing the bit rate of the digital speech coders with high quality or low distortion remains an active research area.

The measurement of speech quality is a difficult and long standing problem [1]. Other than objective measures, the subjective measure of Mean Opinion Score (MOS) has been commonly used. These measures are also used in this thesis. The MOS subjective measure is a subjective rating between 5 and 1 (5: excellent, 4: good, 3: fair, 2:poor, 1: unacceptable). A score of 4.0 is also referred to as *high-quality* or *near-transparent.* When the near-transparent quality is a necessary but not sufficient condition, the term *network quality* is used [1]. Low-delay and robustness to channel errors are examples of such possible additional conditions imposed by the communication network environment.

Other terms often used for scaling the speech quality include: • *Commentary* or *Broadcast quality* which is used when there is no perceivable noise for the wide-band speech (bandwidth of 0-7000 Hz), • *Toll-quality* or *Telephone quality*, narrow

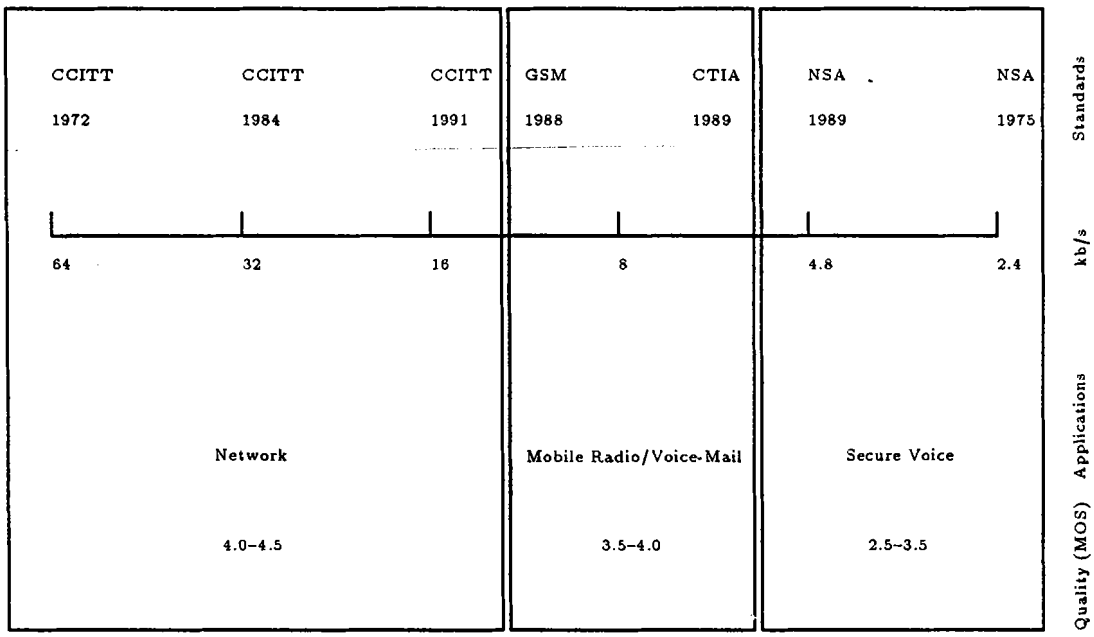


Fig. 1.1 Digital telephony standards [1]

bandwidth speech heard over telephone network with bandwidth of 0-3400 Hz, • *Communication quality*, with perceivable distortion but high intelligibility (MOS of 3.5), and • *Synthetic quality*, with unnatural yet highly intelligible characteristic. The subclass of wideband speech is not the focus of this work. For the narrowband speech coding, Fig. 1.1 summarizes the current state of digital telephony standards showing the bit rate, quality and applications. † The CCITT coding standardization for low-delay high-quality speech coding at 16 kb/s is under way. Other current goals are the achievement of transparent or near-transparent quality at 8 kb/s and robust communication quality at 4.8 kb/s and lower.

A summary of the CCITT standardization specification for the 16 kb/s low-delay coders is shown in Table 1.1. ‡ Note the objective low-delay of 2 ms and the required channel error robustness. In future, similar kinds of requirements can be expected

† CCITT:Consultative Committee for Telephone and Telegraph, GSM: Group Special Mobile, CTIA: Cellular Technology Industry Association, NSA: National Security Agency.

‡ The qdu is a quantization distortion unit where distortion is due to a single stage of 64 kb/s PCM coding with an "average" codec. To calibrate codecs on the qdu scale the CCITT uses the MNRU, a reference unit known with a qdu characteristic. The MNRU is specified in a CCITT recommendation. P_e is the channel probability of error. G.721 is CCITT standardization of 32 kb/s coders in 1984.

Parameter	CCITT Requirement	Objective
Coding Delay	$\leq$ 5 ms	$\leq$ 2 ms
$P_e = 0$ $P_e = 10^{-3}$ $P_e = 10^{-2}$	Distortion < 4 qdu Not worse than G.721 Not worse than G.721	
Tandeming for speech	3 asynchronous tandems with distortion < 14 qdu	Synchronous Tandems
Transmit Signaling Tones	DTMF	
Transmit Music		No annoying effects
Operate at lower rates		Graceful degradation
Complexity		As low as possible

Table 1.1 CCITT standardization - characteristics for low-delay 16 kb/s coders [1]

for coding at bit rates below 16 kb/s. Depending on the area of the application (Network, Mobile Radio/Voice Mail), the requirements can be somewhat different than the ones in Table 1.1. For example for Mobile Radio or in-building wireless applications, channel error rates can be more severe than the rates in Table 1.1. Low-delay coding at bit rates between 8 and 16 kb/s (medium bit rates) with transparent or near-transparent quality is the focus of this thesis. Possible additional requirement such as the ones in Table 1.1, are also considered.

A Low-Delay Code Excited Linear Predictive coder (LD-CELP) coder is the AT&T candidate for the 16 kb/s CCITT standardization [2, 3, 4, 5, 6, 7]. Iyengar and Kabal have suggested a Low-Delay Tree coder [8, 9, 10] (LD-TREE) based on the (M,L) algorithm and reported a performance quality equivalent to 7 bits/sample log-PCM at 16 kb/s. Under clear channel conditions ($P_e = 0.0$) both coders may be considered as potential candidate coders for the low-delay network-quality applications. Satisfactory performance quality, under channels with transmission error, is also reported for the LD-CELP. The performance of the two coders however has not been compared under the same conditions before. This is done when results of the

simulations of the two coders are compared in this work.

The studies and simulations of this work and the AT&T's development of the LD-CELP coder were concurrent. † The original early version of the LD-CELP described in Ref. [2] was used for simulations in this work. However the detailed algorithm description and subsequent modifications reported later [3–7] were also considered. ‡

The evolution of many speech coding techniques contributes to the structure and components used by the LD-CELP and LD-TREE. Any improvements leading to better future speech coders will benefit from the lessons of this evolution. The structure of conventional *Pulse Code Modulation* (PCM) with μ-law or A-law companding schemes is well known. First, the analog speech signal is passed through the band limiting filter (e.g. 3400 Hz). The sampler operates on the output of this filter to meet the Nyquist criteria (e.g. 8000 Hz sampling frequency). The μ-law/A-law compression is introduced at this stage. At the final stage of the encoder, the amplitude quantizer assigns one of the uniform quantization levels (e.g. 256 levels) to the sampled and compressed signal (64 kb/s coder). The nonuniform quantization resulted from the combined compression and uniform quantization operations is more appropriate for the speech signal (log-PCM). The decoder performs the reverse operations (Inverse quantization, Expanding, and Interpolation filter). As seen in Fig. 1.1, in 1972 CCITT standardized these coders (G.711).

Log-PCM coding scheme does not take advantage of the redundancies in the speech signal. *Differential Pulse Code Modulation* (DPCM) and *Adaptive* DPCM (ADPCM) coding strategies, exploit some of these redundancies. Before quantization, a predictor filter is used to estimate the speech sample. The coder in effect quantizes the difference of the actual speech sample and the predicted sample. The structure of

---

† Most of the work for this thesis was done during winter 1989 and the first half of 1990. The writing however was interrupted and only concluded in early 1991. This is why the results of the thesis was reported prior to the conclusion of the final version of the thesis [11, 12, 13]. Also the work is continued by J. Grass, the author and others at INRS-Télécommunications for the 12 kb/s based on the recommendations of this thesis [14].

‡ Other suggestions for modifications, not reported at the time of this thesis were discussed in Ref. [15] (International Conference on Acoustics, Speech, and Signal Processing '91).

such coders is more formally introduced in the next chapter. The result is a gain in use of transmission rate of 2:1 with coding rate at 32 kb/s with equivalent toll-quality of 64 kb/s log-PCM coders. As seen in Fig. 1.1, the CCITT standardization of the 32 kb/s coders took place in 1984 (G.721). The G.721 is also extended to 24 and 40 kb/s in G.723. This 2:1 compression and the so called 2.5:1 gains of Time Assignment Speech Interpolation (TASI) or Digital Speech Interpolation (DSI) (effect of silences in speech) results in an effective circuit gain of 5:1 over conventional telephone systems. The wireless applications (short distance indoor) have benefited from the G.721 for its simplicity. The CCITT G.EMB standards is the extended (draft version) ADPCM application at rates 40, 32, 24, and 16 kb/s.

In the 32 kb/s standardization, no side information is transmitted and in order to minimize transmission delay, all adaptation processes are in a *backward* fashion [16]. The backward and *forward adaptive prediction* differ in the positioning of the buffer window of the samples in the prediction analysis. This means that for the backward adaptation, buffering of tens of milliseconds of the speech signal for prediction analysis does not include "future" samples (as it does in the forward adaptation), i.e. the predictor and quantizer are updated using information contained in the past reconstructed samples. This of course results in a small degradation in prediction quality.

Lowering the ADPCM coder bit rate below 32 kb/s means lower bit rate available for the residual signal coding and higher quantization noise. The backward adaptive prediction which relies on the past reconstructed signal degrades rapidly as quantization noise increases. Hence, the coder toll-quality of the ADPCM algorithm can not be maintained at 16 kb/s.

Although use of forward adaptation improves the performance at the cost of increased delay, as seen in Table 1.1, this encoding delay is specially undesirable for the network applications. The following paragraphs explain how other modifications are needed in order to simultaneously maintain coder quality and low-delay. The interface between the 4-wire to 2-wire lines with impedance mismatch generates echoes. The echo round trip delays result from a combination of encoding delay and (long distance)

propagation time. The application of echo cancellers at the 4-wire/2-wire junction reduces this echo. The amount of tolerable echo by the canceller is a main factor used to define the maximum and objective coding delays in Table 1.1.

For the 16 kb/s CCITT low-delay standardization (currently under way, Table 1.1), the opposing goals of low encoding delay, speech quality have to be accommodated. The backward adaptive prediction has to be used if the delay requirement of Table 1.1 is to be met. The computation cost of the many coding techniques combined in such a complex coder has to be kept limited to the computational power of the DSP chips available. As mentioned earlier, although channel error performance (and other) requirements are constrained by the limits of Table 1.1, for other applications such as mobile radio and indoor wireless, the channel error rate can be much more severe than the ones in Table 1.1.

Both LD-CELP and LD-TREE coder belong to the class of *Delayed Decision* or *Multipath search coding* which is introduced in detail in the next chapter. In the speech model where decision on choice of an excitation sequence deriving a reconstruction filter has to be made, number of samples in the sequence defines the encoding delay. The actual coder delay is 2 to 3 times this encoding delay mostly due to buffering and processing time considerations. In order to obtain 16 kb/s bit rate with a sampling frequency of 8000 samples per second, the coder can only use 2 bits/sample. The choice of tree or codebook for the excitation sequence is the main difference between the LD-CELP and LD-TREE.

The simulations results of this work (presented later) show that the two coders have comparable performance at 16 kb/s under clear channel conditions. Other contributions of this thesis include methods to improve the performance of the two coders, particularly with the view of bringing the bit rate below 16 kb/s (medium rates 8–16 kb/s). The (original) two coders used different methods and strategies for their components. The analysis and improvements to these components (formant-pitch prediction schemes, gain scaling methods of the excitation sequence, types of the perceptual weighting for the error signal, etc.) are also important for the purpose of better coders at 16 kb/s and below. The challenge is to develop high-performance

compatible components for the target future coder, operating at bit rates between 8 to 16 kb/s. In particular, the backward adaptive prediction component is investigated at length in this thesis. Suggestions to improve the two coders include an improved high-order predictor (applicable to both coders), training of the excitation dictionary as well as a better gain adaptation strategy for the tree coder. The high-order prediction alternative is compared to the conventional formant-pitch configurations. The performance of high-order predictors is studied and greatly improved.

Concluding from the advantages of each of the two coders (LD-TREE and LD-CELP) and the choices of the individual components, suggestions are made to combine CELP and tree structures for future coders. A hybrid coder, taking the best components from the two archetypes (possibly with a "Tree-CELP" structure) is a good candidate to push coding rates below the current 16 kb/s. Future high-quality low-delay speech coding at medium rates should benefit from such combined structures and schemes. †

1.1 Organization of the Thesis

In Chapter 2 the overall structure, components, and concepts of low-delay speech coders in general and the two coders (LD-CELP and LD-TREE) in particular are discussed. Chapter 3 focuses on the backward linear prediction. Issues and methods of pitch and formant prediction are investigated. The experiment results on the backward linear prediction methods of Chapter 3 are presented in Chapter 4. Chapter 5 gives a detailed description of the two coders algorithms. It also discusses ways of improving both coders. The discussion on the nature of the differences and similarities between the two coders leads to the recommendations for future hybrid coders which combine the best of both coders. Summary and recommendations for future work are in the final chapter.

† The work of low-delay speech coding at 12 kb/s in Ref. [14] utilizes the above recommendations.

Chapter 2 Low-Delay Speech Coding

This chapter presents a review of the various components of high-quality low-delay speech coders at 16 kb/s and below (medium rates 4.8 to 16 kb/s). The thesis focuses on LD-CELP and LD-TREE coders. The discussion provides both background and insight into the components of these two coders. The general techniques of analysis-by-synthesis and Delayed-Decision coding are first introduced. Then, the overall structure of the LD-CELP and LD-TREE coders are presented. Various components of the two coders as well as similarities and differences between them are studied.

2.1 Introduction

Prior to toll-quality low-delay speech coding at medium rates, many barriers of achieving better coders with lower bit rates were broken. The bandlimiting filter which is used before sampling of the analog telephony signals, limits the signal frequency higher band to 3400 Hz. A sampling frequency of 8000 Hz in the conventional method of Pulse Code Modulation (PCM), along with the μ-law or A-law companding methods and a uniform quantization of 256 levels (8 bits/sample) result in 64 kb/s coding rate. The above method of *log-PCM* which has the obvious advantage of zero delay, does not take advantage of the of the correlations among the samples. The method of Differential Pulse Code Modulation (DPCM) exploits these redundancies to bring down the coding rate. This method includes a linear predictor filter which produces a predicted sample value corresponding to each speech sample. The quantizer quantizes the prediction error signal which is the difference between the current

sample value and the predicted value. If the selection of the predictor coefficients as well as the quantization are chosen to be adaptive to the stationary characteristics of the signal and the dynamic range of the predicted error signal respectively, the coding rate is further reduced (Adaptive Differential Pulse Code Modulation or ADPCM). The conventional ADPCM structure is shown in Fig. 2.1. The two components of the conventional ADPCM are the quantizer Q and the predictor $F(z)$. It is well known that the quantizer Q and the predictor $F(z)$ (the z-transform of the prediction filter response $f(n)$) have to be in a closed-loop configuration (such as the one in Fig. 2.1) so as to minimize the effect of the quantizer error in the coded speech [17, 18]. The CCITT's standardization for the speech coders at 32 kb/s uses ADPCM to produce toll-quality speech.

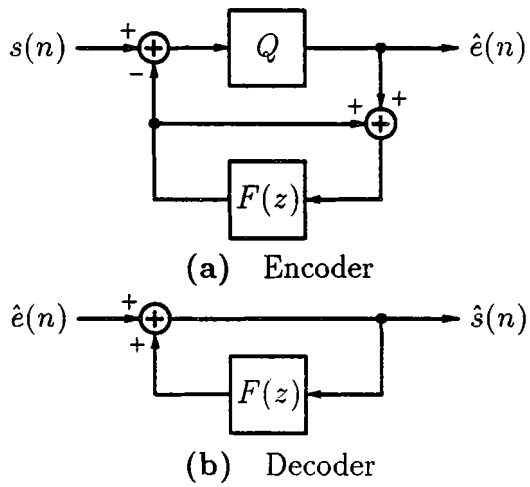


Fig. 2.1 Conventional ADPCM coder

The strategies of the method of ADPCM can not yield 16 kb/s bit rate coders which can maintain both toll-quality speech and low delay. As it was mentioned in Chapter 1, the high quantization noise and the resulted poor backward prediction degrades the coding quality. The alternative of forward adaptation is not considered since it introduces coding delay longer than network application toleration. In the forward adaptation, in order to estimate the parameters of an adaptive operation, often an analysis frame of tens of milliseconds is used. As an example an analysis frame of 160 samples (20 ms for the 8000 Hz sampling frequency) may be used. The

encoder estimates these parameters based on some optimization procedure and then uses the obtained parameters for those 160 samples. The parameters are transmitted as side information to the decoder. Since it is necessary to have an analysis buffer of 160 samples, a delay of 160 samples or 20 ms (for the sampling frequency of 8000 Hz) is resulted. Backward adaptation means that blocks of reconstructed signal, up to the present sample, are used for the analysis. Since these quantized samples are available both to the encoder and decoder, there is no need for transmission of side information. The real advantage is the elimination of the analysis buffer of future samples which means no delay. The cost is a small performance degradation due to two facts: first, the quantization error has a negative effect on the future sample quantization which is based on the reconstructed samples, and second the analysis window in the backward adaptive case only includes past samples and hence the analysis does not benefit from the future samples (the parameters may be "stale").

The *generalized ADPCM* coders may use time-varying linear predictor filters to model far-sample as well as near-sample correlations. The near-sample filter models the spectral envelop with resonances (called formant) and antiresonances. The resonances are due to the poles of the vocal tract frequency response, while some spectral nulls are due to zeros of the response. The spectral fine structure (pitch) is modeled by the far-sample prediction filter. These models discussed in Chapter 3 remove the speech signal formant and pitch redundancies and hence reduce the required bit rate. The "noise shaping" consideration of the *generalized Adaptive Predictive Coding* (APC) structure is an improvement over the ADPCM one. In the ADPCM model the output error signal is approximately white. This does not explore the human perception characteristics. The distortion in the coded speech is less likely to be perceived by the human ear at frequencies where the speech signal energy is high. The desired noise shaping scheme, filters the quantization noise so that it has the same spectral shape as the input speech (increased noise energy in formant regions but decreased elsewhere). As a result, there is a bias toward less noise at low frequencies. This is because the speech spectrum masks the perceived noise to some degree and the fact that the ear is more tolerant to the noise at high frequency. The noise shaping filter

system function is often chosen to be a bandwidth expanded version of the transfer function of the formant predictor. The generalized APC structure allows for this kind of control over noise spectrum. This structure which has a *closed-loop* configuration is shown in Fig. 2.2.

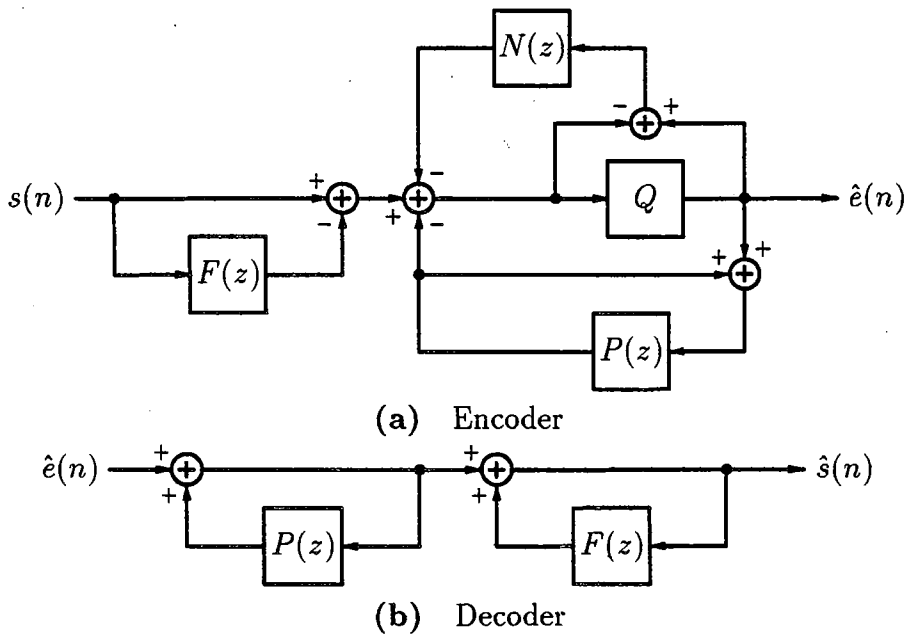


Fig. 2.2 A closed-loop configuration coder

Other than the two components Q and $F(z)$ (quantizer and formant predictor) which were previously seen in the conventional ADPCM of Fig. 2.1, the generalized closed-loop structure includes a pitch predictor $P(z)$ and a noise shaping filter $N(z)$. In this configuration the feedback loop is around the quantizer and the closed-loop includes the pitch predictor. The additional noise-shaping filter reduces the perceived distortion. The configuration of Fig. 2.3 is suggested for the general *open-loop configuration* [19]. In this structure the formant redundancy removal precedes the pitch redundancy removal (F–P open-loop configuration). The open-loop configuration of Fig. 2.3 also includes a noise feedback filter (but not explicit). This configuration which is used in the CELP coders, may also be used as an alternative to the closed-loop configuration in the APC coders. The discussion of the next section introduces

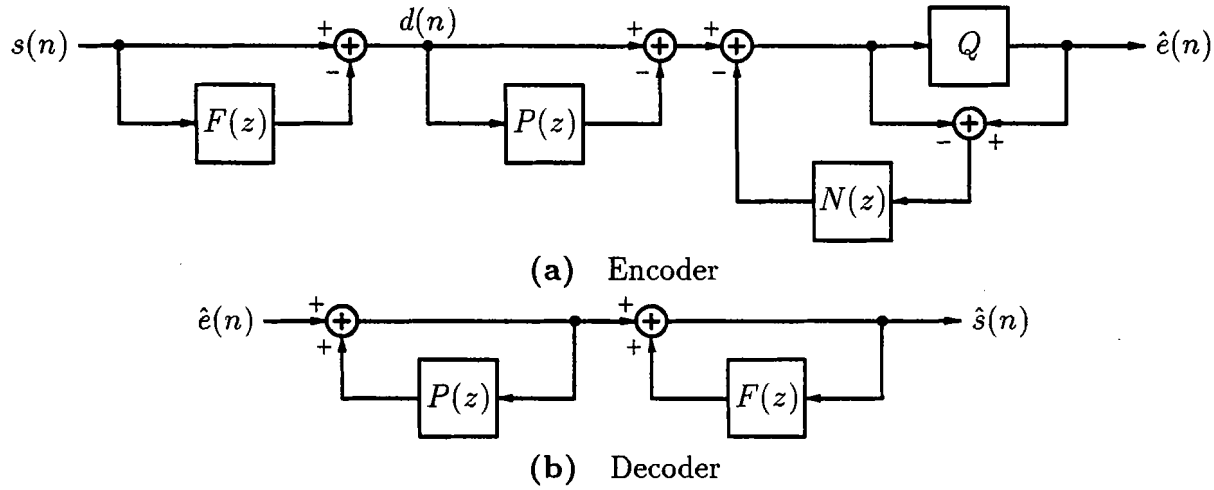


Fig. 2.3 An open-loop configuration APC coder with noise feedback

Linear Predictive Coding (LPC). A full discussion of the formant and pitch predictor used in configurations of figures 2.2 and 2.3 are postponed to the next chapter.

Analysis-by-synthesis scheme which provides a better control over quantization noise is another concept used in the LD-CELP and LD-TREE coders. Next section reviews the LPC model of speech in an analysis-by-synthesis configuration. The class of coders which make the task of high-quality low-delay speech coding possible, is the Delayed-Decision Coding or Multipath Search Coding. *Codebook, Tree* (employed in the LD-CELP and LD-TREE coders respectively) and *trellis coding* structures all belong to this class of coders. They employ encoding delay (here within the allowed low-delay requirements) to identify the best possible sequence out of a set of alternatives. Other low-delay components such as Gain adaptation, Postfiltering are also reviewed in this chapter.

2.2 LPC model and Analysis-by-Synthesis

The redundancies in natural speech are a direct result of human vocal tract structure and the limitations of the generation of speech as well as human hearing and perception. Various coding methods exploit these redundancies taking into consideration the following facts:

- In general, vocal tract shape and thus speech spectrum change relatively slowly compared to the sampling frequency (although abrupt changes occur at closures of vocal tract);
- The vocal cords vibrate rapidly but the change in the vibration rate (i.e. F0, the fundamental frequency) is relatively slow;
- Successive pitch periods are very similar most of the time;
- The vocal tract spectrum varies slowly with frequency, and most speech energy is at low frequency;
- Speech sounds can be modeled as periodic or noisy excitation passing through a vocal tract filter, and each sound can be represented with a few parameters;
- Characteristics and limitations of human audition such as higher sensitivity to lower frequency, insignificance of spectral zeros, phase insensitivity, and masking phenomena can be used [17, 18].

The linear predictive coding (LPC) is an important speech model which uses the above facts and is used in many speech coders including the generalized ADPCM and APC. The basis for the LPC model is the production of speech via a linear filter system representing the human speech synthesis. An excitation source $U(z)$ (representing the z-transform of time sequence $u(n)$) is the input to a shaping filter $H(z)$ to produce $\hat{S}(z)$ the output speech. As a possible excitation signal, $U(z)$ may be chosen from a carefully picked excitation dictionary.

The time domain equivalent representation shows how a linear combination of the p previous output samples and $q+1$ previous input samples are used to predict $\hat{s}(n)$ (with a gain factor G):

$$\hat{s}(n) = \sum_{i=1}^{p} a_i \hat{s}(n-k) + G \sum_{l=0}^{q} b_i u(n-l). \tag{2.1}$$

In most cases, the all-pole Auto-Regressive model (AR, $q = 0$) is chosen over the all-zero Moving Average (MA, $p = 0$) or the general pole/zero ARMA model (Eqn. 2.1). A drawback of this choice is that the spectral zeros due to the glottal source and/or vocal tract response in nasal and unvoiced sounds are not represented. However an additional 2–3 poles can approximate the zeros closely.

When the all-pole model is used, using the z-transform of Eqn. (2.1), $H(z)$ may be written as:

$$H(z) = \frac{G}{A(z)} \quad \text{where} \quad A(z) = 1 - F(z) = 1 - \sum_{k=1}^{p} a_k z^{-k}. \tag{2.2}$$

$A(z)$ is called the inverse filter (the inverse of the all-pole $H(z)$). If the input to the inverse filter $A(z)$ is speech signal $S(z)$, the output is $E(z)$ which is called the prediction error signal.

As shown in the general configurations of the previous section, various coding methods use the LPC model in removing both far-sample and near-sample redundancies. The variations of these structures and adaptation schemes are discussed in detail in the next chapter. The generalized ADPCM and APC use such structures to produce high quality speech at rates between 16 to 32 kb/s. As mentioned earlier, the APC has the advantage of control over the noise spectrum and hence produces better results. When medium rate coders (4.8–16 kb/s) are needed, an even better control over the distortion resulted from the quantization is needed and one needs to increase the efficiency of the residual signal quantization. The method of analysis-by-synthesis improves the control over distortion by minimizing the error at the encoder between the reconstructed (output) signal and the original signal [20]. Fig. 2.4 shows an example of analysis-by-synthesis configuration which is used in the CELP and other coders [21, 22, 23]. This configurations is explained in the next paragraphs.

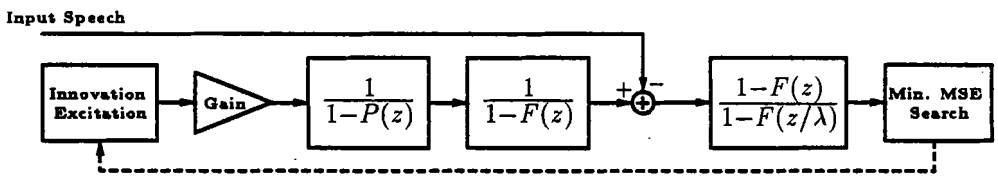


Fig. 2.4 An analysis-by-synthesis configuration

The reconstructed signal is the output of the speech production model. In order to perform the error minimization procedure, encoder must include a replica of the decoder so that the identical reconstructed signal becomes available at the encoder. As seen in the Fig. 2.4, easy incorporation of the perceptual weighting filter ($W(z) = \frac{1-F(z)}{1-F(z/\lambda)}$) is an advantage of the analysis-by-synthesis structure. The synthesis filters $\frac{1}{1-F(z)}$ and $\frac{1}{1-P(z)}$ are all-pole filters similar to the one in Eqn. 2.2 which are used for near-sample and far-sample redundancies. As it is seen in the next chapter, a

single (high order) synthesis filter may replace the two separate formant and pitch filters (pitch synthesis filter is eliminated in Fig. 2.4.). The discussion about the adaptation of parameters in $F(z)$ and $P(z)$ is also postponed until the next chapter. The excitation signal is not necessarily one single sample as it is the case in the simple analysis-by-synthesis structure of Fig. 2.4. As seen in the next section it is possible to have a sequence of samples as the excitation sequence. The excitation sequence (or sample) is determined such that the Mean-Square perceptually weighted Error (MSE) between the original and the reconstructed signal is minimized.

Adaptive gain scaling of the excitation signal (gain component after the excitation signal in Fig. 2.4) improves the excitation representation by reducing the dynamic range of the excitation set. The excitation signal is multiplied by the gain factor and then passed through the synthesis filter(s) to generate the reconstructed signal. This section of the structure is common to the encoder and the decoder. The error signal is passed through a perceptual weighting filter prior to the error minimization.

Methods to decrease the computational complexity of the analysis-by-synthesis configuration are discussed in Section 2.7. Computational reductions are obtained by better placement of the perceptual weighting filter and by the separation of *Zero-Input Response* (ZIR) and Zero-State Response (ZSR) of the system.

2.3 Delayed-Decision Coding

The methods of Delayed-Decision Coding, which result in the CELP, Tree, and Trellis coding, are excitation sequence procedures which are discussed in the this section. Other excitation methods such as *Residual-Excited Linear Prediction*, *Multi-pulse* and *Regular-pulse excitation* coders which also belong to the class of analysis-by-synthesis are found in the literature [24, 25, 17].

