Analysis of Predictor Mistracking in ADPCM Speech Coders

by

Paul M. Yatrou
B.Eng. (Electrical)

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

Department of Electrical Engineering
McGill University
Montréal, Canada

June 1987

Analysis of Predictor Mistracking in ADPCM Speech Coders

Abstract

This thesis investigates the problem of predictor mistracking with narrowband signals in backward ADPCM (Adaptive Differential Pulse Code Modulation) speech coders. Mistracking is an offset between encoder and decoder coefficients caused by error feedback from the signal reconstruction filter to the predictor adaptation process. Conventional adaptive predictors, using transversal or lattice filters, are shown to suffer from mistracking in the presence of transmission errors. Through modification of existing algorithms, a new class of residual-driven lattice predictors (LR) is presented which guarantees tracking for all input signals without regard to the order of prediction.

Comparisons between the LR predictor and four other systems were performed. In the absence of transmission errors, it is shown that a segmental *SNR* drop for speech of as much as 2 dB may be encountered for the LR predictor with respect to the conventional systems. In the presence of errors, however, this degradation is outweighed by the enhanced speech and narrowband signal transmission performance, as required by practical telecommunication applications.

Sommaire

Ce mémoire examine les problèmes de détraquement des filtres de prédiction employés par les codeurs de la parole MICDA (Modulation par Impulsions et Codage Différentiel Adaptatif) lorsque ces filtres sont excités par des signaux à bande étroite. Les prédicteurs adaptatifs classiques, à structure de filtre transversal ou en treillis, sont susceptible à détraquement en présence d'erreurs de transmission. Après modifications des algorithmes classiques on présente une nouvelle classe des prédicteurs en treillis (LR). Ces prédicteurs sont excités par le residu et ils assurent la convergence des coefficients du décodeur pour tous les signaux sans égard à l'ordre de prédiction.

Plusieurs comparaisons entre le prédicteur LR et quatre autres systèmes ont été faites. Pour des signaux de parole, il est montré qu'en absence d'erreurs de transmission, le prédicteur obtient un *RSB* segmental inférieur a 2 dB à la meilleur performance des systèmes classiques. Par contre, en présence d'erreurs de transmission, le prédicteur LR offre une performance supérieure aux autres systèmes pour les deux types de signaux, comme exigé par les applications pratiques en télécommunications.

Acknowledgements

I would like to express my gratitude to my thesis supervisor, Dr. Paul Mermelstein, for his guidance, encouragement, and patience throughout the various stages of this work. By example of his insight and clarity during our discussions, I have learned how to deal with the "grey" areas of a problem.

I would also like to thank the professors, research assistants, and fellow students at INRS-Télécommunications for their assistance in the laboratory and for the many fruitful discussions they have shared with me. In particular, I am indebted to Dr. Douglas O'Shaughnessy who gave me some time to complete the final stages of this document while I was working for him on a different project.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

There is a major ongoing trend towards the digitization of telecommunication and telephone networks in order to increase transmission quality and flexibility. Digital voice communication is preferable to analog because of the non-accumulation of noise over distance. To achieve this higher quality for voice and other signals, we must encode the signal at the source or at the point of analog/digital conversion and decode it at the destination or point of digital/analog conversion. Other advantages in digital communications include: error-protection, encryption, and the ability to process, multiplex, packetize and mix.

An essential component in the telephone network is the *digital encoder*, a device which converts an analog waveform (voice or data) into a stream of bits to be transmitted over a digital communications channel. At the receiving end, a decoder reconstructs the waveform from the transmitted bit-stream. This process is the basis of digital communications.

1.1 Speech Coding Activities: Theory and Standards

Speech coding activities over the last three decades are divided into: theory and development of algorithms, and standardization of speech coding techniques for network applications.

Speech coding theory was always way ahead of standards. *Pulse Code Modulation* (PCM) is the earliest developed and also the best understood coding system due to its simplicity. A PCM encoder is simply a waveform sampler followed by an amplitude quantizer. The main advantage of PCM over analog modulation techniques is the ability to trade bandwidth for increased signal-to-noise ratio — historically, a well known relationship [1].

In the late sixties, the *International Telegraph and Telephone Consultative Committee* (CCITT) defined two PCM standards for the international network: μ-law 64 kilobits-per-second (kb/s) PCM in North America, and A-law 64 kb/s PCM in Europe [2]. These two incompatible representations constitute the current standards in digital encoding of telephone signals, with 64 kb/s as the basic unit in channel capacity. Transcoders map one representation into the other at the appropriate regional interfaces.

The PCM standards took into account only the need to cover a large dynamic range (using logarithmic quantization) and an ability to code all voiceband frequency signals transmitted over the network, not only speech. This latter feature was achieved through an instantaneous coding, with the code depending only on the current sample value for maximum simplicity.

Speech coding theory progressed rapidly in the seventies and early eighties, giving rise to a variety of PCM-based *waveform*[1] encoding schemes. Of all the coders suited for *toll* (telephone) quality speech, the following are the most significant: *Adaptive*

[1] Speech coders can be divided into two classes waveform coders and source coders Waveform coders transmit a vestige of the input signal (*e g.*, a prediction residual) which contains enough information to reconstruct the input at the decoder Source coders transmit a set of parameters describing the input signal (*e.g.*, LPC coefficients, pitch period, voiced/unvoiced excitation). The set of parameters is used as a model to "rebuild" the input. In general, waveform coders are well suited for toll and broadcast quality applications but require correspondingly high bit-rates. Source coders, on the other hand, are used in synthetic and communications quality applications where low bit-rates are necessary Hybrid coders combine features of waveform and source coding (see [3] for a complete discussion)

Differential Pulse Code Modulation (ADPCM), *Adaptive Delta Modulation* (ADM), *Adaptive Transform Coding* (ATC), *Sub Band Coding* (SBC) and *Adaptive Predictive Coding* (APC) [3]. These encoders compress speech signals, with greater efficiency than PCM, by removing redundant components in the input prior to quantization.

The exponential growth of VLSI (Very Large Scale Integration) technology over the past decade has enabled the implementation of encoding techniques by means of single board and single chip DSP packages. In effect, both the theoretical and economic barriers of introducing a new encoding scheme in the telephone network, have disappeared. As a result, the CCITT undertook a four-year study, commencing in June 1981, with the intention of defining a new encoding standard to supplement the current μ-law and A-law PCM standards [4]. Early preference was given to 32 kb/s ADPCM, and in the October 1984 CCITT plenary session a specific algorithm was recommended as an international standard [5][6]. This algorithm could offer a 2:1 reduction in bit rate from 64 kb/s PCM and maintain, as much as possible, the transmission performance of PCM. Thus, an 8 kHz sampling and 4-bit per sample coding are implicit requirements. Although ADPCM systems are adaptive, their performance with non-voice signals is somewhat compromised since their design is based on properties of speech. The most important problem in the 32 kb/s algorithm (G.721) was to select a technique appropriate to data and speech without explicit decision of which signal is being communicated. Moreover, the technique would have to ensure that the decoder recovers to match the encoder after a transmission error has occurred. Some of the constraints observed in the design of the algorithm are:

- transmission of speech, voiceband data, and signalling tones with good fidelity.
- handling of multiple encoding/decoding stages with analog or digital (transcoding to/from 64 kb/s PCM, either A-law or μ-law) intervening links.
- the algorithm should not rely on the transmission of side information and, in or-

der to minimize transmission delay, all adaptation processes should be backward-acting in time.

- fixed bit-rate in time for any channel.
- robustness to transmission errors for all types of-signals, with special attention paid to encoder/decoder tracking recovery after a disturbance.

The proposed scheme uses an adaptive dynamic-locking quantizer (DLQ) (see [7]) and an adaptive short-term predictor to compress the input sequence. The adaptive DLQ was designed specifically for network applications. It employs two modes of adaptation, *fast* for speech and *slow* for voiceband data (vbd) and signalling tones, with a continuum of speeds in between the two modes. The adaptive predictor is composed of a sixth order *moving-average* (MA) section that models zeros in the input signal and a second order *autoregressive* (AR) section that models poles in the input signal. These sections are simple transversal, or tapped-delay line, filters.

According to the transmission constraints, the quantizer and predictor adaptations are backward-acting in time without any reliance on side information. Backward adaptive predictive coders rely on identical quantization and prediction operations at the encoder and decoder to ensure that the signal reconstructed at the decoder is an accurate replica of the signal arriving at the encoder. Thus, with backward adaptation all updates **must** be derived from the transmitted bit-stream, this being the only link between encoder and decoder.

1.2 Predictor Mistracking

Two different tracking requirements must be met by the ADPCM system. First, the encoder must track the input when it is either stationary or transient.[2] This is

[2] Stationarity, in the strict-sense, implies that the signal statistics are constant with respect to time. In

done by adjusting the quantizer and the predictor in order to match the statistics of the input signal. Only when the encoder tracks the input sufficiently well, can there be a gain in transmission quality over an equivalent PCM system.

Second, as mentioned earlier, the adjustment of the decoder onto the encoder is necessary for the reconstructed signal at the decoder to be as close as possible to the input signal. Henceforth, (mis)tracking will refer to the (mis)adjustment of the decoder onto the encoder. Transmission errors introduce differences between the quantized residuals at the encoder and decoder thereby forcing at least short-term deviations between the two systems due to backward adaptation. Permanent mistracking can be catastrophic — if the decoder does not invert the encoder operation, the output may be significantly corrupted. One of the main goals of the CCITT study was to derive adaptation algorithms that are not susceptible to permanent mistracking.

Permanent mistracking does not occur with MA predictors or quantizers. However, mistracking is a major concern with backward-adaptive AR predictors. In the classical gradient adaptation algorithm for AR predictors, coefficient leakage is used in order to mitigate the effects of a transmission error. This leakage is known to ensure predictor convergence in the presence of errors if the input is sufficiently broadband (*e.g.*, speech, white noise). However, for narrowband signals, such as vbd and signalling tones, the "leaky" predictor adaptation process may possess several stable states and it is possible that two predictors with different initial conditions will never converge. Experiments with signalling tones have shown that even a single transmission error can lead to predictor mistracking.

the wide-sense, only the second-order statistics must be constant with time; in this document we will assume wide-sense stationarity unless otherwise indicated. Finally, the transient state implies time-varying statistics; however, for the signals dealt with here (*e.g.* speech), a transient can be sufficiently approximated by a sequence of short-term wide-sense stationary segments

In the CCITT proposal, this problem was overcome by using a modified gradient coefficient adaptation algorithm for the AR predictor. The modification is based on a study on predictor mistracking by Millar and Mermelstein [8]. Gibson [9] has observed that predictors with AR sections tend to track "themselves" rather than the input signal; this suggests that, after a disturbance, the predictor at the decoder can diverge from the predictor at the encoder. Other work on this problem includes a very recent study on the CCITT standard by Bonnet et al. [10].

In an attempt to avoid mistracking altogether, some authors have suggested using predictors with fixed AR sections and/or adaptive MA sections [11][12][13][14]. However, to achieve a given prediction gain, systems employing adaptive AR prediction are computationally more efficient than those employing adaptive MA prediction only. Thus, the development of AR predictors that do not mistrack is an important theoretical and practical objective.

1.3 Focus and Organization of Thesis

This thesis presents a new analysis of the mistracking problem in 32 kb/s ADPCM systems with AR predictors. For simplicity, the MA predictor is not included in the analysis and the DLQ is replaced by a unimodal adaptive quantizer. Since mistracking depends only on the AR predictor, the above changes are not significant. The research objectives are as follows: (1) to re-examine the classical gradient adaptation algorithm for transversal predictors, and to show its susceptibility to mistracking; (2) to analyze the CCITT modified gradient algorithm for transversal predictors; and (3) to develop a better understanding of the class of predictors that does not manifest mistracking. To this end two lattice predictors were also explored: a classical signal-driven algorithm which mistracks and a newly developed residual-driven algorithm which tracks.

In adaptive systems robustness to errors is usually obtained at the cost of performance and complexity. In most practical telecommunication applications, however, these shortcomings are outweighed by the enhanced transmission performance in the presence of errors. Performance must be optimized over a number of expected transmission conditions, at the cost of making it suboptimal in any one condition. Results obtained from this research will serve to determine the performance penalties associated with ensuring robustness in the predictor to expected transmission errors. Although the "setting" of this project is 32 kb/s ADPCM, the results can be used for a variety of systems where tracking between adaptive filters is of great concern.

The thesis is divided into five chapters, including this introduction. Chapter 2, presented as background material, deals with ADPCM speech coding in general. A typical encoder/decoder configuration is illustrated, and the sequence of steps leading to reconstruction of the input signal at the decoder are examined. Objective performance measurements for ADPCM are then discussed. Forward and backward adaptation strategies for predictors and quantizers are compared. A section on robust quantization concludes this chapter.

Chapter 3 constitutes the theoretical core of the thesis and is divided into three sections. The first section introduces the AR model for speech. The second section deals with the transversal AR predictor and the development of stochastic gradient adaptation algorithms. Traditional "signal-driven" adaptation (driven by the cross-correlation between the residual and the reconstructed signal) is shown to be susceptible to mistracking. "Residual-driven" adaptation (driven solely by autocorrelations of the residual) is proven to be robust. The third section is a mirror image of the second, dealing with the lattice structure instead of the transversal structure. A new class of residual-driven lattice predictors is developed which guarantees tracking for all signals without regard to the order of prediction.

Simulations and experimental results for the different prediction algorithms are found in Chapter 4. The algorithms are first optimized with respect to average speech and dual-tone signals. Encoder and decoder tracking is then examined for both types of signal. The tests illustrate the mistracking problem with the signal-driven predictors in the presence of errors, as well as the slight performance degradation with residual-driven adaptation in the absence of errors.

Chapter 5 presents a summary and conclusion of the investigation and ends with some suggestions for improvement and future research.

Chapter 2 Adaptive Differential Pulse-Code Modulation (ADPCM)

2.1 Digital Coding of Speech Waveforms

Digital speech coding refers to a wide range of techniques which compress a speech waveform so that it may be efficiently transmitted over a digital channel. This channel could be part of a communications system (as in telephony) or a storage system (as in digitally recorded announcements). The analog waveform is pre-filtered and sampled at a rate which avoids *aliasing* at the receiver.[1] The time-discretized waveform, or *sequence*, is then coded (compressed) and transmitted. At the receiving end there exists a decoder which performs the inverse operation to the coder, yielding a reconstructed speech sequence. Finally the output speech waveform is recovered from the sequence using an interpolation filter. The entire process is illustrated in Figure 2.1. The sampling and interpolation operations are theoretically error-free for

[1] A basic property of speech waveforms is that they are bandlimited. This natural limitation is due to the speech production process and varies from speaker to speaker. Regardless, most digital coding systems impose a strict bandwidth on the waveform to prevent aliasing. In commercial telephony a bandpass filter of 300 to 3400 Hz is used, and the resulting signal is sampled at 8 kHz This slight oversampling serves to increase the transition region between passband and stopband so that a less stringent filter can be used

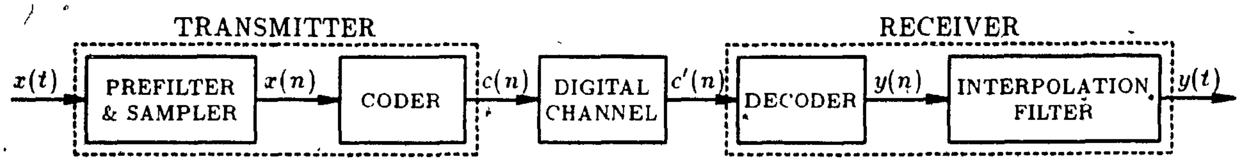


Fig. 2.1 Digital coding system for speech

bandlimited inputs. In this thesis we will take for granted these operations and deal only with coding, transmission and decoding of digital sequences.

The main goal of all waveform coding systems is to reconstruct the input signal with the least error given a fixed transmission bit-rate. An alternate goal is to decrease the bit-rate while maintaining the same reconstruction quality. However, both goals have conflicting requirements as will be shown here.

The bit-rate of a waveform coder is determined by the number of levels in the quantizer. A quantizer with $L = 2^B$ levels requires at least B bits per sample to encode its input. Assuming a sampling frequency of 8 kHz, the resulting bit-rate will be

$$R = 8B = 8\log_2 L \quad \text{kb/s}.$$

As a measure of quality, the *signal-to-noise ratio* (*SNR*) for a PCM system with an L-level quantizer is given by

$$SNR \simeq 6B - \theta = 6\log_2 L - \theta \quad \text{dB},$$

where θ is a parameter depending on the input pdf and the quantizer characteristic (typical values for θ lie in the range: 0 to 10).[2] In effect, if we decrease the number of bits (therefore, levels) but still try cover the same range, the courseness of the new quantizer will result in a 6 dB loss in *SNR* per bit.

Therefore, all other things being equal, a reduction in the transmission bit-rate R implies *(i)* a reduction in the number of levels L in the quantizer and *(ii)* a corre-

[2] This relationship is known to hold for $B \geq 4$ and only when the quantizer range is matched to the signal range (see [15]). For $B < 4$ the quantization error becomes increasingly correlated with the input

sponding decrease in reconstruction quality *SNR*.

ADPCM uses a differential encoding scheme which exploits short-term, or *near sample*, correlations in the input speech sequence in order to achieve *(i)* without incurring *(ii)*.

2.2 ADPCM Encoder/Decoder Configuration

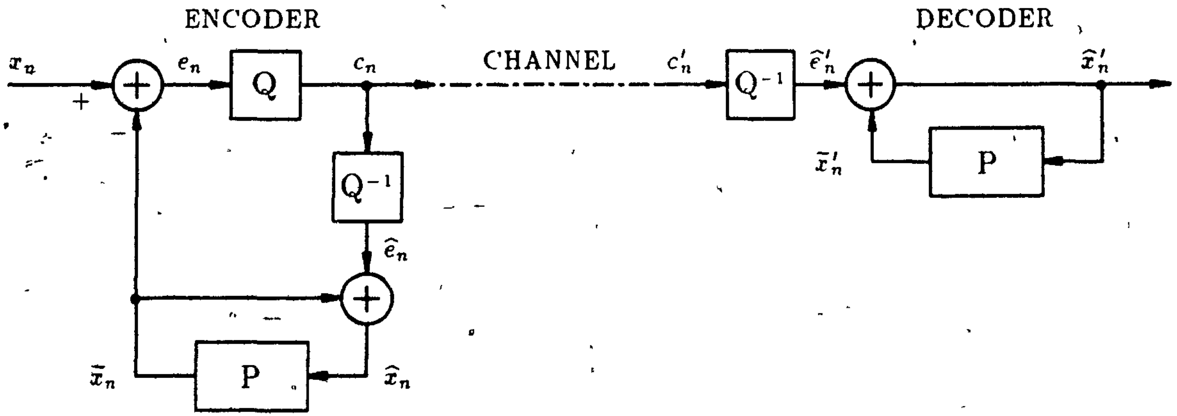


Fig. 2.2 Block diagram of the ADPCM encoder/decoder

The basic ADPCM encoder (see Figure 2.2) is composed of a quantizer **Q**, an inverse-quantizer $\mathbf{Q}^{-1}$, and a predictor **P** In practice, there is an analog-to-digital (A/D) converter outside the loop which quantizes the input to more bits than employed for transmission. Thus, **Q** and $\mathbf{Q}^{-1}$ are digital-to-digital (D/D) converters.[3] This allows all the math operations (+, ×, ÷) to be digital. However, in this thesis we are only interested in the analysis of a single encoder/decoder connection. There is no need to model the transcoding requirements outside the loop To this end, the only *digital* part of the system is the transmission channel.

[3] A quantizer can also map a finite set of symbols to another finite, but smaller, set of symbols (D/D conversion) This can occur, for example, in synchronous transcodings (PCM-ADPCM PCM).

Mathematically, a quantization mapping $\mathcal{Q}(\cdot)$ can be defined as

$$\mathcal{Q}(\cdot) \overset{\triangle}{=} \mathbf{Q}^{-1}[\mathbf{Q}(\cdot)].$$

A more detailed description of the quantizer is found in Section 2.5. The predictor **P**, a linear finite-impulse-response digital (FIR) filter, is discussed in Chapter 3.

Unlike PCM, which simply quantizes, encodes and transmits the input sequence $x(n)$, ADPCM forms a difference sequence

$$e(n) = x(n) - \tilde{x}(n), \tag{2.1}$$

where $\tilde{x}(n)$ is a linear prediction of $x(n)$. This sequence, known as the *prediction residual*, is quantized and a codeword $c(n)$ is transmitted through the channel. The smaller variance of $e(n)$, compared to $x(n)$, allows a reduction in L without a reduction in *SNR* (see Section 2.3). Inverse quantization yields the *quantized residual* sequence

$$\hat{e}(n) = \mathcal{Q}\big(e(n)\big) = e(n) + q(n), \tag{2.2}$$

where $q(n)$ is the quantization error.

In order that the encoder and decoder track and reconstruct the input sequence in synchrony, it is desirable that the prediction $\tilde{x}(n)$ be derived from previous values of the *reconstructed input* sequence $\hat{x}(n)$ instead of the encoder input $x(n)$. For the same reason, $\hat{x}(n)$ itself is formed from current values of $\tilde{x}(n)$ and $\hat{e}(n)$, rather than $e(n)$, giving

$$\hat{x}(n) = \tilde{x}(n) + \hat{e}(n). \tag{2.3}$$

By combining (2.1)–(2.3) the reconstructed signal can be expressed as

$$\hat{x}(n) = x(n) + q(n), \tag{2.4}$$

which implies that the reconstruction error at the encoder is identical to the quantization error.

Thus, a closed-loop structure with feedback around the quantizer allows the portion of the encoder which contains the inverse quantizer $\mathbf{Q}^{-1}$ and the predictor $\mathbf{P}$ to be copied in the decoder (Figure 2.2). At the decoder the codeword $c'(n)$ is received and inverse operations lead to

$$\hat{x}'(n) = \tilde{x}'(n) + \hat{\epsilon}'(n), \tag{2.5}$$

where $\hat{x}'(n)$, $\tilde{x}'(n)$, and $\hat{\epsilon}'(n)$ are the decoder's reconstructed, prediction, and quantized residual sequences, respectively. If the channel is error free, then

$$c'(n) = c(n),$$

$$\hat{\epsilon}'(n) = \hat{\epsilon}(n),$$

$$\tilde{x}'(n) = \tilde{x}(n),$$

and

$$\hat{x}'(n) = \hat{x}(n) = x(n) + q(n). \tag{2.6}$$

Hence, in the absence of channel errors, the reconstructed sequence at the decoder $\hat{x}'(n)$ will differ from the input sequence at the encoder $x(n)$ only by the quantization error $q(n)$. This is an important result which implies that quantization noise does not accumulate over time.

2.3 Objective Performance Measurement

The most common objective indicator of waveform coder performance is the encoder/decoder *SNR*, defined by

$$SNR \triangleq \frac{E[x^2(n)]}{E[(x(n) - \hat{x}'(n))^2]}. \tag{2.7}$$

In the absence of channel errors (2.6) applies, in which case the *SNR* can be expressed as

$$SNR = \frac{E[x^2(n)]}{E[q^2(n)]} = G_P \cdot SNR_Q, \tag{2.8a}$$

where

$$G_P = \frac{E[x^2(n)]}{E[e^2(n)]} \tag{2.8b}$$

and

$$SNR_Q = \frac{E[e^2(n)]}{E[q^2(n)]} \tag{2.8c}$$

are the prediction gain and quantizer *SNR*, respectively.

For a fixed number of levels, the quantization noise power can be made proportional to the quantizer input (prediction residual) power

$$E[q^2(n)] = \epsilon^2 E[e^2(n)], \tag{2.9}$$

where ϵ^2 is the quantizer performance factor (noise power per unit input power). This factor is independent of the quantizer input.[4]

Equations (2.8c) and (2.9) imply that $SNR_Q = 1/\epsilon^2$ is independent of the prediction residual $e(n)$. Clearly, the *SNR* is maximized when G_P is maximized, or equivalently when $E[e^2(n)]$ is minimized. Most predictor adaptation algorithms are indeed based on minimizing $E[e^2(n)]$, as will be shown in Chapter 3.

In practice, the ensemble statistics implied by the expectation operator $E[\cdot]$ are not available and must be replaced by time averages $\langle\cdot\rangle$ defined as

$$\langle u(n)\rangle = \frac{1}{N}\sum_{n=1}^{N} u(n),$$

where N is the length of the arbitrary sample sequence $u(n)$. Time averages are accurate estimates of ensemble statistics if the sample sequence is stationary and ergodic and N is large.

[4] The performance factor ϵ^2 depends mostly on the quantizer structure, i.e, type of quantizer, number of levels, dynamic range However, there is also some dependence on the statistics of the quantizer input but this is negligible if the quantization is fine and the quantizer is not overloaded [15]

So, in practical analysis, the coder performance is represented by the time averaged SNR, defined as

$$SNR = \frac{\langle x^2(n)\rangle}{\langle q^2(n)\rangle} = \frac{\langle x^2(n)\rangle}{\langle e^2(n)\rangle}\frac{\langle e^2(n)\rangle}{\langle q^2(n)\rangle} \tag{2.10a}$$

or, expressed in dB as

$$SNR_{\mathrm{dB}} = 10\log_{10} SNR. \tag{2.10b}$$

Time-averaging over the entire input sequence, as (2.10a) would indicate, tends to under-emphasize the performance in segments where the input energy is weak [15,16].

An objective measure which corresponds more closely to subjective evaluations is found by computing the SNR of (2.10a) or (2.10b) over many contiguous non-overlapping segments of the input sequence. This is more in line with what the human auditory system does when evaluating the quality of a signal. These short term SNR values, denoted here by $SNR_{\mathrm{dB}}(j)$ for the jth segment, also provide segmental performance information that just isn't available when the SNR is computed over the entire input sequence. The segment length is chosen so that the input could be considered stationary for that period of time. An appropriate segment length for speech inputs would be in the order of 16 ms or 128 samples.

An average segmental SNR measure is calculated as

$$SNRSEG = \frac{1}{K}\sum_{j=1}^{K} SNR_{\mathrm{dB}}(j),$$

where K is the number of segments in the input sequence. This average of the short-term SNR values tends to assign more equitable weightings to strong and weak portions of the input signal. Segmental SNR can also be expressed as the sum of segmental Gp and SNR_Q:

$$SNRSEG = GpSEG + SNR_Q SEG = \frac{1}{K}\sum_{j=1}^{K} Gp_{\mathrm{dB}}(j) + SNR_{Q\mathrm{dB}}(j) \tag{2.11}$$

If the quantizer is adaptive, the SNR_Q will be approximately constant under varying inputs. ~~Thus,~~ SNR and G_P are roughly equivalent performance measures (when there are no channel errors).

In the presence of errors, the simplifications imposed by (2.6) cannot be applied and the performance measures at the decoder must be calculated with reference to the encoder signals; *i.e.*,

$$
\begin{aligned}
SNR(p_e) &= \frac{\langle x^2(n)\rangle}{\langle (x(n) - \hat{x}'(n))^2\rangle}, \\
G_P(p_e) &= \frac{\langle x^2(n)\rangle}{\langle (x(n) - \tilde{x}'(n))^2\rangle}, \\
SNR_Q(p_e) &= \frac{\langle e^2(n)\rangle}{\langle (e(n) - \hat{e}'(n))^2\rangle},
\end{aligned}
$$

~~where~~ (p_e) denotes the presence of errors in the form of a bit error rate for a binary symmetric channel.[5] Segmental and average segmental values are computed as above. In this case, however, (2.11) does not hold since

$$SNR(p_e) \neq G_P(p_e) \cdot SNR_Q(p_e).$$

All the segmental measures discussed above will be used in Chapter 4 in order to evaluate the performance of the predictors.

2.4 Adaptation Strategies

The input environment of a coder is generally time-varying. Speech signals are inherently non-stationary, their amplitude statistics and spectral content gradually changing over time [17]. In ADPCM, both the quantizer and the predictor can be

[5] In a binary symmetric channel there is a probability p_e of a bit-error (*e.g.*, a 1 being received as a 0) and a probability $(1 - p_e)$ of no error. This process is (*i*) memoryless since the output depends only on the input and (*ii*) identically distributed since the statistics (p_e) of the channel do not change with time. Despite the simplicity of this model, it is very useful in representing expected transmission impairments.

adapted with time in order to track the changing input signal and provide improved performance.

Adaptation of the quantizer is performed by changing the quantizer's step-size parameter $\Delta(n)$ according to the prediction residual power $E[e^2(n)]$. Similarly, adaptation of the predictor is achieved by varying the set of predictor coefficients $\mathbf{A}(n) = \{a_i(n),\ i = 1, 2, \ldots, p\}$, in order to track the spectral content of the input $x(n)$. Thus it is expected that, for signals exhibiting short-term energy and correlation that are time-varying, a system utilizing adaptive components should outperform one with fixed components.

Adaptation schemes can be classified as either *forward* (open-loop) or *backward* (closed-loop). The forward adaptive process estimates the correlation in the input and uses the results to adjust the parameters of the system. Backward adaptation, on the other hand, uses observations of an output signal, along with the current state of the system, to form the parameter adjustment

The notation ADPCM-AQF-APF and ADPCM-AQB-APB is used to refer to systems with forward and backward adaptation in both the quantizer and predictor, respectively. These two forms of ADPCM are discussed in Sections 2.4.1 and 2.4.2. Mixed adaptation systems (*i.e.* AQF-APB or AQB-APF) are rarely encountered in ADPCM speech coding.

2.4.1 Forward Adaptation (ADPCM-AQF-APF)

Figure 2.3 illustrates an ADPCM system with forward adaptation in both the predictor and the quantizer. Note that the *Quantizer Adaptation Logic* and the *Predictor Adaptation Logic* receive their information from the respective input sequences $e(n)$ and $x(n)$.

