Predictive Split Vector Quantization for Speech Coding

by

Michael Soong

Department of Electrical Engineering

McGill University

Montreal, Canada

March, 1994

Abstract

The purpose of this thesis is to examine techniques for efficiently coding speech Linear Predictive Coding (LPC) coefficients. Vector Quantization (VQ) is an efficient approach to encode speech at low bit rates. However its exponentially growing complexity poses a formidable barrier. Thus a structured vector quantizer is normally used instead.

Summation Product Codes (SPCs) are a family of structured vector quantizers that circumvent the complexity obstacle. The performance of SPC vector quantizers can be traded off against their storage and encoding complexity. Besides the complexity factors, the design algorithm can also affect the performance of the quantizer. The conventional generalized Lloyd's algorithm (GLA) generates sub-optimal codebooks. For a particular SPC such as multistage VQ, the GLA is applied to design the stage codebooks stage-by-stage. Joint design algorithms on the other hand update all the stage codebooks simultaneously.

In this thesis, a general formulation and an algorithm solution to the joint codebook design problem is provided for the SPCs . The key to this algorithm is that every SPC has a reference product codebook which minimizes the overall distortion. This joint design algorithm is tested with a novel SPC, namely "Predictive Split VQ (PSVQ)".

VQ of speech Line Spectral Frequencies (LSF's) using PSVQ is also presented. A result in this work is that PSVQ, designed using the joint codebook design algorithm requires only 20 bits/frame(20 ms) for transparent coding of a 10^{th} order LSF's parameters.

Sommaire

L'objectif de cette thèse est d'étudier des techniques efficaces de codage de coefficients à prédiction linéaire (CPL). La quantification vectorielle (VQ) est l'approche la plus efficace pour compresser la parole à un débit aussi faible, mais sa complexité exponentiellement croissante empêche son application directe. Ainsi, on a normalement recours à des procédures sous-optimales de quantification vectorielle structurée, qui permettent de troquer la performance du codage contre une réduction en complexité.

On distingue habituellement deux facteurs de complexité: la complexité de codage (effort de calcul), et la complexité de stockage (capacité de mémoire). En faisant varier ces facteurs, on obtient différents types de quantificateurs vectoriels structurés: MSVQ, SVQ, $\cdots$ Mais la performance d'un code dépend aussi de l'algorithme de construction du code. L'algorithme généralisé de Lloyd fournit normalement des livres de code sous-optimaux, parce que la construction du livre de code se fait de façon itérative.

D'un autre côté, l'algorithme "conception jointe, mise-à-jour jointe" essaie de mettre à jour tous les livres de code en même temps, en résolvant une équation matricielle. La clef de voûte de cet algorithme est que chaque code-produit SPC a un livre de code-produit de référence qui minimise la distorsion totale. Cette conception jointe du livre de code est appliquée à une famille de quantificateurs vectoriels structurés, les codes-produit de sommation (SPC). En particulier, on teste cet algorithme avec un quantificateur vectoriel structuré SPC particulier" *Predictive Split VQ* (PSVQ).

Des résultats avec des quantificateurs à Lignes de Fréquences Spectrales (LSF) sont aussi présentés. Une des conclusions importantes de ce travail est qu'un quantificateur PSVQ, conçu avec l'algorithme de conception jointe.

Acknowledgements

I would like to thank my supervisors Dr. Wai-Yip Chan and Dr. Peter Kabal for their guidance throughout my graduate studies. I would also like to acknowledge the financial support provided by Dr. Peter Kabal for my Master's studies.

This thesis could not have been completed without the constant support of my parents. Special thanks go as well to Lana for her affection and her understanding. I am also grateful for the help provided by Marc Kimpe, Jean-Paul Chaib and my friends in McGill.

Contents

List of Figures

Chapter 1

Introduction

Digital speech coding is a subject that has received tremendous amount of attention. The goal of speech coding is to transmit speech with the highest possible quality at a given bit rate or to minimize the bit rate at a given distortion level. Typically, the efficiency of speech coders is positively correlated with coder complexity. Finding the best tradeoff between coded speech quality and complexity is a key issue in speech coder design.

There are two basic classes of coders, waveform coding and model based coding. Waveform coding tries to match the incoming signal waveform with the coded signal as accurately as possible. Model-based coders on the other hand parameterize the input signal using models which characterize the human speech production mechanism and/or the human auditory system. Key parameters of the model are being transmitted to the decoder end so that the speech signal can be reconstructed using the same production model. Model based coders can significantly reduce the bit rate by exploiting the redundancies of speech while maintaining high quality.

One of the most widely used techniques in speech coding for representing the short-time spectral envelope information of speech is linear prediction or all-pole synthe-

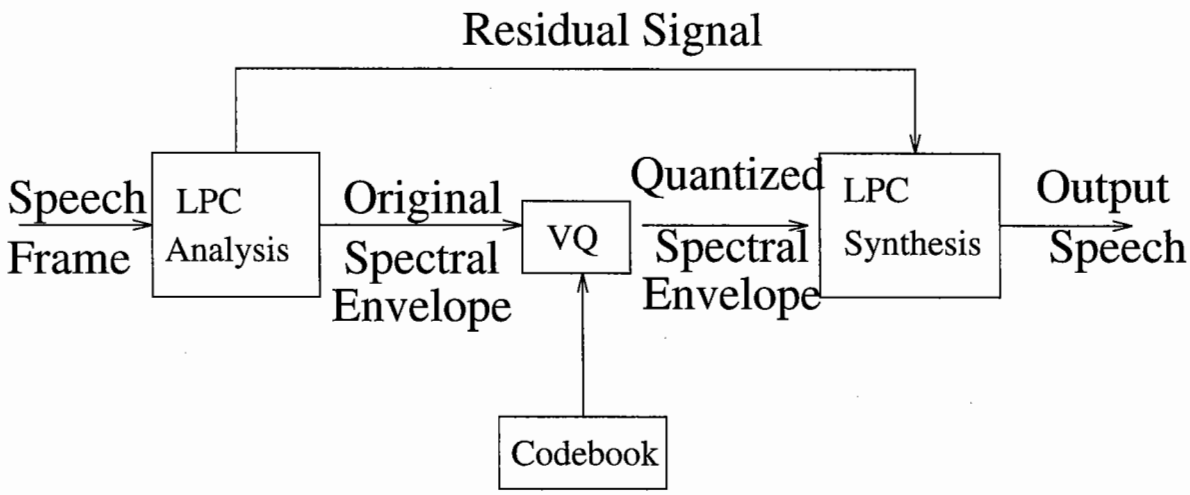


Figure 1.1: Simulation model for vector quantization of LPC coefficients

sis. In a typical application, the input speech samples are passed through a preemphasis filter. The filter output sequence is parsed into frames of 10 to 30ms each. Each frame consists of 100 to 300 samples, depending on the sampling rate. An autocorrelation sequence is then computed from each frame, and then the linear prediction coefficients (LPC) are obtained using Levinson Durbin recursion [28]. After obtaining the LPC coefficients, the frame of speech samples can be processed by the prediction filter to obtain a residual signal. The different natures of the residual signal and the LPC coefficients result in different strategies for coding the two.

Since only the coding of LPC coefficients is investigated in this work, the residual is passed directly to the receiver without any coding and thus without any degradation. Therefore, the effects of quantizing the LPC coefficients can be isolated from the effects of coding the residual signal. Fig(1.1) shows a simulation model for the coding of LPC coefficients. The LPC Analysis block determines the LPC coefficients and inverse filter the input speech signal to produce the residual signal. The vector quantization (VQ) block codes the LPC coefficients. The quantized LPC coefficients are then given to the LPC Synthesis block which uses them to filter the residual signal and produce the output speech.

In narrow band coding of speech sampled at 8 kHz, a prediction order of ten is typically used. The order of an LPC model refers to the number of coefficients in linear prediction. It has been shown that a smaller order starts to have large increase in error while increasing the order to greater than 10 does not improve the performance of the quantization of the parameters [29]. However, the LPC coefficients are known to be inappropriate for quantization because of their relatively large dynamic range and their complex relationship to filter stability. Various sets of LPC parameters representing the same spectral information have been studied, e.g. cepstral coefficients and reflection coefficients. The line spectral frequency (LSF) representation, proposed by Itakura [9], has both a well-behaved dynamic range and a filter stability preservation property. It can be used to encode the spectral information of speech more efficiently. Any assessment of signal quality implies a fidelity measure. For most communication systems, this measure is difficult to specify quantitatively because it involves human perception. Extensive perceptual performance testing of speech coders is time consuming while quick comparisons are required in the early stages of design. An earlier study [12] produced objective criteria that characterizes transparent coding quality, (i.e., there is no distinguishable difference between the quantized LPC parameters and the original unquantized version) for LSF parameters.

For low bit rate speech coding applications, the goal is to achieve transparent coding quality using the least possible number of bits to quantize the parameters. Vector quantization has been proven to be a highly effective technique for minimizing the bit rate. It considers the entire set of LPC parameters as a vector and allows for the direct minimization of quantization distortion. The disadvantage of using unstructured VQ for transparent coding of LPC information is that approximately 20 bits are needed to quantize one parameter vector. This poses an insurmountable search and memory complexity and an inordinate amount of training data. One method to correct this complexity barrier is by imposing a special structure on the quantizer at the cost of higher distortion. Some of the most well-known structures are tree-search VQ (TSVQ),

multi-stage VQ (MSVQ), split VQ (SVQ). They can be grouped into a class of structured vector quantizers called Summation Product Code (SPC). In the conventional training process for summation structured VQ, each stage is designed independently using the Generalized Lloyd Algorithm. A new joint-codebook design algorithm was recently proposed [35].

1.0.1 Contribution of the thesis

The main intention of this thesis is to present a survey of the Summation Product Code family and examine the efficiency and characteristics of the joint-design algorithm on some of the members of the SPC family. A novel SPC structure is also presented. This structure, called Predictive Split VQ, is a hybrid of MSVQ and SVQ. The new PSVQ configuration is evaluated by using it to quantize speech LSF parameters.

1.0.2 Organization of the thesis

This thesis is divided into six chapters. Chapter two reviews the theoretical background of the linear prediction model. The line spectral frequencies representation is examined. Several distortion measures are also discussed.

Chapter three examines the quantization theory. The theory of scalar quantization is studied and is then extended to vector quantization. After the principles are reviewed, different summation product code structures are presented together with their various degrees of storage and encoding complexity.

Chapter four reviews the conventional Generalized Lloyld codebook design algorithm for SPC design. In this chapter, a new joint-codebook design is introduced. The example of a joint design MSVQ method is described.

In the first part of Chapter five, the effectiveness of the joint-codebook design

algorithm on summation product code structures are evaluated for Gaussian sources. In addition, a new summation product code structure, Predictive Split VQ, is presented. In the second part of the chapter, the performance of PSVQ and Split VQ for quantizing LSF parameters is investigated.

Finally, Chapter six presents a summary on the experimental results and indicates areas for further investigation.

Chapter 2

Linear Predictive Coding Of Speech

2.1 Introduction

A source-production model for speech depends on a parametric description of the vocal-tract transfer function. This parametric description can take a variety of forms, for example, values of the short-time amplitude spectrum of the speech signal evaluated at specific frequencies as done in a channel vocoder; linear prediction coefficients that describe the spectral envelope as used in Code-Excited Linear Predictive (CELP) Coding [28]; and frequency values of major spectral resonances, commonly used in formant coders, etc.

Linear predictive coding is widely used among today's low-bit-rate speech coders. This popularity stems from the fact that the linear prediction model defines an "all-pole" synthesis spectrum that matches very well the short-term spectral envelope of speech. The basic idea in linear prediction is to approximate a speech sample as a linear combination of past speech samples. By minimizing the mean square error between the original speech and the prediction, a unique set of predictor coefficients can be determined. Considerable work has been done in the past to develop quantization procedures, both

scalar and vector, to represent this information with a small number of bits. However, linear prediction coefficients have some major disadvantages with respect to maintaining the stability of the synthesis filter and catering to the distribution of the parameter values. As a result, an equivalent representation is often used instead. For example, Viswanathan and Makhoul[38] used the log area ratios (LARs) for scalar quantization of LPC parameters. Gray and Markel [17] used arcsine reflection coefficients for the same purpose. Itakura [10] proposed the line spectral frequency (LSF) representation which has been shown to be an efficient representation for scalar quantization of LPC information [29]. Paliwal and Atal [15] used split VQ with LSF parameters and obtained "transparent coding quality" at 24 bits/frame. Paksoy et al. [16] used LSF parameters for VQ and achieved transparent quality at 21 bits/frame.

