VECTOR QUANTIZATION AND SCALAR LINEAR PREDICTION FOR WAVEFORM CODING OF SPEECH AT 16 kb/s

by

Lloyd Watts

B.Sc. (Eng. Phys.), Queen's University, 1984

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE (ENGINEERING SCIENCE)

in the School

of

Engineering Science

Simon Fraser University

June 1989

APPROVAL

NAME: Lloyd Watts

DEGREE: Master of Applied Science (Engineering Science)

TITLE OF THESIS: Vector Quantization and Scalar Linear Prediction for Waveform Coding of Speech at 16 kb/s.

EXAMINING COMMITTEE:

Chairman: Dr. James Cavers

Dr. Vladimir Cuperman
Senior Supervisor

Dr. John Bird
Supervisor

Dr. Paul Ho
Examiner

DATE APPROVED: June 22/89

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

"Vector Quantization and Scalar Linear Prediction for Waveform Coding of Speech at 16kb/s"

Author: _____
(signature)

Lloyd Watts
(name)

April 5/90
(date)

ABSTRACT

This thesis is an investigation of Vector Quantization, Scalar Linear Prediction and other related signal processing techniques, with the purpose of providing high quality, low delay speech waveform coding at medium data rates (16 kb/s).

Speech waveform coding systems based on adaptive scalar prediction and adaptive scalar quantization have been used to provide toll quality coded speech at high rates such as 32 kb/s (ADPCM). However, the performance of these systems is known to degrade to sub-toll quality at 16 kb/s, due to excessive quantization noise. Vector Quantization (VQ) is well known to provide a significant reduction in quantization noise over scalar quantization; in fact VQ can be shown to have a theoretically optimal rate-distortion performance at very large vector dimensions. This suggests that the performance of 16 kb/s ADPCM may be significantly improved by replacing the scalar quantizer with a vector quantizer.

The resulting configuration, called Vector ADPCM, has an inherently high complexity; however, techniques are described which reduce the complexity to the level where implementation with commercially available digital hardware is feasible. Vector ADPCM is found to provide a 3-dB performance improvement over scalar ADPCM, with a 15 times increase in complexity, while still maintaining an encoding/decoding delay of less than 2 milliseconds. Adaptive Postfiltering significantly improves the subjective quality of the coded speech. Informal listening tests indicate that the coded speech is of very good communications quality.

For Ann

ACKNOWLEDGEMENTS

The author would like to express appreciation to Dr. Vladimir Cuperman for his guidance and supervision, and to Mr. Allan Crawford of Accurex Technology for his guidance and financial support. The author would also like to thank Prof. Daniel Cristall, Dr. Donald Watts, and Dr. John Bird for many helpful discussions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. MOTIVATION FOR RESEARCH.

Digital waveform coding of speech for telecommunications applications began in 1962, when the first commercial digital transmission lines were installed in the United States. The system, still in widespread use today, was based on an early version of Pulse-Code-Modulation (PCM), a simple waveform coding algorithm requiring 64,000 bits of information to be transmitted every second (64 kb/s) for the faithful reproduction of the speech waveform at the receiver.

Advances in solid-state integrated circuit technology and in digital signal processing techniques led to the development of Adaptive Differential Pulse-Code-Modulation (ADPCM), standardized for telecommunications applications by the International Telephone and Telegraph Consultative Committee (CCITT) in 1984. This well-known waveform coding algorithm requires only 32 kb/s for accurate reproduction of the speech waveform, half the data rate required for the original PCM system.

Proposals are now under consideration for a 16 kb/s standard for speech coding, for possible standardization in telecommunications applications by the CCITT in 1990-91. Speech coding at 16 kb/s is therefore a subject of great research interest at the present time. The primary requirements for such a 16 kb/s speech coding algorithm are likely to include:

1. Good subjective speech quality (i.e. toll quality speech). The term *toll quality* is used to describe speech quality acceptable for use in telecommunications applications, and generally implies speech quality comparable to that of analog speech with a 200-3200 Hz bandwidth, a signal-to-noise ratio of 30 dB and less than 2.3% harmonic distortion[19].
2. Medium-to-Low complexity. Characterizations of complexity are loosely defined in the literature and change with advances in technology. In this context, *medium-to-low complexity* means roughly that the algorithm can be

implemented using a single Digital-Signal-Processing Integrated Circuit (DSP chip). More precisely, *low complexity* is generally used to describe algorithms which require a few multiplications per sample (such as PCM); *medium complexity* coders would require up to a few hundred multiplications per sample (such as ADPCM); and *high complexity* coders would require more than about six hundred multiplications per sample, which is beyond the processing power of current state-of-the-art DSP chips.

3. Low Delay. Impedance mismatches in telephone equipment result in echoes of the transmitted signals, which can be perceptible if there is a round-trip delay in the transmission link of over 80 milliseconds. To allow for other sources of delay in the transmission link, it is desirable for the one-way delay (one encoding and one decoding) of the coding algorithm be as short as possible, preferably below 5 milliseconds.

Secondary requirements for the 16 kb/s speech coding algorithm are likely to include:

4. Good performance with voice-band data signals. The speech coding algorithm would also be required to accurately encode modem signals and Dual-Tone-Multi-Frequency (DTMF, or touch-tone) signals.

5. Good performance in the presence of transmission bit errors. Channel noise on the digital link will inevitably corrupt the digital data used by the receiver to reconstruct the speech waveform. A digital link is considered to be usable at average bit-error-rates as high as one bit-error in one thousand bits. The speech coding algorithm should give good performance at error rates up to this level.

6. Good performance for tandem transcodings with PCM and ADPCM. In practice, a coding/decoding device (CODEC) may be used in series with other standard coding devices. Multiple encodings/decodings with the same CODEC or different CODECS should not degrade the signal excessively.

This thesis is an investigation of various signal processing techniques with the purpose of providing high quality, medium-low complexity, low delay speech waveform

coding at medium rates (16 kb/s). The secondary issues of voice-band data transmission, bit error performance, and tandem transcoding performance are beyond the scope of the current work.

1.2. BACKGROUND AND RESEARCH METHODOLOGY.

ADPCM, which is based on Scalar Linear Prediction and Scalar Quantization, is known to provide good quality speech waveform coding at 32 kb/s. However, the performance of ADPCM degrades to an unacceptable level at 16 kb/s, largely due the excessive noise introduced by the quantizer. This suggests that an improvement in the quantizer could result in an improvement of the composite coding algorithm.

Vector Quantization (VQ) has been identified as a promising technique for digital coding of analog signals, with theoretically optimal rate-distortion performance at high vector dimensions. In Vector Quantization, groups of adjacent samples are quantized together by selecting a codeword from a codebook which minimizes some distortion measure (mean-squared-error in this work). In practice, the main obstacle to the use of Vector Quantization is the exponential growth of codebook search complexity with vector dimension. One approach to exploit the performance of the Vector Quantizer is to combine it with other redundancy removal procedures, such as Linear Prediction.

The idea of combining Vector Quantization with Linear Prediction, in order to reduce the complexity for a given performance level, was proposed by Cuperman and Gersho in 1982[16]. The proposed algorithm was Adaptive Differential Vector Coding (ADVC), a vector generalization of ADPCM, which used a Vector Quantizer and a Vector Predictor. This work indicated the promise of Vector Quantization in conjunction with Linear Prediction, however the performance of the Vector Predictor was considerably poorer than the well-known scalar predictor.

Thus, for the current work, a natural candidate for investigation was the combination of a Vector Quantizer and a Scalar Predictor in an ADPCM-like configuration. Such a combination would address the problems of high quantization noise in scalar ADPCM, and poor predictor performance in ADVC.

In summary, the current work focusses on the combination of Vector Quantization, Scalar Linear Prediction and other signal processing techniques, for medium complexity, medium rate speech waveform coding.

1.3. OUTLINE OF THESIS

The history of digital speech waveform coding and its application in the telecommunications network environment are described in Chapter 2. Current and expected future directions of digital speech waveform coding in the network are also described as a motivation for the present work.

Chapter 3 describes the previous development of relevant signal processing techniques, including Scalar Quantization, Linear Prediction, ADPCM, Vector Quantization, Analysis-by-Synthesis techniques, and Adaptive Postfiltering.

In Chapter 4, a new solution called Vector ADPCM (VADPCM) is presented for medium rate speech waveform coding. This speech coding algorithm incorporates a Vector Quantizer and a Scalar Linear Predictor in an Analysis-by-Synthesis configuration. The algorithm has good speech quality and inherently low delay. Methods for substantially reducing the inherently high complexity of the algorithm with little or no degradation in performance are described. Adaptive Postfiltering is used to further improve the subjective speech quality.

Simulation results and complexity estimates for the proposed algorithm are presented in Chapter 5. Conclusions of the research and directions for future research are discussed in Chapter 6.

2. DIGITAL SPEECH WAVEFORM CODING FOR TELECOMMUNICATIONS.

2.1. DIGITAL SPEECH IN THE NETWORK ENVIRONMENT.

Pulse-Code-Modulation (PCM) is the simplest form of digital speech coding, in which the speech signal is sampled and encoded as a stream of binary pulses. PCM was invented in France before World War II[23], and was immediately recognized as having two important advantages over analog (continuous in amplitude and time) transmission of speech:

1. It could tolerate high levels of noise and distortion without impairing the encoded signal, and
2. Repeaters could be used to regenerate the PCM-encoded signal, thus preventing the accumulation of noise and distortion effects in long repeatered systems.

For these reasons, PCM became a subject of research at Bell Laboratories, where several engineering studies of digital speech systems were performed in the 1940's. At the time, it was found that digital speech had two major disadvantages:

1. It required high speed logic circuits, which could not be built inexpensively and reliably using the existing vacuum tube technology, and
2. It required about ten times the bandwidth of a conventional analog system.

The first problem was recognized by M. J. Kelly, director of research (and later president of Bell Laboratories), who realized that the telephone system required electronic switching and better amplifiers to replace vacuum tubes. In 1945, a solid-state physics group was formed with the objective of obtaining "new knowledge that can be used in the development of completely new and improved components and apparatus elements of communication systems" [35]. One of the most important specific goals, the development of a solid-state amplifier, was achieved in 1947-48 by Brattain, Bardeen and Shockley, with the development of the transistor. This important device had improved amplification characteristics over the vacuum tube, without the need for a heated

filament. In 1950, Bell Labs was able to produce the very pure semiconductor crystals required, and by 1951 the transistor was being produced commercially. This led to the development over the next few years of the reliable and inexpensive high speed logic circuits necessary for the feasibility of digital speech coding in the network.

The higher bandwidth requirement for PCM is a serious problem in many radio applications, where bandwidth limitations can be very severe. However, for cable transmission, the higher bandwidth requirement is not such a serious problem. In cables, degradations such as crosstalk and noise increase with frequency, placing a limit on useful bandwidth. However, PCM is more tolerant to these degradations than analog transmission, and thus can use high frequencies that would not have been available for analog transmission[23].

With the two major objections removed, digital speech coding in the telecommunications network became feasible. The first application of digital speech was on exchange trunks - the cables which interconnect switching centers in and around cities. At that time, the Bell System required a new low-cost-per-channel system which could carry both speech and switching control signals over the relatively short (average 6 miles) exchange trunks. Engineering studies indicated that PCM could carry the required signals, and its cost-per-channel was low since many channels could be multiplexed through the same cable and terminal equipment.

As a result of these engineering studies, an exploratory PCM system, called T1, was developed in 1955. The T1 system allowed 24 PCM voice channels to be transmitted over a single cable pair. Each PCM voice channel was encoded by sampling the waveform 8,000 times per second, and encoding the sample amplitude with 8 bits (neglecting signalling bits), resulting in a total bit rate per channel of 64,000 bits/second (64 kb/s). The 24 PCM channels were time-division-multiplexed (TDM), or interleaved in time, with a small amount of synchronization data, for a total bit rate of 1.544 Mb/s.

An experimental T1 system was tested in 1958, and the first successful field trials of T1 were carried out in 1961 and early 1962. Installation of the first commercial T1s followed shortly afterward.

2.2. FROM 64 kb/s PCM TO 32 kb/s ADPCM.

Digital speech transmission systems found increasing popularity with local telephone companies through the 1960s and 1970s because of their reliability, low maintenance cost, and space savings. The deployment of Electronic Switching Systems (ESS), beginning with the No. 4 ESS in 1976, ushered in an era of integrated digital switching and transmission, paving the way for tremendous new features and cost savings.

In particular, it was realized that a further cost saving could be achieved by reducing the data rate for digital transmission of speech from 64 kb/s. This would allow the telephone company to carry significantly more calls with the same equipment. In June, 1982, the need for an international 32 kb/s coding standard was formally identified by the CCITT. An expert group was given the mandate to recommend and fully specify a 32 kb/s waveform coding algorithm[9].

The CCITT expert group quickly identified the requirements of the new 32 kb/s algorithm, including:

1. The algorithm should sample at 8 kHz and encode at 4 bits per sample, for compatibility with existing PCM equipment.
2. The algorithm should not rely on side information to transmit parameters or maintain frame alignment.
3. The algorithm should be able to recover gracefully from transmission errors.
4. The algorithm should be able to carry DTMF tones and voice-band data signals up to 4800 bits/second.
5. The algorithm should maintain adequate performance in the presence of synchronous and asynchronous transcodings with PCM.

ADPCM was identified early in the CCITT submissions process as an algorithm which was likely to meet the requirements outlined by the CCITT expert group. The basic ADPCM algorithm for speech is based on subtracting a prediction of the current sample from the current sample, and quantizing the prediction error with a 4-bit quantizer. However, the CCITT expert group realized that optimizing the ADPCM algorithm

for both voice and non-voice signals was considerably more challenging than optimizing for voice alone.

Over the next 18 months, the expert group selected and fully defined an ADPCM algorithm of reasonable complexity which could meet the above performance requirements. The algorithm, formally approved in the October 1984 CCITT plenary session as an international standard, is specified in detail in CCITT Recommendation G.721. This led to the development of several Very Large Scale Integration (VLSI) single-chip 32 kb/s ADPCM codecs and transcoders by the larger integrated circuit manufacturers.

2.3. THE FUTURE 16 kb/s SPEECH CODING STANDARD.

By 1988, integrated circuit technology and digital signal processing techniques for speech coding had advanced to the point where a moderate complexity, low delay, toll quality 16 kb/s speech coding algorithm appeared to be within reach. A CCITT Ad Hoc Group was established to specify requirements and investigate algorithms for a possible 16 kb/s speech coding standard.

The requirements for the 16 kb/s coding algorithm are expected to be substantially the same as those for the 32 kb/s algorithm, except that it will encode at 2 bits per sample, and it will have to allow for tandem transcoding with both PCM at 64 kb/s and ADPCM at 32 kb/s. The announcement of a 16 kb/s standard is expected in 1990-91.

2.4. LOW RATE SPEECH CODING.

A considerable body of work exists on speech coding at rates below 16 kb/s. In the present environment, these may be classified roughly by their rate and subjective quality:

1. *Toll quality* is currently considered achievable at bit rates above 14 kb/s. Speech of this quality is considered acceptable for use by the general public. The primary data rates of interest have been submultiples of 64 kb/s, such as 32 kb/s and 16 kb/s, for compatibility with existing network equipment. The

most popular coding techniques in this range have been PCM, ADPCM, and modified DPCM algorithms, such as Delta Modulation[29]. These algorithms are called *waveform coders* since they attempt to faithfully reproduce the input waveform. It should be noted that the current work falls in this category.

2. *Communications Quality* is generally achievable at bit rates below 14 kb/s and above 5 kb/s. Speech of this quality is considered acceptable for military, amateur and citizens-band radio operators. The most popular data rates in this range have been 8 kb/s, a submultiple of 64 kb/s for possible network applications, and 9.6 kb/s, for which modems are commercially available. The primary coding methods in this range are Adaptive Transform Coding[39], Subband Coding[14], Adaptive Predictive Coding[4], and Multi-Pulse Linear Predictive Coding[2].

3. *Synthetic Quality* is used to describe speech coding below 5 kb/s. Generally speech of this quality is used where intelligibility is required but human-sounding naturalness can be sacrificed. The most common data rates in this range have been those for which modems are commercially available, such as 4.8 kb/s, 2.4 kb/s, and 1.2 kb/s. Code-Excited Linear Prediction (CELP) is popular at 4.8 kb/s, and Vocoder (VOice CODER) techniques, such as Linear Predictive Coding, are commonly used at 2.4 and 1.2 kb/s. LPC is often called a *parametric* coding algorithm, since it generally depends upon a parametric description of the transfer function of the human vocal tract to achieve its low rates, and does not attempt to reproduce the input waveform exactly.

Several of the signal processing techniques developed originally for these lower rates are applicable to the present work, which addresses the problem of low delay, medium complexity speech coding at 16 kb/s. The next chapter describes the relevant previous work in detail.

3. REVIEW OF PREVIOUS WORK.

3.1. INTRODUCTION.

The focus of the present work is on medium complexity, low delay 16 kb/s speech waveform coding. A natural direction for investigation was an extension of 32 kb/s ADPCM to 16 kb/s. Such a system would encode each prediction error sample with two bits. This avenue has been tested extensively by other researchers [24, 39]. It has been found that the performance of ADPCM degrades to below toll quality at 16 kb/s, due to the excessive quantization noise at only 2 bits/sample.

This indicated two general areas for improvement in the ADPCM configuration:

1. Can the quantization noise be reduced at 2 bits/sample?
2. Can the quantization noise be made less perceptible at 2 bits/sample?

The first question is addressed by an investigation of Vector Quantization, which has been shown to have a theoretically optimal rate-distortion performance. The introduction of a Vector Quantizer into the ADPCM configuration is thus expected to improve the coding performance of the system.

The introduction of a Vector Quantizer into the inherently scalar ADPCM configuration is not trivial. However, there is a precedent in the Analysis-by-Synthesis technique of Atal and Schroeder[3], which has been used to interconnect a vector process and a scalar process, with a considerable increase in complexity.

The second question of making the quantization noise less perceptible in an ADPCM configuration has been addressed by Jayant and Ramamoorthy[27], who added adaptive postfiltering and noise-shaping to ADPCM.

This chapter describes the previous work on the above signal processing techniques, as a motivation for the present solution. We begin in Section 3.2 with a brief description of Scalar Quantization techniques. Section 3.3 follows with a description of Linear Prediction and Differential PCM, with an emphasis on Adaptive Differential PCM. Particular attention is paid to adaptation of the linear predictor and predictor stability. The CCITT

32 kb/s ADPCM algorithm is described in Section 3.4 as an important special case of linear predictive encoding. The performance of ADPCM over the range of bit rates from 32 kb/s to 16 kb/s is discussed as a motivation for quantizer improvement.

Section 3.5 deals with previous work on Vector Quantization. The Vector Quantizer is defined, and the optimal iterative codebook design procedure, the LBG algorithm[30], is described. Adaptive Differential Vector Coding (ADVC), a vector generalization of ADPCM proposed by Cuperman and Gersho[16], was the first approach to combining a Vector Quantizer with a Linear Predictor. The properties of this algorithm are discussed.

The Analysis-by-Synthesis configuration is described in Section 3.6 in the context of Code-Excited Linear Prediction (CELP) [3]. Section 3.7 deals with Adaptive Postfiltering and Noise-Shaping [27] to improve the subjective quality of ADPCM-coded speech.

3.2. SCALAR QUANTIZATION TECHNIQUES.

The function of a quantizer is to map the amplitude of the input sample $x(n)$ into the nearest one of a finite set of possible amplitude levels $y(n)$, where *nearest* means the level which minimizes some distortion measure $D(x(n),y(n))$. Usually the distortion measure is the mean squared error

$$D(x(n),y(n)) = (x(n) - y(n))^2 \tag{3.1}$$

The mean-squared-error distortion measure is used in the present work.

Usually a binary word i is used to represent each of the possible amplitude levels. The inverse quantizer at the decoder then generates the the amplitude value $y(n)$ corresponding to the received binary word.

The quantization noise $q(n)$ introduced by this quantization procedure is the difference between the input sample $x(n)$ and the quantized sample $y(n)$:

$$q(n) = x(n) - y(n) \tag{3.2}$$

The number of amplitude levels L used to represent the sample and the transmission rate R are related by:

$$R = \log_2 L \qquad bits/sample \tag{3.3}$$

For example, a PCM system sampling at 8 kHz and transmitting 64 kb/s has a rate of 64/8=8 bits/sample, and therefore 2^8=256 amplitude levels.

3.2.1. Performance Measures: Signal-to-Noise Ratio and Segmental SNR.

The simplest and most commonly used performance measure is the signal-to-noise ratio, defined in terms of the variance of the input signal σ_x^2, the variance of the quantization error signal σ_q^2.

The *Signal-to-Noise Ratio* is defined as

$$SNR = \sigma_x^2 / \sigma_q^2 \tag{3.4}$$

and in practice, for zero-mean signals, the estimate

$$SNR = \frac{\sum_{i=1}^{N} x^2(i)}{\sum_{i=1}^{N} q^2(i)} \tag{3.5}$$

is used, where N is the number of samples in the estimate. While this measure is very often used, it does not correlate well with subjective impressions of speech quality. The main reason for this is that periods of high energy in the non-stationary speech signal tend to dominate the SNR, obscuring the coder's performance on weak signals.

The *Segmental SNR (SEGSNR)* is based on a dynamic time-weighting to compensate for the under-emphasis of weak signals in the conventional SNR calculation. SEGSNR is computed by measuring the SNR (dB) in short frames (typically 60 ms), and calculating the average frame SNR (dB) value. The log-weighting in the conversion to dB values adds the emphasis to the weak-signal intervals.

3.2.2. Non-uniform Quantizers.

In its simplest form, the *uniform quantizer,* the quantizer levels are equally spaced. However, this does not necessarily yield the smallest error variance. One common approach for improving quantizer performance is the use of a non-uniform quantizer.

Non-uniform quantizers are based on the idea of choosing closer levels where there is a high probability of occurrence of $x(n)$, and more distant levels where the probability of occurrence of $x(n)$ is low. Clearly this requires *a priori* knowledge of the probability density function $p_x(x)$.

There are two well-known techniques for designing non-uniform quantizers for a given input probability density function $p_x(x)$. The first technique, called *companding* for *com*pressing and ex*pand*ing, is used in practical systems where robustness in the presence of signals with a wide dynamic range is needed. This technique is based on passing the input signal through an amplitude nonlinearity which emphasizes small signals and compresses large signals, quantizing with a uniform quantizer, and passing the signal through the inverse of the amplitude nonlinearity. For non-predictive speech coding at high bit rates such as 7 and 8 bits per sample, a logarithmic non-linearity is often used for compression, and the resulting scheme is called *log PCM* or *log-companded PCM*. This companding technique is generally not used at lower bit rates, or in DPCM systems, where adaptive quantizers are preferred.

Optimization of the non-uniform quantizer is based on an iterative solution which minimizes the mean-squared error. The optimum reconstructed amplitude levels $y_{k,opt}$ and decision levels $x_{k,opt}$ were first given by Max[33] and Lloyd[31] :

$$x_{k,opt} = \tfrac{1}{2}(y_{k,opt} + y_{k-1,opt}); \qquad k = 2,3,...,L, \tag{3.6}$$

$$x_{1,opt} = -\infty\,, \; x_{L+1,opt} = \infty \tag{3.7}$$

$$y_{k,opt} = \frac{\displaystyle\int_{x_{k,opt}}^{x_{k+1,opt}} x p_x(x)dx}{\displaystyle\int_{x_{k,opt}}^{x_{k+1,opt}} p_x(x)dx}\,; \qquad k = 1,2,...,L \tag{3.8}$$

Equation (3.6) states that the optimum decision levels are halfway between neighboring reconstruction levels, and equation (3.8) states that a reconstruction level should be the centroid of the pdf within its interval. From these conditions, it is possible to calculate the quantizer decision and reconstruction levels iteratively.

3.2.3. Adaptive Quantizers.

Adaptive Quantization is based on the idea of changing the quantizer characteristics based on the local statistics of the input signal. The most common adaptive quantization schemes adapt the step size $\delta(n)$ (the uniform spacing between reconstruction levels) in response to changes in a short-term estimate of input signal variance $\hat{\sigma}_x^2$. Quantizer adaptation strategies can generally be classified as either *forward-adaptive* or *backward-adaptive*. Block diagrams of the two strategies are shown in Figure 3.1.

In forward-adaptive quantization, the new step size $\delta(n)$ is based on the short-term statistics of the unquantized input signal $\{x(n)\}$. Since this signal is not available at the decoder, these raw statistics (or the new step size) must be quantized and sent to the decoder as *side information*. This is a significant disadvantage for forward-adaptive quantization.

However, in backward-adaptive prediction, the predictor coefficients are based on the short-term statistics of the quantized signal $y(n)$, which is available at the decoder, and therefore there is no need to send any side information. At medium and high data rates ($\geq$ 2 bits/sample), when the quantization noise is small, the statistics of the reconstructed signal agree closely with the statistics of the unquantized input signal, thus good quantizer adaptation can be achieved without the need for side information. For this reason, backward-adaptive systems are preferred at these data rates.

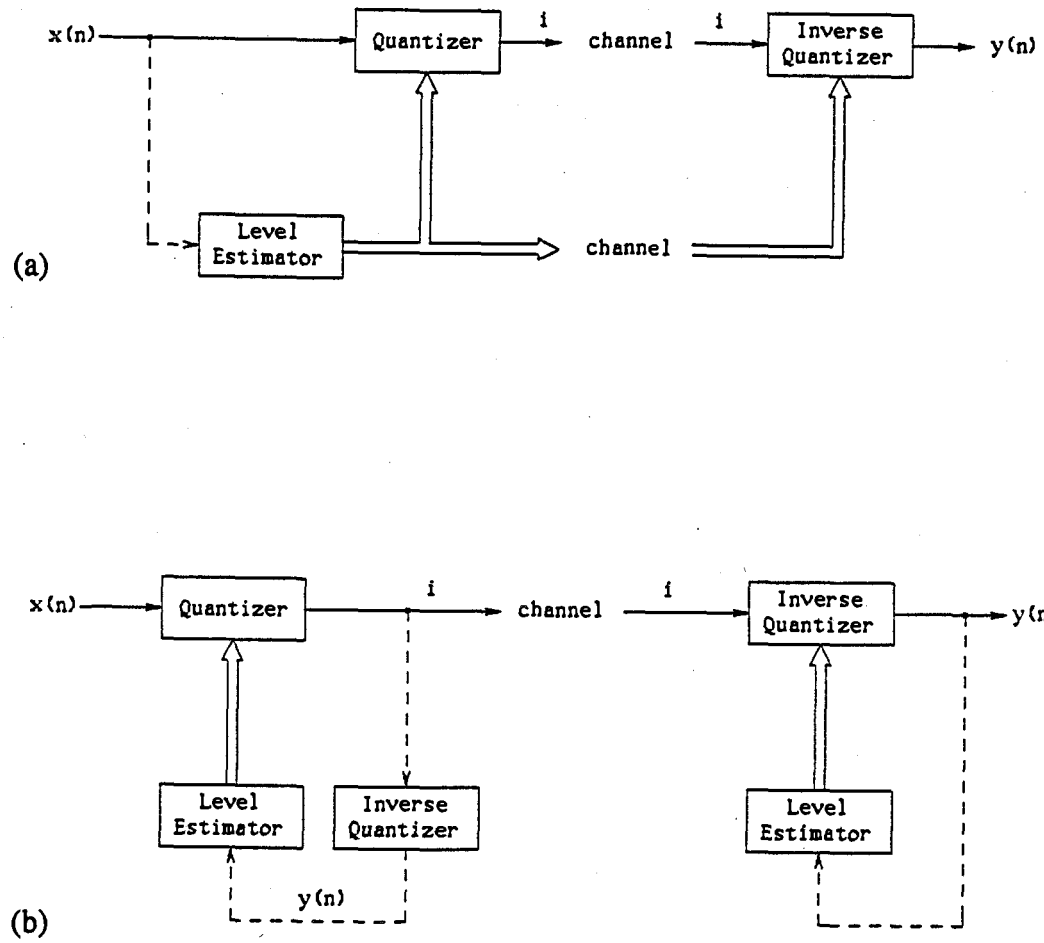

FIGURE 3.1. Adaptive Quantization. (a) forward-adaptive (AQF). (b) backward-adaptive (AQB).