Through the use of Delayed-Decision Coding, an efficient representation of the residual signal is possible. A delayed decision is made as to which optimum residual quantization value has to be selected. A sequence of future values of speech signals as well as the current sample are used. This allows the realization of rates $R < 1$

bits/sample at the cost of the introduced delay. Low-delay requirement of Table 1.1, limits the number of samples in the sequence to 5–8 (0.625–1.0 ms at sampling rate of 8000 Hz). As a result, the coding delay (usually 2–3 times sample sequence length) not only is less than the standardization maximum delay of 5 ms, but also meets the objective of 2 ms. To obtain the 16 kb/s coding rate $R = 2$ bits/sample is used (Coder bit rate=$R\times$sampling frequency). This section focuses on the codebook and tree coders which belong to the delayed-decision coding class. Trellis coding, the other member of this class is only briefly reviewed.

For the Independent, Identically Distributed (i.i.d.) sources, the delayed-decision coding provides closer performance to the rate distortion bound than the zero-memory quantizers [26, 27, 28]. The delayed-decision coding is also beneficial in the case of correlated signals such as speech. The quantization sequence is selected from a set with correlation characteristic similar to the speech signal. Fig. 2.5 shows examples of the 3 classes of delayed-decision coding: a) codebook, b) Tree, and c) trellis.

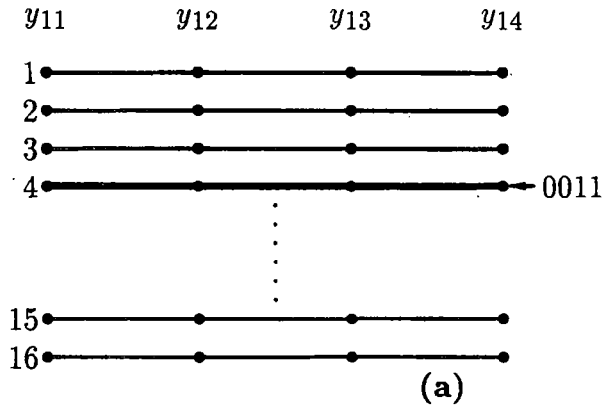
If an R-ary coder sequence of length N and R bits/sample is used, the transmitted code C, is the index of one of:

$$J = (2^R)^N \tag{2.3}$$

entries of a unique non-restricted codebook. For the example in Fig. 2.5, $R = 1$ and $N = 4$ and hence the number of entries in the codebook are $2^4 = 16$. In the case of the CELP coder of this thesis, $R = 2$ bits/sample and $N = 5$ samples/vector are used which results in a codebook of size of $(2^2)^5 = 1024$ entries (code vectors). The actual number of bits transmitted over the channel for each vector is $R \times N$, $1 \times 4 = 4$ for the example and $2 \times 5 = 10$ bits/vector for the CELP coder.

In the CELP coders, a preselected excitation codebook of size J (e.g. 1024) with each entry being a vector with dimension N (e.g. $N = 5$) are stored and are available both at the encoder and the decoder. As easily seen, fractional coding rates (e.g. $R =$ 3/2 or 1/2) are also possible through variations of the codebook size J, and codevector dimension, N (Eqn. 2.3).

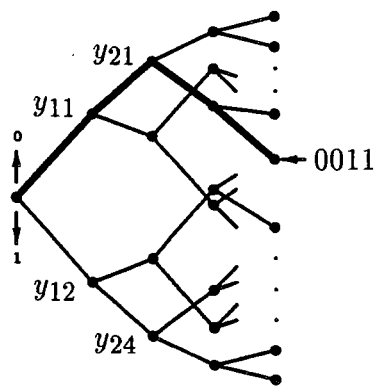
The search for the optimum codebook entry at the encoder in the analysis-by-synthesis configuration is in effect systematically trying each sequence (vector), then



(a)

CODEBOOK

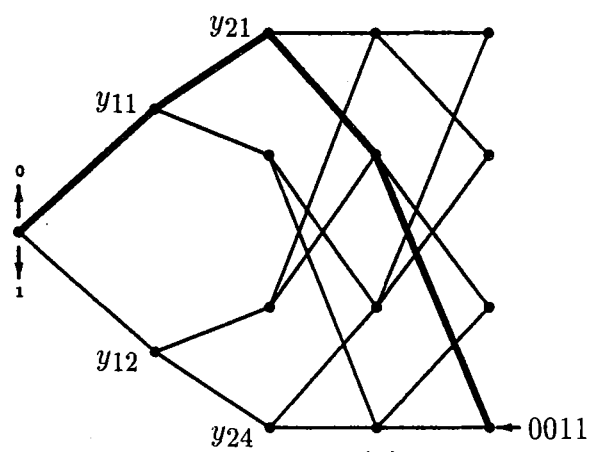
$N = 4$ **Length**
$2^N = 16$ **Sequences**
0011 **Optimum path**



(b)

TREE

$L = 4$ **Length**
$2^L = 16$ **Sequences**
0011 **Optimum path**



(c)

TRELLIS

$L = 4$ **Length**
$2^L = 16$ **Sequences**
$K = 2$ **Trellis intensity**
$2^K = 4$ **Nodes/sample**
0011 **Optimum path**

Fig. 2.5 Examples of multipath search structure with $R = 1$, a) Codebook, b) Tree, and c) Trellis Coder [28]

selecting the one with the lowest perceptually weighted error between the original signal sequence and reconstructed signal sequence (exhaustive search). The index of the selected codebook entry is transmitted to the decoder which reconstructs the sequence using the identical codebook and decoding structure (codebook is the excitation signal selection in Fig. 2.4.).

Codebook coding used in an analysis-by-synthesis structure results in the conventional CELP coder structure which is shown in Fig. 2.6. In this structure both formant and pitch synthesis filters are included. The LD-CELP coder replaces the formant and pitch synthesis filters by a single high-order synthesis filter (Fig. 2.13). As explained further in next chapter, high-order synthesis filter CELP coder (unlike formant and pitch filters configuration of Fig. 2.6) performs well in the presence of channel noise.

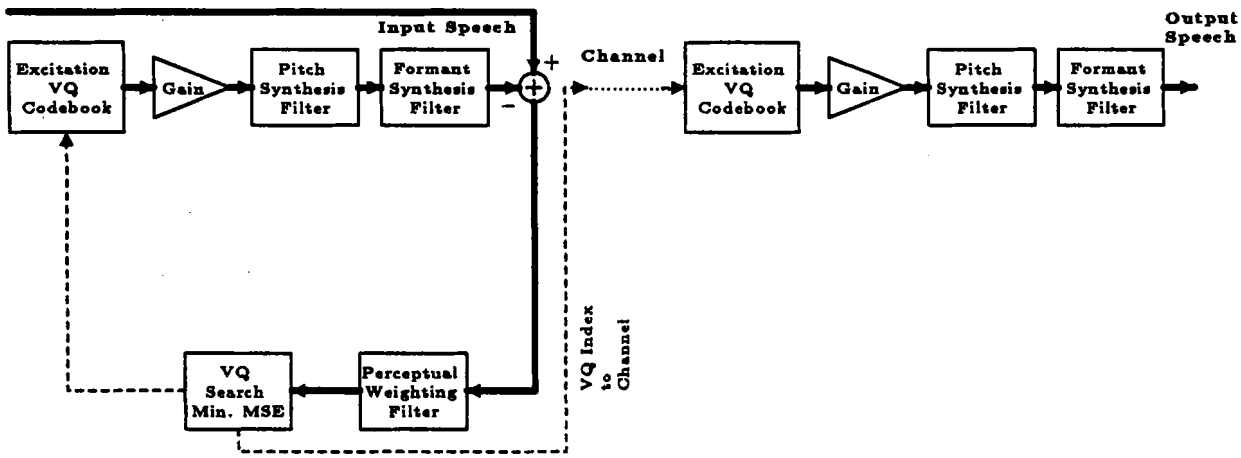


Fig. 2.6 Conventional CELP encoder and decoder block diagram

If the codebook size is very large, as larger size means richer codebook and better performance, the optimum exhaustive search of codebook may be replaced by the sub-optimum methods of tree and trellis coding (sub-optimum for the same size codebook). This is to decrease the search complexity.

In the tree and trellis coding, different sequences have several common elements and individual sequences form a path in the tree or trellis. In these cases the path

information is transmitted to the decoder. The tree structure consists of nodes and branches. The number of branches b, per node is called branching factor. If β symbols per each node are used, the encoding rate R in bits per symbol is given by:

$$R = \frac{1}{\beta}\log_2 b = \frac{k}{\beta} \quad (b = 2^k). \tag{2.4}$$

When $\beta > 1$ is used, here after the term *Multi-symbol/node tree* is used. In effect this structure combines tree and codebook resulting in a Tree-CELP coder where codebook of β samples are used at each node [14]. Note that sampling frequency is a factor in the resulted coder bit rate in these considerations (bit rate=# of bit/sample $\times$ sampling frequency). Here a sampling frequency of 8000 sample per second is assumed. As an example fractional rate of $R = \frac{3}{2} = 1.5$ is obtained when $\beta = 2$ and $b = 8$ are used. This example which is illustrated in Fig. 2.7, is used (sampling frequency of 8000 sample per second) to obtain the coding rate of 12 kb/s in Ref. [14].

A consistent assignment of branch number throughout the tree, results in a unique *path map* for each path sequence. At depth d (distance from the root), there are b^d possible path maps starting from the root. $\beta = 1$ sample/node and branching factor of 4 results in rate 2 bits/sample (Eqn. 2.4). Rates less than 2 bits/sample are needed to obtain coding rates below 16 kb/s.

In the configurations where $\beta > 1$ is used the main issue is how to selected the β symbols at each node. In the Tree-CELP structure, the sequences of multi-symbols for populating nodes (code vectors) may be selected from random numbers with certain distribution. Training as seen later usually improves the results. Other than the Tree-CELP alternative, the concept of *multi-tree* is suggested in Ref. [29, 30]. The idea is that the branching factor of the tree at different depths along all paths changes. As seen in Fig. 2.8 example, branching factor of 2 and 4 means that along all paths, at depth d = odd numbers, branching factor is 2 and at d = even numbers, the branching factor is 4. The resulted multi-tree has a rate which is the arithmetic mean of the component trees:

$$R = \frac{(2+1)}{2} = \frac{3}{2}\text{bits/sample.}$$

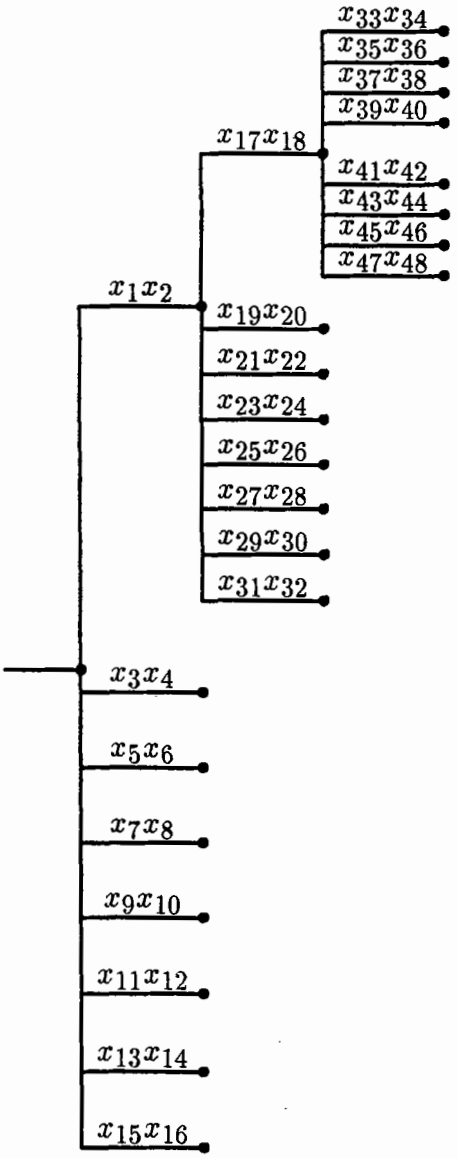


Fig. 2.7 Fractional rate tree of 1.5 bits/symbol using multi-symbol per node approach

In the trellis coding (another sub-optimum structure) [31, 32], the structure (Fig. 2.5) starts as a tree and then collapses to the specific trellis structure (limiting the number of fan-outs). Trellis coding is not considered in this thesis.

2.3.1 Search Methods for the Codebook and Tree Coders

As was seen in the structure of analysis-by-synthesis of Section 2.2, each excita-

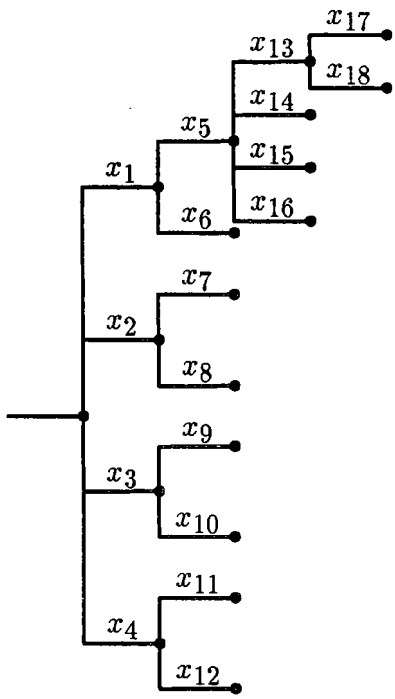


Fig. 2.8 Fractional rate tree of 1.5 bits/symbol using multi-tree approach

tion *innovation* sample will produce a different *reconstruction* output sample. The difference between the reconstructed sample and the speech sample, the error sample is passed through the perceptual filter to form the perceptually weighted error sample. In the CELP coder, the innovation sample sequence is chosen from the gain normalized excitation code vectors. These vectors are the entries in the codebook. In the tree coder, for each tree structure populated with the innovation samples (*innovation tree*), there is a corresponding tree structure populated with the reconstructed output samples (*reconstruction tree*). The innovation excitation sample sequence in this case is the sequence of innovation samples along a particular path in the tree. Hence for each innovation sample sequence there is a corresponding reconstructed output sequence.

In the case of CELP (codebook) coder, the search procedure for the best codevector may simply be by the exhaustive search through the codebook entries to find the entry which provides the reconstructed sequence with the minimum cost (distortion).

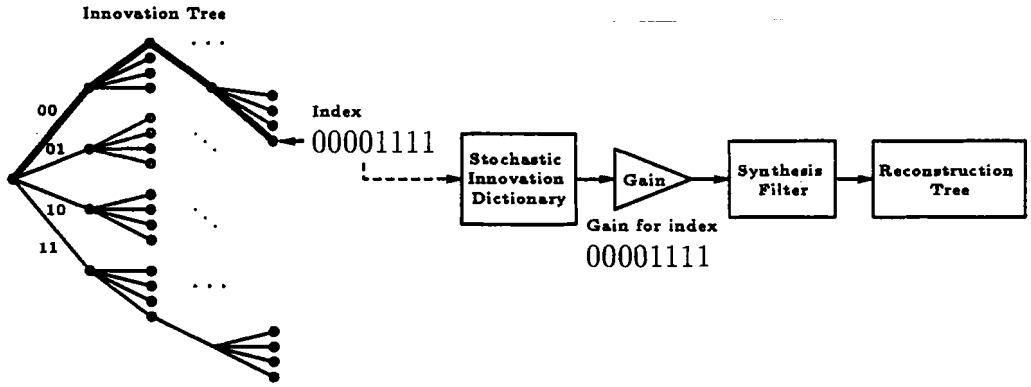


Fig. 2.9 A stochastic tree structure with branching factor $b = 4$, $\beta = 1$, and the resulted 2 bits/sample rate (used in the LD-TREE).

The cost function (distortion measure) is usually based on the Euclidian distance between the vectors. The search is done on vector-by-vector basis and is by passing the difference between the reconstructed sample vector (sequence) and the speech vector through the perceptual filter. Each excitation codebook entry results in a different reconstructed vector and hence a different perceptually weighted error vector. The goal is then to select the entry which produces the minimum MSE distortion i.e. to find the index of the entry which results in the perceptually weighted error sequence with minimum norm.

In the tree coder, the accumulated MSE distortion for each path is used. The sum of accumulated perceptually weighted MSE distortions of all nodes along a path is the cumulated MSE distortion for that path. The search goal is to find the path with minimum accumulated MSE distortion. The search procedure, in this case is less straight forward and is done on sample-by-sample basis. The tree height and the number of path (number of nodes at maximum depth) grow with time. Delay and computational complexity constraints limit the height and the number of paths in contention in the search procedure. The search method of (M,L) which is used in this study is an algorithm based on this idea [33]. The use of the algorithm in the tree

coder is described in the following paragraph.

In a *single path search*, the search for the best innovation sequence is through one line of descendants from a root node, while in the *multipath* search, several path stemmed from an individual root are considered. The multipath search results in a better overall performance due to the fact that, while at time n, a path i may have the minimum distortion among all paths, it may not maintain its lead at time $n+1$ (another path may have a lower accumulated distortion). In the (M,L) algorithm, a maximum of M paths are considered in the search procedure. The length of these M paths in contention can not grow very large since the coder delay is defined by this length L, and the computational complexity is increased as L gets larger. Hence the (M,L) search algorithm keeps a maximum of M paths in contention while number of samples in each path is limited to L. In this study M=16 and L=8 are chosen. The value of M is chosen based on the performance and complexity consideration. As shown in Ref. [8], the coder performance curve versus the M value increases with M but almost flattens around M=16. It is the low-delay requirements which restricts the value of L to 8.

At time instant n, each of the M (maximum) paths in contention are extended. The error accumulated for each of the 4M (the branching factor is 4) extended paths is calculated. As each of the innovation samples are passed through the adaptive gain multiplier and synthesis filter, a reconstructed output sample is obtained. The path with the lowest accumulated perceptually weighted MSE distortion is selected. The two bit branch code ($\in \{00, 01, 10, 11\}$ in the case of branching factor 4) of the root of this path, L samples back, is the only information transmitted to the receiver at time n. Only valid paths that stem from this root are kept (maximum M) for the next time instant. Since synthesis, perceptual weighting filters, and possibly gain adaptation scheme along the M path in contention need M separate sets of memory, the memory requirements can become impractical. Section 2.5 outlines a method to overcome this problem.

2.3.2 Innovation Sequences Population

There are three categories of innovation sequence set (excitation codebook in the in the CELP coder and innovation dictionary in the tree coder) population: *Deterministic*, *Stochastic*, and *Iterative*. Deterministic population means that a restricted uniform or non-uniform alphabet (such as the one used in the R-bit quantization of DPCM) is used. Stochastic population of the innovation sequences is through generation of random numbers with the desired distribution. The statistical characteristic of the innovation sequence set needs to match to that of the long-term histogram of the (gain normalized) residual samples. For the CELP and tree coders, the Laplacian and Gaussian distributions have been suggested [8, 27]. The iterative or trained population is achieved through optimization methods based on Lloyd's basic development (e.g. vector generalized Lloyd algorithm of *Linde et al* (LBG) [34]) using a very large number of speech samples for the training sequence. The speech used for the training has to include various types of speech (various speakers and recording types with an appropriate selection of phonetically balanced sentences). This is to maintain the coding quality for various speech types (better matching of the quantization innovation sequences to the actual speech innovation sequences).

Systems based on the stochastic and iterative populations tend to produce more effective and refined innovation sequences for the CELP and tree speech coders [21, 28] than the deterministic population. For the LD-TREE coder, with only 4 possible quantization values (branching factor of 4), the deterministic population is simple but not rich. Experiments of this work verified the above claims. The LBG-like training method used for the LD-CELP coder is designed to suit the particular CELP structure. The original stochastically populated tree coder of Ref. [8] was improved through a new iterative (trained) method. The training method used for LD-CELP as well as the new method of training used for the LD-TREE are outlined in Section 2.9 and explained in detail in Chapter 5.

To resolve the issue of the mapping of a large size stochastic innovation dictionary to the innovation tree nodes (uniquely assigning the tree nodes to entries in the dictionary), the procedure used in Ref. [8] adopts the following strategy. Starting

with a dictionary (one sample per entry) of size 2^D populated with random numbers with a Laplacian distribution, a unique path map is assigned to each node. This path map is the concatenated branch numbers from the root to the particular node (each branch number $\in \{00, 01, 10, 11\}$ in the case of branching factor 4). The sequence of the D significant bits of the path maps are used as the index to the innovation dictionary. It is easily seen that meaningful size of the stochastic dictionary can be as large as $2^{D_{\max}}$ where $D_{\max}$ is the binary number representing the value 2 × L (for L = 8, $D_{\max}$ = 2 × L = 16). Although larger size dictionary means richer innovation sequences, the memory requirement for the dictionary, duplicated at the encoder and decoder, and the high required processing power will limit the maximum size dictionary. It turns out that the performance curve as a size of dictionary size flattens after $2^D = 1024$ for LD-TREE coder and a choice of $D > 10$ would not be appropriate ($D = 10$ is chosen in this work). More detail discussion and the results related to the above issues are discussed in Chapter 5.

2.4 Perceptual Weighting Filter

Two forms of perceptual weighting (noise shaping) filter are considered. The simpler and more conventional form is

$$W(z) = \frac{1 - F(z)}{1 - F(z/\lambda)}, \quad \text{where} \quad F(z/\lambda) = \sum_{i=1}^{p} a_i \lambda^i z^i \tag{2.5}$$

is the bandwidth expanded version of $F(z)$ (e.g. λ=0.85). The more general weighting filter has the following form

$$W(z) = \frac{1 - N_1(z)}{1 - N_2(z)}, \tag{2.6}$$

where

$$N_1(z) = N(z/\lambda_1) = \sum_{i=0}^{P'} n_i \lambda_1^i z^i \quad \text{and} \quad N_2(z) = N(z/\lambda_2) = \sum_{i=0}^{P'} n_i \lambda_2^i z^i,$$

with $0 < \lambda_2 < \lambda_1 \leq 1$ (e.g. $\lambda_1 = 0.9, \lambda_2 = 0.4$).

If in the conventional closed-loop generalized APC structure of Fig. 2.2 the pitch predictor is removed (Fig. 2.10), the resulted configuration will be equivalent to a

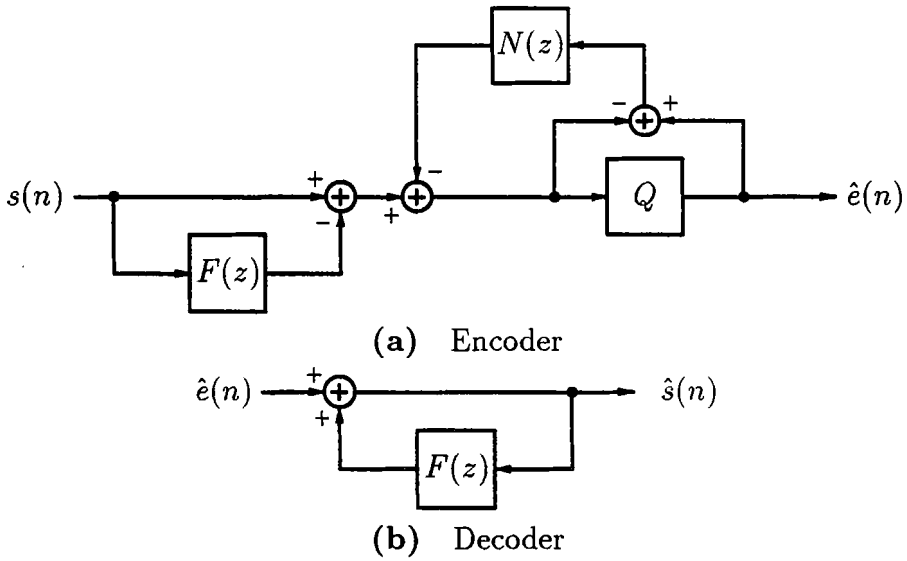


Fig. 2.10 A predictive coder with noise feedback

open-loop structure without pitch predictor. The choice of $N(z) = F(z/\lambda)$ results in an equivalent perceptual weighting filter of the form in Eqn. 2.5. This perceptual weighting filter form is also used in the conventional CELP structures. The LD-TREE coder which is based on the closed-loop configuration uses the simple perceptual weighting filter in Eqn. 2.5. As seen in Section 2.10, it is possible to equivalently represent the LD-TREE in an open-loop configuration similar to the one used by the LD-CELP in Fig. 2.13.

When the perceptual weighting filter of the simple form in Eqn. 2.5 is used in conventional configurations, the disadvantage is that the weighting filter is directly linked to the prediction filter $F(z)$. The more general weighting filter in Eqn. 2.6 does not have this disadvantage. As a result, the general form weighting filter will benefit from the fact that the analysis for the weighting filter coefficient adaptation can be based on the *unquantized* speech. Also, if the order of the synthesis filter is high (as it is the case in the LD-CELP), by using the weighting in Eqn. 2.6 with low filter order (e.g. 10) a better result is obtained. Use of the simple form of Eqn. 2.5 results in speech with artifact quality [2]. The use of separate weighting filter order and analysis has an additional computational complexity cost. The general form weighting filter easily replaces the simple form in the case of open-loop configuration usually used

in the CELP coders (LD-CELP and also the open-loop alternative configuration for the LD-TREE in Fig. 2.12). For the closed-loop configuration which is used in the original LD-TREE, replacement of the simple form weighting filter by the general form is less straightforward. For the closed-loop configuration of Figures 2.2 and 2.10, with the proper choice of noise shaping filter $N(z)$, it is possible to obtain the general weighting filter of the form Eqn. 2.6. A better alternative is discussed in Section 5.4.2.

2.5 Robust Gain Adaptation

To increase the dynamic range of the actual innovation signal, the entries of the excitation codebook or the innovation dictionary are chosen to be gain-normalized values. Hence, at time instant n, each gain-normalized innovation entry $y(n)$, is multiplied by the sample (or sequence) gain $G(n)$, to yield the actual gain-scaled innovation sample (or sequence) $e(n)$, i.e.

$$e(n) = G(n)y(n). \tag{2.7}$$

The sample (or sequence) gain is updated in a backward adaptive fashion. The adaptation strategy may be done using one of the many suggested variance estimators (e.g. [8, 2]). The structure of these estimators differ with each other in the "shape" and length of their memory. The factors to be considered are the estimator quality, complexity as well as robustness to channel errors. The particular structure also has to fit the coder structure i.e. CELP or tree. Other than the *one-word memory Jayant* gain adaptation strategy [35] which is the basis for several gain adaptation algorithms, two other basic strategies are considered here the gain adaptations based on *exponentially averaged variance estimation* [8] and the p-th order logarithmic-gain linear predictor structures [2].

The gain $G(n)$ is proportional to the estimated standard deviation of $e(n)$ ($\hat{\sigma}_e(n)$) or equivalently the RMS value if the signal is zero mean. This estimator consists of a nonlinear operation plus a linear filtering operation $P_G(z)$. Although higher order

filtering produces a better estimate for the filter input signal, the shorter the length of the impulse response of this filter, the more robust the gain adaptor would be. The effect is in how fast the memory of the estimator dies out. The value of an introduced leakage factor effects the channel error performance (robustness) in a similar fashion.

Jayant one-word memory gain adaptation: This adaptation scheme expresses the sample (innovation) gain at time n as the product of the innovation gain of the previous time index and a multiplier $M(n-1)$ which is a function of the innovation sample at time $(n-1)$. The leakage factor β (with value slightly less than 1) is also used. We have $(G(n) = \hat{\sigma}(n))$

$$\hat{\sigma}(n) = \hat{\sigma}^{\beta}(n-1)M(n-1). \tag{2.8}$$

The multiplier values $M(n-1)$ are selected with values less than or greater than 1 depending on the values of the innovation sample (small or large respectively).

The deterministic tree can use this strategy directly. Since the number of choices for the innovation samples is small (4 in this case), the memory requirement is reasonable. The Jayant method is implemented by assigning a gain value to each node. The extended nodes gains are determined by the parent node gain, multiplied by the respective multiplier of the extended nodes. A modified Jayant gain method for stochastic trees is also mentioned in Ref. [8].

Exponentially averaged variance estimation gain adaptation: This method is the preferred method for the stochastic trees. To update the gain at the following formula is used

$$\hat{\sigma}^2(n) = \delta\hat{\sigma}^2(n-1) + (1-\delta)e^2(n-1), \tag{2.9}$$

where $0 < \delta < 1$ ($\delta = 0.86$ is used). In the above adaptation δ controls the effective length of the exponential window (the shorter the length, the more robust the gain adaptor to channel error). This adaptation strategy is well suited for the stochastic tree coding (LD-TREE).

In the LD-CELP excitation vectors are gain-normalized. Therefore the selected codevector $\mathbf{y}(n)$ is multiplied by the estimated gain $\hat{\sigma}(n)$ resulting in the gain-scaled

vector $\mathbf{e}(n)$ which is passed through the synthesis filter. Two methods of vector generalized Jayant gain adaptor and adaptive logarithmic gain predictor are suggested [2] . The former has a fixed coefficient and typically shorter response and thus is more robust to channel errors, while a higher clean channel performance may be obtained with the latter.

Vector generalized Jayant gain adaptation: In this method, the multiplier $M(n-1)$ in Eqn. 2.8 is a function of the gain-scaled vector $\mathbf{e}(n)$ rather than sample $e(n)$. Since it is not practical to have dedicated gain multipliers for each gain-normalized entry in the codebook, the multiplier $M(n-1)$ is chosen to be a function $(f(\cdot))$ of the root-mean-square (RMS) of the selected codevector. Let the symbol x represent the RMS value of the codevector $\mathbf{y}(n-1)$. Only a few parameters which control the function $f(\cdot)$ are optimized. We have

$$M(n-1) = f(x) = \begin{cases} [(1-x)\sigma_{\text{min}}^{1-\beta} + x\sigma_{\text{ave}}^{1-\beta}]\exp\{C_2(x-1)\} & \text{if } 0 \le x < 1 \\ \sigma_{\text{ave}}^{1-\beta}\exp\{C_1(x-1)\} & \text{if } 1 \le x < 4 \\ \sigma_{\text{ave}}^{1-\beta} M_{\text{max}} & \text{if } x \ge 4, \end{cases}$$

where the following parameter values are used for the 16-bit linear PCM input

$$\sigma_{\text{min}} = 1, \sigma_{\text{ave}} = 100, C_1 = \ln(M_{\text{max}}/3), C_2 = -\ln(M_{\text{min}}),$$
$$M_{\text{max}} = 1.8, M_{\text{min}} = 0.8, \beta = (31/32)^5 = 0.853.$$

for the case where there are no channel errors, $\beta = 1$ is used. As a result, the function $f(\cdot)$ consists of two exponential functions $\exp\{C_1(x-1)\}$ and $\exp\{C_2(x-1)\}$ for the two ranges of $0 \le x < 1$ and $1 \le x < 4$. The function is clipped at M_{max} at $x \ge 4$. To have $f(0) = M_{\text{min}}$ and $f(4) = M_{\text{max}}$, the above values for C_1 and C_2 are selected. To compensate for the effect of $\beta < 1$, the term $\sigma_{\text{ave}}^{1-\beta}$ is added. To handle very low input signals, the term $\sigma_{\text{ave}}^{1-\beta}$ is incorporated. To simplify the computation, a look-up table can be used for the multipliers $M(\cdot)$ for the codevectors.