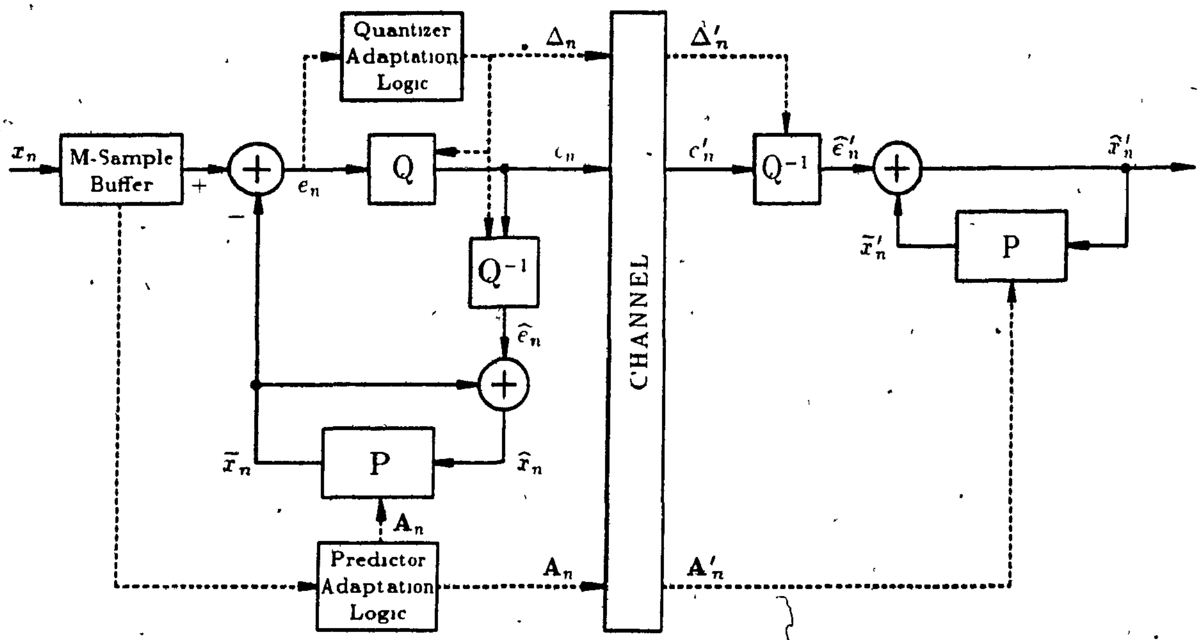


Fig. 2.3 ADPCM system with forward adaptation

In general, a block of M input samples must be collected before the adaptation algorithms can produce valid parameter estimates. In terms of parameter calculation, the block size M must be small enough so that the input can be considered stationary and large enough so that the estimates are reliable (32–128 samples for speech). A practical constraint on M is based on the coding delay introduced; *e.g.*, a delay of 32 samples or 4 ms is significant and perhaps unacceptable in certain applications.

First, M input samples are buffered and released only after the quantizer step-size $\Delta(n)$ and the predictor coefficients $\mathbf{A}(n)$ have been calculated. These parameters are then used to process the corresponding block of samples at the encoder and are transmitted to the decoder at a rate which is determined by the block size M. Transmitted parameters, also called *side information*, must share the channel with the codeword $c(n)$. Framing of the overall bit sequence must be employed so that the correct bits carrying the appropriate side-information can be located. This poses synchronization problems even when the side information rate is low. Moreover, a small percentage of the bit-rate is always dedicated to the transmission of side information.

Transmission of side information and coding delay are the main disadvantages in forward adaptive coders. One important advantage is robustness to channel errors. If a channel error corrupts the received adaptation parameters $\Delta'(n)$ and $\mathbf{A}'(n)$, its effects will be local to that particular block. In effect, block processing is equivalent to having finite memory in the system. Consequently, neither predictor nor quantizer mistracking is a problem in forward adaptive coders.

2.4.2 Backward Adaptation (ADPCM-AQB-APB)

An alternate adaptation strategy is shown in Figure 2.4. In this case, the *Quantizer Adaptation Logic* uses the codeword $c(n)$ and current step-size to generate the new step-size. Similarly, the *Predictor Adaptation Logic* uses $\hat{e}(n)$, $\hat{x}(n)$ and current coefficients to generate the new coefficients. State feedback (not shown in the figure) occurs inside the logic blocks. In the absence of transmission errors all the signals required for adaptation are available at the encoder as well as the decoder, thus no side information needs to be transmitted. As a result, the adaptation processes at both ends are identical.

For simplicity, the parameters can be updated with every new sample — this is called *sequential* adaptation. No block processing is required since the closed loop nature of the adaptation system allows an infinite but fading memory of previous inputs. Therefore, there is no coding delay in a backward adaptive coder.

A transmission error will affect the received codeword $c'(n)$ (no other information is transmitted) and consequently $\hat{e}'(n)$ and $\hat{x}'(n)$. Since these signals drive the adaptation parameters $\Delta'(n)$ and $\mathbf{A}'(n)$, the error will propagate around the synthesis and state-feedback loops possibly causing mistracking.

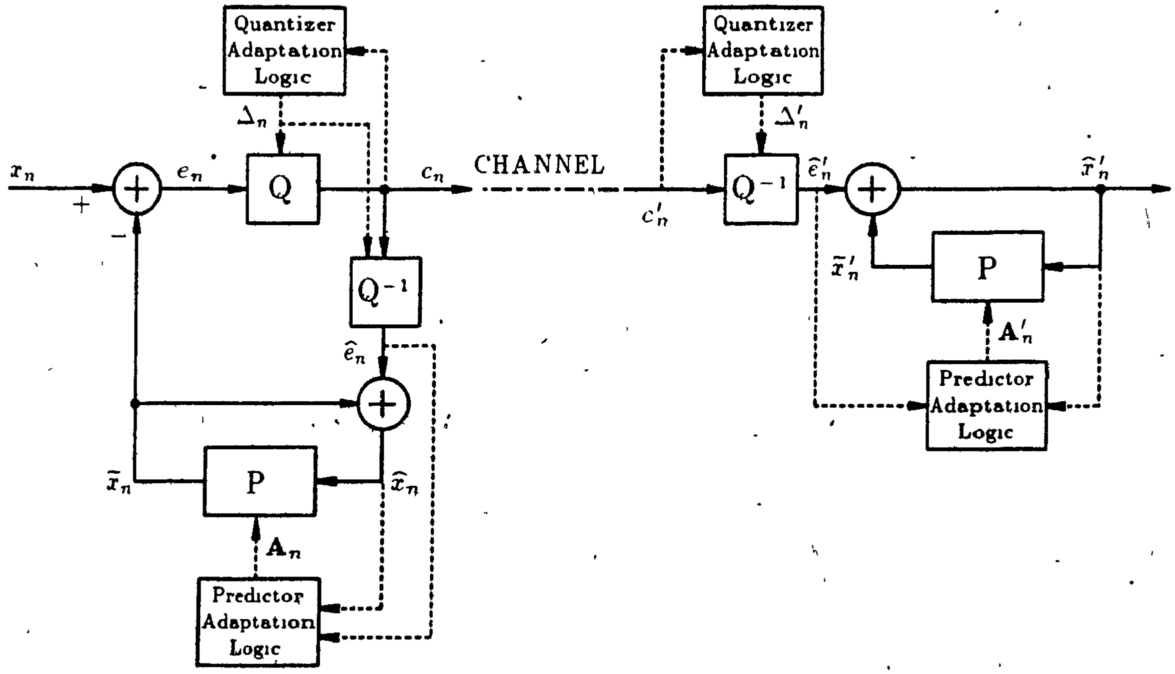


Fig. 2.4 ADPCM system with backward adaptation

The main problem in backward adaptive systems is to design robust algorithms that also perform well in the absence of errors. The next section deals with such an algorithm for the quantizer. Predictor mistracking and its correction are dealt with in the remaining chapters.

2.5 Robust Quantization

Amplitude quantization of a discrete-time signal is an important step in digital coding, since it determines the transmission bit-rate (as discussed in the beginning of this chapter). It is also the only source of distortion (quantization error) in the reconstruction process, when there are no channel errors.

2.5.1 Quantizer Mapping

In ADPCM, the quantization mapping $Q(\cdot)$ transforms the residual signal amplitude $e(n)$ at time n into $\widehat{e}(n)$, one of a finite set of amplitudes determined by the number of levels L. This is done instantaneously and the transformation at time n does not depend on earlier samples (memoryless quantization) *if, for now, we do not consider the effects of prediction.*

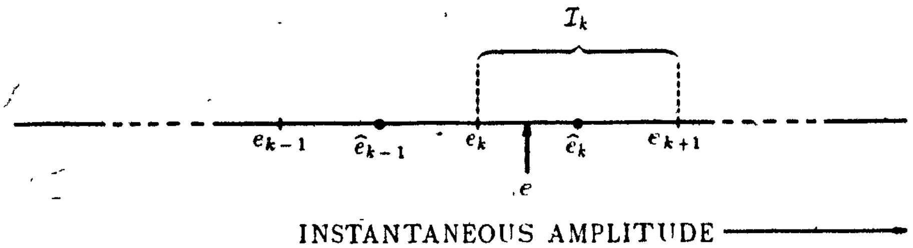


Fig. 2.5 Quantizer decision intervals

Dropping the time index n we denote the *decision levels* by e_k and the *output values* by $\hat{e}_k$ (Figure 2.5); then the signal amplitude e is represented by the index k if it falls in the interval

$$\mathcal{I}_k : \{e_k < e \leq e_{k+1}\}, \qquad k = 1, 2, \ldots, L.$$

The transmitted codeword c is a function of the index k which depends on the particular channel coding scheme being used. For simplicity, c can be chosen as the binary representation of k (although this is not the most beneficial choice). The corresponding output value must lie inside the interval $\mathcal{I}_k$, and is often chosen as the midpoint value

$$\hat{e} = Q(e) = \hat{e}_k = \frac{e_k + e_{k+1}}{2}, \qquad \text{iff} \quad e \in \mathcal{I}_k.$$

The overall mapping $\hat{e} = Q(e)$ is a staircase function with odd symmetry about the origin. In order to match the quantizer to the probability density function (pdf) of the input signal $e(n)$, different quantizer shapes can be defined by appropriately setting the decision levels e_k and output values $\hat{e}_k$.

The optimal values are found by minimizing the quantizer mean-square error (mse) $E[q^2] = E[(e-\hat{e})^2]$ with respect to e_k and $\hat{e}_k$ given a particular input pdf, $p_e(e)$.

There exists an iterative solution for minimizing $E[q^2]$, known as the *Lloyd-Max iteration*. The minimum will be global if the pdf is log-concave; *i.e.*, if $\partial^2 \log p_e(\epsilon)/\partial e^2 < 0$ [15].

Several different types of $Q(e)$ are shown in Figure 2.6 for $L = 7,8$. Without loss of generality, the *uniform midrise* quantizer of Figure 2.6(a) is used in the discussion on quantizer adaptation in the following section. This is because the quantizer shape is not as crucial in an adaptive system as it is in a fixed system, especially when the input is a residual signal.

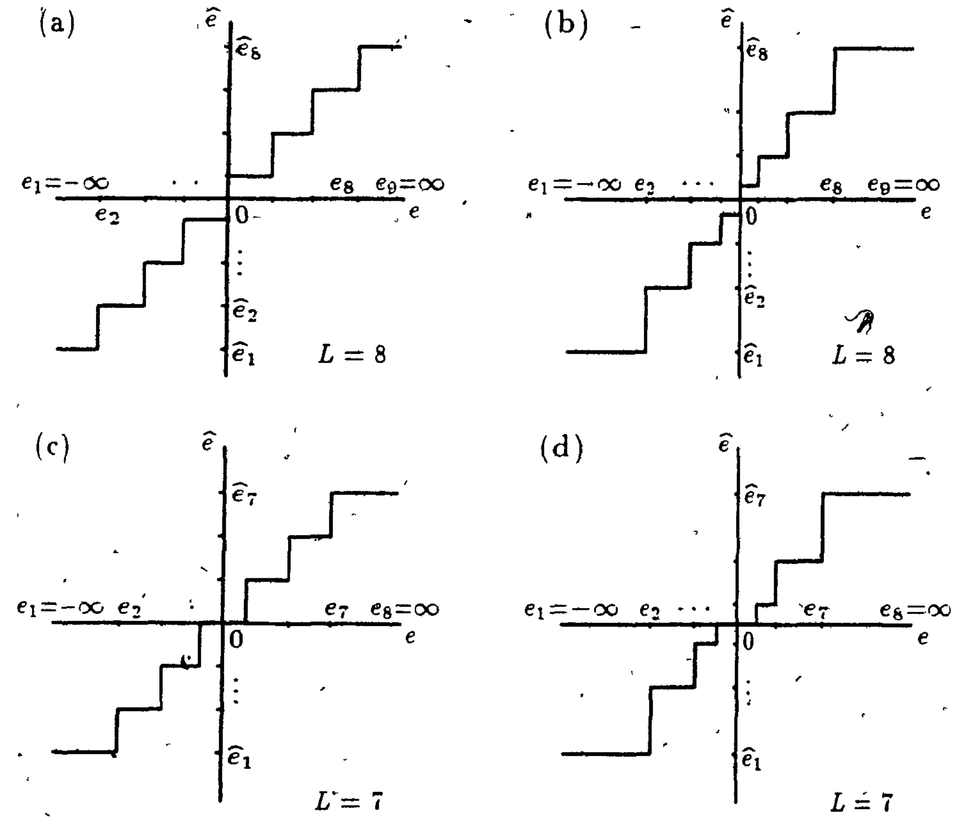


Fig. 2.6 Quantizer mappings: (a) uniform midrise; (b) nonuniform midrise; (c) uniform midtread; and (d) nonuniform midtread. *(Adapted from Jayant and Noll [15], page 117.)*

A 4-bit or 16-level quantizer with a Gaussian characteristic is used in the simulation of the ADPCM system in Chapter 4. This corresponds to Figure 2.6(b) with $L = 16$. The optimum decision values and output values, based on an input with a unit standard deviation ($\sigma_\epsilon = 1$) Gaussian pdf, are given in Table 2.1 [15]. Due to symmetry only the positive values are given. The optimum values have to be multiplied with σ_e for input residuals with a non-unity standard deviation. This quantizer is reported (in [15]) to achieve a maximum SNR_Q of 20.22 dB with stationary Gaussian inputs. In subsequent tests, we observed that the SNR_QSEG is within 17.5- 19.5 dB for most speech and narrowband inputs.

k	$e_{k,\mathrm{opt}}$	$\hat{e}_{k,\mathrm{opt}}$
9	0.000	0.109
10	0.217	0.326
11	0.433	0.542
12	0.650	0.759
13	0.866	0.975
14	1.083	1.192
15	1.299	1.408
16	1.516	1.624

Table 2.1 Optimum decision values e_k and output values $\hat{e}_k$ for a 16-level Gaussian quantizer with $\sigma_e = 1$; due to symmetry only the positive values ($9 \leq k \leq 16$) are shown.

2.5.2 Quantizer Adaptation

In an early paper by Cummiskey, Jayant, and Flanagan [18], a simple but effective adaptation algorithm was proposed. They introduced a simple rule for generating the step-size, namely "for every new input sample, the step-size is changed by a factor depending only on the knowledge of which quantizer slot was occupied by the previous sample."

Formally, the algorithm is

$$\Delta(n) = \Delta(n-1) \cdot M(|c(n-1)|), \tag{2.12}$$

where $|c(n-1)|$ is the magnitude of the previous codeword and $M(\cdot)$ is a step-size multiplier function. Limits imposed on the step-size determine the resulting dynamic range of the quantizer:

$$\Delta_{\text{min}} \leq \Delta(n) \leq \Delta_{\text{max}}, \tag{2.13a}$$

$$\text{Dynamic range} = 20\log_{10}(\Delta_{\text{max}}/\Delta_{\text{min}})\ \text{dB}. \tag{2.13b}$$

If the maximum step-size is 100 times the minimum step-size, then the dynamic range of the quantizer will be 40 dB; this is a suitable value for speech signals in telephony.

The step-size multiplier function $M(\cdot)$ takes on a specific value from the set $\{M_1, M_2, \ldots, M_K\}$, according to which quantizer level $c(n-1)$ was occupying. This implies that there are only $K = L/2$ multiplier values to be chosen (*i.e.*, half the number of quantizer levels, since the polarity of $c(n-1)$ is unimportant). Figure 2.7 illustrates an 8-level or 3-bit uniform quantizer with associated step-size multipliers.

Meaningful adaptation for speech requires a rapid range expansion to handle sudden bursts of speech energy and slow range compression for decaying pre-pausal sounds. To avoid slope overload and minimize granulation noise the multiplier values should respect the following constraints:

$$M_K < 1, \quad M_1 > 1, \quad \text{and} \quad M_1 \geq M_2 \geq \cdots \geq M_K.$$

The most effective values for the step-size multipliers are found through extensive computer simulations on different types of input signals. Some results for 2, 3, 4, and 5-bit quantizer step-size multipliers are tabulated by Rabiner and Schafer [17], and by Jayant and Noll [15]. The set given in Table 2.2 for a 4-bit quantizer is used in the simulations of Chapter 4.

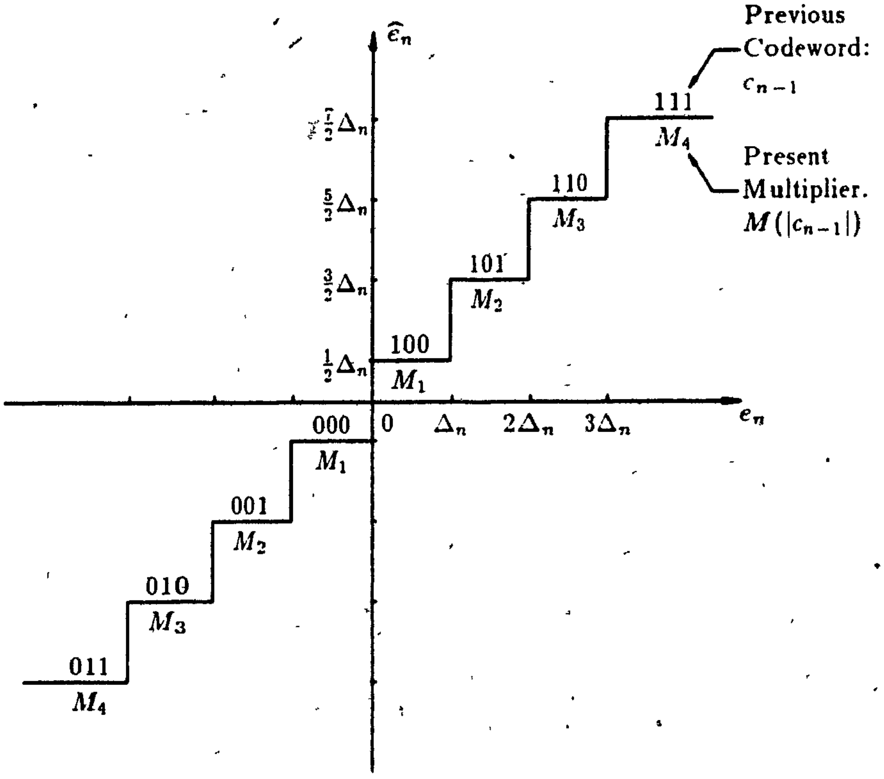


Fig. 2.7 Step-size multipliers for an 8-level quantizer

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
2.4	2.0	1.6	1.2	.9	.9	.9	.9

Table 2.2 Step-size multiplier values for a 4-bit or 16-level quantizer

The algorithm defined by (2.12) is not fully robust to channel errors. This is because each new step-size depends on the entire past of the codeword sequence $c(n)$, thereby forming an infinite memory system where errors propagate indefinitely. Goodman and Wilkinson [19] pointed this out using the following argument: rewrite (2.12) as

$$\Delta(n) = \prod_{k=0}^{n-1} M(|c(k)|) \cdot \Delta(0). \tag{2.14}$$

Let $\Delta'(\cdot)$ and $c'(\cdot)$ be the decoder versions of $\Delta(\cdot)$ and $c(\cdot)$, respectively. Now assume that at time $m < n$, $|c(m)| = i$ while a channel error causes $|c'(m)| = j$. If there are no additional errors, then

$$\Delta'(n) = [M(j)/M(i)] \cdot \Delta(n). \tag{2.15}$$

As a result each error causes a multiplicative offset between the encoder and the decoder that theoretically will persist indefinitely; *i.e.*, quantizer mistracking. The following modification to (2.12) is suggested [19]:

$$\begin{aligned}\Delta(n) &= \Delta(n-1)^{\beta} M(|c(n-1)|) \\ &= \prod_{k=0}^{n-1} [M(|c(k)|)]^{\beta^{(n-k-1)}} \Delta(0)^{\beta^{n}},\end{aligned} \tag{2.16}$$

where $\beta < 1$ is the leakage factor. Now if, at time m, j is received instead of i,

$$\Delta'(n) = [M(j)/M(i)]^{\beta^{(n-m-1)}} \Delta(n). \tag{2.17}$$

Note that when $\beta = 1$, (2.16) and (2.17) simplify to (2.14) and (2.15).

Hence, the offset due to errors will decay exponentially with time, thereby eliminating quantizer mistracking. The time constant τ is controlled by the factor β:

$$\tau = -1/(\ln \beta) \quad \text{samples.}$$

Several values of β with corresponding time constants τ (in ms) are tabulated below.

β	15/16	31/32	63/64	127/128
τ_{ms}	1.94	3.94	7.94	15.94

Table 2.3 Quantizer leakage factor vs. step-size time constant

The design value of β is a compromise between the sensitivity of the quantizer to changes in the signal variance and the sensitivity of the decoder to the effects of channel errors. This implies that quantizer adaptation with leakage will affect the performance of the codec in the absence of channel errors. Compromise values which result in a time constant of 4–10 ms seem to work best for a variety of inputs and error conditions [15][19]. The value $\beta = 63/64$ is used in the coder simulation in Chapter 4.

Chapter 3 Prediction Structures and Adaptation Algorithms

3.1 Speech Signal Model

The motivation for using a differential or predictive coding scheme is based on the fact that a speech signal can be adequately modelled as a linear p-th order autoregressive process, denoted by $AR(p)$. An $AR(p)$ process is defined as

$$x(n) \triangleq \sum_{i=1}^{p} \alpha_i(n) x(n-i) + u(n), \tag{3.1}$$

where the summation term is the predicted component of the speech signal and $u(n)$ is the unpredicted excitation signal. The relationship between the excitation $u(n)$ and the output $x(n)$ could be represented by a slowly time-varying transfer function of the form[1]

$$V_n(z) = \frac{X(z)}{U(z)} = \frac{1}{1 - \sum_{i=1}^{p} \alpha_i(n) z^{-i}}, \tag{3.2}$$

also known as an all-pole model.

[1] An unconventional notation is used in the z-transform equations since we are dealing with non-stationary systems. Conceptually, the subscript n denotes slow time-variation in the parameters $\alpha_i(n)$. Mathematically, the equations are treated under the assumption of short-term stationarity, *i.e.*, with fixed parameters.

In general, most of the poles of (3.2) occur in complex conjugate pairs which model the resonances, or *formants*, of the speech spectrum. However, some of the poles are heavily damped and contribute only to the overall spectral shape.

The excitation signal $u(n)$ varies from a quasi-periodic train of pulses for voiced speech to random noise for unvoiced speech. Thus, the speech signal model consists of a time-varying linear system $V_n(z)$ driven by a set of parameters $\{a_i(n), i = 1, 2, \ldots, p\}$ and excited by a random or quasi-periodic input $u(n)$ (see Figure 3.1).

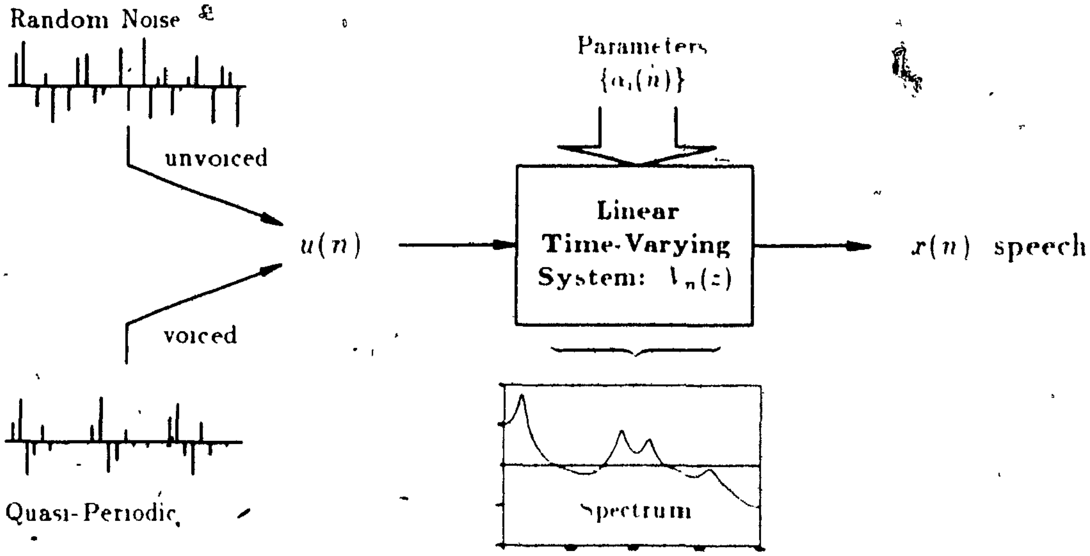


Fig. 3.1 Speech signal model

A linear predictor can be defined as a p-th order FIR filter with coefficients $\{a_i(n), i = 1, 2, \ldots, p\}$ and output

$$\tilde{x}(n) = \sum_{i=1}^{p} a_i(n)\hat{x}(n-i). \tag{3.3}$$

The prediction is formed using past samples of the reconstructed signal $\hat{x}(n)$, as explained in Section 2.2. The predictor transfer function is

$$P_n(z) = \frac{\tilde{X}(z)}{\hat{X}(z)} = \sum_{i=1}^{p} a_i(n) z^{-i} \tag{3.4}$$

$P_n(z)$ is also known as the *analysis* filter. The main task of this FIR filter is to track the "predictable" part of (3.1), namely the summation term $\sum \alpha_i(n)x(n-i)$, by adjusting the coefficients so that $a_i(n) - \alpha_i(n)$. When equality is achieved for stationary inputs ($a_i = \alpha_i$), the prediction residual power will be equal to the excitation power plus the filtered quantization noise:

$$E[e^2(n)] = E[u^2(n)] + \sum_{i=1}^{p} \alpha_i^2 E[q^2(n)], \tag{3.5}$$

assuming the quantization is fine. Recalling Eq. (2.9), the above equation is further simplified to

$$E[e^2(n)] = \frac{E[u^2(n)]}{1 - \epsilon^2 \sum_{i=1}^{p} \alpha_i^2} \approx E[u^2(n)], \tag{3.6}$$

since the factor $\epsilon^2 \sum_{i=1}^{p} \alpha_i^2 \ll 1$ for most speech inputs. Thus, Eq. (3.6) states that the prediction residual power is minimal since $u(n)$ is either random noise or a train of pulses.

The synthesis filter is an AR structure which reconstructs $\hat{x}(n)$ from $\hat{e}(n)$, and is defined by the equation

$$\hat{x}(n) = \tilde{x}(n) + \hat{e}(n) = \sum_{i=1}^{p} a_i(n)\hat{x}(n-i) + \hat{e}(n) \tag{3.7}$$

with transfer function

$$H_n(z) = \frac{\widehat{X}(z)}{\widehat{E}(z)} = \frac{1}{1 - P_n(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_i(n) z^{-i}}. \tag{3.8}$$

Comparing (3.2) and (3.8) it is clear that, were it not for the quantization error $q(n)$, $V_n(z)$ and $H_n(z)$ would become identical when $E[e^2(n)]$ is minimized.

An adaptation algorithm for the coefficients can be derived from a gradient search of minimum $E[e^2(n)]$; *i.e., using the goal to find the means!* At this point, the implementation of $P_n(z)$ becomes important as adaptation algorithms are dependent on filter structure. There are two basic types of structures to be examined in this paper: *transversal* and *lattice*.

3.2 Transversal Filter

This section discusses the transversal filter structure and associated adaptation algorithms. The transversal filter, illustrated in Figure 3.2, is a direct implementation of $P_n(z)$ in (3.4). It is a canonical structure, containing the minimum number of unit delays, p, required to implement a p-th order filter. In order to ensure stability of the synthesis filter, there exists a set of constraints on the allowable range of values for the prediction coefficients.

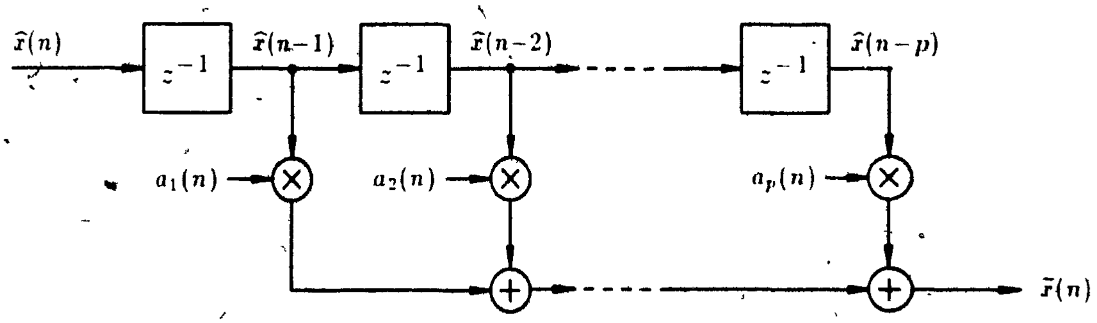


Fig. 3.2 Block diagram of a p-th order transversal filter

Region of Stability for a Second Order Filter. The poles of $H_n(z)$ must lie inside the unit circle in order for the synthesis filter to be stable. Limiting the discussion to a second order filter and dropping the time index n, (3.8) is written here as

$$H(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} = \frac{z^2}{z^2 - a_1 z - a_2}.$$

The poles of $H(z)$ are simply the roots of the denominator

$$z^2 - a_1 z - a_2 = 0 \quad \Longrightarrow \quad z_{1,2} = \frac{a_1}{2} \pm \sqrt{\frac{a_1^2}{4} + a_2}.$$

Imposing the constraint $|z| < 1$ on the roots $z_{1,2}$ yields the following constraints on the coefficients:

$$a_1 < 1 - a_2,$$

$$a_2 < 1.$$

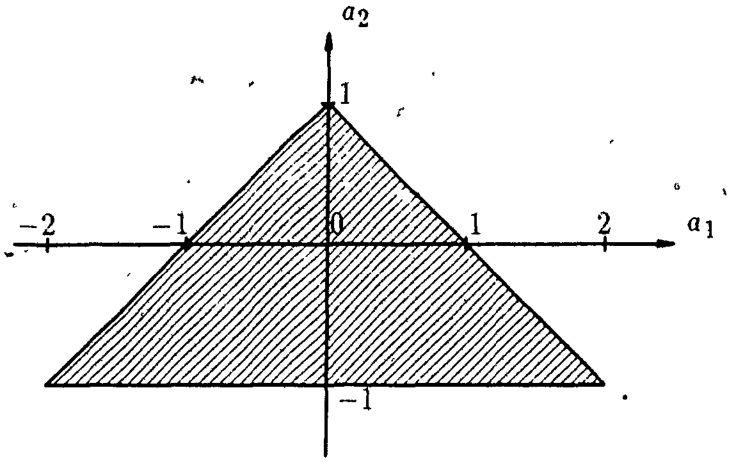


Fig. 3.3 Stability region for a second order filter: $H_n(z)$

The resulting stability region is shown in Figure 3.3. Therefore, after adaptation, the stability of $H_n(z)$ must be enforced by limiting the coefficients to the stable region.