3.3. LINEAR PREDICTION AND PREDICTIVE CODING.

3.3.1. Basic Configuration.

Predictive Coding or *Differential Coding* systems are based on predicting the current waveform sample by a linear combination of previous samples (a Linear Prediction), and quantizing the prediction error. The most common Differential Coding configuration, called Differential Pulse-Code-Modulation (DPCM), is shown in Figure 3.2. Such a configuration is called a *closed-loop* or *feedback-around-quantizer* structure. In this configuration, the prediction $\hat{x}(n)$ is based on quantized values $y(n)$, rather than unquantized values $x(n)$, thus allowing identical signal reconstruction at both encoder and decoder (in the absence of transmission errors).

The prediction $\hat{x}(n)$ is subtracted from the current sample $x(n)$ to give the prediction error

$$e(n) = x(n) - \hat{x}(n). \tag{3.9}$$

The quantized prediction error

$$u(n) = Q[e(n)] = e(n) + q(n) \tag{3.10}$$

is added back to the prediction $\hat{x}(n)$ to produce a reconstructed sample

$$y(n) = \hat{x}(n) + u(n). \tag{3.11}$$

It can easily be shown that the quantization error $q(n)$ and reconstruction error $\varepsilon(n)$ are equal, i.e.

$$q(n) = e(n) - u(n) = \varepsilon(n) = x(n) - y(n). \tag{3.12}$$

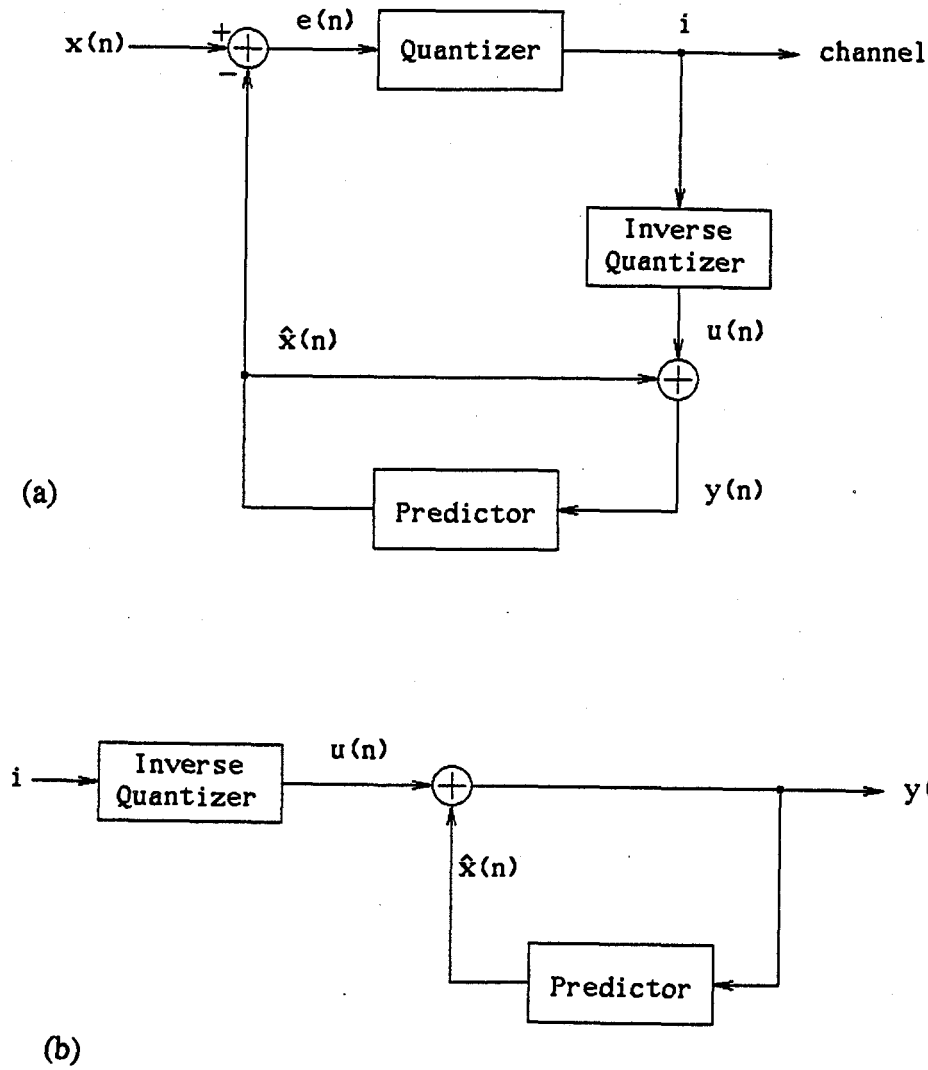

FIGURE 3.2. Differential PCM (a) encoder, and (b) decoder.

3.3.2. Performance Measure: Prediction Gain.

The performance of the DPCM coding scheme is determined by the predictor performance and the quantizer performance. The usual quantizer performance measures have been described in section 3.2.1. The most common predictor performance measure, the prediction gain, is defined in terms of the variance of the input signal σ_x^2, the variance of the prediction error signal σ_e^2.

The *Prediction Gain* is defined as

$$G_P = \sigma_x^2 / \sigma_e^2 \tag{3.13}$$

and represents the relative amount of energy removed from the quantizer input by the predictor. Since the variance of the quantizer output is proportional to the variance of the quantizer input for a given quantizer, the reduction of quantizer input variance leads directly to a reduction of reconstruction error variance[26]. This is expressed in the useful relation

$$SNR_{DPCM} = G_P \, SNR_{Quant} \tag{3.14}$$

3.3.3. Predictor Structure and Optimal Predictors.

The most general form of the linear predictor equation is

$$\hat{x}(n) = \sum_{j=1}^{p} h_j \, y(n-j) + \sum_{j=1}^{z} g_j \, u(n-j) \tag{3.15}$$

in which the prediction is a linear combination of p previous reconstructed samples and z previous quantized prediction error samples. The above relation defines a *pole-zero* predictor, so named since the transfer function

$$P(z) = \frac{Y(z)}{U(z)} = \frac{1 + \sum_{i=1}^{z} g_i \, z^{-i}}{1 - \sum_{i=1}^{p} h_i \, z^{-i}} \tag{3.16}$$

has p non-trivial poles, and z non-trivial zeroes.

The type of predictor to be used in a given application (i.e. the number of poles and zeroes, and the predictor coefficients) depends upon the statistics of the input process $\{x\}$.[6] It is common to model the input process $\{x\}$ as the output of a discrete-time linear filter $F(z)$ which is being excited by a white input sequence $\{w\}$ (sometimes called the *innovations process*). The properties of the filter $F(z)$ determine the statistics of the process $\{x\}$, and therefore determine the optimal predictor $P(z)$ for coding $\{x\}$. It can be shown that $P(z) = F(z)$ is the optimal predictor for coding the process $\{x\}$[26].

In the case where an all-pole predictor of order p is being used to model a stationary process, it can be shown that the optimum predictor coefficients are related to the sample autocorrelation function by the well-known *Wiener-Hopf* equations:

$$r_{xx}(k) = \sum_{j=1}^{p} h_{j,opt}\, r_{xx}(k-j); \qquad k = 1,2,\cdots,p \tag{3.17}$$

or

$$\begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ r_{xx}(3) \\ \cdots \\ r_{xx}(p) \end{bmatrix} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) & \cdots & r_{xx}(p-1) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(p-2) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(p-3) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{xx}(p-1) & r_{xx}(p-2) & r_{xx}(p-3) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} h_{1,opt} \\ h_{2,opt} \\ h_{3,opt} \\ \cdots \\ h_{p,opt} \end{bmatrix} \tag{3.18}$$

or in matrix notation,

$$\mathbf{r}_{xx} = \mathbf{R}_{xx}\mathbf{h}_{opt} \tag{3.19}$$

$$\mathbf{h}_{opt} = \mathbf{R}_{xx}^{-1}\mathbf{r}_{xx} \tag{3.20}$$

There are special recursive algorithms for solving for $\mathbf{h}_{opt}$ which exploit the Toeplitz symmetry of $\mathbf{R}_{xx}$ and the fact that $\mathbf{r}_{xx}$ and $\mathbf{R}_{xx}$ have $p-1$ identical elements[18].

Long-time averaged plots of autocorrelation functions for low-pass filtered (LPF) speech and band-pass filtered (BPF) speech are available in Jayant and Noll [26]. Thus,

one straightforward approach to designing an optimal all-pole predictor would be to simply solve the Weiner-Hopf equations for the optimal predictor coefficients $\{h_{j,opt}\}$ using these long-term average autocorrelation values.

Unfortunately, such an approach assumes that the speech is a stationary process (i.e. that the first- and second-order statistics are constant over time). It is well-known that this is not the case[26], and as a result, fixed predictors based on long-term average speech statistics have a limited prediction gain of about 8 dB for BPF speech, and about 10-11 dB for LPF speech.

3.3.4. Adaptive Prediction.

A considerable performance improvement is possible if the predictor is allowed to adapt to the short-term statistics of the input signal. Predictor Adaptation strategies can generally be classified as either *forward-adaptive* or *backward-adaptive*. Block diagrams of the two strategies are shown in Figure 3.3.

In forward-adaptive prediction, the predictor coefficients are based on the short-term statistics of the unquantized input signal $\{x\}$. Since this signal is not available at the decoder, these raw statistics (or the predictor coefficients) must be quantized and sent to the decoder as *side information*. However, in backward-adaptive prediction, the predictor coefficients are based on the short-term statistics of the quantized signals, which are available at the decoder, and therefore there is no need to send any side information.

At high data rates, when the quantization noise is small, the statistics of the reconstructed signal agree closely with the statistics of the unquantized input signal, thus good

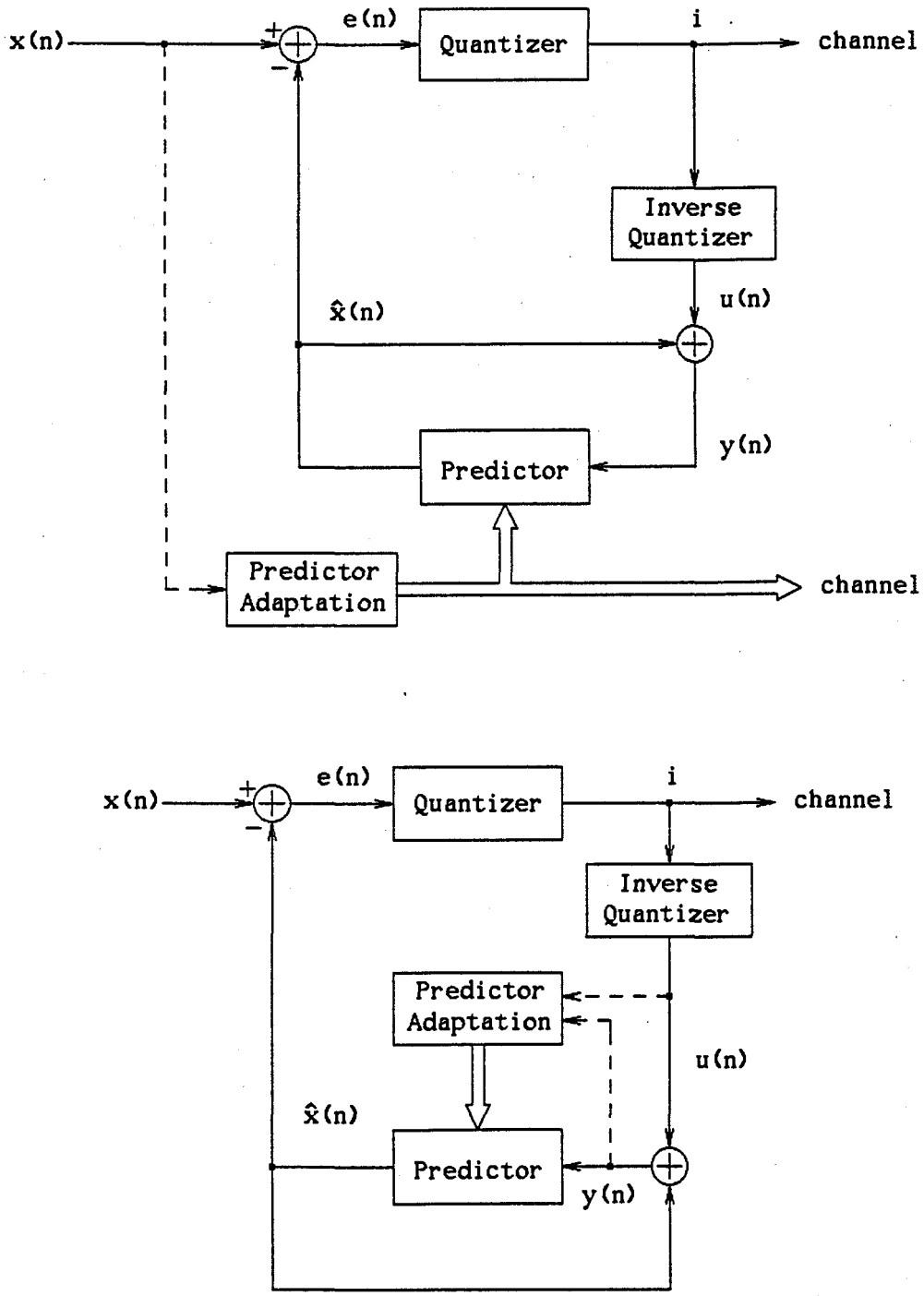

FIGURE 3.3. Adaptive Prediction in DPCM. (a) forward-adaptive prediction (APF). (b) backward-adaptive prediction (APB).

predictor adaptation can be achieved without the need for side information. For this reason, backward-adaptive systems are preferred at high data rates. At rates below 16 kb/s (2 bits/sample), the higher quantization noise results in poor predictor adaptation, and forward prediction is preferred.

There are two well-known approaches for forward adaptation of the predictor: the *Autocorrelation* method and the *Covariance* method. In both methods, the speech signal is blocked into frames, typically of length $M = 128$ samples (16 ms), $\mathbf{R}_{xx}$ is calculated for the frame of samples, and then the Wiener-Hopf equations for the optimal all-pole predictor are solved. The all-pole predictor coefficients are then quantized and transmitted to the decoder as side information.

The difference between the Autocorrelation method and the Covariance methods lies in the procedure used to estimate $\mathbf{R}_{xx}$. In the Autocorrelation method, only the M samples of the frame are used to calculate $\mathbf{R}_{xx}$, while in the Covariance method, $\mathbf{R}_{xx}$ is based on the M samples of the current frame and the last p samples of the previous frame. This is justified by the fact that these p samples are used to predict the first samples of the current frame.

The Autocorrelation method has the advantage that the $\mathbf{R}_{xx}$ matrix is Toeplitz, and therefore the efficient recursive algorithms can be used to solve for $\mathbf{h}_{opt}$. The Covariance method produces an $\mathbf{R}_{xx}$ matrix which is symmetric but not Teoplitz, and therefore this method has a considerable computational disadvantage. The Autocorrelation method also has the advantage that the recursive filter which uses $\mathbf{h}_{opt}$, and hence the DPCM decoder, will also be stable.

In both the Autocorrelation method and the Covariance method for forward-adaptive prediction, the optimal predictor coefficients $\mathbf{h}_{opt}$ cannot be computed until the entire frame has been input. This results in a one-way coding delay on the order of the frame length. This is a significant disadvantage in some applications, including the current work, where coding delay is severely constrained.

Since backward-adaptive prediction does not require transmission of the optimal predictor coefficients as side information, there is the possibility of updating the predictor much more frequently, i.e. after every sample. Such sequentially adaptive backward predictors are usually based on the method of *steepest descent* or *gradient search,* derived below[26].

We define the optimum predictor $\mathbf{h}_{opt}$ as the predictor which produces the minimum mean-squared prediction error σ_e^2. In general, the mean-squared prediction error as a function of $\mathbf{h}$ is a quadratic function of the weights:

$$\sigma_e^2(\mathbf{h}) = \sigma_x^2 - 2\mathbf{h}^T\mathbf{r}_{xx} + \mathbf{h}^T\mathbf{R}_{xx}\mathbf{h}. \tag{3.21}$$

Geometrically, this function defines a p-dimensional paraboloid (hyper-paraboloid), shown in Figure 3.4 for the two-dimensional case.

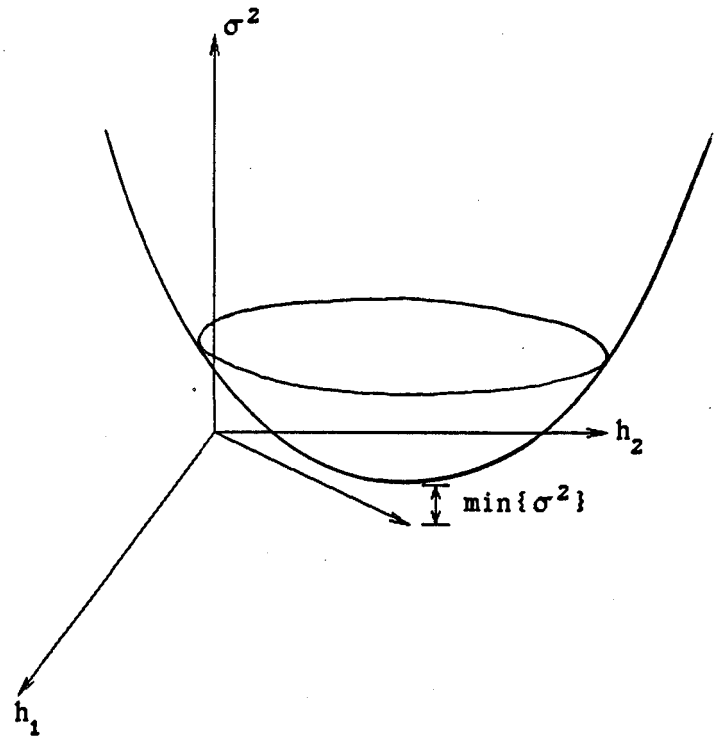

FIGURE 3.4. Variance of Prediction Error as a function of Coefficient Vector of second order.

The bottom of the concave surface defines $\mathbf{h}_{opt}$. Given any $\mathbf{h}$, with its corresponding point on the concave surface, adaptation towards $\mathbf{h}_{opt}$ can be defined as moving towards the bottom of the surface. The optimal direction in which to move is determined by the surface gradient

$$\nabla\sigma_e^2 = \frac{\partial\sigma_e^2(\mathbf{h})}{\partial\mathbf{h}} \tag{3.22}$$

In the method of steepest descent, the vector of predictor coefficients $\mathbf{h}(n)$ at time n is adjusted after each sample, by subtracting a small fraction of an estimate of the gradient:

$$\mathbf{h}(n+1) = \mathbf{h}(n) - \tfrac{1}{2}\,\alpha(n)\,\hat{\nabla}\sigma_e^2(n) \tag{3.23}$$

The parameter $\alpha(n)$ determines the rate of adaptation, and is often simply chosen to be a constant α. Large values of α result in fast adaptation, but greater steady-state estimation noise, since $\mathbf{h}$ will tend to oscillate around $\mathbf{h}_{opt}$. For stationary signals, the condition for convergence is[40]:

$$0 < \alpha < \frac{1}{\lambda_{max}} \tag{3.24}$$

where λ_{max} is the largest eigenvalue of the autocorrelation matrix $\mathbf{R}_{xx}$.

3.3.5. The Least Mean Square Algorithm.

A simple and often-used method for estimating the gradient $\nabla\sigma_e^2$ is the least mean square (LMS) algorithm[40]. In this method, the square of instantaneous quantized difference signal $u^2(n)$ is used as the error criterion, instead of the long-term average of the unquantized difference signal σ_e^2. This method has the advantage that $u^2(n)$ is available at both the transmitter and receiver, and therefore the the decoder predictor can track along with the encoder predictor. Now the estimate of the gradient is

$$\hat{\nabla}\sigma_e^2 \approx \nabla u^2(n) = 2u(n)\nabla u(n) = -2u(n)\mathbf{y}(n) \tag{3.25}$$

Substituting this into equation (3.23) gives the updating algorithm

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \alpha(n)\, u(n)\mathbf{y}(\mathbf{n}) \; ; \quad or \tag{3.26}$$

$$h_j(n+1) = h_j(n) + \alpha_j(n)\, u(n)y(n-j) \; ; \quad j = 1,2,...,p \tag{3.27}$$

For stationary signals, the crosscorrelation term $u(n)\mathbf{y}(n)$ vanishes if prediction is optimal and quantization effects are small. However, in practice, a non-zero quantization error prevents the update term $u(n)\mathbf{y}(n)$ from vanishing, and the mean-squared-error will not be reached. However, by using a small α or a decreasing function $\alpha(n)$, the mean-squared-error may nearly reach the optimal minimum value.

For non-stationary signals such as speech, the centre of the bowl in Figure 3.4 is constantly moving. The gradient coefficient α must be chosen large enough to track the changing input statistics, and yet small enough so that deviations about the intended values are within tolerable limits.

To reduce complexity, many practical speech coding systems use a slightly suboptimal update term, based on polarity crosscorrelations and a constant gradient coefficient. Also, to allow the predictor at the decoder to recover from transmission bit errors, a *leak factor* λ_j is used. The form of the predictor update equation is now:

$$h_j(n+1) = \lambda_j h_j(n) + \alpha_j(n)\, sgn\, u(n)\, sgn\, y(n-j) \; ; \quad j = 1,2,...,p \tag{3.28}$$

Typical values are $\lambda = 0.996$ and $\alpha = 0.001$.

In the case of an all-zero predictor, the update term is proportional to $u(n)\mathbf{u}(n)$, instead of $u(n)\mathbf{y}(n)$[36]. In this case, the all-zero adaptation equations become

$$g_j(n+1) = \lambda_j g_j(n) + \alpha_j(n)\, sgn\, u(n)\, sgn\, u(n-j); \qquad j = 1,2,...,z \tag{3.29}$$

3.3.6. Stability Constraints.

It is possible for an adaptive predictor to adapt to an unstable state in the presence of non-stationarities in the input signal or transmission errors. To ensure that this does not happen, it is common to apply stability constraints to the predictor.

Since the predictor is simply an adaptive filter, we may draw upon the well-known stability constraints in the adaptive filter literature. However, this must be done with care, since the predictor is not an isolated filter; it is an embedded component in a larger system. In this section, the well-known stability constraints for an isolated filter are described. Then the impact of these stability issues within the DPCM configuration are described.

Figure 3.5 shows a general pole-zero filter, of the same form as the predictor discussed in the previous sections.

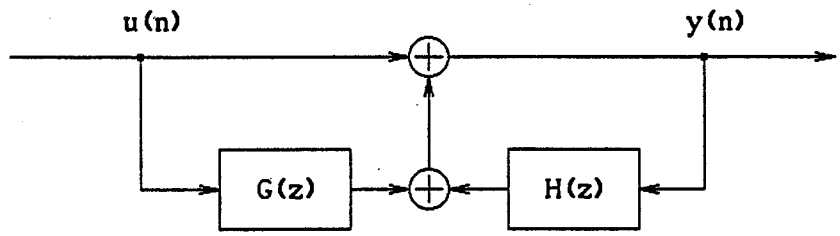

FIGURE 3.5. General pole-zero filter.

The current output of the filter $y(n)$ is a linear combination of the current input $u(n)$, and the previous z inputs and p outputs:

$$y(n) = u(n) + \sum_{j=1}^{z} g_j\, u(n-j) + \sum_{j=1}^{p} h_j\, y(n-j) \tag{3.30}$$

This results in the following transfer function between the output Y(z) and the input U(z):

$$P(z) = \frac{Y(z)}{U(z)} = \frac{1 + \sum_{i=1}^{z} g_i\, z^{-i}}{1 - \sum_{i=1}^{p} h_i\, z^{-i}} = \frac{1 + G(z)}{1 - H(z)} \tag{3.31}$$

For Bounded-In-Bounded-Out or BIBO stability, it is well known that the stability condition for such a filter is that the poles of the filter, or the roots of the denominator polynomial $1 - H(z)$, must lie within the unit circle in the complex plane[5].

While numerical procedures exist for solving for the positions of the poles of a filter given its coefficients, it is highly desirable from a complexity standpoint to have explicit constraints on the filter coefficients themselves, rather than the pole locations. Such a set of constraints is very well-known in the case of a two-pole filter, and is given by[5]:

$$\begin{aligned} h_1 + h_2 &< 1 \\ -h_1 + h_2 &< 1 \\ |h_2| &< 1 \end{aligned} \tag{3.32}$$

The stability region for a third-order filter is given by[6]:

$$\begin{aligned} h_1 + h_2 + h_3 &< 1 \\ -h_1 + h_2 - h_3 &< 1 \\ -h_3(h_3 - h_1) - h_2 &< 1 \\ |h_3| &< 1 \end{aligned} \tag{3.33}$$

A general procedure for deriving the stability constraints for filters of arbitrary order may be based on the Schur-Cohn criterion [32], which gives the number of zeroes of a given polynomial within a circle of a specified radius. While this method will give the desired constraints, it does not necessarily give the constraints in the simplest form for low order filters.

We now apply a similar stability analysis to the more complicated DPCM system. To do this, it is necessary to write the transfer function between the inputs and outputs at both the transmitter and receiver, and then ensure that the roots of the denominator polynomials are inside the unit circle.

The transfer function between the DPCM receiver input and output is

$$P(z) = \frac{Y(z)}{U(z)} = \frac{1 + G(z)}{1 - H(z)} \tag{3.34}$$

and therefore for BIBO stability at the receiver we must ensure that the roots of the polynomial $1 - H(z)$ are within the unit circle.