Adaptive logarithmic gain predictor: The next adaptation strategy has a filter $P_G(z)$ of higher order and hence delivers a better clear channel performance at the cost of higher complexity. It uses a 10th order adaptive linear predictor acting in the logarithmic domain. The RMS values of $\mathbf{y}(n)$ and $\mathbf{e}(n)$ are defined as $\sigma_{\mathbf{y}}(n)$ and

$\sigma_{\mathbf{e}}(n)$ respectively. The Eqn. 2.7 for the vector case becomes

$$\mathbf{e}(n) = \sigma(n)\mathbf{y}(n),$$

The predictor filter $P_G(z)$ has to match the gain $\sigma(n)$, to the RMS of the vector $\mathbf{e}(n)$ at time n ($\sigma_{\mathbf{e}}(n)$). Expressed in the logarithmic domain, this formulation means that the logarithmic gain, $\log[\sigma(n)]$, is estimated from the past samples of RMS of $\mathbf{y}(n)$, $\sigma_{\mathbf{y}}(n)$. The predictor has the following form

$$\log[\sigma(n)] = \sum_{i=1}^{10} p_i \log[\sigma_{\mathbf{e}}(n-i)],$$

where the adaptation uses a backward analysis similar to the ones discussed in the next chapter for the formant predictor filters. As seen in detailed algorithm for this gain adaptor in Chapter 5, a log-gain offset value is subtracted from the RMS value $\sigma_{\mathbf{y}}(n)$, to reduce adverse effects caused by the fact that the logarithmic gain is not a zero-mean signal. The value chosen is approximately equal to the averaged excitation gain level (38 dB for the 8 bit log-PCM input speech) during the voiced segments of the speech. Although the performance of this gain adaptation scheme is not as robust to channel errors as the vector generalized Jayant one (longer impulse response), the robustness may be improved using the bandwidth expansion method (Chapter 5).

2.6 Postfiltering

Postfiltering is a method by which the perceptual quality of the speech is enhanced through passing the output speech at the decoder through an additional adaptive filter. The conventional CELP in Fig. 2.6, may include the postfilter block right after the output speech at the decoder). Because of the characteristics of the human auditory system, in addition to the noise shaping methods previously discussed, additional perceptual quality improvements are possible [36, 37, 38]. The original LD-TREE [8] uses postfiltering. The poles and zeros of this filter are a damped version of the predictor poles $F(z)$. The damping factor is adapted according to the predictability of the signal.

Tandeming and Non-voice signal Considerations: Postfiltering which improves the performance of the coder in the conventional ADPCM, CELP or in the original LD-TREE [8], is eliminated in the LD-CELP and LD-TREE studied in this work for severe accumulated distortion during tandem coding. The simulations in this study did not include tandeming performance. Ref. [4] has reported that the performance of the LD-CELP under asynchronous tandeming condition is worst than the G.721 requirements and needs further research. When decoding non-speech (such as modem) signals, postfiltering resulted in phase distortion. Elimination of postfiltering removed the concern for this undesirable effect. †

2.7 Computation Reductions in CELP and Tree Coders

The complete structures of the CELP and Tree coders, as they are evolving, are structurally complex and computationally expensive. Computational reduction techniques suggested for the original two coders in this study (LD-CELP [2] and LD-TREE [8]) are discussed in this section. The techniques used can be divided to two categories; first the schemes which may be used in both tree and codebook structure and the ones which may be used in one of the two structures. As an example of type of computational complexity reductions which may be used in both LD-CELP and LD-TREE coders (or the general analysis-by-synthesis systems), one may move the perceptual filter to the left of the adder, as shown in Fig. 2.11-b. In general when analysis-by-synthesis is used and candidate innovation signals have to be compared, to reduce the computational cost, the computations which are common and constant are moved outside the "loop".

2.7.1 Use of Gain/Shape Vector Quantization

In the LD-CELP, Product Vector Quantization (PVQ) [39] is used to bring down

† However the continuation of the work on the LD-CELP [15], suggests a "specially tuned" postfiltering to be used in the LD-CELP which not only does not have any adverse effects in the tandeming cases but also further improves the quality of the coded speech. The use of appropriate postfiltering is therefore recommended. The addition of postfiltering to both coders only can enhance the quality of both coders and does not go against the discussions in this work.

the computation load by using a 7-bit shape and a 3-bit gain book (1-bit sign and 2-bit amplitude). The codevector index in LD-CELP is the concatenation of 3 indices i, j, and k. The codevector is the product of μ_k, the sign portion of the gain vector ($+1$ or -1), g_i, the magnitude portion of the gain vector, and y_j, the selected shape codevector ($g_i y_j \mu_k$). The gain/shape separation can have a possible positive effect on channel error robustness. This is due to the fact that gain information is partially transmitted to the decoder.

2.7.2 Separation of ZSR and ZIR

In the structures of Fig. 2.11 a single predictor is used. If the order of the perceptual weighting filter and the single high-order synthesis filter were the same, the cascade of synthesis and perceptual filter would have become simplified (Fig. 2.11-b). If the single predictor is a high-order one, the cascade filter is not simplified. This is because the use of high-order perceptual filter is reported [2] to result in an artifact synthesis quality, the cascade is not simplified (see Section 2.4 on perceptual weighting filter). The method of separating the Zero-Input Response (ZIR) from the Zero-State Response (ZSR) suggested in Ref. [22] further reduces the computational complexity and results in the structure of Fig. 2.11-c. The zero-input response is the component of the filtering operation which is also called "ringing" (memory) from previous excitations and is fixed throughout the search for the optimum excitation entry for all entries (computed only once in the beginning of the search). The second component in the filtering operation, ZSR, is the resulted filter output when all filter's memories are set to zero and the cascade filter is excited with one of the excitation entries. This component has to be calculated for all entries. Computational savings result through separating these two components [38].

In the LD-CELP, both separation of ZIR/ZSR and gain/shape (product) VQ are used to reduce the complexity of the search as follows. If $F(z)$ is the synthesis filter transfer function and $W(z)$ the perceptual weighting transfer function, one can form the cascaded filter: $H(z) = F(z)W(z)$. The backward adaptive gain $\sigma(n)$ is known prior to the search. The MSE minimization of the distortion between the difference

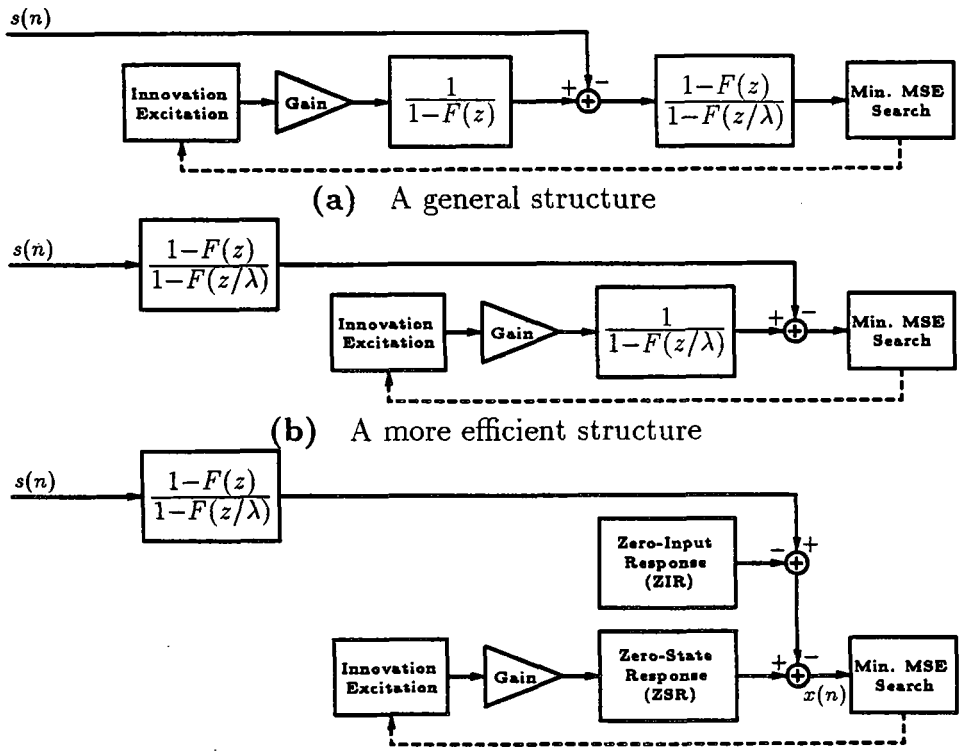


Fig. 2.11 Simplifing the analysis-by-synthesis configuration

target vector $\mathbf{x}(n)$ (vector version of $x(n)$ in Fig. 2.11-c) and the synthesis vector $\mathbf{x}_{i,j,k} = \sigma(n)\mu_k g_i \mathbf{H} y_j$ may be written as

$$D = |\mathbf{x}(n) - \mathbf{x}_{i,j,k}|^2 = |\mathbf{x}(n) - \sigma(n)\mu_k g_i \mathbf{H} y_j|^2. \tag{2.10}$$

$\mathbf{H}$ is the lower triangular matrix

$$\mathbf{H} = \begin{bmatrix} h(0) & 0 & 0 & 0 & 0 \\ h(1) & h(0) & 0 & 0 & 0 \\ h(2) & h(1) & h(0) & 0 & 0 \\ h(3) & h(2) & h(1) & h(0) & 0 \\ h(4) & h(3) & h(2) & h(1) & h(0) \end{bmatrix}$$

and $\{h(0), h(1), .., h(4)\}$ are samples of the impulse response of the cascaded filter.

The minimization further reduces to

$$\begin{aligned} \hat{D} &= -2\sigma(n)\mu_k g_i p^t(n) y_j + \sigma^2(n) g_i^2 E_j \quad \text{or} \\ \hat{D} &= -\mu_k b_i g_i p^t(n) y_j + c_i E_j, \end{aligned} \tag{2.11}$$

where

$$b_i = 2\sigma(n)g_i, \quad c_i = \sigma^2(n)g_i^2,$$
$$p(n) = \mathbf{H}^t\mathbf{x}(n), \quad \text{and} \quad E_j = |\mathbf{H}y_j|^2.$$

2.7.3 Adaptation Coefficient Updates

Less Frequent Adaptation Updates: Since the spectral changes in the speech signals are relatively slow varying, significant computation load reduction is obtained with a minimal loss of performance by updating the coefficients less often (e.g. every 4 or 8-th vector instead of every vector in the LD-CELP). This computational saving is applicable to both coders.

Delayed Adaptation Update: The formant prediction filter coefficients are updated in a delayed update fashion in the LD-TREE. In Fig. 2.12 this is shown as $\{a_i(n-2\mathrm{L})\}$ or $\{a_i(n-\mathrm{L})\}$ which means that the update algorithm at time instant n uses samples as recent as 2L or L samples back. Ref. [8] shows that L sample delay update strategy actually results in better prediction gains than zero delay update strategy. As seen in Fig. 2.12, this also results in complexity reduction of the coder since only one update of the prediction coefficients is done for all paths in contention (also see algorithm in Section 5.1).

2.8 Channel Error Protection Methods

Gray Coding of the Codebook Indices: Shape and gain codevector indices are Pseudo Gray coded [40, 41] to increase channel error robustness. This is because, in the case of a single error occurrence, the received codevector will tend to be close to the transmitted one. The result is a significant improvement in a noisy channel environment. The gray coding is used in the LD-CELP coder.

2.9 Training of the Codebook

For the CELP coder, using the gain adaptive Vector Quantization (VQ) speech coding of [42, 43, 38] and the product VQ training methods of [39], the shape and

gain codebooks may be trained. In the training algorithm (described in detail in Chapter 5), the closed-loop design strategies include structural considerations for gain-shape VQ and use of adaptive gain-scaling of the excitation signal. The initial shape and gain codebook are iteratively refined to produce the final codebooks. The distortion-versus-iteration does not monotonically decrease or converge for the closed-loop design. The codebook with the lowest distortion after a preset number of iterations is stored. When the individually optimized gain/shape VQ algorithm suggested in [39] was used, the initial shape book codevectors were chosen from numbers with gaussian probability distribution, and the initial 3 bit gain codebook scalar values were selected from uniformly distributed gain values. In Chapter 5, a new method of training is outlined to improve the performance of the LD-TREE.

2.10 Comparison of LD-CELP and LD-TREE structures

As described in detail in Ref. [8], the LD-TREE structure is based on the closed-loop generalized APC structure of Fig. 2.2. The LD-CELP structure [2] is similar to the general CELP structure (open-loop structure of Fig. 2.3). Up to this section, using a conceptual approach various concepts and components of the two coders were discussed. In an attempt to better unify the two coders, an equivalent interpretation for the LD-TREE is given in Fig. 2.12. Comparing with the LD-CELP block diagram shown in Fig. 2.13, the following common features may be identified.

- parameter selection using analysis-by-synthesis,
- high performance predictors for redundancy removal,
- gain scaling unit and the gain adaptation,
- perceptual weighting (noise-shaping), and
- innovation sequence dictionary or codebook with delayed search.

The utilized concepts explained earlier in this chapter, can be summarized in this unifying comparison of the two coders. The use of Delayed-Decision Coding of codebook (CELP) and tree can efficiently represent the excitation signal. In an analysis-by-synthesis approach, the search for the optimum excitation dictionary or codebook entry at the encoder is done by systematically trying each sequence. The sequence

with the lowest perceptually weighted error is selected. To generate the reconstructed signal, the encoder uses a replica of the decoder. The index corresponding to the selected sequence is transmitted to the decoder. Adaptive gain scaling of the excitation signal improves the excitation representation by reducing the dynamic range of the excitation set. The gain-normalized excitation signal is multiplied by the gain factor and then passed through the synthesis filter to generate the reconstructed signal. The error signal, the difference of the reconstructed and original signals, is passed through a perceptual weighting filter prior to the error minimization. Note that in Fig. 2.12 (LD-TREE), the flow of the speech sample processing is on a sample-by-sample basis while in the Fig. 2.13 (LD-CELP), the flow is on a vector-by-vector basis.

The essential difference between LD-CELP and LD-TREE coding is the corresponding difference between the *block* and *sliding* source coding. As seen in the discussions later, without channel error considerations, sliding window coders would seem to be preferable in terms of performance alone [26]. There are no block edge effects with sliding window techniques. However, with channel errors propagate for longer times within the sliding block structure.

Both coders contain structures with similar functions. The various components can be mixed and matched between the coders. It must be kept in mind that in a backward adaptive structure, each component must perform well. For instance a good residual quantization results in an accurate reconstructed signal which in turn is used to adapt the predictor. A breakdown in either the residual coder or the predictor update results in breakdown of the coder. Further discussions on comparing the two coders is postponed so that simulation comparisons are first considered.

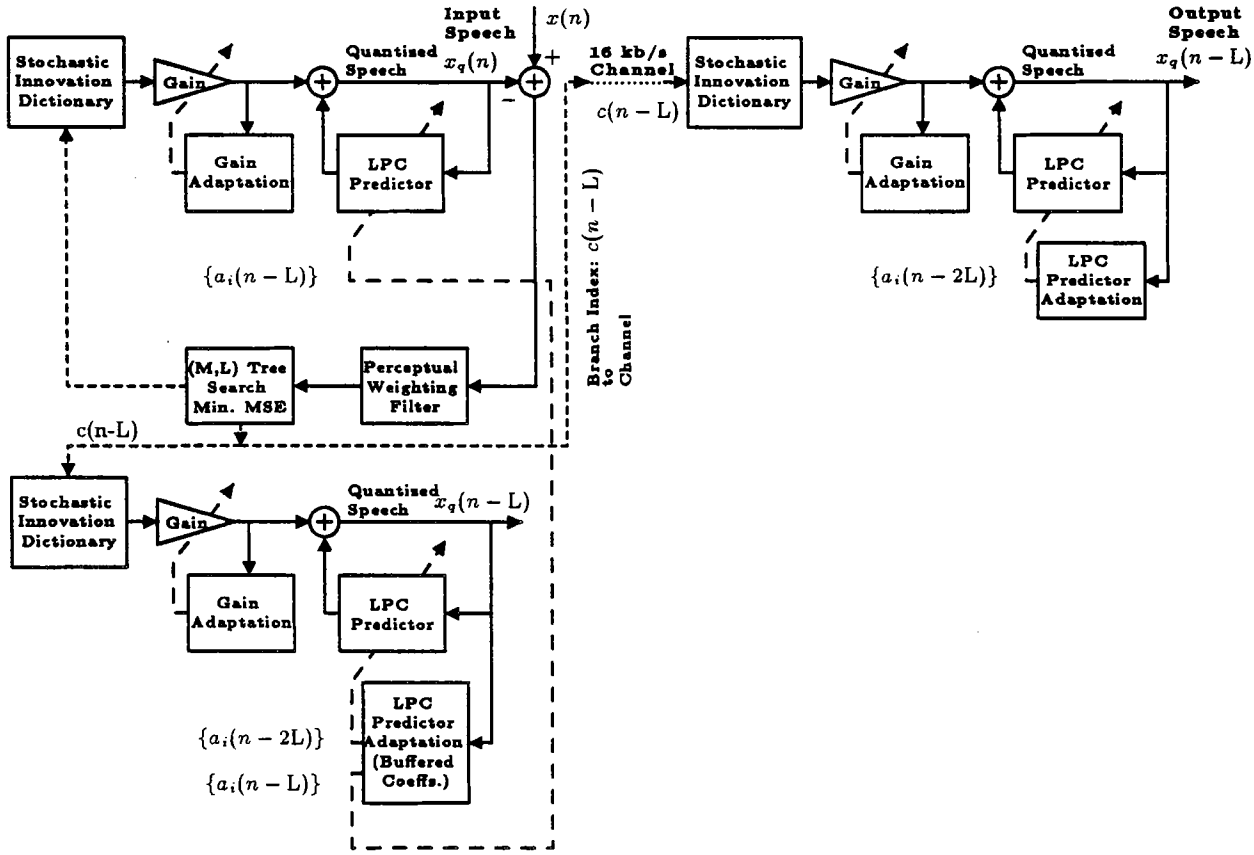


Fig. 2.12 LD-TREE encoder and decoder block diagram

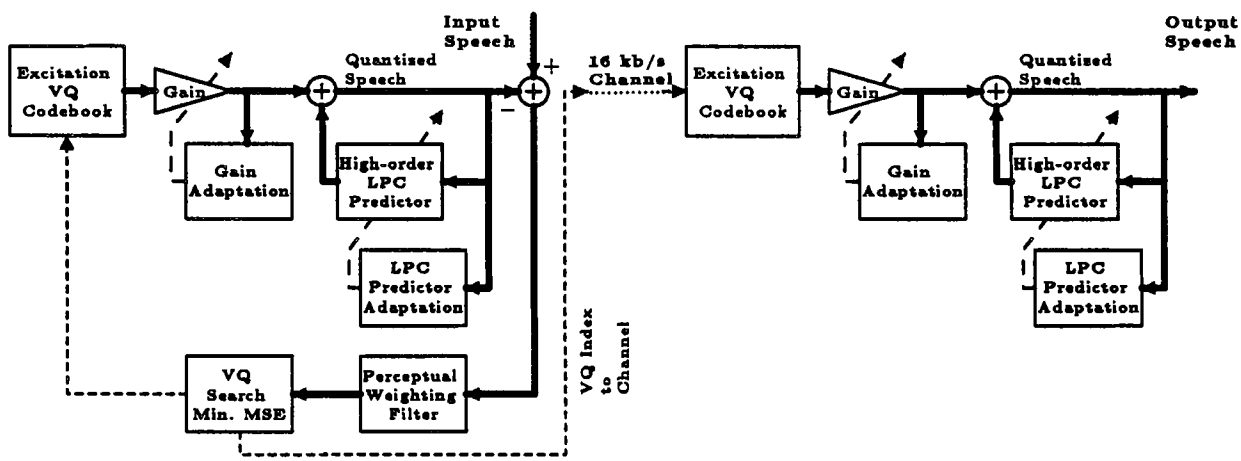


Fig. 2.13 LD-CELP encoder and decoder block diagram

Chapter 3 Backward Linear Prediction

This chapter reviews several approaches to backward linear prediction. The components of the backward linear prediction analysis are studied. To obtain the best prediction performance, an investigation of various schemes is conducted. The constraints considered by this thesis such as maximum delay of 2 ms, robustness to channel errors, and moderate complexity are used.

3.1 Introduction

The objective performance of the linear predictors is measured by the *prediction gain* (PG) which is expressed in dB and defined by

$$\mathrm{PG} = \log_{10} \frac{\hat{\sigma}_s^2}{\hat{\sigma}_e^2} = \log_{10} \frac{\sum_i s_i^2}{\sum_i e_i^2} \tag{3.1}$$

where $\hat{\sigma}_s^2$ is the estimate of the variance of the input signal and $\hat{\sigma}_e^2$ is the estimate of the prediction error variance. The index of the sum, i, is over long period of appropriate signal. Another objective measure which is often used is the *segmental prediction gain* (segPG). The prediction gain calculated for segments of 16 ms (128 samples) is calculated in dB. The average of these segment prediction gains over the entire speech file is the segmental prediction gain (segPG). The subjective performance (the listener's rating) may also be used by comparing the performance of the the linear predictors in a complete coder environment (e.g. APC or CELP).

A good redundancy removal prediction scheme must consider both near-sample (formant) and far-sample (pitch) correlations. Predictors are characterized by an

analysis frame which is used to adapt the prediction coefficients. The predictor is then applied to a block of samples. The length and shape of this frame as well as the overlapping of the analysis frames (the frame update rate) have to be chosen.

In the backward adaptive prediction analysis, where analysis frame precedes the block, parameters are not transmitted to the receiver and no buffering of the future samples is needed, allowing for lower processing delay. Estimates of the parameters are based on the past quantized speech samples which are available both to the encoder and decoder. Since the prediction coefficients are not transmitted in the backward prediction adaptation, an increase in the rate of update (increased overlapping of the analysis frames) does not result in an increased bit rate, but it does increase the encoder/decoder complexity. For a better tracking of the spectral changes highly overlapped analysis frames are desirable.

If a single predictor is used to remove both near-sample and far-sample redundancies, the choice of the length of the analysis frame has to be a compromise to fit the characteristics of the near and far-sample redundancies. Hence the single predictor, attempts to track both formant (spectral envelope) and pitch (spectral fine structure) variations. The desired effective analysis frame length for the near-sample correlations is around 5–10 ms (40–80 samples for sampling rate of 8000 Hz), while for the far-sample redundancies this length may be up to 50 ms (400 samples.) For large frames of this kind (400 samples), the stationarity assumption for the formant characteristic becomes invalid and the spectral formants are tracked less faithfully. The compromise analysis frame length has to be chosen. It is also necessary that the order of this all-pole single predictor be high enough so that the removal of the far-sample redundancies is possible.

The alternative to a single predictor is a configuration in which two cascaded predictors for near-sample and far-sample correlations are used. Ref. [19] suggests the configuration of Fig. 2.3 in which formant redundancy removal precedes the pitch redundancy removal (F–P open-loop configuration). Fig. 2.2 showed the closed-loop F–P alternative configuration.

The analysis frames for near and far-sample predictors, in the cascaded configuration of Figures 2.3 and 2.2 may have different effective size and update rate. Joint optimization methods of pitch and formant predictors are also suggested [19]. As a result better overall prediction gains are obtained at the cost of increased complexity. Due to some of the constraints of this work (backward prediction and noisy channel considerations) there is a drawback to the cascaded methods: channel errors result in a very high distortion which is due to the coupling of the two predictors. In the cascaded formant and pitch configuration, the pitch filter uses a backward adaptive pitch lag and coefficient values. The formant filter uses backward adaptive coefficient values. Erroneous lag estimates at the receiver can cause severe error propagation. Attempts to overcome this problem includes the work of Cuperman *et al* [44] in which the serial cascade structure is replaced by a "parallel" scheme to decouple the two predictor filters. The proposed method by AT&T [2] on the other hand, uses the first alternative of a single prediction with a very high prediction order. Higher processing power has made the alternative of a single high-order predictor possible. This chapter compares the two alternatives. Issues such as backward adaptation effects, complexity, windowing, and numerical ill-conditioning will be investigated.

The linear predictor filter may be implemented using either of the forms of transversal or lattice. The transversal structure uses the direct-form digital filter, the classical least-squares method is used to estimate the prediction coefficients $\{a_k\}$. The least-squares method in the backward adaptation case means the minimization of the mean energy in the error signal over a frame of speech samples prior to the "current" sample.

3.2 Auto-Correlation and Covariance Methods

The general analysis method of Fig. 3.1 maybe used to represent the windowing of data and/or error to estimate the prediction coefficients using the least-squares method [45, 46]. The speech input $s(n)$ is multiplied by the data window $w_d(n)$ to give $s_w(n)$, while multiplication of the error signal by the error window $w_e(n)$ results in the windowed error signal $e_w(n)$.

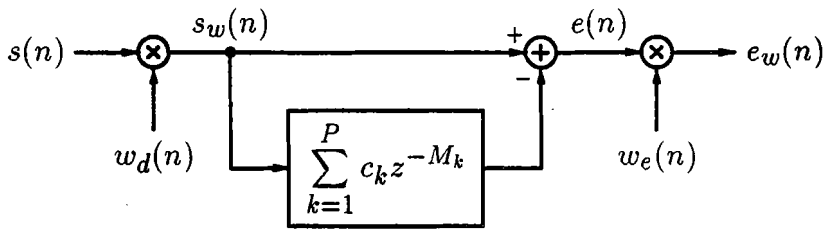


Fig. 3.1 Data window and error window

The formant and pitch predictors of figures 2.3 and 2.2 have the transfer function:

$$\begin{aligned} F(z) &= \sum_{i=1}^{N_f} a_i z^{-i}, \\ P(z) &= \sum_{i=1}^{N_p} b_i z^{-M_p-i-1}, \end{aligned} \tag{3.2}$$

where N_f and N_p are the number of formant and pitch predictor coefficients and M_p is the pitch lag which is updated along with the coefficients. If the pitch filter ($P(z)$) order is only one, the pitch lag estimation can only be a multiple of the sampling frequency. The multi-tap pitch filter $P(z)$ ($N_p = 3$ is often suggested) allows for pitch lag estimation with "interpolating" characteristics. Using the model of Fig. 3.1 with prediction order P, the error signal for prediction adaptation update maybe defined as:

$$\begin{aligned} e_w(n) =& w_e(n)e(n) = w_e(n)\left[x_w(n) - \sum_{k=1}^{P} a_k x_w(n-M_k)\right] \\ =& w_e(n)x_w(n) - w_e(n)\sum_{k=1}^{P} a_k x_w(n-M_k). \end{aligned} \tag{3.3}$$

The values M_k are used instead of k, to allow selection of arbitrary but distinct delays in the prediction filter. To employ the method of least-squares, the mean energy of the error has to be minimized:

$$E = \sum_{n=-\infty}^{\infty} {e_w}^2(n). \tag{3.4}$$

By setting the partial derivatives $\frac{\delta E}{\delta a_k} = 0$ for $k = 1, 2, \cdots, P$, the linear system of equations $\mathbf{\Phi a} = \mathbf{\Psi}$ results ($\mathbf{\Phi}$ is called the covariance matrix.). If the M_k's are

chosen as: $\{M_1, M_2, \cdots, M_P\} = \{1, 2, \cdots, P\}$ (grouped and uniformly spaced), the expanded covariance system of equations will have the form:

$$\begin{bmatrix} \phi(1,1) & \phi(1,2) & \cdots & \phi(1,P) \\ \phi(2,1) & \phi(2,2) & \cdots & \phi(2,P) \\ \vdots & \vdots & & \vdots \\ \phi(P,1) & \phi(P,2) & \cdots & \phi(P,P) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_P \end{bmatrix} = \begin{bmatrix} \phi(0,1) \\ \phi(0,2) \\ \vdots \\ \phi(0,P) \end{bmatrix}, \tag{3.5}$$

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

$$\phi(i,j) = \sum_{n=-\infty}^{\infty} w_e^2(n)\, x_w(n-i) x_w(n-j)\ . \tag{3.6}$$

The Auto-Correlation method will result when $w_e(n)$=1 for all n. One characteristic of the auto-correlation matrix $\mathbf{R}$ ($\mathbf{Ra} = \mathbf{r}$ replacing $\mathbf{\Phi a} = \mathbf{\Psi}$) is that it is positive-definite and Toeplitz. Hence the Levinson recursion solves the above system of linear equations of order P [17]. In the Covariance methods $w_d(n)$ is set to one for all n. Cholesky decomposition can be used for solving the linear system of equation where the (covariance) matrix is symmetric. The choice of the error window in the Covariance method and the data window in the Auto-Correlation method is the subject of the investigation in a later section in this chapter.

3.3 Modified Covariance Method

The alternative method of modified Covariance method suggested in Ref. [47, 48] guarantees the minimum phase property (this property is explained in Section 3.6.1.). The first two steps of this method are identical to the Covariance method. The $\mathbf{\Phi}$ matrix is expressed as a product of lower triangular matrix $\mathbf{L}$ and its transpose $\mathbf{U} = \mathbf{L^t}$ using the Cholesky decomposition. The resulted set of linear equations is then:

$$\begin{aligned} \mathbf{\Phi} a &= \mathbf{\Psi} \\ \mathbf{LUa} &= \mathbf{LY} = \mathbf{\Psi} \qquad \text{Step 1} \\ \mathbf{y} &= \mathbf{\Psi} L^{-1} \qquad \text{Step 2} \end{aligned} \tag{3.7}$$

As the next step, the partial correlation at delay m is found:

$$r_m = \frac{y_m}{\epsilon_{m-1}^2} = \frac{-\operatorname{sgn}(y_m) y_m}{\left[\psi_0 - \sum_{i=1}^{m-1} y_i^2\right]^{\frac{1}{2}}}, \tag{3.8}$$

where ϵ_{m-1} is the mean-squared prediction error at the $(m-1)$-th step of the prediction and y_m is the m-th component of the $\mathbf{y}$. Finally, given partial correlations $\{r_m, m = 1, 2, \ldots, P\}$, the prediction coefficients $\{a_i, i = 1, 2, \ldots, P\}$, are computed using the relationship:

$$a_m^{(m)} = -r_m \tag{3.9.a}$$

$$a_i^{(m)} = a_i^{(m-1)} + r_m a_{m-i}^{(m-1)}, \quad 1 \leq i \leq m-1, \quad m = 1, 2, \ldots, P. \tag{3.9.b}$$

Based on the residual energy ratios, the above method guarantees a minimum phase prediction error filter by ensuring that all zeros of the polynomial $(1 - F(z))$ are inside the unit circle. Although the method does not minimize an error criterion, its properties as explained in Ref. [49] makes it a more suitable parameterization scheme.

3.4 Lattice and Covariance-Lattice Methods

The Lattice formulation of the linear prediction has the property of taking into consideration both forward and backward prediction error, as it makes a distinction between them. Fig. 3.2 shows the lattice filter of order P. At each stage m (representing a m-pole model), $f_m(n)$ and $b_m(n)$, the forward and backward error (residual) signals are defined. $s(n)$ is the input speech, $r(n)$ is the final prediction error (residual) signal, and index n indicates the time-varying nature of the adaptations.