In practice, the stable region is slightly reduced by a factor $(1 - \epsilon)$, where ϵ is a small positive constant. This is done to keep the synthesis filter from resonating when the coefficients are near the boundary. The constraints for the reduced region are

$$|a_1| \leq 1 - a_2 - \epsilon, \tag{3.9a}$$

$$|a_2| \leq 1 - \epsilon, \qquad \epsilon \gtrsim 0. \tag{3.9b}$$

For higher order transversal filters the roots of the polynomial are difficult to compute, and the region is more complex. In this case, the easiest method for checking stability is to use a non-linear mapping $\mathcal{M}$ to transform the $\{a_i\}$ to a set $\{k_i\}$, known as reflection coefficients [17]. The synthesis filter is guaranteed to be stable if the magnitudes $|k_i| < 1$ (see Section 3.3.1). So, limiting the $\{k_i\}$ to this region and mapping back ($\mathcal{M}^{-1}$) will yield a stable set of $\{a_i\}$.

3.2.1 Mean Square Error (MSE) Formulation of the Prediction Residual

As mentioned in Section 3.1, the main goal of the predictor is to minimize the residual power, or mean square error (MSE). For FIR filters, the MSE equation $E[e^2(n)]$ can be formulated as a function of the coefficients $\{a_i\}$, given a stationary input $x(n)$.

Using a compact vector notation and assuming wide-sense stationarity, the prediction residual of (2.1) is written here as

$$e(n) = x(n) - \mathbf{A}^{\mathrm{T}}\widehat{\mathbf{X}}(n-1) = x(n) - \widehat{\mathbf{X}}^{\mathrm{T}}(n-1)\mathbf{A}, \tag{3.10}$$

where

$$\mathbf{A} \overset{\triangle}{=} [\, a_1 \quad a_2 \quad \dots \quad a_p \,]^{\mathrm{T}}$$

is the prediction coefficient vector, assumed to be in steady-state, and

$$\widehat{\mathbf{X}}(n-1) \overset{\triangle}{=} [\, \hat{x}(n-1) \quad \hat{x}(n-2) \quad \dots \quad \hat{x}(n-p) \,]^{\mathrm{T}}$$

is the reconstructed signal *information* vector. If the quantization is fine and the predictor is tracking the input, so that $q(n)$ is uncorrelated with $x(n)$ and the quantization error power is much smaller than the input signal power, the following approximation can be made:

$$\widehat{\mathbf{X}}(n-1) \approx \mathbf{X}(n-1) \overset{\triangle}{=} [\, x(n-1) \quad x(n-2) \quad \dots \quad x(n-p) \,]^{\mathrm{T}}. \tag{3.11}$$

After squaring and taking the expectation of (3.10), we get the MSE equation:

$$\begin{aligned} E[e^2(n)] =& E[x^2(n)] - 2E[x(n)\mathbf{X}^{\mathrm{T}}(n-1)]\mathbf{A} \\ &+ \mathbf{A}^{\mathrm{T}}E[\mathbf{X}(n-1)\mathbf{X}^{\mathrm{T}}(n-1)]\mathbf{A}. \end{aligned} \tag{3.12}$$

This quantity can be plotted as a function of the coefficients to yield an error surface for the stationary input $x(n)$. In the case of transversal filters, the surface is a quadratic function, concave upward, with a single global minimum. By superimposing coefficient trajectories onto the error surface, we obtain an indicator of the tracking performance of an adaptation algorithm. Plots of the error surface for second order predictors are used in Chapter 4.

By defining the autocorrelation function of $x(n)$ as

$$\phi(i-j) \overset{\triangle}{=} E[x(n-i)x(n-j)] = \phi(j-i), \quad \text{for all } i, j, \tag{3.13a}$$

the autocorrelation vector as

$$\begin{aligned} \mathbf{P} &\overset{\triangle}{=} E[x(n)\mathbf{X}^{\mathrm{T}}(n-1)] \\ &= [\,\phi(1) \quad \phi(2) \quad \ldots \quad \phi(p)\,]^{\mathrm{T}}, \end{aligned} \tag{3.13b}$$

and the symmetric autocorrelation matrix as

$$\begin{aligned} \mathbf{\Phi} &\overset{\triangle}{=} E[\mathbf{X}(n-1)\mathbf{X}^{\mathrm{T}}(n-1)] \\ &= \begin{bmatrix} \phi(0) & \phi(1) & \ldots & \phi(p-1) \\ \phi(1) & \phi(0) & \ldots & \phi(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(p-1) & \phi(p-2) & \ldots & \phi(0) \end{bmatrix}, \end{aligned} \tag{3.13c}$$

the MSE equation (3.12) can be formulated as

$$E[\epsilon^2(n)] = \phi(0) - 2\mathbf{P}^{\mathrm{T}}\mathbf{A} + \mathbf{A}^{\mathrm{T}}\mathbf{\Phi}\mathbf{A}. \tag{3.13d}$$

Note that none of the quantities in (3.13a)-(3.13d) is a function of time, due to the stationarity assumption.

It is clear that (3.13d) is a quadratic form in the coefficient vector **A**. Hence, there exists a unique minimum solution in **A** which can be obtained by completing the square or setting the gradient equal to zero. The optimal solution and resulting minimum mean-square-error value are

$$\mathbf{A}_{\mathrm{opt}} = \mathbf{\Phi}^{-1}\mathbf{P} \tag{3.14a}$$

and

$$\begin{aligned} E[\epsilon^2(n)]_{\min} &= \phi(0) - \mathbf{P}^{\mathrm{T}}\mathbf{\Phi}^{-1}\mathbf{P} \\ &= \phi(0) - \mathbf{P}^{\mathrm{T}}\mathbf{A}_{\mathrm{opt}}. \end{aligned} \tag{3.14b}$$

Brute force calculation of $\mathbf{A}_{\text{opt}}$ in (3.14a) requires the solution of a system of linear equations, as indicated by the matrix formulation. Although there exists an efficient algorithm for the solution (the *Levinson-Durbin procedure* [17]), a different approach is taken here leading directly to the stochastic gradient algorithm.

3.2.2 Minimum MSE Solution for Optimal Coefficients

The derivation of the optimal coefficient vector is based on minimizing the mean-square error $E[e^2(n)]$. Simply differentiating this quantity with respect to the coefficients and setting the result to zero will establish a set of conditions for reaching the optimal solution.

The error gradient is defined as

$$\nabla_{\mathbf{A}} \triangleq \frac{\partial}{\partial \mathbf{A}} E[e^2(n)]. \tag{3.15}$$

Setting this to zero, we obtain

$$\begin{aligned} \mathbf{0} &= \nabla_{\mathbf{A}} \\ &= 2E[e(n)\frac{\partial}{\partial \mathbf{A}}e(n)] \\ &= -2E[e(n)\widehat{\mathbf{X}}(n-1)]. \end{aligned} \tag{3.16}$$

Equation (3.16) demonstrates that, as a side effect, the predictor residual and reconstructed signal are uncorrelated when the optimal coefficient set has been found. This is known as the *principle of orthogonality* [20]. The residual is uncorrelated with the input signal as well, based on the assumption leading to (3.11).

Adapting the coefficients in a direction opposing the gradient will ensure that the optimal set is approached after a finite number of iterations. Thus, the gradient adaptation algorithm can be written as

$$\begin{aligned} \mathbf{A}(n+1) &= \mathbf{A}(n) - \frac{\mu}{2}\nabla_{\mathbf{A}(n)} \\ &= \mathbf{A}(n) + \mu E[e(n)\widehat{\mathbf{X}}(n-1)], \end{aligned} \tag{3.17}$$

where μ is a small adaptation gain constant or step-size parameter which scales the size of the change in $\mathbf{A}(n)$ at each update. This is equivalent to perturbing the coefficients towards the global minimum by descending the error surface in a direction opposing the gradient (which is the direction of *maximum* increase in the mean-square error).

The algorithm will converge if the gradient direction can be assumed unchanged over the region of the step. Thus, the step-size parameter μ must be large enough so the algorithm converges quickly to $\mathbf{A}_{\text{opt}}$, but not so large that the algorithm hunts in the vicinity of the minimum.

The selection of μ depends on the eigenvalues of the input autocorrelation matrix $\mathbf{\Phi}$. The range of values for μ which ensures convergence is

$$0 < \mu < \frac{2}{\lambda_{\text{max}}},$$

where λ_{max} is the largest eigenvalue of $\mathbf{\Phi}$. The optimal value, resulting in the fastest convergence, is

$$\mu_{\text{opt}} = \frac{2}{\lambda_{\text{min}} + \lambda_{\text{max}}}.$$

For when $\mu = \mu_{\text{opt}}$, the adaptation modes corresponding to both the minimum and maximum eigenvalues converge at the same rate:

$$\text{convergence rate} \propto \left(\frac{\frac{\lambda_{\text{max}}}{\lambda_{\text{min}}} - 1}{\frac{\lambda_{\text{max}}}{\lambda_{\text{min}}} + 1} \right)^{n}.$$

Thus, the factor $\lambda_{\text{max}}/\lambda_{\text{min}}$, called the *eigenvalue spread*, determines the speed of convergence of the gradient algorithm [20]. That this quantity depends on the input signal is a major drawback in the use of adaptive transversal filters.

3.2.3 Gradient Algorithm for a Time-Varying Environment

In a practical coding environment, the input signals are time-varying but can be considered stationary in the sort-term, as discussed in Section 2.4. Thus, two modifications must be applied to the basic gradient algorithm in order to track the changing signal: coefficient leakage and step-size normalization.

Coefficient Leakage. The first modification involves the insertion of coefficient leakages so that in the absence of an input to the algorithm, *i.e.* $\nabla_{A(n)} = 0$, the coefficient vector will decay to the origin.[2]

The use of leakage ensures that the algorithms at the encoder and decoder converge after a channel error has caused the decoder algorithm to **momentarily** have incorrect input data. However, as shall be shown later, mistracking is not prevented if such an error corrupts the input data **indefinitely**; *i.e.*, by permanently changing the decoder reconstructed sequence $\hat{x}'(n)$.

An interesting method of applying coefficient leakage is found by radially scaling the poles of the synthesis filter $H_n(z)$ by a factor $\beta < 1$. Suppose $H_n(z)$ has a pole at $z = z_1$, then $H_n(\beta^{-1}z)$ will have an equivalent pole at $z = \beta z_1$. Letting $H_{n+1}(z) = H_n(\beta^{-1}z)$, we get

$$H_{n+1}(z) \triangleq \frac{1}{1 - \sum_{i=1}^{p} a_i(n+1)z^{-i}} = \frac{1}{1 - \sum_{i=1}^{p} a_i(n)\beta^i z^{-i}}, \tag{3.18a}$$

which implies that

$$a_i(n+1) = \beta^i a_i(n), \qquad i = 1,2,\ldots,p \tag{3.18b}$$

when $\nabla_{A(n)} = 0$. This way all the poles of the synthesis filter decay to the origin at the same rate. In the literature, β is usually denoted as $(1-\delta)$, where δ is a small positive value. Using the approximation $(1-\delta)^i \simeq (1-i\delta)$ for small δ, (3.18b) simplifies to

$$a_i(n+1) = (1-i\delta)a_i(n), \qquad i = 1,2,\ldots,p. \tag{3.19}$$

[2] Some forms of the algorithm have the coefficient vector decay to a non-zero point, usually an average point for speech signals that is determined experimentally. This allows faster convergence when going from silence to speech, but has no other consequence in the analysis of predictor behaviour

Step-Size Normalization. The second modification is the normalization of the step-size parameter so that the size of the updates in (3.17) remain approximately the same when the input signal level varies considerably. This will ensure that convergence of the gradient algorithm is not strongly dependent on input signal power when the signal is not stationary. Most differences between various adaptation algorithms for transversal predictors evolve around the step-size normalization.

The most commonly used normalizer is the reconstructed signal power $E[\widehat{x}^2(n)]$, this being approximately equal to the input power. This is equivalent to having a possibly time-varying step-size parameter

$$\mu(n) = \frac{\mu}{E[\widehat{x}^2(n)] + K}, \tag{3.20}$$

where K is an appropriately chosen constant with the purpose of preventing $\mu(n)$ from growing too large when the reconstructed signal power is small.

In the computer implementation of the algorithm, K is set to zero, and the updates at time n are simply not performed if $E[\widehat{x}^2(n)]$ is less than a fixed threshold value; *i.e.*, $a_i(n+1) = a_i(n)$. Subsequent experiments show that this does not introduce any *limit cycle* problems where the coefficients "lock-up" for long periods of time.

Substituting (3.19) and (3.20) (with $K = 0$) into (3.17) and replacing $e(n)$ by $\widehat{e}(n)$ (since the former signal is not available at the decoder) yields the modified gradient algorithm, written here in scalar notation,

$$a_i(n+1) = (1 - i\delta)a_i(n) + \frac{\mu E[\widehat{e}(n)\widehat{x}(n-i)]}{E[\widehat{x}^2(n)]}, \qquad i = 1, 2, \ldots, p. \tag{3.21}$$

So far, the discussion has dealt with the motivation and derivation of the generic gradient algorithm for adapting a transversal predictor. In the following section we show that algorithms of the form (3.21) are susceptible to predictor mistracking.

3.2.4 Effects of a Channel Error: Possibility of Mistracking

3.2.4.1 Traditional Adaptation

In traditional backward adaptive prediction the reconstruction processes at the encoder and decoder can be described as systems in which a reconstruction filter $\mathcal{F}$ is driven by an adaptation algorithm $\mathcal{G}$ with system input $\hat{e}(n)$, output $\hat{x}(n)$, and coefficient vector $\mathbf{A}(n)$. Without regard to the implementation details of $\mathcal{F}$ and $\mathcal{G}$, filter/algorithm interaction can be described using a simple operator notation, where the signal reconstruction is

$$\hat{x}(n) = \mathcal{F}\Big\{\hat{e}(n); \widehat{\mathbf{X}}(n-1); \mathbf{A}(n)\Big\}, \tag{3.22}$$

and the predictor adaptation is

$$\mathbf{A}(n+1) = \mathcal{G}\Big\{\hat{e}(n); \widehat{\mathbf{X}}(n-1); \mathbf{A}(n)\Big\}. \tag{3.23}$$

The reconstruction process, illustrated conceptually in Figure 3.4, is in a coupled configuration where the output of $\mathcal{F}$ is fed-back to $\mathcal{G}$, and vice versa. The p-sample buffer simply generates the information vector $\widehat{\mathbf{X}}(n-1)$ from $\hat{x}(n)$ and the z^{-1} block delays the coefficient vector by one sample.

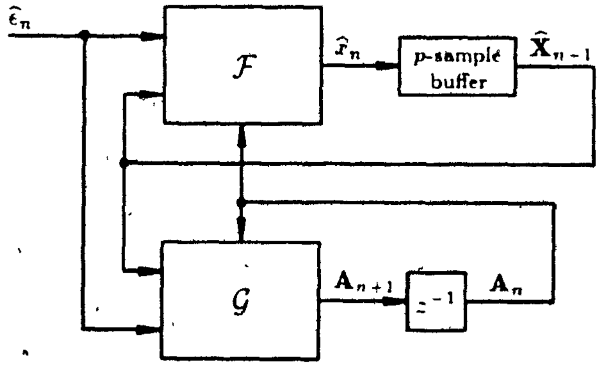


Fig. 3.4 Reconstruction process (signal-driven algorithm)

For the ADPCM systems considered here, the $\mathcal{F}$ and $\mathcal{G}$ operators are computed identically at the encoder and the decoder but the inputs $\hat{e}(n)$ and $\hat{e}'(n)$ can differ due to a transmission error. Suppose that at time m a transmission error causes

$$\hat{e}'(m) = \hat{e}(m) + \varepsilon$$

to be received at the decoder. The immediate effect of this error will be slight perturbations in $\hat{x}'(m)$ and $\mathbf{A}'(m+1)$, depending on the value of ε and the state of the decoder process. These perturbations will propagate around $\mathcal{F}$ and $\mathcal{G}$ in the feedback and cross-feedback loops, resulting in persistently incorrect information at the inputs of both blocks at the decoder. This will happen even when $\hat{e}'(n) = \hat{e}(n)$ for $n > m$, or when the received signal is identical to the transmitted signal after the error has occurred. In the long-term the error may force the decoder process into a different mode of operation, perhaps an unstable one, where $\hat{x}'(n) \neq \hat{x}(n)$ and $\mathbf{A}'(n) \neq \mathbf{A}(n)$.

Theoretically, convergence of the two processes after an initial offset is ensured only if the quantized residual is trivially zero (or white and zero-mean) for an extended period of time. Then the driving term in (3.21) will disappear, enabling the coefficients to decay exponentially to zero at both ends (due to leakage). This will force the poles of the synthesis filter to the origin, effectively resetting both reconstruction processes.

In fact, this effect is witnessed with speech signals where periods of low energy (or silence) between utterances and the wideband nature of fricatives both serve as natural re-synchronizers for the reconstruction process (see Section 4.3.2 for experimental results).

In the case of narrowband inputs such as DTMF signals, however, convergence cannot be ensured. Looking ahead at Section 4.4.2 Figure 4.10 shows, for the dual-tone input signal DTMF-3 (composed of two equi-amplitude sinusoids at 697 and 1477 Hz), the coefficient trajectories for three second-order prediction processes (one at the encoder and two at the decoder), each with different initial conditions. We see that the encoder (unaffected by transmission errors) is tracking near the optimal point for the DTMF-3 signal, but the two decoders are tracking near points corresponding to the distinct frequencies 697 and 1477 Hz. The dotted line is the boundary of the region of stability for a second order predictor and the three contours represent the error surfaces of DTMF-3, 697 Hz sinusoid, and 1477 Hz sinusoid signals. The resulting power twist in the reconstructed output signals at the decoders can be as large as 20 dB (see Figure 4.11). Thus, mistracking is a phenomenon normally associated with stationary narrow-band input signals, such as voiceband data (vbd) and signalling tones.

3.2.4.2 Modified Adaptation

Tracking can be ensured for all signals if the cross-feedback path from the output of $\mathcal{F}$ to the input of $\mathcal{G}$ is eliminated, thereby de-coupling the two blocks (Figure 3.5 illustrates such an arrangement). With this modification, $\mathcal{G}$ is no longer influenced by incorrect input data $\hat{x}'(n)$, and leakage will allow the coefficients to re-track in a finite number of samples after the error has occurred. The reconstructed signal at the decoder will also converge due to the uniqueness property of the reconstruction process: *i.e.*, if $\hat{e}'(n) = \hat{e}(n)$, $\mathbf{A}'(n) = \mathbf{A}(n)$ and $\hat{x}'(n) \neq \hat{x}(n)$ then $\hat{x}'(k) \rightarrow \hat{x}(k)$ for $k > n$, assuming no additional errors [21].

As a result, algorithms driven only by current and previous samples of the quantized residual are not susceptible to mistracking. Thus, we define a new class of adaptation algorithms such that

$$\mathbf{A}(n+1) = \mathcal{G}\Big\{\hat{e}(n); \hat{\mathbf{E}}(n-1); \mathbf{A}(n)\Big\}, \tag{3.24}$$

where the residual vector

$$\widehat{\mathbf{E}}(n-1) = [\,\widehat{e}(n-1) \quad \widehat{e}(n-2) \quad \ldots \quad \widehat{e}(n-p)\,]^{\mathrm{T}}$$

replaces $\widehat{\mathbf{X}}(n-1)$ in (3.23). However, using $\widehat{\mathbf{E}}(n-1)$ instead of $\widehat{\mathbf{X}}(n-1)$ entails a loss of performance in the absence of errors. This is because in the context of MMSE gradient estimation the quantized residual contains significantly less information about the input than does the reconstructed signal. Such performance/robustness tradeoffs are often encountered in adaptive system design. In this case the loss in performance can be minimized by careful design of $\mathcal{G}$ in (3.24).

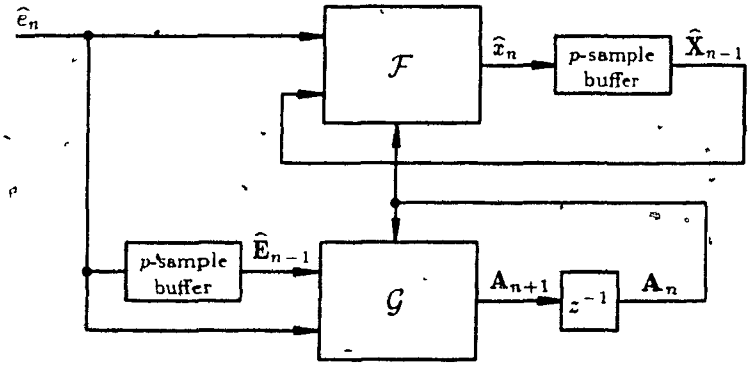


Fig. 3.5 Reconstruction process without cross-feedback from $\mathcal{F}$ to $\mathcal{G}$ (residual-driven algorithm)

The labels "**signal-driven**" and "**residual-driven**" will be used to denote whether the adaptation algorithm is of the form (3.23) or (3.24). The next section describes specific implementations of the gradient algorithm, one signal-driven and one residual-driven, both using simple estimates of the gradient.

3.2.5 Transversal Stochastic Gradient Algorithm (LMS)

3.2.5.1 Signal-Driven Adaptation

The gradient algorithm in (3.21) is only applicable to short-term stationary signals and uses ensemble averages that are not readily available in the system. The stochastic gradient (SG) algorithm, on the other hand, uses an instantaneous estimate of the deterministic gradient in (3.16), given by

$$\widehat{\nabla}_{\mathbf{A}(n)} = -2\widehat{e}(n)\widehat{\mathbf{X}}(n-1). \tag{3.25}$$

This results in a noisy gradient vector with the quantized residual $\widehat{e}(n)$ replacing $e(n)$. Substituting this quantity in (3.17) and employing coefficient leakage and step-size normalization as in (3.21) yields the signal-driven SG algorithm,

$$a_i(n+1) = (1-i\delta)a_i(n) + \frac{\mu\widehat{e}(n)\widehat{x}(n-i)}{\widehat{\sigma}_x^2(n)}, \qquad i = 1,2,\ldots,p, \tag{3.26a}$$

where

$$\begin{aligned}\widehat{\sigma}_x^2(n) &= (1-\lambda)\sum_{i=0}^{\infty}\lambda^i\widehat{x}^2(n-i)\\ &= \lambda\widehat{\sigma}_x^2(n-1) + (1-\lambda)\widehat{x}^2(n), \qquad 0 < \lambda < 1\end{aligned} \tag{3.26b}$$

is an unbiased estimate of the reconstructed signal power. The constant λ determines the memory in the normalization term

This update formula is very easily implemented in real-time digital hardware within the allotted time posed by the sampling rate. This is the main reason the SG algorithm has been preferred over other sequential algorithms. The price to be paid for using the SG method, instead of a more accurate estimator of the gradient, is *(i)* slower convergence and *(ii)* larger fluctuations about $\mathbf{A}_{\text{opt}}$ once convergence has occurred. This is due to the noisy estimates in (3.25) [20].

3.2.5.2 Residual-Driven Adaptation

Millar and Mermelstein [8] have suggested a residual-driven algorithm for a *second order* transversal predictor which has been adopted by the CCITT as part of an international standard for 32 kb/s ADPCM [5]. The actual adaptation scheme proposed in [8] and [5] is a simplified version of what is to follow, and will be discussed in Section 3.2.5.3. The approach taken is based on the fact that MA predictors (*i.e.*, ones which lead to all-zero synthesis filters) do not give rise to mistracking. This suggests that a two pole synthesis filter $H_n(z)$ can be represented in terms of an infinite number of zeros,

$$H_n(z) = \frac{1}{1 - \sum_{i=1}^{2} a_i(n) z^{-i}} = 1 + \sum_{i=1}^{\infty} c_i(n) z^{-i}. \tag{3.27}$$

A SG adaptation for a p-th order MA predictor with coefficients $c_i(n)$, input $\tilde{e}(n)$ and output $\tilde{x}(n)$ is given by [8][14]

$$c_i(n+1) = (1 - i\delta)c_i(n) + \frac{\mu \hat{\tilde{e}}(n)\hat{e}(n-i)}{\hat{\sigma}_{\tilde{e}}^2(n)}, \qquad i = 1, 2, \ldots, p. \tag{3.28}$$

The above equation is derived by solving and estimating the MMSE solution for a MA predictor, and is similar to (3.26a) but with the updates driven only by the quantized residual and the normalization based only on the residual power.

The adaptation procedure for the $a_i(n)$ is found by approximating $H_{n+1}(z)$ by the infinite summation in (3.27) at time $n+1$, after appropriate updates to the first two zero coefficients $c_i(n)$, $i = 1, 2$. The remaining $c_i(n)$ may assume "natural" values at time $n+1$. Thus, for a second order predictor, setting $\delta = 0$ and $p = 2$ in (3.28) and substituting the result into (3.27) at time $n+1$, will give

$$\begin{aligned} H_{n+1}(z) &= \frac{1}{1 - \sum_{i=1}^{2} \left(a_i(n) + \Delta_{a_i}(n)\right) z^{-i}} \\ &\simeq 1 + \sum_{i=1}^{2} \left(c_i(n) + \Delta_{c_i}(n)\right) z^{-i} + \sum_{i=3}^{\infty} c_i(n+1) z^{-i}. \end{aligned} \tag{3.29}$$

Now, equating like powers of z yields

$$\Delta_{a_1}(n) = \Delta_{c_1}(n),$$

$$\Delta_{a_2}(n) = \Delta_{c_2}(n) - 2a_1(n)\Delta_{c_1}(n),$$

where Δ refers to a coefficient update. The resulting adaptation algorithm is [8]

$$a_1(n+1) = (1 - \delta)a_1(n) + \frac{\mu \tilde{e}(n)\tilde{e}(n-1)}{\hat{\sigma}_{\tilde{e}}^2(n)}, \tag{3.30a}$$

$$a_2(n+1) = (1-2\delta)a_2(n) + \frac{\mu}{\hat{\sigma}_{\tilde{e}}^2(n)} \cdot \Big(\tilde{e}(n)\tilde{e}(n-2) - 2a_1(n)\tilde{e}(n)\tilde{e}(n-1)\Big), \tag{3.30b}$$

where

$$\begin{aligned}\hat{\sigma}_{\tilde{e}}^2(n) &= (1-\lambda)\sum_{i=0}^{\infty} \lambda^i \tilde{e}^2(n-i) \\ &= \lambda\hat{\sigma}_{\tilde{e}}^2(n-1) + (1-\lambda)\tilde{e}^2(n), \qquad 0 < \lambda < 1.\end{aligned} \tag{3.30c}$$

Note that this satisfies a reconstruction process of the form shown in Figure 3.5 with $\hat{x}(n)$ not appearing anywhere in the above update equations.

Robustness of the Algorithm. To prove that mistracking due to channel errors is eliminated it is sufficient to show that the algorithms at the encoder and decoder converge after an initial offset in the coefficients. That is, simply let $\big(p_1(n), p_2(n)\big)$ and $\big(q_1(n), q_2(n)\big)$ be two distinct solutions to (3.30a) and (3.30b), each with different initial conditions and identical quantized residual sequence $\tilde{e}(n)$. Then the difference between the first coefficients of the two solutions,

$$p_1(n+1) - q_1(n+1) = (1-\delta)\big(p_1(n) - q_1(n)\big), \tag{3.31a}$$

approaches zero exponentially with time constant $1/\delta$. Similarly,

$$\begin{aligned}p_2(n+1) - q_2(n+1) = (1-2\delta)\big(p_2(n) - q_2(n)\big) \\ - \frac{2\mu\tilde{e}(n)\tilde{e}(n-1)}{\hat{\sigma}_{\tilde{e}}^2(n)}\big(p_1(n) - q_1(n)\big)\end{aligned} \tag{3.31b}$$

Assuming an upper bound on the factor

$$\left|\frac{2\mu\tilde{e}(n)\tilde{e}(n-1)}{\hat{\sigma}_{\tilde{e}}^2(n)}\right| \le M < \infty,$$

the magnitude of (3.31b) can be bounded by

$$\begin{aligned}|p_2(n+1) - q_2(n+1)| \le (1-2\delta)|p_2(n) - q_2(n)| \\ + M|p_1(n) - q_1(n)|.\end{aligned}$$

where the first term decays to zero with time constant $\simeq 1/2\delta$ and the second term decays to zero due to (3.31a). Thus, the coefficients are shown to converge from different initial conditions.

The assumption that the encoder and decoder algorithms are driven by identical signals $\widehat{e}(n)$ is a valid one, since a channel error affects the received signal for only a finite number of samples (due to robust quantization). Naturally, the actual recovery time after a channel error is longer if the received signal $\widehat{e}'(n) \neq \widehat{e}(n)$ for a finite number of samples.

The fact that this algorithm is sub-optimal, due to the approximation in (3.29), is not as crucial as the prevention of mistracking. We can, therefore, allow a slight loss in predictor performance in order to ensure robustness in the presence of errors (performance/robustness tradeoff).

3.2.5.3 Residual-Driven Adaptation Using Sign Correlation Multipliers

In order to reduce the complexity of implementation, the adaptation algorithm in (3.30) can be further simplified by using only the sign information of the residual samples; *i.e.*,

$$a_1(n+1) = (1-\delta)a_1(n) + \mu' \operatorname{sgn}[\widehat{e}(n)] \operatorname{sgn}[\widehat{e}(n-1)], \tag{3.32a}$$

$$\begin{aligned} a_2(n+1) = (1-2\delta)a_2(n) + \mu'\Big(&\operatorname{sgn}[\widehat{e}(n)] \operatorname{sgn}[\widehat{e}(n-2)] \\ &- 2a_1(n)\operatorname{sgn}[\widehat{e}(n)]\operatorname{sgn}[\widehat{e}(n-1)]\Big), \end{aligned} \tag{3.32b}$$

where sgn[] is the signum function. Generally speaking, the step-size parameter μ' should be the same as μ in (3.30). However, in subsequent experiments μ' is set to $\frac{\mu}{2}$ in order to optimize the performance of all three transversal algorithms with respect to the same range of values in μ. This is not possible when $\mu' = \mu$. Note, also, that normalization by $\hat{\sigma}_e^2(n)$ is not required when the sign multiplier is used instead of the true multiplier.