Neglecting quantization effects, the transfer function between the DPCM transmitter input and and output is:

$$P^{-1}(z) \approx \frac{U(z)}{X(z)} \approx \frac{1 - H(z)}{1 + G(z)} \tag{3.35}$$

and therefore for BIBO stability at the transmitter we must ensure that the roots of the polynomial $1 + G(z)$ are within the unit circle.

Thus for BIBO stability at both the DPCM transmitter and receiver, we require the predictor transfer function $P(z)$ and its inverse $P^{-1}(z)$ to be stable. This is the definition of a *minimum phase* predictor filter.

3.4. THE CCITT 32 kb/s ADPCM ALGORITHM.

3.4.1. Description of CCITT ADPCM Algorithm.

The CCITT 32 kb/s ADPCM Algorithm is an important special case of predictive coding schemes in the context of the present work. The system block diagram is shown in Figure 3.6. The ADPCM algorithm uses a 4-bit backward-adaptive quantizer, called a *dynamic locking quantizer* which stops adapting in the presence of a stationary input[26]. The dynamic locking feature was added to improve the performance for voice-band data signals. Since this consideration is beyond the scope of the current work, this feature will not be described in detail here.

The predictor used in the ADPCM algorithm is a two-pole six-zero backward-adaptive predictor. The adaptation of the all-zero coefficients $g_j(n)$ follows equation (3.29), with λ_j=0.9961 and $\alpha_j(n)$=0.0078125 for all j:

$$g_j(n+1) = 0.9961 g_j(n) + 0.0078125\ sgn\ u(n)\ sgn\ u(n-j) \tag{3.36}$$

A more complicated strategy is used for adapting the all-pole coefficients, in order to provide improved performance on voice-band data signals, and to provide more robust performance in the presence of transmission errors[34]. Here, the adaptation of the all-pole coefficients is based on a polarity correlation of the partial output (zero-based reconstruction term)

$$p(n) = u(n) + \hat{x}_{az}(n) \tag{3.37}$$

where

$$\hat{x}_{az}(n) = \sum_{j=1}^{6} g_j(n)u(n-j) \tag{3.38}$$

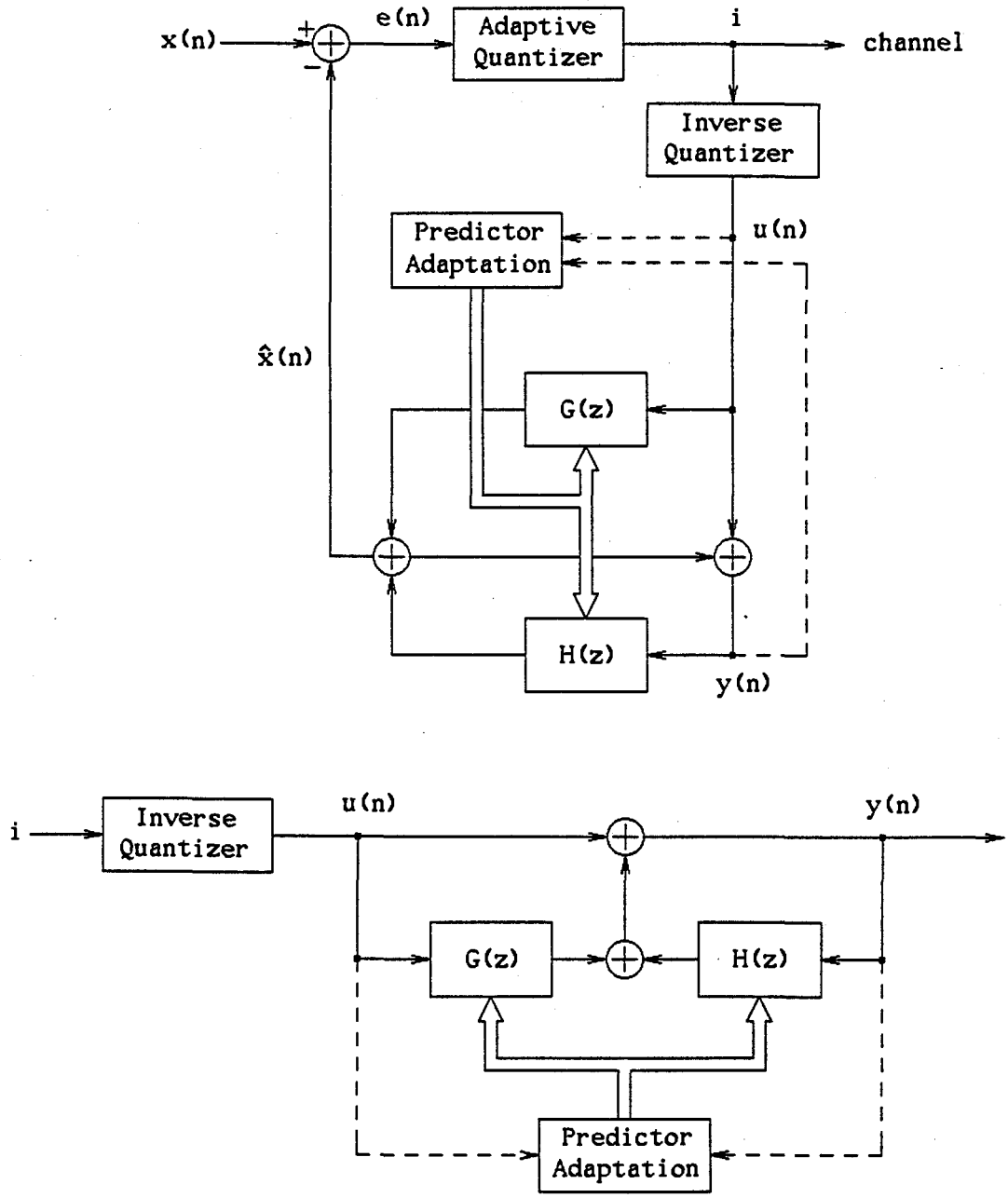

FIGURE 3.6. CCITT ADPCM block diagram.
(a) transmitter and (b) receiver.

The all-pole predictor update equations are

$$h_1(n+1) = 0.9961 h_1(n) + 0.01172\, sgn\, p(n)\, sgn\, p(n-1)$$

$$h_2(n+1) = 0.9922 h_2(n) + 0.0078125\, sgn\, p(n)\, sgn\, p(n-2)$$

$$- 0.0078125\, f[h_1(n)]\, sgn\, p(n)\, sgn\, p(n-1)$$

$$f[h_1(n)] = \begin{cases} 4\, h_1(n) & \text{if } |h_1(n)| \le 0.5 \\ 2\, sgn\, h_1(n) & \text{if } |h_1(n)| > 0.5 \end{cases} \tag{3.39}$$

Conservative constraints are explicitly applied to the all-pole predictor coefficients to ensure stability at the receiver:

$$|h_2| \le 0.75$$

$$|h_1| \le 0.9375 - h_2 \tag{3.40}$$

The fixed-point implementation specified in CCITT G.721[10] imposes an implicit constraint on the all-zero predictor coefficients:

$$|g_i| \le 2 \tag{3.41}$$

However, it should be noted that there are no constraints to ensure that the zeroes of the predictor filter stay inside the unit circle. As described in Section 3.3.6, such constraints are necessary to ensure that the predictor is minimum phase. Stability at the transmitter in the CCITT ADPCM algorithm therefore cannot be guaranteed.

In practice, it appears that the leak factors in the predictor adaptation equations slow the growth of the all-zero prediction coefficients, so the roots of $1 + G(z)$ tend to stay inside the unit circle; in fact, it should be noted that no occurrences of instability have been observed with the specified CCITT algorithm leak factors. However, this is no guarantee of stability. The effects of these omitted stability constraints on predictor performance and the role of the leak factors are investigated further in Section 5.2.

3.4.2. ADPCM Performance.

In the research environment, it is common to test the performance of a new coding algorithm on samples of telephone-bandwidth speech over a wide range of speakers. The objective performance of the coding algorithm is usually expressed in terms of the Signal-to-Noise Ratio (SNR), and occasionally in terms of the Segmental Signal-to-Noise Ratio (SEGSNR). For example, 64 kb/s log companded PCM speech achieves approximately 38 dB SNR over a wide range of input signal levels[26], and a 32-34 dB SEGSNR.

The subjective performance of the coding algorithm is usually expressed in terms of Mean-Opinion-Scores, in which an ensemble of subjects listen to speech recordings and give rankings on a scale of 1 to 5, with 5 representing excellent quality, and 1 representing very bad quality[26]. 64 kb/s log-companded PCM speech achieves a MOS score of about 4.5.

In the interests of repeatability, it is also common to present SNR results with deterministic inputs such as sine waves and narrow-band gaussian noise. Acceptance tests for CCITT standards are often expressed in terms of compliance with certain performance templates using deterministic waveforms[8].

The CCITT 32 kb/s ADPCM algorithm has been extensively tested. With speech, its SNR performance is typically 25-30 dB. It has a MOS score of 4.0 after one encoding/decoding operation, and a MOS score of 3.5 after 4 asynchronous transcodings[26]. It also meets the dynamic range and SNR requirements for certain important deterministic waveforms[8].

The performance of ADPCM degrades significantly at rates below 32 kb/s. The SNR values for ADPCM-coded speech at 16 kb/s are in the range of 13-14 dB, and MOS scores for ADPCM-coded speech at 16 kb/s are well below 3.0[24]. This level of performance is unacceptable for telecommunications applications.

The focus of the present work is on improving the performance of the basic ADPCM system, by improving the quantizer performance, and making the quantizer noise less perceptible. The latter issue is addressed by postfiltering the reconstructed

signal, and is described in Section 3.6. The issue of improving the quantizer performance is addressed by an investigation of Vector Quantization in the next section.

3.5. Vector Quantization.

A vector quantizer (VQ) is a system for mapping a sequence of k-dimensional vectors into a digital sequence. The origins of vector quantization may be traced back to Shannon's source coding theory[38], which states that memoryless vector quantization can achieve nearly optimal rate-distortion performance at very large vector dimensions.

The discussion of VQ follows the discussion of scalar quantization in section 3.2; scalar quantization is in fact a special case of vector quantization.

In waveform vector quantization of speech, the speech waveform is sampled and blocked into groups of k consecutive samples (k-dimensional vectors) $\mathbf{x}_n = (x_{n_0}, x_{n_1}, \cdots, x_{n_{k-1}})$ which are then quantized simultaneously as a single unit. The quantization procedure may be regarded as finding a codevector $\mathbf{y}_n = VQ(\mathbf{x}_n)$ in a codebook with L codevectors which is nearest to the input vector $\mathbf{x}_n$, in the sense that it minimizes some distortion measure $D(\mathbf{x}_n,\mathbf{y}_n)$. In the present work, the distortion measure is usually the mean squared error

$$D(\mathbf{x}_n,\mathbf{y}_n) = \|\mathbf{x}_n - \mathbf{y}_n\|^2 = \sum_{j=1}^{k}(x_{n_j} - y_{n_j})^2 \tag{3.42}$$

i.e. the square of the Euclidean distance between the vectors.

A common geometric interpretation for the two-dimensional case is shown in Figure 3.7, in which in the input vector $\mathbf{x}_n$ and the codevectors are represented by points in the two-dimensional space. The quantization procedure is simply that of choosing the nearest codevector point.

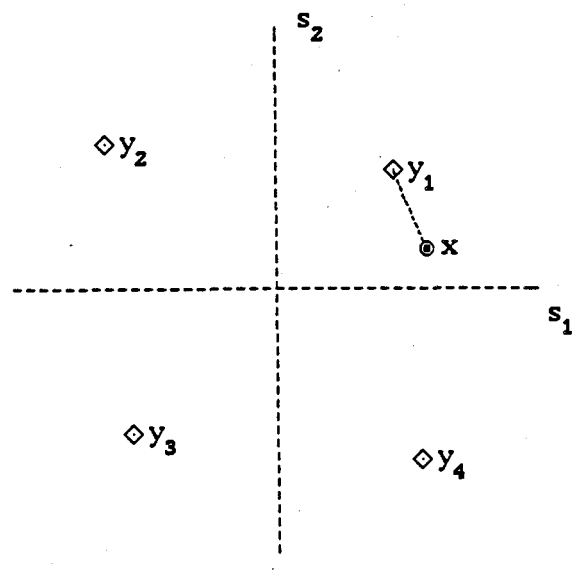

FIGURE 3.7. Geometric Interpretation of VQ for k=2.

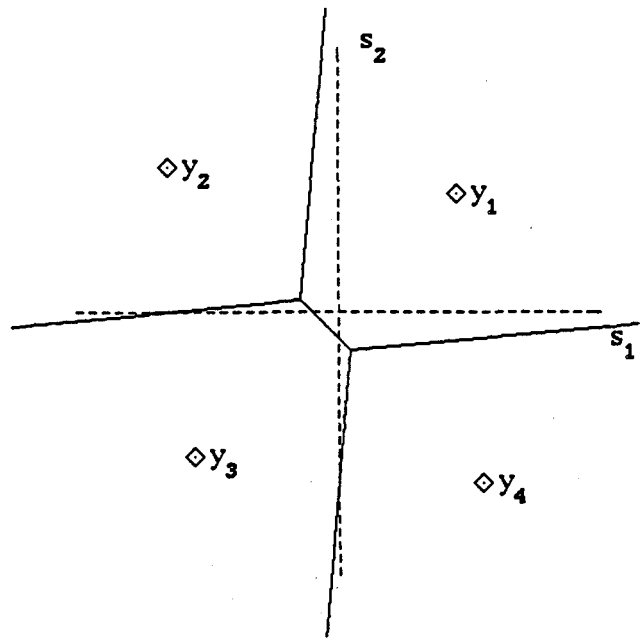

FIGURE 3.8. Voronoi Cells for k=2 example.

It is possible to partition the space into L regions S_i according to a nearest-neighbour rule, where each region is associated with a codevector, and the resulting regions are called Voronoi regions or Voronoi cells[20]. The set of cell boundaries is called a Voronoi partition. The Voronoi cells for the above codevectors are shown in Figure 3.8. It can be shown that the centroids of the Voronoi regions are the optimum codevectors for that Voronoi partition, and in fact this is the basis for the codebook optimization procedure to be described in section 3.5.1.

Usually a binary word i is used to represent each of the possible codevectors. The inverse quantizer at the decoder then retrieves the the quantized value $\mathbf{y}_n$ from an identical codebook.

The quantization noise $\mathbf{q}_n$ introduced by this quantization procedure is the difference between the input vector $\mathbf{x}_n$ and the quantized vector $\mathbf{y}_n$:

$$\mathbf{q}_n = \mathbf{x}_n - \mathbf{y}_n \tag{3.43}$$

The number of codebook elements L used to represent the k-dimensional vector and the transmission rate R are related by:

$$L = 2^{kR} \tag{3.44}$$

For example, a VQ system sampling at 8 kHz and quantizing at R=2 bits/sample in vector dimension k=3 has $L=2^{3.2}=64$ codevectors in the codebook, and has a bit rate of $(8kHz)(2bits/sample)=16kb/s$.

3.5.1. Vector Quantizer Design.

The design of the Vector Quantizer follows an iterative design procedure which is a vector generalization of the Lloyd-Max algorithm for scalar quantizer. This vector generalization was first described by Linde, Buzo and Gray[30], and is now well-known as the LBG algorithm.

In the LBG algorithm, a long training sequence from a wide group of speakers is used to optimize the codebook. To tell if the training sequence was long enough, the trained codebook should be tested on out-of-training-sequence data; if the performance is close to the in-sequence performance, we can have some confidence that it will give roughly the same performance in the future[21]. If the training and test performance differ significantly, then probably the training sequence is not long enough.

Linde, Buzo and Gray developed a codebook optimization procedure which minimizes the average value of the encoding distortion, based on the following observations:

1. The optimum codevector $\mathbf{y}_i$ for a given Voronoi cell S_i should be chosen to minimize the distortion corresponding to that cell:

$$E[D(\mathbf{x},\mathbf{y}_j) \mid x \in S_i] \leq E[D(\mathbf{x},\mathbf{u}) \mid x \in S_i] \ , \quad any\ \mathbf{u} \in \mathbf{R}^k \tag{3.45}$$

2. For a given set of codevectors $\{\mathbf{y}_i\}$, the optimal partition $\{S_i\}$ is defined by the nearest neighbour rule:

$$\mathbf{x} \in S_i \ \ iff \ \ D(\mathbf{x},\mathbf{y}_j) \leq D(\mathbf{x},\mathbf{y}_m) \ , \quad m = 1,2,\cdots,L \tag{3.46}$$

It can be shown that, if the distortion function is the mean-squared-error, the first condition is equivalent to stating that the optimum codevector $\mathbf{y}_i$ within each partition S_i is simply the centroid of the input vectors which fall in that partition.

The LBG algorithm is thus:

1. GIVEN: a training sequence $\{\mathbf{x}\}$ and an initial codebook $\{\mathbf{y}_i\}$.
2. Encode the training sequence using the current codebook.
3. For each cell, move the codevector to the new centroid of all the input vectors which fell into that cell.
4. Repeat steps 2 and 3 until average distortion is acceptable, or is not changing with further iterations.

Finally, the new codebook should be tested on out-of-sequence data to ensure that training sequence was long enough. To ensure fairness in comparing various algorithms

based on Vector quantization, this work emphasizes out-of-sequence test results.

This method will generally converge to a locally optimum codebook, and while convergence to a globally optimum codebook is not guaranteed, a good choice of initial codebook will usually result in a good final codebook[21].

There are several ways to generate initial codebooks. The simplest method is to simply choose a codebook with L random codevectors, or to choose L vectors at random from the training sequence. However, one of the most-often used methods of generating the initial codebook is the "splitting" technique. In this method, the initial codebook contains only one codevector, located at the centroid of the entire training sequence. This single codevector is then split to form two codevectors, i.e. a second codevector is created by adding some small perturbation to the first codevector. The algorithm is then run on the new 2-codevector codebook, and the resulting codevectors are then split. This procedure continues until the final codebook size is reached.

It is possible under some circumstances that no input vectors will map into a particular cell. This "empty-cell" problem and a procedure for dealing with it was first described by Cuperman[15]. In this modified LBG algorithm, the number of input vectors mapped into each cell is checked; if an empty cell is found, the corresponding codevector is deleted and a new codevector is defined by splitting the codevector with the highest distortion.

3.5.2. Performance of Vector Quantizers

Early application of vector quantization to speech waveform coding began with Gray and Linde[22], and Abut, et. al.[1], using the LBG algorithm to design vector quantizers for Gauss-Markov sources. The vector quantizer results were found to be significantly better than the scalar predictive coders which were tested.

In addition, Abut et al. tested their VQ on speech. Their best out-of-sequence results at 2 bits/sample gave SNR = 13 dB, which is about 6 dB better than the optimum scalar quantizer, and comparable to quoted figures for scalar ADPCM (13.5 dB).

It should be noted that the performance gain of vector quantizers over scalar quantization is strongly related to the characteristics of the input signal for which the quantizers are designed. In practice, a significant part (but not all) of the vector quantizer gain is based on exploiting the adjacent-sample correlation of the source, as in the above examples on speech and Gauss-Markov sources. In the case of a memoryless source, the vector quantizer gain increases very slowly with vector dimension, and comes at a great increase in complexity. For example, in the classic paper by Linde, Buzo and Gray, vector quantization with k=6 at 1 bit/sample is shown to give better performance than scalar quantization for a memoryless Gaussian source, but closer inspection reveals that the performance gain in this case was 0.54 dB, at the cost of a factor of 32 increase in complexity!

3.5.3. Complexity Reduction for Practical Applications of Vector Quantization.

The main disadvantage of vector quantization is the exponential increase of complexity with vector dimension, associated with performing the codebook search. For a vector quantizer of vector dimension k operating at a bit rate of R bits/sample, the computational complexity is proportional to 2^{kR} and the memory requirements are proportional to $k2^{kR}$. In practice, this prevents the use of high-dimensionality vector quantization which is necessary to approach the asymptotic performance promised by the Shannon coding theorem.

There are two common approaches to this problem:

1. A constrained codebook structure may be used to simplify the codebook search, allowing a higher dimension codebook to be used for a given level of computational complexity. These structured codebooks are generally suboptimal, in the sense that their performance is worse than an unstructured codebook at the same rate and dimension. However, when compared on the basis of equal complexity, the structured codebooks tend to perform better than the unstructured codebooks. Examples of this approach include tree-search vector

quantization[7], and shape-gain VQ[37].

2. Vector quantization may be combined with other data compression techniques, to improve its performance at a given vector dimension and complexity level.

The prime example of the second technique is Vector Predictive Coding, first described by Cuperman and Gersho[17]. This algorithm is a vector generalization of DPCM, in which the residual of a all-pole vector predictor is vector quantized. The block diagram of the system appears in Figure 3.9.

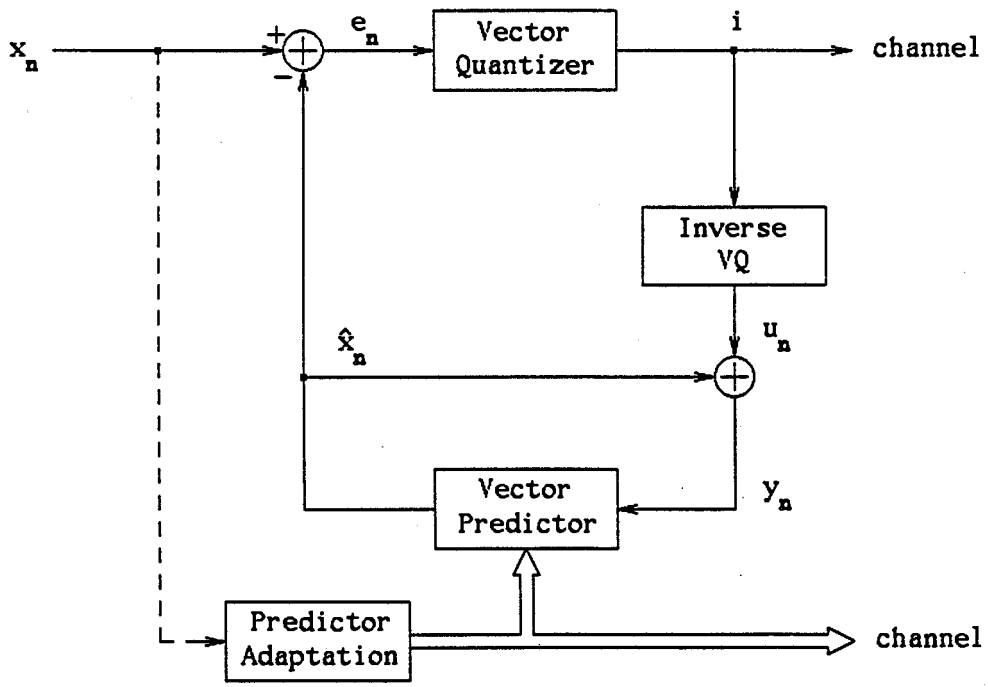

FIGURE 3.9. Vector Predictive Coding block diagram.

The block diagram is clearly a vector generalization of the DPCM shown in Figure 3.2. The predictor adaptation was based on a frame classifier which selected one of three

vector predictors, each optimized for a particular signal type, and the frame classification index was transmitted as side information.

Two approaches to the joint optimization of the vector quantizer and the vector predictor were considered. In the *open-loop* approach, the predictor is optimized on the unquantized training data, and then the prediction residuals are computed once and used to train the vector quantizer. This is a valid approach when the quantization noise is small i.e. at high transmission rates.

In the *closed-loop* approach, the predictor is optimized on the basis of quantized training data after each codebook optimization. This approach was found to give a 1-2 dB improvement on in-sequence data. While convergence cannot be guaranteed in this case, no convergence problems were observed.

The algorithm was found to give approximately SNR = 20.5 dB at vector dimension k=5 on in-sequence data, and approximately 17.0 dB on out-of-sequence data. These results compared very favourably with other coding algorithms and prompted considerable further research in predictive quantization techniques.

One of the weaknesses reported in the vector predictive coding algorithm was the low prediction gain of the vector predictor, due to the decreasing autocorrelation function of speech with increasing lag. The present work attempts to address this problem by combining a vector quantizer with a scalar predictor. This combination may be achieved using the analysis-by-synthesis technique, in which a codebook excitation vector is used to excite a scalar linear filter. The analysis-by-synthesis technique is discussed in Section 3.6.

3.5.4. Gain-Adaptive Vector Quantization.

Gain-Adaptive Vector Quantization was first described by Chen and Gersho[13], and is a generalization of adaptive scalar quantization to the vector case. In both the scalar and vector cases, the quantizer is adapted in response to a short-term estimate of the input signal standard deviation $\hat{\sigma}_n$. This may be achieved by actually scaling all the codevector elements by a gain factor $\hat{\sigma}_n$, however it is preferable from a complexity standpoint to divide the input to the quantizer by the estimated gain. Forward-gain-adaptive VQ and backward-gain-adaptive VQ block diagrams are shown in Figure 3.10.

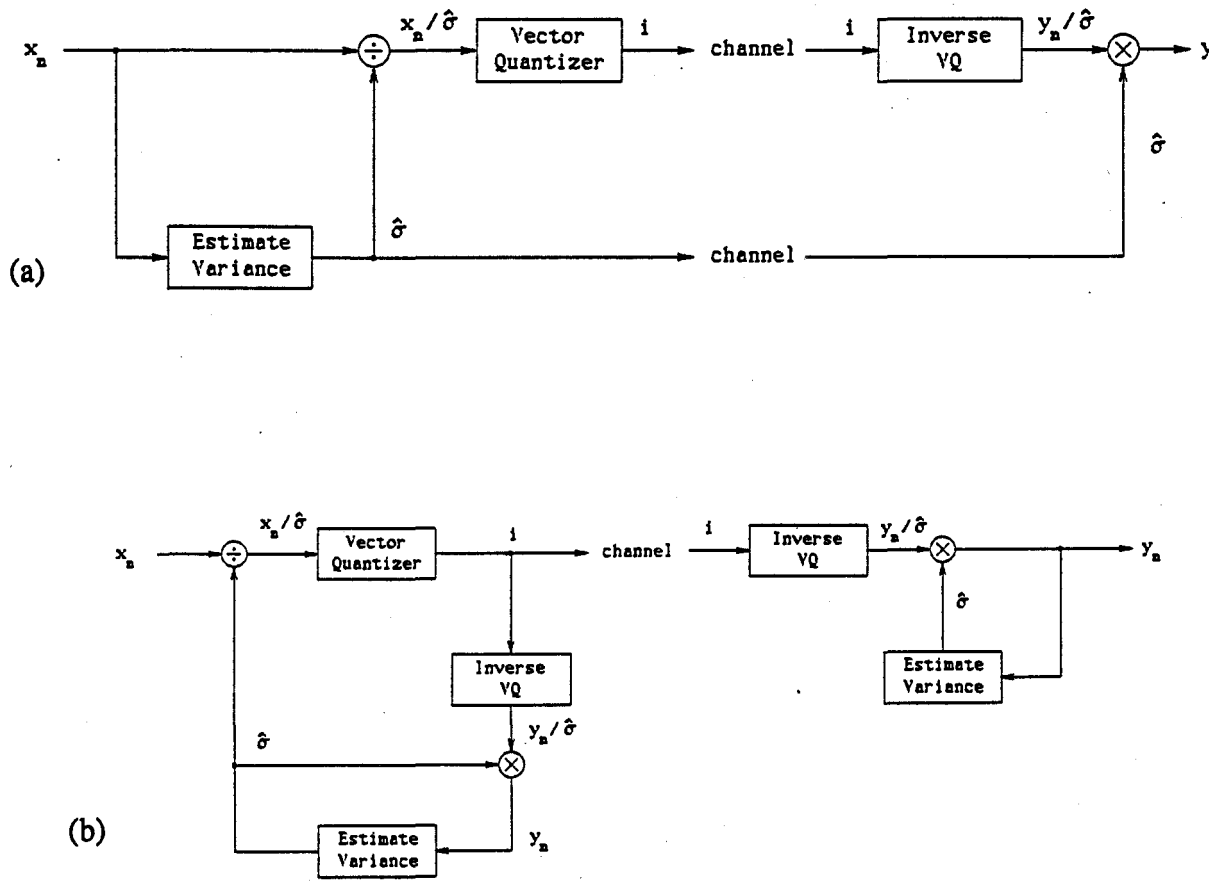

FIGURE 3.10. Gain-Adaptive Vector Quantization. (a) forward gain adaptation. (b) backward gain adaptation.