The choice of distortion measure is another important factor in designing a speech coder. A distortion measure can be used to guide the quantization search and to evaluate the performance of a speech coding systems. Even though the sound quality of a given speech coder is best evaluated by the human ear, this is time consuming and quick comparisons are required in the early design phase. Different distortion measures will be discussed in this chapter; however, emphasis is placed on measures that preserve perceptually important speech-model parameters, such as spectral distortion measures.

2.2 The Characteristics of Speech Signals

Source coding reduces the bit rate by exploiting the natural redundancies that exist in the source signal. Therefore, an understanding of human speech communication system before investigating any kind of speech modelling is beneficial. Speech communication involves two processes: the speech generation process and auditory process. In the speech generation process, an excitation flow is generated by forcing air from the lung through the vocal cords into the vocal tract. This excitation flow can either be (quasi-

)periodic or randomly fluctuating (turbulent). To generate a voiced sound, (such as /a/,/e/,/i/,/o/,/u/) , the air flow is interrupted (quasi-)periodically by the vocal fold. The opening and closing of the vocal tract determines the fundamental frequency or the pitch. Moreover, this excitation signal of strongly periodic nature has smooth glottal waveform transitions most of the time. The vocal tract, starting from the vocal cord, and ending at the lips, can be modelled as an acoustic tube. By varying the shape of this tube, different sounds can be produced. This vocal tract acts as a filter, and amplifying energy near the formant frequencies, while attenuating frequencies between the formants.

An unvoiced sound (such as /k/,/p/,/f/,/v/), on the other hand, is generated at a narrow constriction in the vocal tract and the vocal cords do not vibrate. The spectrum of an unvoiced fricative is typically broad in bandwidth with gentle attenuation at the band edge. For coding purposes, the excitation source for voiced sounds can be represented by a periodic pulse generator. The source for unvoiced sounds can be represented by a random noise generator.

In speech auditory processing, the middle ear acts as a low pass filter with a cutoff frequency of 1500-2000 hertz, and a gain ratio of 15 to 1. The inner ear has thousands of sensory hair cells attached to the basilar membrane. The studies of human ear sensitivity have shown that the human ear is more sensitive at low frequency than high frequency. Moreover, it is found that the ear is more sensitive to the peaks than valleys of the power spectrum.

The intent of the brief description of the human speech production and auditory apparatus [28] is a motivation for predictive coding. By appropriately modeling the glottal excitation and the vocal track with only a few parameters to be transmitted, substantial bit savings can be achieved. Furthermore, by exploiting the limitations and properties of the human auditory system such as masking phenomena, increased sensitivity to lower frequencies and the insignificance of spectral zeros, the perceived

speech quality can dramatically be improved.

2.3 Linear Prediction Model

The speech model used in Linear Predictive coding is based on the discrete speech signal $x[n]$ being the output of a system with an input $u[n]$. The model can be represented mathematically as follows,

$$x[n] = -\sum_{k=1}^{M} a_k x[n-k] + G\sum_{l=0}^{N} b_l u[n-l], \quad b_0 = 1. \tag{2.1}$$

where $a_k, 1 \leq k \leq M, b_k, 1 \leq l \leq N$ and the gain G are the parameters of the system. The model shows that the output speech samples, $x[n]$, is a linear combination of past speech samples as well as past and present input signals $u[n]$. By taking the z-transform on both sides of Eq(2.1), the transfer function of the system $H(z)$ can be represented as

$$\begin{aligned} H(z) &= \frac{X(z)}{U(z)} \\ &= G\frac{1+\sum_{l=1}^{M} b_l z^{-l}}{1+\sum_{k=1}^{M} a_k z^{-k}}. \end{aligned} \tag{2.2}$$

where $X(z)$ and $U(z)$ are the z-transform of $x[n]$ and $u[n]$ respectively. $H(z)$ in Eq(2.2) is the general pole-zero model, known also as the Auto-Regressive Moving Average(ARMA) model. For simplicity, most speech coders only consider the poles from the model and simplify Eq(2.2) to

$$H(z) = G\frac{1}{1+\sum_{k=1}^{M} a_k z^{-k}} \tag{2.3}$$

Here M is the order of LPC analysis. The all-pole synthesis model of linear prediction considers only the past output speech samples and the present input samples. It reduces the amount of computation required to determine the filter coefficients but the simplification has a major disadvantage as the zeroes in the vocal/nasal tract response and the glottal source are neglected. The effect of neglected zeroes is reduced as the number of poles used increases. The all-pole model is a desirable choice also due to the fact that

human ear sensitivity is higher at spectral formants where the poles are located and lower at spectral valleys where zeroes are located. Assuming the input $u[n]$ to be totally unknown, the signal $x[n]$ can be predicted approximately by a linear combination of past samples. If the predicted value of the signal is denoted by $\hat{x}[n]$ where

$$\hat{x}[n] = -\sum_{k=1}^{M} a_k x[n-k] \tag{2.4}$$

then the error between the actual value $x[n]$ and the predicted value $\hat{x}[n]$ is given by

$$e[n] = x[n] - \hat{x}[n] = x[n] + \sum_{k=1}^{M} a_k x[n-k] \tag{2.5}$$

As a result, a speech model can be derived from the above equation. With an excitation signal $e[n]$ passing through a shaping filter $H(z)$ (Fig(2.1)), the reconstructed speech $x[n]$ is reproduced. $H(z)$ is known as the synthesis filter and can be expressed as

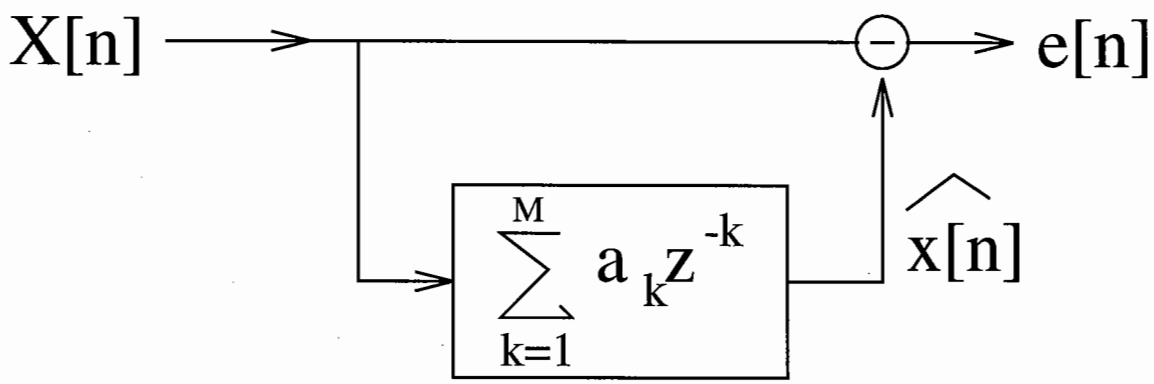
$$H(z) = \frac{1}{1 + \sum_{k=1}^{M} a_k z^{-k}} = \frac{1}{A(z)} \tag{2.6}$$

where

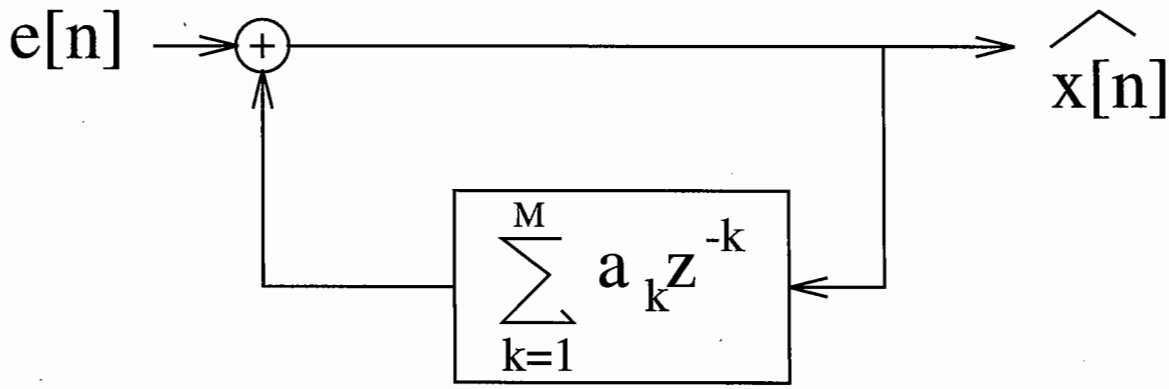
$$A(z) = 1 - \sum_{k=1}^{M} a_j z^{-k} \tag{2.7}$$

is the inverse filter to remove the formant structure from the original speech file. If the prediction system is based on past original speech samples, it is referred as forward adapted prediction because the predictor coefficients have to be sent to the receiver as side information. However, if the prediction system is based on past reconstructed speech samples, then it is called backward adaptive prediction and no side information is transmitted. The backward adaptive predictor coefficients can be calculated both at the transmitter and the receiver end.

The predictive coefficients a_k can be obtained as a result of minimizing the mean square error with respect to each of the parameters, using the least square method.



Prediction Model.



Synthesis Model

Figure 2.1: Predictive filter and synthesis filter

2.4 Linear Predictor Coefficients

One of the major tasks in linear predictive coding is to obtain the predictive coefficients and the gain in order to minimize the error energy. The speech signal itself is not stationary and its statistics are not usually known. It is commonly assumed that the speech signal is stationary over a short interval of time (about 20ms). The predictor coefficients can thus be estimated from a sequence of speech samples obtained from an interval over which the signal is considered to be stationary.

Windowing the sampled signal is the first step in any linear prediction parameter calculation. Several techniques have been developed to compute the predictor coefficients. Among these, two classical methods are commonly used: the Autocorrelation method and the Covariance method.

2.4.1 Autocorrelation method

Autocorrelation least-square technique multiplies the speech signal by a time window, typically a Hamming window before filtering it using the inverse filter $A(z)$ defined in previous section.

$$x[n] = w[n]s[n] \tag{2.8}$$

The window limits the speech signal to a finite interval, $0 \leq n \leq N-1$. The energy in the residual signal is then

$$E = \sum_{n=-\infty}^{\infty} e^2[n] = \sum_{n=-\infty}^{\infty} \left[x[n] - \sum_{k=1}^{M} a_k x[n-k] \right]^2 \tag{2.9}$$

The least-square method minimizes the energy by differentiating the energy with respect to $a_k, k = 1, 2, \ldots, M$, and setting the equation to zero.

$$\frac{\delta E}{\delta a_k} = 0 \tag{2.10}$$

The resulting equation becomes

$$\sum_{n=-\infty}^{\infty} x[n-i]x[n] = \sum_{k=1}^{M} a_k \sum_{n=-\infty}^{\infty} x[n-i]x[n-k], i = 1, 2, \ldots M. \tag{2.11}$$

The autocorrelation function of the time-limited signal $x[n]$ is defined as

$$R(i) = \sum_{n=i}^{N-1} x[n]x[n-i], \quad i = 1, 2, \ldots M. \tag{2.12}$$

The term $R(0)$ is equal to the energy in x[n]. It should be noted that $R(i)$ is an even function such that

$$R(i) = R(-i) \tag{2.13}$$

Substituting the autocorrelation function into Eq(2.11), the system of equations can be expressed in matrix form as $Ra = r$. The expanded form of the system is

$$\begin{bmatrix} R(0) & R(1) & \ldots & R(M-1) \\ R(1) & R(0) & \ldots & R(M-2) \\ \vdots & \vdots & & \vdots \\ R(M-1) & R(M-2) & \ldots & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(M) \end{bmatrix} \tag{2.14}$$

where each entry R_{ij} in the matrix R is given by $R_{ij} = R(|i-j|)$. The matrix is symmetric and Toeplitz and Eq(2.14) is in fact the Yule-Walker equation. This equation can be solved by using the Levinson-Durbin recursion [28] method.