It is known that the use of such a crude correlation multiplier does not affect the convergence of the coefficients in the steady-state [22]. The speed of convergence, however, is decreased since the updates are no longer sensitive to changes in the residual sample magnitudes. As a result, there is a slight performance degradation associated with the sign multiplier, particularly with non-stationary inputs. Results for second order predictors indicate that, on the average, this scheme suffers from less than a 1 dB loss in prediction gain with speech signals compared to the true multiplier algorithm; this is even lower with dual-tone inputs (see Section 4.3.1, Table 4.3).

The original motivation for using sign multipliers in the updates was to reduce implementation complexity [8],[5]. However, in this paper we are not concerned with hardware implications such as complexity — only with algorithm performance. So there is a different reason for including the sign algorithm in this discussion: After simulations of the true multiplier algorithm in (3.30) it was discovered that values of the parameters μ and δ which yield good speech performance result in very poor dual tone performance, and vice-versa (see Section 4.2.1). Moreover, it was impossible to select compromise values of μ and δ without severely degrading the performance for either type of input signal. This was found to be true regardless of the value of λ in (3.30). Thus, the performance of the transversal predictor utilizing residual-driven adaptation (3.30) is highly sensitive to: *(i)* the type of input (speech or dual-tone) and *(ii)* the parameters μ and δ. In particular, it was observed that good performance for dual-tone or other narrowband inputs is limited to a very narrow region in the parameter space (μ, δ).

This is also true of the sign multiplier algorithm in (3.32), but to a lesser extent.

In this case, there exist some compromise values of μ and δ which yield relatively good performance for both types of input; of course, an unequal weighting is given to the selection of values, favouring speech performance over dual-tone performance.

Robustness of the Sign Multiplier Algorithm. Using an analysis identical to that of the previous section, it can be easily shown that the sign algorithm is also immune to mistracking. Note that, in this case, M is always equal to $2\mu'$.

3.2.5.4 Disadvantages of Transversal Residual-Driven Algorithm

There are two main problems associated with the residual-driven algorithm for the transversal filter (both true and sign-multiplier forms). First, the performance is sensitive to parameter values and type of input, although to a lesser extent with the sign algorithm. Second, the algorithm cannot easily be extended for higher order predictors. This is because equating higher order powers of z in (3.29) yields increasingly complex update terms $\Delta_{a_i}(n)$.

Hence, it would be desirable to find a residual-driven adaptation scheme which is independent of predictor order and whose performance is not so sensitive to parameter values and types of input. This leads us to a discussion of the lattice filter.

3.3 Lattice Filter

The lattice filter is an alternate realization of the linear predictor $P_n(z)$ in (3.4). The filter is a modular structure composed of a cascade of identical lattice stages, as illustrated in Figure 3.6.

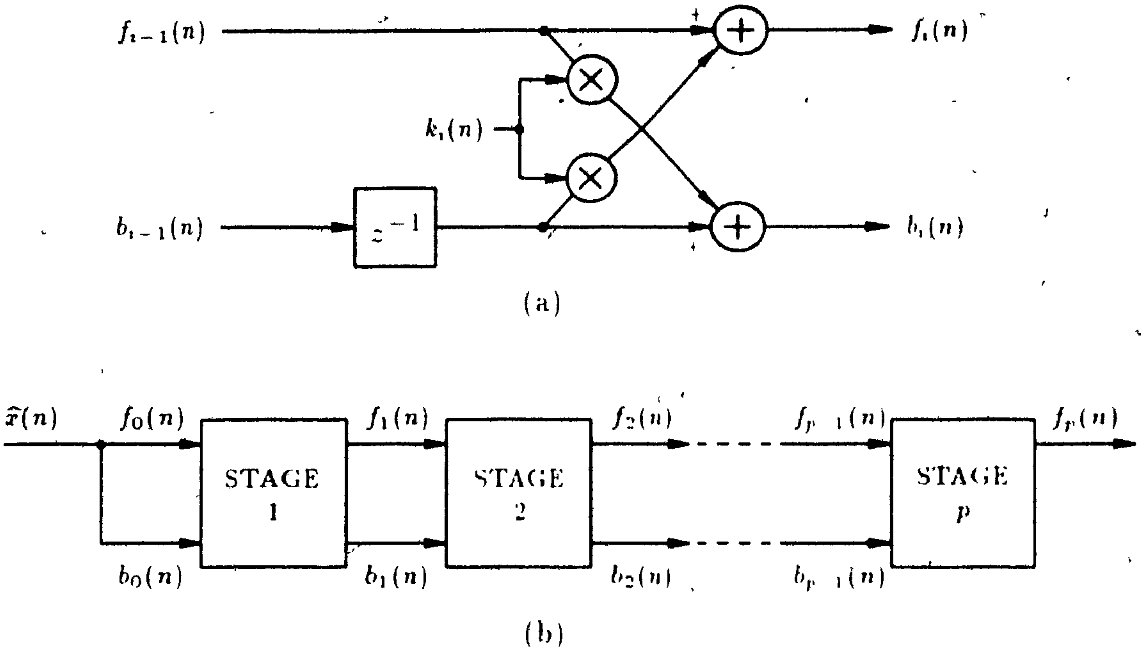


Fig. 3.6 Lattice filter: (a) single stage i; (b) overall p-th order structure

Each stage is described by the *order update* equation

$$f_i(n) = f_{i-1}(n) - k_i(n)b_{i-1}(n-1), \tag{3.33a}$$

$$b_i(n) = b_{i-1}(n-1) - k_i(n)f_{i-1}(n), \tag{3.33b}$$

where $k_i(n)$ is the reflection coefficient for stage i, sometimes referred to as the *partial correlation* (PARCOR) coefficient. The input to the first stage is defined by the initial condition

$$f_0(n) = b_0(n) = \hat{x}(n). \tag{3.34}$$

The signals $f_i(n)$ and $b_i(n)$ are forward and backward prediction residuals of order i. This is shown by iterating (3.33a) and (3.33b) with the initial condition (3.34), so that

$$f_i(n) = \hat{x}(n) - \sum_{j=1}^{i} a_j^{(i)}(n)\hat{x}(n-j), \tag{3.35a}$$

$$b_i(n) = \hat{x}(n-i) - \sum_{j=1}^{i} c_j^{(i)}(n)\hat{x}(n-i+j), \tag{3.35b}$$

where the summation terms above correspond to i-th order forward and backward predictions of $\hat{x}(n)$ and $\hat{x}(n-i)$, respectively. Note that $f_i(n)$ and $b_i(n)$ are *true* prediction residuals of the reconstructed signal, unlike $e(n)$ in (2.1), which is a *noisy* residual of the input signal. The optimal forward and backward coefficients of order i are related by

$$a_{j,\mathrm{opt}}^{(i)} = c_{j,\mathrm{opt}}^{(i)}, \qquad 1 \leq j \leq i, \tag{3.36}$$

assuming wide-sense stationarity.

Successive stages of the filter generate higher order forward and backward prediction errors, where the inputs to the first stage correspond to prediction residuals of order 0 and the output of the final stage is the overall forward prediction error $f_p(n)$. Hence, the lattice filter whitens its input by removing increasingly better predictions at every stage.

However, the structure in Figure 3.6 cannot be used directly as a predictor $P_n(z)$ because the output is a residual signal rather than a prediction signal. Iterating (3.33a) in terms of the backward residuals yields

$$f_p(n) = \hat{x}(n) - \sum_{i=1}^{p} k_i(n)b_{i-1}(n-1), \tag{3.37}$$

where the summation term corresponds to a p-th order forward prediction (due to (3.35a))

$$\bar{x}(n) = \sum_{i=1}^{p} k_i(n)b_{i-1}(n-1) = \sum_{i=1}^{p} a_i^{(p)}(n)\hat{x}(n-i). \tag{3.38}$$

The lattice predictor must be modified accordingly, as shown in Figure 3.7. As in the transversal case, the synthesis filter here is simply the structure whose input and output are $\widehat{e}(n)$ and $\widehat{x}(n)$, respectively.

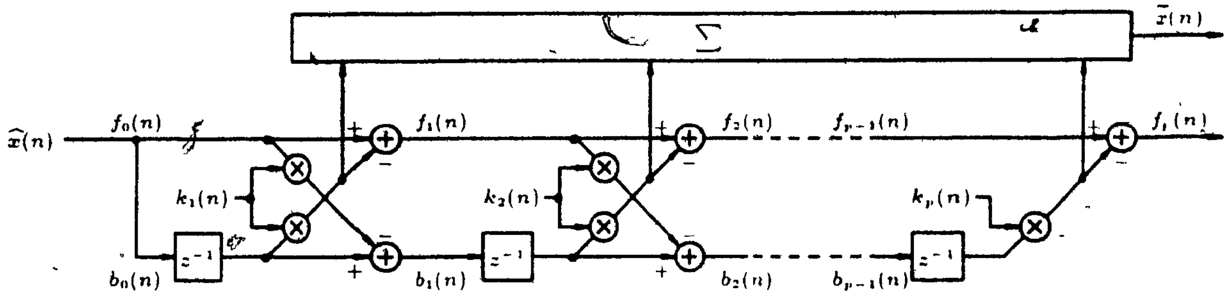


Fig. 3.7 Lattice predictor of order p

3.3.1 Properties of the Lattice Filter

The lattice filter satisfies a number of interesting properties, including *(i)* a form of independence between successive stages due to correlation properties of the residual signals, *(ii)* a simple verification procedure for the stability of the synthesis filter, and *(iii)* modularity of the structure.

Correlation Properties. The first group of properties *(i)* are based on the principle of orthogonality for whitening filters. A large number of correlation properties between the residual signals and the input have been summarized by Haykin in [23]. The most important of these are

$$E[b_i(n)\hat{x}(n-l)] = 0, \qquad 0 \le l \le i-1, \tag{3.39a}$$

$$E[f_i(n)\hat{x}(n-l)] = 0, \qquad 1 \le l \le i, \tag{3.39b}$$

$$E[b_i(n)b_j(n)] = \begin{cases} P_i, & i = j \\ 0, & i \neq j \end{cases}, \tag{3.39c}$$

$$E[f_i(n)b_j(n)] = \begin{cases} k_iP_j, & i \ge j \\ 0, & i < j \end{cases}, \tag{3.39d}$$

where

$$P_i \overset{\Delta}{=} E[f_i^2(n)] = E[b_i^2(n)]. \tag{3.40}$$

The equality in (3.40) follows from (3.35a), (3.35b) and (3.36). Equations (3.39a), (3.39c) and (3.38) imply that the backward residuals $b_{i-1}(n-1)$ form an orthogonal basis for the prediction signal $\hat{x}(n)$, with optimal coefficients k_i. In the transversal filter, the prediction signal was formed by a correlated (linearly dependent) set of signals: $\{\hat{x}(n-1), \hat{x}(n-2), \ldots, \hat{x}(n-p)\}$. In fact, the backward residuals may be generated by applying the *Gram-Schmidt orthogonalization procedure* to the above set of signals [23]. This orthogonalization process allows a form of decoupling between successive lattice stages.

Synthesis Filter Stability. The stability of the synthesis filter is ensured if and only if [20]

$$|k_i| < 1, \qquad 1 \le i \le p. \tag{3.41}$$

Hence, unlike the transversal case, a simple stability verification exists for lattice predictors of any order. As in (3.9), the region is reduced to avoid resonances in the synthesis filter,

$$|k_i| \le 1 - \epsilon, \qquad 1 \le i \le p, \quad \epsilon > 0. \tag{3.42}$$

Furthermore, the residual powers at each stage can be expressed recursively as

$$P_i = P_{i-1}(1 - k_i^2) = E[\hat{x}^2(n)] \prod_{j=1}^{i}(1 - k_j^2). \tag{3.43}$$

This is an appealing result, which implies that the forward and backward residual powers always decrease at each stage when the filter is stable.

Modularity. The design of the lattice filter allows simple addition or removal of single stages (without modifying the remaining stages) to form higher or lower order filters. It is shown that this property is also valid for the adaptation algorithm. Hence, the optimal $p+1$-th order lattice filter contains the optimal p-th order filter.

3.3.2 Local MMSE Solution for Optimal Coefficients

In the transversal filter, the optimal coefficients were found by a global minimization of the residual power $E[\epsilon^2(n)]$. The decoupling property of the lattice filter makes it possible to achieve this global minimization via a sequence of local minimizations of the residual powers at each stage.

The local MMSE solution at stage i is found in the following manner. Suppose that stages 1 through $i-1$ have already been optimized. Then the optimal coefficient for stage i is the one which minimizes P_i or, equivalently,

$$J_i = E[f_i^2(n) + b_i^2(n)], \tag{3.44}$$

the sum of the forward and backward residual powers.

Substituting the order update equations (3.33a) and (3.33b) into (3.44), and differentiating with respect to k_i, we get

$$\begin{aligned} \frac{\partial J_i}{\partial k_i} &= 2k_i E[f_{i-1}^2(n) + b_{i-1}^2(n-1)] \\ &\quad - 4E[f_{i-1}(n) b_{i-1}(n-1)]. \end{aligned} \tag{3.45}$$

Setting this gradient to zero results in the optimal solution

$$k_{i,\text{opt}} = \frac{2E[f_{i-1}(n) b_{i-1}(n-1)]}{E[f_{i-1}^2(n) + b_{i-1}^2(n-1)]}. \tag{3.46}$$

The above equation satisfies the condition

$$|k_{i,\text{opt}}| \leq 1, \qquad 1 \leq i \leq p,$$

which would not be the case if only $E[f_i^2(n)]$ or $E[b_i^2(n)]$ were minimized instead of the sum [23][21]. Thus, except for the equality, the optimal solution always leads to a stable synthesis filter.

Adaptive algorithms based on (3.46) have the property that the convergence of the i-th coefficient can occur only after the lower order coefficients have converged.

This is because the inputs to stage i are effectively non-stationary when the previous coefficients are still varying. As a result, the global convergence, occurring progressively on a stage-by-stage basis, depends upon the order of the filter [20].

However, the convergence rate for each stage (after its inputs have stabilized) is approximately the same because updates based on (3.46) are normalized by the input power of that stage. In effect, suitable step-size values are automatically generated for each stage. This implies that convergence of (3.46) is highly insensitive to eigenvalue spread [24]. Conversely, in the transversal filter all updates are scaled by the same step-size parameter, $\mu(n)$, making convergence eigenvalue dependent as was described in Section 3.2.2.

On the Possibility of Mistracking. An algorithm based on (3.46) can be considered as being signal-driven since the forward and backward residuals are derived from the reconstructed signal. In fact, these residuals individually span the same signal space as $\widehat{\mathbf{X}}(n-1)$ so that the discussion in Section 3.2.4 is equally valid in the case of lattice filters.

Thus (3.46) is susceptible to mistracking, and the development of a residual-driven lattice algorithm would be beneficial. In the next section two specific lattice algorithms are examined: one signal-driven and one residual-driven.

3.3.3 Lattice Stochastic Gradient Algorithm (GAL)

3.3.3.1 Signal-Driven Adaptation

The lattice SG algorithm (also known as the Gradient Adaptive Lattice algorithm) is derived by approximating the ensemble averages in (3.46) by fading memory time averages,

$$k_i(n+1) = \frac{2\sum_{j=1}^{n} \gamma^{n-j} f_{i-1}(j) b_{i-1}(j-1)}{\sum_{j=1}^{n} \gamma^{n-j}\left\{ f_{i-1}^2(j) + b_{i-1}^2(j-1)\right\}}, \quad 0 < \gamma < 1, \tag{3.47}$$

where the fading factor γ is required in case of non-stationary inputs.

The reflection coefficient estimate in (3.47) can also be defined by the following recursive equations:

$$C_i(n) = \gamma C_i(n-1) + 2 f_{i-1}(n) b_{i-1}(n-1), \tag{3.48a}$$

$$D_i(n) = \gamma D_i(n-1) + f_{i-1}^2(n) + b_{i-1}^2(n-1), \tag{3.48b}$$

$$k_i(n+1) = \frac{C_i(n)}{D_i(n)}. \tag{3.48c}$$

Thus, the numerator and denominator are approximated separately and the ratio of these two terms becomes the new estimate of the optimal reflection coefficient. Note that this estimate is biased in the asymptotic mean [25], for as $n \to \infty$,

$$E[k_i] = E\left[\frac{C_i}{D_i}\right] \neq \frac{E[C_i]}{E[D_i]} \simeq k_{i,\text{opt}}.$$

Equation (3.48c) is an **indirect** coefficient update; *i.e.*, it is not of the form $k_i(n+1) = k_i(n) + \Delta_{k_i}(n)$. However, by manipulating (3.48) and (3.33) the coefficient updates can be equivalently expressed in the **direct** form as

$$k_i(n+1) = k_i(n) + \frac{f_{i-1}(n) b_i(n) + b_{i-1}(n-1) f_i(n)}{D_i(n)}, \tag{3.49}$$

where the step-size μ is implicitly generated in the driving term, and the fading factor γ has the same role as λ in the transversal algorithm (3.26). This is effectively a stochastic gradient update equation, similar to the transversal case.

To complete the algorithm we must add coefficient leakage for the same reasons as stated earlier for the transversal filter. Thus, the signal-driven SG algorithm is defined, in the direct form, as

$$k_i(n+1) = (1-\delta) k_i(n) + \frac{f_{i-1}(n) b_i(n) + b_{i-1}(n-1) f_i(n)}{\sum_{j=1}^{n} \gamma^{n-j}\left\{ f_{i-1}^2(j) + b_{i-1}^2(j-1)\right\}} \tag{3.50a}$$

or, in the indirect form, as

$$k_i(n+1) = (1 - \delta)\frac{C_i(n)}{D_i(n)}. \tag{3.50b}$$

Although (3.48c) and (3.49) are equivalent, it is important to note that (3.50a) and (3.50b) are not, since in the direct form the leakage is applied only to the previous coefficients, whereas in the indirect form it is applied to the driving term as well. However, the difference between the two is insignificant if δ is close to zero.

3.3.3.2 Residual-Driven Adaptation

One way to derive a residual-driven algorithm for the lattice filter would be to follow the procedure given in Section 3.2.5.2. However, this would prove to be very inefficient since *(i)* the reflection coefficients must first be mapped into transversal coefficients, *(ii)* the approximation in (3.29) must be performed to find the residual-driven updates in (3.30), and *(iii)* the result must be mapped back to reflection coefficient updates:

$$k_i \xmapsto{\mathcal{M}^{-1}} a_i \xmapsto{(3.29)} \Delta_{a_i} \xmapsto{\mathcal{M}} \Delta_{k_i}.$$

Even if these calculations were feasible on a sample-by-sample basis, there are other disadvantages to consider. First, the result would not be very accurate due to the combined effects of the approximation in (3.29) and the non-linearity of the coefficient transformation $\mathcal{M}$. Second, the resulting adaptation algorithm, being a transformed version of the transversal algorithm, completely ignores the correlation properties of the lattice filter discussed earlier. Third, the method would get increasingly complex with higher order filters (as was the case with the transversal algorithm).

Instead, the derivation of a residual-driven algorithm for the lattice filter is based on the following heuristic approach: In ADPCM, the residual signal is far from white and is, in fact, correlated with the input signal. This is because the prediction process is not ideal, particularly with low-order predictors. Intuitively, it would appear that a sub-optimal set of coefficients can be selected by maximally de-correlating the quantized residual $\hat{e}(n)$ instead of the reconstructed signal $\hat{x}(n)$. These coefficients can then be used to form the prediction, as in Figure 3.7.

For example, Figure 3.8 illustrates the reconstructed and quantized residual signals for a segment of male speech of the vowel /a/ in the word "dark". The signals are from a simulated ADPCM coder utilizing a second order lattice predictor with residual-driven adaptation (referred to as LR). Comparing the waveforms in (a) and (b), or better yet, the magnitude spectra in (c) and (d), it is obvious that the quantized residual is quite correlated with the reconstructed signal (and hence the input). The same trend is observed with other speech sounds and narrowband signals.

Implementation of Residual-Driven Algorithm. An implementation based on this approach requires two lattice filters: one for de-correlating $\hat{e}(n)$ and generating $k_i(n)$, and the other for the prediction. This is depicted in Figure 3.9. Note that the $k_i(n)$ generated by the top filter are used in the bottom filter.

Following the results thus far for the lattice filter and signal-driven algorithm, a residual-driven algorithm for the above scheme can be defined by the following equations:

$$\bar{f}_i(n) = \bar{f}_{i-1}(n) - k_i(n)\bar{b}_{i-1}(n-1), \tag{3.51a}$$

$$\bar{b}_i(n) = \bar{b}_{i-1}(n-1) - k_i(n)\bar{f}_{i-1}(n), \tag{3.51b}$$

$$\bar{f}_0(n) = \bar{b}_0(n) = \hat{e}(n), \tag{3.51c}$$

$$\overline{C}_i(n) = \gamma\overline{C}_i(n-1) + 2\bar{f}_{i-1}(n)\bar{b}_{i-1}(n-1), \tag{3.51d}$$

$$\overline{D}_i(n) = \gamma\overline{D}_i(n-1) + \bar{f}^2_{i-1}(n) + \bar{b}^2_{i-1}(n-1), \tag{3.51e}$$

$$k_i(n+1) = (1-\delta)\frac{\overline{C}_i(n)}{\overline{D}_i(n)}. \tag{3.51f}$$

Performance Limitations. Using such a method entails a sacrifice in coefficient

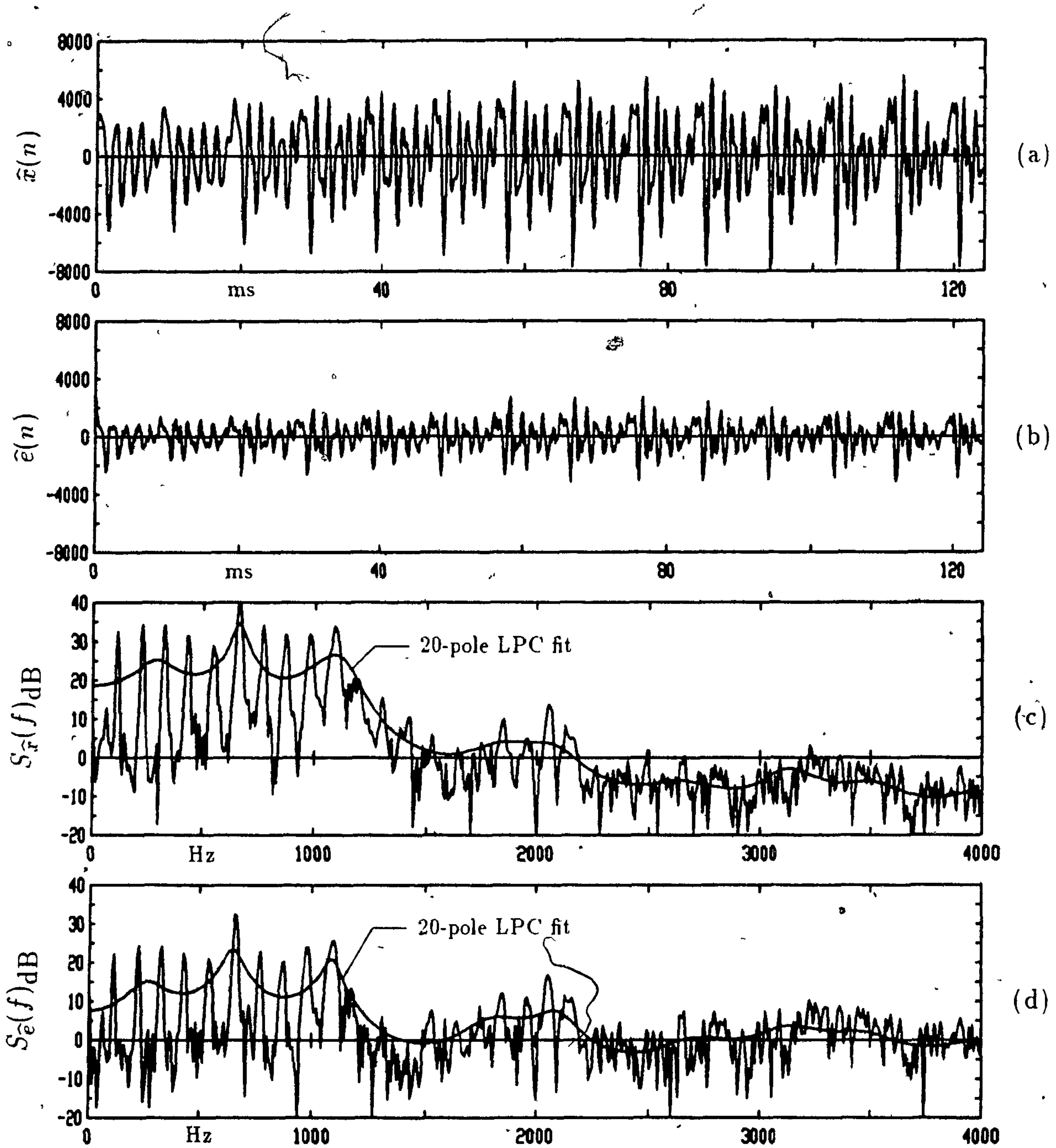


Fig. 3.8 Correlation between $\widehat{x}(n)$ and $\widehat{e}(n)$ for speech vowel /a/ in the word "dark" (LR predictor): (a) reconstructed signal; (b) quantized residual; (c) short-term spectrum of (a); (d) short-term spectrum of (b).

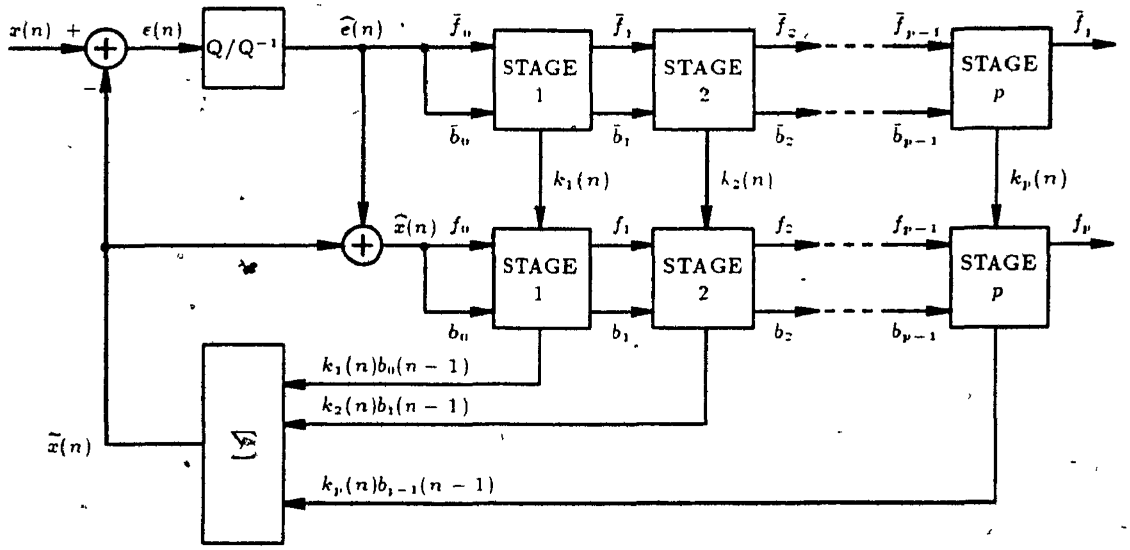


Fig. 3.9 Implementation of residual-driven system (using two p-th order lattice filters)

tracking performance (compared to the signal-driven algorithm), particularly with wide-band input signals. Figure 3.10 compares the tracking behaviour of the residual-driven predictor (LR) with that of the signal-driven predictor (LS) for the input speech segment /a/.

The reason for this loss in performance is as follows: When prediction is poor and $\widehat{e}(n)$ is strongly correlated with $\widehat{x}(n)$, the coefficients will converge towards the optimal point. As the prediction accuracy increases with time, the correlation between $\widehat{e}(n)$ and $\widehat{x}(n)$ weakens (whitening effect); *i.e.*, the informational content of $\widehat{e}(n)$ will decrease. Thus, the generated coefficients will no longer track the input. An equilibrium is achieved where the two forces balance, and the mean of the coefficient trajectory will differ from the optimal point by a small amount. This offset is smaller for narrowband signals since there is a stronger correlation between $\widehat{x}(n)$ and $\widehat{e}(n)$ than is the case with wideband signals, even at high prediction gains.