It is necessary to modify the codebook design algorithm to account for the gain-normalization. It may be shown [13] that the optimal code vector in the Voronoi cell S_j is the weighted centroid of S_j:

$$y_i = \frac{E\,[\hat{\sigma}_n^2\, \mathbf{y}_n \mid S_j\,]}{E\,[\hat{\sigma}_n^2 \mid S_j\,]} \tag{3.47}$$

Gain-Adaptive vector quantization was found to be superior to non-gain-adaptive VQ in subjective performance, SNR, SEGSNR, and performance on inputs with a wide dynamic range, with only a very small increase in complexity.

3.6. Analysis-by-Synthesis Methods.

The first coding system to use the analysis-by-synthesis technique was called Code Excited Linear Prediction (CELP), and is closely related to Adaptive Predictive Coding (APC) . This method was originally proposed by B. S. Atal and M. R. Schroeder for low rate applications (below 8 kb/s or 1 bit/sample)[3].

The basic CELP analysis-by-synthesis configuration is shown in Figure 3.11. In this configuration, a trial codevector **u** is selected from an innovations codebook. The samples $u(n)$ are filtered by a synthesis filter $H(z)$ to produce the trial reconstructed speech samples $y(n)$. The synthesized speech samples are subtracted from the input speech samples $x(n)$ to produce an error sample $q(n)$, which is then filtered by a perceptual weighting filter $W(z)$ to produce a weighted error sample $q_f(n)$. The codevector which results in the smallest weighted mean-square error is selected, and its index is transmitted to the receiver.

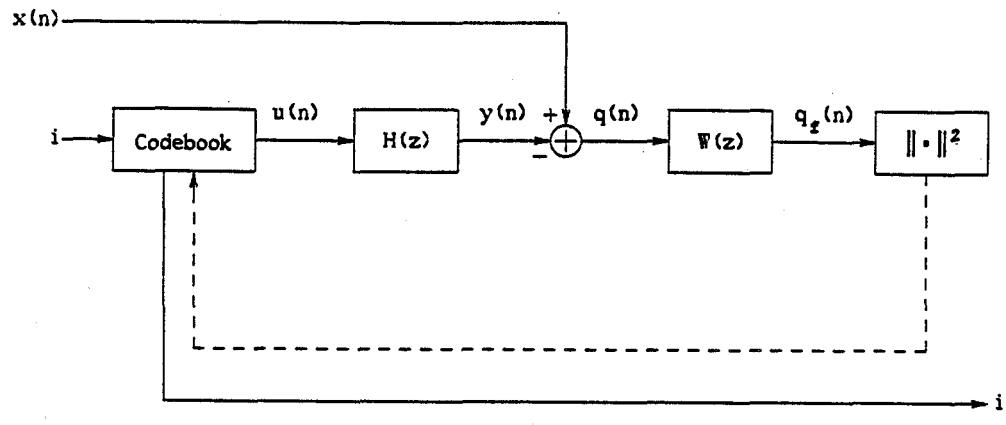

FIGURE 3.11. Code-Excited Linear Prediction.

In this system, the synthesis filter consists of a short-delay predictor and a long-delay or pitch predictor. Both of these predictors are forward-adaptive, requiring the transmission of predictor parameters as side information. Sometimes, a gain-factor is used to scale the innovations sequence before the synthesis filter.

The CELP technique promised good speech quality at very low rates, using an innovations codebook of size 1024 codevectors, each of length 40 samples. Several techniques were used to generate the codevectors, including random selection of unit-variance gaussian numbers. This was justified since it was found that the probability distribution function of the prediction error samples after both short delay and long delay predictions is nearly Gaussian [3]. Of course, it is possible to use the standard Vector Quantization LBG algorithms techniques to train the codebook, and this has been done

by Chan and Cuperman[11].

The main disadvantage to the analysis-by-synthesis configuration is that it leads to very high complexity. This is because each codevector must be filtered through the synthesis filter before the optimum innovations codevector may be selected. The complexity of the basic CELP configuration was estimated at over 500 million float-point- operations per second (500 Mflops), which is well beyond the reach of currently available DSP chips.

Techniques to reduce the complexity of the CELP coding process include using a structured codebook, and pre-computing the Zero-input-response of the synthesis filter. The latter technique was first described in by Chen and Gersho[12] in the context of their Vector APC 9.6 kb/s codec design, and is used in the present work.

It should be noted that the present work was approached from the point of view of introducing a vector quantizer into the backward-adaptive ADPCM configuration in an analysis-by-synthesis configuration. The resulting configuration may equivalently be regarded as a CELP configuration in which the predictor is backward-adaptive.

3.7. Adaptive Noise-Shaping and Post-Filtering.

Noise-shaping and Post-filtering were developed to improve the subjective quality of coded speech, by exploiting the fact that noise which has the same spectral shape as speech tends to be perceived by the human ear as speech[27].

This technique has been used in many speech coding algorithms, including APC[4], CELP[3], and more recently ADPCM[25,27]. All-pole noise-shaping is used at the transmitter to shape the spectrum of the quantization noise, by weighting the reconstruction error. All-zero or pole-zero post-filtering is used at the receiver to filter the reconstructed speech, so that noise is emphasized in the spectral regions where is signal is strong, and suppressed where the signal is weak. Often, the noise-shaping and post-filtering filters are made adaptive by using a scaled version of the adaptive predictor filter.

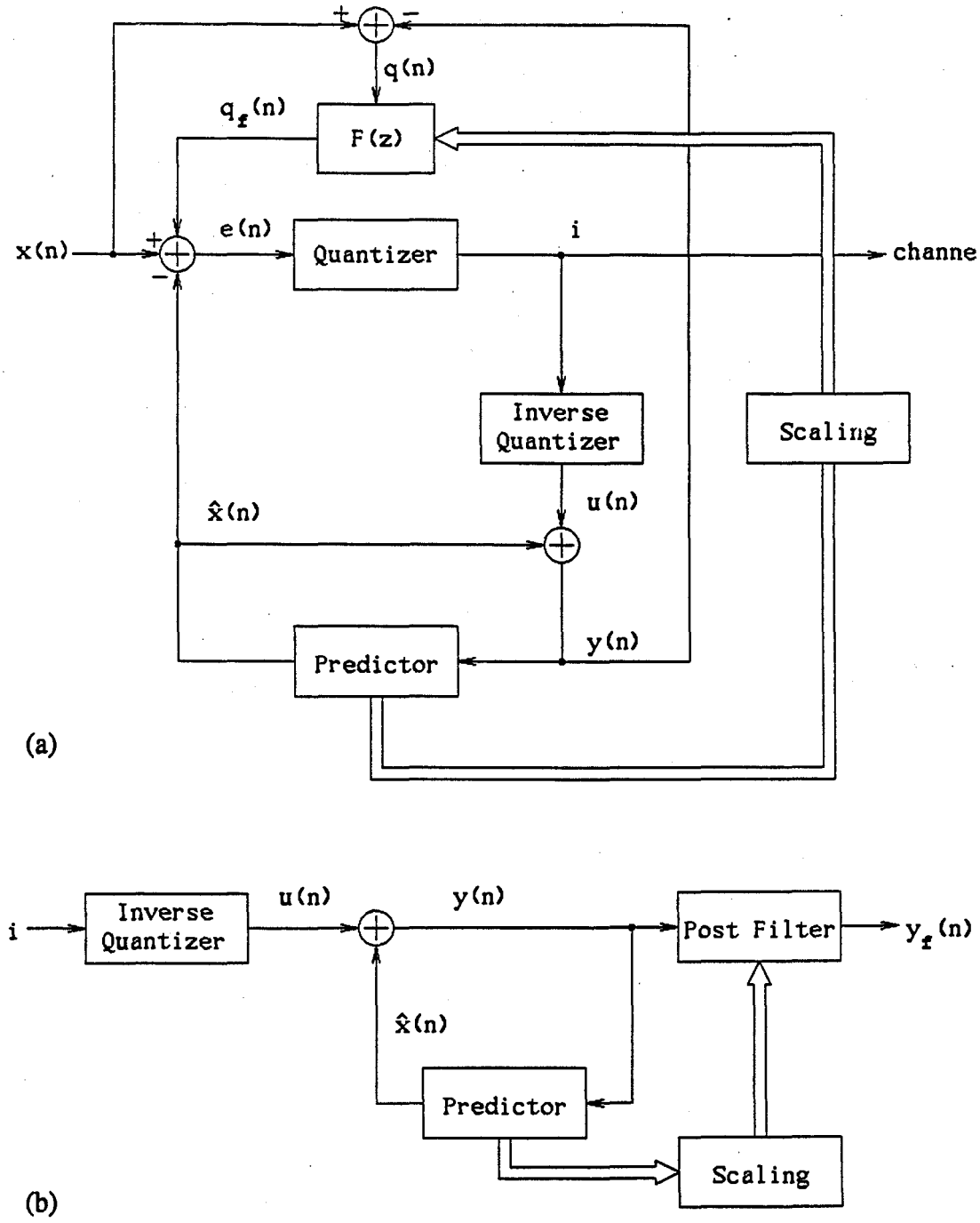

FIGURE 3.12. DPCM with (a) Noise-Shaping at the transmitter and (b) Post-Filtering at the receiver.

The basic noise-shaping and post-filtering configurations are shown in the ADPCM configuration in Figure 3.12. At the transmitter, the reconstruction error $q(n)$ is filtered by an error feedback filter $F(z)$ to produce a filtered reconstruction error sample $q_f(n)$. This filtered error sample is then added to the prediction residual before quantization.

Given the all-pole part of the predictor $H(z)$ as defined in equation (3.3.1), where

$$H(z) = \sum_{j=1}^{p} h_j z^{-j} \tag{3.48}$$

it is common to use a scaled noise-shaping filter

$$F(z) = H'(z) = \sum_{j=1}^{p} h'_j z^{-j} = \sum_{j=1}^{p} h'_j \alpha^j z^{-j} \tag{3.49}$$

where $0 \leq \alpha \leq 1$. This results in a shaped reconstruction error $Q'(z)$ which is related to the unshaped reconstruction error $Q(z)$ by the all-pole transfer function

$$\frac{Q'(z)}{Q(z)} = \frac{1}{1 - H'(z)} \tag{3.50}$$

If $\alpha=0$, there is no noise-shaping. If $\alpha=1$, the poles of the noise spectrum tend to mimic the poles of the input speech spectrum. An intermediate value of α, typically 0.5 is usually found to provide the best subjective results.

For post-filtering, the all-zero or pole-zero predictor coefficients are scaled similarly to achieve a post-filter, which may then be used to filter the reconstructed speech. Previous research [25] has found a subjective preference for all-pole noise-shaping in combination with all-zero post-filtering.

It should be noted that, while these techniques achieve a considerable reduction in the perceived noise, the SNR actually decreases. One method of estimating the "improvement" by post-filtering is to measure the mean-squared-error between the post-filtered signal and same input signal which has been filtered with the same post-filter. Unfortunately, this is not a very meaningful measurement since it does not account for the distortion ("muffling") of the speech associated with too much postfiltering. For this reason,

this method of estimating postfilter performance is not commonly used, and generally, postfilter parameters are determined on the basis of subjective preference.

4. VECTOR ADPCM FOR 16 kb/s SPEECH WAVEFORM CODING.

4.1. INTRODUCTION.

The present work addresses the problem of low delay, medium complexity, high quality speech waveform coding at 16 kb/s. It has been shown in Chapter 3 that scalar Linear Predictive coding schemes such as ADPCM are very effective at high data rates such as 32 kb/s, but their performance degrades significantly at 16 kb/s, due to excessive quantization noise at only 2 bits/sample. Vector Quantization has been shown to provide a significant performance improvement over scalar quantization, and therefore the combination of a scalar linear predictor and a vector quantizer appears to be a promising avenue for investigation.

The CCITT 32 kb/s ADPCM algorithm contains a 2-pole, 6-zero predictor, which is known to have low complexity and good performance on speech and voice-band-data signals, with and without transmission errors. For this reason, the "CCITT predictor" is used as a starting point in the present work, to be combined with a Vector Quantizer. Variations of this predictor are also considered.

The Analysis-by-Synthesis configuration used in CELP is found to be suitable for combining the vector quantizer and the backward-adaptive scalar linear predictor. The combination of the Vector Quantizer and the backward-adaptive scalar linear predictor in the Analysis-by-Synthesis configuration constitutes the basic Vector ADPCM solution.

The basic Vector ADPCM solution is found to have good performance but very high complexity even at low vector dimensions such as 4. For this reason, considerable attention is paid to complexity reduction techniques, particularly in reducing the number of computations in the exhaustive codebook search for the optimum codevector. With negligible loss in performance, it is possible to reduce the complexity by up to a factor of 3 simply by precomputation of key parameters before each search through the codebook, or by periodic update of slowly varying parameters.

A considerable performance improvement is possible if the vector quantizer is made gain-adaptive, i.e. the prediction residual is normalized before being vector quantized. This also allows a fair comparison between scalar ADPCM (which includes an adaptive scalar quantizer) and Vector ADPCM. A subjective improvement is also realized by adaptively postfiltering the reconstructed speech.

In Section 4.2, the basic Vector ADPCM configuration is described, including the Vector Quantizer and the pole-zero linear predictor in an Analysis-by-Synthesis configuration. The issue of complexity reduction is addressed in Section 4.3. Section 4.4 deals with variations of the predictor which may provide a performance improvement over the CCITT predictor. Adaptive postfiltering is discussed in Section 4.5. Finally, in Section 4.6, Gain-Adaptive Vector Quantization is introduced into the Vector ADPCM system. Simulation results and complexity estimates are presented in Chapter 5.

4.2. BASIC CONFIGURATION

Figure 4.1 shows a block diagram of the Analysis-by-Synthesis (A-S) configuration, containing a Vector Quantizer and backward-adaptive pole-zero Scalar Predictor. The transmitter configuration is shown in Figure 4.1(a), and the receiver configuration including Postfiltering is shown in Figure 4.1(b).

The A-S configuration is necessary to allow the Vector Quantizer (VQ) and Scalar Predictor to operate together, since the VQ introduces a block delay in the encoding process, while the Scalar Predictor requires sample-by-sample update of the quantized prediction residual. The nearest-neighbor codebook search proceeds as follows: for a trial codevector i, the codevector elements $u(n,i)$ are processed through the predictor filter to produce the predicted samples $\hat{x}(n,i)$. The predictor equation is

$$\hat{x}(n,i) = \sum_{j=1}^{p} h_j(n,i)y(n-j,i) + \sum_{j=1}^{z} g_j(n,i)u(n-j,i) \qquad (4.1)$$

where $y(n-j,i)$ and $u(n-j,i)$ may not depend on i if the index $(n-j)$ refers to samples of a

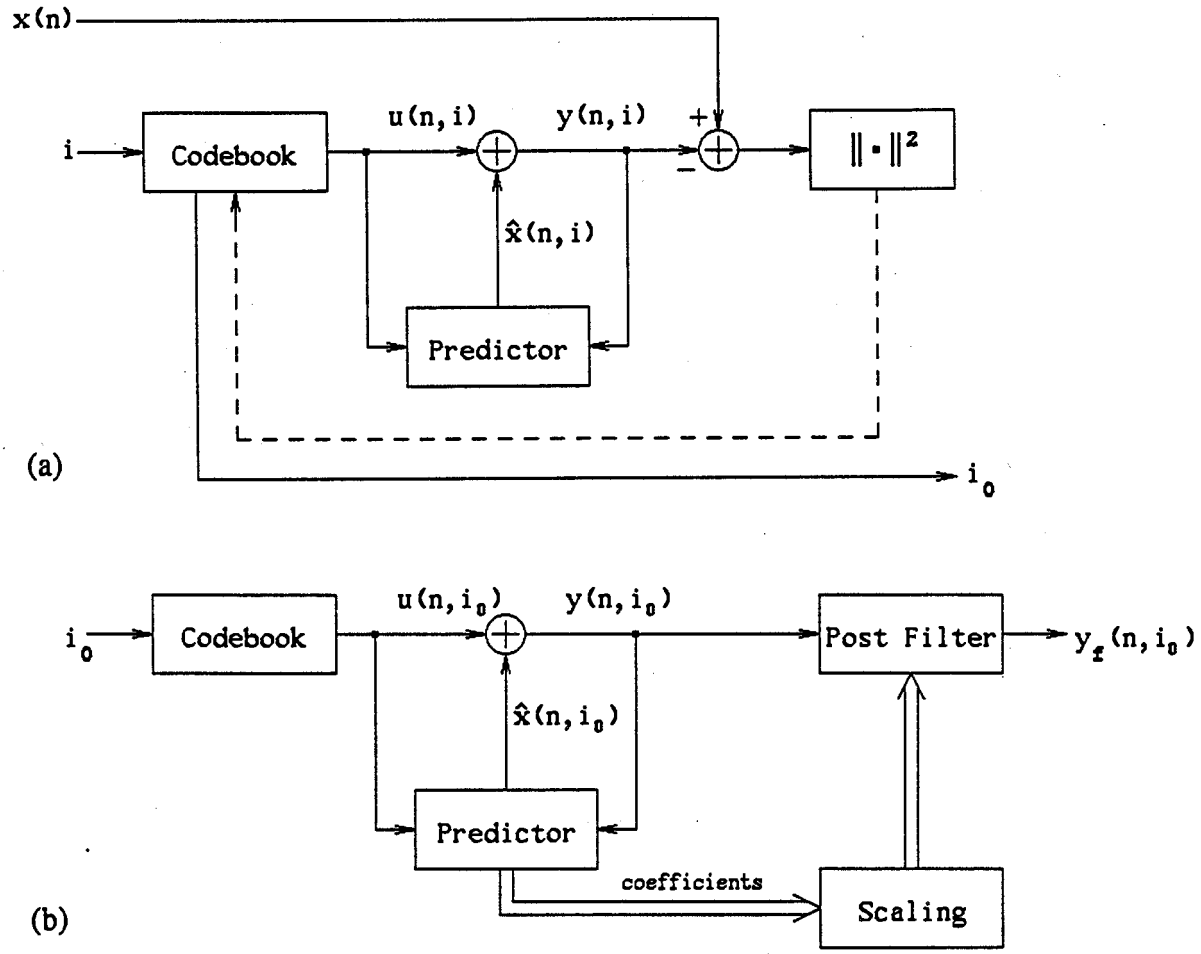

FIGURE 4.1. Vector ADPCM. (a) Transmitter. (b) Receiver with Postfiltering.

previous vector. This fact can be exploited to reduce computational load by precomputing the component due only to the previous vectors (the Zero-Input-Response), as described below.

The reconstructed samples are generated by adding the predicted samples to the codevector elements:

$$y(n,i) = \hat{x}(n,i) + u(n,i) \tag{4.2}$$

and the squared reconstruction error for the codevector is

$$D(i) = \sum_{n=n_0}^{n_0+k-1} (x(n) - y(n,i))^2 \tag{4.3}$$

where k is the vector dimension and n_0 is the sample number of the first sample in the vector. This procedure is repeated for i=1,2,...,N, where N is the number of codevectors in the codebook, and the codevector which minimizes the squared reconstruction error is selected:

$$i_0 = ARGMIN_i [D(i)]. \tag{4.4}$$

In the codebook training phase, the prediction residuals

$$e(n,i_0) = x(n) - \hat{x}(n,i_0) \tag{4.5}$$

are grouped into vectors of the form $\left[e(n,i_0)\right]_{n=n_0}^{n_0+k-1}$ and clustered using the LBG algorithm[30].

The predictor is adapted using the CCITT predictor sign algorithm adaptation equations (3.36-41), or with variations as described in Section 4.4.

4.3. COMPLEXITY REDUCTION

Three methods are used to reduce the number of computations required by the A-S technique. The first step in complexity reduction is based on the fact that the predictor coefficients $h_j(n,i)$ and $g_j(n,i)$ in equation (4.1) change slowly, and thus these coefficients

need not be updated during the optimal codevector selection. Hence, the index i in $h_j(n,i)$ and $g_j(n,i)$ may be dropped. This results in a negligible performance degradation.

The second complexity reduction method exploits the fact that the output of the predictor filter consists of two components[12]. The Zero-Input-Response $\hat{x}_{ZIR}(n)$ is the filter output due only to the previous vectors. The Zero-State-Response $\hat{x}_{ZSR}(n,i)$ is the filter output due only to the the trial codevector i, such that

$$\hat{x}(n,i) = \hat{x}_{ZIR}(n) + \hat{x}_{ZSR}(n,i). \tag{4.6}$$

For each search through the codebook, the ZIR may be precomputed and subtracted from the input samples, to produce the partial input sample

$$x^*(n) = x(n) - \hat{x}_{ZIR}(n). \tag{4.7}$$

The partially reconstructed speech sample

$$y_{ZSR}(n,i) = u(n,i) + \hat{x}_{ZSR}(n,i) \tag{4.8}$$

is then subtracted from the partial input sample $x^*(n)$ to produce the reconstruction error

$$x(n) - y(n,i) = x^*(n) - y_{ZSR}(n,i) \tag{4.9}$$

The resulting configuration is shown in Figure 4.2(a).

The third complexity reduction method is based on the following observation: the filter coefficients change slowly, and thus the partially reconstructed samples $y_{ZSR}(n,i)$ for a given codevector also change slowly. Therefore, the $y_{ZSR}(n,i)$ filter outputs may be periodically computed and stored in a new ZSR codebook, resulting in the configuration shown in Figure 4.2(b). The use of the ZSR codebook was described by Chen and Gersho [12]. This results in a substantial reduction in computational load with only a slight performance degradation (see Chapter 5 for performance results).

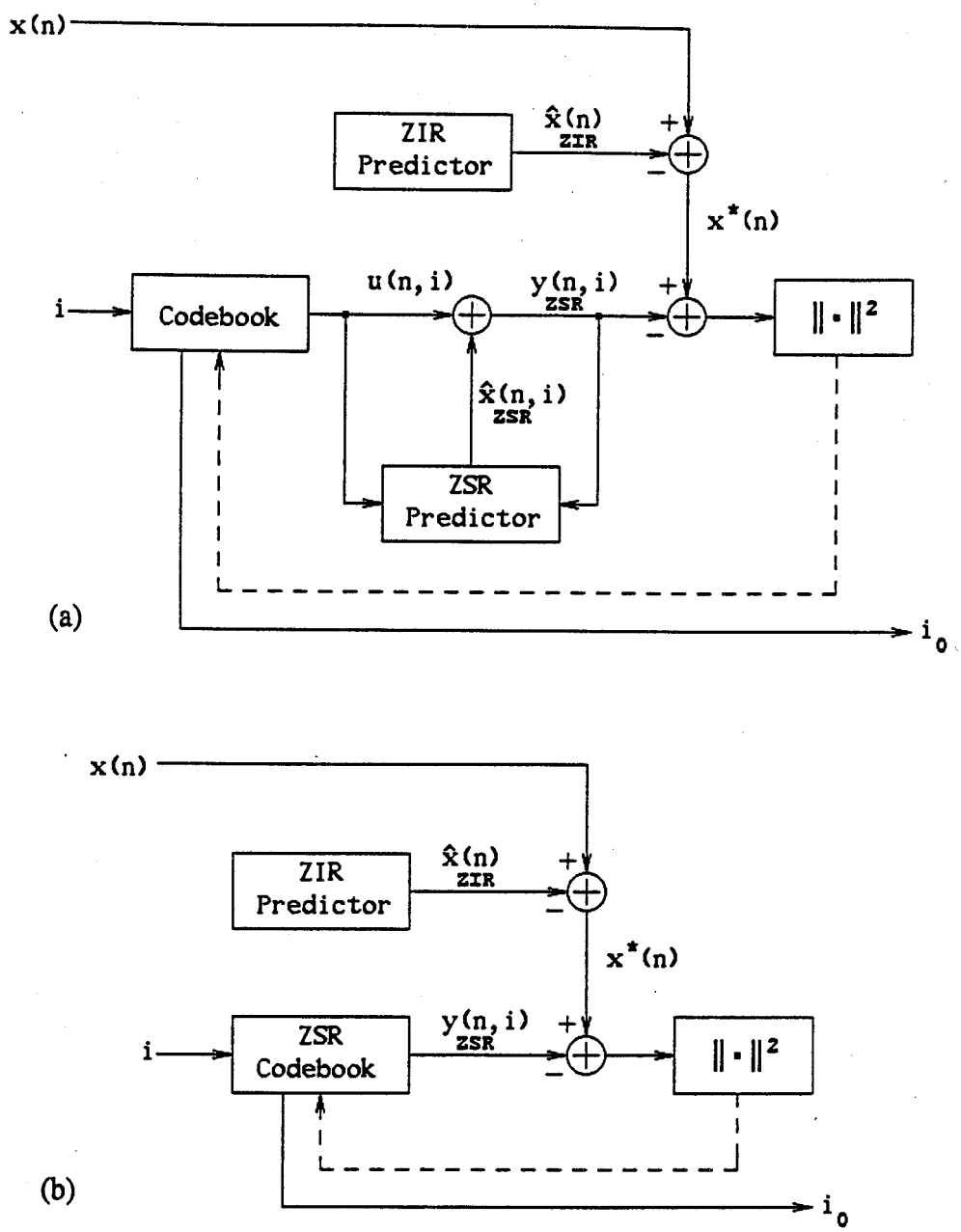

FIGURE 4.2. Reduced complexity Vector ADPCM. (a) Separation of predictor $\hat{x}_{ZIR}(n)$ and $\hat{x}_{ZSR}(n,i)$. (b) Periodic update of $y_{ZSR}(n,i)$ predictor outputs.