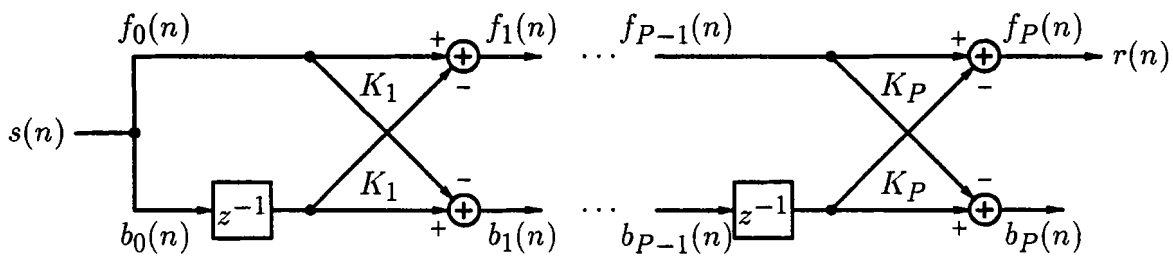


Fig. 3.2 Lattice filter of order P

Related to the Durbin recursion (in solving the linear system of equations in the previous sections), and in accordance with Fig. 3.2, one may express the forward and

backward error signals, $f_{m+1}(n)$ and $b_{m+1}(n)$, recursively as (at the stage m and time n):

$$f_{m+1}(n) = f_m(n) - K_{m+1} b_m(n-1), \tag{3.10.a}$$

$$b_{m+1}(n) = -K_{m+1} f_m(n) + b_m(n-1). \tag{3.10.b}$$

The initial and final conditions maybe written as:

$$\begin{aligned} f_0(n) &= b_0(n) = s(n), \\ f_P(n) &= f_{P-1}(n) - K_P b_{P-1}(n-1) = r(n). \end{aligned} \tag{3.11}$$

Other quantities often encountered in the Lattice methods are:

$$\begin{aligned} F_m(n) &= E\{f_m^2(n)\} \\ B_m(n-1) &= E\{b_m^2(n-1)\} \\ C_m(n) &= E\{f_m(n) b_m(n-1)\} \end{aligned} \tag{3.12}$$

where $E\{.\}$ is the expectation, operating on forward, backward, and cross forward/backward residual terms. The Burg algorithm minimizes the weighted sum of the forward and backward residuals over the length of an analysis window $w(n)$:

$$E_m(n) = \sum_{k=-\infty}^{n} w(n-k) e_m^2(k) \ , \tag{3.13}$$

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

$$e_m^2(k) = (1-\gamma) f_m^2(k) + \gamma b_m^2(k), \qquad 0 \le \gamma \le 1. \tag{3.14}$$

The minimization of the above weighted error with respect to the $K_m(n)$ leads to the update formulation:

$$\begin{aligned} K_{m+1}(n+1) &= \frac{C_m(n)}{D_m(n)} \quad \text{where} \\ C_m(n) &= \sum_{k=-\infty}^{n} w(n-k) f_{m-1}(k) b_{m-1}(k-1) \\ D_m(n) &= \sum_{k=-\infty}^{n} w(n-k) [\gamma f_{m-1}^2(k) + (1-\gamma) b_{m-1}^2(k-1)]. \end{aligned} \tag{3.15}$$

In order for the all-zero lattice of Fig. 3.2 or its equivalent direct form $1 - \sum_{k=1}^{p} a_k z^{-k}$ have a stable synthesis filter of the form $\frac{1}{1-\sum_{k=1}^{p} a_k z^{-k}}$, γ (the lattice

stability constant) has to be chosen to be 0.5 and $w(n)$ has to be a causal window function for which $w(n) \geq 0$ for $n \geq 0$ [50]. If $\gamma = 0.5$ is used, the reflection coefficient update expressed in terms of expectation quantities (instead of windowing) of Eqn. 3.12 will have the form:

$$K_{m+1}(n+1) = \frac{2C_m(n)}{F_m(n) + B_m(n)}. \tag{3.16}$$

The adaptation of parameters in the above Burg-Lattice method is performed on a sample-by-sample basis. The methods of Covariance-Lattice [51, 52] use the recursion formulations of the regular Lattice (Burg) method to obtain a more computationally efficient procedure. Using the recursion formulae of 3.10, quantities of 3.12, and the Burg update formulation of 3.16, one may arrive at:

$$\begin{aligned} F_m(n) &= \sum_{k=0}^{m}\sum_{i=0}^{m} a_k^{(m)} a_i^{(m)} \phi(k,i), \\ B_m(n-1) &= \sum_{k=0}^{m}\sum_{i=0}^{m} a_k^{(m)} a_i^{(m)} \phi(m+1-k, m+1-i), \\ C_m(n) &= \sum_{k=0}^{m}\sum_{i=0}^{m} a_k^{(m)} a_i^{(m)} \phi(k, m+1-i), \\ \phi(k,i) &= E\{s(n-k)s(n-i)\}, \end{aligned} \tag{3.17}$$

where $\phi(k,i)$ is the covariance (non-stationary auto-correlation) of the signal. Consequently, the reflection coefficients of the Lattice method (Equations 3.12 and 3.16) are updated using the estimated covariance (hence the name Covariance-Lattice). Using this method the computation cost will change from $5PN$ (Burg method) to $PN + 1/6P^3 + 3/2P^2$ (N being the analysis window length and P the predictor order).

Another important modification to the above calculation is the one suggested by Cumani [52]. In an attempt to fit the method better to fixed-point arithmetic, in effect all quantities are scaled for better numerical stability. The price is a slight increase in computational complexity. The detail description of Cumani method is

now described. First the following quantities are defined.

$$\begin{aligned} F_m(i,j) &= \sum_{k=0}^{m}\sum_{l=0}^{m} a_k^{(m)} a_l^{(m)} \phi(i+k, j+l), \\ B_m(i,j) &= \sum_{k=0}^{m}\sum_{l=0}^{m} a_k^{(m)} a_l^{(m)} \phi(m+1+i-k, m+1+j-l), \\ C_m(i,j) &= \sum_{k=0}^{m}\sum_{l=0}^{m} a_k^{(m)} a_l^{(m)} \phi(i+k, m+1+j-l), \\ \phi(k,i) &= E\{s(n-k)s(n-i)\}, \end{aligned} \tag{3.18}$$

with special cases of

$$\begin{aligned} F_m(0,0) &= F_m(n), \\ B_m(0,0) &= B_m(n-1), \\ C_m(0,0) &= C_m(n). \end{aligned} \tag{3.19}$$

Using the calculated covariance matrix (Eqn. 3.6) with entries $\{\phi(k,i)\}$, we initialize the algorithm with $m = 0$,

$$\begin{aligned} F_0(i,j) &= \phi(i,j), \\ B_0(i,j) &= \phi(i+1, j+1), \\ C_0(i,j) &= \phi(i, j+1). \end{aligned} \tag{3.20}$$

To obtain $F_m(n)$, $B_m(n-1)$, and $C_m(n)$, using the relationship between the reflection and prediction coefficients in Eqn. 3.9 and the above quantities the following recursions are derived and used.

$$\begin{aligned} F_{m+1}(i,j) &= F_m(i,j) + K_{m+1}^2 B_m(i,j) + K_{m+1}[C_m(i,j) + C_m(j,i)] \\ B_{m+1}(i,j) &= B_m(i+1, j+1) + K_{m+1}[C_m(i+1, j+1) + C_m(j+1, i+1)] \\ C_{m+1}(i,j) &= C_m(i, j+1) + K_{m+1}[F_m(i, j+1) + B_m(j, j+1)] + K_{m+1}^2 C_m(j+1, i) \end{aligned} \tag{3.21}$$

Results in Chapter 4 indicate that the Cumani method has excellent numerical properties in the context of backward adaptive high-order prediction (Use of Cumani algorithm for the high-order predictors is new.). Strobach [53] discusses this method further and generalizes it to the methods of *Generalized Residual Energy* (GRE) with the same numerical properties. In this general class (which also includes solution of the true Recursive Least-Squares (RLS) Covariance-Lattice estimation problem), coefficients are constructed completely anew at each step, avoiding round-off error accumulation. As later seen in Section 3.6.5, the computational complexity disadvantage

of Cumani algorithm ($O(P^3)$) can be overcome using Strobach Covariance-Lattice methods ($O(P^2)$). The class of such algorithms is also termed *Pure Order Recursive Ladder Algorithms* (PORLA).

3.5 Near and Far-Sample Redundancy Removal

The methods of removal of both near and far-sample redundancies (prediction) from the speech signal using prediction analysis methods maybe summarized as follows:

1– Combinational formant/pitch Single Predictor (possibly high-order):
  - 1(a) With equally spaced predictor taps
  - 1(b) With selectively spaced predictor taps

2– Separate pitch and formant Predictors:
  - 2(a) Sequential
  - 2(b) Jointly Optimized Sequential
  - 2(c) Decoupled Sequential of Ref. [44]

The range of pitch which has to be considered for the natural speech is from 64 Hz to 400 Hz. Using the sampling frequency of 8000 Hz, the required upper correlation lag to be considered is 125 samples (120 samples is commonly used). Typical male speech uses a F_0 (fundamental frequency) range of 80 to 160 Hz [17]. With average male and female F_0 at 132 and 223 Hz, the corresponding distance between pitch spikes in the time domain signal are 61 and 36 samples. Hence for the average pitch lag considerations, correlations corresponding up to 61 lags are needed.

The first method 1(a), attempts to remove the far-sample redundancies using the same predictor used to remove the near-sample redundancies by having the order of the prediction error filter $A(z) = 1 - F(z)$ very high. Ideally the order of the predictor has to be high enough to include past samples corresponding to the lowest pitch (around 120 samples).

As mentioned earlier, this approach has the following two problems. First is that the computational cost of such prediction analysis would be enormous and as shown from the experiments of the next chapter, there are numerical problems which

arise at such high orders. As a compromise solution to this problem, a prediction order of manageable computational complexity has to be chosen (e.g. 50). Order 50 allows "capturing" of the female speaker pitch and some portion of the male speaker pitch range. Hence the full range of pitch for all speakers (especially for male) which "extends" to delays as far back as 120 samples is not covered.

The second problem with the very high predictor orders is related to the correlation estimations and the analysis window selection. All prediction methods are based on some kind of correlation estimation for which the length of the analysis window must be large enough to provide a valid estimate of the far-sample correlations (2–3 times the longest lag 120, which gives a window lag of up to 400 samples long).

If the formant structure is stationary during analysis periods of this size (400 samples), this problem would not cause any loss in prediction gain. However since formant structure is not stationarity during periods longer than 100 samples, the use of long analysis windows results in poorer tracking of the changes in the formant structure. The commonly used analysis window size of around 20 ms for prediction order 10 proved to be a good compromise for most cases for high-order prediction (see the results of the experiments in the next chapter). However this compromise along with the numerical problems are the reasons why the prediction gains for male speaker only increases slightly at prediction orders around 60 to 70. One would have expected a higher rise (as is for the female speaker) in the prediction gain around these orders for male speakers. One reason that Ref. [2] probably chose order 50 is because they did not get significant prediction gains beyond order 50 while in fact it should be possible especially for male speakers.

The second method 2(b), may be used with different variations in selecting the spacing of the prediction taps. The goal is to place the taps where correlations are high and the effect of redundancy removal is high both in terms of prediction gain measure and the subjective quality measure. Experiments were carried out to investigate the allocation of taps. Results of some of these experiments are mentioned in the next chapter. One attempt allocated 20 taps (delays 1–20) for formant tracking and 30 taps for the far-sample correlation tracking. The difficulty in this case is how to

choose the location and spacing of the far-sample taps. One way is to select them with fixed positions. The goal would be to use the avaliable taps to either cover the whole range of delays from 20 to 120 lags or use them so as they are located at more probable pitch lags (e.g. 30 for female and 60 lags for male). Another way to allocate the taps would be to set them around the current pitch lag in such a way that the tracking of the pitch is adaptive. The windowing problem discussed earlier is one difficulty encountered for this approach. A consequence of the arbitrary tap-spacing method is that the auto-correlation matrix would not be Toeplitz. Another problem which occurs more strongly in the case of arbitrary tap-spacing methods is higher ill-conditioning.

The alternative configuration of two separate predictor (methods of $2(a)$, $2(b)$, and $2(c)$) for pitch and formant as in Fig. 2.3 and 2.2, does not have the windowing problem of the method of a single predictor. Selecting two different sizes of analysis window used for the two predictors is now possible. Better overall prediction gain at the cost of higher computational cost is obtained when the pitch and formant coefficients are joint optimized [19] ($2(b)$). As discussed earlier, due to the coupling between the two sequential predictors, the performance of such schemes is poor under channel errors. Some success is reported, when in a special configuration, the two predictors are decoupled [44] (method of $2(c)$). In the suggested configuration, a pole/zero model is used for the formant predictor. The adaptation of the pole-coefficients of the formant predictor (decoder of Fig. 2.3) is done not based on the synthesised speech but rather on the signal which does not have the pitch content reasserted (only formant zeros of the speech are reasserted)(see Ref. [44] for details).

3.5.1 Pitch Adaptation in Sequential Configuration

The adaptation of pitch parameters for the sequential pitch/formant configuration, described in detail in Ref. [46], is now briefly reviewed. Assuming that the pitch predictor's order is N_p, and the estimated pitch lag is M_p, the z-transform of the

predictor may be written as:

$$P(z) = \sum_{i=1}^{N_p} b_i z^{-M_p-i-1}. \tag{3.22}$$

The covariance formulation for the linear predictor using an analysis window size of N samples, results in the following set of equations:

$$\begin{bmatrix} \phi(M_p, M_p) & \cdots & \phi(M_p, M_p+N_p-1) \\ \vdots & & \vdots \\ \phi(M_p+N_p-1, M_p+N_p-1) & \cdots & \phi(M_p+N_p-1, M_p+N_p-1) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{N_p} \end{bmatrix} = \begin{bmatrix} \phi(0, M_p) \\ \vdots \\ \phi(0, M_p+N_p-1) \end{bmatrix}, \tag{3.23}$$

where

$$\phi(i,j) = \sum_{n=0}^{N-1} d(n-i)d(n-j)\ . \tag{3.24}$$

and $d(n)$ is the input signal to the pitch predictor (Fig. 2.3). The short form notation for the above set of equations is $\mathbf{\Phi c} = \mathbf{\Psi}$. The estimation of the pitch lag M_p is obtained by the minimization of the mean-squared error which has the form: $\varepsilon^2 = \phi(0,0) - \mathbf{c}^T\mathbf{\Psi}$. Consequently $\mathbf{c}^T\mathbf{\Psi}$ is maximized as $\phi(0,0)$ is constant. Using the assumption that the near-sample correlations are removed from the input $d(n)$, the simple approximation of

$$\mathbf{c}^T\mathbf{\Psi} \approx \sum_{m=M_p}^{M_p+N_p-1} \frac{\phi^2(0,m)}{\phi(m,m)} \tag{3.25}$$

results, where the off-diagonal terms in the matrix $\mathbf{\Phi}$ are neglected.

3.6 Other Issues in the Backward Linear Prediction

In carrying out the experiments of the next chapter, the following issues were found of importance:

1– Synthesis filter stability and the minimum phase property,
2– Optimality,
3– Window shape and size,
4– Computational issues related to the ill-conditioning,
5– Complexity (including parameter update rate),
6– Backward adaptation and quantization noise effects.

This section discusses the above issues in the various analysis methods. The experiments of the next chapter complement this discussion and some of the comparisons made here. The discussion and the experiments also show how factors like numerical stability and optimality effect each other. The choice of method of practice will also dependents on the real time implementation circumstances.

3.6.1 Synthesis Filter Stability/Minimum Phase Property

Lattice and Lattice-Covariance methods have the advantage that stability (minimum phase) of the resulting synthesis filter can be guaranteed. This is by ensuring that all reflection coefficients are all less than one. This is equivalent to guaranteeing that all zeros of the prediction error filter $1 - F(z)$ be inside the unit circle. Also in these methods it is possible to have a control over weighting of the forward and backward error in minimization of the mean squared error (although stability is only guaranteed under the weighting factor of 0.5 for forward and backward error). The Auto-Correlation method and the Covariance methods lead to stable (minimum phase) prediction error filters while the general form of arbitrary tap-spacing methods does not guarantee this [19]. The equally spaced tap delay of the form $M_k = kM_1$ is a special case for which the minimum phase property is true.

3.6.2 Optimality

In a general sense, various linear prediction methods use different optimality criteria. In a mean-squared forward error minimization sense however, the Auto-Correlation and Covariance methods are the only methods which give optimality. The methods of Lattice and Lattice-Covariance are only possibly sub-optimal [51]. One may argue that in the backward adaptation, the "optimality" is with regard to the analysis frame (and not with respect to the sample(s) for which analysis is actually used). Ref. [10] suggests that the simple white noise correction actually results in a better prediction gain by "de-tuning" the adaptation which is too "tuned" to the analysis frame (past samples).

3.6.3 Window Shape and Size Considerations

The shape and size of the two windows of Fig. 3.1 are important elements which effect the performance of the predictor. The selection of window also has cross-effects on other analysis components listed previously (e.g. the ill-conditioning and complexity). In the backward adaptation analysis, exponential windows which emphasis the more recent samples seem to perform better than commonly used Hamming and Rectangular windows. The length of the window (or the effective length of causal semi-infinite windows such as exponential windows) is chosen so that the correlation estimations contains enough number of samples to make the estimation valid. Once again the general rule of 20 ms window length maybe used where compromise has to be made between accuracy of estimation for long delay correlations and minimum averaging for short correlations. Computational complexity as well as stability of the resulted synthesis filter are also important when selecting the window $w(n)$. If the window is chosen so that $w(n) \geq 0$ for all $n \geq 0$, the sufficient condition for the stability of the synthesis filter is satisfied.

A class of windows obtained using the impulse response of casual pole/zero filters provide easy control over the shape of the window. At the same time their recursive implementation is advantageous (complexity and implementation considerations). These windows may also be regarded as exponential windows since the impulse response may be generated as the sum of exponential terms. The z-transform of the general filter of this type has the form:

$$W(z) = \frac{1 - \sum_{i=1}^{N_1} \alpha_i z^{-i}}{1 - \sum_{i=1}^{N_2} \beta_i z^{-i}}. \tag{3.26}$$

Examples of windows considered in the experiments of the next chapter are the simple

one-pole exponential and other windows with a small number of zero and poles:

$$\begin{aligned}
&1 \text{ pole}: \quad w_1(n) = \beta^n u(n), \quad W_1(z) = \frac{1}{1-\beta z^{-1}}, \\
&2 \text{ pole}: \quad w_2(n) = (n+1)\beta^n u(n), \quad W_2(z) = \frac{1}{(1-\beta z^{-1})^2}, \\
&3 \text{ pole}: \quad w_3(n) = \frac{(n+2)(n+1)}{2}\beta^n u(n), \quad W_3(z) = \frac{1}{(1-\beta z^{-1})^3}, \\
&2 \text{ pole} + 1 \text{ zero}: \quad w_{2pz}(n) = (\beta_1)^n - a(\beta_2)^n, \\
&\qquad W_{2pz}(z) = \frac{(1-a) + (a\beta_1 - \beta_2)z^{-1}}{1 - (\beta_1+\beta_2)z^{-1} + \beta_1\beta_2 z^{-2}},
\end{aligned} \tag{3.27}$$

where β is a number close to one ($0 < \beta \leq 1$).

The choice of the window shape and length (effective length in the case of exponential windows) is important. The effective length of the general window $w(n)$ is defined as [50]:

$$L = \frac{\sum_{n=0}^{\infty} w(n)}{\sum_{n=0}^{\infty} w^2(n)}. \tag{3.28}$$

When this definition is applied to the pole/zero exponential windows of Eqn. 3.27, the effective length of these windows (in samples) are found:

$$\begin{aligned}
1 \text{ pole}: \quad L_1 &= \frac{1+\beta}{1-\beta}, \\
2 \text{ pole}: \quad L_2 &= \frac{(1+\beta)^3}{(1-\beta)(1+\beta^2)}, \\
3 \text{ pole}: \quad L_3 &= \frac{(1+\beta)^5}{(1-\beta)(1+4\beta^2+\beta^4)}, \\
2 \text{ pole} + 1 \text{ zero}: \quad L_{2pz} &= \frac{[\frac{1}{1-\beta 1} - \frac{a}{1-\beta 2}]^2}{\frac{1}{1-\beta 1^2} + \frac{a^2}{1-\beta 2^2} - \frac{2a}{1-\beta 1 \beta 2}}.
\end{aligned}$$

The effect of the shape of the window maybe expressed as a function of what is termed as "measure of capture" in Ref. [50]. Although helpful, this factor does not give the full effect of the window shape in the predictor performance. This measure quantifies the amount of weight which window puts on the immediate past speech or the amount of signal "captured" under the dominant portion of the window and is defined by:

$$C = \frac{\sum_{n=0}^{L-1} w(n)}{\sum_{n=0}^{\infty} w(n)}.$$

L is the effective length of the window and C is the fraction of area under this effective length. For the 1 and 3-pole windows, C_1 and C_3 are 0.9865 and 0.904 and C for the Rectangular window is 1.

Slightly better results are obtained with higher order windows (2 or 3 pole over 1 pole window) at the expense of increased complexity. Fig. 3.3 shows the time response of the windows obtained using a 1 pole and a 2 pole + 1 zero model. Two windows obtained using 2 pole model with different effective lengths are shown in Fig. 3.4. To emphasis the effect of the far-sample correlations, the effective length of the window used with the 50th order predictor should be slightly longer than the one used for the 10th order predictor (20 ms versus 14 ms in this case).

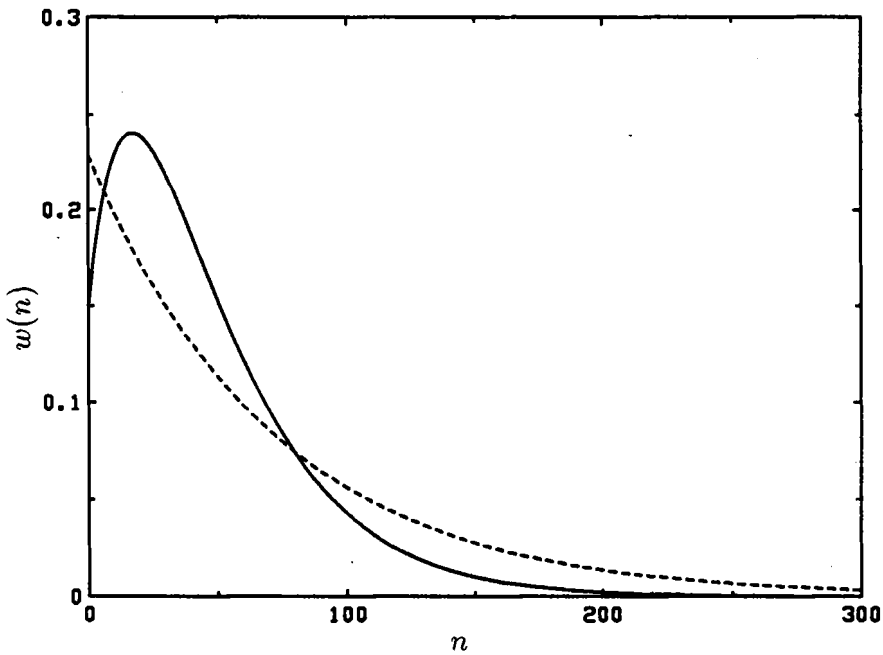


Fig. 3.3 1 pole window with $\beta = 0.986$ (dashed line) and 2 pole + 1 zero window with $\beta_1 = 0.97, \beta_2 = 0.95$, and $a = 0.85$ (solid line) (Effective length of the dashed window is 142 samples and of solid line is 97 samples. Two windows are normalized to the same area.)

The above windows may be applied to the data or error signal as was shown in Fig. 3.1. The exponential data-windowing of 1, 2, or 3-pole (Eqn. 3.27) was studied by Barnwell [54] in order to provide a recursive windowing method in generating auto-

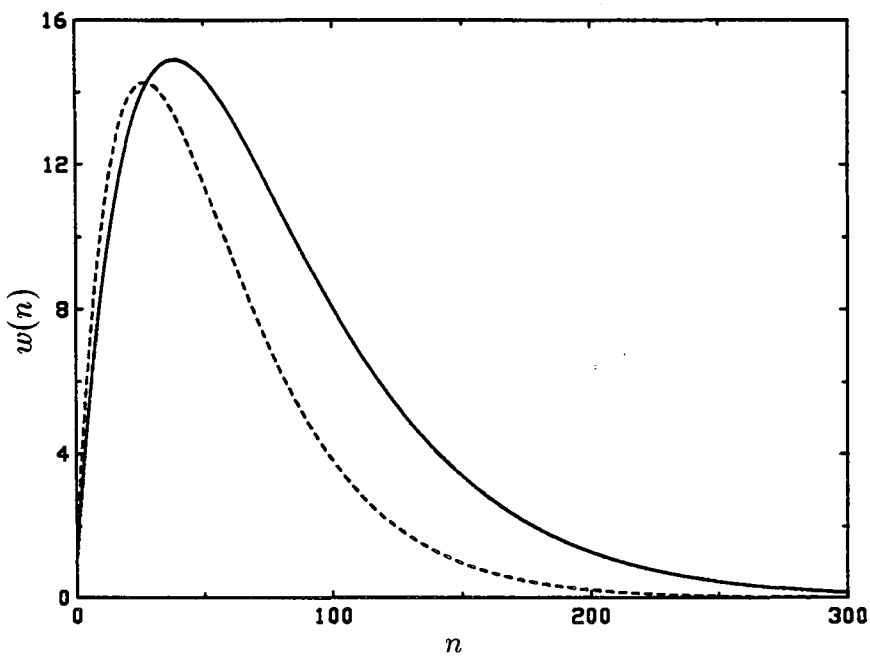


Fig. 3.4 2 pole window with $\beta = 0.965$ (dashed line) and 2 pole window with $\beta = 0.975$ (solid line) (Effective length of dashed window is 112 samples and of solid line is 158 samples.)

correlation lags in prediction analysis. For the real-time implementation, especially if high-order prediction analysis is used, the recursive feature is beneficial. Windows of Fig. 3.4 are examples of the Barnwell windows. As experiments of next chapter and Ref. [3] show, the Barnwell Auto-Correlation method used in the backward adaptive prediction analysis gives better subjective and objective prediction gains over the Hamming window. This is due to the fact that the exponential windows emphasis the immediate past samples more heavily. To calculate the auto-correlation terms, the speech signal is passed through specially-structured filter banks [54]. For the one pole window (Eqn. 3.27), each filter in the filter bank is a first order filter which is used to obtain one correlation term. For the two pole window, each filter is a third

order filter. The transfer function of such filters are as follows.

$$\begin{aligned}
&\text{1 pole (or 1 pole data window)} \quad W_m(z) = \frac{\beta^{m/2}}{1-\beta z^{-1}}, \\
&\text{3 pole/1 zero lag window} \quad W_m(z) = \frac{\beta^{m/2}[(1+m)+(1-m)\beta z^{-1}]}{(1-\beta z^{-1})^3}, \\
&\text{(or 2 pole data window)}
\end{aligned} \tag{3.29}$$

which is the z-transform of weighting window $w_m(n)$ applied to the lag-products $(s(n)s(n-m))$ with the subscript m referring to the lag number. For the two pole window, a preferred implementation of the above 3 pole/1 zero lag window is by cascading two 1-pole windows and one 1-pole/1-zero filter. The transfer function of these two filters are as follows

$$\begin{aligned}
&\frac{1}{1-\alpha^2 z^{-1}}, \\
&\frac{(m+1)\alpha^m - (m-1)\alpha^{m+2} z^{-1}}{1-\alpha^2 z^{-1}}.
\end{aligned} \tag{3.30}$$

This implementation has computational advantages for the real-time implementation [7].

The exponential windows are also applied to the error signal both in the Covariance method [55] and the Lattice method [50, 9]. The results of the next chapter (and of Ref. [9]) show a better performance when the exponential error window is used in the Lattice method and when Hamming window is used in the Covariance method [55]. Once again, this is due to heavier weighing of the immediate past error.

3.6.4 Computational Issues Related to the Ill-Conditioning

Due to gradual roll-off in the frequency response of the low-pass filter used before the Analog-to-Digital-Convertor (ADC) used on the speech signals, artificially low eigenvalues for the covariance matrix $\mathbf{\Phi}$ are produced ($\mathbf{\Phi a} = \mathbf{\Psi}$). These eigenvalues are related to the missing high frequency components in the sampled speech signal near half the sampling frequency. This condition creates an almost singular covariance matrix (with large eigenvalue spread) which results in non-unique solution for the prediction coefficients. The small eigenvalues produce artificially large valued

prediction coefficients which if used, results in noise enhancement [48]. Here for this matrix "ill-conditioning" the term "Physical ill-conditioning" is used [56].

Error accumulation resulting from the use of fixed-point arithmetic is another reason for ill-conditioning. An almost singular matrix can easily become singular if error is accumulated due to computer smaller memory word length. The term "numerical ill-conditioning" maybe used for this kind of condition. The high frequency correction procedure described in Ref. [47, 48] decreases the physical ill-conditioning. It uses a new covariance matrix which is obtained by adding another matrix proportional to the covariance matrix of the high-pass filtered white noise to the original covariance matrix. The solution considered, artificially introduces the missing high-frequency components. The (i,j)-th entry of a new covariance matrix $\widehat{\mathbf{\Phi}}$, and the i-th entry of a new correlation vector $\widehat{\mathbf{\Psi}}$, are defined by:

$$\begin{aligned} \widehat{\phi}(i,j) &= \phi(i,j) + \lambda \epsilon_{min} \mu(i-j), \\ \widehat{\psi}(i) &= \psi(i) + \lambda \epsilon_{min} \mu(i), \end{aligned} \tag{3.31}$$

where λ is a small constant (10^{-2}–10^{-7}), ϵ_{min} is the minimum value of the mean-squared prediction error, and $\mu(i)$ is the auto-correlation of the high-passed filtered white noise at delay i. The resulted new system of equation is then solved. The suggested high-pass filter (cancelling the effect of the band-limiting low-pass filter before ADC) has the following z-transform:

$$H(z) = \left[\frac{1}{2(1 - z^{-1})} \right]^2,$$

for which the white-noise auto-correlation terms are

$$\mu(i) = \begin{cases} \frac{3}{8} & \text{for } i = 0, \\ \frac{1}{4} & \text{for } i = 1, \\ \frac{1}{16} & \text{for } i = 2, \\ 0 & \text{for } i \geq 3. \end{cases}$$

The value of ϵ_{min} is obtained by the Cholesky decomposition of the original matrix $\mathbf{\Phi}$. If the solution to the matrix is obtained through Cholesky decomposition, this

method requires a much higher complexity (the decomposition has to be used twice, once to obtain ϵ_{min} from $\mathbf{\Phi}$ and once to solve $\hat{\mathbf{\Phi}}$). To avoid high computation cost of the above method, one may reduce the above white noise correlation technique to a scheme in which $\lambda\epsilon_{min}$ is a small constant (10^{-3} to 10^{-9}) and the only non-zero $\mu(i)$ is $\mu(0) = 1$ (or possibly $\mu(0)$ and $\mu(1)$). The effect of the white noise correction technique is a reduction in the dynamic range of the signal spectrum hence reducing ill-conditioning.