2.4.2 Covariance method

The covariance method, on the other hand, determines the predictor coefficients by windowing the error signal $e[n]$ rather than the speech signal $x[n]$. As a result, the error in the residual signal becomes

$$E = \sum_{n=-\infty}^{\infty} e^2[n]w[n] \quad = \sum_{n=-\infty}^{\infty} \left[s[n] - \sum_{k=1}^{M} a_k s[n-k] \right]^2 w[n] \tag{2.15}$$

The error is minimized over a finite interval $0 \leq n \leq N-1$ as determined by a rectangular window function $w[n]$ to reduce Eq(2.15) to

$$E = \sum_{n=0}^{N-1} \left[s[n] - \sum_{k=1}^{M} a_k s[n-k] \right]^2 w[n] \tag{2.16}$$

Differentiating the residual energy with respect to $a_k, k = 1, 2, \ldots, M$ and setting the equation to zero will result in the set of equations given by

$$\sum_{n=0}^{N-1} s[n-i]s[n] = \sum_{k=1}^{M} a_k \sum_{n=0}^{N-1} x[n-k]x[n-i] \tag{2.17}$$

where $i, k = 1, \ldots, M$. The covariance function of $x[n]$ is defined by

$$\phi(i,k) = \sum_{n=0}^{N-1} s[n-k]s[n-i] \tag{2.18}$$

Substituting the covariance function into Eq(2.16), the system of equation can be expressed in matrix form as $\Phi a = \phi$ or

$$\begin{bmatrix} \phi(1,1) & \phi(1,2) & \ldots & \phi(1,M) \\ \phi(2,1) & \phi(2,2) & \ldots & \phi(2,M) \\ \vdots & \vdots & & \vdots \\ \phi(M,1) & \phi(M,2) & \ldots & \phi(M,M) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} \phi(0,1) \\ \phi(0,2) \\ \vdots \\ \phi(0,M) \end{bmatrix} \tag{2.19}$$

The covariance matrix is also a symmetric but not Toeplitz and can be solved by using the Cholesky decomposition method. One major requirement for the predictor coefficients is dictated by the stability of the synthesis filter. Stability of the synthesis filter is guaranteed by having all the zeros of the inverse filter $A(z)$ inside the unit circle in the z-domain, i.e. minimum phase. The autocorrelation method always results in a stable synthesis filter. The covariance method, unfortunately, does not guarantee a minimum phase inverse filter though it may result in better performance.

2.4.3 Reflection coefficients

The reflection coefficients are an alternative representation for the LPC Coefficients. They arise as intermediate variables when solving for predictor coefficients in the auto-

correlation method, using what is called Levinson-Durbin recursion. The procedure is as follows: for $i = 1, 2, 3, \ldots, M$,

$$E_0 = R(0)$$

$$k_i = \frac{R(i) - \sum_{k=1}^{i-1} a_{i-1}(k) R(i-k)}{E_{i-1}},$$

$$a_i(i) = k_i,$$

$$a_k(i) = a_k(i-1) - k_i a_{i-k}(i-1), k = 1, 2, 3, \ldots, i-1,$$

$$E_i = (1 - k_i^2) E_{i-1} \tag{2.20}$$

The reflection coefficients are the k_i's while the predictor coefficients are

$$a_k = a_k(M), \quad k = 1, 2, 3, \ldots, M$$

The reflection coefficients can be computed directly from the predictor coefficients using the Schur Cohn procedure. The reflection coefficients can be used to construct the lattice form of the inverse filter $A(z)$. The lattice filter corresponds to an acoustic tube model of the vocal tract with the k_i coefficients representing the reflection coefficients at the boundaries of the tube sections. An important property of the reflection coefficients is that if their magnitudes are less than unity, then the inverse filter is minimum phase and invertible. If the predictor coefficients are known, the reflection coefficients can be determined by the following recursion for $m = M, M-1, \ldots, 3, 2$

$$a_k = a_M(k),$$

$$a_{m-1}(i) = \frac{a_m(i) + k_m a_m(m-i)}{1 - k_m^2}, \quad i = 1, 2, \ldots, m-1 \tag{2.21}$$

$$k_{m-1} = a_{m-1}(m-1)$$

With this recursion, predictor coefficients can be checked for stability by converting them to reflection coefficients.

2.5 Line Spectral Frequencies

Line Spectral Frequencies (LSF's) are another representation of the LPC coefficients which have been shown to be useful for speech coding. The distributions of the LSF frequencies are plotted in Fig(2.2) as histograms. It is clear that the distribution range varies from one LSF histogram to another [11]. In order to define the LSF's, the inverse filter polynomial, $A(z) = 1 - a_1 - \ldots - a_M z^{-M}$, is used to construct two polynomials.

$$P(z) = A(z) + z^{-(M+1)}A(z^{-1}) \tag{2.22}$$

$$Q(z) = A(z) - z^{-(M+1)}A(z^{-1}) \tag{2.23}$$

The roots of $P(z)$ and $Q(z)$ uniquely characterize the LPC filter and are called LSFs. The transformation from linear predictive coefficient to LSF parameters is reversible. The LSF's corresponds to the angular position of the roots of Q(z) and P(z). The roots all lie on the unit circle and occur in complex-conjugate pairs. There are M LSF's lying between 0 and π. In addition, there are two extraneous roots at $\omega = 0$ and $\omega = \pi$. Three methods of calculating the LSF's parameter are presented in the following:

The first method is introduced by Kang and Fransen [11]. They use an iterative approach to find the roots on the unit circle based on the all-pass ratio filter

$$R(z) = \frac{z^{-(M+1)}A(z^{-1})}{A(z)} \tag{2.24}$$

The phase spectrum of this filter is then determined from Eq(2.24). The LSF's correspond to the frequencies where the phase response takes on values which are multiples of π. A second approach, also proposed by Kang and Fransen [11], makes use of two polynomials to determine the LSF's.

$$G(z) = \frac{P(z)}{1 + z^{-1}} \quad \text{and} \quad H(z) = \frac{Q(z)}{1 - z^{-1}}, \quad \text{M even} \tag{2.25}$$

$$G(z) = P(z) \quad \text{and} \quad H(z) = \frac{Q(z)}{1 - z^{-2}} \quad \text{M odd.} \tag{2.26}$$

These polynomials are of degree $2M$ and may be expressed in terms of their coefficients as

$$G(z) = \sum_{i=0}^{2l}(g_i z^{-1} + g_{l-i} z^{-(l+i)}), \quad g_0 = 1, \tag{2.27}$$

$$H(z) = \sum_{i=0}^{2m}(h_i z^{-1} + h_{m-1} z^{-(m+i)}), \quad h_0 = 1, \tag{2.28}$$

where $l = m = M/2$ for M even and $l = (M+1)/2, m = (M-1)/2$ for M odd. The polynomials $G(z)$ and $H(z)$, by separating their linear phase components can be expressed as

$$\begin{aligned} G(z) &= e^{-j\omega l} G'(\omega) \\ H(z) &= e^{-j\omega l} H'(\omega) \end{aligned} \tag{2.29}$$

where

$$\begin{aligned} G'(\omega) &= 2\textstyle\sum_{i=0}^{l} g_i cos(l-i)\omega \\ H'(\omega) &= 2\textstyle\sum_{i=0}^{l} h_i cos(m-i)\omega \end{aligned} \tag{2.30}$$

The second method proposed by Soong and Juang[29] determines the LSF's by applying a discrete cosine transformation to the coefficients of $G(z)$ and $H(z)$ in Eq(2.28,2.27). The roots, corresponding to the LSF's, are found by searching along the $\omega = [0, \pi]$ range iteratively for sign changes in the polynomials $G(z)$ and $H(z)$.

The third method by Kabal and Ramachandran[9] makes use of Chebyshev polynomials

$$T_m(x) = \cos m\omega, \quad x = \cos\omega. \tag{2.31}$$

The function $x = \cos(\omega)$ maps the upper semicircle in the z-plane to the real interval $[-1, +1]$. The polynomials $G'(\omega)$ and $H'(\omega)$ can be expanded using the Chebyshev polynomials as follows,

$$\begin{aligned} G'(x) &= 2\textstyle\sum_{i=0}^{l} g_i T_{l-i} \\ H'(x) &= 2\textstyle\sum_{i=0}^{m} h_i T_{m-i} \end{aligned} \tag{2.32}$$

The roots of these Chebyshev expansions give the LSF's after the inverse transformation $\omega = \cos^{-1}(x)$. The roots are determined iteratively by searching for sign changes along the interval $[-1, +1]$.

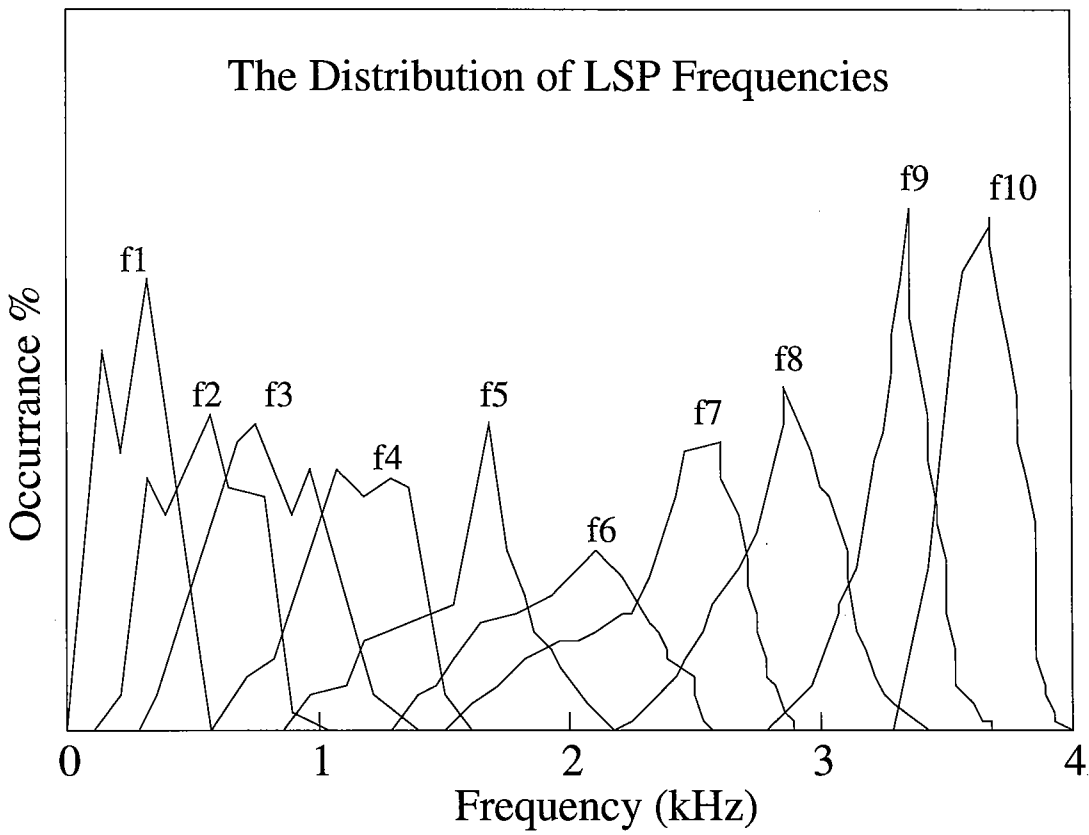


Figure 2.2: The Distribution of line spectral frequencies (from Kang [11])

2.5.1 Properties of line spectral frequencies

The polynomials $P(z)$ and $Q(z)$ have the following properties which make LSF more amenable to efficient encoding than other LPC representation: A(z) is minimum phase if and only if all the roots of $P(z)$ and $Q(z)$ lie on the unit circle, and the roots of $P(z)$ and $Q(z)$ are interlaced with each other. Thus, the stability of the LPC synthesis filter can be ensured by quantizing the LPC information in LSF domain. The LSF parameters satisfy the ordering properties,

$$0 = \omega_0 < \omega_1 < \omega_2 < \ldots < \omega_{M-1} < \omega_M < \omega_{M+1} = \pi \tag{2.33}$$

Therefore, in order to guarantee stability for the LSF synthesis filter, the quantized version of the LSF parameters must satisfy the ordering property. This property makes LSF's more attractive than the original LPC parameters for quantization since the latter needs to be transformed to reflection coefficients for the stability testing. The LSF's correspond to the local minima of the power spectra of the polynomials $G'(\omega)$ and $H'(\omega)$. Also, since the $P(z)$ polynomial is even and the $Q(z)$ polynomial is odd, it is possible to decompose the power spectrum $\|A(\omega)\|^2$:

$$\|A(\omega)\|^2 = (\|P(\omega)\|^2 + \|Q(\omega)\|^2)/4 \tag{2.34}$$

Therefore, the power transfer function associated with $H(z)$ can be calculated as follows:

$$\|H(\omega)\|^2 = \frac{1}{\|A(\omega)\|^2} \tag{2.35}$$

$$= \frac{4}{\|P(\omega)\|^2 + \|Q(\omega)\|^2} \tag{2.36}$$

Eq(2.36) implies that LSF parameters can be interpreted as a representation of an all-pole filter by means of the location density of discrete frequencies, namely $[\omega_1, \omega_2, \ldots, \omega_M]$, in the frequency domain. If the spectrum has resonant frequencies, the LSF's become closely spaced near these frequencies (Fig(2.4)) because the phase angle of the ratio filter changes rapidly in the region. Moreover, the spectral sensitivities of LSF's are localized