4.4. PREDICTOR VARIATIONS.

The 2-pole 6-zero backward-adaptive scalar linear predictor used in the CCITT 32 kb/s algorithm is expected to give very good performance in the Vector ADPCM configuration. However, several variations on this predictor were investigated in the hopes of finding a performance improvement.

These variations included:

1. Using 3 poles instead of 2, since stability constraints are readily available [6]. This predictor is expected to achieve some improvement on low-pass filtered speech signals, but no significant improvement on band-pass filtered speech, since the third pole must necessarily be real and therefore can only contribute a low-frequency peak to the predicted speech spectrum.
2. Using 3 zeroes instead of 6, and applying explicit stability constraints which would ensure that the predictor is minimum phase. It is noted again that instability of the inverse CCITT predictor filter is possible due to the lack of stability constraints on the zeroes of the filter. However, no occurrences of instability have been observed, apparently due to the presence of the leak factors λ_g which limit the growth rate of the all-zero predictor coefficients. Since, in practice, the predictor seems to stay minimum phase without the need for explicit minimum phase constraints, reducing the number of zeroes from 6 to 3 is not expected to achieve an improvement.
3. Using an adaptive step size algorithm, in which the size of the predictor coefficient update term is made dependent upon the recent variances of the cross-correlated signals. This is expected to increase the complexity slightly and offer a small performance improvement by allowing a better adaptation of the predictor.

In the adaptive step size algorithm, the update equations take the form:

$$h_j(n+1) = \lambda_j h_j(n) + \frac{\alpha}{\sigma_u \sigma_y + \gamma} u(n)\, y(n-j) \; ; \quad j = 1,2,...,p \tag{4.10}$$

$$g_j(n+1) = \lambda_j g_j(n) + \frac{\alpha}{\sigma_u^2 + \gamma} u(n)\, u(n-j) \; ; \qquad j = 1,2,...,z \tag{4.11}$$

where γ is a small number to ensure that division by zero does not occur, and a running estimate of the variances is used:

$$\sigma_u^2(n) = \delta\, \sigma_u^2(n-1) + (1-\delta)\, u^2(n)$$

$$\sigma_y^2(n) = \delta\, \sigma_y^2(n-1) + (1-\delta)\, y^2(n) \tag{4.12}$$

4.5. ADAPTIVE POSTFILTERING.

Postfiltering is an effective method of improving the subjective quality of the coded speech [25]. The postfilter is derived simply by scaling the coefficients of the Scalar Predictor. Note that this also gives a motivation for using the powerful scalar predictor rather than the weaker vector predictor, since postfilter performance is directly related to predictor gain. The introduction of the postfilter at the receiver does not require retraining of the VQ codebook.

A signal-to-noise measure which takes into account the effects of postfiltering is obtained by comparing the postfiltered decoded speech with the original speech filtered by the same filter[27].

It should also be noted that the postfilter does not have a constant gain. In practice, uncorrelated speech sounds such as fricatives tend to be strongly attenuated which significantly degrades subjective speech quality. This problem is easily remedied by applying an automatic gain control to the output of the postfilter, to ensure that the local estimates of variance of the reconstructed and postfiltered speech are the same, as shown in Figure 4.3.

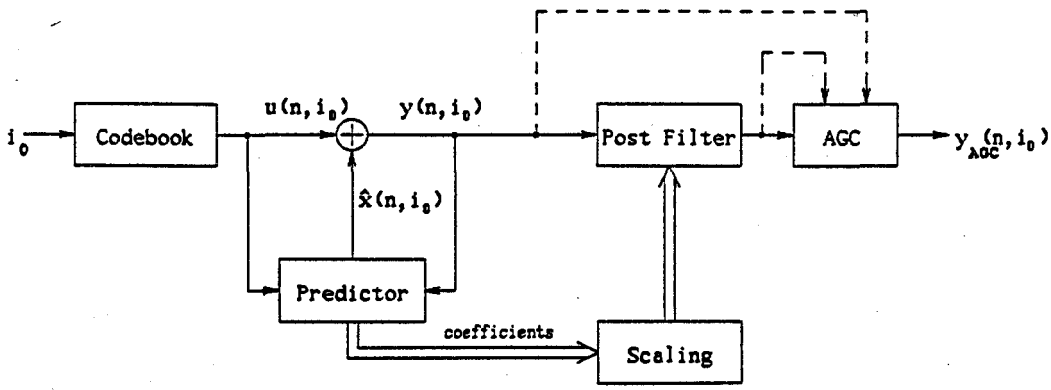

FIGURE 4.3. Vector ADPCM Receiver with Postfilter and AGC.

4.6. GAIN-ADAPTIVE VECTOR QUANTIZATION.

A further improvement in subjective performance is possible with Gain-Adaptive Vector Quantization[13], in which the input to the Vector Quantizer is normalized before optimal codevector selection. In the analysis-by-synthesis configuration, a gain-normalized codebook is used, and the normalized codevector $\mathbf{u}_{norm}$ under consideration must be scaled by an estimate of prediction residual variance σ_u before filtering through the predictor filter. The estimate of variance follows equation (4.12), with a new gain-adapter memory coefficient δ_{gain}:

$$\sigma_u^2(n) = \delta_{gain}\,\sigma_u^2(n-1) + (1-\delta_{gain})\,u^2(n) \qquad (4.13)$$

The resulting configuration is shown in Figure 4.4. This configuration requires retraining of the codebook.

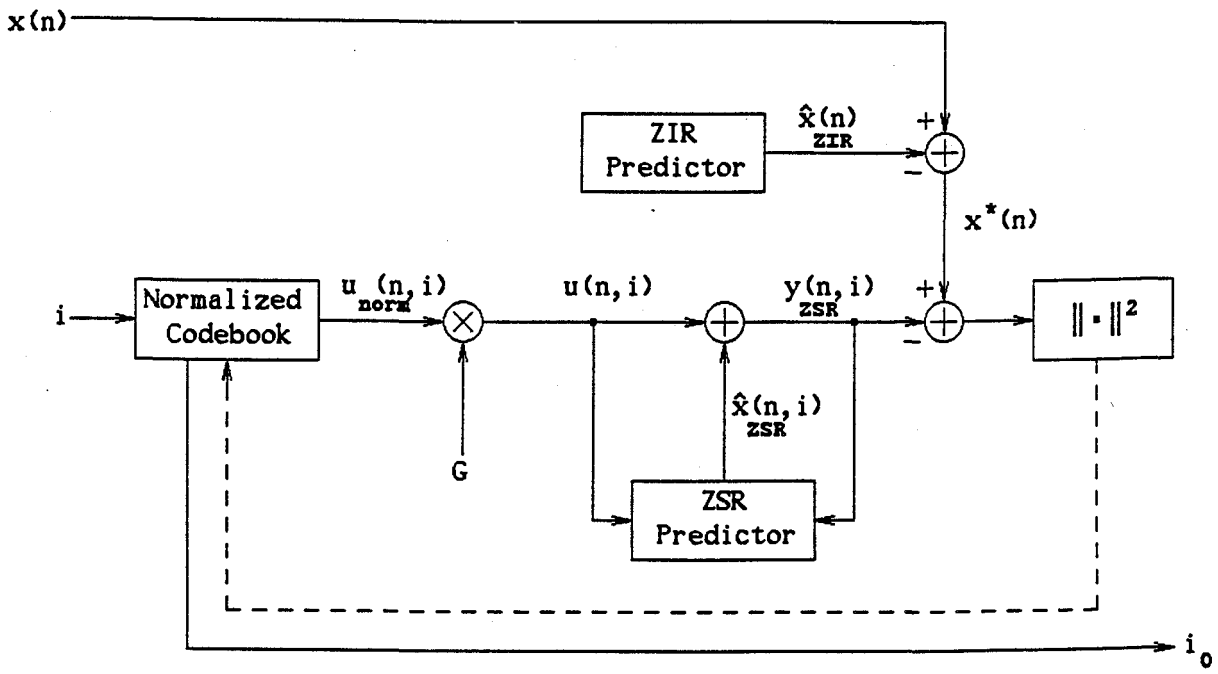

FIGURE 4.4. Vector ADPCM Transmitter with Gain Adaptation.

Unfortunately, the above configuration may result in a substantial increase in complexity depending on the size of the codebook, since each codevector must be multiplied by the gain before filtering through the predictor. An equivalent configuration with no loss in performance is possible by dividing the partial input sample $x^*(n)$ by the gain, which need only be done once per vector. The resulting configuration is shown in Figure 4.5.

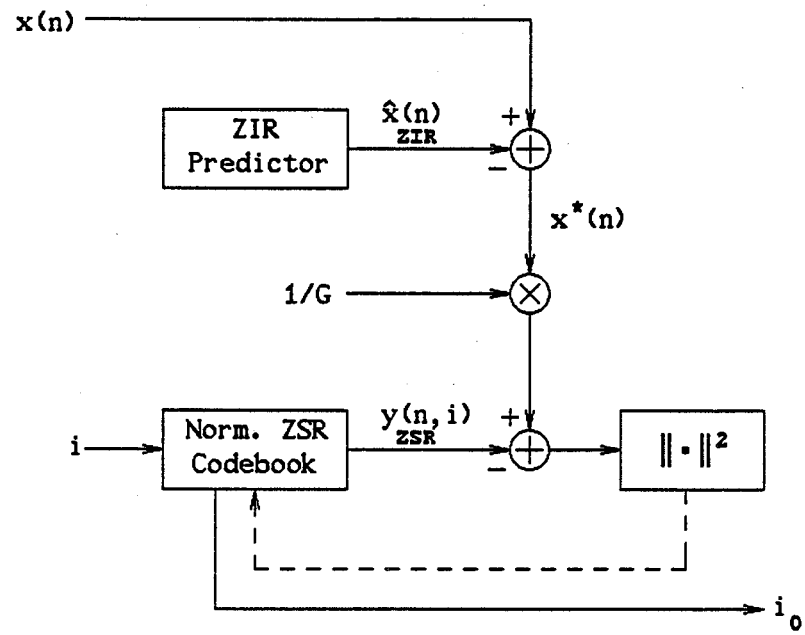

FIGURE 4.5. Complexity-Reduced Vector ADPCM with Gain Adaptation.

4.7. PROPOSED SOLUTION.

The proposed solution consists of the Vector Quantizer and CCITT predictor in an Analysis-by-Synthesis configuration, as shown in Figure 4.6. The Zero-Input-Response (ZIR) of the predictor is precomputed and subtracted from the input vector to produce the samples of the partial input vector $x^*(n)$ before the codebook search is done. In addition, the partial input vector is divided by the gain (a local estimate of prediction residual variance) before the codebook search is done. The zero-state-response (ZSR) table is updated periodically (typically every 48 samples) based on the adapted predictor

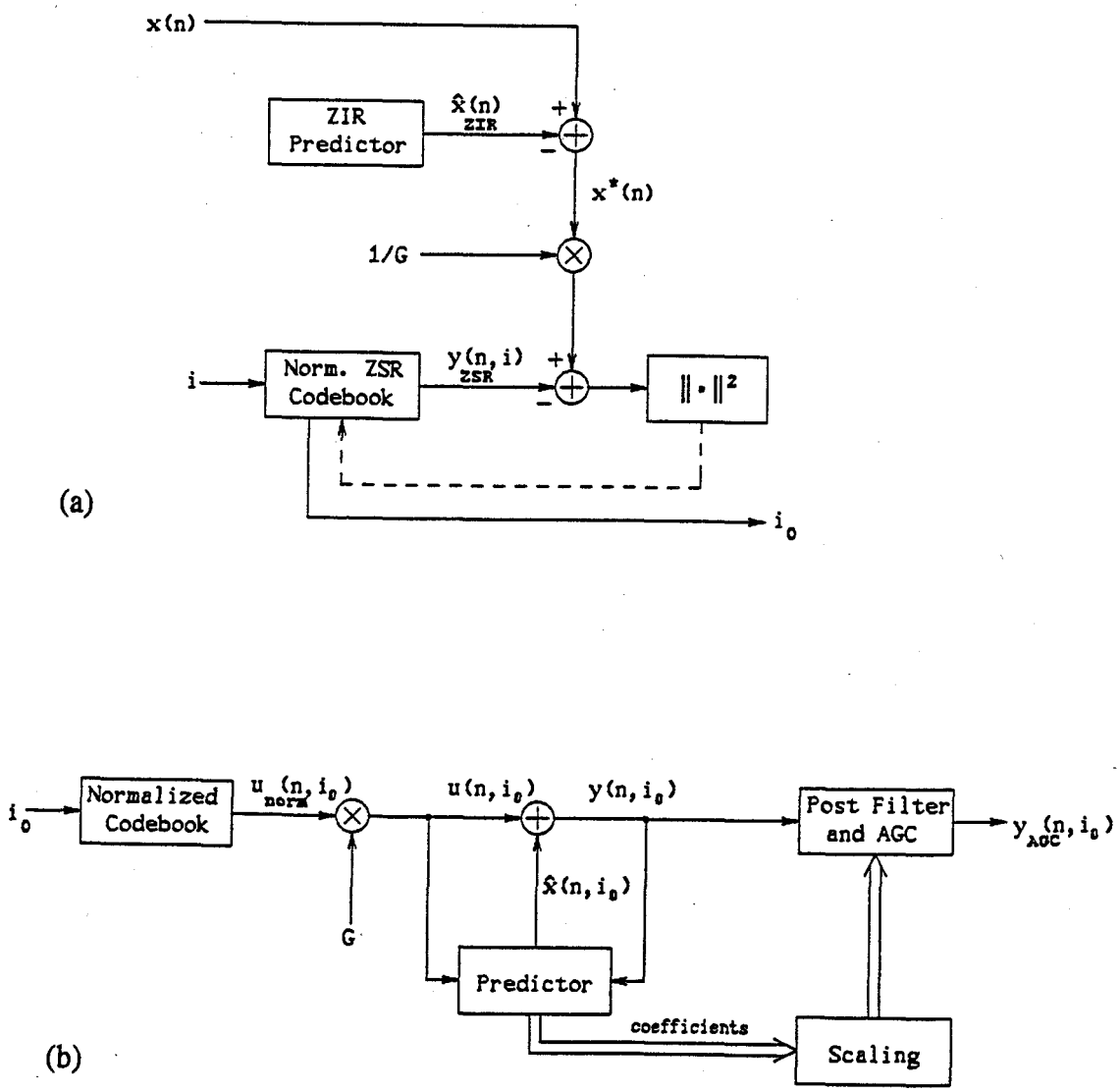

FIGURE 4.6. Proposed Vector ADPCM Solution
(a) Transmitter. (b) Receiver.

coefficients. The codevector which minimizes the mean-square-error between the gain-normalized partial input sample and the partial reconstructed sample is selected, and its index is transmitted to the receiver.

The receiver takes the transmitted codevector index and generates the samples of the corresponding normalized codevector $u_{norm}(n)$; these are multiplied by the gain (the local estimate of prediction residual variance) and filtered through the predictor, to generate the reconstructed samples $y(n)$. The reconstructed samples are then filtered through the postfilter and scaled by the automatic gain control to produce the final output coded speech.

The performance and complexity of the proposed solution are described in Chapter 5.

5. EXPERIMENTAL RESULTS.

Tests have been performed on the proposed Vector ADPCM system to determine the level of performance and complexity. In order to allow a comparison with other related systems, such as direct waveform vector quantization and scalar ADPCM, these systems have also been simulated.

The test conditions and waveform databases used for evaluating the algorithms are described in detail in Section 5.1. The database of waveforms for testing includes uniformly distributed random samples, gaussian random samples, and bandpass filtered speech. In section 5.2, the CCITT predictor and several important variations are tested in isolation (with no quantizer) on speech data, to determine the open-loop prediction gain of the various algorithms. In section 5.3, Waveform Vector Quantization is applied to the uniformly distributed samples, the gaussian distributed samples, and the speech data, to determine the performance of the vector quantizer in the absence of the predictor. Section 5.4 describes the performance of Vector ADPCM, and in Section 5.5, complexity estimates for the Vector ADPCM algorithm and its variations are given.

5.1. TEST CONDITIONS.

As described in Section 3.4.1, all comparisons involving Vector Quantizer-based algorithms will be made on the basis of out-of-training performance i.e. the codebook will be trained on one file, and tested on another file. In order to ensure good training and a fair evaluation of performance, it is necessary to use a long training sequence. However, there is no fixed rule which states how long a training file must be to ensure good training. A reasonable approach to determine the required length of the training and testing files is to measure the first- and second-order statistics of the two files. If the statistics are reasonably close (i.e. within a few percent) the two files are judged to be representative of each other, and therefore, if they were generated independently, we may have some confidence that they are representative of the class of signals to which they

both belong.

The waveform database files used to test the various algorithms include uniformly distributed random sample files Uniform1, Uniform2; gaussian random sample files Gauss1, Gauss2; and bandpass filtered (300 - 3400 Hz) speech files Speech1, Speech2.

The statistics of the database files are summarized below in Table 5.1.

Statistic	Database File					
	Uniform1	**Uniform2**	**Gauss1**	**Gauss2**	**Speech1**	**Speech2**
# of samples	256000	256000	256000	256000	499200	486400
duration (s)	32	32	32	32	63	61
mean	-0.1	0.5	0.2	-0.3	-0.2	0.0
st. dev.	1064.9	1067.5	308.3	307.7	246.4	241.0
minimum	-1849	-1849	-1207	-1255	-1850	-1850
maximum	1849	1849	1249	1270	1850	1850
$r_{xx}(1)$	0.00	0.00	0.00	0.00	0.83	0.84
$r_{xx}(2)$	0.00	0.00	0.00	0.00	0.54	0.56
$r_{xx}(3)$	0.00	0.00	0.00	0.00	0.34	0.35
$r_{xx}(4)$	0.00	0.00	0.00	0.00	0.18	0.16

TABLE 5.1. Statistics of database files used in algorithm evaluation.

It should be noted that the speech files are longer than the random sample files. This was necessary to achieve similar statistics between the two speech files - shorter speech files resulted in significant differences between the first- and second-order statistics. The file Speech1 contains 24 phonetically balanced sentences, 12 spoken by males and 12 spoken by females. The file Speech2 contains 22 phonetically balanced sentences, 10 spoken by males and 12 spoken by females. Both speech files are approximately one minute in duration.

5.2. PREDICTOR PERFORMANCE

As described in Chapter 4, the CCITT 2-pole 6-zero backward-adaptive predictor is used as a starting point for the present work. The performance of this predictor and several variations is described in this section. In this section, all tests are done assuming a perfect quantizer, (i.e. prediction and predictor adaptation are done on the basis of unquantized data) to isolate the performance of the predictor. All tests are performed on the speech file Speech2.

The CCITT predictor is found to give a prediction gain of 9.93 dB on this speech file. A sample of the input waveform and the prediction residual are shown in Figure 5.1.

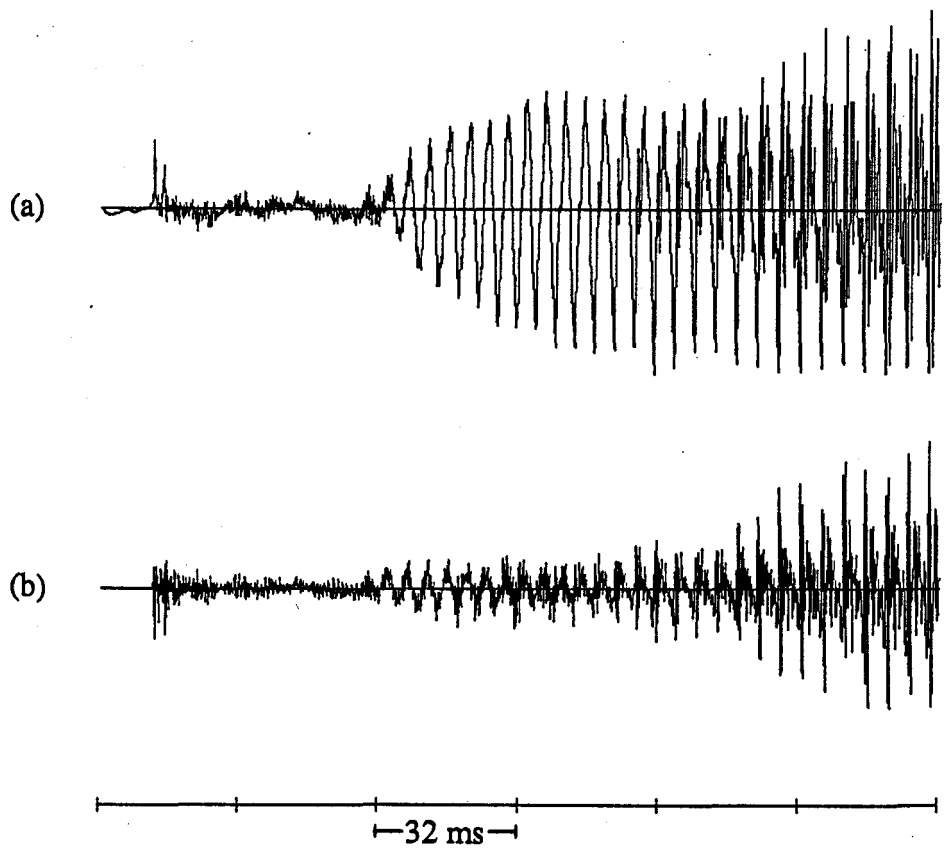

FIGURE 5.1. Effect of CCITT predictor on speech. (a) Input signal and (b) unquantized prediction residual.

It has been shown in Chapter 3 that the CCITT predictor is not constrained to be minimum phase, and therefore that the prediction residual may become unbounded (or very large) if the zeroes of the predictor go outside the unit circle; however in practice it appears that this does not happen. The most likely reason for this is the presence of the leak factors, which limit the growth rate of the all-zero predictor coefficients g_j, and may prevent the zeroes from going outside the unit circle.

This hypothesis was tested by setting the leak factors to 1.0 and running the predictor on the file Speech2. The prediction gain was large and negative, indicating that very large prediction error(s) had occurred. Figure 5.2 shows one such event.

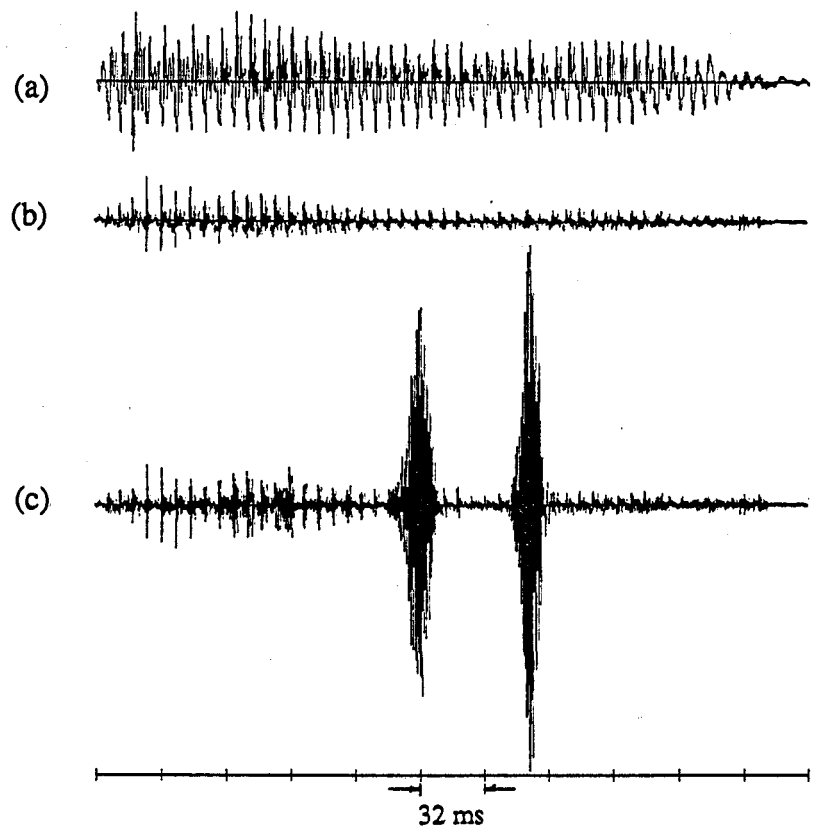

FIGURE 5.2. Effect of leak factors in CCITT predictor. (a) Input signal. (b) unquantized prediction residual with standard CCITT leak factors. (c) unquantized prediction residual with leak factors λ=1.0.

From this, we conclude that the leak factors provide a stabilizing effect in the absence of explicit minimum phase constraints in the CCITT predictor.

In order to be certain that the predictor remains minimum phase, it would be necessary to apply explicit constraints to the zeroes of the predictor. Such constraints exist in a simple form in the third-order case, as described in section 3.3.6, so a 2-pole 3-zero predictor with checks on the zeroes was simulated. The prediction gain dropped to 9.06 dB, indicating that about 0.8 dB of the prediction gain is due to the last three zeroes. When the same 2-pole 3-zero predictor was simulated without the checks on the zeroes, the prediction gain was unchanged at 9.06 dB, indicating that the stability checks were never actually used. This appears to confirm the hypothesis that the stability checks on the zeroes are not needed in the presence of well-chosen leak factors.

The literature states that the zero-based reconstruction terms were introduced in equations (3.37-3.39) to improve the performance of the predictor in the presence of transmission errors, and to improve the performance with voice-band data signals. For this reason, the performance of the simple sign algorithm of equation (3.28-3.29) was expected to be better than the CCITT predictor on speech signals in the absence of transmission errors. However, this was not the case. The 2-pole 6-zero "sign only" algorithm achieved a prediction gain of only 8.92 dB, about 1 dB lower than the CCITT predictor, which includes the zero-based reconstruction. One possible explanation for this is that the pole-zero adaptation equations of (3.37-3.39) specifically account for the presence of the zeroes in the predictor, and therefore may result in a better adaptation of the all-pole coefficients.

The addition of the a third pole to the CCITT algorithm was found to provide a small improvement on lowpass filtered speech, however when tested on bandpass filtered speech, the prediction gain was 9.64 dB, slightly lower than the 2-pole 6-zero CCITT algorithm. This result is expected since the third pole must be real, and therefore can only contribute a low-frequency peak to the modelled speech spectrum.

The effect of the specific 2-pole stability constraints given in equations 3.39 was also investigated. It was found that using the CCITT predictor but using the all-pole

stability checks $|h_2| \leq 0.75$ and $|h_1| \leq 0.75 - h_2$ resulted in a slightly lower prediction gain of 9.78 dB. Similarly, it was found that using the all-pole stability checks $|h_2| \leq 0.9375$ and $|h_1| \leq 0.9375 - h_2$ also resulted in a lower prediction gain of 9.54 dB.

Finally, the adaptive step size algorithm was simulated, in which the step size of the CCITT predictor equations (3.37-3.39) was made inversely proportional to the standard deviation of the cross-correlated terms in the update equation, as in equations (4.10-4.11). With 2 poles and 6 zeroes, the prediction gain improved to 10.22 dB, up about 0.3 dB from the standard CCITT algorithm.