Numerical ill-conditioning is related to the particular algorithm and the computational procedure used. Error accumulation can be more severe for some methods. For example formulation of the algorithm in Cumani Covariance-Lattice method reduces the numerical ill-conditioning by resolving the problem anew at each step. The results of the next chapter for the high-order predictors show that the simplified technique (adding a percentage to the diagonal elements) sufficiently takes care of the inherent physical and numerical ill-conditioning.

As seen in the results of the next chapter, the accumulation of numerical errors seem to have more severe effect in recursive methods for obvious reason. Also the Covariance method showed more severe problem than the Auto-Correlation method. In the experiments, to "cure" the "physical" and "numerical" ill-conditioning, the simplified white noise correction method (adding a percentage to the diagonal elements) gave better results than the full white noise correction method (it is also less complex.).

3.6.5 Computation Complexity

Table 3.1 compares the computational complexities of the various methods. Examples for prediction order (P) and the resulting cost shows how the cost consideration may effect the choice of the computationally economical method for different orders. N is the analysis window size and M is the analysis and coefficient update rate (in samples). Note that the Cumani method with the best prediction gain performance has a very high computational cost. However, the computational disadvantage

Method	Computation order *	$P = 10$	$P = 50$
Auto-Correlation	$(PN+P^2)/M$	1700	10500
Covariance	$(PN+P^3/6+3P^2/2)/M$	1917	32583
Lattice (Burg)	$5PN$	8000	40000
Cov.-Lat. (Makhoul)	$(PN+P^3/2+2P^2)/M$	2300	75500
Cov.-Lat. (Cumani)	$(PN+4P^3/3-P^2)/M$	2833	172167
GRE (Strobach)	$(PN+3P^2)/M$	1900	15500

* Zeroth order terms have been neglected.

Table 3.1 Comparison of the computation costs ($N = 160$)

of the Cumani algorithm ($O(P^3)$) can be overcome using the Strobach Covariance-Lattice methods ($O(P^2)$).

In the backward adaptive prediction analysis, the parameter update may be done on a sample-by-sample basis or maybe done less frequently. When the update is not on a sample-by-sample basis ($M > 1$), estimated prediction coefficients based on an analysis frame of past samples are used for the current samples and a few future samples. At the cost of small degradation in prediction gain, less frequent update rate reduces the computational complexity. In obtaining a good prediction gain with an economical computational cost, one may choose to use a computationally expensive scheme but with an update rate which is not on a sample-by-sample basis.

Some of the analysis schemes mentioned in the earlier sections may not allow update rate less frequent than the sample-by-sample update ($M > 1$). Auto-Correlation, Covariance, and Covariance-Lattice methods have the advantage that analysis of the blocks are independent and may be done at any frequency (less often than sample-by-sample). This computational advantage is maintained even when a semi-infinite window is used, since most of the computational cost is for the solution of the linear system. For the Lattice analysis, an update with frequency less than sample-by-sample is not possible. As seen in the experiments presented in the next chapter, the degradation in prediction gain as a result of less frequent update than sample-by-sample, is minimal in comparison with the computational saving resulted.

3.6.6 Backward Adaptation Effect and Quantization Noise

The backward adaptation does not require transmission of any side information (prediction coefficients) and eliminates any analysis delay due to the buffering of the data for the analysis. The method however has two important drawbacks. First, the prediction gain is slightly reduced due to the fact that the prediction analysis is based on the quantized signal rather that clean signal. Second, the use of past speech for the estimation of prediction coefficients of a present sample makes the coefficients less fit for that sample ("stale" coefficients). This effect is even more pronounced when the update rate is less frequent.

Effects of the quantization noise in the backward adaptation may be modeled and studied by the incorporation of a simulated quantization noise model. For the purpose of this model, the following assumptions are made. (1) The quantization noise is white, (2) The quantizer noise energy depends on the energy of the prediction residual. (3) The quantizer Signal to Noise Ratio (SNR) (the ratio of the prediction residual energy σ_R^2 to quantization error energy σ_N^2) is fixed for an analysis frame. If SNR_Q represents the fixed signal to noise ratio of the quantization (e.g. 10 dB), one may derive the following relationship for the noise variance σ_N^2:

$$\begin{aligned} \text{SNR}_Q &= \frac{\sigma_R^2}{\sigma_N^2} = 10 \text{ dB} = 10, \\ \text{PG} &= \frac{\sigma_S^2}{\sigma_R^2}, \\ \sigma_N^2 &= \frac{\sigma_S^2}{\text{PG}} = \frac{\sigma_R^2}{\text{SNR}_Q} = \frac{\sigma_R^2}{10}. \end{aligned} \tag{3.32}$$

where PG is the prediction gain and σ_S^2 is the input speech signal variance. The quantization noise energy is added to the diagonal elements of the Covariance matrix or the auto-correlation matrix. The level of noise is iteratively adjusted for each analysis interval until the specified SNR_Q is achieved.

The algorithm outline is as follows. For each analysis block

1– Start with a high level of quantization noise (a fraction of the input signal energy, e.g. $\sigma_N = 0.8\sigma_S$),

2– Calculate $\mathbf{\Phi}$ or $\mathbf{R}$ matrix based on clean speech,

3– Add σ_N^2 to the diagonal elements and perform prediction and calculate the residual variance,

4– If SNR_Q has converged sufficiently to the fixed value (e.g. 10), stop, if not calculate the new σ_N and go to step 3.

In the next chapter, results of the several experiments using the above quantization model are presented. In these experiments the above algorithm converged rapidly (with less than 5 iterations). One observation made is that the quantization noise reduces the ill-conditioning as it has the same effect as the simple white noise correction technique.

Chapter 4

Experiment Results on Backward Linear Prediction

This chapter presents the experimental results on the backward adaptive linear predictor, discussed in the previous chapter. The primary model of the study in the forthcoming experiments is a prediction error filter ($A(z) = 1 - F(z)$), for which the backward adaptive prediction analysis uses the unquantized signal. The update rate of the prediction coefficients is not always on a sample-by-sample basis. In some of the later experiments, the quantization noise model of Section 3.6.6 is used in order to study the effects of the quantization noise. Many experiments are carried out with the goal of measuring the best prediction gain obtained over a speech file. For most of the experiments, the speech files OAKF8 and OAKM8 (Appendix A) were chosen as representative female and male utterances. The reason for this was that, as far as the experiments of this chapter were concerned, little difference was found among the results using different speech files (example files in Appendix B). Subjective ratings are postponed to the experiments of the future chapters where a full coder is implemented.

The focus of the experiments are the following issues:

- analysis schemes,
- window shape/size variations,
- remedies for the ill-conditioning, and
- near/far-sample configurations,
- quantization noise and backward adaptation effects,
- complexity (including frequency of parameter update).

Although an attempt is made to study each issue independently, results show the inter dependance among the above issues. Hence, as it becomes clear during the presentation of the results, the effect of each issue on the performance may not be easily judged in isolation.

4.1 Analysis Schemes

Experiments based on the many prediction analysis methods of the previous chapter were conducted. The performance of the predictor, measured in prediction gain obtained over a speech file, is plotted against the order of prediction analysis. Ill-conditioning is the main factor effecting the performance of prediction analysis at high-orders. In the simulations performed, a *counter* was used to keep track of the number of ill-conditioned cases (singular matrices and sometimes almost singular cases). To minimize this number, the simple white noise correction technique with the required parameter value is used. As the number of ill-conditioning cases increases in number, the effect is reflected in the downturn in the prediction gain curve. Some methods show a more "conditioned" characteristics. Nevertheless the ill-conditioning remains an inherent problem in the high-order prediction analysis. For the very high-order prediction analysis, white-noise correction technique always had to be used in order to make the results acceptable.

Fig. 4.1 shows the prediction gain comparison among three methods: Covariance, modified Covariance, and windowed modified Covariance methods. For orders up to 10–20, the performance difference among different methods is not great. However, for higher order prediction analysis, the prediction gain difference can be up to 4–5 dB. The Covariance method showed very high level of ill-conditioning for orders higher than 20. White-noise correlation technique "cured" the ill-conditioning to a small degree. The modified Covariance method is much "conditioned". Windowing of the error signal (windowed modified Covariance method) improved the prediction gains further. This windowing and ill-conditioning inter dependance is discussed in the next sections. As the downturn of the prediction gain curves shows, the ill-conditioning

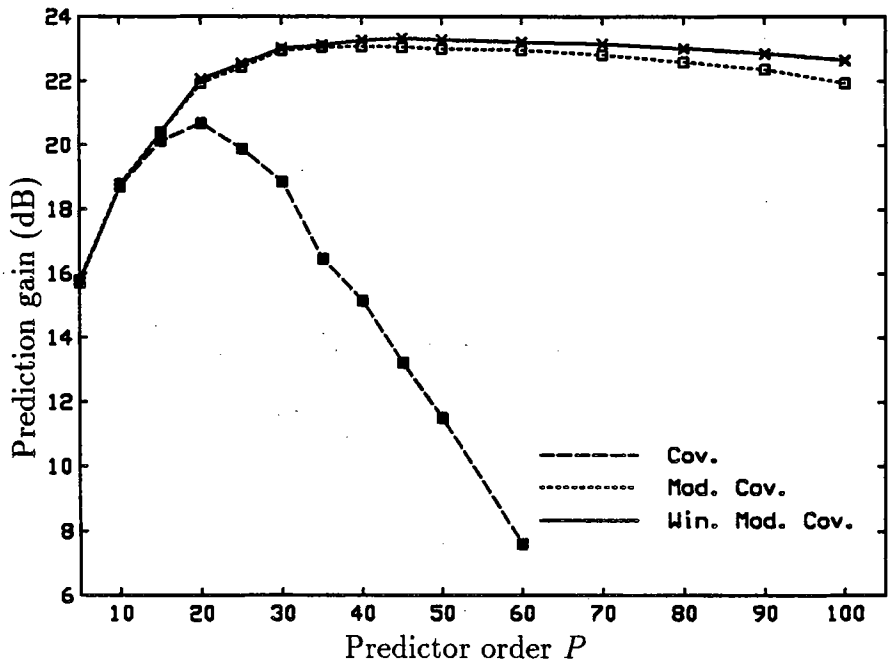
problem still persists. Even when the Singular Value Decomposition was used and the ill-conditioned cases where eliminated, the improvements of the ill-conditioning problem were small.

Fig. 4.2 shows the prediction gain comparison among the three methods which showed most promise: windowed modified Covariance, Barnwell windowed Auto-Correlation, and Cumani Covariance-Lattice methods. The Cumani Covariance-Lattice method is the most conditioned method and hence results in the best prediction gains at very high prediction order. As seen, the windowed modified Covariance method performs almost as well at high-orders up to 50. The prediction gains obtained here (using Covariance-Lattice method) are 2–3 dB higher than the previously reported prediction gains in the literature. Other considerations such as computational complexity and quantization noise effects have to be used for the final choice of the analysis method. Best performance with the less computationally expensive method of Auto-Correlation is obtained when Barnwell windowing is used. Use of other window types is discussed in the next section.

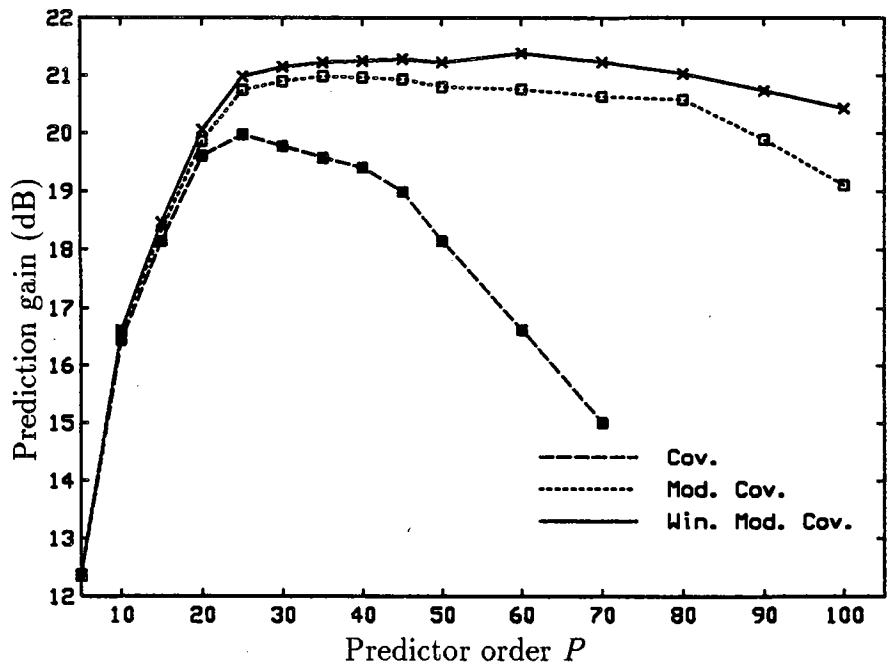
As shown in Fig. 4.3, compare to the windowed modified Covariance method and Cumani Covariance-Lattice method, the Lattice method does not perform well at higher order prediction. It also does not have the low computational complexity advantage of the Auto-Correlation method at high-order prediction. The iterative computation procedures of the Lattice method may be the reason for the accumulation of error and degradation of prediction gain. The well conditioned characteristics of the Cumani method is clearly related to its computation procedure while the reason for the better conditioned characteristics of the modified Covariance method is not obvious.

4.2 Window Shape/Size Variations

Data and error windows and the significance of the various size and shape of windows were discussed in Chapter 3. For the case of high-order predictors, the choice of window is resulted from the compromise made to capture near and far-sample



(a) Female

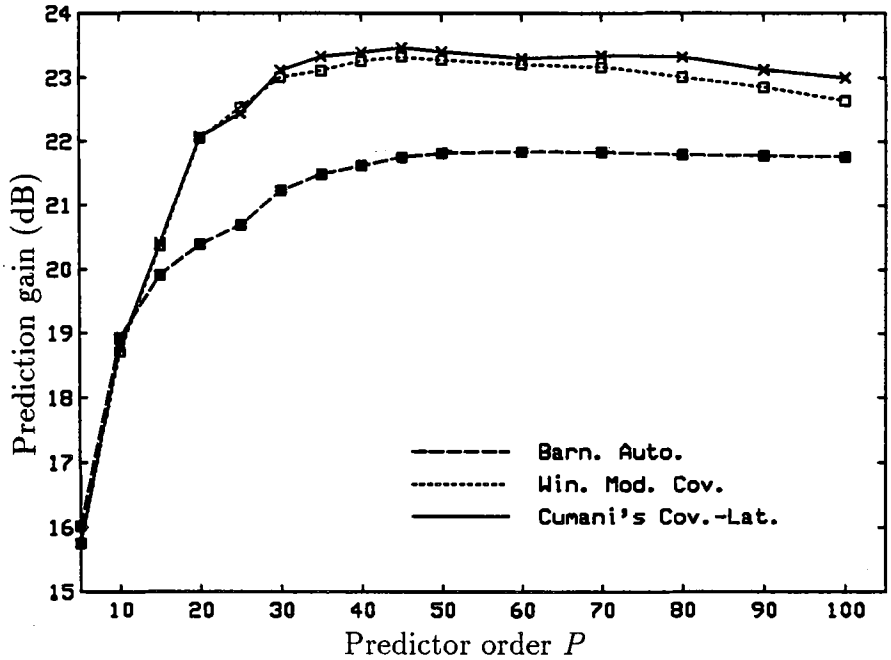


(b) Male

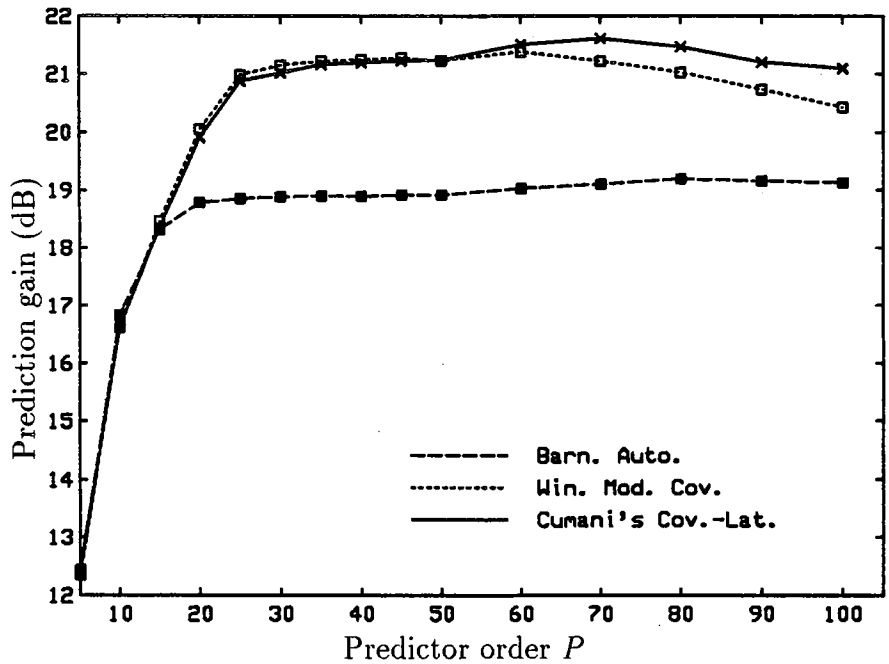
Fig. 4.1 Comparision of performance as a function of predictor order for Covariance method with the modified Covariance method and the windowed modified Covariance method

characteristics. In the Auto-Correlation method, the windowing is performed on the data and the estimation of the correlation lags is based on the assumption that the prediction order P is much smaller than the window size N. This assumption does not hold for the high-order predictors (e.g. P=50, N=160). Also the violation of the assumption of $N \gg P$ in the high-order predictors in Auto-Correlation analysis, causes more severe damage for male utterances and is seen in the experiment results. In Fig. 4.2 shows that the Auto-Correlation method performs worst than the Covariance and Covariance-Lattice methods. The gap between the prediction gain between the Auto-Correlation method and the other two methods is more for the male speech.

The results of experiments indicate that the optimum window size is not much larger than 20 ms or 160 samples. Windows with effective size of around 20–22 ms (160–180 samples) produced best results. For the higher order prediction 22 ms effective size is more suitable. This is to better capture of the far-sample correlations. The resulted prediction gain is not always higher for these slightly longer windows. However, the subjective performance as a result of use of such windows in complete coder experiments of future chapters is noticeable. The lack of accuracy for higher lag correlation estimations, resulted from violation of the assumption $N \gg P$, will cause a great deal of degradation in the prediction gains. The near-sample non-stationarity for windows much larger than 20 ms, caused more harm than the removal of the far-sample redundancies. The better results obtained using the Barnwell window compared to the Hamming window shows that the "long tail" of the exponential Barnwell window allows for some of the "capturing" of the far-sample correlations. As well the Barnwell windows emphasis the more recent samples and hence their use is advantageous in the backward adaptive analysis. Fig. 4.4 shows the prediction gain comparison among Barnwell or Hamming window Auto-Correlation methods and windowed modified Covariance method. An improvement of 1–2 dB in prediction gain is resulted through use of Barnwell window over Hamming. The windowed modified Covariance method is shown for reference. This method uses Hamming window on the error signal. As seen in Fig. 4.1, the use of error window also improves the performance.



(a) Female



(b) Male

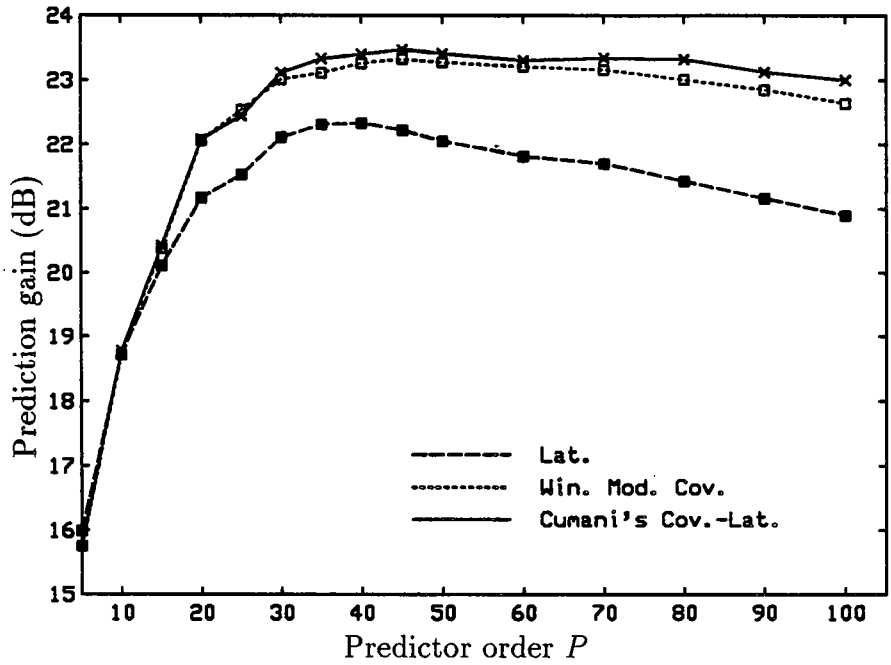
Fig. 4.2 Prediction gain comparison as a function of predictor gain among three analysis methods: Barnwell window Auto-Correlation, windowed modified Covariance, and Cumani Covariance-Lattice methods

Although the use of higher order windows produced slightly higher prediction gains in the experiments performed, these differences were neither significant (within 1 dB) nor always consistent. For example the prediction gain of the second order Barnwell window was not always better than the first order one (comparable prediction gain). Nevertheless, as seen from the results of next chapter, in a complete coder the second order exponential window produces better subjective quality. Taking into consideration both complexity and performance, as a compromise the second order window is recommended (over first and third order ones). Similar conclusion can be made about windowed Covariance and Lattice methods.

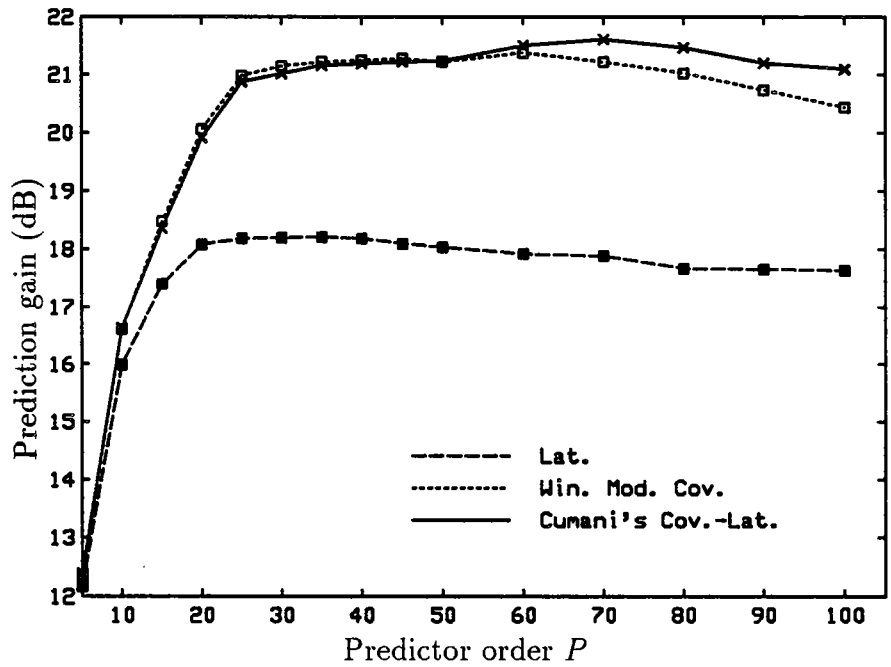
4.3 Remedies for the Ill-Conditioning Problem

The results of experiments on the remedies used to overcome the ill-conditioning problem is already seen in the previous sections. These experiment results showed how ill-conditioning can be cured to some degree using the "remedies" discussed in Chapter 3. The more complex white-noise correction technique of Section 3.6.4 did not cure the ill-conditioning to satisfaction, while the simple white-noise correlation technique worked very well. Fig. 4.5 shows the effectiveness of this method.

The shape and size of window effects the ill-conditioning problem. Hence the discussion of the previous section in turn must be tied to the ill-conditioning problem. In the experiments of Fig. 4.4, the ill-conditioning was less severe for the Auto-Correlation method than for the other methods. Also for the Auto-Correlation method, when Barnwell window was used instead of the Hamming window, a higher white-noise correction factor was required (about 100 times more to cure the increased ill-conditioning). The effects of the window shape on the dynamic range of correlation values may be one explanation for the above observations. A higher dynamic range can result in higher ill-conditioning and vice versa. The rectangular windowed Covariance method shows great ill-conditioning at high-orders (Fig. 4.5). As seen in this figure, the ill-conditioning "kills" a great deal of gains obtained from the far-sample correlations. Ill-conditioning is almost cured in the windowed modified Covariance method.



(a) Female



(b) Male

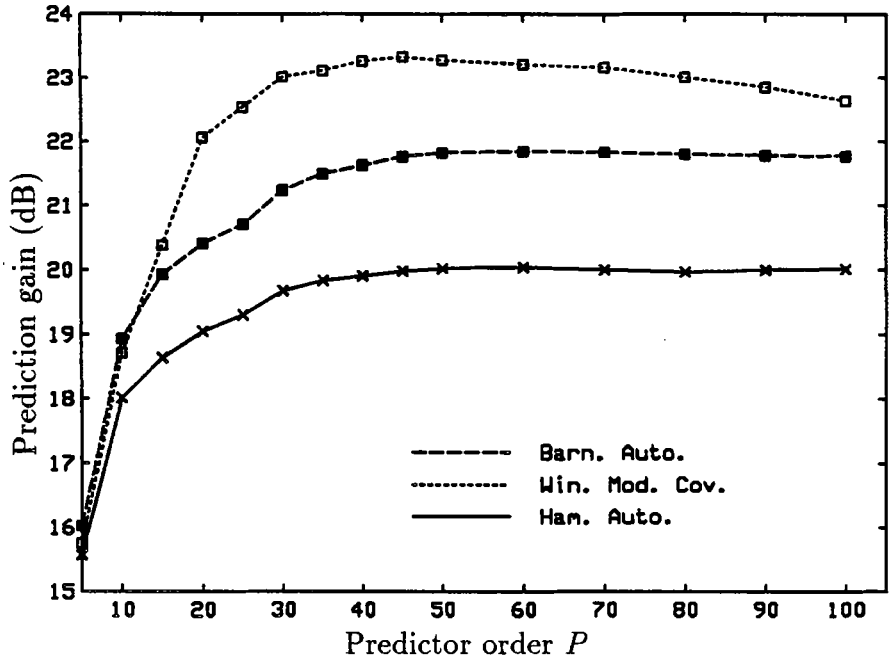
Fig. 4.3 Comparision of performance as a function of predictor order for Lattice method with the Cumani Covariance-Lattice method and the windowed modified Covariance method

From the observations made up to this point, one may conclude that, although the high-order Auto-Correlation analysis has the advantage of a better conditioned solution, it only partially exploits the high-order correlations. At the cost of higher computational cost, the Covariance and Covariance-Lattice methods are better suited for higher order prediction analysis.

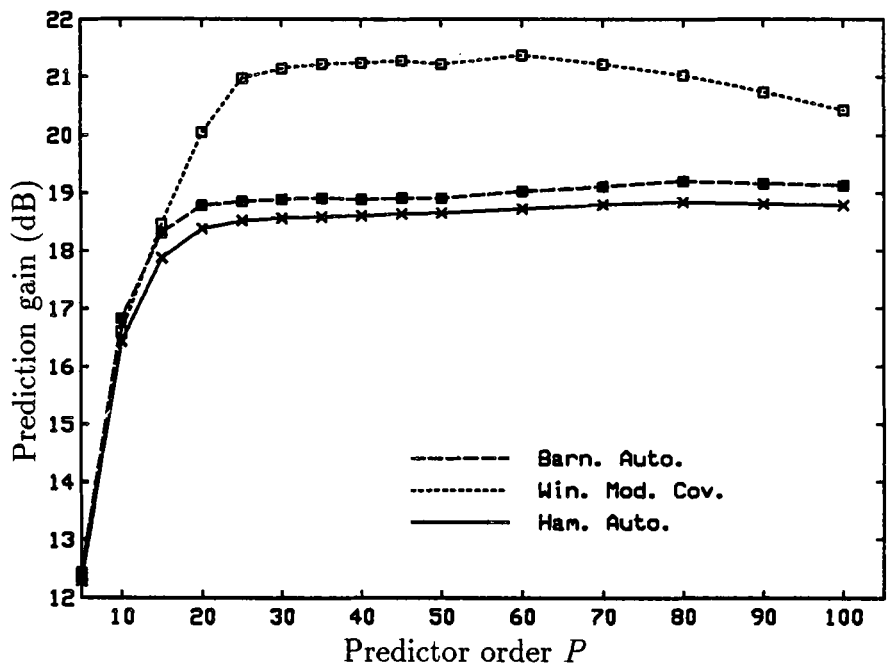
4.4 Near/Far-Sample Configurations

The effectiveness of the far-sample redundancy removal using high-order prediction analysis may be demonstrated by a comparison between LPC spectrum estimated using a predictor of order 10 and one using a high-order (50) predictor. Fig. 4.6 shows this comparison for prediction analysis using Auto-Correlation method with a Barnwell window of effective size 160 samples (for comparison purposes the same window size is used, although it is better to use window with a shorter effective length for order 10). The selected frame is during the phoneme /α/ in sentences OAKF8 and OAKM8 (Appendix A). Fig. 4.7 shows the segmental prediction gain (segPG), over the whole sentence for prediction orders 10 and 50. As seen in the figure, the pitch capture, as a result of use of high-order predictors, is better in the female utterance. This is due to the fact that the average pitch period of the female (approximately 30 samples in this case) is well within the capture range of the 50th order predictor, while the average pitch period of the male (in this case approximately 80 samples) is not well captured by the 50th order predictor.