With the signal-driven lattice, on the other hand, the informational content of

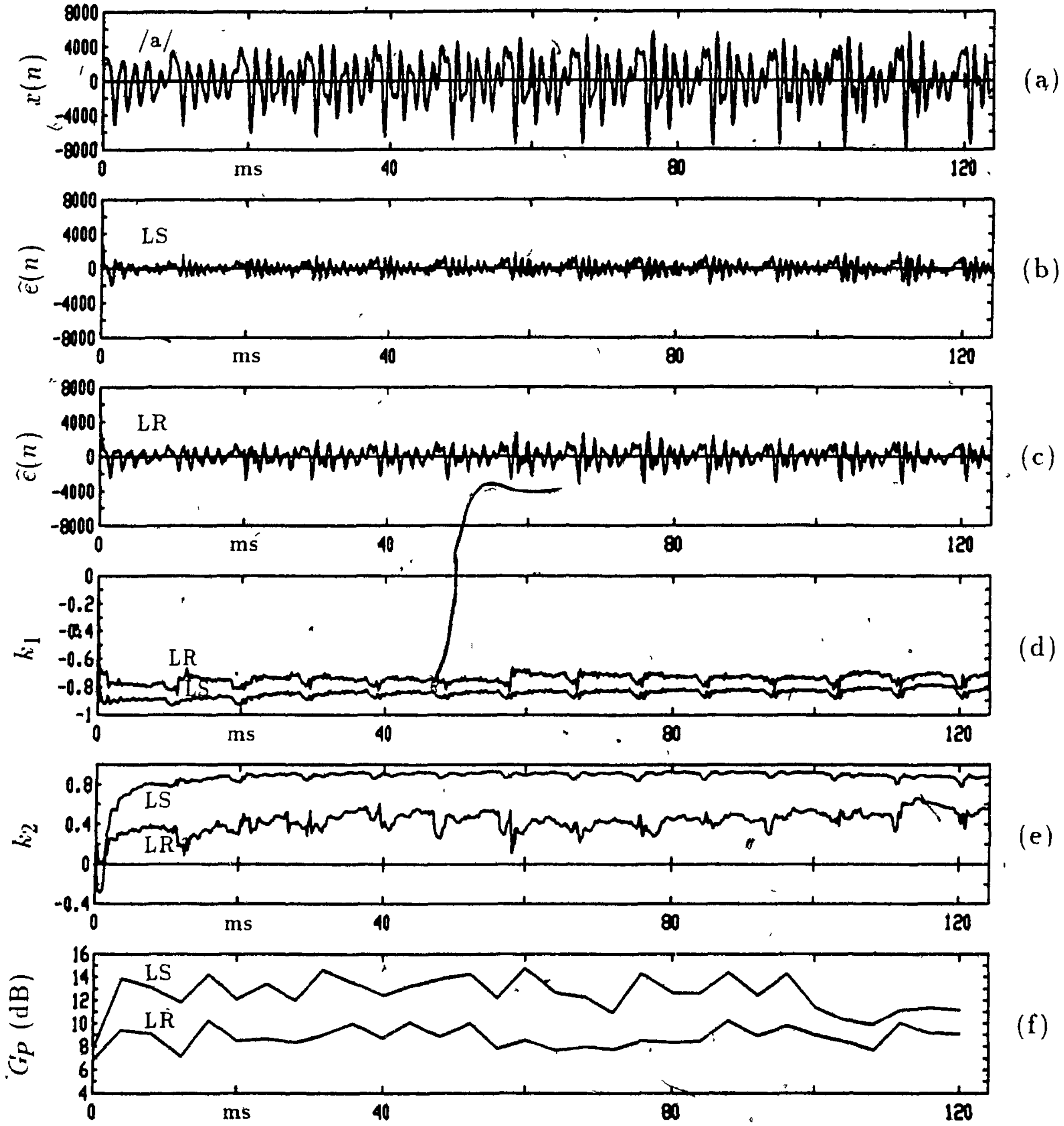


Fig. 3.10 Encoder tracking of LS and LR predictors for speech vowel /a/ in the word "dark": (a) input signal /a/; (b) LS quantized residual; (c) LR quantized residual; (d) k_1 trajectory; (e) k_2 trajectory; (f) prediction gain Gp in dB.

$\hat{x}(n)$ does not decrease as the prediction improves. The only effect, as $\hat{e}(n)$ becomes white, is a small fluctuation in the coefficient trajectory about the optimal point; the offset is negligible in the mean.

Robustness of the Algorithm. Using proof similar to that in Section 3.2.5.2, it can be shown that the encoder and decoder algorithms converge after an initial offset in the coefficients. Let $p_i(n)$ and $q_i(n)$, $1 \le i \le p$, be two distinct solutions to (3.51f), each with different i.c.'s but with identical input $\hat{e}(n)$. Rewriting (3.51f) as

$$\frac{k_i(n+1)}{(1-\delta)} = \left(1 - \frac{\bar{f}^2_{i-1}(n) + \bar{b}^2_{i-1}(n-1)}{\sum_{j=1}^{n} \gamma^{n-j}\left\{\bar{f}^2_{i-1}(j) + \bar{b}^2_{i-1}(j-1)\right\}}\right) k_i(n) + \frac{2\bar{f}_{i-1}(n)\bar{b}_{i-1}(n-1)}{\sum_{j=1}^{n} \gamma^{n-j}\left\{\bar{f}^2_{i-1}(j) + \bar{b}^2_{i-1}(j-1)\right\}}, \qquad (3.52)$$

we get the difference between the first coefficients of the two solutions,

$$p_1(n+1) - q_1(n+1) = (1-\delta)\big(1 - \phi_0(n)\big)\big(p_1(n) - q_1(n)\big),$$

where $\phi_0(n)$ is a function of the current and past values of $\hat{e}(n)$ and is bounded by $(0,1)$. This difference equation approaches zero exponentially with maximum time constant $1/\delta$. Once the first coefficients converge, the residuals at the output of the first lattice stage will also converge for the two systems. To complete the proof we simply show that

$$\big|p_i(n) - q_i(n)\big| \rightarrow 0 \quad \text{as} \quad n \rightarrow \infty,$$

given that the previous stages have converged,

$$p_j(n) = q_j(n) \quad \text{for} \quad j = i-1, i-2, \ldots, 1.$$

This follows immediately from (3.52) where the second term is common to both solutions and thus the difference equation can be written as

$$p_i(n+1) - q_i(n+1) = (1-\delta)\big(1 - \phi_{i-1}(n)\big)\big(p_i(n) - q_i(n)\big),$$

where $\phi_{i-1}(n)$ is again a function of the current and past values of $\widehat{e}(n)$ bounded by $(0,1)$. This equation also approches zero exponentially with maximum time constant $1/\delta$. Hence, the re-adjustment of the decoder onto the encoder occurs stage-by-stage, with a global maximum time constant which depends on p, γ, and $(1-\delta)$.

Chapter 4 Experimental Results

This chapter investigates the performance of a 32 kb/s ADPCM encoder/decoder with speech and signalling tone inputs. Voiceband data inputs were not applied as it was felt that the results obtained with the simpler signalling tone inputs typify the behaviour of the system for all types of narrowband signals. The adaptive predictors, described in the previous chapter, are first optimized and then compared in terms of tracking ability, convergence, and prediction gain in the presence and absence of channel errors.

For convenience, the following notation will be used in order to precisely refer to each of the five adaptive predictors in subsequent experiments:

1) $$\mathrm{TS}(p;\delta;\mu;\lambda) \longrightarrow \left\{ \begin{array}{lll} \text{transversal predictor} & : & (3.4) \\ \text{signal-driven algorithm} & : & (3.26) \\ \text{stability constraint} & : & (3.9), \quad \epsilon = .05 \end{array} \right\},$$

2) $$\mathrm{TR}(p;\delta;\mu;\lambda) \longrightarrow \left\{ \begin{array}{lll} \text{transversal predictor} & : & (3.4) \\ \text{residual-driven algorithm} & : & (3.30), \quad p = 2 \\ \text{stability constraint} & : & (3.9), \quad \epsilon = .05 \end{array} \right\},$$

3) $$\mathrm{TRsgn}(p;\delta;\mu) \longrightarrow \left\{ \begin{array}{lll} \text{transversal predictor} & : & (3.4) \\ \text{residual-driven algorithm} & : & (3.32), \quad p = 2 \\ \text{stability constraint} & : & (3.9), \quad \epsilon = .05 \end{array} \right\},$$

4) $$\mathrm{LS}(p;\delta;\gamma) \longrightarrow \left\{ \begin{array}{lll} \text{lattice predictor} & : & (3.33), (3.34), (3.38) \\ \text{signal-driven algorithm} & : & (3.48a), (3.48b), (3.50b) \\ \text{stability constraint} & : & (3.42), \quad \epsilon = .05 \end{array} \right\},$$

5) $$\text{LR}(p;\delta;\gamma) \longrightarrow \left\{\begin{array}{lll} \text{lattice predictor} & : & (3.33),(3.34),(3.38) \\ \text{residual-driven algorithm} & : & (3.51) \\ \text{stability constraint} & : & (3.42), \quad \epsilon = .05 \end{array}\right\},$$

where the parameters above are

p → prediction order,

δ → coefficient leakage factor,

μ → adaptation step-size (transversal),

λ → normalization memory constant (transversal),

γ → adaptation fading factor (lattice),

ϵ → stability constant.

Note that the TR and TRsgn predictors in (2) and (3) are restricted to a second order implementation due to the derivation of the residual-driven algorithm (3.30). In any case, most of the experiments in this chapter are performed with second order predictors. Adaptation equations for higher order TR and TRsgn predictors have been derived, following the same steps in Section 3.2.5.2 but with $p > 2$. However, the resulting updates are nonlinear and too cumbersome to be illustrated in this paper, even after simplification. It is my belief that the disadvantages of higher order residual algorithms for the transversal predictor far outway any gains in performance.

Quantization is performed by a 4-bit or 16-level Gaussian quantizer utilizing robust step-size adaptation (see Section 2.5). The Gaussian mapping seems to be a good compromise for quantization of both speech and data residual signals. The same adaptive quantizer is used in all five ADPCM systems. and is defined by:

$$Q/Q^{-1} \longrightarrow \left\{\begin{array}{lll} \text{mapping} & : & \text{4-bit Gaussian} \\ & : & e_k \text{ and } \hat{e}_k \text{ as in Table 2.1, with } \sigma_e = 1500, \\ \text{codeword} & : & k \text{ in sign-magnitude code (SMC)} \\ \text{algorithm} & : & (2.16) \text{ with } \beta = 63/64 \\ & : & \text{step-size multipliers as in Table 2.2} \\ & : & (2.13) \text{ dynamic range} = 65 \text{ dB} \end{array}\right\}.$$

4.1 Computer Simulation Procedure

The work to be described in this chapter was performed at the INRS-Télécommunications/Bell Northern Research audio laboratory in Montréal. The ADPCM system in Figure 2.4 was simulated on a DEC VAX 11/8600 computer using the predictor and quantizer definitions described above. The simulation program was written in FORTRAN using single precision real arithmetic.[1] In addition, a number of audio processing, playback, and display utilities were used for examining the results.

A vast library of phonetically balanced speech sequences is conveniently maintained for audio research. All the test sequences used in this work were originally processed through an anti-aliasing filter, sampled at 8 kHz, and stored in the library as *audio files*.[2] Single-tone and dual-tone sequences were generated by computer at 8 kHz and also stored as audio files. The simulation program reads segments of an input audio file, performs the encoding and decoding operations sample by sample, and stores the reconstructed segments in an output audio file.

The program also generates other information such as quantized residuals, predictor coefficients and performance measures (SNR, G_P) that can be displayed (along with the input and output sequences) as functions of time.

4.2 Selection of Optimal Predictor Parameters

A meaningful and fair comparison of the five predictors must be preceded by an optimization of the speech and dual-tone performance of each predictor with respect to its adaptation parameters. Thus, the main objective of this section is to establish

[1] We are concerned only with algorithm performance, not with a real-time hardware simulation in fixed point arithmetic.

[2] An audio file contains integers in the range $[-32768, 32767]$ representing the sample amplitudes in a sequence. The full range is never used, allowing samples to be processed without arithmetic overflow.

(for each predictor) the optimal or best compromise parameter values for both types of input signal.

For simplicity, the optimization is limited to second order predictors. The predictor performance measure is segmental prediction gain ($G_P SEG$; see Section 2.3, Equation (2.11)) using 16 ms or 128 sample segments. Segmental *SNR* (*SNRSEG*) is also valid, though it does not isolate the predictor performance from the combined predictor-quantizer performance. However, this is not important if *(i)* the test inputs are chosen so that the quantizer does not operate near the limits of its dynamic range and *(ii)* there are no channel errors. Under these conditions, the quantizer performance is relatively constant, with $SNR_Q SEG \approx 18$ dB, regardless of the type of input. Thus, the *SNRSEG* value is roughly equal to $G_P SEG + 18$ dB, and either value may be used to evaluate predictor performance. In the presence of errors this relationship no longer holds, so $SNRSEG(p_e)$ is used to evaluate the overall encoder/decoder performance (see Section 2.3).

Eight speech inputs, defined in Appendix A, are used for speech optimization. The results from coding each input are averaged to give an overall speech performance measure. An input signal composed of two equi-amplitude sinusoids at 697 and 1477 Hz is used for optimization on narrowband signals,

$$x(n) = A \sin \frac{2\pi f_1}{f_s} n + A \sin \frac{2\pi f_2}{f_s} n, \qquad \text{(DTMF-3)}$$

with $f_s = 8$ kHz, $f_1 = 697$ Hz, and $f_2 = 1477$ Hz. This signal, referred to as DTMF-3, corresponds to the tones generated by pushbutton #3 on commercial telephone sets using DTMF (dual-tone multi-frequency) signalling.

The most important parameters for predictor adaptation are: δ (coefficient leakage factor), μ (adaptation step-size for transversal predictors), and γ (adaptation fading factor for lattice predictors). The stability constant, ϵ, and the normalization memory constant for transversal predictors, λ, play a secondary rôle in predictor performance, if adequate values are chosen. In the following experiments they are fixed to the values 0.05 and 0.9, respectively. Furthermore, and without loss of generality, the prediction order is constrained to $p = 2$; the optimal adaptation parameters (δ, and μ or γ) should have little dependence on prediction order.

An optimization procedure for each predictor is outlined below.

1) Encoder performance using speech and DTMF-3 inputs.

   a) find the average *GpSEG* as a function of δ, μ for the transversal predictors and δ, γ for the lattice predictors using the speech inputs of Table A.1.

   b) do the same with the DTMF-3 input signal.

   c) using the results of (1a) and (1b) find, as δ varies, some values of μ in the transversal predictors and γ in the lattice predictors which provide high *GpSEG* for both types of input (these values should be on the *locus* of maximum *GpSEG*, a function of δ).

2) Encoder/decoder speech performance with transmission errors.

   a) with speech inputs in the presence of transmission bit errors of different rates p_e, find the average $SNRSEG(p_e)$ at the decoder as a function of δ and p_e; use the values found in (1c) for μ and γ.

   b) select the δ in (2a) which yields the best overall speech performance in the presence and absence of errors; this value along with the corresponding μ or γ in (1c) constitute the best overall choice of predictor parameters.

Of course, an evaluation of DTMF-3 performance with errors is not possible due to mistracking in the signal-driven predictors. Tracking performance will be examined in Section 4.4, using the optimal parameters found here.

4.2.1 Encoder Optimization

The experiments in this section correspond to parts (1a), (1b), and (1c) in the optimization procedure. The predictor parameters were varied exponentially in steps of .5, taking on the following values:

$$\delta = 2^i, \qquad i = -12, -11.5, \ldots, -4,$$

$$\mu = 2^i, \qquad i = -8, -7.5, \ldots, 0,$$

$$\gamma = 1 - 2^i, \qquad i = -10, -9.5, \ldots, -2.$$

This results in 17 × 17 points in each of the parameter spaces (δ, μ) and (δ, γ). The range of values chosen includes typical values found in the literature for SG adaptation algorithms.

There are several reasons for selecting values that are negative integer powers of two (the intermediate steps of .5 are included only to improve the resolution of the parameter space). First, an exponential scale is required in order to test a wide range of parameter values. Second, constants of the form 2^i can be represented by shift operations in digital hardware, thus leading to very efficient implementations.

Figures B.1 through B.5, in Appendix B, illustrate the speech and DTMF-3 performance of the predictors, using contour and surface plots of $GpSEG$ as a function of the parameters. The input signals were ADPCM coded at each point in the parameter space of the five predictors and the $GpSEG$ values were measured. The dotted lines in the contour plots represent smoothed loci of maximum $GpSEG$ as δ varies from 2^{-12} to 2^{-4}. Thus, given any value of δ, the corresponding value for μ or γ on the locus will yield maximum prediction gain.

Transversal Predictors. The average speech performance (over eight inputs) of the TS, TR and TRsgn predictors is illustrated in Figs. B.1(a), B.2(a) and B.3(a), respectively. The results indicate that the best speech performance, say $GpSEG \geq 9$ dB, for all three predictors is limited to a triangular operating region in the parameter space (δ, μ). In all cases, there is a gradual decrease in $GpSEG$ which is almost independent of δ as $\mu \rightarrow 1$, and a rapid drop which is maximal in the direction $\log_2 \delta = -\log_2 \mu$ as $\mu \rightarrow 0$. Also, in the absence of errors $GpSEG$ always increases as $\delta \rightarrow 0$, as expected.

Compared to the TS predictor, the TR and TRsgn predictors have a narrower operating region with steeper drops in *GpSEG* on all sides (particularly as μ — 0). In the operating region, *GpSEG* values for the TR predictor are approximately 1 dB higher than those for TS or TRsgn (compare the contour at 11.2 dB in Fig. B.2(a) with the contours at 10 dB in Figs. B.1(a) and B.3(a)). In all three cases, the loci of maximum *GpSEG* follow similarly tilted trajectories suggesting a dependence on **both** parameters.

DTMF-3 performance for the transversal predictors is shown in Figs. B.1(b), B.2(b) and B.3(b). The TS predictor exhibits the same characteristics as with speech inputs but with lower *GpSEG* values throughout the parameter space; in fact, all five predictors achieve consistently higher *GpSEG* values with speech than with DTMF-3. The locus of maximum *GpSEG* for DTMF-3 appears slightly to the right of the one for speech, when measured along the $\log_2 \mu$ axis. Hence, the optimal speech and DTMF-3 parameters for the TS predictor are nearly identical.

The residual predictors, on the other hand, behave differently with DTMF-3 inputs than with speech inputs. Although similar maximum values are obtained as in the TS predictor, the shape and location of the DTMF-3 operating regions have changed with respect to the speech operating regions in Figs. B.2(a) and B.3(a). The DTMF-3 regions are narrow rectangular ridges approximately centered along the locii

$$\log_2 \delta = \log_2 \mu - 1, \qquad -8 \leq \log_2 \mu \leq -3$$

for TR, and

$$\log_2 \delta = \log_2 \mu - 3, \qquad -8 \leq \log_2 \mu \leq -2$$

for TRsgn. This is precisely where speech performance begins to drop. Thus, finding suitable parameters in the TR and TRsgn predictors for both speech and DTMF-3 will involve a compromise.

Lattice Predictors. Speech and DTMF-3 performance for the lattice predictors is depicted in Figs. B.4 and B.5. As expected, the operating region of lattice filters is relatively insensitive to different inputs and to variations in the parameters δ, γ. Both lattice predictors exhibit rectangular operating regions with very gradual drops in $GpSEG$ in all directions. This insensitivity to parameter values is further exemplified by the fact that the loci are virtually parallel to the $\log_2 \delta$ axis. The implication here is that the optimal γ value is independent of δ; an advantage which allows for fine tuning of lattice parameters one at a time.

The main difference between the LS and LR predictors lies in their speech performance shown in Figs. B.4(a) and B.5(a). Although both sets of contours follow the same pattern, the LS predictor attains an average of 3.5 dB more of $GpSEG$ throughout the parameter space (this difference is quite significant). With the DTMF-3 input the two predictors perform almost identically (see Figs. B.4(b) and B.5(b)) except for slightly higher $GpSEG$ values for the LS predictor. One observation is that the speech performance drops as $\gamma \rightarrow 1$, whereas the DTMF-3 performance remains flat. This discrepancy is due to the long-term non-stationarity of speech signals (requiring a certain amount of adaptation fading) as opposed to the stationarity of dual-tone signals (which can be tracked more optimally without any fading).

Choosing a Reduced Set of Parameters. Using the previous results, a set of five points was selected from the parameter space of each predictor. Each set of points, given in Table 4.1, was obtained by overlapping the speech and DTMF-3 contours of each predictor (Figures B.1–B.5) and then choosing, for five centrally located and equi-spaced δ, values for μ or γ as near as possible to both locii.

The last two rows indicate the minimum segmental prediction gains for speech and DTMF-3 signals attained with each set of points (this minimum occurrs with the

TS		TR		TRsgn		LS		LR		
δ	μ	δ	μ	δ	μ	δ	γ	δ	γ	
2^{-6}	2^{-3}	2^{-6}	2^{-4}	2^{-6}	2^{-3}	2^{-4}	$1-2^{-5}$	2^{-6}	$1-2^{-5}$	
2^{-7}	$2^{-3.5}$	2^{-7}	2^{-5}	2^{-7}	2^{-4}	2^{-5}	$1-2^{-5}$	2^{-7}	$1-2^{-5}$	
2^{-8}	2^{-4}	2^{-8}	2^{-6}	2^{-8}	2^{-5}	2^{-6}	$1-2^{-5}$	2^{-8}	$1-2^{-5}$	
2^{-9}	2^{-4}	2^{-9}	2^{-7}	2^{-9}	2^{-6}	2^{-7}	$1-2^{-5}$	2^{-9}	$1-2^{-5}$	
2^{-10}	$2^{-4.5}$	2^{-10}	2^{-8}	2^{-10}	2^{-7}	2^{-8}	$1-2^{-5}$	2^{-10}	$1-2^{-5}$	
≥ 9 dB		≥ 10 dB		≥ 8.8 dB		≥ 11.5 dB		≥ 9.2 dB		†
≥ 5 dB		≥ 5 dB		≥ 6.2 dB		≥ 6.8 dB		≥ 6.5 dB		‡

Table 4.1 Reduced set of parameters from (δ, μ) and (δ, γ).
† $G_P SEG$ for speech, ‡ $G_P SEG$ for DTMF-3

largest leakage value δ). Since the LS predictor is more sensitive to channel errors than the other four predictors (see next Section), the selected leakage values for this predictor are larger than for the others.

4.2.2 Encoder/Decoder Optimization with Transmission Errors

This section corresponds to parts (2a) and (2b) in the optimization procedure. The speech inputs of Table A.1 were encoded using the values in Table 4.1. Each encoding was subjected to channel errors of four different rates and a zero-error reference: $p_e = (10^{-2}, 5 \cdot 10^{-3}, 10^{-3}, 5 \cdot 10^{-4}, 0)$. Bit errors within a single encoding/decoding were randomly generated according to p_e. However, for each p_e, the same random error sequence was applied to each system. This was done in order to maintain some consistency in the tests. To reduce the simulation time, only a 10,112 sample segment of each input utterance was encoded. As a result, the performance measures when $p_e = 0$ will differ slightly from those in the previous section, where each utterance was encoded in its entirety. The results are illustrated in Figures 4.1(a)–(e).

Each figure contains five curves of segmental *SNR* at the decoder as a function of

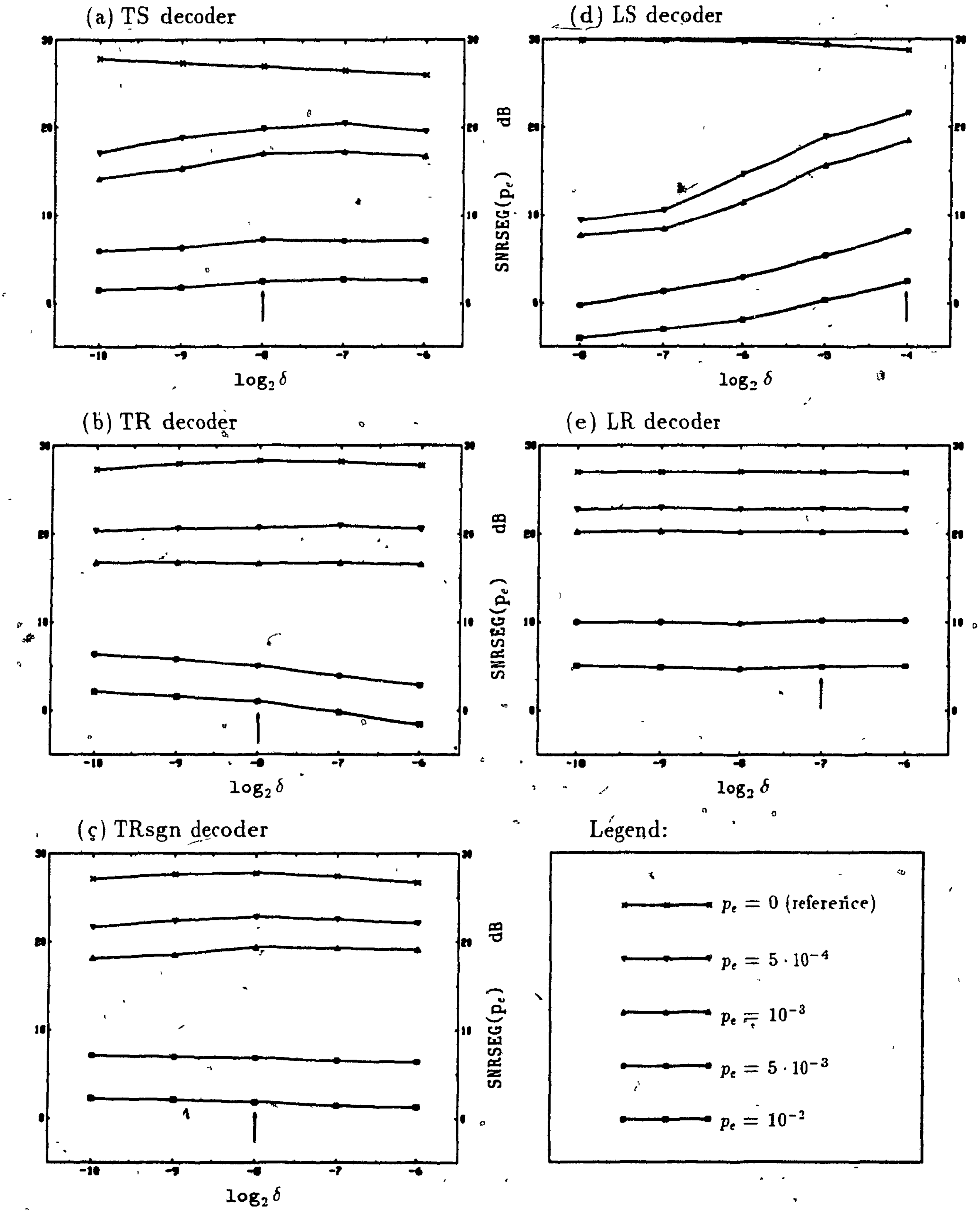


Fig. 4.1 Speech performance in $SNRSEG(p_e)$ as a function of leakage δ and bit error rate p_e. *(The arrows indicate the optimal δ value in terms of maximum SNR_W.)*

δ. Each curve represents a different bit error rate, p_e. It is clear from these figures that the effects of leakage are most pronounced in the LS predictor in (d), which exhibits a significant improvement in $SNRSEG(p_e)$ (for all $p_e > 0$) as δ is increased from 2^{-8} to 2^{-4}. There is some improvement for the TS predictor in (a) and little or no improvement for the residual-driven predictors in (b), (c), and (e). In the latter case, δ has practically no effect on performance. Generally, the most robust predictor in the presence of errors is LR followed by TRsgn, TR, TS, and LS. This result was expected, since residual-driven predictors are designed to minimize the effects of error propagation around the predictor loop.

It is desired to find the δ which maximizes the overall performance of each system, given the various bit error rates. This performance measure is a *weighted* average $SNRSEG(p_e)$ value, given by

$$\begin{aligned} SNR_W &= 0.5 \cdot SNRSEG(0) + 0.25 \cdot SNRSEG(5 \cdot 10^{-4}) + 0.15 \cdot SNRSEG(10^{-3}) \\ &\quad + 0.07 \cdot SNRSEG(5 \cdot 10^{-3}) + 0.03 \cdot SNRSEG(10^{-2}). \end{aligned}$$

Such a weighting measures the performance of the encoder/decoder under the condition: 50% of the time there are no errors ($p_e = 0$), 25% of the time $p_e = 5 \cdot 10^{-4}$, *etc.* Different weightings can be used; this one favours performance under low error rates.

The δ values that maximize SNR_W are indicated by the small arrows in Figure 4.1 and constitute, along with the appropriate μ or γ values, the best overall choice of predictor parameters. This is all summarized in Table 4.2, where the last four rows give an indication of average segmental performance for the selected values. As an aside, note that the difference between $SNRSEG$ and $GpSEG$ for speech without errors, ranges from 17.9 dB for TS to 18.5 dB for LR; this difference is precisely SNR_QSEG, as described earlier. Using the parameter values of Table 4.2, the tracking performance of the five systems can now be evaluated.

TS	TR	TRsgn	LS	LR	
δ μ	δ μ	δ μ	δ γ	δ γ	
2^{-8} 2^{-4}	2^{-8} 2^{-6}	2^{-8} 2^{-5}	2^{-4} $1-2^{-5}$	2^{-7} $1-2^{-5}$	
21.6 dB	22.2 dB	23.0 dB	23.2 dB	23.1 dB	*
27.9 dB	28.8 dB	28.2 dB	29.6 dB	27.8 dB	◇
10.0 dB	10.6 dB	9.9 dB	11.5 dB	9.3 dB	†
6.1 dB	5.8 dB	7.0 dB	6.8 dB	6.6 dB	‡

Table 4.2 Best overall choice of predictor parameters.
* SNR_W for speech. ◇ $SNRSEG$ for speech (no errors),
† $GpSEG$ for speech (no errors),
‡ $GpSEG$ for DTMF-3 (no errors)

4.3 Tracking of Speech Signals

This section explores the tracking behaviour, or dynamics, of the five predictor adaptation algorithms for different speech inputs. The first part deals with the prediction accuracy at the encoder, or the ability of the encoder to track the input. The second part deals with prediction recovery at the decoder after a burst of transmission errors, or the ability of the decoder to adjust to the encoder.

4.3.1 Encoder Tracking

In Chapter 3 it was shown that the predictor coefficients are updated in order to minimize the prediction residual power (MMSE). For the signal-driven predictors, this is achieved by descending the MSE surface in a direction opposing the stochastic gradient (SG). With short-term stationary inputs, these algorithms should converge in the mean with a small misadjustment in the coefficient trajectories due to the stochastic gradient estimates. However, the leakage factor δ acts as an extra noise term in the adaptation process which prevents the coefficients from converging in the mean: a small offset is always present. In addition, the inaccuracy of step-size normalization tends to increase both the misadjustment noise and offset in the coefficients. But even with these modifications, we would like to think of the signal-driven algorithms as being *optimal*, in the sense that they have been derived directly from an MMSE criterion.

In contrast, the residual-driven algorithms have been derived from approximations to the MMSE criterion. In the transversal predictors, the second order approximation in Equation (3.30) truncates the impulse response of the equivalent all-zero filter and thus prevents a correct minimization of the MSE. In the lattice predictor, the lack of spectral information in the residual signal (compared to the reconstructed signal) leads to incorrect decorrelation of the reconstructed signal. We have already shown the suboptimality of this scheme in Figure 3.10.

Thus, we would expect that the signal-driven algorithms should outperform the residual-driven ones, in the absence of errors. Although this is true of the lattice predictors, we found that the "suboptimal" TR and TRsgn track better than the TS. Table 4.3 compares the segmental prediction gain and SNR obtained at the encoder for eight speech inputs. Surprisingly, the TR and TRsgn predictors perform quite well on the average, despite their suboptimality. However, only 2.2 dB on the average separates the best (LS) from the worst (LR).