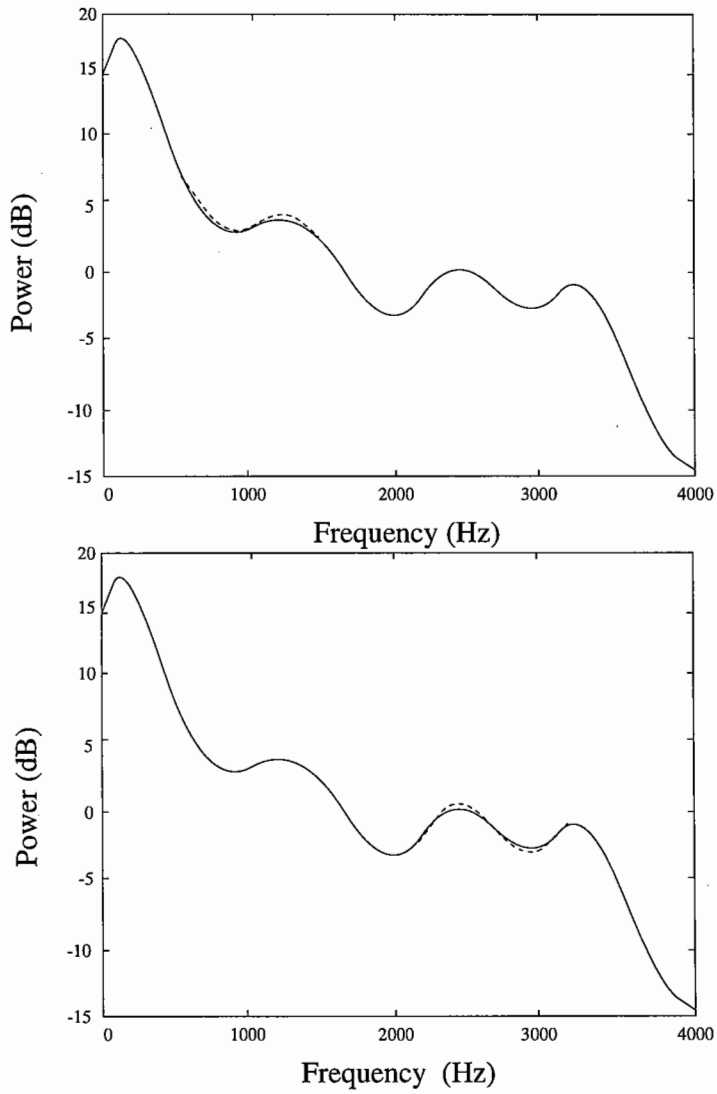


Figure 2.3: Effect of changing LSF on LPC power spectrum. The original spectrum is shown by solid line and the changed spectrum by dotted line. The first figure makes a change of the fourth LSF from 1285 to 1310 Hz. The second figure has a change of the eighth LSF from 2725 to 2690 Hz.

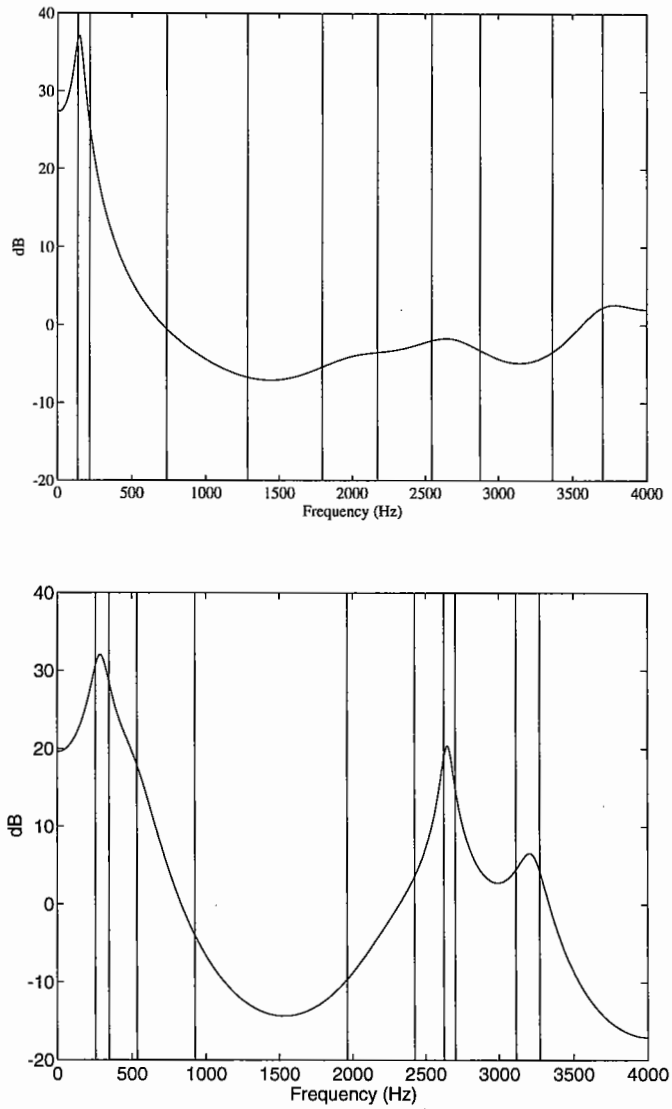


Figure 2.4: LPC power spectrum and associated LSF's

as shown in Fig(2.3). In other words, any change in a given LSF produces a change in the LPC power spectrum only in a neighbourhood. For the other representations, such as the reflection coefficients, the log area ratios, and the predictor coefficients, the spectral variation due to deviation of a single parameter usually is not localized. This is another indication that the LSF's are superior to the other representations.

2.6 Distortion Measures

The techniques used to estimate the LPC parameters in the previous section effectively fit the power spectrum of the associated synthesis filter to that of the speech signal. In a similar manner, vector quantization of LPC parameters can be viewed as selecting from a quantization codebook the LPC vector that yields the best matching spectral envelope to the given spectrum of a short frame of speech. The matching criterion can be based on minimizing the energy of the speech error incurred after quantizing the LPC parameters. However, even with moderate size codebooks, the computational load is very large. Therefore quantitative distortion measures that directly match the candidate LPC codevectors to LPC source parameter vector are needed. The Euclidean distance between a source vector and a candidate codevector has been widely used in early vector quantizers. The limitations of such a measure reveal themselves in unsatisfactory reconstructed speech quality. Taking perceptual considerations into account, the Euclidean measure can nevertheless be appropriately modified to achieve high-quality quantization, namely by appropriately weighting the individual components of the LPC parameter vector. Since the modified distortion measures quantitatively compare the candidate codevector and the LPC parameters in the frequency domain, they are termed *"spectral distortion measures"*. Depending on the selected parameter domain for quantization, an appropriate distortion measure is used as a selection criterion for a codevector. A spectral distortion measure and the Euclidean LSF distance will be briefly introduced in this chapter.

2.7 Spectral Envelope Distortion Measures

The basis for defining and comparing the spectral envelope distortion measures is the comparison of the original speech LPC spectrum obtained from the synthesis filter $1/A(z)$ and the power spectrum of the synthesis filter associated with the quantized LPC parameters, $1/A'(z)$. Both spectrum are taken on a frame-by-frame basis.

2.7.1 Spectral distortion measure

The spectral distortion measure has been used extensively in the past to measure LPC quantization performance. Earlier studies have used an average spectral distortion of 1 dB and a number of outlier criteria to characterize the spectral coding transparency. Spectral distortion for the i^{th} frame, D_i, is defined in(dB) as follows

$$D_i^2 = \frac{2}{F_s/2} \int_0^{F_s/2} \left[10\log_{10}(P_i(f)) - 10\log_{10}(\hat{P}_i(f))\right]^2 df \tag{2.37}$$

where F_s is the sampling frequency in Hz and $P_i(f)$ and $\hat{P}_i(f)$ are the LPC power spectra of the i^{th} frame given by

$$P_i(f) = \frac{1}{\| \left[A_i(e^{j2\pi f/F_s})\right] \|^2}, \tag{2.38}$$

and

$$\hat{P}_i(f) = \frac{1}{\| \left[\hat{A}_i(e^{j2\pi f/F_s})\right] \|^2}, \tag{2.39}$$

where $A_i(z)$ and $\hat{A}_i(z)$ are the unquantized and quantized version of the LPC polynomials respectively for the i^{th} frame. Spectral distortion is computed for all the frames in a data set X. One major disadvantage of using the Spectral Distortion Measure for codebook design is its computation complexity. As a result, another distortion measure is used to approximate this spectral distortion measure to avoid the high computation complexity.

2.7.2 Weighted Euclidean distance

Exploiting the relationships between LSF's and the spectral envelope, a weighted Euclidean (squared error) distance measure may be used.

$$D = \sum_{i=1}^{p} [c_i w_i (x[i] - \hat{x}[i])]^2 \tag{2.40}$$

As shown in Fig(2.2), the LSF's spread out in frequency between 0 and 4 kHz. Since the sensitivity of the human ear to speech sounds decreases as the frequency increases, more weight should be given to the low frequencies. Thus, the following weight values are used by Paliwal & Atal [15]:

$$c_i = 1 \quad \text{for} \quad i = 1, 2, \ldots, 8 \tag{2.41}$$

$$= .8 \quad \text{for} \quad i = 9 \tag{2.42}$$

$$= .4 \quad \text{for} \quad i = 10 \tag{2.43}$$

The peaks of the spectral envelope correspond to formant frequencies which are considerably more important for human speech intelligibility than the valleys of the spectral envelope. Advantage can be taken of this physical characteristic of the human ear by weighing the LSF's near the formant frequencies more than the LSF's in the spectral valleys. As a result, the weights, w_i, are varied from frame-to-frame depending on the LPC power spectrum. The weighing w_i is determined as

$$w_i = [P(f_i)]^r \tag{2.44}$$

$$= \left[\frac{1}{\| 1 - \sum_{j=1}^{10} a_j e^{j2\pi f_j} \|^2} \right]^r \tag{2.45}$$

The value of the exponent r is in the range of $0 < r < 1$. A value of 0.25 is chosen.

2.8 Comparison of Distortion Measures

The most important question that arises from examining the many distortion measures in the previous section is which measure is the most useful. The two main purposes of

the distortion measure in this work are for evaluating the performance of speech coders and selecting vectors from a codebook in vector quantization. In vector quantization, the codebook consists of a set of LSF vectors. These LSF vectors represent the spectral envelopes. The distortion measure is used to select a codevector whose spectral envelope best perceptually matches the spectral envelope to be coded. Thus, it is crucial for the distortion measure to correspond to the perceptual error between two spectral envelopes as heard by the human ear. An important point when evaluating speech coders is that not only is the average distortion critical but so is the error distribution. A large error in one frame of data can ruin the sound quality of an entire sentence. Hence the amount of large errors that occur must be monitored. Thus, just average distortion alone is not adequate for evaluating the performance of speech coders. As mentioned earlier, 1dB is often used as the boundary for transparent coding quality when using the average spectral distortion. In addition, the number of frames of speech that have distortions between 2dB and 4dB must be strictly less than 2% and the number of frames of speech that have distortion greater than 4dB must be equal to 0% [16]. Unfortunately, the spectral distortion measure is hard to use directly during the design phase because of the huge computation complexity entailed . Therefore, an approximation to the spectral distortion measure is selected instead. The most common measure for codebook design is the Weighted Euclidean distance measure with a weighting values motivated by speech properties.

Chapter 3

Vector Quantization

3.1 Introduction

Vector Quantization (VQ) is a powerful source coding technique, commonly used in audio and visual signal compression. Quantization implies a many-to-one mapping, and VQ operates on k-dimensional input vectors. A quantizer may be viewed as a cascade of an encoder and a decoder. The encoder identifies in which region of the source vector space the input vector lies according to some criteria and assigns a binary index. The decoder utilizes this index and generates the output reproduction vector drawn from a look-up table or codebook.

Utilizing vector quantization instead of scalar quantization can result in dramatic performance improvements for a given level of distortion fidelity. This benefit of VQ relative to scalar quantization was first explained in a classic paper on Information Theory by Shannon. Lookabaugh et al. [32] describe three categories of VQ advantage, namely the space filling advantage which depends on the geometry of the partition cells, the shape advantage which corresponds to the "shape" of the marginal density, and the memory advantage which relies on intersample correlation. However, only the first

advantage is source independent. Thus, it is desirable to quantize high dimensional vectors at a given bit rate in order to fully exploit the statistical advantage for a given source. Despite this gain in performance, the computational and storage complexity of VQ, which grow exponentially with the product of the vector dimension and the bit rate, presents a major barrier to exploiting the full power of this compression technique.