From the above test results, the following conclusions may be drawn:

1. The CCITT 2-pole 6-zero predictor with zero-based reconstruction gives 9.93 dB prediction gain on band-pass filtered speech, in the absence of quantization noise.
2. The leak factors in the CCITT algorithm appear to have a stabilizing effect on the zeroes of the predictor, which are not constrained to be within the unit circle. For this reason, it appears that stability checks on the zeroes are not required.
3. The addition of a third pole does not provide a significant improvement on bandpass filtered speech.
4. The use of an adaptive step size algorithm, as described above, appears to improve performance only slightly, at the expense of considerable additional complexity.

In addition, the CCITT predictor is known to have other desirable properties which were not investigated above, including good performance in the presence of transmission errors and with voice-band data. For these reasons, the CCITT 2-pole 6-zero predictor is used without modification in all subsequent tests.

5.3. WAVEFORM VECTOR QUANTIZER PERFORMANCE

The performance of vector quantizers operating directly on particular waveform types has been evaluated, with a view to providing a performance comparison with Vector ADPCM, in which the vector quantizer operates on a prediction residual. Vector Quantizer performance was evaluated on three signal types: uniform random samples, gaussian random samples, and speech. As described in Section 5.1, different files are used in each case for training and testing. Final comparisons are made on the basis of out-of-sequence test results.

5.3.1. Uniformly Distributed Source.

Table 5.2 shows the performance of the vector quantizer on the uniformly distributed source, for vector dimensions k=1 (scalar) to k=4, and a rate of 2 bits/sample.

Vector Dimension	In-Sequence		Out-of-Sequence	
	SNR	SEGSNR	SNR	SEGSNR
1	12.03	12.03	12.05	12.05
2	12.03	12.03	12.05	12.05
3	12.02	12.02	12.02	12.02
4	12.13	12.13	11.99	11.99

TABLE 5.2. Waveform vector quantizer performance on uniformly distributed random samples.

The performance on the uniformly distributed random samples is seen to be approximately 12 dB, essentially independent of vector dimension k. The slight deviations at higher vector dimensions are due to the decreasing training ratio with increasing vector dimension; in the k=1 case, there are 4 centroids, with an average of 64,000 vectors associated with each. In the k=4 case, there are 256 centroids, with an average of 250 vectors associated with each. This generally results in slightly better in-sequence performance,

and slightly worse out-of-sequence performance, as seen above. However, for all practical purposes, at 2 bits/sample, there appears to be no benefit to using Vector Quantization over scalar quantization on uniformly distributed random samples.

In the scalar and k=2 cases, it is convenient and instructive to give a visual display of the input data distributions and the resulting centroids. Figure 5.3 shows the input distribution, as a histogram indicating likelihood of occurrence, with the resulting centroids superimposed on the figure. As should be expected for the uniformly distributed random samples, the centroids are also uniformly distributed.

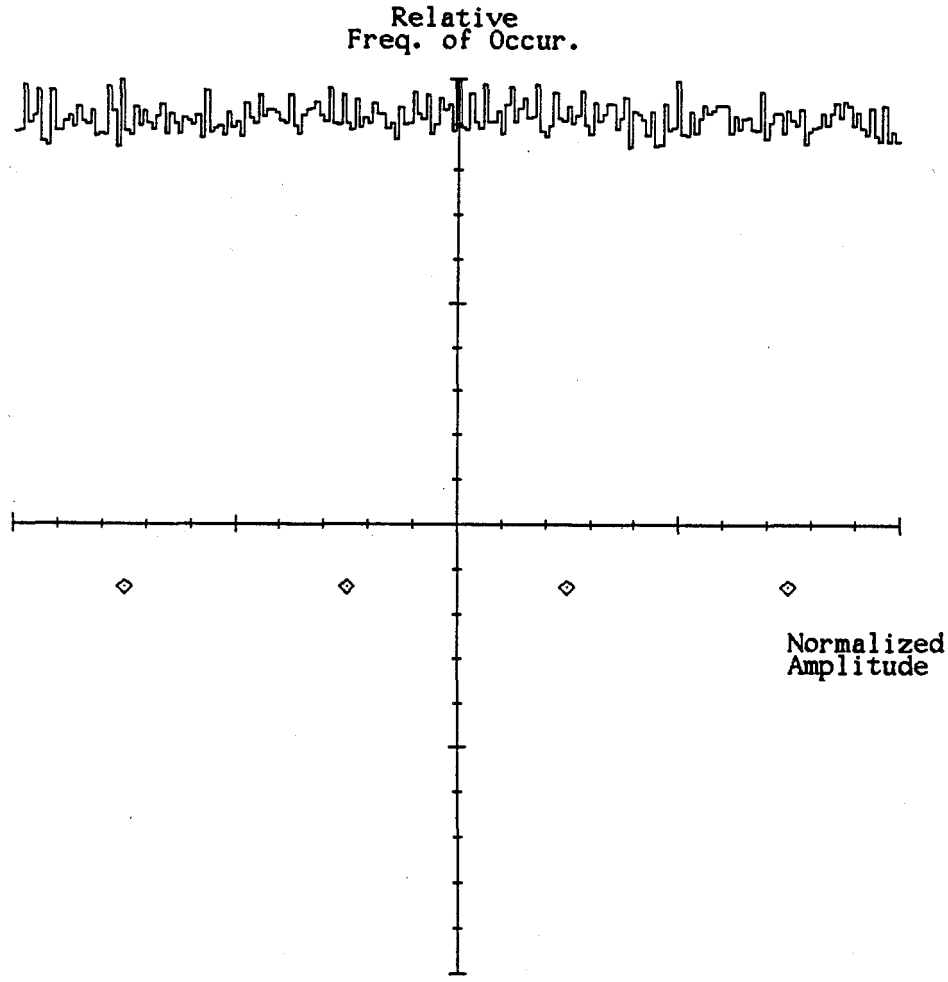

FIGURE 5.3. Input distribution and centroids for uniformly distributed random samples (scalar quantization).

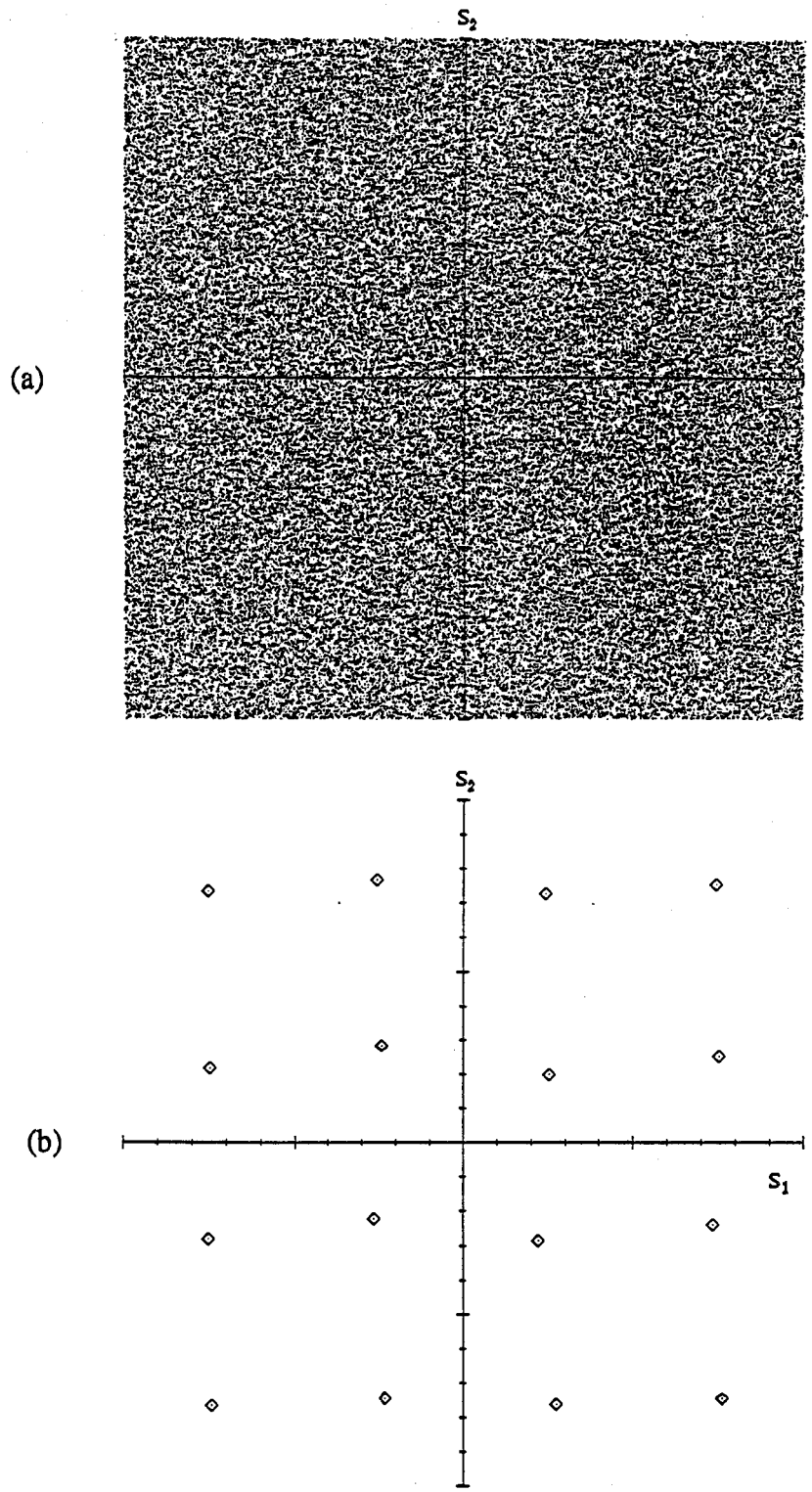

FIGURE 5.4. (a) Input distribution and (b) centroids for uniformly distributed random samples (dimension 2 vector quantization).

For k=2, the input distribution is represented as a two-dimensional scatter-plot in Figure 5.4 (a). The corresponding centroids are shown in Figure 5.4 (b). Clearly the centroids fall in a uniform square grid pattern, with slight deviations which may be accounted for by the particular input distribution of the training sequence.

5.3.2. Gaussian Source.

Table 5.3 shows the performance of the vector quantizer on the gaussian source, for vector dimensions k=1 (scalar) to k=4, and a rate of 2 bits/sample.

Vector Dimension	In-Sequence		Out-of-Sequence	
	SNR	SEGSNR	SNR	SEGSNR
1	9.38	9.40	9.41	9.42
2	9.77	9.79	9.77	9.78
3	10.00	10.01	9.95	9.96
4	10.31	10.32	10.13	10.14

TABLE 5.3. Waveform vector quantizer performance on gaussian samples.

The performance on the gaussian random samples is seen to improve significantly with vector dimension. On out-of-sequence results, there is approximately a 0.7 dB improvement, at a cost of a factor of 64 increase in complexity. Figure 5.5 shows the input distribution with the resulting centroids superimposed on the figure. As should be expected for the gaussian samples, the centroids are clustered in more closely towards the histogram peak.

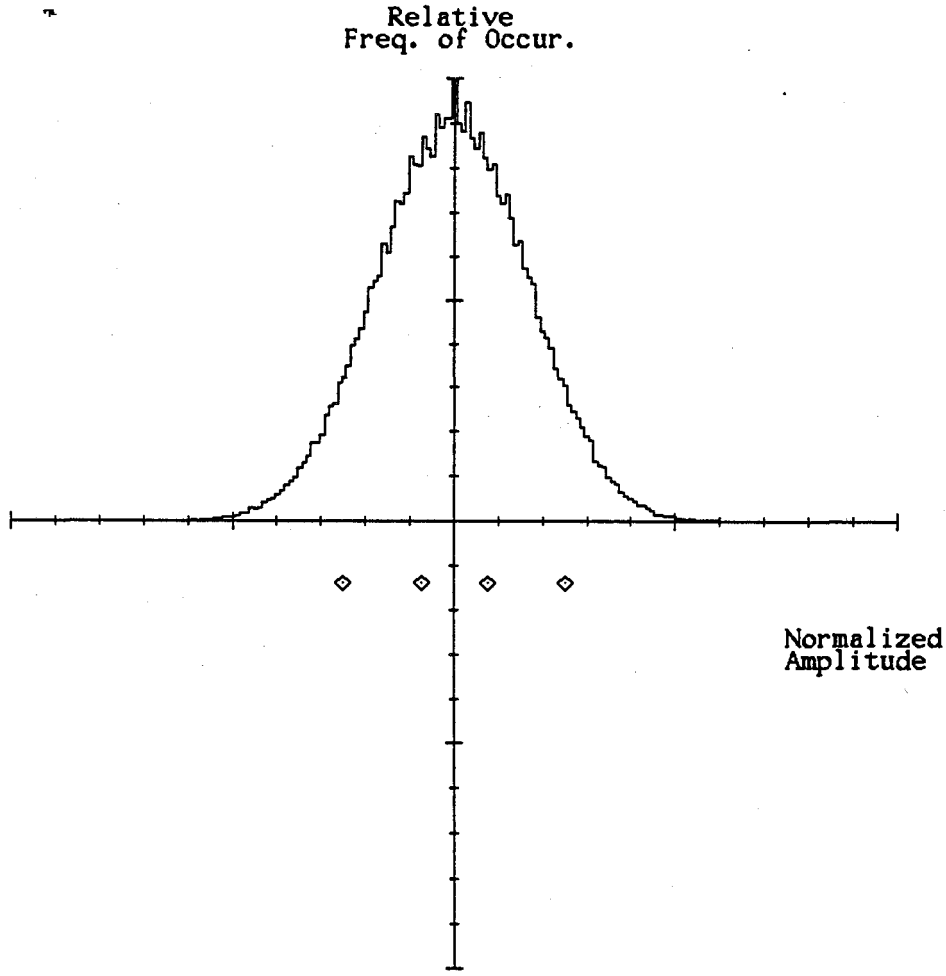

FIGURE 5.5. Input distribution and centroids for gaussian random samples (scalar quantization).

For k=2, the input distribution and corresponding centroids are shown in Figure 5.6. There is one centroid located at the origin (the mean of the distribution), with six centroids in a hexagonal pattern around the first centroid, and the remaining nine centroids distributed evenly around the outside. This clearly shows how vector quantization can achieve its performance improvement, even for an independent random source.

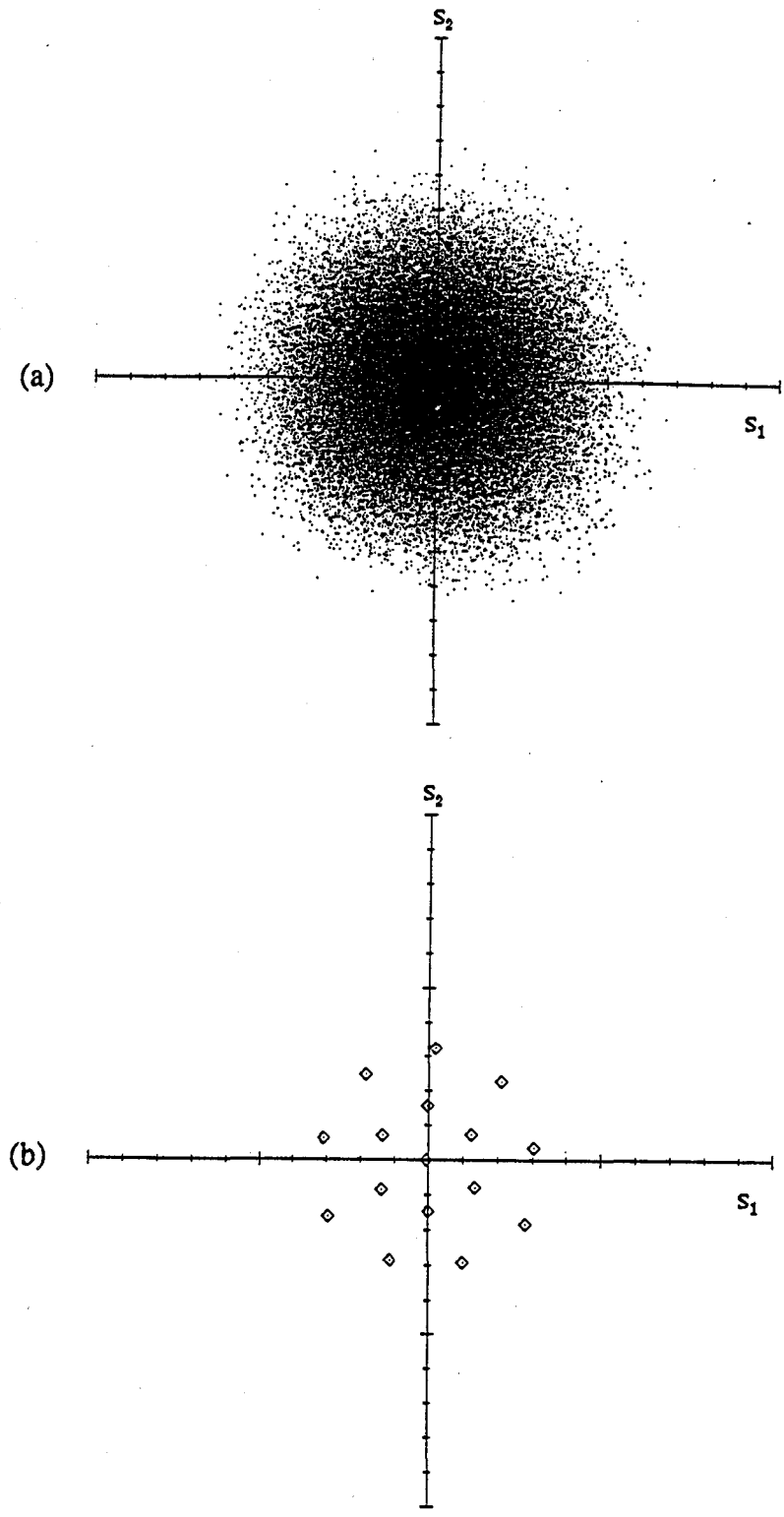

FIGURE 5.6. (a) Input distribution and (b) centroids for gaussian random samples (dimension 2 vector quantization).

5.3.3. Speech Source.

Table 5.4 shows the performance of the vector quantizer on speech, for vector dimensions k=1 (scalar) to k=4, and a rate of 2 bits/sample.

Vector Dimension	In-Sequence		Out-of-Sequence	
	SNR	SEGSNR	SNR	SEGSNR
1	7.19	2.61	7.36	2.92
2	10.88	7.38	10.92	7.56
3	12.67	10.67	12.74	10.55
4	13.69	12.85	13.51	12.55

TABLE 5.4. Waveform vector quantizer performance on speech.

The performance on speech is seen to improve dramatically with vector dimension. On out-of-sequence results, there is over 6 dB improvement in SNR and over 9 dB improvement in SEGSNR, at a cost of a factor of 64 increase in complexity. Such a dramatic improvement is possible because speech is a highly correlated source, as may be seen by the inspecting the first few autocorrelation coefficients for the speech files in Table 5.1.

Figure 5.7 shows the input distribution with the resulting centroids superimposed on the figure. The large peak at very small amplitude samples is due to the high occurrence of low-amplitude sounds in speech, such as fricatives, and silence in stop consonants and between sentences. One centroid is essentially devoted to representing this central peak; this leaves three centroids to be distributed on the two sides of the peak. In this case, the training algorithm converged to a solution in which there were two centroids on the right and one on the left. However, it is clear that another good solution could be the mirror image of this solution. In fact, it is possible that this other solution could result in a higher SNR, since the training algorithm is only guaranteed to converge to a local minimum, not a global minimum.

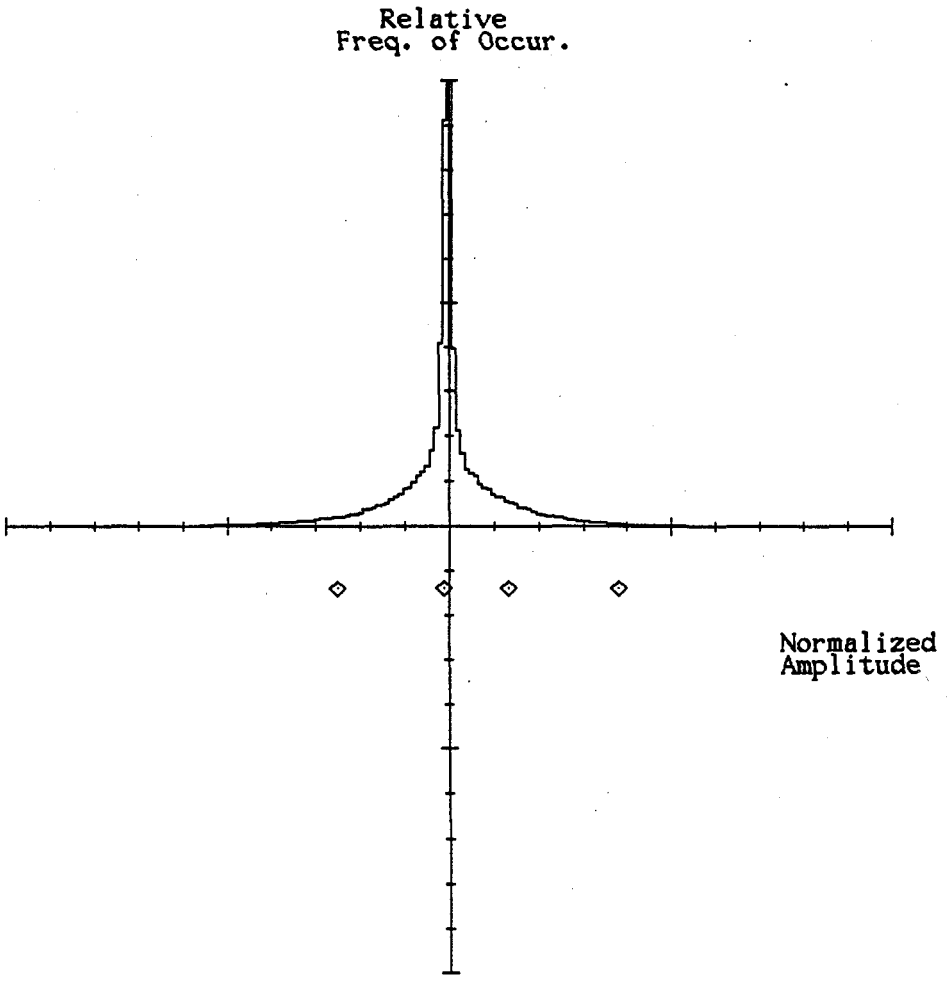

FIGURE 5.7. Input distribution and centroids for speech (scalar quantization).

For k=2, the input distribution and corresponding centroids are shown in Figure 5.8. The dark diagonal band in the input distribution is a result of the high adjacent-sample correlation in speech, upon which the vector quantizer capitalizes to achieve its dramatic performance improvement.

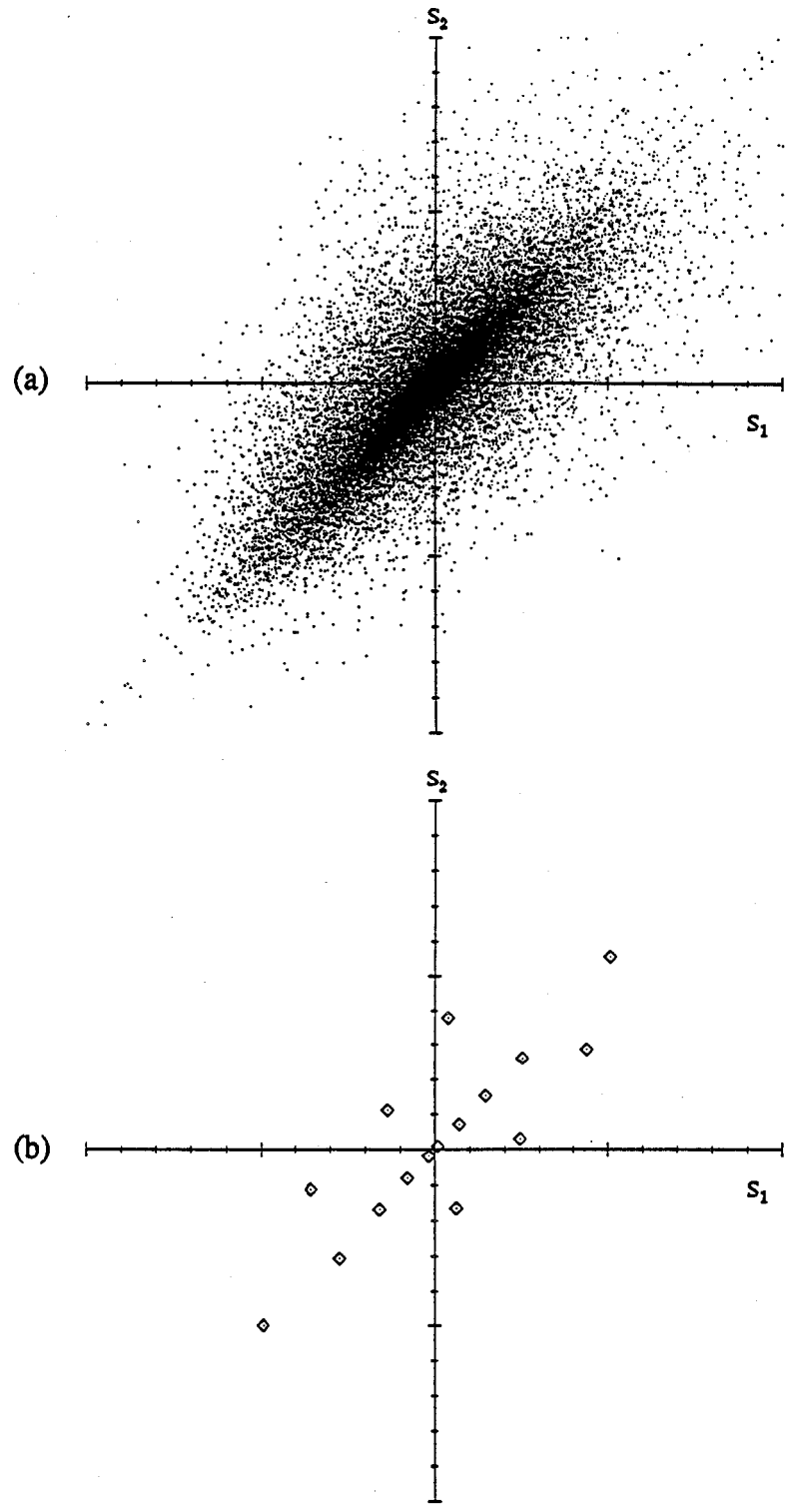

FIGURE 5.8. (a) Input distribution and (b) corresponding centroids for speech (dimension 2 vector quantization).