To what extent are these differences due to misadjustment and/or offset in the coefficient trajectories? As an example, we illustrate predictor tracking for OPEN-M, an input with relatively low prediction gain as compared to the other inputs (see Table 4.3). The results, limited to a 5,500 sample segment containing the utterance "*Open the crate...*", are shown in Figures 4.2, 4.3, and 4.4. Note that the lattice coefficients, k_i, have been converted to transversal coefficients, a_i, for comparison purposes.

Looking at Figure 4.2, it is evident that the LS, TR, and TRsgn predictors have the highest G_P, especially in voiced segments. The coefficient trajectories for these

Speech	*GpSEG / SNRSEG* (dB)				
Input	TS	TR	TRsgn	LS	LR
GLUE-M	11.6/29.9	13.1/31.4	11.6/30.2	13.0/31.3	11.1/29.9
GLUE-F	12.3/29.9	9.4/27.3	10.4/28.7	13.2/30.9	11.1/29.4
HOGS-M	9.3/27.4	9.3/27.7	8.8/27.4	10.4/28.8	8.4/27.2
HOGS-F	10.7/28.0	9.4/27.3	8.8/27.0	11.3/28.8	9.4/27.4
OPEN-M	8.5/26.1	9.9/27.9	9.7/27.6	10.8/28.6	8.2/26.5
OPEN-F	9.4/27.5	10.1/28.4	9.5/27.9	11.1/29.3	8.9/27.3
PIPE-M	9.0/26.8	10.5/28.7	9.9/28.2	11.0/29.2	8.5/27.2
PIPE-F	9.3/27.7	10.9/29.4	10.1/28.8	11.3/29.8	9.0/27.8
Average	10.0/27.9	10.6/28.8	9.9/28.2	11.5/29.6	9.3/27.8

Table 4.3 Comparison of speech performance

predictors are similar, with the main difference being in the amount of misadjustment noise which is least for the LS and most for the TRsgn.

The TS and LR predictors attain lower *Gp* throughout most of the input. It appears that their coefficient trajectories are also similar, again with the exception of misadjustment noise.

All five predictors perform equally poorly with the plosives /p/ in *"open"* and /k/ in *"crate"*. This is normal, since plosives contain very little "predictable" component.

The segment /ej/ in *crate* (between samples 6,000 and 6,500), is the only one for which the TS and LR predictors performed better than TR and TRsgn. The LS predictor tracked this segment.

Thus, for speech inputs, the predictors can be divided into three groups: (1) TR and TRsgn, (2) TS and LR, and (3) LS. It appears that group (1) attains good tracking for most sounds, while group (2) doesn't track as well. For a small percentage of segments, the opposite is true. The LS predictor in (3) seems to combine the good tracking performance of (1) and (2).

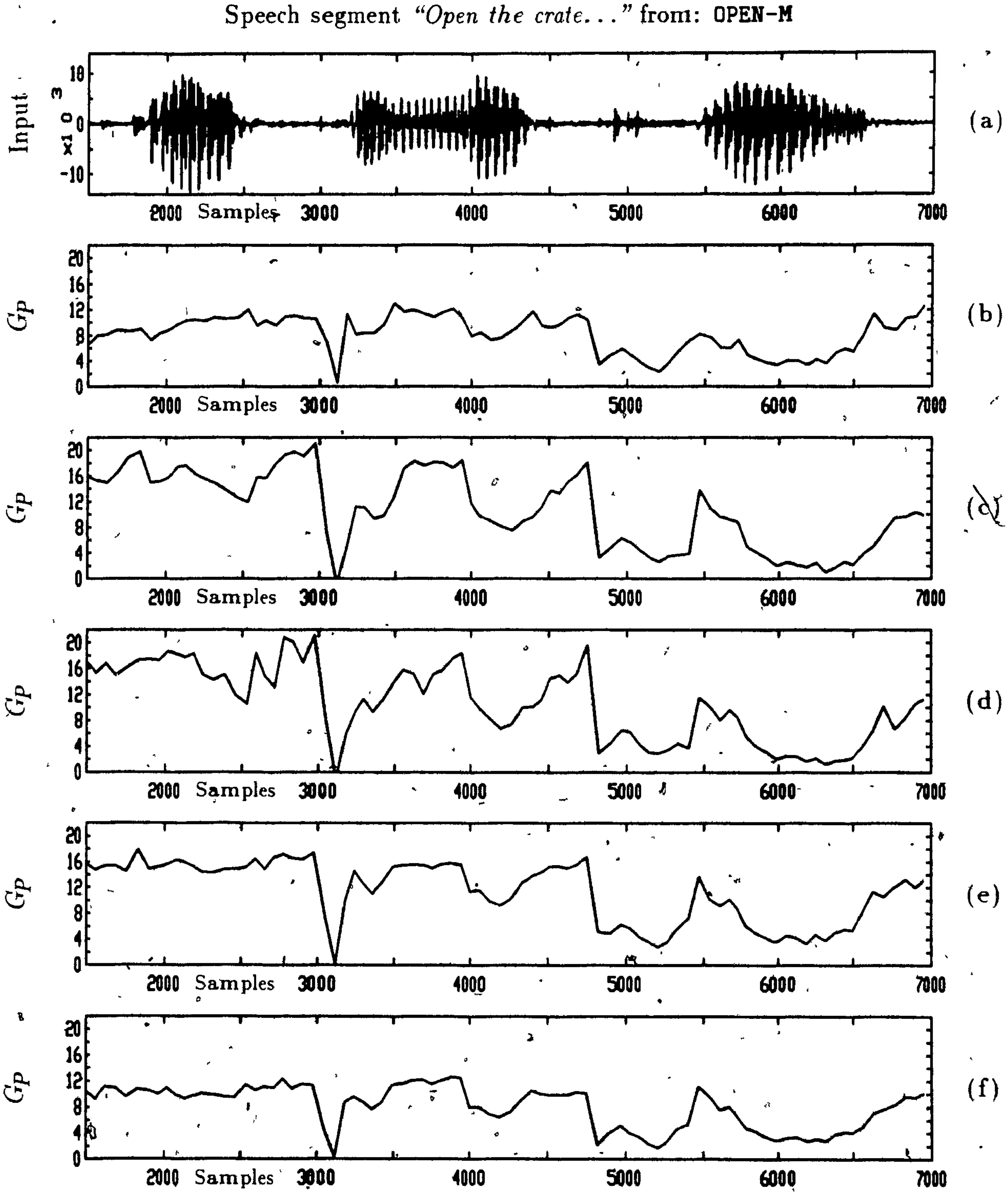


Fig. 4.2 Prediction gain at the encoder: (a) Speech Input OPEN-M, (b) TS($2; 2^{-8}; 2^{-4}; .9$), (c) TR($2; 2^{-8}; 2^{-6}; .9$); (d) TRsgn($2; 2^{-8}; 2^{-5}$), (e) LS($2; 2^{-4}; 1-2^{-5}$), (f) LR($2; 2^{-7}; 1-2^{-5}$).

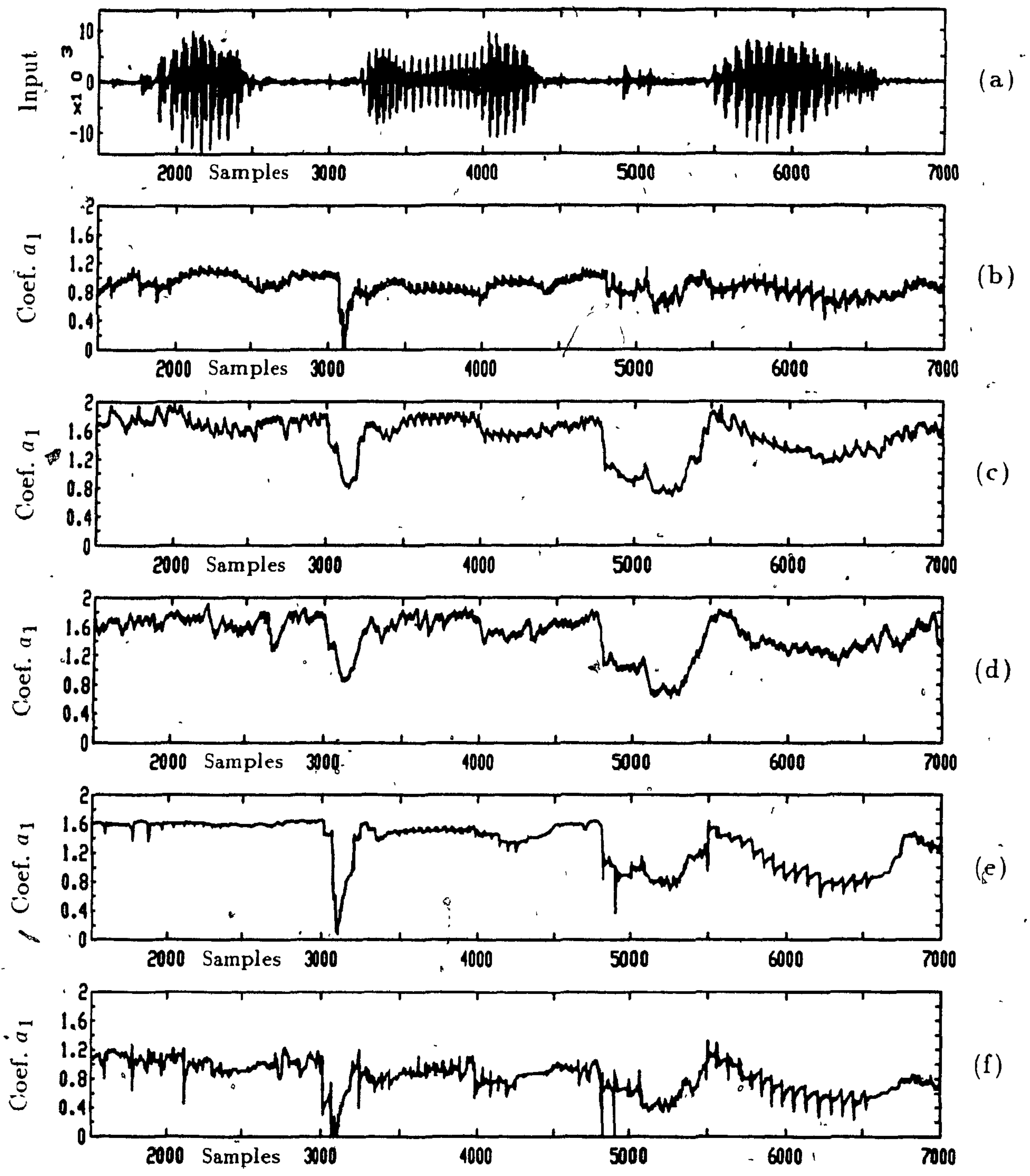

Speech segment *"Open the crate..."* from: OPEN-M

Fig. 4.3 Trajectories of coefficient a_1 at the encoder: (a) Speech Input OPEN-M, (b) TS($2; 2^{-8}; 2^{-4}; .9$), (c) TR($2; 2^{-8}; 2^{-6}; .9$), (d) TRsgn($2; 2^{-8}; 2^{-5}$), (e) LS($2; 2^{-4}; 1-2^{-5}$), (f) LR($2; 2^{-7}; 1-2^{-5}$).

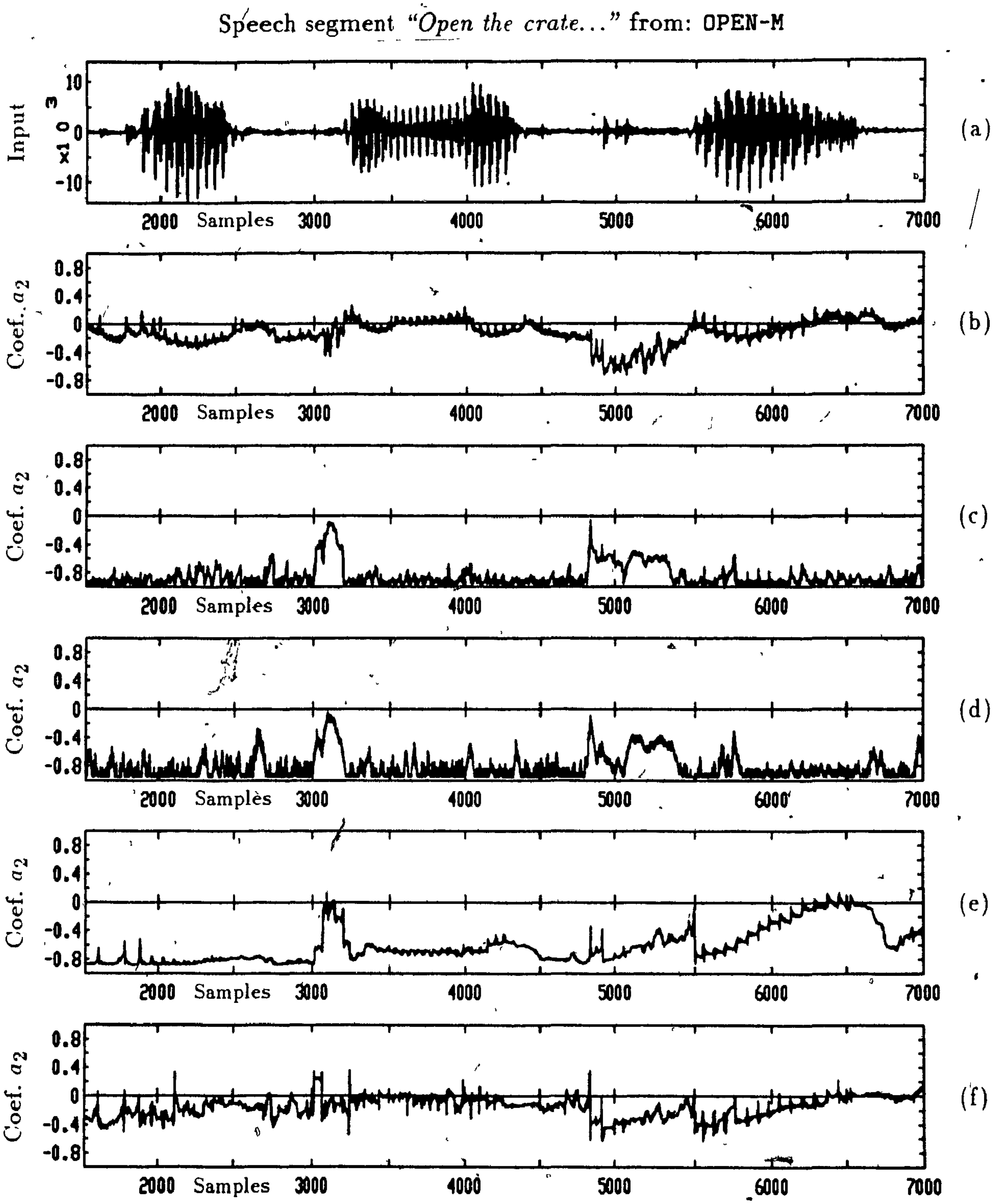
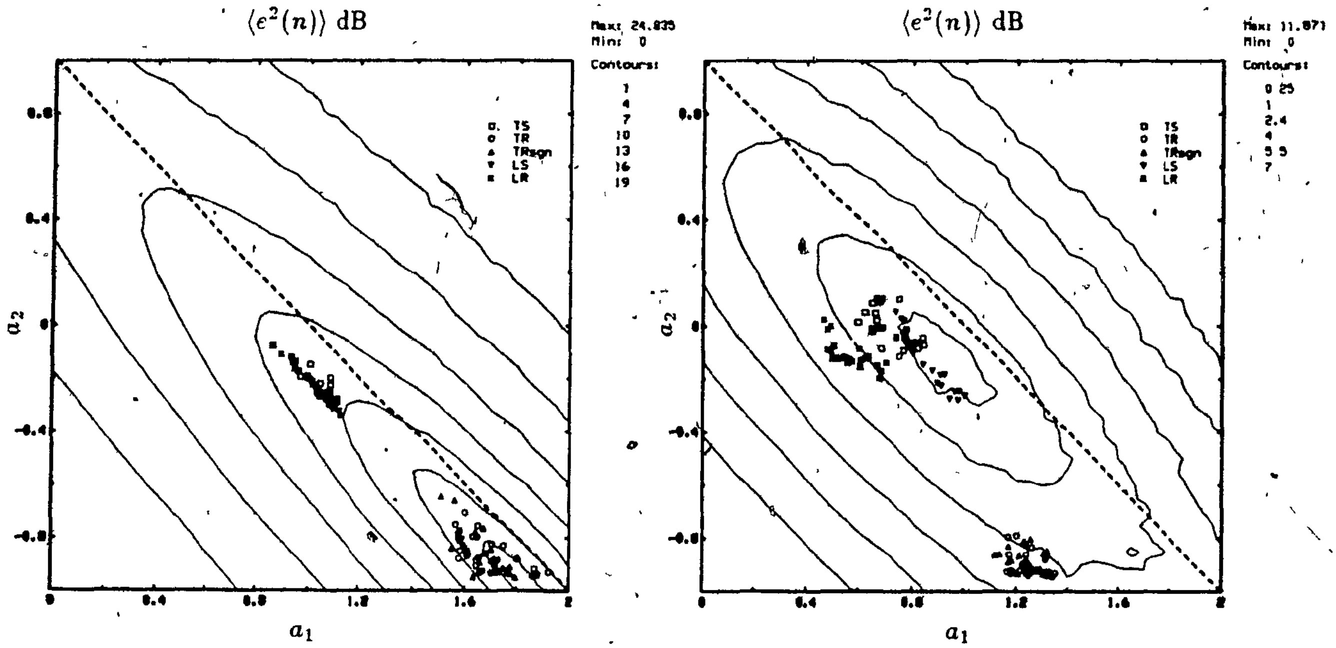


Fig. 4.4 Trajectories of coefficient a_2 at the encoder: (a) Speech Input OPEN-M, (b) TS$(2; 2^{-8}; 2^{-4}; .9)$, (c) TR$(2; 2^{-8}; 2^{-6}; .9)$, (d) TRsgn$(2; 2^{-8}; 2^{-5})$, (e) LS$(2; 2^{-4}; 1-2^{-5})$, (f) LR$(2; 2^{-7}; 1-2^{-5})$.

To understand the differences in tracking, we examine the (time-average) error surface for two voiced segments of speech (see Figure 4.5). The two segments, /o/ and /ej/, are from the preceding input OPEN-M. The error surface $\langle \epsilon^2(n) \rangle$ was calculated at a finite number of points in the coefficient space, using a fixed second order predictor at each point. Some of the contours are situated outside the stability region (dotted line) because the averaging was done over a small finite segment, yielding finite values of MSE in this region. Each point in the coefficient trajectories in Figure 4.5 represents the average of 16 consecutive samples.



(a) (b)

Fig. 4.5 Time-averaged error surface: (a) Segment /o/ → samples 1,985–2,368 of OPEN-M; (b) Segment /ej/ → samples 6,017–6,400 of OPEN-M.

In the first segment (Figure 4.5(a)), groups (2) and (3) have fully descended the error surface and are tracking near the optimum point (1.8, −.9). Group (1)'s trajectory is incapable of descending past the third contour level and is tracking the suboptimal point $(1, -.2)$, with the result of a 6.2 dB drop in G_P, on the average, compared to groups (2) and (3). Since the surface is steep, the coefficient offset in group (1) is significant.

In the second segment (Figure 4.5(b)), groups (1) and (3) have descended the error surface towards the point $(.9, -.25)$, although the LR predictor trajectory is slightly offset from this point at $(.6, -.11)$. The result of this offset is a 1.1 dB drop in G_P compared to the LS predictor. Group (2) is tracking near the point $(1.25, -.9)$, resulting in a 2.35 dB drop in G_P compared with LS. In this case the surface is flatter, and the coefficient offset in group (2) does not result in much performance loss.

This behaviour was observed with other inputs as well. It is clear from the results that, in terms of encoder tracking, the predictors of group (1) are suboptimal, those of group (2) are better, and the LS predictor performs the best.

4.3.2 Decoder Tracking

Given the tracking performance of each predictor at the encoder, we now show the decoder adjustment onto the encoder in the presence of transmission errors. Speech performance in the presence of transmission errors was determined in Section 4.2.2 for the five systems. Averaged results over the eight speech inputs (extracted from Figure 4.1) show that all five systems have nearly the same relative drops in *SNRSEG* as p_e increases (see Figure 4.6). The LR predictor, however, is uniformly better than the others for all error rates tested. The figure also indicates that the lattice predictors, as a group, are less sensitive to transmission errors than the transversal predictors. Similar results were obtained in [26] with 16 kbit/s ADPCM systems using lattice and transversal AR predictors with signal-driven adaptation.

The better performance of the lattice predictors over the transversal predictors

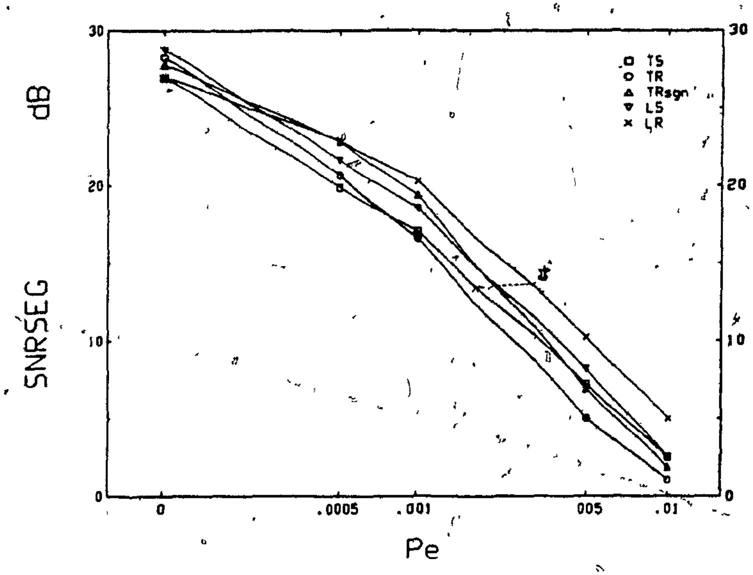


Fig. 4.6 Average speech performance with transmission errors

under noisy transmission conditions is due to the speed of recovery in the decoder coefficients after an initial offset. As an example, the input OPEN-M was encoded/decoded with a burst of 60 random bit errors between samples 3,265 and 3,520 (/E/ in "*open*"). For ease of comparison, the same burst was used with each predictor. In all five cases, the coefficients at the decoder readjusted to those at the encoder after the errors ceased (see Figure 4.7). During the burst, there is a large error at the reconstructed output which is mainly due to quantizer mistracking at the decoder. Predictor and quantizer convergence after the burst is very quick for the three predictors: TRsgn, LS, and LR, at less than 100 samples. The two remaining predictors, TR and TRsgn, required 400-500 samples to converge. Further tests confirmed that: *(i)* mistracking was limited to the duration of the voiced segments in which errors occurred, with retracking usually taking place in less time than that; *(ii)* errors during other segments of speech (fricatives, plosives, *etc*) had even a less lasting effect.

Thus, transmission errors during speech are of concern only during the segment in which they occur — there is no long lasting effect, such as permanent mistracking

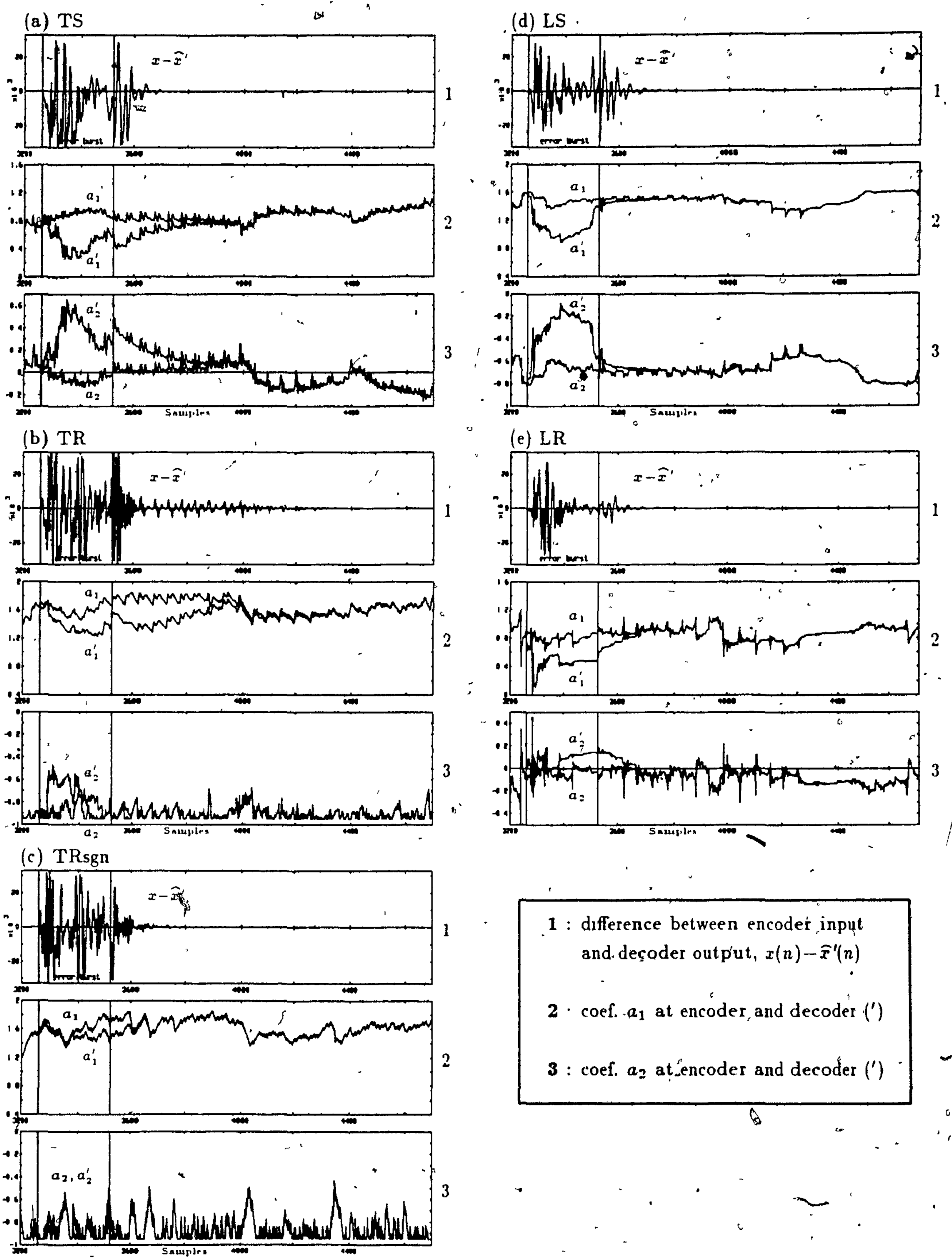


Fig. 4.7 Decoder readjustment after a burst of errors during transmission of speech

of the decoder coefficients.

4.4 Tracking of Dual-Tone Signals

In this section, the tracking performance for dual-tone input signals is investigated. As in the previous section, both encoder and decoder tracking are examined. With data signals, however, the latter property is of more interest.

4.4.1 Encoder Tracking

The energy of DTMF signals is concentrated in two narrow bands in the spectrum (see Figure 4.8). For a stationary DTMF-3 signal we can compute the optimal second order predictor, namely

$$\mathbf{A}_{\mathrm{opt}} = [1.15 \quad -0.83]^{\mathrm{T}}.$$

The a_1 component of this optimal point is half way between the optimal a_1 components of sinusoids with frequencies 697 and 1477 Hz. a_2 is inside the stability region but close to -1 (the a_2 of a pure sinusoid lies on -1). This is because second order predictors insufficiently model the poles of a DTMF signal — a fourth order predictor is required for modelling two complex-conjugate pole pairs. However, the next section will show that model insufficiency is not the reason why decoder mistracking occurs with second order TS and LS predictors.

Although some prediction gain at the encoder is desired for DTMF signals, precise coefficient estimation for the same signals is not an important objective in ADPCM. This is because the tone-pair is correctly demodulated at the receiver, despite a lower *SNR*. Table 4.4 shows, for DTMF-3, the average coefficients, G_P, and *SNR* in the steady state. Similar results were obtained with other DTMF and multi-tone signals.

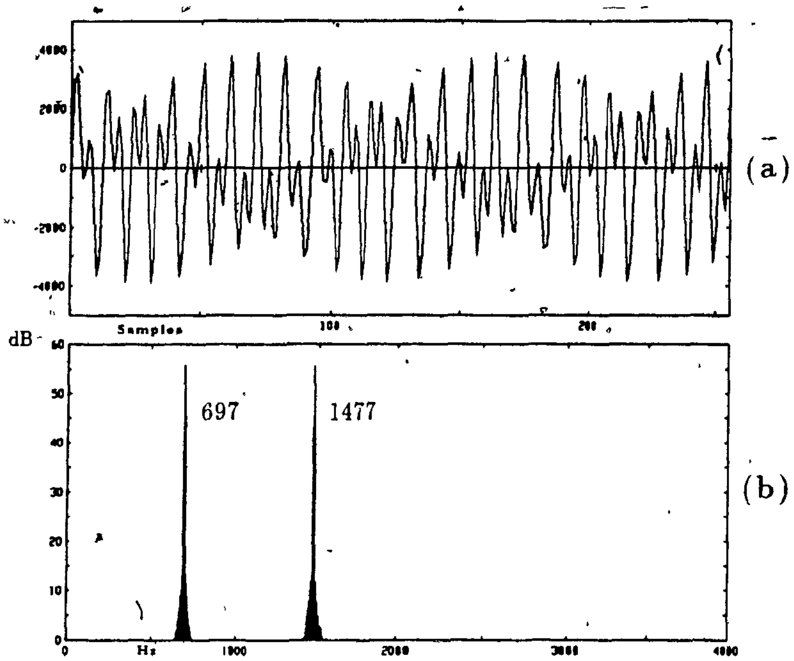


Fig. 4.8 DTMF-3: (a) time waveform; (b) magnitude spectrum

	TS	*TR*	*TRsgn*	*LS*	*LR*
$\mathbf{A}_{\mathrm{av}}$	$[0.93 \quad -0.62]^{\mathrm{T}}$	$[1.45 \quad -0.94]^{\mathrm{T}}$	$[1.23 \quad -0.93]^{\mathrm{T}}$	$[1.03 \quad -0.77]^{\mathrm{T}}$	$[0.97 \quad -0.65]^{\mathrm{T}}$
$G_{P\,\mathrm{av}}$(dB)	6.15	5.81	7.03	6.75	6.54
SNR_{av}(dB)	25.57	25.76	26.15	25.88	25.75

Table 4.4 Average coefficients and prediction gain after convergence of encoder for DTMF-3

It is interesting to note that, despite its suboptimality, the TRsgn predictor tracks the optimal point accurately. There are several reasons for this: *(i)* compared to the other four, this predictor was best optimized for DTMF-3 signals; *(ii)* the sgn updates are accurate (after convergence) if the signal is stationary and the step-size small. The large leakage chosen for the LS predictor inhibits better tracking for the sake of robustness — with a smaller leakage, it outperforms TRsgn.