Several methods have been proposed to circumvent this complexity obstacle. One approach, using a fast search algorithm [3], [4] ,[5], involves an intensive one- time pre-computation which results in an efficient non-uniform tree-structure. This non-uniform structure enables the search process to eliminate a large number of codewords in each search step. However, this approach often trades an increase in storage complexity for a decrease in arithmetic search complexity. Another approach is to put restrictions on the structure of the codebook. Structured quantizers can surmount both the exponential memory complexity and search complexity, and yet they never achieve the same low distortion as full search of the best unstructured VQ codebook. On the other hand, for a given complexity, a well-designed structured VQ can outperform the unstructured counterpart. The crucial justification for a structured VQ depends on its performance-complexity tradeoffs. There are two basic types of complexity, codebook storage complexity and encoding complexity. By varying these two complexity-performance tradeoffs, different structured VQ can be achieved. The purpose of this chapter is to review a family of structured VQ, called summation product code VQ [6] with a synthesis function $\hat{x} = y_1 + \ldots + y_P$ where $y_i, i = 1, \ldots, P$ are residual codevectors and P is the number of stages. That is, the reproduced output vector can be synthesized by summing P residual codevectors. Various well known structured VQ family such as multi-stage VQ, tree-structured VQ, and split VQ [7] all belong to this group.

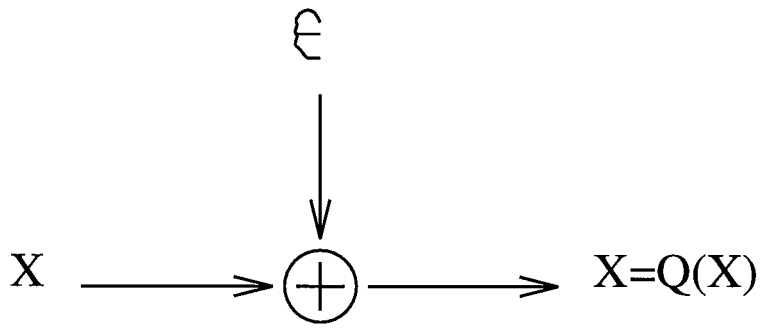


Figure 3.1: Additive noise model of a quantizer

3.2 Scalar Quantization

Before studying quantization in higher dimensions, an understanding of one-dimensional scalar quantizer is essential. An L-level one-dimensional quantizer, Q, consists of $L+1$ decision levels $A = [a_0, a_1, a_2, \ldots, a_L]$ and a corresponding output set $Y = [y_1, y_2, \ldots, y_L]$. The a_j specify a set of partition intervals where the quantizer output is y_i when an input sample x falls within the interval $S_j : [a_{j-1} \leq x < a_j]$ as in Fig(3.2). The collection of partition intervals forms the partition $S = [S_1, S_2, \ldots, S_L]$. The value of y_i is usually chosen to lie within the interval S_i. The end levels a_0 and a_L are generally chosen to be the smallest and largest values the input samples may obtain.

The quantization process can be modelled as in Fig(3.1). A random error or noise component $\sigma = Q(x) - x$, dependent upon the amplitude of the input signal x, is added during quantization to form the output signal. The quantization noise can be categorized into two forms. The first noise form, granular noise, is bounded in magnitude and occurs when the input sample lies within the finite region defined by decision levels $a_1 < x < a_{L-1}$. The amplitude of the noise signal is restricted by the size of the interval the input signal lies within. The second noise form, overload noise, occurs when the signal lies in one of the end regions and is unbounded in amplitude.

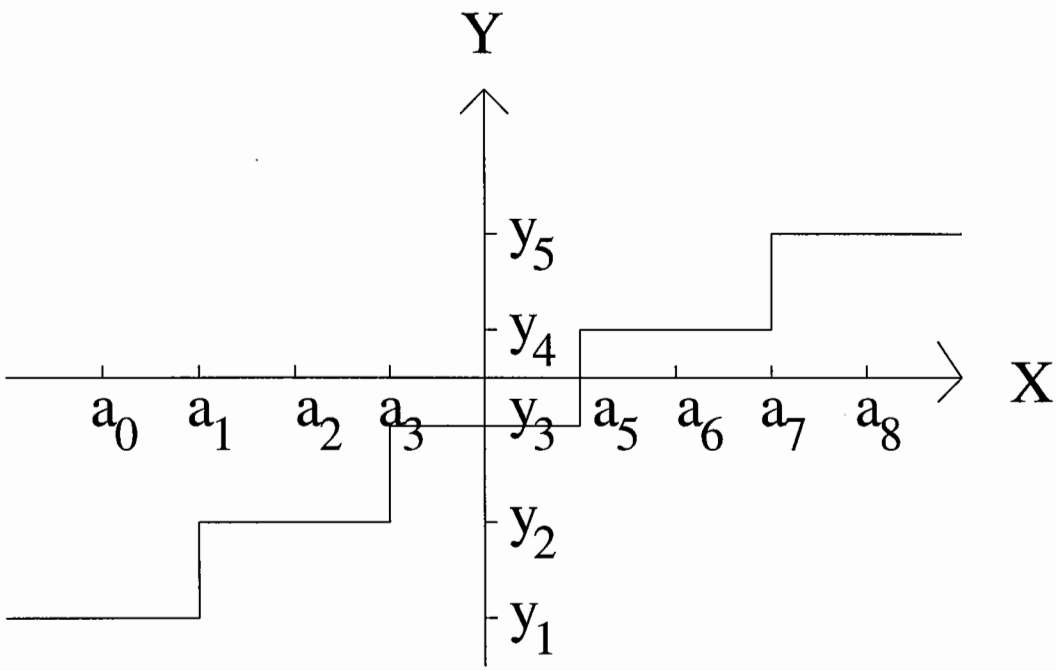


Figure 3.2: A scalar quantizer

One of the most common distortion measures is the mean square error

$$D = \sum_{j=1}^{L} \int_{a_j-1}^{a_j} (x - y_j)^2 f(x) dx \tag{3.1}$$

where $f(x)$ is the probability density function of x.

For a large L, each interval S_i can be made quite small with the exception of the overload regions. It is reasonable to approximate the probability density function $f(x)$ as being constant in S_i so that the probability $p(x) \simeq p(y_i)$ and letting $p(x) \simeq 0$ for the overload regions. Bennett [7] shows that if the number of output levels is large with the output levels lying close to the midpoints of the corresponding quantization intervals, and if successive input samples are only moderately correlated, then the quantization noise is approximately white. Bennett also used a companding model for non-uniform quantization and approximated the scalar distortion by

$$D = \frac{1}{12L^2} \int \frac{p(x)}{E'(x)^2} dx, \tag{3.2}$$

where $E'(x)$ is the slope of the compressor curve. This equation assumed that the overload decision levels a_0 and a_L are chosen so that overload noise is negligible compared to the granular noise.

In general, necessary conditions for the optimality of the quantizer follow from the minimization of Eq(3.1)

$$\frac{\delta D}{\delta y_j} = 0; \text{for} \quad j = 1, 2, \ldots, L-1 \tag{3.3}$$

$$\frac{\delta D}{\delta a_j} = 0; \text{for} \quad j = 0, 1, 2, \ldots, L \tag{3.4}$$

The above equations indicate that the partition boundaries satisfy

$$a_j = \frac{y_j + y_{j+1}}{2} \quad \text{for} \quad 1 \leq j \leq L-1 \tag{3.5}$$

and the output y_j satisfies

$$y_j = \frac{\int_{a_{j-1}}^{a_j} x f(x) dx}{\int_{a_{j-1}}^{a_j} f(x) dx} \quad \text{for} \quad 1 \leq j \leq L. \tag{3.6}$$

Thus, the optimal partition boundary lies in between each quantized output point. The optimal output point, in return, located at the centroid of the interval.

3.3 Vector Quantization

The extension of scalar quantization to higher dimensions leads to vector quantization. For a b bit, $2^b = L$ level quantizer, an input vector $x = (x_0, \ldots, x_{k-1})$, where k is the dimension of the vector, is assigned by the encoder a reproduction index i. Define cell $S_i = \{x : E(x) = i\}$, where $E()$ is the encoder mapping of the quantizer. The set $S = [S_i; i = 1, \ldots, L]$ forms partition with L regions. The index i is used to look up a reproduction codebook $Y = [y_i; i = 1, \ldots, L]$, to generate a reproduction output $y_i = D(i)$, where $D()$ corresponds to the decoder mapping of the quantizer. The optimal partition region S_i is determined by minimizing the distortion measure $d(x, y)$. For any

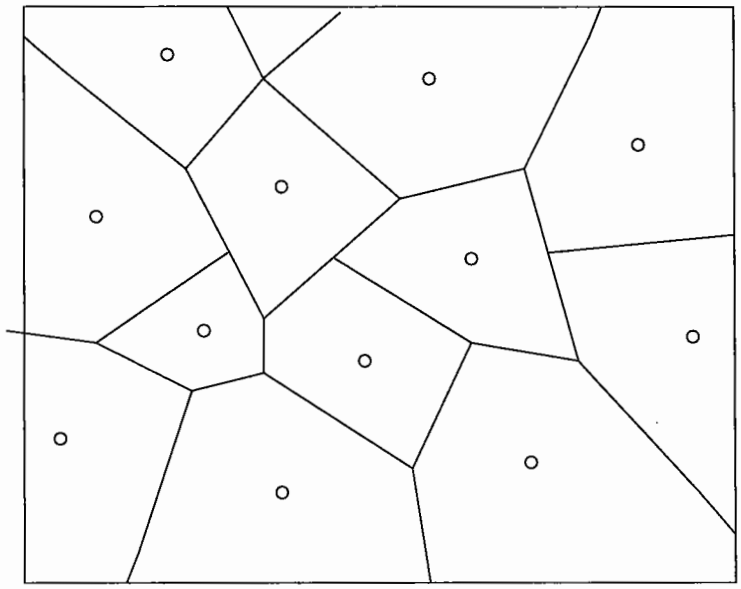


Figure 3.3: A Voronoi partition

vector quantizer, with a fixed decoder, the encoder partition that minimizes the squared error distortion satisfies

$$\|x - y_j\|^2 \leq \|x - y_i\|^2, j \neq i \quad \text{for all} \quad j, i \in L \tag{3.7}$$

This equation generate a Voronoi partition of R^k. An input vector may fall onto the boundary of the regions, that is, there is more than one codevector at the same minimum distance from the input vector x. In this case, a tie breaking rule is used to assigned the input vector to an arbitrary codevector.

For a given partition, the codebook that minimizes the mean square error distortion satisfies

$$y_i = E\left[x | x \in S_i\right] \tag{3.8}$$

Since Eq(3.7) generates a Voronoi partition, illustrated in Fig(3.3), whose regions are strictly convex, and y_i is always inside S_i.

Asymptotic quantization theory, an extension of the classic Bennett integral Eq(3.2) to vector quantizers, shows quantitatively how the mean-squared error of a

many point VQ depends on the distribution of the output points and the shapes of the partition cells. At high bit rates, the best quantizer will have cells that approximate the optimal polytope Q that can tessellate R^k and has the least moment of inertia among all other possible tessellating polytopes. For example, triangles, quadrilaterals, and hexagons are all admissible tessellations for $k = 2$, but hexagons are the optimal polytopes in two dimension. In addition, at high resolution, the codewords are at the centers of the hexagons.

3.3.1 Quantization distortion

Since the cells S_i in VQ can be approximated by suitably rotated , translated and scaled optimal polytope Q, the expected mean squared distortion D is

$$D = L^{-r/k} C(k) \sum_{i=1}^{L} p(y_i) V(S_i) \left[\lambda(y_i)\right]^{-2/k} \tag{3.9}$$

where $\lambda(y_i)$ is the point density of the VQ which is inversely proportional to the volume of the cells in the vicinity of y_i. $C(k)$ is the coefficient of quantization for squared error distortion, and $p(x)$ is the probability density function of the source vector x. Gersho conjectured that the coefficient of quantization is determined by the moment of inertia of the cell shape as

$$C(k) = \frac{1}{k} \inf_{Q \in R^k} \int_p \frac{\|x - \hat{y}\|^2}{\left[V(Q)\right]^{1+2/k}} dx \tag{3.10}$$

where $\hat{y}$ is the centroid and $V(Q)$ is the k-dimensional volume of Q. For a uniformly distributed random variable, $C(k)$ may be thought of as the mean distortion of the normalized polytope for a squared error distortion measure. Assuming the point density $\lambda(x)$ is

$$\lambda(x) = cp(x)^{k/k+2} \tag{3.11}$$

where c is chosen to make $\lambda(x)$ integrate to one. Then the distortion becomes

$$D(L;k) = C(k) L^{-2/k} \|p(x)\|_{k/k+2} \tag{3.12}$$

where $\|p(x)\|_{k/k+2}$ is defined as

$$\|p(x)\|_{k/k+2} = \left[\int p(x)^{k/k+2} dx\right]^{1+2/k}. \tag{3.13}$$

This is the well-know Zador-Gersho [24] formula which gives the least distortion of any k-dimensional quantizer for high bit rates. Eq(3.9) is essentially an extension of Bennett's one-dimensional formula to k dimensions.