As mentioned in the previous chapter, the sequential formant and pitch (F–P) predictor configuration may be used to remove near and far-sample redundancies. A relatively low-order formant predictor is needed for this configuration (e.g. order 10). The pitch adaptation method of Section 3.5.1 is used to estimate the coefficients of an order 3 pitch predictor. During the unvoiced segments of the speech, the pitch predictor stage does not produce any prediction gain for the obvious reason. This stage may even be harmful due to the backward adaptation. This was encountered in one experiment and negative segPG's were observed. A threshold on the energy



(a) Female



(b) Male

Fig. 4.4 Comparision of performance as a function of predictor order for Barnwell window or Hamming window Auto-Correlation method with the windowed modified Covariance method

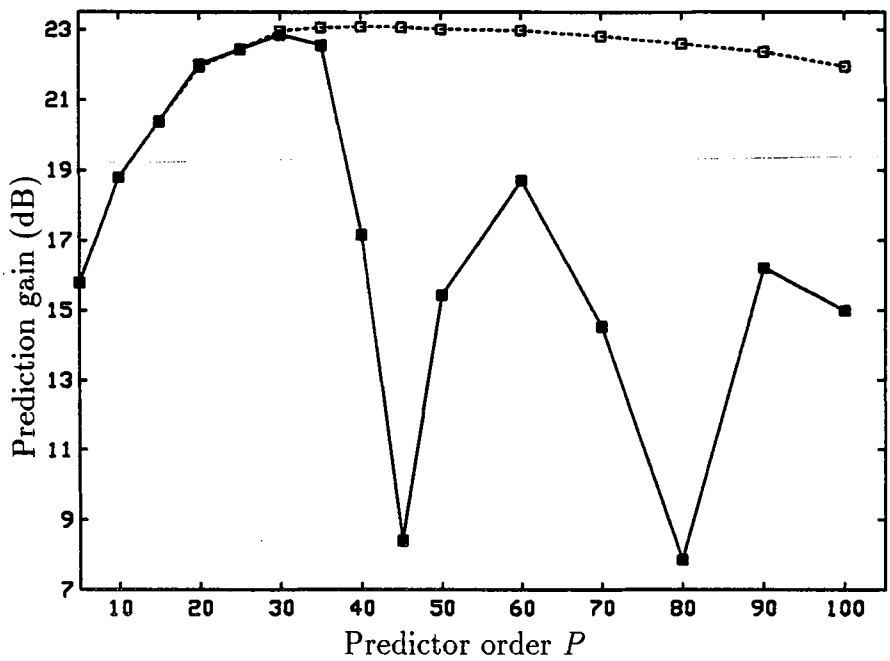
of the frame of signal is used to "turn off" the pitch predictor during these speech segments. Nevertheless additional prediction gain is obtained as a result of removal of pitch redundancies in the voiced segments.

As seen in Table 4.1, the single high-order predictor outperforms the F–P configuration (update rate is every 5 samples). The gap between the single high-order predictor and the overall prediction gains in the F–P configuration is more than the pitch prediction gain in the F–P configuration. For the high-order predictor, if an additional pitch predictor stage is used, further prediction gains do not occur. This can be seen from a comparison between results in Table 4.2 (where formant predictor order is high) and Table 4.1 (where formant prediction order is relatively low). Another illustration of segmental prediction gains comparison between F–P configuration (order 10+3) and a single high-order (50) configuration is shown in Fig. 4.8 (corresponding to Table 4.1). Also segmental prediction gains comparison between F–P configuration of order 50+3 and a single high-order (50) configuration is shown in Fig. 4.9 (corresponding to Table 4.2). †

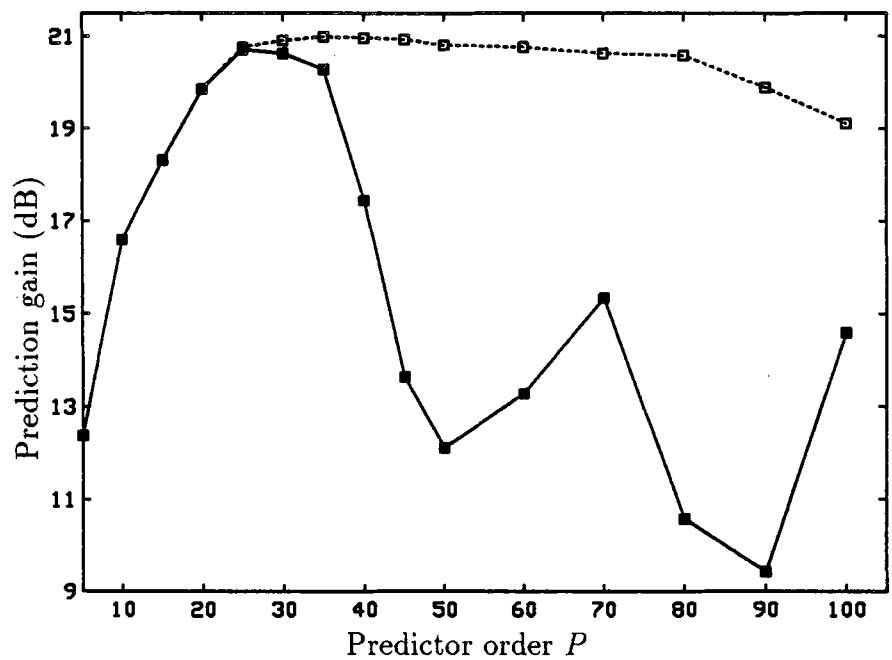
In general, the subjective performance improvements resulted from use of a single high-order predictor agree with the above conclusions. Results presented in the future chapters show how a "crisper" coded speech is obtained when the single high-order predictor is used in a complete coder. It is important to consider subjective measure comparisons when the gap between the performance of the two methods under consideration is small. This is because the small objective prediction gain improvements (prediction gain in dB) do not always correspond to the similar subjective improvements. For example in the case of occasional small negative pitch prediction gains (see Table 4.1) the coded speech still sounds better.

---

† Since the results presented in the Tables 4.1 and 4.2 are obtained using the Auto-Correlation method, the previous conclusions about disadvantage of this method for the male utterances is once again observed: For male utterances unlike the female ones, pitch prediction gains can be obtained even after the 50th order formant predictor. The same goes for the results in Figures 4.8 and 4.9, where the pitch capture for the female speech using the single high-order predictor is apparent (consistent higher segPG) while for the male speech the segPG is not consistently higher. As expected, this disadvantage was not found for the Covariance and Covariance-Lattice methods.



(a) Female



(b) Male

Fig. 4.5 Comparision of performance as a function of predictor order for white-noise correction factor of 0.0 (solid line) and 10^{-7} (dashed line) for the Covariance method

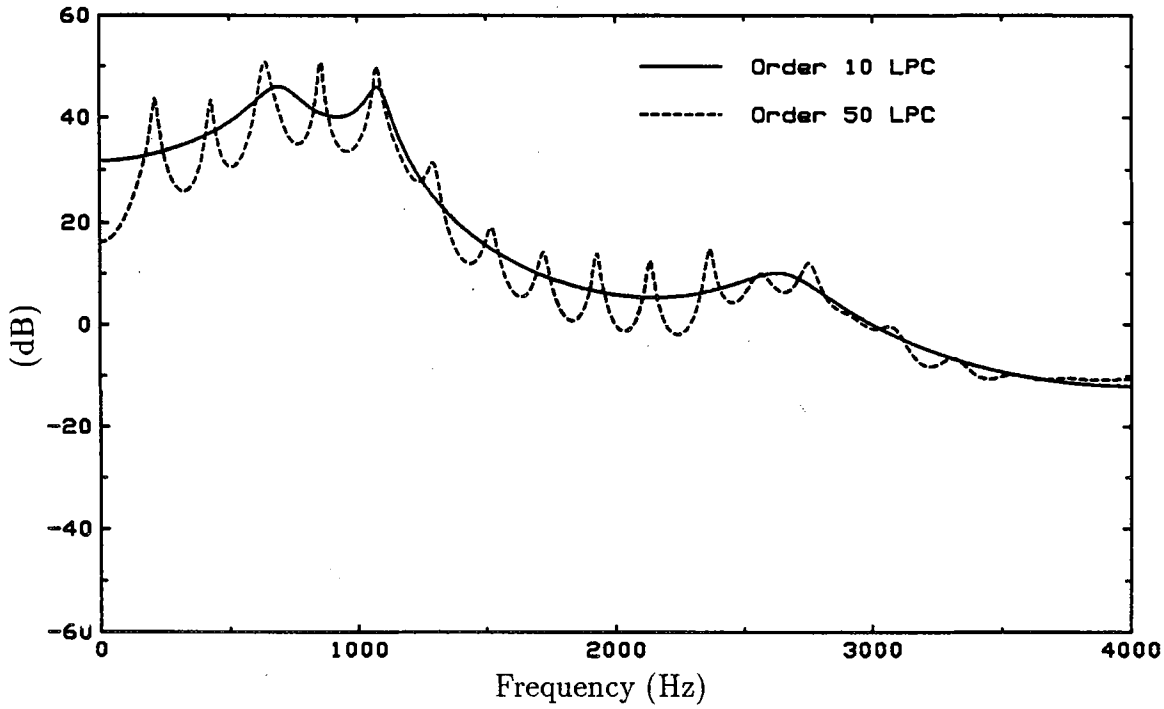
For a complete comparison between the two configurations of single high-order predictor and F–P configuration, other factors such as ill-conditioning and higher complexity of the high-order predictors have to be taken into consideration (recalling that the full capture of male and female pitch range requires very high-order predictors (at least 80).).

In Chapter 3, the general idea of arbitrary tap spacing for the transversal predictor filter was discussed. The purpose of this exercise is generalization of schemes of better removal of formant and pitch redundancies. The simulation results were not promising. Further investigation is needed for a conclusive understanding of arbitrary tap spacing analysis configurations. From the simulations performed on various tap spacing suggestions of Chapter 3, several conclusions can be made:

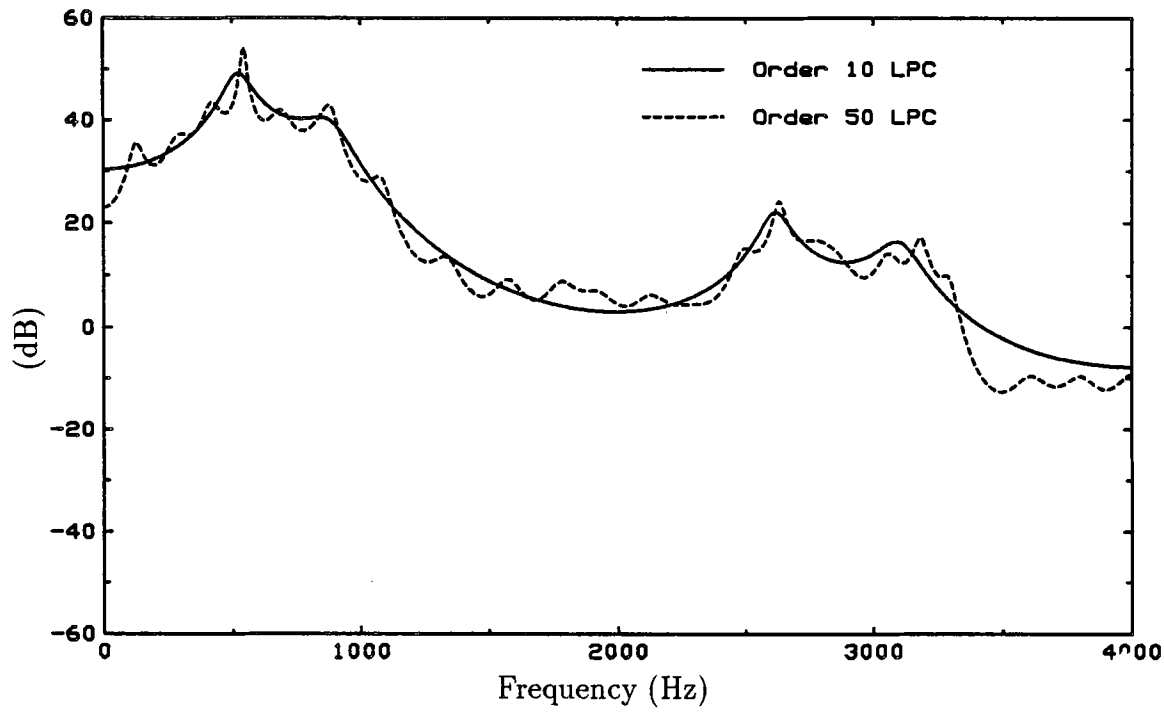
- arbitrary tap spacing in general can be helpful;
- if the pitch tap locations is adaptive, the task of identification of locations for these taps is not easy;
- as a result of the arbitrary tap spacing, ill-conditioning becomes even more severe;
- identification of tap locations, is computationally expensive.

4.5 Complexity (Frequency of Parameter Update)

Results of experiments studying the effects of less frequent update of coefficients in different prediction analysis schemes are presented in this section. Fig. 4.10 shows how in the Auto-Correlation method with Barnwell windowing, the less frequent update rate effects the prediction gains. The loss of prediction gain due to less frequent update is not significant for update rates up to 40 samples. The effect of update rate is not as clear for the male sentence (due to poor pitch tracking capability of Auto-Correlation method). Pitch tracking is more sensitive to frequency of update in comparison with the formant tracking. For example if update rate is changed from every 20 samples to every 5 samples in a F–P configuration, the prediction gain increase is mostly in the pitch portion. This may also be seen from the results of Ref. [57] where a method of tracking pitch without an increase in the update rate is suggested.

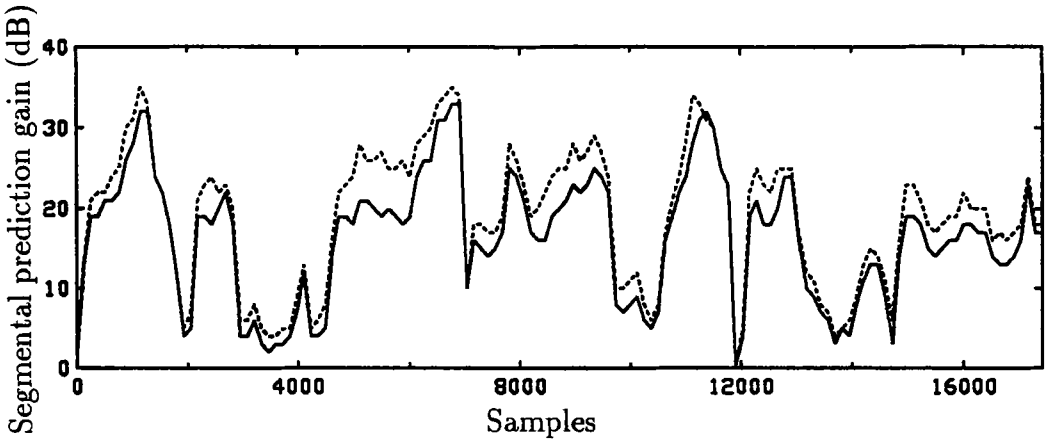


(a) Female

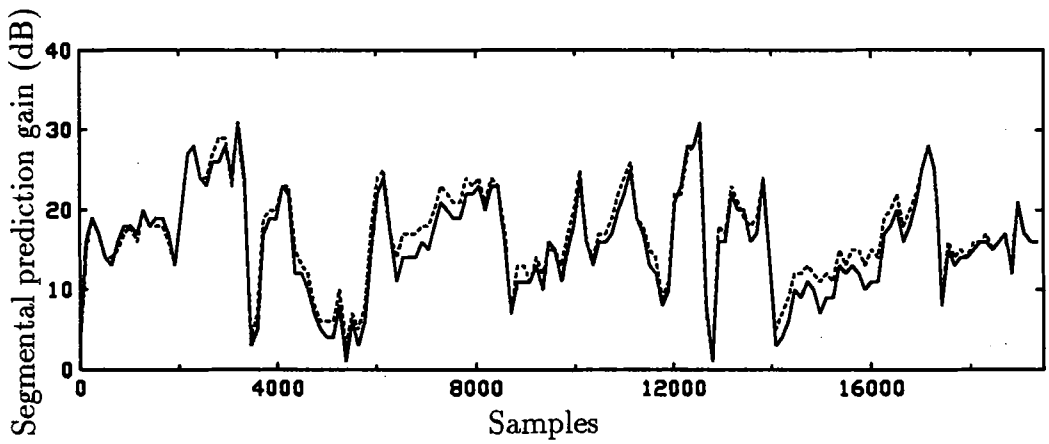


(b) Male

Fig. 4.6 Comparision of LPC spectrum estimation using order 10 and order 50 prediction analysis



(a) Female



(b) Male

Fig. 4.7 SegPG comparison between speech analysed using the prediction of order 10 (solid line) and of order 50 (dashed line) using Auto-Correlation method

4.6 Quantization Noise and Backward Adaptation Effects

The model used in all experiments up to this point is a prediction error filter ($A(z) = 1 - F(z)$) with the backward adaptive prediction analysis based on the clean signal. If the model of Section 3.6.6 which includes the quantization noise effects is considered, some of the above results may get effected. Experiments were carried out to investigate some of these possible effects. The conclusion made is that, as a result of inclusion of quantization noise model, the results presented in this chapter do not change significantly.

method	sentence (gender)	formant gain dB	pitch gain dB	overall gain dB
F–P sequential	OAKF8 (female)	19.1	1.5	20.6
Single high-order		–	–	22.2
F–P sequential	OAKM8 (male)	16.9	−0.4	16.5
Single high-order		–	–	18.7
F–P sequential	CATF8 (female)	13.9	1.0	14.9
Single high-order		–	–	17.6
F–P sequential	CATM8 (male)	12.8	1.7	14.5
Single high-order		–	–	15.3

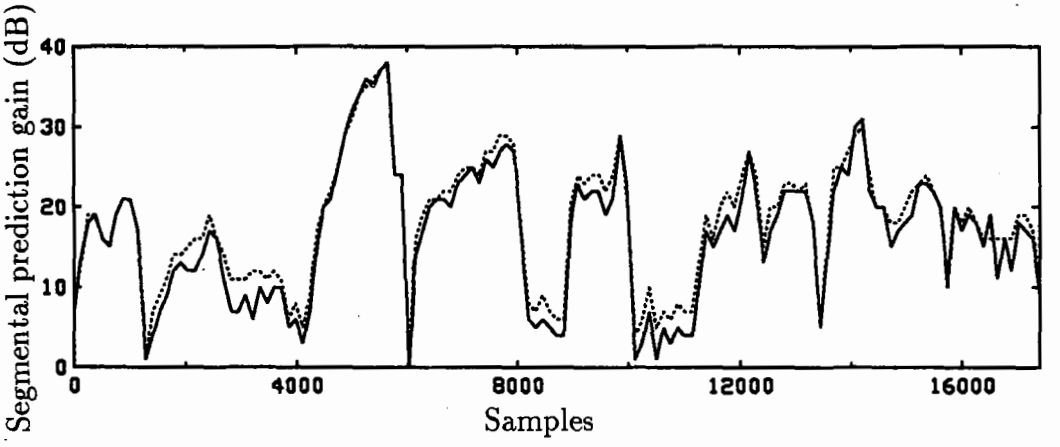
Table 4.1 Comparison of prediction gains between F–P configuration (orders: 10+3) and single high-order predictor configuration (order 50)

method	sentence (gender)	formant gain dB	pitch gain dB	overall gain dB
F–P sequential	OAKF8 (female)	22.2	-0.5	21.7
Single high-order		–	–	22.2
F–P sequential	OAKM8 (male)	18.7	-0.5	18.2
Single high-order		–	–	18.7
F–P sequential	CATF8 (female)	17.6	−1.0	16.6
Single high-order		–	–	17.6
F–P sequential	CATM8 (male)	15.3	1.3	16.6
Single high-order		–	–	15.3

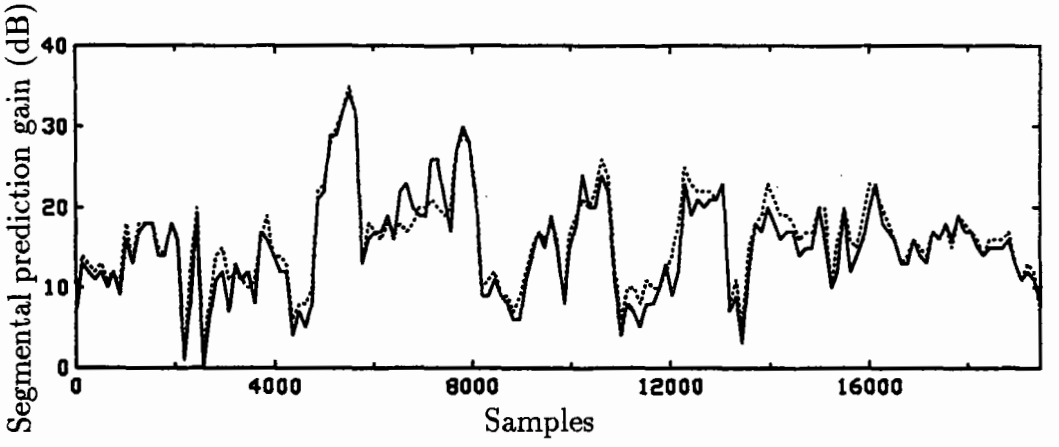
Table 4.2 Comparison of prediction gains between high-order F–P configuration (orders: 50+3) and single high-order predictor configuration (order 50)

Table 4.3 shows the prediction gain comparison among orders 10, 30, and 50 for the Auto-Correlation method in the two possible models (with and without the quantization noise). Although the resulted prediction gains for the model with quantization noise are generally lower than the model without quantization noise, the conclusion made about the high-order predictors do not change (increments in the prediction gain stay similar). One observation made was that in the model with quantization noise, the ill-conditioning is reduced. This is due to the fact that the effect of the quantization noise is similar to the simple white-noise correction technique.

Although the quantization noise model is a useful simulation tool and can give some preliminary idea about quantization noise effects, some of the assumptions made by this model (see Chapter 3), may not always hold. For example, for the case of lower bit rate coders where the assumption of white quantization noise does not hold, the model is less valid. More realistic results may be obtained by studying the backward adaptation in a complete coder environment. This however is a much more time consuming simulation task.

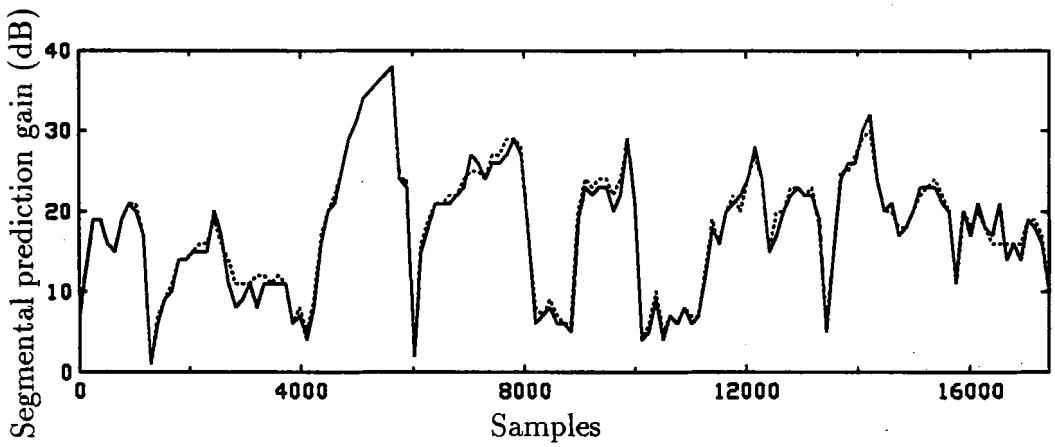


(a) Female

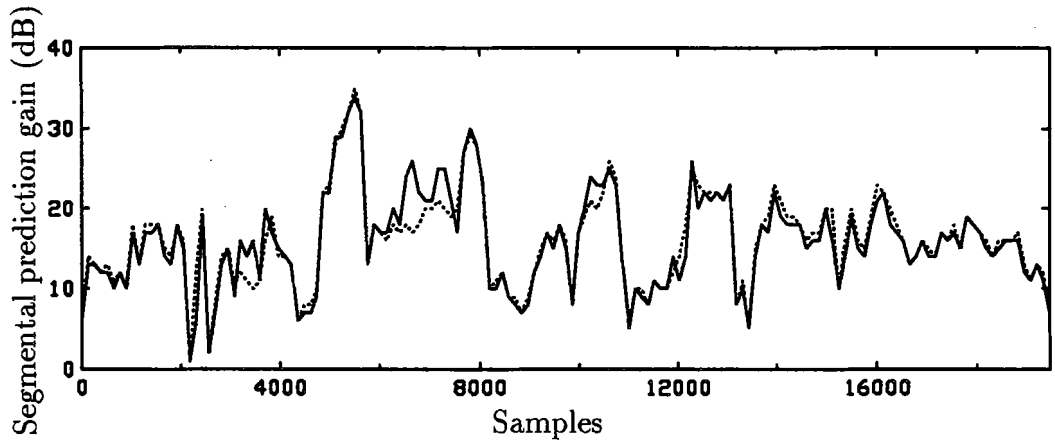


(b) Male

Fig. 4.8 SegPG comparison between speech analysed using the F–P configuration of order 10+3 (solid line) and single high-order configuration of order 50 (dashed line) (Auto-Correlation method))



(a) Female

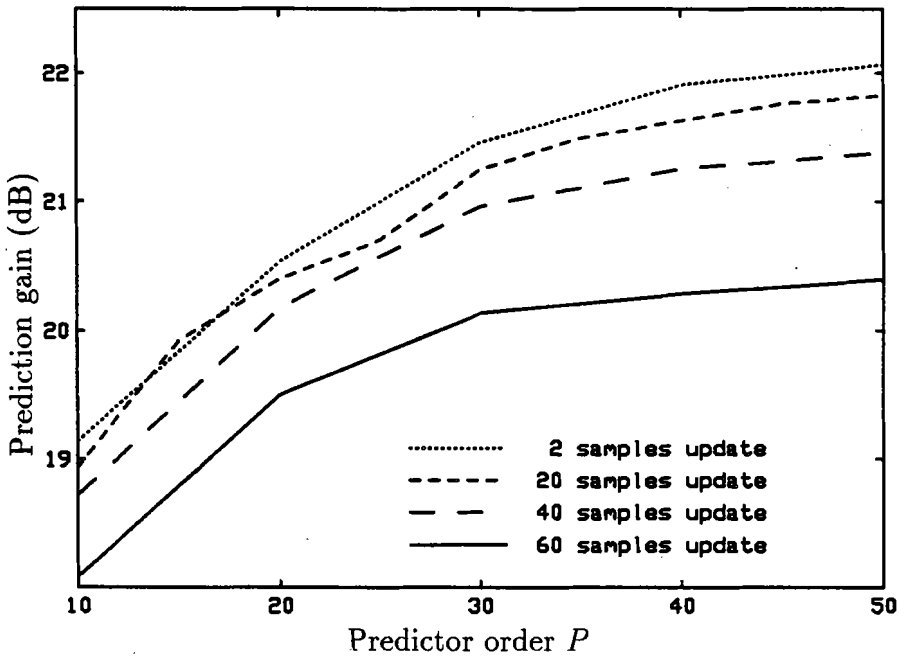


(b) Male

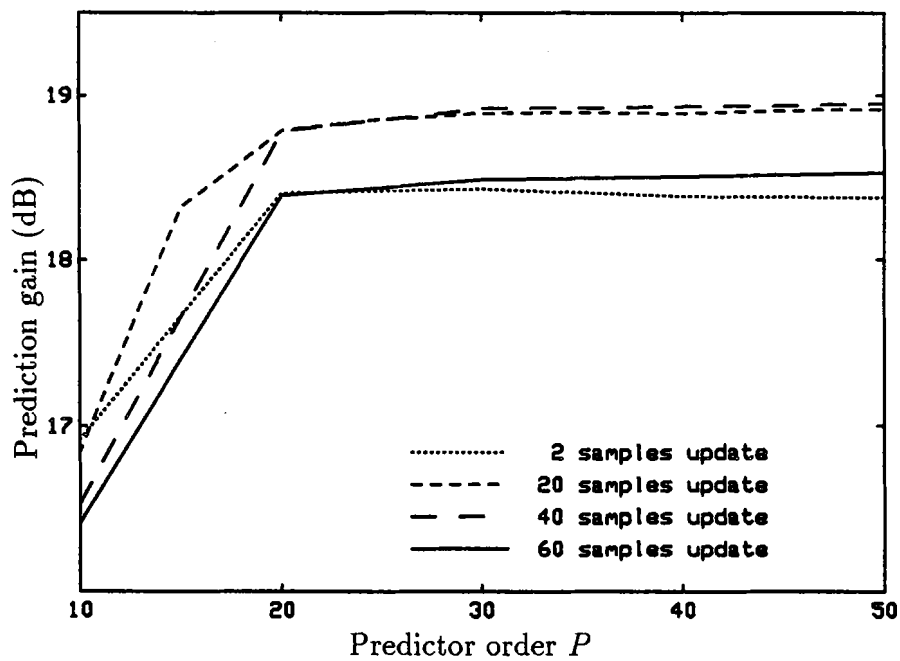
Fig. 4.9 SegPG comparison between speech analysed using the F–P configuration of order 50+3 (solid line) and single high-order configuration of order 50 (dashed line) (Auto-Correlation method)

Analysis based on	sentence (gender)	order 10 gain dB	order 30 gain dB	order 50 gain dB
1- Clean speech	OAKF8 (female)	18.93	21.25	21.83
2- Quantized speech		18.51	20.27	20.83
1- Clean speech	OAKM8 (male)	16.84	18.89	18.91
2- Quantized speech		16.74	18.51	18.17

Table 4.3 Comparison between prediction analysis based on clean signal and prediction analysis based on simulated quantized signal



(a) Female



(b) Male

Fig. 4.10 Comparision of performance as a function of predictor order for the Auto-Correlation method with different update rates

Chapter 5 LD-TREE and LD-CELP: Algorithms and Simulation Results

The basic ideas of the low-delay coding using LD-TREE and LD-CELP coders along with various components of the two coders were introduced in Chapter 2. Chapters 3 and 4 focussed on the performance of the formant and pitch predictor component, studying various prediction schemes. In order to study and compare the performance and characteristics of complete LD-TREE and LD-CELP coders, simulations based on the original configurations [8, 2] were carried out. †

In this chapter first, a more detailed description of the two coders algorithms is outlined. Then, results of the study of the comparison between the two coders, their similarities and differences along with the simulation results are presented. The first goal is to compare and study the performance of the two coders. Second objective is to analyse the particular characteristics which makes the two coders different from one another. The final goal is the improvement of the performance of the coders with the view of bringing down the bit rate below 16 kb/s.

The simulations are done in FORTRAN language programs on a VAX-8600 computer. Single precision floating-point arithmetic is used in almost all the programs. In a few places in the program where precision is crucial, double precision arithmetic is utilized. The comparison between the two coders are performed using objective

† As it was mentioned in Chapter 1, for the LD-CELP, the detailed algorithm description and subsequent modifications reported later [3–7] are also considered. In particular the changes to parameter specification and use of Barnwell window instead of Hamming window are included.

measures of Signal to Noise Ratio (SNR) and segmental SNR (segSNR). The segSNR measure is defined in a similar fashion to segPG using blocks of 16 ms (Section 3.1). Some informal subjective tests were also used.