4.4.2 Decoder Tracking

Decoder tracking for the DTMF-3 signal was examined using the following procedure:

1) For the first 2,304 samples, the input was encoded/decoded without transmission errors in order to let the decoder coefficients adjust to the encoder coefficients.

2) For the next 256 samples, bit errors with a rate $p_e = .05$ were injected in the channel and, at the same time, the decoder coefficients were held at the point $\mathbf{A}' = [\,0.1 \quad 0.8\,]$ in order to introduce a known coefficient offset and not a random offset due to transmission errors alone.[3]

3) Finally, for the remaining samples, the errors were removed and the decoder coefficients were released to see whether the predictor would recover.

Figure 4.9 illustrates the encoder/decoder reconstruction error $(x(n) - \hat{x}'(n))$ and coefficient trajectories at step (3). The results show that the signal-driven predictors are mistracking, while the residual-driven predictors are recovering. As with speech, the LR predictor recovers quickly, with the coefficient offset rapidly decreasing in the first few samples after the removal of errors. The TR and TRsgn predictors require approximately 1,000 samples to retrack, and actually exhibit a "burst" in the interim. This is due to the slower convergence of the transversal SG algorithm for stationary inputs.

The DTMF-3 operating point for signal-driven predictors at the decoder is **unstable**. An initial offset in the coefficients leads to a slight power twist in the components (697 and 1477 Hz) of the reconstructed signal. Both the offset and the twist will increase in a direction favouring the growing component, causing a steady shift in the operating point. The direction of the shift depends on the initial conditions at

[3] Either one of these disturbances alone (errors or coefficient offset) can lead to mistracking in the signal-driven predictors; in fact, experiments have shown that a single bit error is enough. Step (2) simply results in a more controlled experiment.

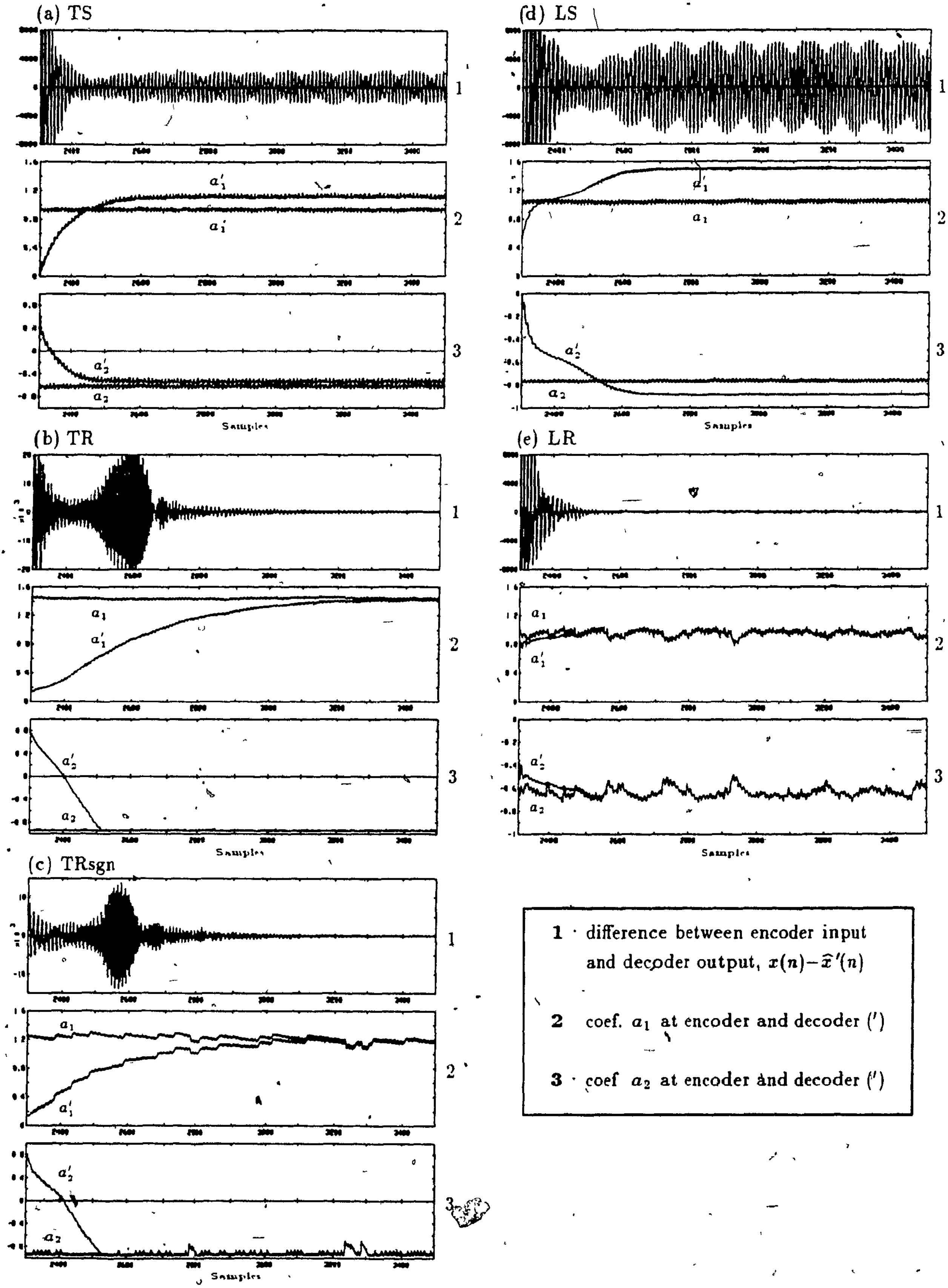


Fig. 4.9 Decoder adjustment after a burst of errors and initial coefficient offset during DTMF-3 transmission

the decoder (offset, received residual, predictor memory, quantizer memory). Tests indicate that mistracking is usually in the direction of the lower frequency component (697 Hz).

Eventually, this process will force the predictor to track one of the sinusoidal components. This is a **stable** mode for the decoder and tests have shown that the process cannot be reversed with additional errors (even when the decoder coefficients are momentarily reset to the DTMF-3 operating point, the decoder continues to drift towards one of the two stable modes). Other data signals containing multiple tones may lead to similar mistracking problems. This instability is not due to model insufficiency in the predictor. Tests confirm that mistracking also occurs with fourth, fifth, and sixth order predictors.

As an example, Figure 4.10 compares the coefficient trajectories at the LS encoder (A–A') with two diverging decoder trajectories (B–B' and C–C'). The three predictors have different initial conditions (A, B, C). We note that the encoder descends the DTMF-3 error surface, decoder 1 descends the 697 Hz error surface, and decoder 2 descends the 1477 Hz error surface. The points B' and C' are stable decoder modes, and the point A' is a stable encoder mode. The resulting power twist between the sinusoids at the reconstructed output ($\hat{x}'(n)$) is shown in Figure 4.11.

The experiments of this section were also carried out with the DTMF-7 signal (852 Hz and 1209 Hz) with similar results. In conclusion, mistracking in the signal-driven predictors causes severe distortion at the reconstructed output of the decoder. The residual-driven predictors always recover after the errors are removed, with the LR decoder, in particular, converging very quickly to the encoder.

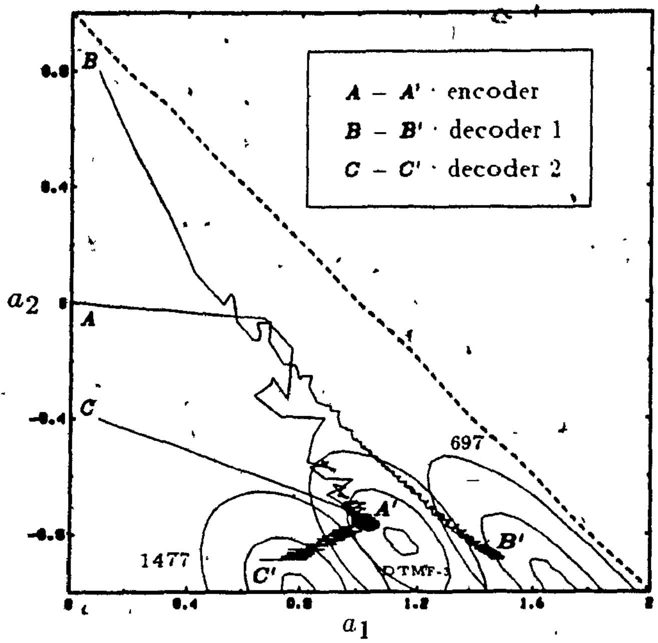


Fig. 4.10 Coefficient trajectories for the LS predictor with different initial conditions

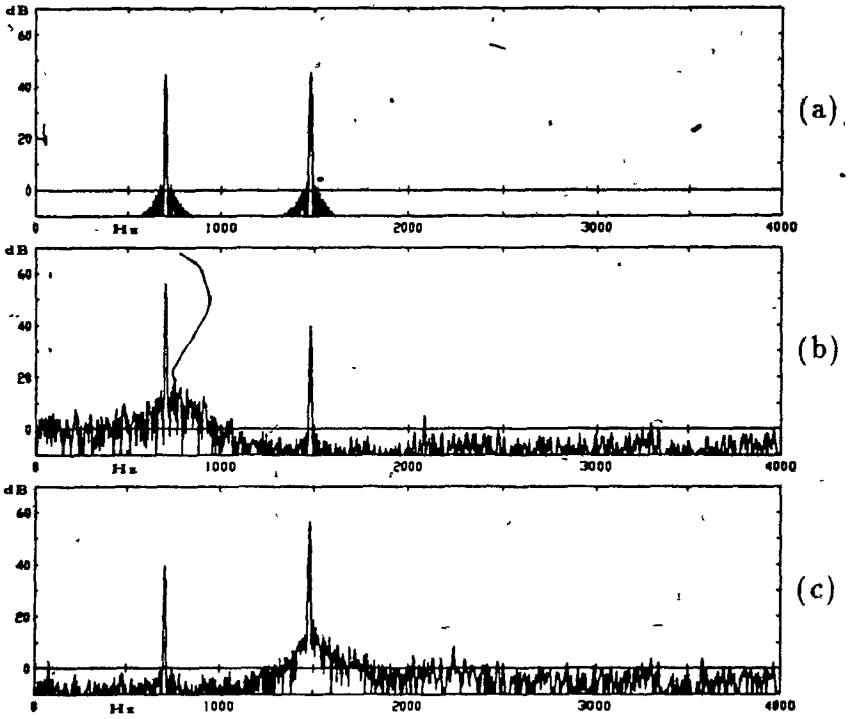


Fig. 4.11 Magnitude spectrum of the reconstructed output: (a) LS encoder; (b) LS decoder 1; (c) LS decoder 2

4.5 Prediction Accuracy for Single-Tone Inputs

This section investigates the tracking accuracy for pure sinusoids. A sine wave,

$$x(n) = A \sin \phi n, \quad \phi = \frac{2\pi f}{f_s}, \quad 0 \le f \le \frac{f_s}{2}, \quad f_s = 8000 \text{ Hz},$$

is perfectly modelled by an $AR(2)$ filter with optimal coefficients

$$\mathbf{A}_{\text{opt}} = [\, 2\cos\phi \quad -1 \,].$$

Its complex conjugate poles lie on the unit circle in the z-plane at an angle ϕ.

Although sine wave *SNR* is not an important objective in ADPCM, we would expect a high *SNR* and G_P with all five predictors for this type of input, with the only degradation in the tracking caused by the limiting stability constraint of (3.9). A high G_P is achieved when the predictor tracks near the optimal coefficient point of the stationary signal. Since the error surface for sinusoidal inputs is elliptic, with the minor axis almost parallel to the a_1 axis, an offest in the a_1 coefficient yields a large drop in G_P. Offsets in the a_2 coefficient do not result in as much of a loss.

After encoding inputs of various frequencies, large offsets in the a_1 coefficient were found for the two transversal residual-driven predictors but not for the other predictors. These offsets result in poor G_P for some of the frequencies. Table 4.5 shows the prediction gain and Figure 4.12 illustrates the shift from the optimal coefficients for each predictor; only sine waves with $f \le \frac{f_s}{4} \doteq 2000$ Hz were tested, due to the symmetry about $a_1 = 0$.

This tracking problem is particularly evident for the TR predictor in Figure 4.12(b), where four of the sinusoids (404, 643, 920, and 1140 Hz) are tracked by the single point $(1.9, -0.95)$.

Bonnet, Macchi, and Jaidane-Saidane [10] give a mathematical analysis of this impairment for the TRsgn predictor. In summary, it is due to the inaccuracy of the

		Prediction Gain G_P (dB)				
f Hz	$\mathbf{A}_{\text{opt}}$	*TS*	*TR*	*TRsgn*	*LS*	*LR*
404	[1 9 −1]	11.4	24.4	23 8	18.1	12 5
643	[1 75 −1]	11.3	12.8	13 9	18.2	12.3
920	[1.5 −1]	12.8	6.5	6.7	18.1	13.5
1140	[1.25 −1]	14.2	2.9	5.6	18.3	13.4
1333	[1 −1]	15.6	3.0	8.3	18.6	17.6
1511	[0.75 −1]	16.9	5.2	11.6	18 8	18.7
1678	[0.5 −1]	17.8	8.0	15.4	18.9	21.9
1840	[0.25 −1]	18.6	14.26	20.3	19 1	20.7
1990	[.016 −1]	18.8	23.54	23.8	19.1	20.9

Table 4.5 Predictor performance for sine waves

gradient estimates in (3.32) which, in the limit, force the coefficients to converge to suboptimal points depending on the frequency of the sinusoid. The same type of impairment plagues the TR predictor. Therefore, these two predictors should not be used for frequency estimation applications.

The TS and LR predictors exhibit large offsets in the a_2 coefficient for the lower frequencies. However, the corresponding loss in G_P is not high. The LS predictor yields the least overall shift and highest average G_P over all frequencies. With a smaller leakage value, δ, the tracking performance of the LS is even better. Thus, this predictor is the most accurate frequency estimator of the five.

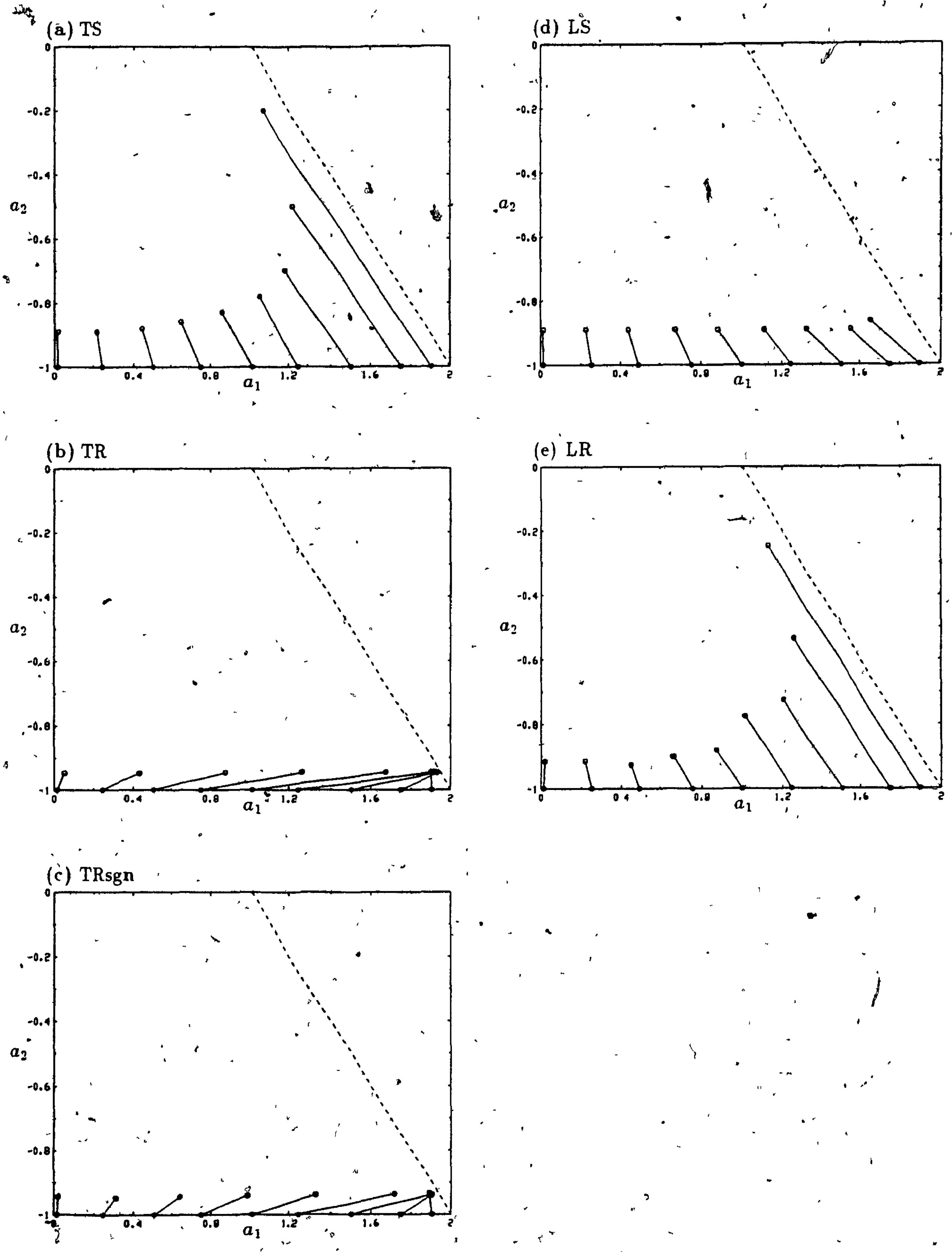


Fig. 4.12 Coefficient shift for sine waves

4.6 Higher Order Predictors

Up to this point, only second order predictors were examined. Increasing the order of the TS predictor does not yield a significant improvement in prediction gain, particularly when we consider the added complexity for verifying the stability of three or more coefficients. The TR (and TRsgn) algorithm cannot be directly extended to higher order prediction; in any case, it is felt that the gain in performance would not justify the high complexity of the algorithm.

Both lattice predictors, on the other hand, can be easily extended to higher orders by adding new lattice stages and ensuring that the new reflection coefficients have magnitude less than one. Prediction gain versus prediction order for the LS and LR predictors is plotted in Figure 4.13.

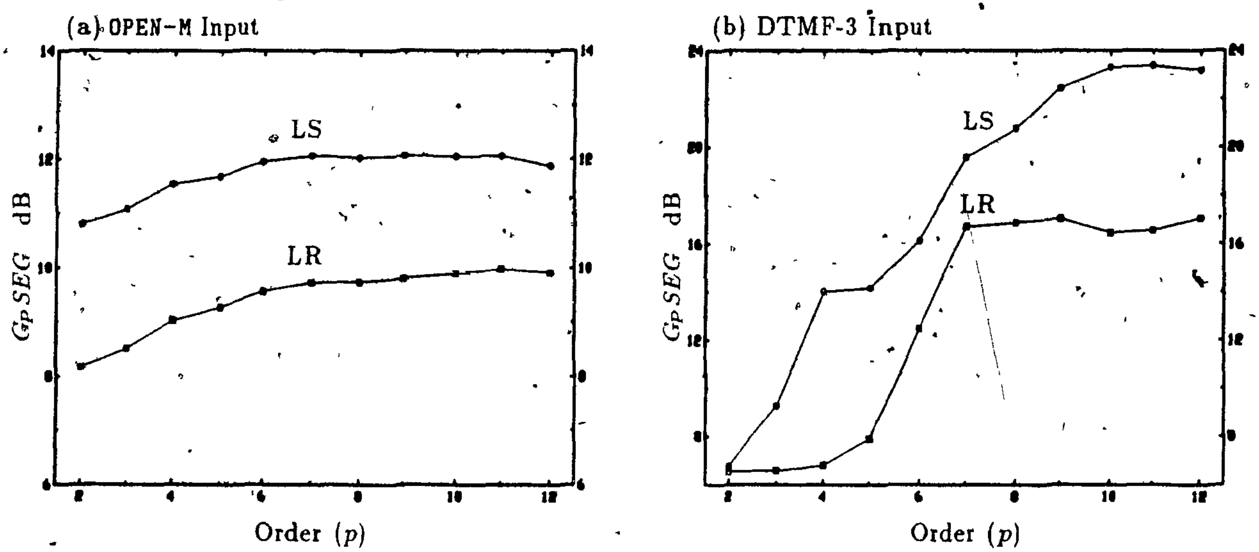


Fig. 4.13 Prediction gain versus prediction order: (a) Speech input OPEN-M; (b) DTMF-3 input

With the OPEN-M input we observe in Fig. 4.13(a) a steady increase in $GpSEG$ for both predictors for $2 \leq p \leq 7$. For $p \geq 8$ there is no significant increase. The maximum increase in $GpSEG$ is 1.25 dB for the LS and 1.8 dB for the LR. These figures may be improved if the predictor parameters are optimized for each order p

and not left at the values found best for $p = 2$, but this was not done. In addition, much higher gains are observed for the LS predictor when a smaller leakage value is used (say $\delta = 2^{-8}$). In this case, however, the robustness of the system would be compromised. With the DTMF-3 input in Fig. 4.13(b) significant increases are noted at $p = 3, 4, 6, 7, 8, 9, 10$ for the LS predictor, and at $p = 5, 6, 7$ for the LR predictor. Maximum increase is 16.6 dB for LS and 10.5 dB for LR. Noisy channel tests indicate that higher order LR predictors never mistrack, whereas higher order LS predictors are prone to mistracking.

4.7 Predictor Complexity

The complexity of the TS, TR, LS, and LR predictors was calculated in terms of the number of arithmetic operations (+/−, ×/÷) and the amount of memory required for processing each sample (*i.e.*, the complexity of generating $\tilde{x}(n)$ from the input $\hat{e}(n)$ and the current state of the predictor).

	Filter			Algorithm			Total		
	Mem	+/−	×/÷	Mem	+/−	×/−	Mem	+/−	×/÷
TRsgn	2	1	2	3	5	5	5	6	7
TS	2	1	2	3	5	11	5	6	13
TR	2	1	2	4	6	14	6	7	16
LS	2 p	5 $3p-1$	3 $2p-1$	4 $2p$	8 $4p$	16 $8p$	6 $3p$	13 $7p-1$	19 $10p-1$
LR	2 p	5 $3p-1$	3 $2p-1$	6 $3p$	12 $6p$	20 $10p$	8 $4p$	17 $9p-1$	23 $12p-1$

Table 4.6 Complexity of adaptive predictors

Table 4.6 shows the filter, algorithm, and total complexity of the five systems as defined at the beginning of this chapter. Constants such as $(1 - \delta)$ are a single entity and thus do not contribute towards the (+/−) count.

Only second order transversal predictors were examined, while the complexity of lattice predictors was generalized for $p > 2$. Note that the LR predictor is about twice as complex, overall, as the TS predictor.

Chapter 5 Summary and Conclusions

The intention of this work was to examine different ADPCM predictor structures utilizing stochastic gradient adaptation algorithms, and to compare their performance in the presence and absence of channel errors In particular, the predictor mistracking problem with narrowband input signals was investigated.

Prevention of mistracking requires that sub-optimal adaptation algorithms, driven solely by the residual signal, be used for updating the predictor coefficients. To this end we have implemented and compared five different adaptive predictors according to filter structure/adaptation algorithm: TS (transversal/signal-driven), TR (transversal/residual-driven), TRsgn (transversal/residual-driven/sgn multipliers), LS (lattice/signal-driven), and LR (lattice/residual-driven).

5.1 Summary

The signal-driven predictors, TS and LS, use traditional transversal and lattice stochastic gradient algorithms based on minimizing the residual power. Coefficient updates in the TS algorithm are derived from correlations between the quantized residual and reconstructed signal. The LS algorithm generates its updates by maximally de-correlating the reconstructed signal. These algorithms utilize coefficient leakage with the aim of improving robustness to channel errors and step-size normalization in order to better track short-term stationary signals. However, in both schemes a form of cross-feedback exists between the prediction synthesis filter and the coefficient adaptation which may lead to encoder/decoder mistracking.

Residual-driven algorithms were designed in order to eliminate this problem. The TR algorithm mimicks the TS algorithm by deriving its coefficient updates through the updates of an approximately equivalent synthesis filter utilizing zeros instead of poles. As a result, the updates are determined by autocorrelations of the quantized residual signal. The TRsgn algorithm is a simplified version of the TR using only the sign of the autocorrelation term. A form of this algorithm has been adopted by the CCITT as a standard for 32 kb/s ADPCM.

The LR algorithm was developed using a different approach. In practice, prediction is not perfect and the quantized residual and reconstructed signals are correlated. Since the LS algorithm generates the optimal coefficients by de-correlating the reconstructed signal, then a set of sub-optimal coefficients can be generated by de-correlating the residual signal. These are subsequently used to drive the lattice predictor. Therefore, an implementation of the LR predictor requires two lattice structures: one for the algorithm and one for the predictor. In the residual-driven schemes, cross-feedback between the synthesis filter and the coefficient adaptation has been eliminated.

The five adaptive predictors were subjected to various tests in order to compare their performance and behaviour in a typical 32 kb/s ADPCM system. All of the experiments were carried out with second order predictors unless otherwise stated. The results are summarized as follows.

(a) Speech performance in the absence of errors. Experiments with several phonetically balanced sentences by both male and female speakers indicate that the TS and LS second order predictors attain an average *GpSEG* (segmental prediction gain) of 10 and 11.5 dB respectively. However, the LS can achieve a gain of over 13 dB if a smaller leakage value is used (large leakage was required in order to improve the predictor's robustness for speech during errors). The difference in *GpSEG* can be attributed to the fact that the LS algorithm automatically generates a different step-size variable at each stage that is normalized by the input power of the stage, whereas the TS algorithm uses a fixed step-size for all coefficients which is normalized by the reconstructed signal power (*i.e.*, input power of the entire predictor). As a result, the LS predictor tracks stationary and non-stationary inputs faster and more accurately than the TS predictor.

Average *GpSEG* values for the TR, TRsgn, and LR predictors are 10.6, 9.9, and 9.3 dB respectively. So, using the residual-driven predictors instead of the TS predictor would not entail a big loss in speech performance; in fact, TR fairs slightly better with most of the test speech inputs. This is a surprising result in view of the sub-optimality of the TR algorithm.

It can be seen from Table 4.3 that the performance variation of the different algorithms is only 1.8 dB in *SNRSEG*. Thus from the point of view of subjective speech quality, little difference is expected to be audible. An overall speech performance rating in order of decreasing *GpSEG* is: LS, TR, TS, TRsgn, and LR.

(b) Speech performance in the presence of errors. Tests with speech signals in the presence of channel errors show that the signal-driven predictors can be made robust by using a large enough value of δ (particularly in the LS predictor). The TS predictor is more robust than the LS predictor with most leakage values. In this case, the automatic step-size generation and normalization in the LS algorithm plays a detrimental role because this process is sensitive to channel errors. Since the TS algorithm uses a fixed step-size, only the normalization term is sensitive to errors. Both schemes are robust to channel errors with bit-error rates (p_e) as high as 10^{-3}. In fact, at the higher bit-error rates it is the quantizer, not the predictor, which limits the overall performance of the system.

The residual-driven predictors are generally more robust than the signal-driven predictors and are not as dependent on the choice of leakage δ. This is to be expected, as error propagation in the quantized residual is not as severe as in the reconstructed signal. Table 4.2 shows that the performance variation under a weighted sum of expected error conditions (SNR_W) is only 1.6 dB. An overall rating in order of decreasing SNR_W is: LS, LR, TRsgn, TR, and TS. Thus the LR predictor, which yields the worst performance without errors yields nearly the best performance in the presence of errors.

The effects of transmission errors during speech inputs are temporary. Specifically, a burst of errors during a voiced speech segment is shown to cause an offset between the encoder and decoder coefficients which remains only for the duration of that segment. Typical convergence times after an error has occurred during voiced speech are on the order of 62.5 ms (500 samples) for TS and TR, and 12.5 ms (100 samples) for TRsgn, LS, and LR. Errors during fricatives, plosives or silence segments are even less severe.

Thus, the signal-driven predictors do not mistrack with speech inputs. Natural re-synchronizing properties of speech (fricatives, silence) and the use of coefficient leakage allow the decoder coefficients to retrack at the end of a voiced segment.

(c) Dual-tone input with channel errors. However, with a dual-tone input signal (DTMF3) there is no guarantee that the encoder and decoder coefficients will converge after an error has caused an initial offset. Experiments have confirmed that both TS and LS can mistrack after a single channel error. This is due to the dynamic instability of the operating point for signal-driven sequential algorithms with dual-tone inputs. Eventually, cross-feedback between the synthesis filter and adaptation algorithm force the predictor to track one of the two components. This is a stable operating point, or mode, for the decoder since experiments have shown that the process cannot be reversed with additional errors.

The observed instability is not due to insufficient modelling of the input signal; *i.e.*, higher order signal-driven predictors are also prone to mistracking. Thus, the above problem can be eliminated only by using residual-driven predictors.

Tests with dual-tone inputs and multiple channel errors confirm that neither residual-driven predictor mistracks. However, there is a significant difference in the convergence time after an error has caused an initial offset in the coefficients. The LR predictor was able to converge after 25 ms (200 samples) while the TR and TRsgn predictors required about 113 ms (900 samples). This is due to the superior convergence properties of the lattice filter.

(d) Tracking a single-tone input. In order to gain further insight in the tracking behaviour of the five predictors, some tests with pure sinusoidal signals (single-tones) were performed. In theory, a sinusoid is perfectly modelled by a second order IIR filter. The chosen frequencies correspond to equally spaced points in the transversal coefficient space (a_1, a_2) with $a_2 = -1$ and $a_1 \geq 0$ (due to symmetry, only positive a_1 were considered). These tests serve in pointing out any model deficiencies in the residual-driven algorithms.

The two signal-driven algorithms were able to track all the sinusoidal inputs with little offset in the coefficients. Average *GpSEG* values are 15.3 dB for TS and 18.6

dB for LS.