3.3.2 Vector quantization advantages

By comparing the distortion of the vector quantizer with that of the scalar quantizer using a squared error criterion, Lookabaugh and Gray [32] expressed vector quantization gain as the distortion ratio

$$\Delta(k) = \frac{D(L;1)}{D(L;k)} \tag{3.14}$$

By substituting Eq(3.12) into Eq(3.14)

$$\Delta(k) = \frac{C(1)}{C(k)} \frac{\|\hat{p}(x)\|_{1/3}}{\|p^*(x)\|_{k/(k+2)}} \frac{\|p^*(x)\|_{k/(k+2)}}{\|p(x)\|_{k/(k+2)}} \tag{3.15}$$

where $\hat{p}(x)$ is the marginal density. Define

$$p^*(x) = \prod_{i=0}^{k-1} \hat{p}(x_i) \tag{3.16}$$

as the distribution that would result if the vector coordinates x_i were in fact independent. The gain were then decomposed into three components.

The first component: space filling advantage defined by

$$F(k) = \frac{C(1)}{C(k)} \tag{3.17}$$

depends only on the coefficient of quantization which, in turn, relies on the moment of inertia of the polytope Q.

A classic isoperimetric result states that every convex polytope has a greater moment of inertia with respect to its centroid than a k-dimensional sphere with the

same volume. This leads to a lower bound on $C(k)$ and an upper bound on $F(k)$. Under this spherical upper bound assumption, as the dimension k approaches ∞, the ratio $F(k)$ approaches a value of $(2\pi e)^{-1}$. This value coincides with the other bounds such as Conway and Sloane's conjectured upper bound, lattice lower bound and the Zador's lower bound. Thus, the maximum contribution by the space filling advantage is 1.43dB.

The second advantage of vector quantizer over scalar quantizer is called "shape advantage", defined as

$$S(k) = \frac{\|\hat{p}(x)\|_{1/3}}{\|p^*(x)\|_{(k/k+2)}} \tag{3.18}$$

This advantage only considers the shape of the marginal probability density function. The vector quantizer has different performance for a different probability density function. For example, the maximum shape gain for square error distortion and Gaussian source with infinite dimension is 2.81dB, while the gain for Laplacian source is 5.63dB.

The last component of VQ advantage is called "memory advantage". It captures the non-linear characteristic of the source distribution. The ratio is defined as

$$M(k) = \frac{\|p^*(x)\|_{k/(k+2)}}{\|p(x)\|_{k/(k+2)}} \tag{3.19}$$

The value $M(k)$ depends on the correlation factor. If the vector components are totally independent and identically distributed, the ratio $M(k)$ is unity. The more dependent the components of the vector, the larger is the value $M(k)$.

In unstructured VQ, each source vector x of dimension k is to be encoded with a single codebook. This codebook consists of $L = 2^b$ codewords, where $b = kr$, and r is the number of bits per vector dimension. This quantizer encodes the source vector in such a way that minimizes the total distortion $d(x, \hat{x})$, by searching through the L codewords. Thus both the encoding complexity and codebook storage complexity is equal to L which grows exponentially with the bit rate. To reduce the encoding and the storage complexity, we introduce a family of structured quantization schemes in the following sections.

3.4 Summation Product Code

A summation product code VQ reduces the storage and encoding complexity associated with unstructured VQ by allocating the b bit to P-stage of feature codebooks. The term "product code" implies a structure of several component vectors, each derived from a separate codebook, with their Cartesian products jointly representing the source vector. The storage cost of a b-bit product code is lower than the storage cost of a b-bit unstructured codebook.

While the storage cost is reduced through the use of a product code, computation cost is reduced only by using certain assumptions and approximations. Normally, the encoding technique used by the product code can be based on either sequential or joint search. Joint-search product code has the same computational cost as unstructured VQ and incurs a distortion that may be only slightly worse than unstructured VQ. Joint-search involves searching over the entire equivalent product codebook (every possible combination of the P feature codebooks) to find a reproduction to minimize the overall distortion measure. As a result, a joint-search induces the smallest distortion error among all product code search techniques at the expense of high computational complexity. Other techniques can drastically reduce this computational complexity but at the cost of increasing distortion.

In sequential quantization, one of the component vectors is quantized first, then the quantized value of that vector is used in the quantization of a second component vector, and so on. The "gain-shape" product code is an example where the best product code vector can be obtained by quantizing the "shape" first, then using the quantized "shape" vector to quantize the gain such that the overall distortion is minimized. In practice, sequential quantization is suboptimal solution to reduce computations.

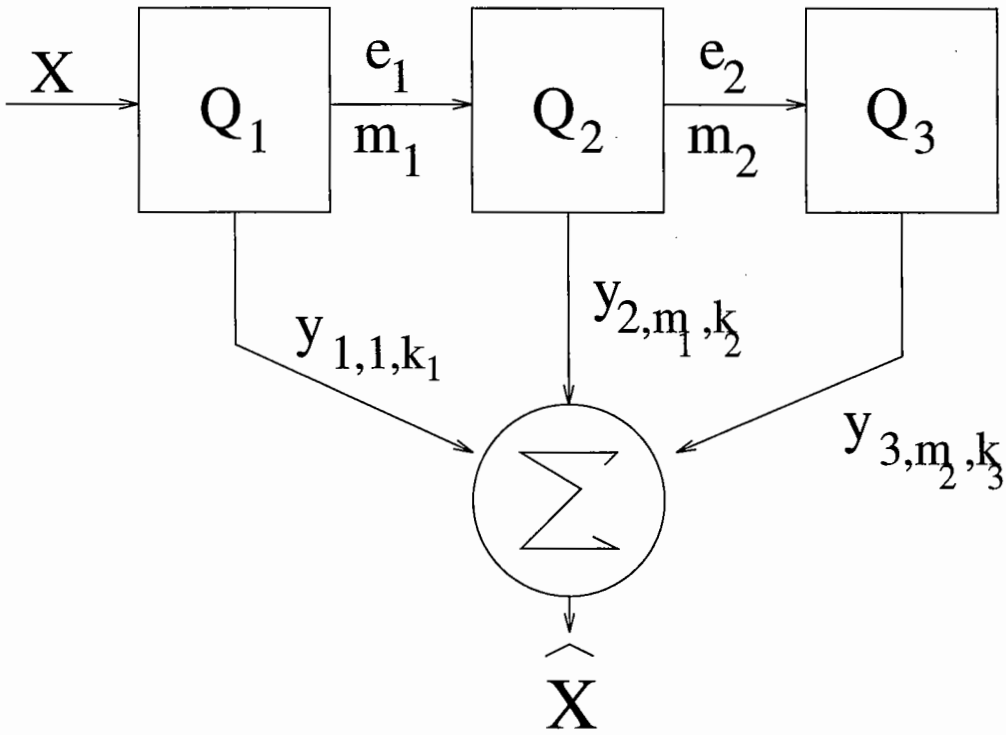


Figure 3.4: Summation product code

3.4.1 Formulation

In a summation product code all feature vectors are summed up to generate the reproduction (Fig(3.4)). Let x be a source vector of dimension k. A P-stage SPC quantizer consists of a finite sequence of P quantizers $Q^i; 1 \leq i \leq P$, which is ordered such that Q^1 quantizes the source vector x and $Q^i, 2 \leq i \leq P$ quantizes the residual vector from stage $i-1$. The SPC structure divides b bits among P stages of codebooks, such that $\sum_{i=1}^{P} b_i = b$, with b_i be the number of bits allocated to stage i. The sizes of the stage feature codebooks are $L_i = 2^{b_i}, i = 1, \ldots P$. In each stage i, there are M_i feature codebooks, C_{i,j_i}, each comprising 2^{b_i} codevectors. Thus, it is necessary to index the codevectors as y_{i,j_i,p_i}, where $1 \leq i \leq P$, indexes the stages, $1 \leq j_i \leq M_i$, is associated with the address of the codebook in stage i, and $1 \leq k_i \leq 2^{b_i}$ corresponds to the index of the codevector . Moreover, associated with each codeword y_{i,j_i,k_i}, is a pointer $\mu_{i,j_i,k_i} = m_i$. The pointer μ_{i,j_i,k_i} determines which codebook is to be used in the $(i+1)-$th stage. The mapping

Q^1 applied to the input x yields an output codeword which minimizes the distortion $d(x, y_{1,1,p_1}) \leq d(x, y_{1,1,q_1}), p_1 \neq q_1$, where $p_1, q_1 = 1, 2, \ldots 2^{b_1}$. Taking the difference of x and $y_{1,1,p_1}$ produces an error vector $e_1 = x - y_{1,1,p_1}$ as an input to the second stage quantizer Q^2. In general, Q^i maps the error vector e_{i-1} to an output residual vector y_{i,μ_{i-1},p_i} and an error vector $e_i = e_{i-1} - y_{i,\mu_{i-1},p_i}$. The reproduction vector is formed by adding all the P-stage residual vectors

$$\hat{x} = \sum_{i=1}^{P} y_{i,\mu_{i-1},p_i} \tag{3.20}$$

3.4.2 Storage complexity-distortion tradeoff

In general, the memory storage complexity of the summation product code structure is of the order of $\sum_{i=1}^{P} M_i 2^{b_i} k$. The degree of fanout M_i, or the number of codebooks in each stage, is no more than the number of codewords stored in the codebooks of the previous stage, that is, $2^{b_{i-1}} M_{i-1}$. The computation complexity for a sequential search is of the order of $\sum_{i=1}^{P} 2^{b_i}$ which is independent of the degree of fanout.

By varying the storage complexity and/or independently the encoding search complexity of the summation product code, different performance-complexity tradeoffs can be achieved. There are generally three ways to vary the storage complexity. The first one is to vary the degree of codebook sharing or the codebook fanout in each stage. Specific examples of fanout variation are tree-structured VQ and multistage VQ, with respectively maximum and unity fanout. A second way is to simply vary the bit allocation among the stages. A third way is to vary the number of vector components in each stage as in split VQ. Split VQ is equivalent to summing codevectors of the same dimension but each having non-zero components in a mutually exclusive interval of component locations.