The LD-TREE and LD-CELP algorithm are presented in Sections 5.1 and 5.2. In Section 5.3, a comparison between the structure and methods used in the two coders, along with the simulation results comparing them are presented. Section 5.4 outlines methods by which the two coders performance may be improved. Use of a better prediction scheme, is applicable to both coders. For the tree coder, a training method for the innovation dictionary is devised and outlined. Use of a better gain adaptation strategy is discussed. A new structure for the LD-TREE, more suitable for the higher-order prediction filters, is suggested. The issues related to the components and various schemes discussed in Chapters 2 and 3 are also considered and conclusions are made. Suggestions to improve the robustness of the LD-TREE to the channel errors are presented.

5.1 LD-TREE Algorithm

Chapter 2 introduced delayed decision tree coding and other concepts used by the LD-TREE coder in brief. The original LD-TREE [8, 9] is a tree coder based on the generalized APC structure of Fig. 2.10 (closed-loop structure). In order to match the components of the LD-TREE to the LD-CELP original version, the postfiltering and pitch predictors were eliminated from the original LD-TREE. Other than this difference, the LD-TREE used in this study is kept identical to the original one in [8] and the same simulation programs are used. It was also suggested in Chapter 2 that the block diagram of the coder can be represented in a open-loop structure (usually used in the CELP coders) to show its resemblance to the LD-CELP coder. The resulted structure was shown in Fig. 2.12. This configuration is similar to the LD-CELP coder structure of Fig. 2.13. The equivalence of the two LD-TREE structures holds only if a similar perceptual weighting filter is used by both coders.

The algorithm description of the LD-TREE coder as it was originally presented in Ref. [8] is now reviewed. First the main ideas are summarized. Then with the help of

the algorithm block diagram in Fig. 5.1, the steps are shown. The block diagram of the closed-loop APC structure used by the algorithm is shown in Fig. 2.10. Occasionally, references to the equivalent open-loop structure shown in Fig. 2.13 are also made.

The block responsible for the search of the best innovation sequence in Fig. 2.12, is identified as the (M,L) Tree search using minimum MSE distortion criteria. This block which is a part of the quantizer block Q in Fig. 2.10 is expanded in Fig. 5.1 to show the steps of tree coding quantization. The corresponding block in the LD-CELP coder is identified as the VQ search also using the minimum MSE distortion criteria. The methods of tree and codebook delayed decision coding discussed in Chapter 2, are used by these blocks to select one of the innovation entries in the innovation dictionary (LD-TREE) or the VQ codebook (LD-CELP).

The (M,L) tree search algorithm explained in Chapter 2 (M=16, L=8) [58], is a MSE distortion minimization procedure in which at each time instant n, the maximum number of paths kept in contention is M=16 and the number of nodes in each path is limited to L=8. Once each of the maximum M path are extended and the accumulated error (MSE distortion) for these paths are updated, decision is made to select the path with the minimum (MSE) accumulated error. The index of the root of this path (a 2 bit code) is transmitted to the decoder. This tree decision making may be characterized as a *sliding block* structure in which every time a speech sample arrives, as a result the tree "grows" new leaves but the root gets "trimmed". This means that once a decision is made, the 8-sample-long (L=8) sliding block moves, bringing in the new leaves of the extended M path in the sliding blocks and leaving the winning root of the path with the least accumulated (MSE) distortion out. The new maximum M path kept in contention are only the ones which stem from the released root.

In the LD-TREE algorithm, distinction is made between the innovation tree and reconstruction tree. The innovation tree is populated from a stochastic dictionary with each node having a unique path map which maps the location of the node in the tree to the entry in the stochastic tree. The one-to-one corresponding nodes in the reconstruction tree are populated with the resulted reconstructed speech samples. Each value is obtained by scaling the innovation sample $y(n)$, to obtain the gain-scaled

innovation sample $e_q(n)$ ($(e_q(n) = \sigma(n)y(n)$).

The stochastic innovation tree nodes are populated from a Laplacian random number dictionary of size 2^k (in this study 2^k=4096). The adaptation of the 8th order formant predictor $F(z)$ in the LD-TREE coder is using the Lattice method. The reflection coefficients are converted to the direct form before using in the prediction filter. The use of noise feedback filter $N(z) = F(z/\lambda)$ results in the simpler perceptual weighting filter of the form in Eqn. 2.5.

The path map for each of the extended leaf nodes in the innovation tree is as follows. Since the branching factor of the tree is 4, each branch has a two bit binary index, and each path map may be uniquely identified as the concatenated sequence of these two bit indices representing the branch numbers from the root to the node. In the multi-search (M,L) algorithm M such path with length L are kept in contention. The stochastic dictionary of size 2^{D} ($\mathrm{D} < 2 \times \mathrm{L}$) is addressed using the D least significant bits in the above path map sequence.

In the multi-search scheme the filtering operations along each path using this structure need separate memory. For example each of the M path j (with $j = 1, \cdots, \mathrm{M}$) require M set of filter coefficients $\{a_i^j\}$ for the filter $F(z)$. Similarly, the noise shaping filter $N(z)$ would have separate memories when is updated along various paths. As discussed in Chapter 2, it is possible to go around this large memory and complexity requirement, by a delayed update of the filter coefficients. This delayed update actually results in a better prediction gain for the backward adaptations in $(1 - F(z))$ filter of Fig. 2.10. As a consequence of the delayed update, it is not necessary to keep separate sets of memory for the M path. As it was mentioned in Chapter 2 and is seen in Fig. 2.12, the filtering operation at time n for the extended nodes uses the coefficient sets L samples back ($\{a_i(n - \mathrm{L})\}$. It is easily seen that the filtering operation for the released root would have coefficients 2L samples back ($\{a_i(n - 2\mathrm{L})\}$ in Fig. 2.12).

As the gain for the nodes along each path are updated using Eqn. 2.9, each extended node would require a separate gain update. The gain-scaled innovation becomes available by multiplying this gain by the unscaled-innovation sample, popu-

lating the innovation tree (gain-normalized innovation dictionary). Note that the gain update strategy of Eqn. 2.9 has the advantage that there is little memory requirements. If the P-th order logarithmic gain predictor adaptation strategy of LD-CELP were to be used, the memory requirements (keeping long memories along maximum of M path) and complex analysis (separately for each path) would not be practical. It is however possible to to adopt a delayed gain update strategy. This means that gain adaptation is along the already released path (single) with a small cost in quality degradation . The effect of such scheme needs further investigation.

In response to the speech sample input at time n, for each path j, there is an output $r^j(n)$ for the prediction error filter $(1 - F(z))$ in Fig. 2.10 and Fig. 5.1. The superscript j indicates the dependance of the output on the path number j. This dependance would not exist for the delayed update case (Fig. 5.1). At this point we need to obtain the outputs of the noise-feedback filters $N(z)$ for the various paths. This value, $x^j(n)$ is then added to the corresponding residual values $r^j(n)$ to obtain $e^j(n)$. The MSE distortion of the quantization for each of the extended node is the square of the difference between this value and the gain-scaled innovation value of the corresponding node in the innovation tree. The gain-scaled innovation value is updated for all extended nodes ($e_q^j = \sigma^j(n)y^j$ where y^j is the node innovation value, and $\sigma^j(n)$ is the gain for the node). The accumulated MSE distortion for the tree structure is the sum of MSE distortion for the nodes along a particular path. The MSE minimization procedure in the analysis-by-synthesis tree structure means trying all extended innovation samples using the above procedure and finding the one which results in the lowest accumulated MSE distortion. Among the 4M extended paths, the one with the lowest relative accumulated error is chosen (root code is released). Since only the relative errors between various path is important, only relative accumulated errors from the released root is accumulated. After the release of the index of the winner root, trimming of the tree is done by discarding the paths which do not stem from the released root and by only keeping a maximum of M paths with the lowest accumulated error.

To present a better description for the algorithm, steps of the LD-TREE algorithm

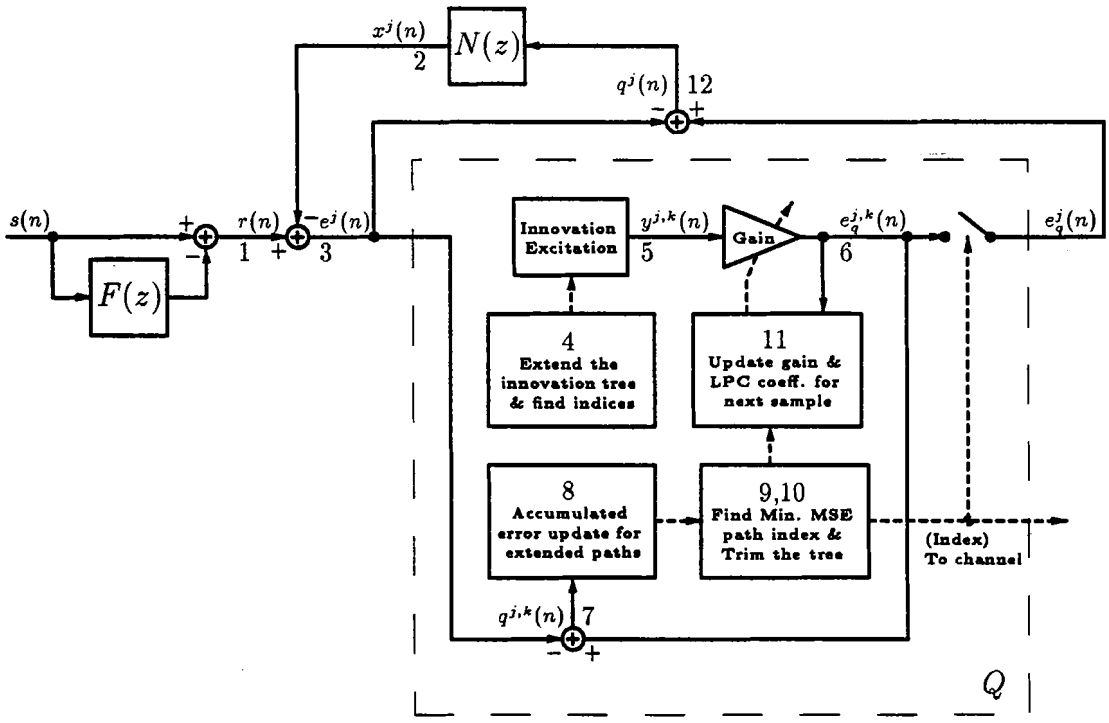


Fig. 5.1 LD-TREE encoder algorithm flow block diagram

are now summarized. The steps correspond to the algorithm flow presentation in Fig. 5.1 and is in accordance with the LD-TREE based on closed-loop APC block diagram of Fig. 2.10. The numbers in Fig. 5.1 correspond to step numbers below.

1– The input speech sample with time index n is passed through the prediction error filter $A(z) = 1 - F(z)$, the response $r(n)$ is the residual sample (delayed update).

$$r(n) = s(n) - s(n) * a(n).$$

In the above formula $*$ corresponds to the convolution operation.

2– Multipath (M,L) tree coding with quantization noise feedback is used. Let the variable N_{path} refer to the number of path in contention in the (M,L) algorithm. This variable is different than M=16 which is the fixed parameter for the maximum number of path in contention. The branching factor parameter in this case is 4 (wherever 4 is used, it refers to the branching factor). Lets assume that there are N_{path} quantization noise values ($q^j(n)$, $j = 1, 2, \ldots, N_{path}$) from previous sample quantization stage. Prediction coefficients $\{a_i\}$ calculated at previous stage (step 11) are converted to the noise filter coefficients $\{n_i = \lambda^i a_i\}$. The N_{path} quantization noise values of various paths are passed through the noise filter $N(z)$ (separate memories are kept for the N_{path} filtering operations),

$$x^j(n) = n(n) * q^j(n) \quad j = 1, 2, \ldots, N_{path}.$$

3– Adding these noise filtered quantization error samples to the residual sample $r(n)$, the excitation sample $e^j(n)$ for each path j is obtained (a total of N_{path} paths),

$$e^j(n) = x^j(n) + r(n) \quad j = 1, 2, \ldots, N_{path}.$$

4– Each tree leaf is extended to obtain $N_{path}\times 4$ newly extended leaves (maximum of 8×4=64).

5– Using the method of addressing for the stochastic innovation dictionary (concatenation of branching factor, explained in Section 2.3.2), the values of the innovation samples are obtained from the dictionary,

$$y^{j,k}(n) \quad j = 1, 2, \ldots, N_{path} \quad k = 1, 2, 3, 4.$$

6– Each innovation sample $y^{j,k}(n)$ is scaled by the j-th path gain calculated for each path at the previous stage (step 11). $N_{path}\times 4$ candidate gain-scaled innovation samples are obtained,

$$e_q^{j,k}(n) = \sigma(n) y^{j,k}(n) \quad j = 1, 2, \ldots, N_{path} \quad k = 1, 2, 3, 4.$$

7– These candidate gain-scaled innovation samples are subtracted from to-be-quantized noise-added residual samples $e^j(n)$ $j = 1, 2, \ldots, N_{path}$ to obtain quantization errors for each candidate:

$$q^{j,k}(n) = e^j(n) - e_q^{j,k}(n) \quad j = 1, 2, \ldots, N_{path} \quad k = 1, 2, 3, 4.$$

8– The accumulated error for each of the newly extended path is updated.

9– The winner path with the minimum accumulated MSE stems from the root whose index is transmitted to the decoder.

10– The tree trimming is done: eliminating the paths which do not stem from the newly obtained root in step 9 and possibly eliminating paths from the newly extended path so that a maximum of M=16 paths with the least amount of cumulated error is kept. New value of N_{path} is the number of path which is resulted at this step.

11– The gain values for the N_{path} paths are updated. The gain-scaled innovation samples (including the new extended ones) are used for this update. Also the prediction coefficients $\{a_i\}$ are updated using reconstructed speech. The reconstructed (quantized) speech is obtained by passing the newly selected gain-scaled innovation samples (root sample) through the synthesis filter $\frac{1}{1-F(z)}$.

12– The quantization noise for the N_{path} paths (calculated at step 7) are made available for the next stage of the algorithm at step 1 (index k is now not necessary, $q^j(n)$)

5.1.1 Initializations

The initial tree has one saved path. The nodes are populated with zero value and the gain values are set to identity. The start-up filter coefficients are also zero. Similar intial conditions are used both at the encoder and decoder.

5.1.2 Parameter Selection

As explained in Ref. [8], the performance curve as a function of M flattens around 16. The value M = 16 is used in the experiments. A similar curve plot for L presented in Ref. [8] suggests the use of L = 8 in the algorithm. The algorithm uses the adaptive Lattice algorithm with an order 8. The two pole+one zero window type of Eqn. 3.27 with $\beta_1 = 0.97$, $\beta_2 = 0.95$, and $a = 0.85$ is used. As mentioned earlier, the stochastic innovation tree nodes are populated from a Laplacian random number dictionary of size 2^k=4096 is used. The SNR degradation for smaller dictionary size down to 1024 was also acceptable (less than 1 dB).

5.2 LD-CELP Algorithm

The LD-CELP, like the conventional CELP [21], searches the codebook for the best matching codevector (each vector is 5 samples long) using analysis by synthesis and by minimizing the perceptually weighted error. Fig. 2.13 shows the block diagram of the LD-CELP encoder and decoder. The algorithm based on the March 1989 description released by AT&T [2] was implemented. †

The algorithm description of the LD-CELP coder as it was originally presented in Ref. [2] is now reviewed. First the main ideas are summarized. Then with the help of the algorithm block diagram in Fig. 5.2, the steps are shown. The block diagram of the CELP structure used by the LD-CELP shown in Fig. 2.13 is also used.

The concepts and components of the LD-CELP were introduced in Chapter 2. Blocks (vectors) of 5 input speech samples are formed. For each vector encoder in effect passes the 1024 excitation codevectors through gain scaling and synthesis units. The codevector which produces the minimum perceptually weighted MSE is selected. The index of this selected codevector is transmitted to the decoder. The details of the algorithm is pictorially represented in Fig. 5.2. The encoder includes a replica of

† The simulation works based on that document had finished when a new release describing the algorithm in detailed Pseudo-code, was published [3]. The results obtained based on this new release on the DEC Work-station [14] agreed with the previous results of experiments on the DEC VAX.

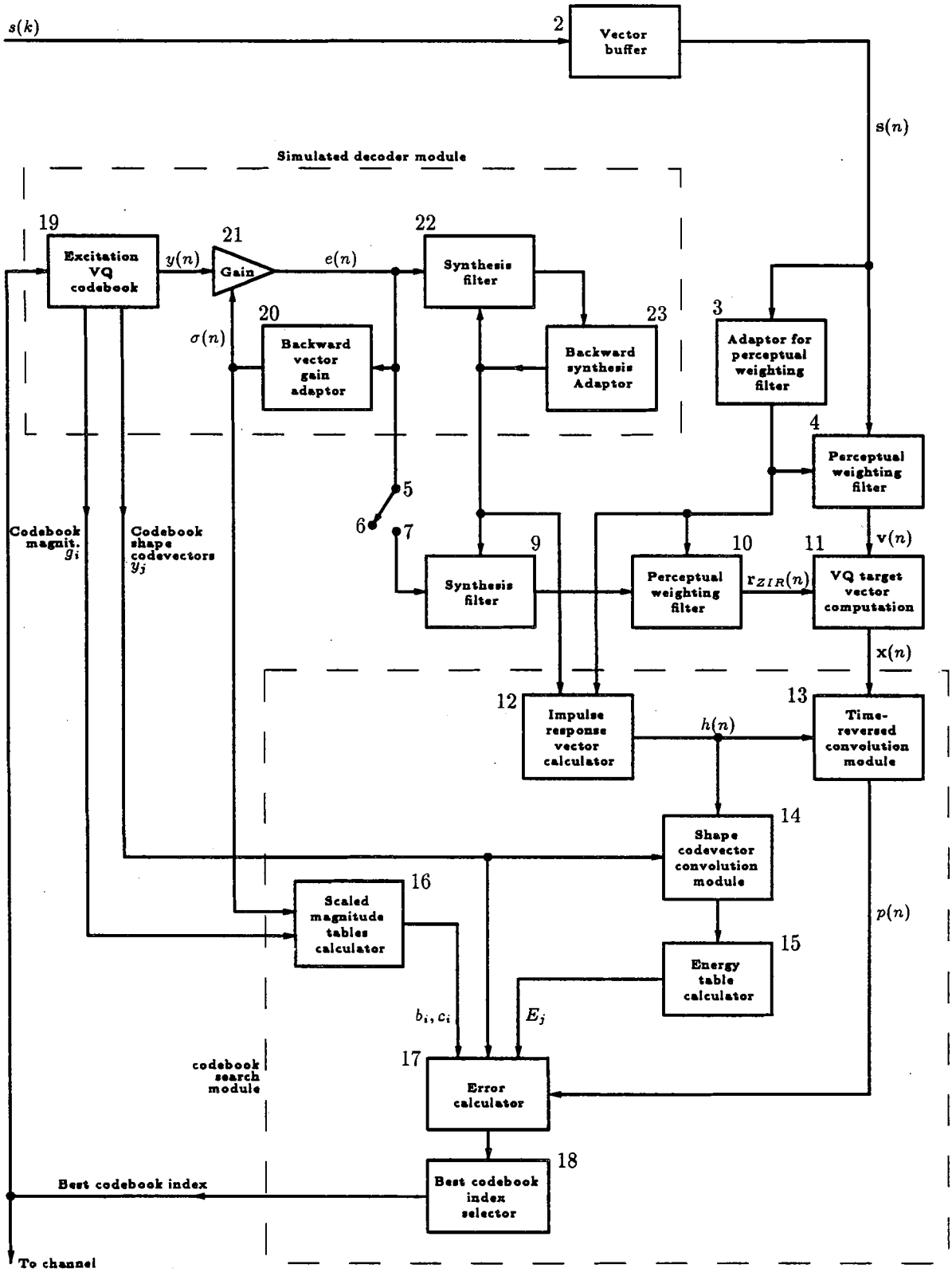


Fig. 5.2 LD-CELP encoder algorithm flow block diagram

the decoder. Here again, block numbers in Fig. 5.2 are referenced (the step numbers have no relation to the block numbers.).

I This block groups every 5 incoming speech samples into a block (vector). Time index n is now used for blocks instead of samples ($\mathbf{s}(n)$ instead of $s(k)$).

II This block updates the perceptual weighting filter coefficients. The update rate is every 5 vectors (20 samples). The process of the unit includes Barnwell windowing, Levinson-Durbin recursion for solving the set of Auto-Correlation equations, and bandwidth expansion to obtain pole/zero transversal coefficients.

III The n-th input speech vector $\mathbf{s}(n)$ is passed through the perceptual weighting filter.

IV There are two synthesis filter blocks 9 and 22 in Fig. 5.2 with identical coefficients. The coefficient update is by block 23. Barnwell windowing, Levinson-Durbin recursion, and simple white noise correction technique are used. The Bandwidth expansion with factor $\lambda = 0.9883$ is also employed.

V Backward adaptive logarithmic gain prediction (block 20) details are shown in Fig. 5.3. The block of one-vector-delay is done implicitly and shown for clarity. The input vector is the gain scaled excitation vector determined from previous encoded speech vector. RMS of the vector is calculated and its dB value is found by the next two blocks. A logarithmic gain offset value is determined in the coding stage and is stored in the log-gain value holder. The purpose of applying this offset value is to reduce some adverse effects caused by the the fact that the log-gain is not a zero mean signal in general (e.g. 38 dB). This new signal is what is used as the input to the log-gain predictor. Barnwell windowing, Levinson-Durbin recursion,, and Bandwidth expansion with $\lambda = 0.9$ are used. The offset is added back to the predicted logarithmic gain before log-gain limiter (2000) and inverse dB operations are applied.

VI ZIR response for the input is calculated by putting the switch 5 to position 6. Using the previous memory of the cascaded synthesis and perceptual filters 9 and 10 ZIR is found ($\mathbf{r}_{ZIR}(n)$).

VII Block 11 computes target vector $\mathbf{x}(n) = \mathbf{v}(n) - \mathbf{r}_{ZIR}(n)$

VIII Codebook search is done by block 12 through 18. In this special implementation, a 7 bit gain codebook and a 3 bit gain codebook are used. The gain index is further subdivided to a 1 bit sign and 2 bit magnitude. In effect the gain scaled excitation signals are passed though cascaded synthesis and perceptual filters ($H(z) = F(z)W(z)$). To obtain ZSR (filter memories are set to zero), block 12 calculates response $H(z)$ using the filter coefficients calculated in steps *II* and *IV*.

IX The MSE minimization procedure reduced to Eqn. 2.11 (Section 2.7.2) is implemented by blocks 13 through 18. $p(n) = \mathbf{H}^t\mathbf{x}(n)$ in Eqn. 2.11 is the time reversed convolution computed by 13. Shape codevector convolution module

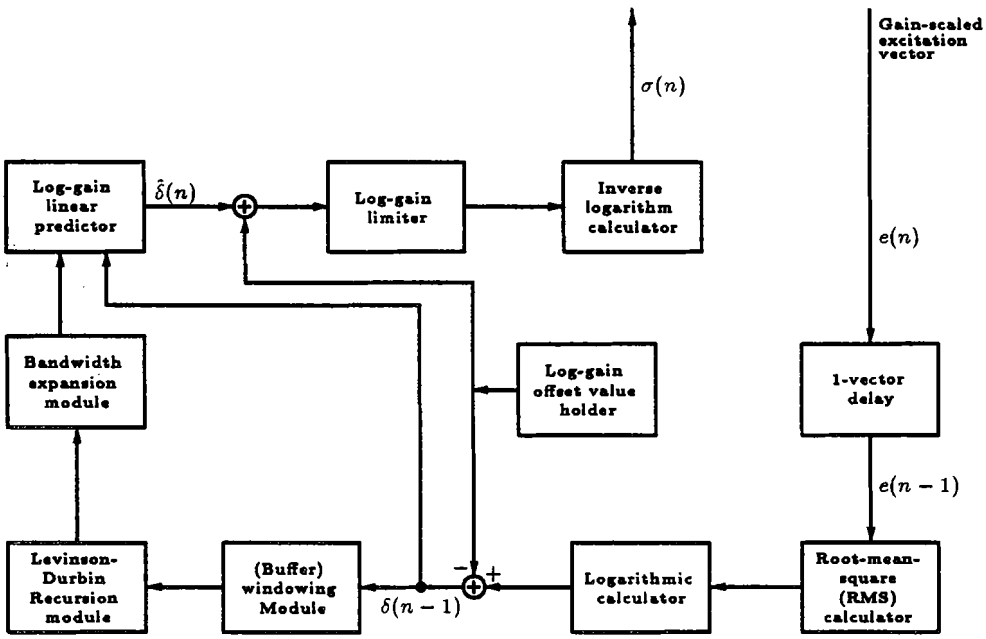


Fig. 5.3 LD-CELP logarithmic gain prediction block (block 20 in Fig. 5.2))

14 and energy table calculator 15 obtain $E_j = |\mathbf{H}y_j|^2$ in Eqn. 2.11. The resulted error in Eqn. 2.11 with the following convenient form

$$\hat{D} = -2\sigma(n)\mu_k g_i p^t(n) y_j + \sigma^2(n) g_i^2 E_j \quad \text{or}$$
$$\hat{D} = -\mu_k b_i g_i p^t(n) y_j + c_i E_j$$
$$(b_i = 2\sigma(n) g_i \quad \text{and} \quad c_i = \sigma^2(n) g_i^2),$$

is calculated by module 17. Finally, the best codevector index is selected by block 18 which outputs the indices i, j, and k of the selected gain (sign and magnitude) and shape codevectors (i.e. y_j, μ_k, and g_i). This selection means that the 128 shape codevectors and the 8 gain codevectors are tried and the ones which result in minimum $\hat{D}$ are selected. The indices are transmitted to the decoder.

X The codevector indices obtained in step *IX* are also used to form the excitation signal $e(n)$ (also gain scaled by the gain scaling unit). i.e.

$$e(n) = \mu_{k_{min}} g_{i_{min}} y_{i_{min}} \sigma(n).$$

XI Synthesis filters 22 uses $e(n)$ to produce the reconstructed signal. The ZIR filter memories for the block 9 and 10 are updated by passing $e(n)$ (switch 5 positioned at 7) through $F(z)$ and $W(z)$. The two sets of memory (ZIR and ZSR) are kept separate during encoding steps.

Note that the decoder algorithm is identical to its replica at the encoder.

5.2.1 Training of the Shape and Gain Codebooks

The shape/gain training procedure for CELP structures using gain scaled excitation codevectors is outlined in [59, 39, 43, 23]. The *individually optimized* gain and shape approach of [39] is used. This means shape and gain codebooks design are done independently. The initial gain codebook is "hand selected" (4 values plus the sign). The initial shape codebook was chosen from numbers with a Gaussian distribution. In a closed-loop gain-adaptive training algorithm, the distortion-versus-iteration does not necessarily decrease monotonically. The codebook with the lowest distortion after a preset number of iterations is saved [43]. The iterative clustering algorithm using a different set of training speech sentences (Appendix B) than the test speech sentences (Appendix A) were used [23]. The 5-dimensional space is divided to 128 Voronoi cells. The v-th cell has N_v elements. The distortion formulation of Eqn. 2.10 is used. The total distortion for each cluster with N_v elements is minimized. By taking the derivative of this total distortion with respect to the unknown centroid vector $\mathbf{y}$ and setting it to zero, a linear system of equations is obtained. The centroid codevectors are the solution of this linear system. Let n be the index of the training vector. For each training vector, there is a corresponding gain $\sigma(n)$ and a cascaded impulse response $\mathbf{H}(n)$. Hence using the Eqn. 2.10, the total distortion for the cluster v with N_v elements is

$$\begin{aligned} E =& \frac{1}{N_v} \sum_{n=1}^{N_v} |D|^2 \\ =& \frac{1}{N_v} \sum_{n=1}^{N_v} |\mathbf{x}(n)\mu(n)g(n)\mathbf{H}(n)\mathbf{y}|^T |\mathbf{x}(n)\mu(n)g(n)\mathbf{H}(n)\mathbf{y}|. \end{aligned}$$

Now to get the centroid vector $\mathbf{y}$, we take the derivative with respect to the $\mathbf{y}$ and set the result to zero. The following linear system of equations is resulted:

$$\sum_{n=1}^{N_v} [\sigma(n)^2 \mathbf{H}^T(n)\mathbf{H}(n)]\mathbf{y} = \sum_{n=1}^{N_v} \sigma(n)^2 \mathbf{H}^T(n)\mathbf{x}(n) \tag{5.1}$$

The matrices $\mathbf{H}(n)$ and the corresponding vectors $(\mathbf{x}(n), \sigma(n))$ are accumulated separately for each cluster. The resulted linear system of equation (Eqn. 5.1) is formed

the same way for all 128 clusters. The codebooks were trained using the above method using the training speech files in Appendix B. SNR improvements of about 1-2 dB was obtained for speech outside the training sequence (Appendix A). The published trained codebooks of [2, 3] gave similar results. The training of the codebooks for each new design feature is essential, if the results are to be compared properly.

5.3 Comparison Between the Two Coders

The objective (SNR, segSNR) and informal subjective results comparing the LD-CELP and LD-TREE are presented in this section.

5.3.1 On the Nature of the Differences and Similarities

The design of the original LD-TREE coder (unlike the LD-CELP) does not consider the noisy channel performance. Improving the LD-TREE coder robustness to channel errors is further discussed in Section 5.4.2.

The sliding block versus the block characteristics in the LD-TREE and LD-CELP coders is the main difference between the two coder. The use of the 50th order prediction filter in the LD-CELP as opposed to the 8th order filter in the LD-TREE constitutes another difference. Ref. [2] reports that the prediction gains and SNR gains obtained as a result of using high-order predictor justify the added complexity. The perceptual weighting filters used in the LD-TREE and LD-CELP coders not only differ in form but also are different in the use of *quantized* or *unquantized* speech signal to update their coefficients. The use of high-order synthesis filter in the LD-CELP has forced the coder to use a separate predictor filter for the perceptual weighting filter. The advantage of the general form perceptual weighting filter used in the LD-CELP coder is at the cost of additional computational complexity (separate prediction analysis).

The LD-CELP uses a 20 ms Hamming window or alternatively a recursive modified Barnwell window (to distribute the computation load for implementation considerations) for the backward adaptation of the prediction filter. The method of

choice for the analysis is Auto-Correlation and the update of the coefficients is done every 8th vector (5 ms). The LD-TREE on the other hand uses the Lattice adaptation algorithm to obtain reflection coefficients which are then converted to the direct form for use in the 8th order predictor filters. A one-pole or exponential window is used on the analysis data. The shape and the effective length of this window maybe controlled by a parameter. Computational savings are obtained by using this window. The coefficient updates are done on a sample-by-sample basis but in a delayed update configuration. The Lattice adaptation algorithm used in the LD-TREE is suitable for the low order prediction configuration. The exponential window used in the LD-TREE has a computational advantage over the windowing methods used in the LD-CELP, yet with the one-pole exponential window the control over the shape and the effective length is restricted and may not be suitable for the noisy channel conditions. The bandwidth expansion applied to the high-order predictor of LD-CELP can also be applied to the predictor in the LD-TREE coder. This improves the robustness to channel errors by making the noise less perceivable [2].