The LR predictor performed quite well throughout the frequency range, attaining an average *GpSEG* value of 16.8 dB. Thus, the very nature of the LR algorithm — deriving its updates by de-correlating the residual signal — does not limit its tracking performance with highly predictable signals, only with signals with a large unpredictable component such as speech.

A much poorer result was observed with the TR predictor. All frequencies under 2000 Hz (a quarter of the sampling frequency) were tracked near the single point $(2, -1)$ and those over 2000 Hz near the point $(-2, -1)$. In essence, the TR algorithm is not able to track the frequency content of pure sinusoids, attaining an average *GpSEG* of 11.2 dB but as little as 2.9 dB for specific tones. The TRsgn algorithm performs a little better, with an average *GpSEG* value of 14.4 dB and a minimum of 5.63 dB, but is plagued by the same modelling impairment as the TR.

(e) Sensitivity to predictor parameters. While searching for the optimal predictor parameters (μ, γ, δ) to be used with average speech and dual-tone signals, it was discovered that:

- the performance of the transversal predictors is quite sensitive to different values of μ and δ; in particular, the TR and TRsgn predictors dual-tone performance is limited to a very narrow range of parameter values
- the performance of both lattice predictors is relatively insensitive to the values of μ and γ.

Moreover, in the transversal predictors, the optimal parameters for speech signals do not coincide with those for dual-tone signals. Thus, a compromise set of parameters affects the performance for both types of input. In the case of lattice filters this does not occur, and the selection of parameters which result in good speech and dual-tone performance is a simple task.

(f) Predictor complexity. The complexity of the five predictors (filter and algorithm) was calculated in terms of the number of arithmetic operations $(+/-, */\div)$ and the amount of memory (z^{-1}) required for processing each sample. A rating in order of increasing complexity is: TRsgn, TS, TR, LS, and LR. The results, shown in Table 4.6, indicate that the LR predictor is about twice as complex as the TS predictor.

It is estimated that the current state of DSP chip technology allows a real-time implementation of the LR predictor on a single chip (such as the TMS 32020).

5.2 Conclusions

The final choice for a candidate predictor scheme depends on the following interrelated factors: application, performance, robustness and complexity (cost).

This study was geared towards the application of 32 kb/s ADPCM systems in telecommunications networks. Basic services in the network include speech, voiceband data for modem communications, and signalling tones. Systems in the network must provide acceptable transmission of the basic services in the presence of channel errors.

Based on the above requirements and the results of this study we must preclude the use of the signal-driven adaptive predictors, TS and LS, in the network. Other ADPCM systems, utilizing combinations of fixed-recursive and adaptive-nonrecursive synthesis filters, have been examined by various authors [11][12][13][14]. Although these systems are robust in the presence of errors and are not prone to mistracking, it appears that, due to the inefficiency of fixed and/or nonrecursive synthesis filters, the resulting speech performance is inadequate. Increasing the predictor complexity (*e.g.*, filter order) does not result in a significant increase in performance.

The results of this work indicate that the residual-driven adaptive predictors, TR, TRsgn, and LR, satisfy the basic network requirements. They are shown to be robust in the presence of errors, do not mistrack with data signals, and perform well with speech inputs. A 32 kb/s ADPCM international standard using the TRsgn predictor was adopted by the CCITT in October, 1984 [6].

The LR predictor was developed in this study as an alternative residual-driven scheme. Its main advantages over the TR predictor include:

- modularity in both filter structure and adaptation algorithm which allows for a simple extension of the prediction order
- insensitivity to predictor parameter values
- faster convergence for both speech and data signals in the presence and absence of errors
- higher accuracy in tracking single sinusoids.

All of these are due to the superior properties of the lattice filter. The main disadvantages are:

- slightly poorer speech performance in the absence of transmission errors
- higher implementation complexity.

The drop in speech performance of the LR predictor with respect to the LS predictor is quite significant (2.2 dB in $GpSEG$ for second order filters) and it appears that there is room for improvement here. There are some questions as to whether some further processing of the LR coefficients (before they are used in the lattice prediction filter) would yield an improvement in performance. This processing should reflect the spectral difference between the quantized residual and the reconstructed signal in order to provide more accurate coefficient estimates. To avoid mistracking, however, the processing should not depend in any way on the reconstructed signal. This is left for future research.

References

[1] B. M. Oliver, J. R. Pierce and C. E. Shannon, "The Philosophy of PCM," Proc. IEEE, pp. 1324-1331, Nov 1948

[2] CCITT, Yellow Book, "Pulse Code Modulation of Voice Frequency Signals," Vol. III, Fasc. III.3, Rec. G 711, 1980

[3] J. L. Flanagan, *et al*, "Speech Coding," *IEEE Trans. Commun* , vol. COM-27, pp. 710-736, April 1979

[4] X. Maitre and T Aoyama, "Speech Coding Activities within CCITT. Status and Trends," in *Proc IEEE Int. Conf. Acoust , Speech, Signal Processing*, Paris, France, May 1982. pp. 954-959

[5] W R. Daumer, X. Maitre, P. Mermelstein, and I Tokizawa, "Overview of the ADPCM Coding Algorithm," in *Proc IEEE Global Telecomm. Conf*, Atlanta. GA, November 1984, pp. 774-777.

[6] CCITT, Red Book, "32 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)," Vol. IV, Fasc. III.3, Rec. G 721, 1984

[7] D. W Petr, "32 Kbps ADPCM-DLQ Coding for Network Applications," in *Proc. IEEE Global Telecomm. Conf*, Miami, FL, Nov 29 - Dec 2 1982, pp 239-243.

[8] D. Millar and P Mermelstein, "Prevention of Predictor Mistracking in ADPCM Coders," in *Proc Int Conf Commun*, Amsterdam, Netherlands, May 1984, pp 1508-1512.

[9] J D. Gibson, "Backward Adaptive Prediction as Spectral Analysis Within a Closed Loop," *IEEE Trans Acoust., Speech, Signal Processing*, vol ASSP-33, pp 1166-1174, October 1985

[10] M. Bonnet, O. Macchi and M. Jaidane-Saidane, "A Multicriterion View of Jointly Adaptive Prediction/Quantization with application to ADPCM transmission Part I· Adjustment of the Encoder/Decoder Connection in the 32 kbit/s CCITT Algorithm," paper submitted to *IEEE Trans Inform · Theory* in Nov. 1986

[11] D. Cointot and G De Passoz, "A 60-Channel PCM-ADPCM Converter Robust to Channel Errors," in *Proc IEEE Global Telecomm Conf*, Miami, FL, Nov 29 - Dec. 2 1982, pp 249-253

[12] D. Cointot, "A 32-kbit/sec ADPCM Coder Robust to Channel Errors," in *Proc IEEE Int Conf. Acoust , Speech, Signal Processing*, Paris, France, May 1982, pp. 964-967

[13] R. Maruta, S. Aikoh, T. Nishitani, and N. Kawayachi, "Per-Channel ADPCM Codec for Multi-Purpose Applications," in *Proc. IEEE Global Telecomm. Conf.*, Miami, FL, Nov. 29 - Dec. 2 1982, pp. 244–248.

[14] T. Nishitani, S. Aikoh, T. Araseki, K. Ozawa, and R. Maruta, "A 32 kb/s Toll Quality ADPCM Codec using a Single Chip Signal Processor," in *Proc. IEEE Int Conf Acoust , Speech, Signal Processing*, Paris, France, May 1982, pp. 960–963

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

[16] J. D. Gibson, "Adaptive Prediction in Speech Differential Encoding Systems," *Proc. IEEE*, vol. 68, pp. 488–525, April 1980

[17] L. R. Rabiner and R W Schafer, *Digital Processing of Speech Signals*, Englewood Cliffs, N.J. Prentice-Hall, Inc., 1978

[18] P. Cummiskey, N. S. Jayant, and J. L. Flanagan, "Adaptive Quantization in Differential PCM Coding of Speech," *Bell Syst Tech J.*, vol 52, pp 1105–1118, September 1973

[19] D J. Goodman and R. M. Wilkinson, "A Robust Adaptive Quantizer," *IEEE Trans Commun.*, vol COM-23, pp. 1362–1365, November 1975.

[20] M L. Honig and D G. Messerschmitt, "Comparison of Least-Squares and Stochastic Gradient Lattice Predictor Algorithms Using Two Performance Criteria," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 441–445, April 1984.

[21] M. L Honig and D G Messerschmitt, *Adaptive Filters. Structures, Algorithms, and Applications*, Hingham, MA: Kluwer Academic Publishers, 1984.

[22] D. L. Duttweiler, "Adaptive Filter Performance with Nonlinearities in the Correlation Multiplier," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 578–586, August 1982.

[23] S Haykin, *Adaptive Filter Theory*, Englewood Cliffs, N J Prentice-Hall, Inc., 1986

[24] W. S Hodgkiss and J A. Presley, "Adaptive Tracking of Multiple Sinusoids Whose Power Levels are Widely Separated," *IEEE Trans. Acoust , Speech, Signal Processing*, vol ASSP-29, pp. 710–721, June 1981.

[25] M L. Honig and D G. Messerschmitt, "Convergence Properties of an Adaptive Digital Lattice Filter," *IEEE Trans. Acoust., Speech, Signal Processing*, vol ASSP-29, pp 642–653, June 1981.

[26] R C. Reininger and J. D. Gibson, "Backward Adaptive Lattice and Transversal Predictors in ADPCM," *IEEE Trans. Commun.*, vol COM-33, pp. 74–82, January 1985

Appendices

Appendix A. Speech Inputs Used in the Study

Table A.1 illustrates a corpus of eight speech inputs which is used in the experiments. The corpus is composed of four phonetically balanced sentences each uttered by a group of two speakers. Each group consisted of a male and a female speaker. The signals were bandpass filtered and sampled at 8 kHz. The number of samples in each sequence, as well as the standard deviation, mean, maximum positive sample, and maximum negative sample are also given.

Input	Speaker	Sentence	# Samples	Std. Dev.	Mean	Max	Min
GLUE-M	M1	A	27648	1144	-25 1	5623	-8095
GLUE-F	F1	A	37376	306	-19 7	1475	-2418
HOGS-M	M1	B	25856	1326	-24.8	8514	-9573
HOGS-F	F1	B	36608	253	-18 1	1418	-1641
OPEN-M	M2	C	16640	1771	-17 1	9768	-13850
OPEN-F	F2	C	22784	2174	-18 8	12612	-15514
PIPE-M	M2	D	18434	1553	-18 6	8695	-13550
PIPE-F	F2	D	19712	1879	-0 5	8358	-13051

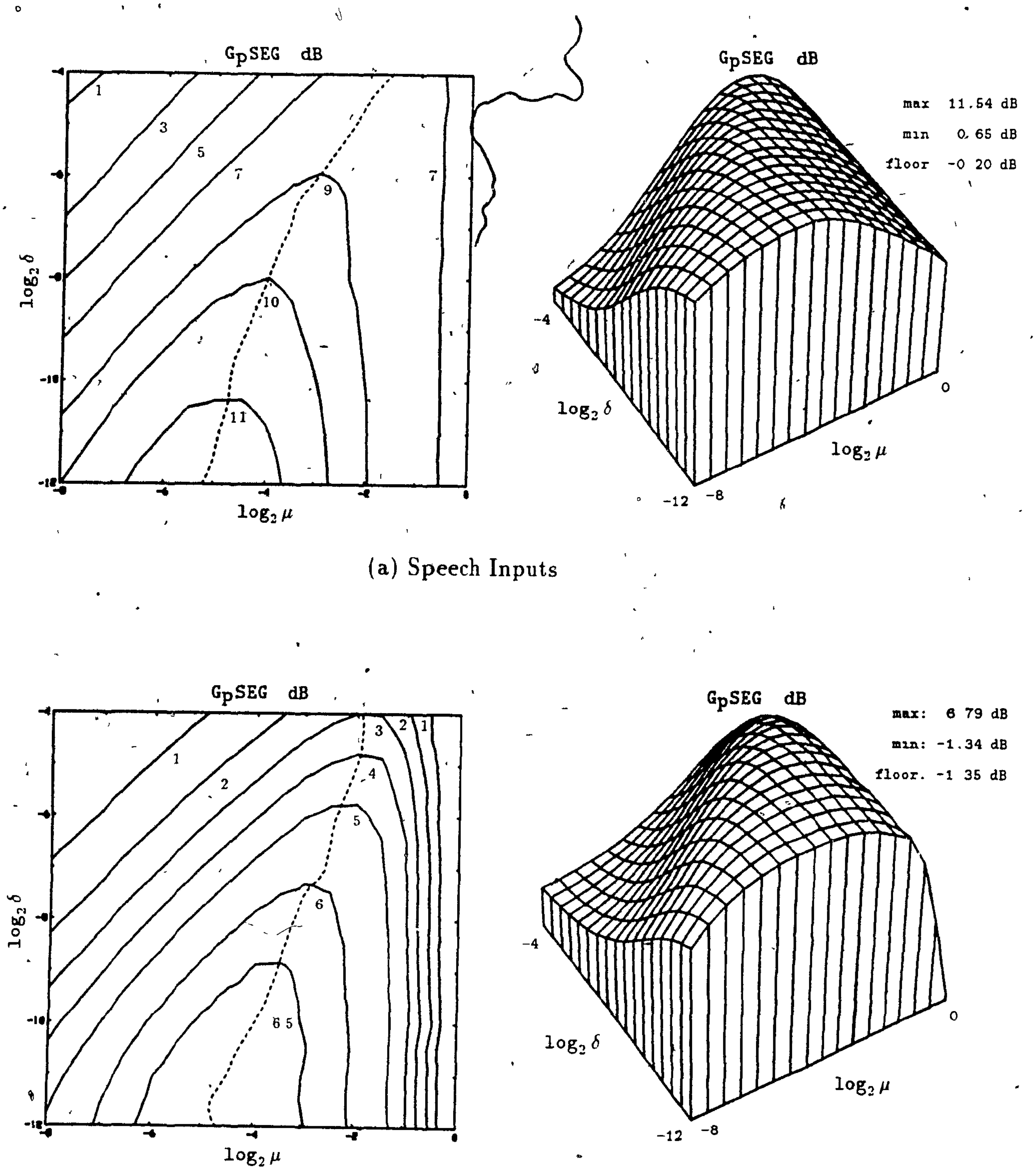
Table A.1 Speech inputs

Speakers:

M1 :	Male
F1 :	Female
M2 :	Male
F2 :	Female

Sentences:

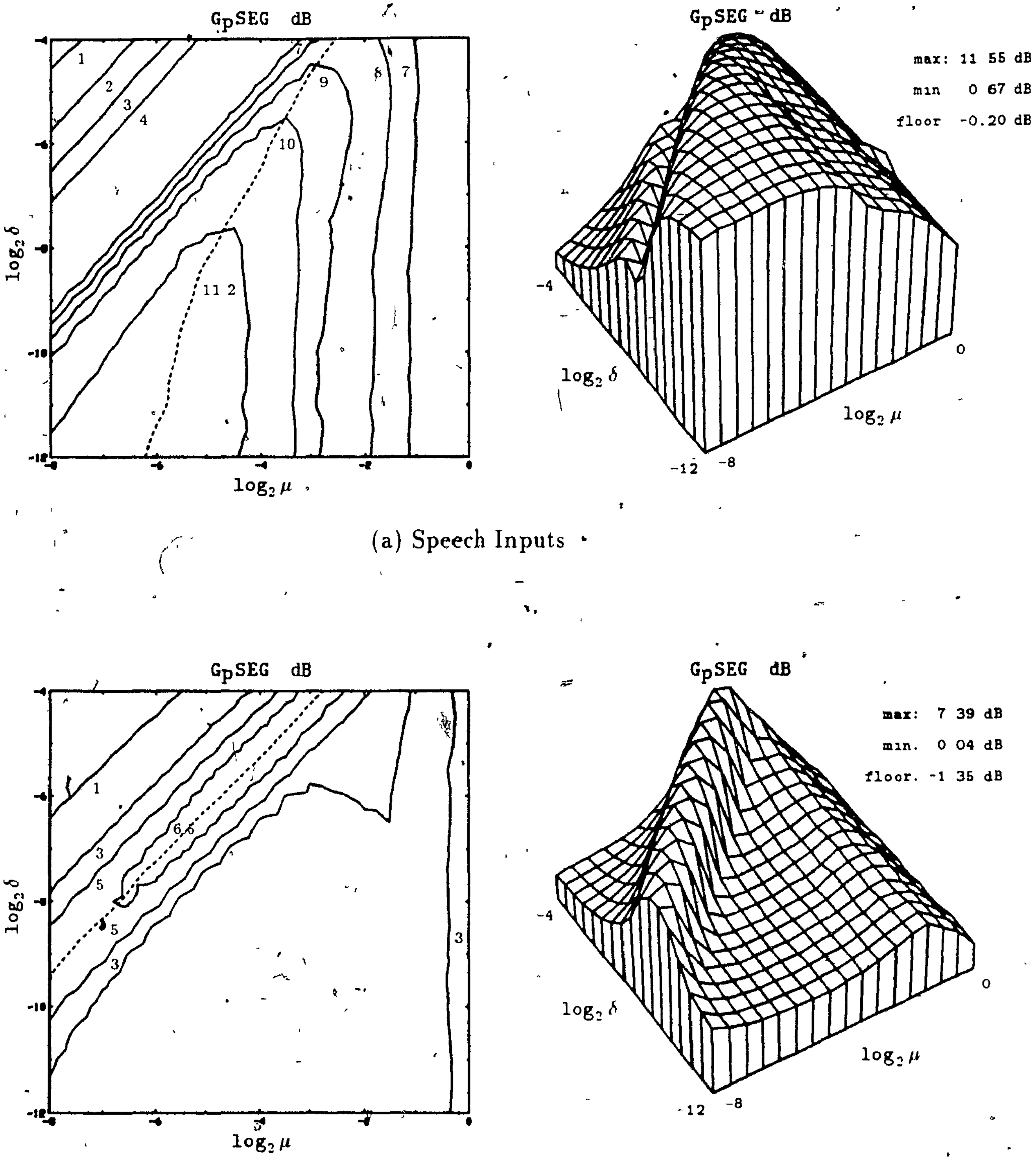
A : Glue the sheet to the dark blue background.
B : The hogs were fed chopped corn and garbage.
C : Open the crate, but don't break the glass.
D : The pipe began to rust while new.



(a) Speech Inputs

(b) DTMF-3 Input

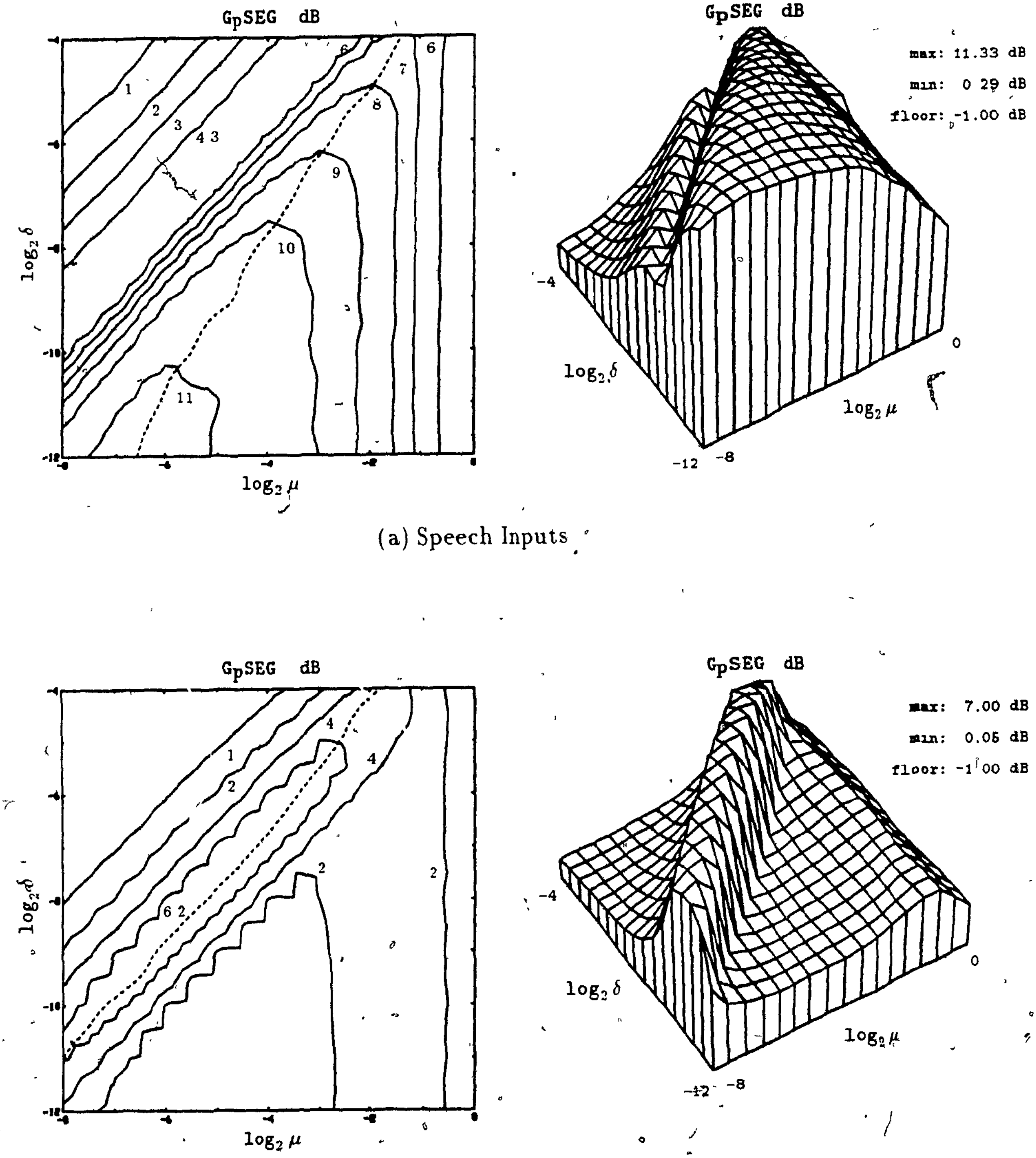
Fig. B.1 Speech and DTMF-3 performance of TS($2; \mu; \delta; .9$) predictor: contour and surface plots of *GpSEG* (dB)



(a) Speech Inputs

(b) DTMF-3 Input

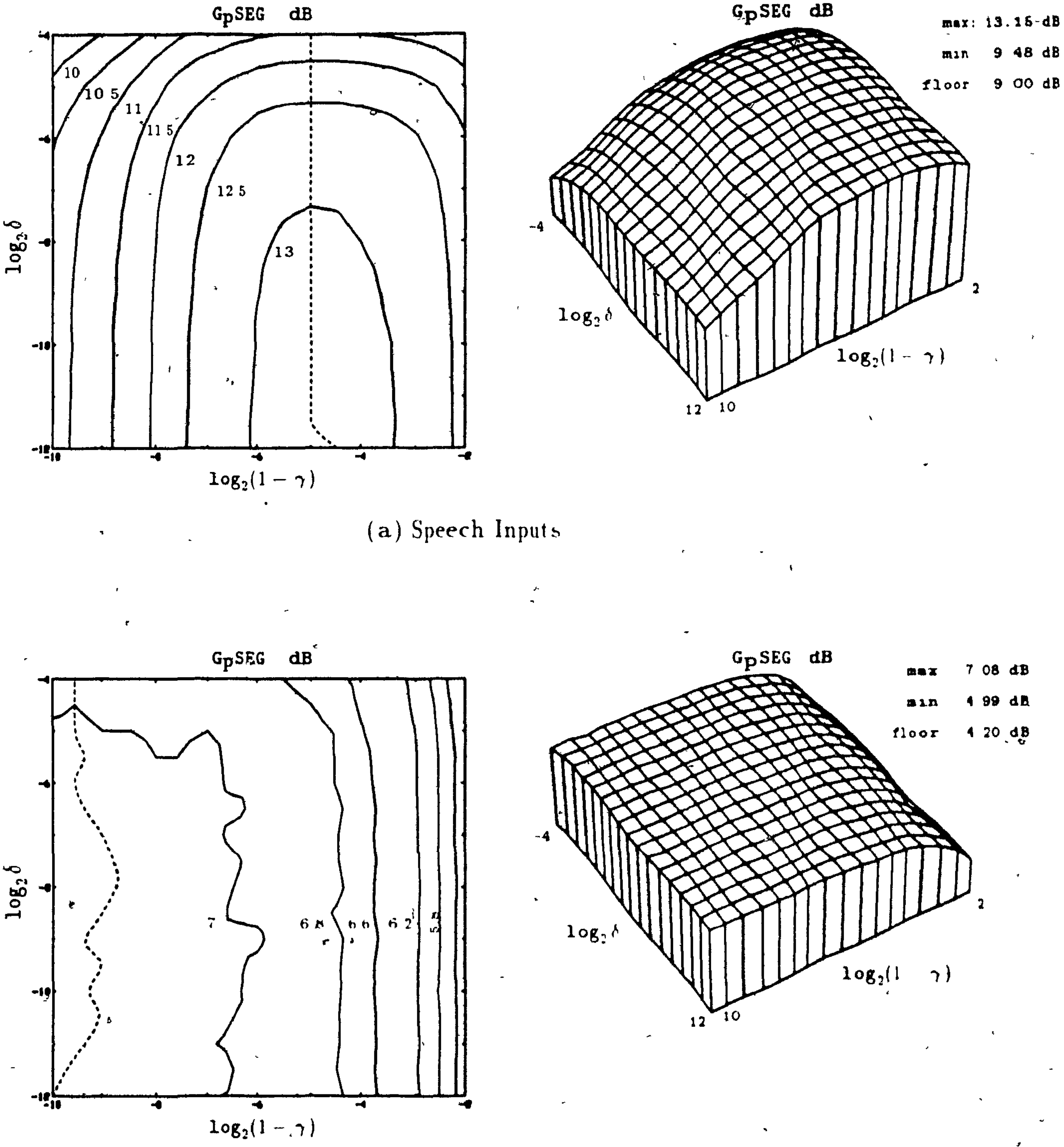
Fig. B.2 Speech and DTMF-3 performance of TR(2; μ; δ; .9) predictor: contour and surface plots of $GpSEG$ (dB)



(a) Speech Inputs

(b) DTMF-3 Input

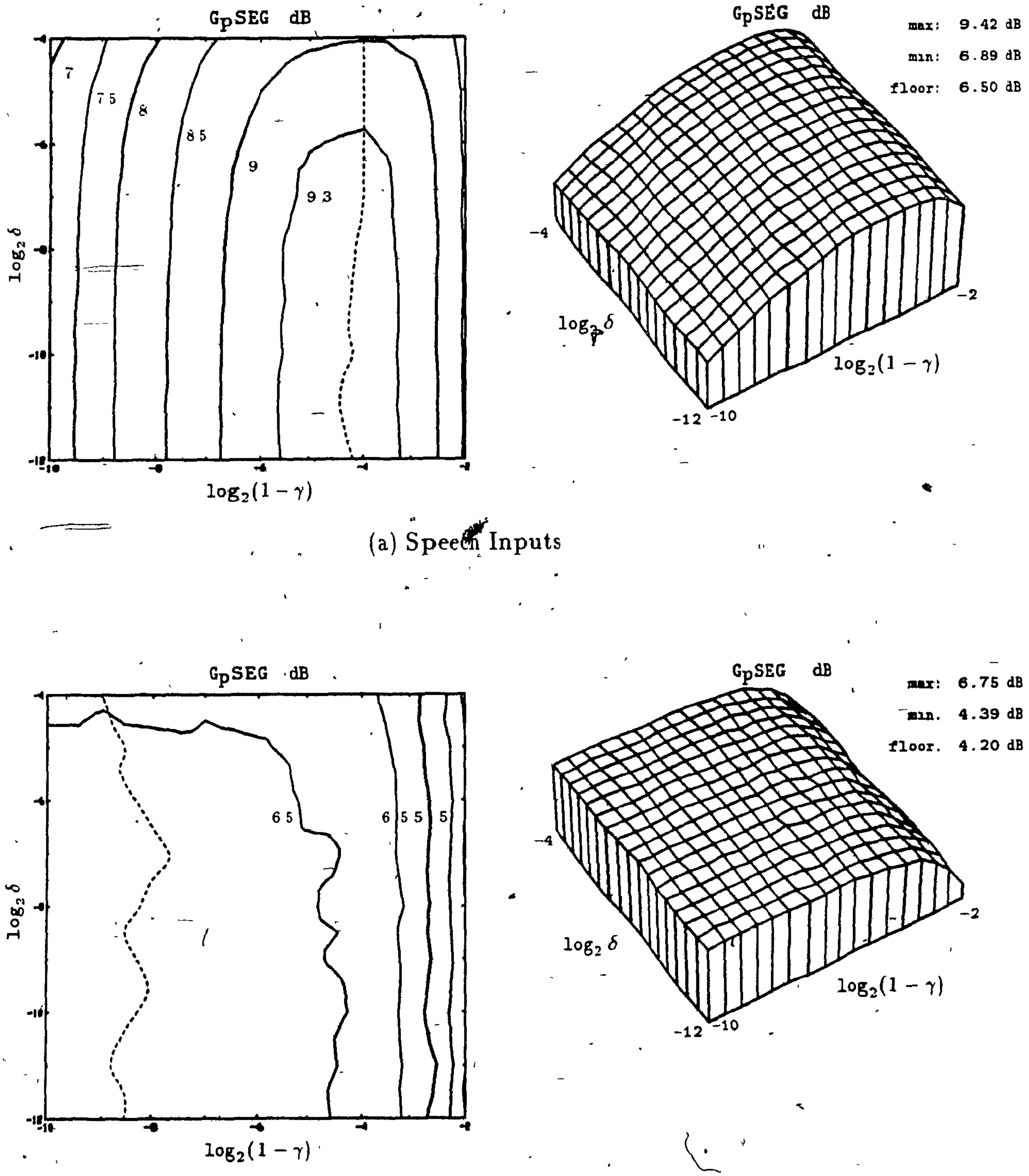
Fig. B.3 Speech and DTMF-3 performance of TRsgn$(2;\mu;\delta)$ predictor: contour and surface plots of *GpSEG* (dB)



(a) Speech Inputs

(b) DTMF-3 Input

Fig. B.4 Speech and DTMF-3 performance of LS$(2;\delta;\gamma)$ predictor: contour and surface plots of $GpSEG$ (dB)



(a) Speech Inputs

(b) DTMF-3 Input

Fig. B.5 Speech and DTMF-3 performance of LR(2; δ; γ) predictor: contour and surface plots of *GpSEG* (dB)