5.4. VECTOR ADPCM PERFORMANCE

In this section, the performance of the proposed solution and its variations is evaluated. The huge number of possible variations of predictor type, predictor order, vector dimension, etc. makes an exhaustive study of all possible variations impossible. Therefore, the following approach was used to evaluate the various configurations of the proposed solution:

1. Within the Vector ADPCM configuration, the CCITT predictor and several variations were investigated. For this study, a vector quantizer of dimension k=3 was used. The objective of this study is to determine which predictor variations to use in subsequent tests.
2. The effect of the ZSR table update period was measured, by using the CCITT predictor, a k=3 vector quantizer, and varying the ZSR update period. The objective of this study is to determine the largest ZSR update period which may be used without a significant performance degradation.
3. With the predictor type and ZSR update period chosen, the Vector ADPCM configuration was then tested as a function of vector dimension. The objective of this study is a performance comparison between non-gain-adaptive scalar ADPCM and non-gain-adaptive Vector ADPCM.
4. Next, several variations on the gain adaptation method were investigated, using the CCITT predictor and vector dimension k=3. The objective of this study is to determine which gain adaptation method should be used in subsequent tests.
5. The best gain adaptation method from (4) was combined with the CCITT predictor, and tested as a function of vector dimension. The objective of this study is a comparison between scalar ADPCM and Vector ADPCM.
6. Finally, the effect of post-filtering was investigated for the proposed configuration.

5.4.1. Predictor Variations

Table 5.5 shows the performance of various predictor algorithms within the non-gain-adaptive Vector ADPCM configuration. In all tests, the vector dimension was k=3. The objective of this study was to determine which predictor to use in subsequent tests.

Predictor	Seq.	SNR	SEGSNR	G_P
2p6z sign	in	15.47	11.59	7.91
(ccitt)	out	15.48	11.79	8.33
2p6z adap	in	16.00	13.64	8.32
	out	16.02	13.61	8.76
2p3z sign	in	14.94	11.69	7.47
	out	15.01	12.06	7.82
2p3z adap	in	15.50	12.65	7.79
	out	15.20	12.70	8.04
3p3z adap	in	14.50	10.94	6.87
	out	14.69	11.67	7.26

TABLE 5.5. Performance of Non-Gain-Adaptive Vector ADPCM with predictor variations.

The CCITT predictor achieves very good performance. A 0.5 dB improvement in SNR and a 2.0 dB improvement in SEGSNR are possible with the adaptive step size algorithm. Adding a third pole does not help; nor does deleting three zeroes and applying stability constraints to the remaining three zeroes.

Based on the above results, the CCITT predictor is favoured due to its good performance and low complexity, and the 2p6z predictor with adaptive step size is favoured for overall performance.

5.4.2. ZSR Table Update Period.

The effect of the ZSR table update period was measured, by using the CCITT predictor, a k=3 vector quantizer, and varying the ZSR update period. The objective of this study is to determine the largest ZSR update period which may be used without a significant performance degradation. Figure 5.9 shows the performance of the non-gain-adaptive Vector ADPCM algorithm as a function of ZSR update period in samples.

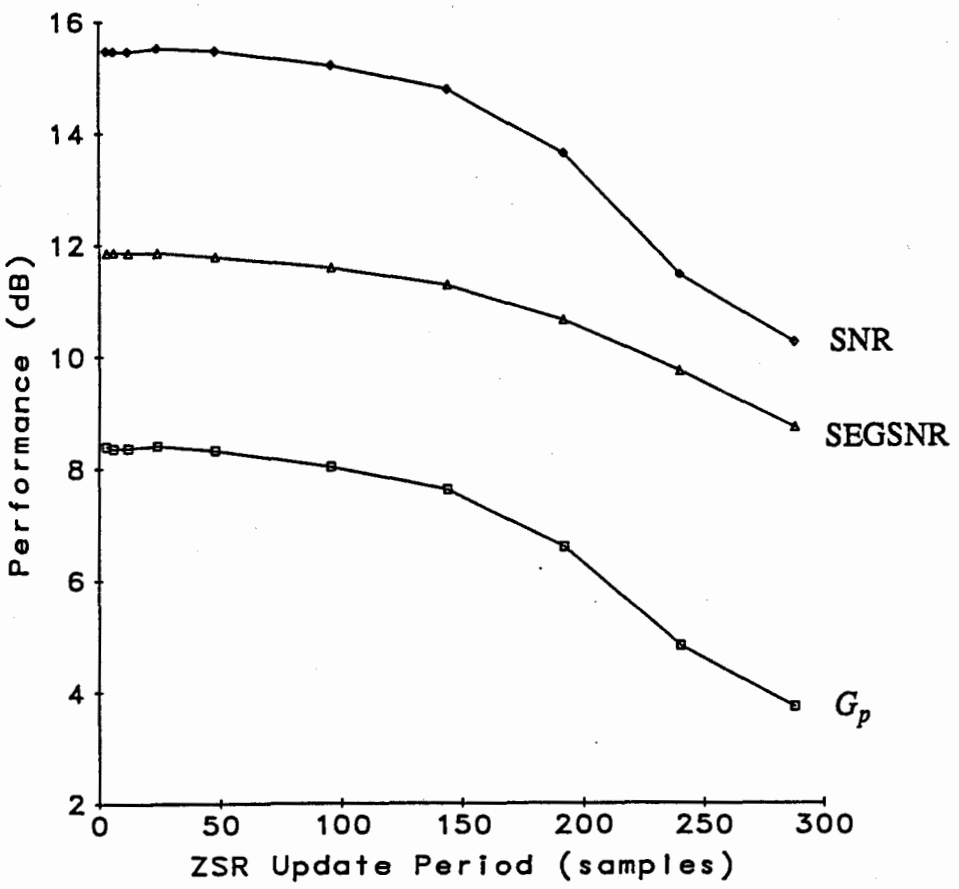

FIGURE 5.9. Effect of ZSR update period on Vector ADPCM performance.

From the above figure, we conclude that we may use a ZSR period of up to 50 samples (about 6 ms) with no significant performance degradation.

5.4.3. Non-Gain-Adaptive Vector ADPCM

Using the CCITT predictor and ZSR update period of 48 samples, the Vector ADPCM configuration was then tested as a function of vector dimension. The objective of this study is a performance comparison between non-gain-adaptive scalar ADPCM and non-gain-adaptive Vector ADPCM.

Seq.	k	SNR	SEGSNR	G_P
in	1	12.12	6.93	6.14
	2	14.01	8.61	7.24
	3	15.47	11.59	7.91
	4	16.52	13.70	8.38
out	1	12.26	7.32	6.48
	2	13.71	7.95	7.53
	3	15.48	11.79	8.33
	4	16.38	13.69	8.89

TABLE 5.6. Performance of Non-Gain-Adaptive Vector ADPCM as a function of Vector Dimension k.

As shown in Table 5.6, Non-Gain-Adaptive Vector ADPCM up to vector dimension k=4 achieves a 4 dB SNR improvement and over 6 dB SEGSNR improvement over non-gain-adaptive scalar ADPCM in out-of-sequence testing. It should also be noted that the improved quantizer performance with vector dimension leads to improved predictor performance, since the quantized values used to adapt the predictor and predict the following samples are closer to the unquantized values.

The large improvement with vector dimension is encouraging, however, it should be noted that this is not yet a fair comparison with practical scalar ADPCM systems which normally contain gain-adaptation. This comparison follows in section 5.4.5.

The input distributions and centroids are shown in Figure 5.10 for the non-gain-adaptive scalar ADPCM algorithm.

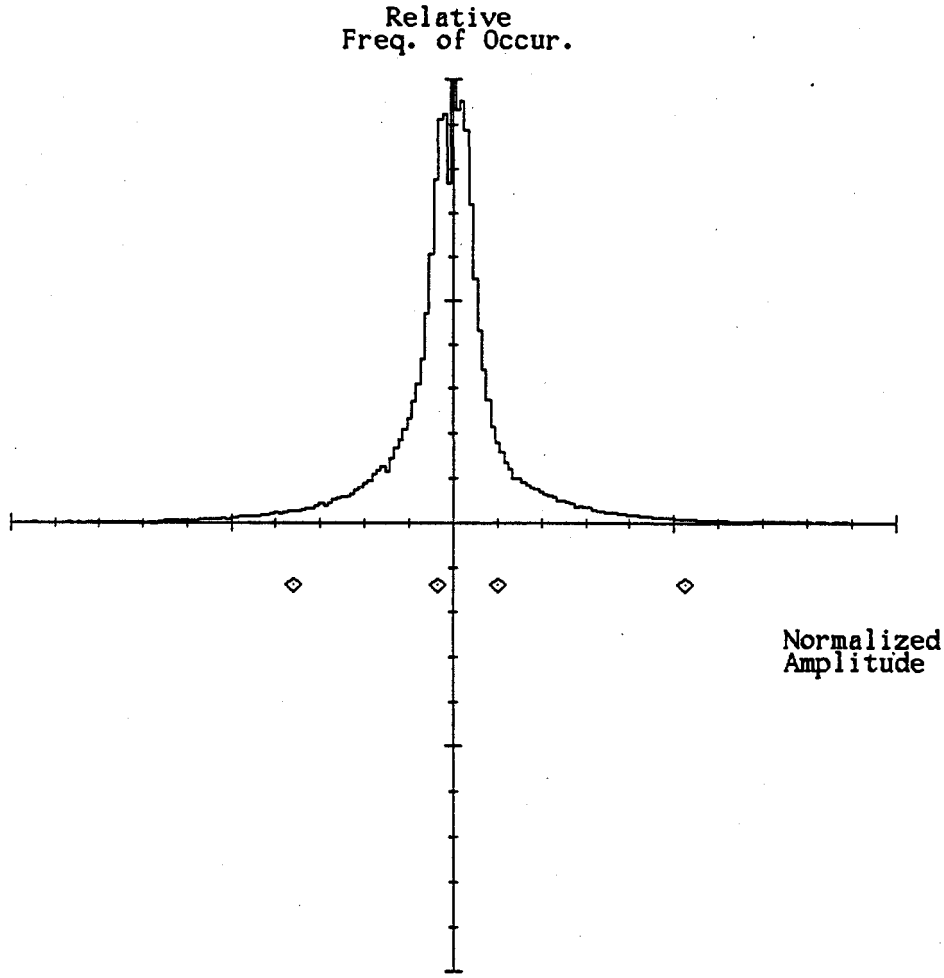

FIGURE 5.10. Input distribution and centroids for speech (Non-gain-adaptive scalar ADPCM).

The input distributions and centroids are shown in Figure 5.11 for the two-dimensional non-gain-adaptive Vector ADPCM algorithm. The curved high-density regions in the input distribution were unexpected - these appear to be an artifact of the closed-loop training procedure.

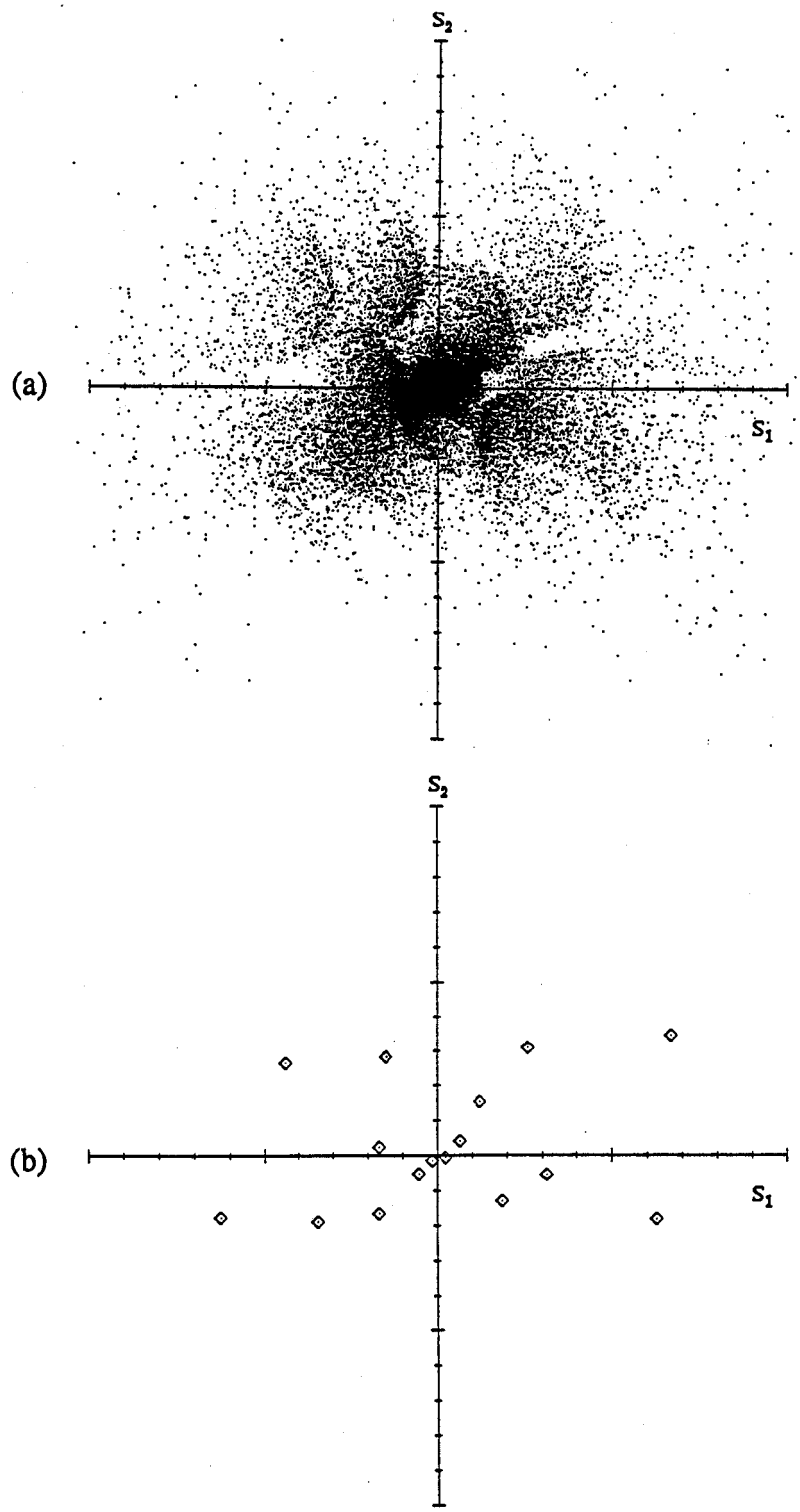

FIGURE 5.11. Input distribution and centroids for speech (Non-gain-adaptive Vector ADPCM). (a) k=2 input distribution. (b) k=2 centroids.

5.4.4. Gain Adaptation Variations

Next, several variations on the gain adaptation method were investigated, using the CCITT predictor and vector dimension k=3. The objective of this study is to determine which gain adaptation method should be used in subsequent tests.

Table 5.7 shows the performance of the Gain-Adaptive Vector ADPCM algorithm as a function of gain-adapter memory coefficient δ_{gain}. In all tests, the vector dimension was k=3 and the CCITT predictor was used. The objective of this study was to determine which value of δ_{gain} to use in subsequent tests.

δ_{gain}	Seq.	SNR	SEGSNR	G_P
0.90	in	17.10	19.73	8.32
	out	16.30	19.10	8.65
0.92	in	17.09	19.69	8.30
	out	16.50	19.07	8.64
0.95	in	17.11	19.60	8.34
	out	16.79	19.04	8.89
0.97	in	17.06	19.44	8.36
	out	16.80	18.96	8.86

TABLE 5.7. Performance of Gain-Adaptive Vector ADPCM as a function of gain-adapter memory coefficient δ_{gain}.

Clearly, the performance is not particularly sensitive to the choice of gain-adapter memory coefficient, however there is a slight advantage in terms of both SNR and SEGSNR in out-of-sequence tests by using δ_{gain}=0.95. This value was used in all subsequent tests.

5.4.5. Performance of Proposed Solution.

The best gain adaptation method from the previous section was combined with the CCITT predictor, and tested as a function of vector dimension. The objective of this study is a comparison between gain-adaptive scalar ADPCM and gain-adaptive Vector ADPCM.

Seq.	k	SNR	SEGSNR	G_P
in	1	14.77	17.29	6.89
	2	16.16	18.74	7.74
	3	17.11	19.60	8.34
	4	17.70	20.07	8.65
out	1	14.18	16.77	7.16
	2	15.82	18.39	8.22
	3	16.79	19.04	8.89
	4	17.06	19.34	9.04

TABLE 5.8. Performance of Gain-Adaptive Vector ADPCM as a function of Vector Dimension k.

In out-of-sequence tests, Vector ADPCM with k=4 is seen to give approximately 3 dB improvement over scalar ADPCM. The input distributions and centroids are shown in Figure 5.12 for the gain-adaptive scalar ADPCM algorithm, and in Figure 5.13 for the. gain-adaptive Vector ADPCM algorithm with vector dimension k=2.

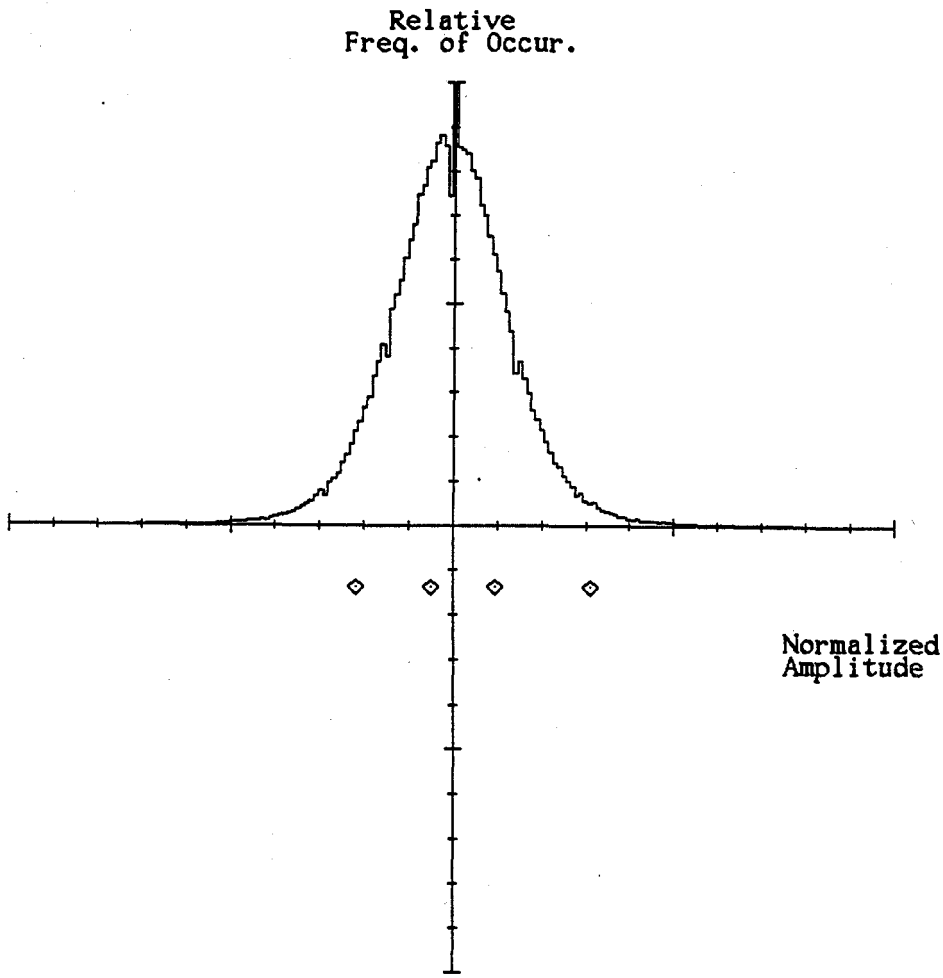

FIGURE 5.12. Input distribution and centroids for speech (gain-adaptive scalar ADPCM).

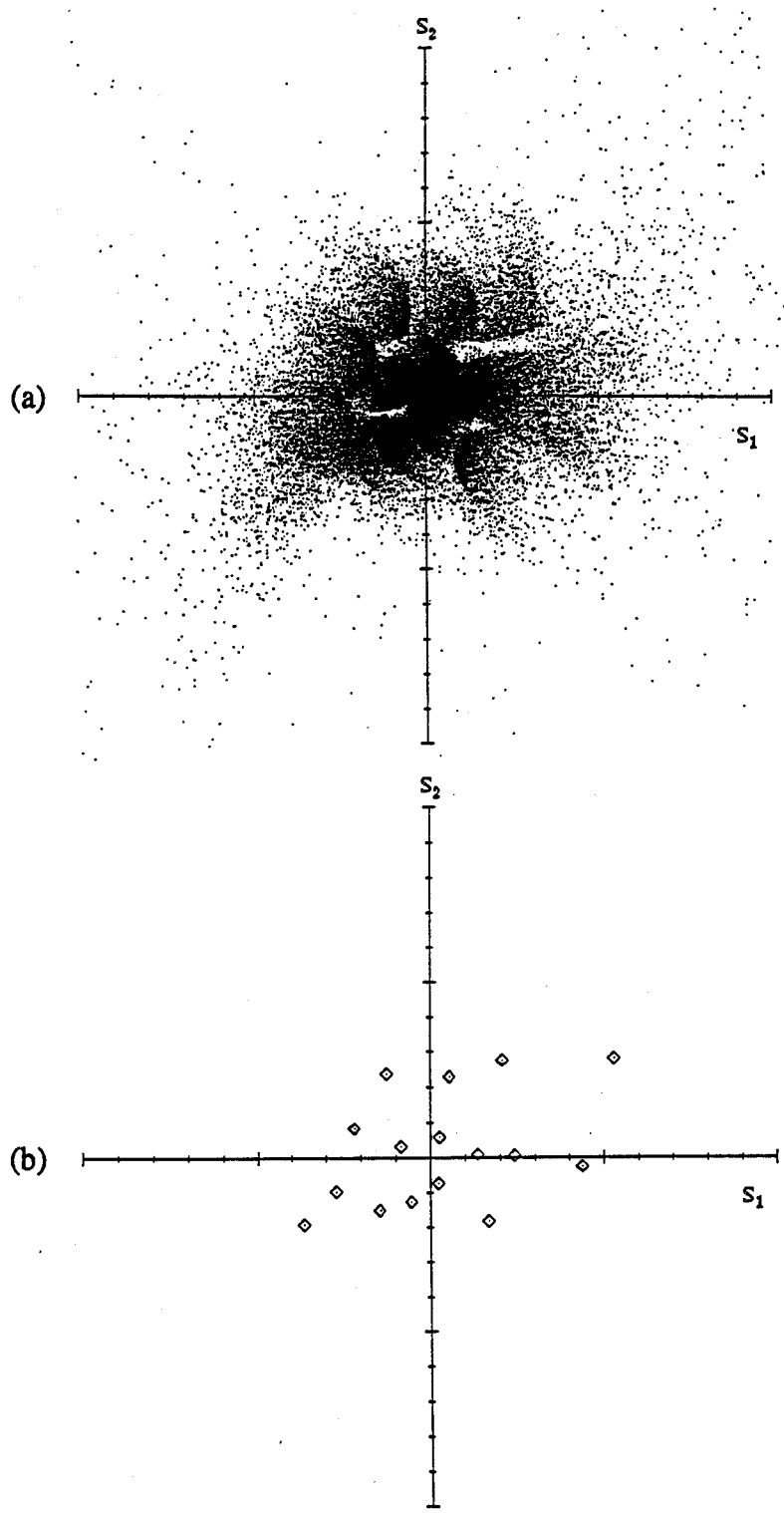

FIGURE 5.13. Input distribution and centroids for speech (gain-adaptive vector ADPCM). (a) k=2 input distribution. (b) k=2 centroids.

5.4.6. Post-Filtering

Postfiltering is found to provide a significant improvement in subjective speech quality. The best subjective improvement is achieved with the all-pole scaling coefficient = 0.5 and the all-zero scaling coefficient = 1.0. Without postfiltering, the quantization noise is clearly audible and is somewhat annoying. With postfiltering, the speech is judged informally to be very intelligible, noise-free, but slightly muffled, a quality which prevents an informal assessment of toll quality. Therefore, the coded speech is informally judged to be of very good communications quality.

5.4.7. Coding Delay

The delay of the proposed algorithm is simply

$$delay = \frac{2k}{f} \tag{5.1}$$

where k is the vector dimension, f is the sampling rate, and the factor of two accounts for transmitter and receiver operation. For vector dimension 4 and 8 kHz sampling rate, the delay is 1 ms, well below the CCITT 16 kb/s requirement of less than 5 ms, and the CCITT 16 kb/s objective of less than 2 ms.

5.4.8. Speech Waveforms

A display of typical waveforms produced by the various coding methods is shown in Figure 5.14-5.16. In all three figures, the original speech was the syllable "ta" (as in the word "tack"), spoken by a female talker.

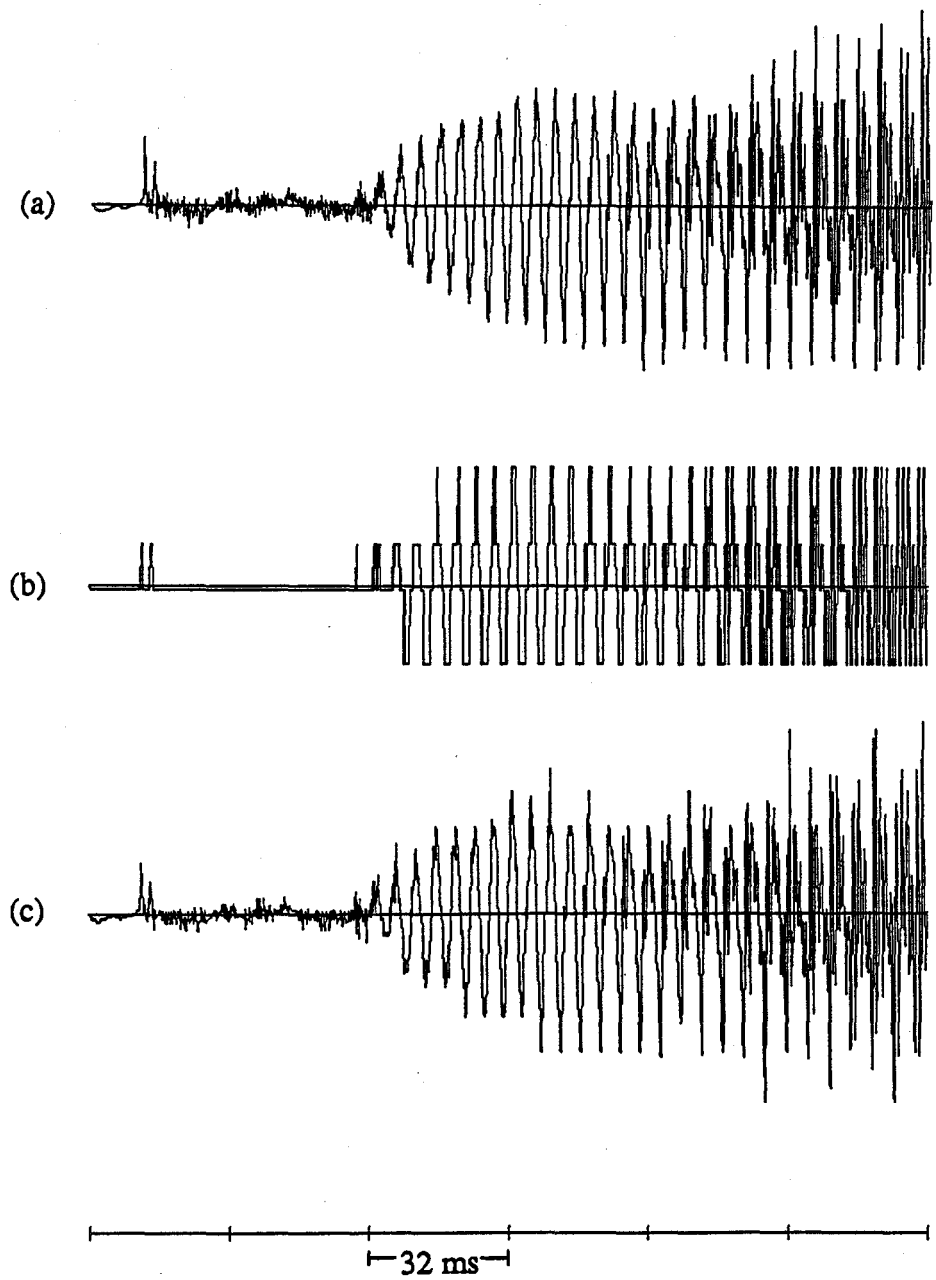

FIGURE 5.14. Coded speech waveforms at 16 kb/s. (a) Original signal. (b) scalar quantization. (c) k=4 vector quantization.