Tree structured codebooks

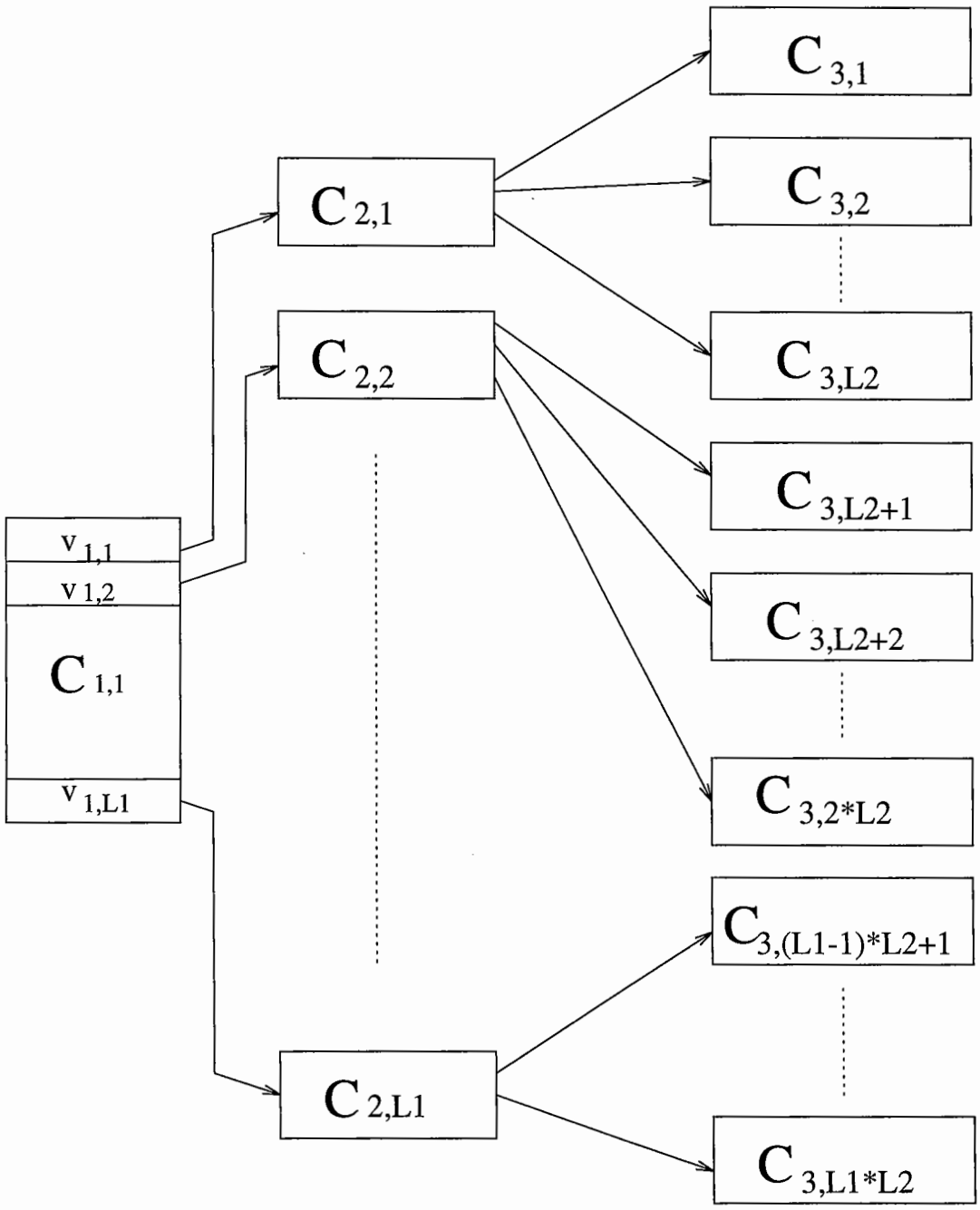


Figure 3.5: Balanced tree structured vector quantizer

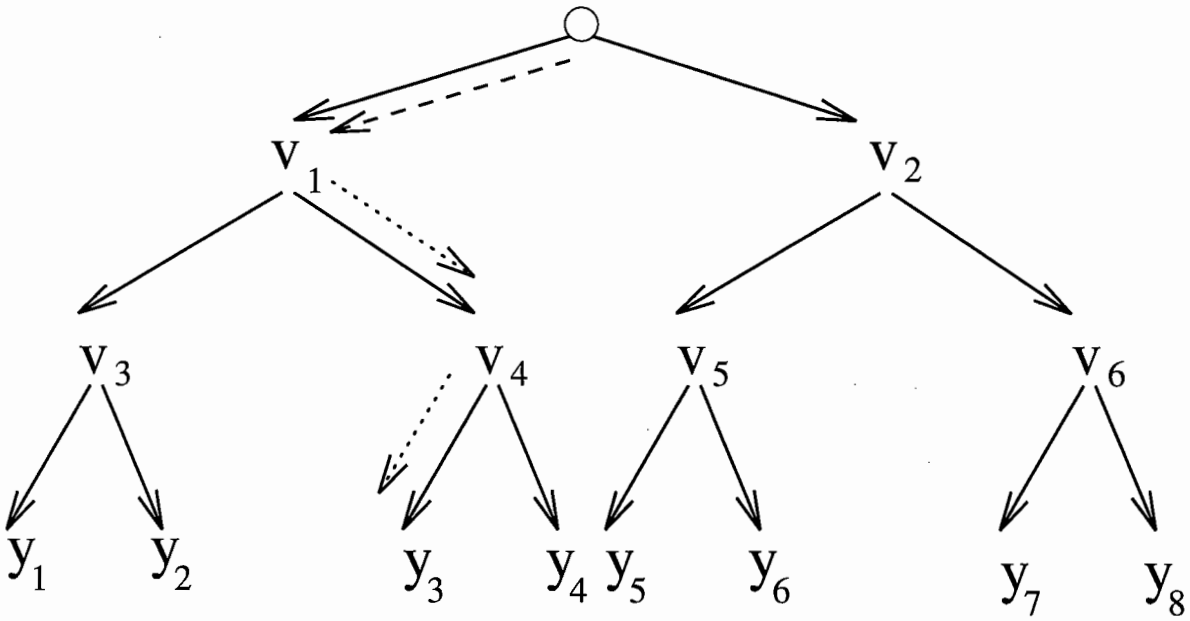


Figure 3.6: Binary tree structured vector quantizer

Tree structured VQ (TSVQ) is formed (Fig(3.5)), when every summation product code-vector branches out to a new succeeding codebook,. In other words, each codevector except the leaf codevectors of TSVQ, has a unique descendant codebook and no codebook is shared. TSVQ can be considered as a fast search approach which reduces the encoding complexity at the expense of using much more memory and incurring lower SNR performance than would be obtained with an optimal unstructured codebook. In each intermediate step in a conventional TSVQ codebook search, a decision is made to determine which branch to be taken for the next step. As a result, a large part of storage the codebook can be regarded as guiding for the search. For an M-ary balanced tree-structure quantizer, the input vector is compared with M pre-designed codevectors at each stage. The nearest neighbour code vector determines which one of the m paths to take in order to reach the next stage. The computational complexity of tree-structured VQ increases only linearly with the number of tree level. On the other hand, the memory required is twice the memory needed for standard unstructured VQ.

Most of the time, binary tree-searched VQ is used for its simplicity. It partitions

the space in such a way that the search complexity proportional to $\log_2 L$ rather than L where L is the number of reproduction vectors. Specifically, for binary TSVQ, the k-dimensional source vector space is first divided into two regions, then each of the two regions is divided further into two subregions, and so on, until the space is divided into L regions or cells. Here, L is restricted to be a power of 2, $L = 2^R$, where $R = kr$ is an integral number of bits. Associated with each region is its centroid. At the first level, there exists only two centroids, V_1 and V_2. At the second level, there are four regions with centroids V_3 through V_6. The number of centroids increases exponentially with the level. And at the last level, the L codevectors $y_i, i = 1, \ldots L$ are stored. An input vector x is quantize by traversing (searching) the tree along a path that gives the minimum distortion at each node as shown in Fig(3.6). Thus the distortion between x, V_j and V_{j+1} is compared. If $d(x, V_j) < d(x, V_{j+1})$, for instance, then the path leading to V_j is taken. Clearly, the total number of distortion computations is equal to $2\log_2 L$. The storage cost, however, has increased due to the additional storage of the intermediate codevectors. Besides the storage obstacle, the traditional design of binary tree faces another obstacle: empty cells. During codebook design, as the tree is grown and the training set partitioned into smaller subsets, there is a high probability that no training vector will fall into some cells. This prevents the maintenance of a balanced tree. As a result, TSVQ is used with limited number of levels or limited number of bits. Normally the number of levels range up to around 12.

Multistage VQ (MSVQ)

As discussed in the previous section, the advantage of binary TSVQ is the substantial decrease in computation cost relative to full search, at the price of relatively small increase in distortion. However, the storage cost is doubled relative to unstructured VQ. A multistage VQ is a unity-fanout summation product code where only a single codebook is stored in each stage. MSVQ reduces storage as well as computation cost.

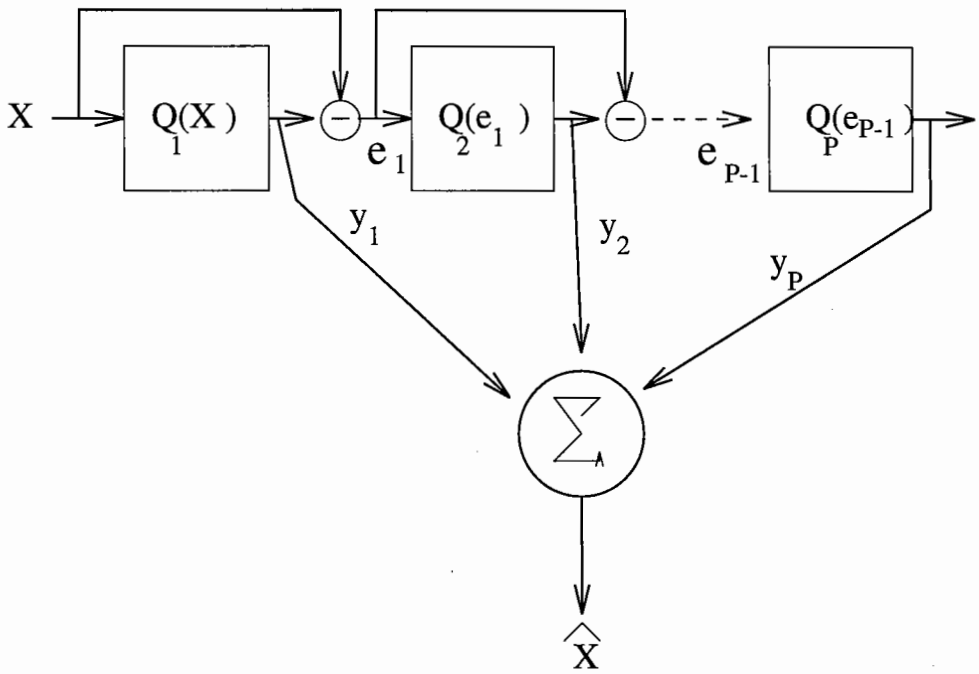


Figure 3.7: Multistage vector quantizer

In tree-search VQ, one is trying to find the desired codevector by searching the space in a systematic manner; at each step, one gets closer and closer to the desired code vector. The role played by the intermediate vectors V as in Fig(3.6) is simply to guide the search; the desired code vector is found at the end of the search. Multi-stage VQ, on the other hand, sums up all the residual codevectors from the stages and forms the output codevector. The difference between MSVQ and TSVQ is that TSVQ stores at each stage intermediate reproductions where as MSVQ stores at each stage differences between the intermediate reproductions from two successive stages.

In multistage VQ, the encoding task is divided over multiple stages, where the first stage performs a relatively crude quantization of the input vector using a small codebook. Then a second stage quantizer operates on the error vector between the original and quantized first stage output. The quantized error vector then provides a second approximation to the original input vector thereby leading to a refined or more

accurate representation of the input. A third stage quantizer is then used to quantized the second stage error vector to provide a further refinement and so on.

In multistage VQ, as shown in the Fig(3.7), The input vector x is first quantized using a b_1 bit vector quantizer. The first stage residual error e_1 is formed by taking the difference between the original vector x and the quantized vectors y_1. The residual error is then used as an input to a b_2 bit VQ stage whose output is y_2. The process can be repeated by feeding the second stage error e_2 into a third stage vector quantizer, and so on. For a P-stage MSVQ, the final quantized value of x is simply the sum of the P quantized residual vectors $y_1 \ldots y_P$

$$Q(x) = \hat{x} = \sum_{i=1}^{P} y_i. \tag{3.21}$$

The performance of multistage VQ tends to deteriorate as more stages are used, due to greater structural constraints on the equivalent product codebook. On the other hand, a major appeal of MSVQ is its provision for decreasing both the storage and encoding complexity by increasing the number of stages. The computational and storage costs for a P-stage MSVQ are simply $\sum_{i=1}^{P} 2^{b_i}$.

Constrained storage VQ

It is well known that MSVQ does not perform as well as TSVQ. In order to see why MSVQ gives larger distortion than TSVQ with the same dimension and rate, note that in the training process for multistage VQ, after the first stage, the errors from different partition cells are pooled together to form the training data for designing the second quantizer. If all the cells in the first stage have the same size, shape, and orientation, and if the probability density function is approximately flat on each cell, then pooling of errors from different cells is justified. However, generally this is not the case.