5.3.2 Objective Comparisons

The LD-TREE coder showed great promise by producing subjective quality equivalent to 7-bit/sample log-PCM with encoding delay not more than 1.125 ms [9]. Using simulations of the coders in this chapter, many comparisons between the LD-TREE and LD-CELP coders were made. Results of a typical experiment shown in Table 5.1, imply that the segSNR of the coded speech using the LD-TREE and LD-CELP coders for the two speech sentences "CAT" and "OAK" are very close. The LD-TREE does somewhat better for the male utterances and the LD-CELP is slightly better for female utterances.

5.3.3 Subjective Comparisons

The informal subjective tests also agree with the above conclusion. The above comparison is for the clean channel condition.

Coder	CAT		OAK	
	Male	Female	Male	Female
LD-CELP	17.9	19.1	18.4	20.1
LD-TREE	19.1	19.1	19.6	19.3

Table 5.1 Comparison of coders segSNR

5.4 Improving the Two Coders

In this section methods to improve the two coders performance are suggested. Some improvements are applicable to both coders and some to a particular one.

5.4.1 Improvements Applicable to Both Coders

The 50th order predictor in LD-CELP does capture pitch effects within its lag range. However, the use of the Auto-Correlation method does not fully exploit the high lag correlations because window edge effects. The Cumani Covariance-Lattice method does not have the above window/order problems and produces higher prediction gains [12]. The Cumani algorithm is one of a larger class of algorithms (PORLA) which are potentially useful for high-order prediction [53, 60, 61].

Results of extensive experiments to obtain high-quality prediction analysis for high-order predictors (as presented in Chapter 4) can be used to improve the performance of both coders. For clean speech, the prediction gain of the Cumani Covariance-Lattice method [52]) is several dB higher (approximately 2 dB for female and 3 dB for male utterances) than the Barnwell Auto-Correlation method. The reasons for this better performance were discussed in Chapter 4. Table 5.2 uses the average performance over several speech files to compare the Cumani Covariance-Lattice method with the Auto-Correlation method in LD-CELP (both order 50). The effect of backward adaptation based on the (noisy) reconstructed signal is such that the objective performance of the two methods is not noticeably different. However, informal subjective tests indicate that the differences are either absent (both are of very high quality

Analysis Method	Male		Female	
	SNR	segSNR	SNR	segSNR
Barnwell-Auto	19.6	21.0	24.1	22.6
Covariance-Lattice	19.6	20.7	22.9	21.7

Table 5.2 Objective coder performance comparison of LD-CELP with two analysis techniques (dB)

with no noticeable degradations) or there is a slight preference for the Covariance-Lattice method (specially for male speakers).

5.4.2 Improving the LD-TREE

Starting from a stochastic tree an algorithm was devised for the training of the innovation dictionary. The algorithm generalizes the well known LBG approach [34] to the stochastic tree coders. The obtained results showed how the training of the innovation dictionary improves the coder performance. Use of higher order filter with a better prediction analysis scheme (Chapter 3 and 4) also improves the coder performance and allows for the capture of pitch content in the speech signal.

In order to use a single high-order synthesis filter, the new open-loop configuration of Fig. 2.12 is suggested. As a result of the use this structure with modifications such as use of separated ZIR and ZSR (to obtain computational saving), and delayed prediction coefficient update, one may have a tree coder with similar advantages as the LD-CELP. The higher-order synthesis filter with the appropriate perceptual weighting filter allows for the capture of the pitch content without malfunctioning under noisy channel conditions.

Training the Innovation Dictionary: The new training algorithm for the tree (and Tree-CELP) coder innovation dictionary is now discussed. The idea of training the stochastic dictionary is not as straightforward as the training of the codebook in the CELP coders. This is due to the fact that in the tree coder, the switching among selected entries in the dictionary resulted from the training is somewhat random (or

at least is less controllable). Given the training sequence $\{s(n)\}$, the key steps based on the Linde *et al* [34] basic algorithm is as follows:

1– Populate the initial codebook with random numbers (example Laplacian distribution);

2– The coder is run using the training sequence, accumulating the unquantized prediction errors associated with the released node of the tree;

3– The centroid for the dictionary entry is found and used to re-populate the dictionary.

After the initial first step, steps 2 and 3 are repeated alternatively as in the generalized Lloyd algorithm until some convergence criteria is met. Again since the closed-loop design is taking place, the distortion-versus-iteration does not monotonically decrease. The dictionary with the lowest distortion after a preset number of iterations is saved. It was found that 3–5 iterations results in satisfactory results. Note the advantage of this relatively simple design procedure is the adaptation of excitation dictionary to many characteristics of the system components. The disadvantage is the small dimensionality (one) of the used Voronoi space (LBG algorithm). As a result the inter-sample relations are not exploited (as it is in the case of the CELP training). Other approaches similar to the ones used for the trellis [32] and VQ coders may result in better gains (special considerations related to the stochastic properties of the dictionary has to be made). Simulation results using the above simple training algorithm resulted in SNR improvements of about 1 to 2 dB for speech files outside the training sequence (Appendix A). Subjective quality improvement was also noticeable.

The New Tree Coder Structure for Higher Order: In Section 2.4, the two simple and general forms of perceptual weighting filters were discussed. The advantages of the general form alternative were shown. Among them was the fact that the general form is more appropriate for coders which use high-order predictors. Here an alternative structure for the closed-loop configuration LD-TREE which allows for the use of general form perceptual weighting filter is introduced.

The block diagram of the alternative structure is shown in Fig. 5.4 [47]. The

resulted general perceptual weighting filter has the form

$$W(z) = \frac{1 - N_1(z)}{1 - N_2(z)} \tag{5.2}$$

where

$$N_1(z) = N(z/\lambda_1) = \sum_{i=0}^{P'} n_i \lambda_1^i z^i \quad \text{and} \quad N_2(z) = N(z/\lambda_2) = \sum_{i=0}^{P'} n_i \lambda_2^i z^i,$$

with $0 < \lambda_2 < \lambda_1 \leq 1$ (e.g. $\lambda_1 = 0.9, \lambda_2 = 0.4$). The weighting filter $W(z)$ does not have a direct link to the predictor $F(z)$. The adaptation of the noise filter coefficients $\{n_i\}$ can be based on the *unquantized* speech. The noise filter order P' and the order of the predictor filter $F(z)$ may be different (e.g. $F(z)$ order = 50 and $P' = 10$).

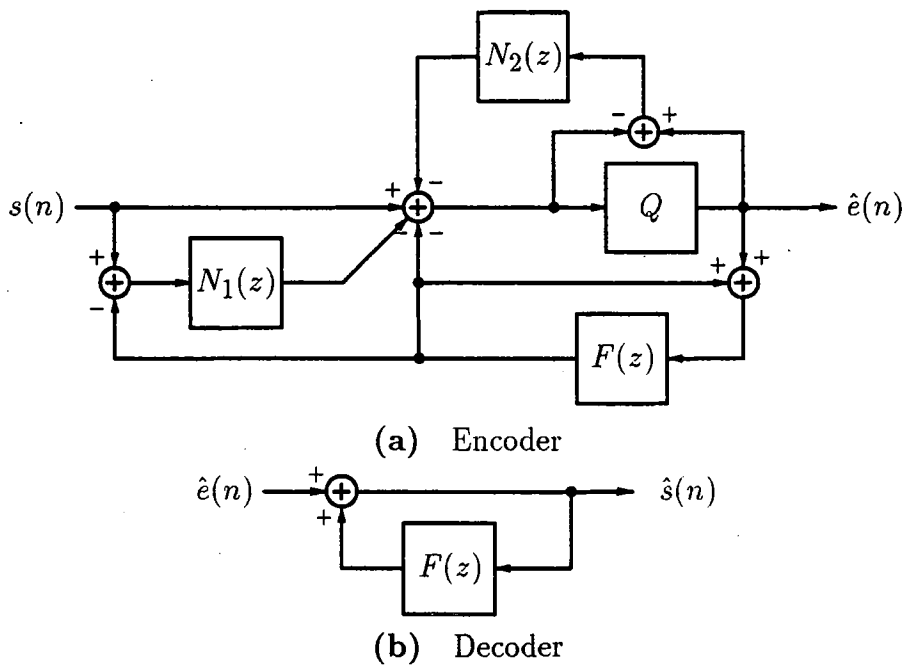


Fig. 5.4 A closed-loop configuration with generalized noise feedback [47]

The following steps give the derivation for Eqn. 5.2. At the encoder we have:

$$q(n) = x_q(n) - x(n) \tag{5.3}$$

$$x(n) = s(n) - \sum_{i=1}^{P} a_i \hat{s}(n-i) - \sum_{i=1}^{P'} n_{2_i} q(n-i) - \sum_{i=1}^{P'} n_{1_i} [s(n-i) - \hat{s}(n-i)] \tag{5.4}$$

Now at the decoder we obtain

$$\hat{s}(n) = x_q(n) + \sum_{i=1}^{p} a_i \hat{s}(n-i). \tag{5.5}$$

Using Eqn. 5.3 in 5.5, we have

$$\hat{s}(n) = x(n) + q(n) + \sum_{i=1}^{p} a_i \hat{s}(n-i).$$

Substituting Eqn. 5.4 in the above, we get

$$\begin{aligned}\hat{s}(n) =& s(n) - \sum_{i=1}^{P} a_i \hat{s}(n-i) - \sum_{i=1}^{P'} n_{2_i} q(n-i) - \sum_{i=1}^{P'} n_{1_i}[s(n-i) - \hat{s}(n-i)] \\ &+ q(n) + \sum_{i=1}^{p} a_i \hat{s}(n-i).\end{aligned}$$

Taking the z-transform, we get

$$\hat{S}(z) = S(z) - P(z)\hat{S}(z) - N_2(z)Q(z) - N_1(z)S(z) + N_1(z)\hat{S}(z) + Q(z) + P(z)\hat{S}(z)$$

$$[\hat{S}(z) - S(z)][1 - N_1(z)] = Q(z)[1 - N_2(z)]$$

$$\hat{S}(z) - S(z) = E(z) = \frac{1 - N_2(z)}{1 - N_1(z)} Q(z)$$

$$W(z) = \frac{1 - N_1(z)}{1 - N_2(z)}.$$

Robustness to Channel Errors and Tandeming Performance: The original LD-TREE was not designed with considerations with respect to the channels with non-zero probability of error (P_e). The performance of the LD-TREE coder as it was in the original coder of Ref. [8] sharply dropped with simulated channels with non-zero probability of error (even $P_e = 10^{-6}$ did not give an acceptable result). The CCITT standardization specification for performance under noisy channel (Table 1.1) showed how channel noise performance can be an important criterion. Although channel error performance and tandeming were not the focus of this work, some simulations were carried out and from the results, the following remarks may be made with regard to the less robust behaviour of the LD-TREE coder.

- Pitch predictor originally used in the LD-TREE coder is actually harmful under noisy channel conditions and hence can be replaced by a single high-order synthesis filter. Use of high-order filter requires a different structure such as the ones suggested above.

- The gain adaptation plays a key role in the behavior of the coder under channel errors, gain adaptation with shorter impulse response is more suitable to obtain a better robustness to channel errors,
- When LD-TREE is operating under noisy channel condition, catastrophic effects resulting from switching between paths has to be overcome. In the tree structures, once an error occurs, the effects can last for some time.
- The above effect can be minimized if delayed gain-adaptation strategy (suggested earlier) is implemented. This is provided that the coder behaviour is acceptable with this strategy (future simulations are needed),
- Adaptive postfiltering improves the coder performance under clear channel conditions. When tandeming performance and robustness to channel errors are among the required criteria, postfiltering can be harmful. However as mentioned before, from the recent experiments of AT&T, a specially tuned postfiltering is beneficial.

Chapter 6 Summary and Recommendations For Future Research

The network-quality 16 kb/s CCITT standardization has brought new focus to low-delay speech coding at 16 kb/s. In the standardization specification, additional requirements (e.g. delay less than 2 ms and robustness to channel errors) over the near-transparent quality are put forth. In future similar kinds of requirements can be expected for coders operating at rates between 8 to 16 kb/s. The LD-TREE and LD-CELP coders may both be considered as potential candidate coders for low-delay network-quality applications. Performance quality equivalent to 7 bits/sample log-PCM with delays less than 2 ms under clear channel conditions is achieved by the two coders. Satisfactory performance quality, under noisy channel conditions, is also reported for the LD-CELP.

The performance of the two coders, however, has not been compared under the same conditions before. Simulations in this work compared the two coders under similar conditions. The simulation results indicated that the segSNR's of the coded speech using the LD-TREE and LD-CELP coders are very close. The LD-TREE does somewhat better for the male utterances and the LD-CELP is slightly better for female utterances. The informal subjective tests agree with the above conclusion.

The above comparison is for the clean channel condition. The design of the original version LD-TREE coder (unlike the LD-CELP) does not have an emphasis on the noisy channel performance. The coded speech using the LD-TREE degrades rapidly under the noisy channel conditions while the LD-CELP withstands channel

probability of error of 10^{-3} and 10^{-2} with acceptable levels of quality loss.

The essential difference between the coders is the block versus sliding window excitation coding. Without channel error considerations, sliding window coders would seem to be preferable in terms of performance alone. There are no block edge effects with sliding window techniques. However, channel errors propagate for longer times within the sliding block structure. In addition, pseudo-Gray coding to mitigate the effect of errors is possible with block codes. Coarse simulation execution time comparisons indicate that the two coders have comparable complexities. The tradeoff here would seem to be clean channel performance versus noisy channel performance. Note that there are important applications (e.g. undersea fibre transmission) in which channel errors are not significant. On the other hand, for other applications (mobile radio or in-building wireless), channel error rates can be much more severe than the rates in the CCITT objectives.

Both coders contain structures with similar functions. The various components can be mixed and matched between the coders. It must be kept in mind that in a backward adaptive structure, each component must perform well. For instance a good residual coder results in an accurate reconstructed signal which in turn is used to adapt the predictor. A breakdown in either the residual coder or the predictor update results in breakdown of the coder.

Under clean channel conditions, even though LD-CELP uses a 50th order predictor compared to the LD-TREE 8th order predictor, the overall speech quality is very similar. A small advantage accrues to the high-order predictor for female speech in which the pitch range falls within 50 samples. High-order predictors can be used in either coder.

The perceptual weighting filters used in the LD-TREE and LD-CELP coders not only differ in form but also are different in the use of *quantized* or *unquantized* speech signal to update their coefficients. The use of high-order synthesis filter in the LD-CELP has forced the coder to use a separate predictor filter for the perceptual weighting filter. Again this strategy can be carried out along with high-order predictors in the LD-TREE coder.

The bandwidth expansion applied to the high-order predictor of LD-CELP can also be applied to the predictor in the LD-TREE coder. This can also improve the robustness to channel errors by making the noise less perceivable.

The speech quality of both LD-CELP and LD-TREE at 16 kb/s is very high. "Improvements" applied to the coders are not very noticeable. The ultimate goal is that such improvements will allow the high quality coding at lower rates. Of course, there may be substantial computational penalty to be paid for these improvements.

The 50th order predictor in LD-CELP does capture pitch effects within its lag range. However, the use of the Auto-Correlation method does not fully exploit the high lag correlations because of window edge effects. The well-conditioned high-order Cumani Covariance-Lattice method does not have the above window/order problems and produces higher prediction gains. The Cumani algorithm is one of a larger class of algorithms (PORLA) which are potentially useful for high-order prediction.

Results of extensive experiments to obtain high-quality prediction analysis for high-order predictors can be used to improve the performance of both coders. For clean speech, the prediction gain of the Cumani Covariance-Lattice method is several dB higher (approximately 2 dB for female and 3 dB for male utterances) than the Barnwell Auto-Correlation method. The reasons for this better performance were discussed in Chapters 3 and 4. The effect of backward adaptation based on the (noisy) reconstructed signal is such that the objective performance of the two methods is not noticeably different. However, informal subjective tests indicate that there is a slight preference for the Covariance Lattice method (especially for male speakers).

As it is done for the LD-CELP codebooks, training of the innovation signal boosts the performance of the LD-TREE coder. A new training procedure was implemented for the LD-TREE dictionary. The segSNR improvements of about 1–2 dB were obtained when the stochastic innovation dictionary was trained using this procedure.

When LD-TREE is operating under noisy channel condition, catastrophic effects resulting from switching between paths lasts for some time (characteristic of the tree structure). Once an error occurs, the effects propagate to the future samples. This problem has to be addressed by limiting the error effect along the path.

Although the exponentially averaged gain adaptation method of LD-TREE is adequate for clean channels, a better gain adapter suited for the stochastic tree coders is required to overcome the malfunctioning of the LD-TREE coder under noisy channel conditions. Preliminary experiments have verified this effect. Simple remedies were applied to achieve much better robustness, but with a loss of coder performance. Use of more complex gain adaptation strategies similar to the ones used in the LD-CELP could provide error robustness with no loss or even a possible increase in performance with no errors. The new gain adaptation strategy however has to fit to the tree coder characteristics. If the gain adaptation updates are delayed (similar to the prediction coefficient update), the complex gain adaptation method becomes computationally practical.

6.1 Future "Tree-CELP" Coders

The sliding block versus the block characteristics of the LD-TREE and LD-CELP is the main difference between the two coder. Other advantages come from the use of better quality components such as the high-order predictor with highest prediction gain, better gain predictors, dictionary training, and the type of perceptual weighting and postfiltering. A Tree-CELP coder taking the best components from the two structures is a good candidate for future low-delay coders at medium rates of 8–16 kb/s. For such coders, choice of open-loop or closed-loop alternatives as well as the type of fractional tree configuration (namely multi-sample/node tree as in Fig. 2.7 or multi-tree as in Fig. 2.8) have to be considered.

The Tree-CELP coder based on the closed-loop APC configuration discussed in Chapter 2 was later modified in Chapter 5 (Fig. 5.4) to incorporate general perceptual weighting filter. The structure of a Tree-CELP coder based on the open-loop CELP-like structure is similar to the one shown in Fig. 2.12. These two configuration which can be used by the future Tree-CELP coders, have their own advantages. As it was discussed in Chapter 2, some computational savings can be obtained when the second structure is used due to ease of separation of of ZIR and ZSR. This computational saving strategy was explained in Chapter 2.

Another advantage is related to the recent techniques suggested to be used in the CELP-like structures [62]. "Direct VQ" techniques allows for computational reductions resulting from elimination of some convolution operations of the filters in the CELP-like structures. The CELP-like open-loop structure provides a more suitable structure for the general perceptual weighting filter (e.g. closed-loop configuration in Fig. 5.4 is more complex than open-loop configuration in Fig. 2.13). For the lower bit rate coders where the assumption of white residual quantization noise is violated, one may speculate that the behaviour of the closed-loop versus open-loop structure can be different and one structure can be more suitable. This kind of comparison needs further investigation.

The multi-symbol/node or the multi-tree structures of Figures 2.7 and 2.8 (discussed in Chapter 2) show the excitation structure for the Tree-CELP coder. The performance of these two structures in an open-loop or closed-loop alternative configurations at rates below 16 kb/s also needs further research.

A hybrid Tree-CELP coder with high-quality compatible components is a good bet for the future low-delay coders at rates below 16 kb/s. This coder benefits from both sliding block and block characteristics. Using a similar innovation sequence selection as in the LD-TREE coder, large innovation dictionary size is possible. Other advantages comes from the use of better quality components such as high-order predictor with highest prediction gain, better gain predictors, dictionary training, perceptual weighting, and postfiltering. Choosing these components, ensuring their compatability, and the fine tuning of the coder is important and is a time consuming exercise. Consider a Tree-CELP coder coder with a trained codebook, high-order Covariance-Lattice predictor, appropriate postfiltering and a good gain adaptation strategy. Such a coder would probably allow for high-quality speech coding at rates of 12–14 kb/s. This suggestion was partially pursued in the continuation work of this thesis for 12 kb/s low-delay coders [14]. Further work is needed for coders operating at rates between 8–12 kb/s.

APPENDIX A

Test speech files

Male Speaker:

OAKM8 - Oak is strong and also gives shade.

CATM8 - Cats and dogs each hate the other.

Female Speaker:

OAKF8 - Oak is strong and also gives shade.

CATF8 - Cats and dogs each hate the other.

APPENDIX B

Speech files used for codebook and dictionary training

Male Speaker # 1:

ADDM8 - Add the sum to the product of these three.
OPNM8 - Open the crate but don't break the glass.
PIPM8 - The pipe began to rust while new.
THVM8 - Thieves who rob friends deserve jail.

Male Speaker # 2:

DOUG1 - The birch canoe slid on the smooth planks.
DOUG2 - Glue the sheet to the dark blue blackground.
DOUG3 - It's easy to tell the depth of the well.
DOUG4 - These days a chicken leg is a rare dish.

Female Speaker # 1:

ADDF8 - Add the sum to the product of these three.
OPNF8 - Open the crate but don't break the glass.
PIPF8 - The pipe began to rust while new.
THVF8 - Thieves who rob friends deserve jail.

Female Speaker # 2:

VOICF1 - The birch canoe slid on the smooth planks.
VOICF2 - Glue the sheet to the dark blue blackground.
VOICF3 - It's easy to tell the depth of the well.
VOICF4 - These days a chicken leg is a rare dish.

References

1. N. S. Jayant, "High-quality coding of telephone speech and wideband audio," *IEEE Communication magazine*, pp. 10–20, Jan. 1990.
2. AT&T contributions to CCITT study group XV and T1Y1.2, Oct. 1988 - July 1989.
3. AT&T *Detailed description of AT&T's LD-CELP algorithm*, Contribution to CCITT Study group XV, Nov. 1989.
4. J. H. Chen, "A robust low-delay CELP speech coder at 16 kb/s," *Proc. GLOBCOM Conf.*, pp. 1237–1240, 1989.
5. J. H. Chen, "A robust low-delay CELP speech coder at 16 kb/s," *IEEE workshop on speech coding for telecommunications*, Vancouver, Canada, Sep. 1989.
6. J. H. Chen, "High-quality 16 kb/s speech coding with a one-way delay less than 2 ms," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* , pp. 453–456, 1990.
7. J. H. Chen, M. J. Melchner, R. V. Cox, and D. O. Bowker, "Real time implementation and performance of a 16 kb/s low-delay CELP speech coder," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* , pp. 181–184, 1990.
8. V. Iyengar, "A Low delay 16 kb/s coder for speech signals," *Master of Eng. Thesis , Dept. Of Elec. Eng. McGill University*, Aug. 1987.
9. V. Iyengar and P. Kabal, "A low delay 16 kb/s speech coder,"in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 243–246, 1988.
10. V. Iyengar and P. Kabal, "A low delay 16 kb/s speech coder," *IEEE Trans. Signal Processing*, vol. 39, pp. 1049–1057, May 1991.
11. M. Foodeei and P. Kabal "Low-delay speech coders at 16 kb/s: a CELP and a tree coder," in *Proc. 15th Binnial Symp. Comm.*, p.p. 320–323, Kingston, Ont. June 1990.
12. M. Foodeei and P. Kabal "Backward adaptive prediction: high-order predictors and formant-pitch configuration," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 2405–2409, Toronto, Ont., May 1991.
13. M. Foodeei and P. Kabal "Low-delay CELP and tree coders: comparison and performance improvements," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 25–29, Toronto, Ont., May 1991.
14. J. Grass, P. Kabal, M. Foodeei and P. Mermelstein, "High-quality low-delay speech coding at 12 kb/s," in *IEEE workshop on speech coding for telecommunications*, B.C., Canada, Sep. 1991.
15. J. H. Chen, Y. C Lin, and R. V. Cox, "A fixed-point 16 kb/s LD-CELP algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* , pp. 21-24, Toronto, Ont., May 1991.

16. W. R. Daumer, P. Mermelstein, X. Maitre, and I. Tokizawa, "Overview of the ADPCM coding algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* San Diego, CA, pp. 23.1.1–23.1.4., Mar. 1984.

17. D. O'Shaughnessy, *Speech communication, humman and machine,* Massachusetts: Addison Wesley, 1987.

18. L. R. Rabiner and R.W. Schafer, *Digital Processing of speech signals,* New Jersey: Prentice-Hall, 1978.

19. P. Kabal and R. P. Ramachandran, "Joint optimization of linear predictors in speech coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 642–650, Oct. 1989.

20. P. Kroon and E. F. Deprettre, "A class of analysis-by-synthesis predictive coders for high quality speech coding at rates between 4.8 and 16 kb/s," *IEEE Journal on Selected Areas in Communication*, vol. 6, no. 2, pp. 353–363, Feb. 1988.

21. M. R. Schroeder and B.S. Atal, "Code-Excited Linear Prediction (CELP): high quality speech at very low bit rates," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* Tampa, Florida, pp. 25.1.1–25.1.4, April 1985.

22. J. H. Chen A. Gersho, "Vector adaptive coding of speech at 9.6 kb/s," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* , Tokyo, Japan, p.p. 33.4.1–33.4.4, 1986.

23. G. Davidson, M. Yong, and A. Gersho, "Real-time vector excitation coding at 4800 bps," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 51.4.1–51.4.4, 1987.

24. B. Atal and J. Remde, "A new model of LPC excitation for producing natural-sounding speech at low bit rates," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 614–617, 1982.

25. B. S. Atal, "High-quality spech at low bit rates: multi-pulse and stochasticaly excited linear predictive coders," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 33.1.1–33.1.4., Tokyo, 1986.

26. R. M. Gray, *Source coding theory* Kluwer Academic Publishers, 1990.

27. N. S. Jayant and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, 1984.

28. H. Fehn and Peter Noll, "Multipath search coding of stationary signals with applications to speech ," *IEEE Trans. Comm.*, vol. COM-30, pp. 687–701, April 1982.

29. J. D. Gibson and W-W Chang, "Fractional rate multi-tree speech coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. 53.1.1–53.1.4., 1989

30. W-W Chang, "Analysis and design of low rate tree codes," *PHD Thesis , Texas A&M University*, May 1989.

31. M. W. Marcellin, T. R. Fischer, and J. D. Gibson, "Predictive trellis coded quantization of speech," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* pp. S6.7.1–S6.7.4, 1988.

32. B-H Jung, "Design and performance of the trellis vector quantizers for speech signals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 1423–1431, Sep. 1988.

33. N. S. Jayant and S. A. Christensen, "Tree encoding of speech using (M,L)-Algorithm and adaptive quantization," *IEEE Trans. on Comm.*, vol. COM-26, No. 9, Sep. 1987.

34. Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantization design," *IEEE Trans. on Comm.*, vol. COM-28, pp. 84–95, Jan 1980.

35. N. S. Jayant, "Adaptive quantization with one-word memory," *AT&T Bell Lab. Tech. Journal*, vol. 52, No. 7, pp. 1119–1144, Sep. 1973.

36. V. Ramamoorthy and N. S. Jayant, "Enhancement of ADPCM speech by adaptive postfiltering," *AT&T Bell Lab. Tech. Journal*, vol. 63, No. 8, pp. 1465–1475, Oct. 1984.

37. N. S. Jayant and V. Ramamoorthy, "Adaptive postfiltering of 16 kb/s ADPCM speech," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* Tokyo, Japan, pp. 829–832, April 1986.

38. J. H. Chen and A. Gersho, "Real-time vector APC speech coding at 4800 bps with adaptive postfiltering," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 2185–2188, 1987.

39. M. J. Sabin and R. M. Gray, "Product code vector quantizers for waveform and voice coding," *IEEE Trans. ASSP*, vol. ASSP-32, no. 3, pp. 474–488, June 1984.

40. J. R. B. De Marca and N. S. Jayant, "An algorithm for assigning binary indices to the codevectors of multidimentional quantizer," *ICC Conf. Rec.*, pp. 1128–1132, 1987.

41. K. A. Zeger and A. Gersho, "Zero redundancy channel coding in vector quantization," *Electronics Letters*, pp. 654–656, June 1987.

42. J. H. Chen A. Gersho, "Gain-adaptive vector quantization for medium-rate speech coding," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* , Chicago, p.p. 1456–1460, June 1985.

43. J. H. Chen and A. Gersho, "Gain-adaptive vector quantization with application to speech coding," *IEEE Trans. Comm.*, pp. 918–930, Sep. 1987.

44. V. Cuperman, A. Gersho, R. Pettigrew, J. J. Shynk and J. H. Yao "Backward adaptation for low delay vector exitation coding of speech at 16 kb/s," *Proc. Int. Conf. Acoust., Speech, Signal Processing,* pp 34.2.1–34.2.5, 1989.

45. R. P. Ramachandran and P. Kabal, "Stability and performance analysis of pitch filters in speech coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, No. 7, pp. 937–946, July 1987.

46. R. P. Ramachandran and P. Kabal, "Pitch prediction filters in speech coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, No. 4, pp. 467–478, April 1989.

47. B. Atal and M. Schroeder, "Predictive coding of the speech signals and subjective error criteria," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 247–254, June 1979.

48. B. S. Atal, "Predictive coding of speech at low bit rates," *IEEE Trans. on Communications*, vol. COM-30, pp. 600–614, April 1982.

49. B. W. Dickinson, "Autoregressive estimation using residual energy ratios," *IEEE Trans. on Information Therory*, vol IT-24, pp. 503–504, July 1978.

50. J. I. Makhoul and L. K. Cosell, "Adaptive Lattice Analysis of Speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 654–659, April 1981.

51. J. Makhoul, "Stable and efficient lattice methods of linear prediction," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 423–428, Oct. 1977.

52. A. Cumani, "On a covariance lattice algorithm for linear prediction," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Paris, France, pp. 651–654, 1982.

53. P. Strobach, "Pure Order Recursive Least-Square Ladder algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 880–897, Aug. 1986.

54. T. P. Barnwell, III, "Recursive windowing for generating auto-correlation coefficients for LPC analysis," *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-29, No. 5, pp. 1062–1066, Oct. 1981.

55. S. Singhal and B. S. Atal, "Improving performance of multi-pulse LPC coders at low bit rates," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, San Diego, CA, pp. 1.3.1–1.3.4., Mar. 1984.

56. G. H. Golub and H. F. Van Loan, *Matrix computations, 2nd edition.* Baltimore, Maryland: The John Hopkins University Press, 1989.

57. R. Pettigrew and V. Cuperman, "Backward pitch adaptation for low-delay speech coding," *Proc. Int. Conf. Acoust., Speech, Signal Processing*, pp 34.3.1–34.3.6, 1989.

58. J. B. Anderson and J. B. Bodie, "Tree coding of speech," *IEEE Trans. on Information Therory*, vol IT-21, pp. 379–387, July 1975.

59. M. J. Sabin and R. M. Gray, "Product code vector quantizers for speech waveform coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, , pp. E6.5.1–E6.5.4, April 1982.

60. P. Strobach, *Linear prediction theory, a mathematical basis for adaptive systems*, Springer-verlag, 1990.

61. P. Strobach, "Recursive Triangular Array Ladder algorithms," *IEEE Trans. on Signal Processing*, vol. SP-39, pp. 122–136, Jan. 1991.

62. Y. Shoham, "On the use of direct vector quantization in LPC-based analysis-by-synthesis coding systems," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. S.1.2.1–S.1.2.4, 1991.