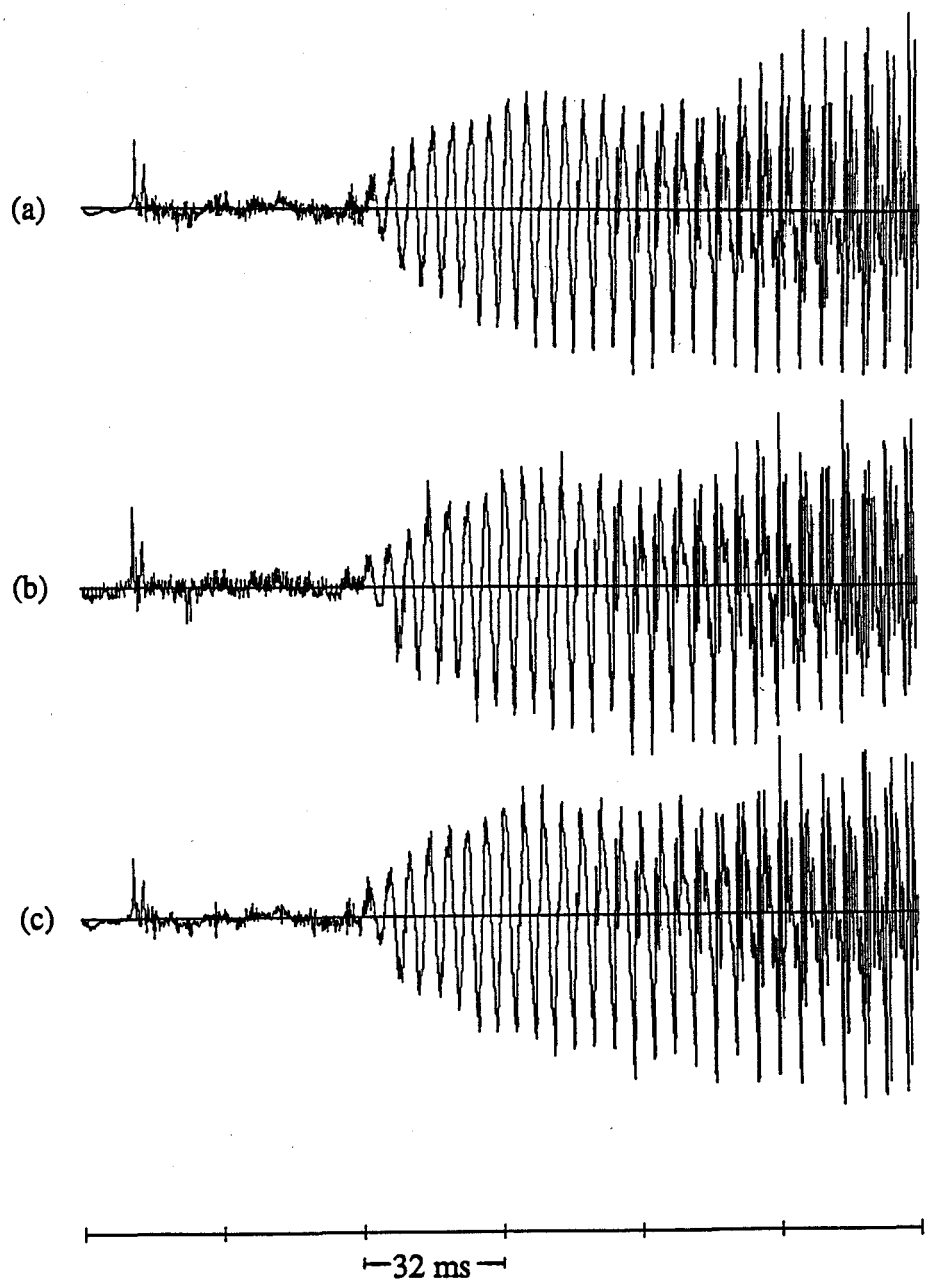

FIGURE 5.15. Coded speech waveforms at 16 kb/s. (a) Original signal. (b) scalar (non-gain-adaptive) ADPCM. (c) k=4 Vector (non-gain-adaptive) ADPCM.

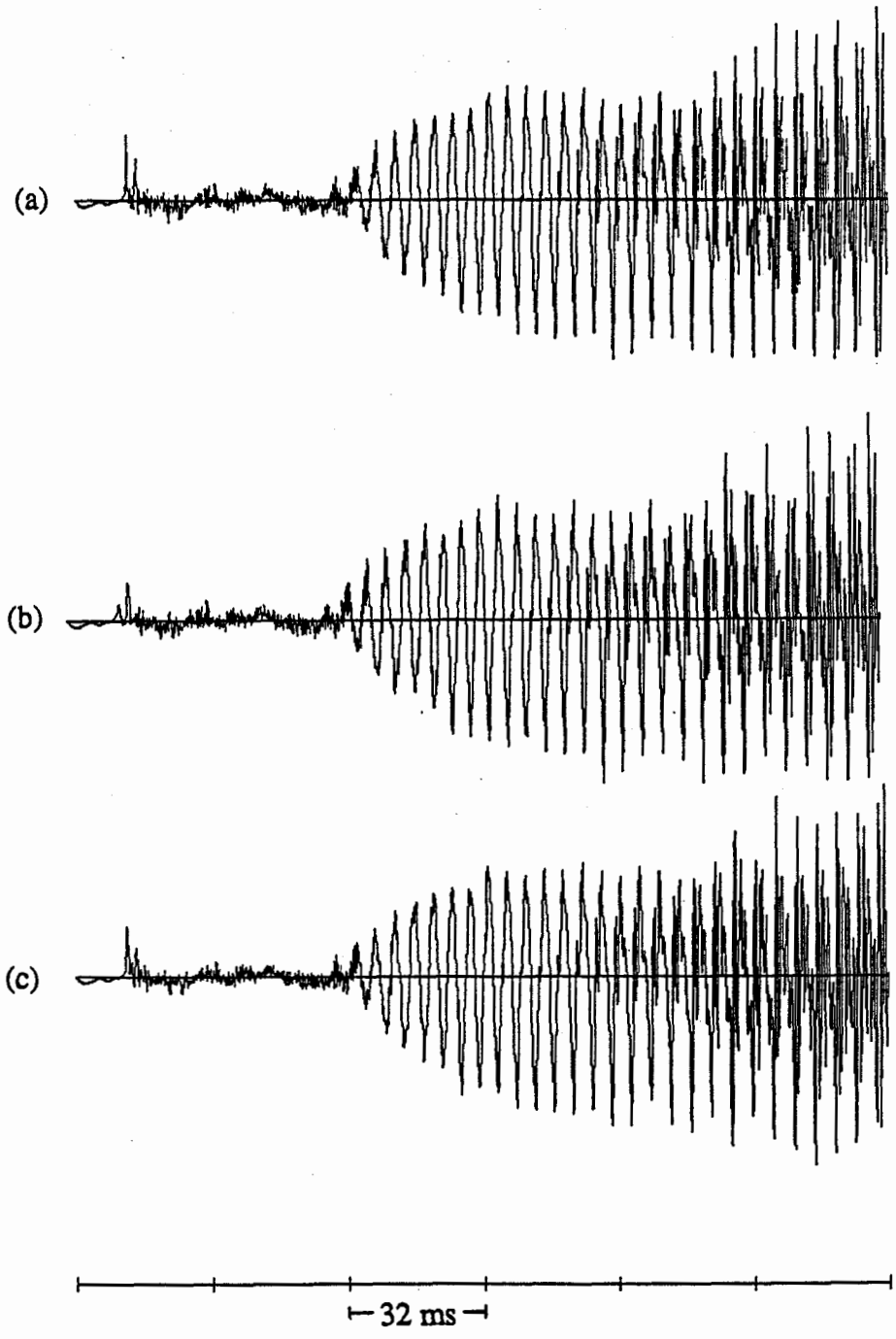

FIGURE 5.16. Coded speech waveforms at 16 kb/s. (a) Original signal. (b) scalar (gain-adaptive) ADPCM. (c) k=4 Vector (gain-adaptive) ADPCM.

5.5. Complexity Estimates

Complexity of the proposed algorithm and its variations has been estimated on the basis of the number of floating-point operations per second (flops) required for implementation. Table 5.9 shows a breakdown of the number of flops required for various tasks as a function of vector dimension k, codebook size $N=2^{2k}$, sampling rate f, total predictor order $P=p+z$, and ZSR update period μ in samples.

Algorithm	Task	No. of computations
Simple VQ	search	$2Nf$
	compare	$Nflk$
Simple VADPCM	predict first sample	$Pflk$
	predict next samples	$(k-1)(P+1)Nflk$
	search	$2Nf$
	compare	$Nflk$
	adapt predictor	$2Pf$
	stability check	$3f$
VADPCM, with	calc ZIR	$[\sum_{j=1}^{k-1}\sum_{\min(k,p)}(p+1-j)+\sum_{j=1}^{k-1}\sum_{\min(k,z)}(z+1-j)]flk$
ZIR	calc ZSR	$[k+\sum_{j=1}\min(j,p)+\sum_{j=1}\min(j,z)]Nflk$
	search	$2Nf$
	compare	$Nflk$
	adapt predictor	$2Pf$
	stability check	$3f$
VADPCM, with	calc ZIR	$[\sum_{j=1}^{k-1}\sum_{\min(k,p)}(p+1-j)+\sum_{j=1}^{k-1}\sum_{\min(k,z)}(z+1-j)]flk$
ZIR	calc ZSR	$[k+\sum_{j=1}\min(j,p)+\sum_{j=1}\min(j,z)]Nfl/\mu$
and ZSR update	search	$2Nf$
	compare	$Nflk$
	adapt predictor	$2Pf$
	stability check	$3f$
VADPCM, with	calc ZIR	$[\sum_{j=1}^{k-1}\sum_{\min(k,p)}(p+1-j)+\sum_{j=1}^{k-1}\sum_{\min(k,z)}(z+1-j)]flk$
ZIR,	calc ZSR	$[k+\sum_{j=1}\min(j,p)+\sum_{j=1}\min(j,z)]Nfl/\mu$
ZSR update,	search	$2Nf$
and gain-adapt.	compare	$Nflk$
	adapt predictor	$2Pf$
	stability check	$3f$
	mult by gain	f
	predict gain	$(3k+1)flk$

TABLE 5.9. Computational load of sub-tasks within VADPCM algorithms.

The above expressions have been evaluated for vector dimensions k=1 to k=4, using ZSR update period μ=48 *samples*, 2 poles, 6 zeroes, and a sampling rate f=8 *kHz*. The results are shown in Table 5.10.

k	VQ	VADPCM	VADPCM ZIR	VADPCM ZIR,ZSR	VADPCM ZIR,ZSR GAVQ
1	0.10	0.33	0.33	0.33	0.37
2	0.32	1.10	0.80	0.55	0.59
3	1.19	4.46	2.95	1.51	1.54
4	4.61	18.62	12.50	5.46	5.49

TABLE 5.10. Computational load of various VADPCM algorithms in Mflops/second.

Finally, the performance and the complexity of the proposed VADPCM algorithm, which includes precomputation of zero-input-response, periodic update of the Zero-state-response table, and gain-adaptive vector quantization, are shown together in Figure 5.17.

Clearly Vector ADPCM provides a valuable performance improvement over scalar ADPCM at 16 kb/s, although for vector dimensions greater than k=3, performance begins to saturate and the complexity approaches or exceeds the limits of state-of-the-art digital hardware.

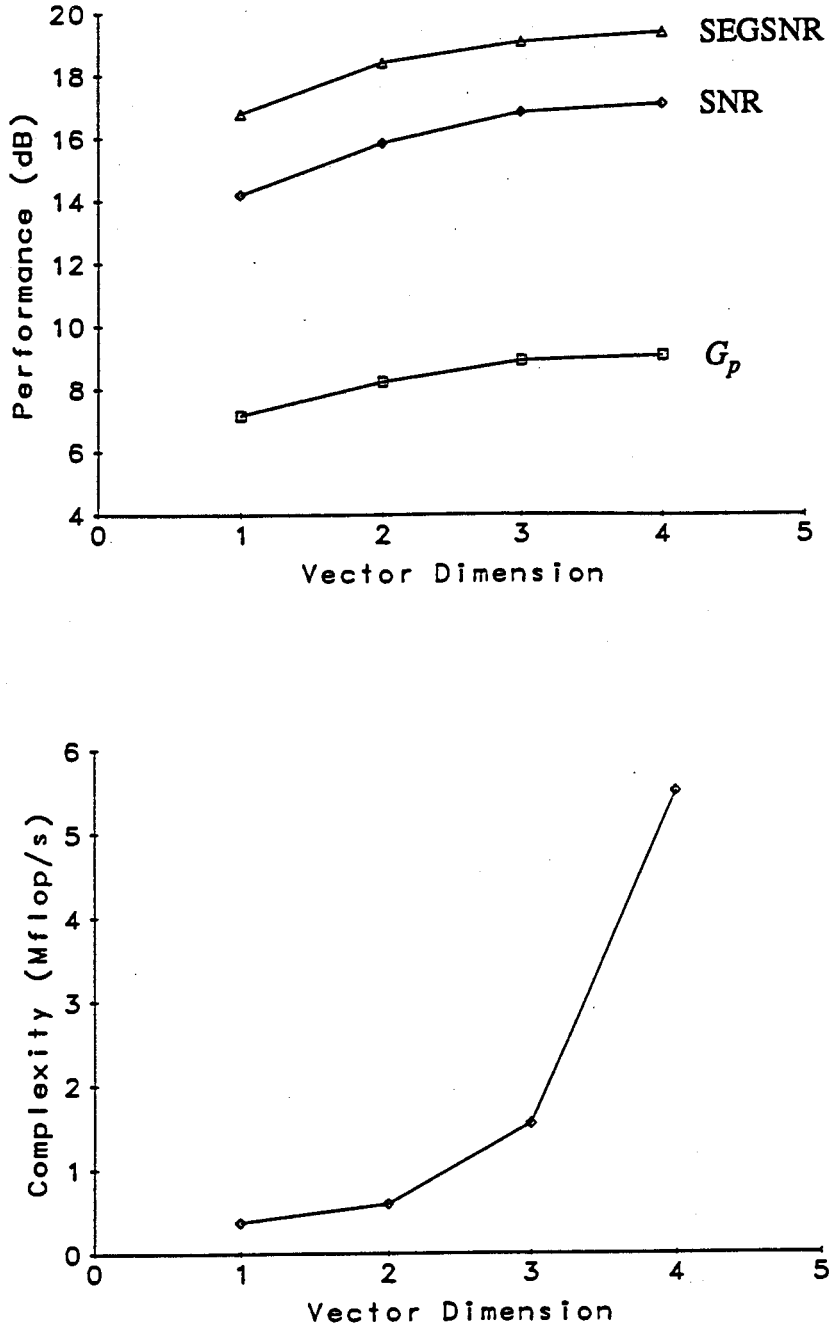

FIGURE 5.17. Effect of Vector Dimension on (a) Performance and (b) Complexity of Proposed Solution.

6. CONCLUSIONS.

Vector ADPCM at vector dimension k=4 has been shown to provide a 3 dB performance improvement over scalar ADPCM, with a factor of 15 increase in complexity, while still maintaining an encoding/decoding delay of less than 2 milliseconds. Adaptive postfiltering is found to improve the subjective quality of the coded speech to approximately the level of very good communications quality. Using the complexity reduction techniques described in this thesis, implementation of Vector ADPCM at k=3 using commercially available digital hardware is feasible. The complexity of the current algorithm at k=4 exceeds the limits of current state-of-the-art DSP chips.

In the context of the previous state of the art in low-delay waveform coding (scalar ADPCM), the present work has been successful in providing a significant performance improvement with a tolerable complexity increase. However, it should be noted that, even under ideal conditions, the proposed algorithm does not provide toll quality coded speech.

There are some steps which may be taken to improve the quality of the coded speech, while still maintaining the low delay and moderate complexity of the proposed algorithm. However, it should be noted that there are also many practical issues which are expected to degrade the performance of the proposed algorithm from its ideal performance.

The following strategies are expected to bring improvements to the ideal performance of the proposed algorithm:

1. The use of higher vector dimensions is expected to improve the performance, although a saturation in performance is evident already at k=3 and k=4. Since the complexity increases exponentially with vector dimension, such a strategy would have to include new complexity reduction techniques, i.e. structured codebooks. In order to ensure adequate training at high vector dimensions, very long training runs would be required.

2. Another technique for improving the performance of ADPCM algorithms is the use of pitch prediction, i.e. a long-term predictor which exploits the quasi-periodicity of voiced speech. While this technique has been found to provide a substantial improvement to scalar ADPCM under ideal conditions, it has been found to have poor performance in the presence of transmission errors [26].

Below is a list of practical issues which must be addressed before the proposed algorithm could be considered for application in the switched telephone network:

1. Compatibility with PCM frame format. The basic Vector ADPCM algorithm without the ZSR table update complexity reduction technique is completely backward-adaptive and requires no synchronization between transmitter and receiver, other than the fact that the transmitted codeword indices must be correctly received. For compatibility with the 8-bit PCM frame format, the preferred vector dimensions are 1, 2, and 4, with 2-bit, 4-bit, and 8-bit codeword indices respectively. The use of vector dimension 3 with a 6-bit codeword index would impose a 3-byte frame structure on the transmitted information which would require synchronization.

2. Synchronization of ZSR table update. The ZSR table update technique described in the present work assumes that the transmitter and receiver ZSR tables are updated at the same times. This will give the best performance, however, it may not be strictly necessary. It should be possible to simply allow an arbitrary update phase difference between transmitter and receiver, and eliminate the need for synchronization of the ZSR table update.

3 Real-time implementation issues. In the basic Vector ADPCM algorithm without the ZSR table update technique, the computational load is essentially constant - a fixed number of computations must be done for each input vector. However, the ZSR table update technique implies that the ZSR table is updated only once every μ/k vectors, and therefore, the computational load is uneven. Two implementation strategies are possible: In the first implementation strategy, a high-capacity DSP chip with an average processing power equal to the peak load may be used, however this is somewhat wasteful. In the

second strategy, the computational load may be distributed more evenly and a lower-capacity DSP chip may be used. In this strategy, a partial update of the ZSR table update would be performed at each input vector. This is not expected to degrade the performance of the proposed algorithm significantly.

4. Voice-band Data performance. In order to provide good performance with voice-band data signals, the training sequence would have to include DTMF tones, samples of modem signals, etc. Optimizing the codebook for this broader class of signals will result in degraded performance on speech alone.

5. Transmission error performance. The use of the CCITT predictor, which is known to have good performance in the presence of transmission errors, is expected to give the present algorithm some robustness in this regard. However, the effect of transmission errors on the gain predictor and the vector quantizer is not known and would have to be tested if any variation of the present work were to be considered for application in the switched telephone network. It is likely that some kind of Gray coding of the codewords would help in the event of single bit errors.

6. Synchronous transcoding with PCM and ADPCM. In order to allow synchronous transcoding with PCM and ADPCM, the vector quantizer must be designed to make a locally non-optimal decision, so that the distortion function *after transcoding* is minimized. This is expected to increase the complexity of the proposed algorithm and degrade its performance under non-transcoding conditions.

Modifying the present design to account for the above practical issues is expected to result in somewhat degraded performance on speech. In conclusion, the combination of vector quantization with scalar linear prediction has provided an improvement to the state of the art in low delay waveform coding of speech at 16 kb/s, however providing low delay, moderate complexity, toll quality speech at this data rate remains an unsolved problem.

LIST OF REFERENCES

References

1. Abut, H., Gray, R. M., and Rebolledo, G., "Vector Quantization of Speech and Speech-Like Waveforms," *IEEE Trans. on ASSP*, vol. ASSP-30, pp. 423-436, June. 1982.
2. Atal, B. S. and Hofacker, R. Q., Jr., *The telephone voice of the future,* pp. 4-10, Bell Laboratories Record, Bell Telephone Laboratories, July, 1985.
3. Atal, B. S. and Schroeder, M. R., "Code-Excited Linear Prediction (CELP): high-quality speech at very low bit rates," *Proc. 1985 ICASSP*, pp. 25.1.1-4.
4. Atal, B. S. and Schroeder, M. R., "Adaptive Predictive Coding of Speech Signals," *Bell System Technical Journal*, vol. 49, pp. 1973-1986, Oct. 1970 .
5. Bellanger, M., *Digital Processing of Signals,* Wiley, 1984.
6. Box, G. E. and Jenkins, G. M., *Time Series Analysis: Forecasting and Control,* Holden-Day, 1970.
7. Buzo, A., et al., "Speech coding based upon Vector Quantization.," *IEEE Trans. on ASSP*, vol. ASSP-28, pp. 562-574, Oct. 1980.
8. CCITT Recommendation G.712, "Line Transmission," *CCITT Orange Book, Fascicle III.2*, Sept. 1976.
9. CCITT COM XVIII-R10,, "Report on the meeting held in Geneva (18-22 June 1982)," *source: Working Party XVIII/2 (Speech Processing), Geneva*, July 1982.
10. CCITT Recommendation G.721, "32 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)," *CCITT Red Book, Fascicle III.3*, pp. 55-93, October 1984.
11. Chan, S. and Cuperman, V., "The Noise Performance of a CELP speech coder based on Vector Quantization," *Canadian Conference on Electrical and Computer Engineering, 1988*, pp. 795-798.
12. Chen, J.-H. and Gersho, A., "Vector Adaptive Predictive Coding of Speech at 9.6 kb/s," *Proc. 1986 ICASSP*, pp. 33.4.1-4.

13. Chen, J.-H. and Gersho, A., "Gain-Adaptive Vector Quantization with Application to Speech Coding," *IEEE Trans. on Comm.*, vol. COM-35, no. 9, pp. 918-930, Sept. 1987.

14. Crochiere, R. E., "Sub-band Coding," *Bell System Technical Journal*, vol. 60, no. 7, pp. 1633-1653, Sept. 1981.

15. Cuperman, V., *Efficient Waveform Coding of Speech using Vector Quantization*, Ph.D. Dissertation, University of California, Santa Barbara, February, 1983.

16. Cuperman, V. and Gersho, A., "Adaptive differential vector coding of speech," *Conf. Rec. IEEE GLOBECOM*, pp. 1092-1096, Dec. 1982.

17. Cuperman, V. and Gersho, A., "Vector Predictive Coding of Speech at 16 kbit/s," *IEEE Trans. on Comm.*, vol. COM-33, pp. 685-696, July. 1985.

18. Durbin, J., "Efficient Estimation of Parameters in Moving-Average Models," *Biometrika*, vol. 46, pp. 306-316, 1959.

19. Flanagan, J. L., et al., "Speech Coding," *IEEE Trans. on Comm.*, vol. COM-27, pp. 710-737, April. 1979.

20. Gersho, A., "On the structure of vector quantizers," *IEEE Trans. on Inf. Theory*, vol. IT-28, pp. 157-166 , March 1982.

21. Gray, R. M., "Vector Quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4-29, April, 1984..

22. Gray, R. M. and Linde, Y., "Vector Quantization and Predictive Quantizers for Gauss-Markov Sources," *IEEE Trans. on Comm.*, vol. COM-30, pp. 381-389, Feb. 1982.

23. Hoth, D. F., *The T1 Carrier System,* Bell Laboratories Record, Bell Telephone Laboratories, 1962.

24. Jayant, N. S., "Digital Coding of Speech Waveforms: PCM, DPCM, and PM Quantizers," *Proc. IEEE*, vol. 62, pp. 611-632, May 1974.

25. Jayant, N. S., "ADPCM Coding of speech with backward-adaptive algorithms for Noise Feedback and Postfiltering," *Proc. 1987 ICASSP*, pp. 29.10.1-4.

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

27. Jayant, N. S. and Ramamoorthy, V., "Adaptive Postfiltering of 16 kb/s ADPCM Speech," *Proc. 1986 ICASSP*, pp. 16.4.1-4.

28. Joel, A. E., Jr., et al., *A History of Engineering and Science in the Bell System* , Bell Telephone Laboratories, 1982.

29. Keiser, B. E. and Strange, E., *Digital Telephony and Network Integration,* Van Nostrand Reinhold , 1985.

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

31. Lloyd, S. P., "Least Squares Quantization in PCM's," *Bell Telephone Laboratories Paper, Murray Hill, N.J.*, 1957 .

32. Marden, M., *Geometry of Polynomials,* American Mathematical Society, 1966.

33. Max, J., "Quantizing for minimum distortion," *IRE Trans. on Inf. Theory*, vol. IT-6, pp. 7-12 , March 1980.

34. Millar, D. and Mermelstein, P., "Prevention of Predictor Mistracking in ADPCM Coders ," *Proc. Int. Conf. Comm.*, pp. 1508-1512, May 1984.

35. Millman, J., *Microelectronics,* McGraw-Hill, 1979.

36. Nishitani, T., et. al., "A 32 kb/s Toll Quality ADPCM Codec using a Single Chip Signal Processor," *Proc. 1982 ICASSP*, pp. 960-963.

37. Sabin, M. J. and Gray, R. M., "Product code vector quantizers for waveform and voice coding," *IEEE Trans. on ASSP*, vol. ASSP-32, pp. 474-488, June 1984.

38. Shannon, C. E., "A mathematic theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656, 1948 .

39. Tribolet, J. M., et. al., "A Comparison of the performance of four low-bit-rate speech waveform coders," *Bell System Technical Journal*, vol. 58, no. 3, pp. 699-712, March 1979.

40. Widrow, B., et al., "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," *Proc. IEEE*, vol. 64, no. 8, pp. 1151-1161, Aug. 1976.