According to asymptotic quantization theory, it can be argued that when the first stage of an MSVQ is operating in a high-rate region, the probability distributions of the

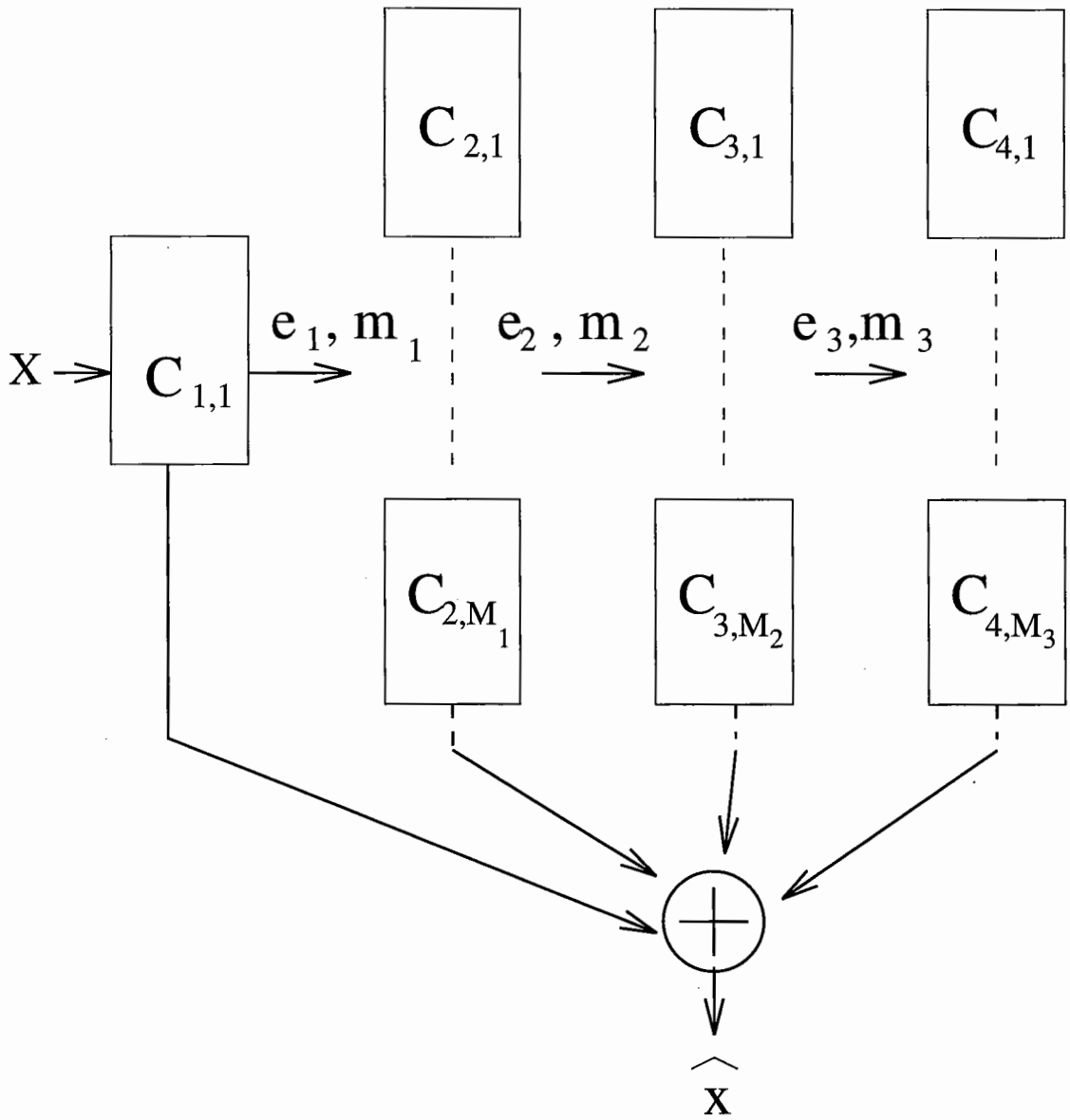


Figure 3.8: Constrained storage vector quantizer

conditional residual sources in the second stage are all similar in shape but different from each other by rotation and scaling of the support regions. As a result, there are some similarity between the residual sources. With this assumption, a constrained storage VQ algorithm was introduced [35] to exploit the similarities between the conditional residual sources, by assigning a small number of codebooks to be shared amongst the sources. By clustering together sources with similar statistics, one could reduce the storage requirement of TSVQ while suffering only a marginal performance degradation. CSVQ provides a bridge between the full fanout of TSVQ and the unity fanout of MSVQ.

However, to directly measure the statistical similarity between sources is difficult, especially, when the source distributions are known only from training data. Another approach is to directly design a set of M codebooks that optimize an overall performance objective without an explicit measure of similarity.

Let x be a k-dimensions source. Suppose that there are M codebooks in each stage, and the first stage has only a single codebook. The first stage quantizer Q_0 produces a set of 2^{b_1} residual sources. Let's represent each of such sources by a k dimensional random vector $Z_i, i = 1 \ldots 2^{b_1}$. Let, the overall distortion objective for CSVQ be

$$D = \sum_{i=1}^{2^{b_1}} p_i E\left[d(Z_i, Q_{\mu(i)}(Z_i))\right] \tag{3.22}$$

where $\mu(i) \in \{1, \ldots, M, M \leq N\}$ for N sources sharing M codebooks $C_1, \ldots, C_M$, and $\sum_{i=1}^{N} p_i = 1, p_i > 0, \forall i$. Two necessary optimality conditions are provided in [6] as the basis for an iterative design algorithm. Firstly, for each m, $\mu(n) = m$ only if

$$Ed(Z_i, Q_m(Z_i)) \leq Ed(Z_i, Q_j(Z_i)) \tag{3.23}$$

Secondly, assuming that the pointer function $\mu(.)$ is given, $\mu(i) = m$, then the codebook C_m should be the optimal codebook for the pdf that is a weighted average of the pdf's of those random vectors that are to be quantized with codebook C_m

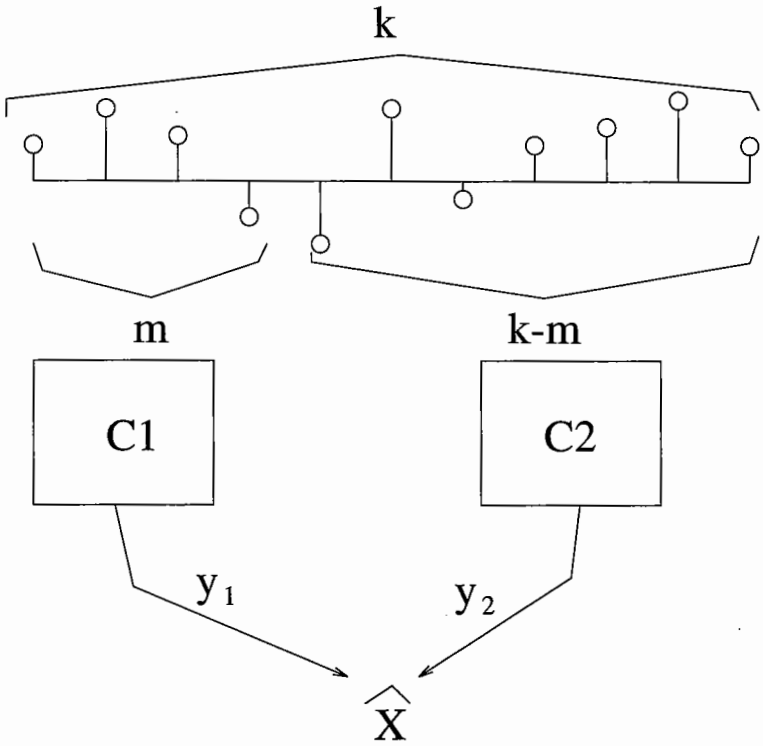


Figure 3.9: Split vector quantizer

Split VQ

A simple way to reduce the search and storage complexity in coding a high dimensional vector is to partition the vector into two or more subvectors. Thus an k-dimensional input vector $x = (x_1, x_2, \ldots, x_k)$ can be partitioned into two vectors x_a and x_b of dimension m and $k - m$ respectively. For the squared error distortion measure,

$$\|x - \hat{x}\|^2 = \|x_a - y_1\|^2 + \|x_b - y_2\|^2 \tag{3.24}$$

the encoder can simply find the nearest code vector to y_1 in codebook C_1 and independently, the nearest codeword y_2 in the codebook C_2. Furthermore, codebook design involves deriving two separate training sets T_a and T_b of m-dimensional and $(k - m)$-dimensional vectors respectively, from the original training set T of k-dimensional vectors, and separately designing a codebook for each training set.

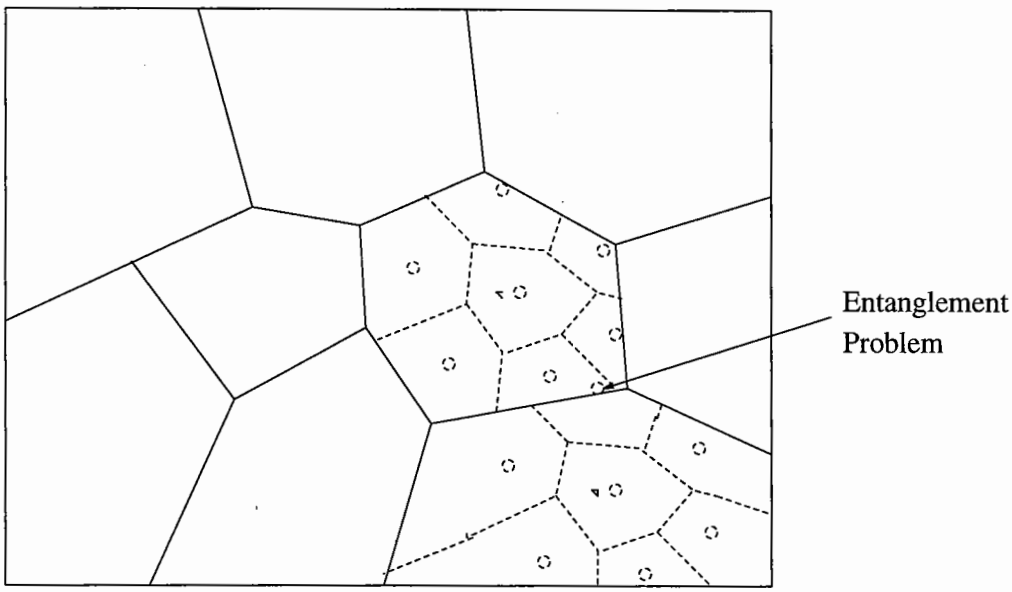


Figure 3.10: Voronoi partition of MSVQ with sequential search

3.4.3 Encoding complexity-distortion tradeoff

Besides the tradeoff between distortion and storage complexity, the encoding complexity can also be traded with distortion by adjusting the number of survivors used in encoding. The single-survivor search in conventional sequential search SPC induces a partition which is highly suboptimal with respect to the equivalent product codebook. In general, the partition is optimal only if an exhaustive search is performed over the product codebook. A sequential search SPC replaces the hyperplane boundaries making up the optimal partition by another set of hyperplanes which are suboptimal with respect to joint search SPC. The suboptimal partition is built hierarchically by partitioning the confined region that is defined by the previous quantizer into a set of subregions as shown in Fig(3.10). As a result, some of the the source vectors will not be assigned to their nearest neighbour reproductions. Moreover, this sequential search of SPC VQ can produce a serious inefficiency, the so-called entanglement problem (Fig(3.10)), i.e., some of the equivalent codeword falling out of the regions boundary. The problem becomes more acute as more stages are introduced. Keeping multiple survivors or using the M-

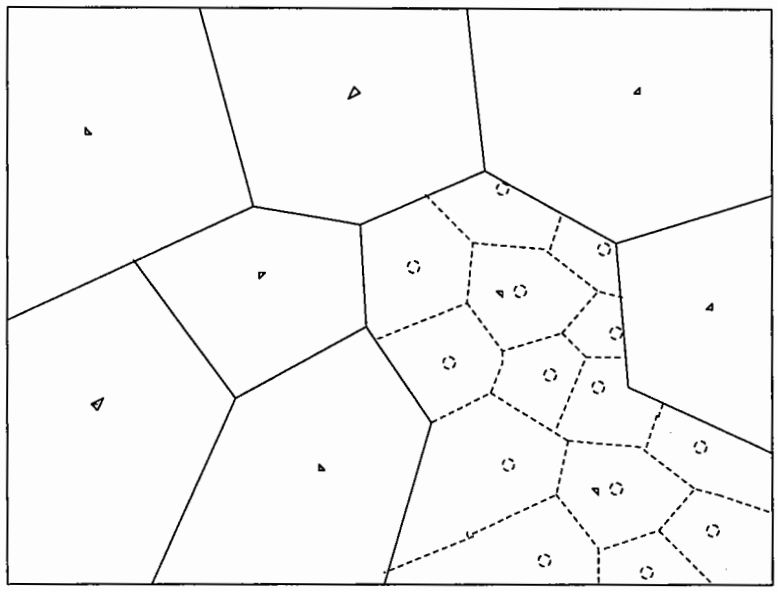


Figure 3.11: Voronoi partition of MSVQ with 2-survivor search

algorithm [2] is equivalent to making a delayed decision on choosing the regions the input vector falls. The survivors define a neighbourhood in which finer hyperplane partitioning is induced when the descendants of the survivors are compared. As more survivors are kept (Fig(3.11)), the partition in the neighbourhood can be a better approximation of the optimal exhaustive-search partition. It is quite conceivable that only a relatively small number of survivors are required, since only a small neighbourhood is needed to approach the quality of the optimal partition; this is so because each partition cell is neighbour to only a small subset of all the cells.

To conduct a multiple survivor search for the P-stage SPC, the input vector is encoded with the first stage quantizer which produces 2^{b_1} distortion values. The indexes of the N_1 codevectors which give the N_1 least distortion values are retained in a set of indexes G_1. If there are two identical distortion values, a tie-breaking rule is used to decide which codevector to use. Each of the N_1 residual vectors will be encoded with the second stage quantizer. A total of $N_1 2^{b_2}$ distortion values are generated. Then the N_2 codevectors associated with the N_2 least distortion values are kept as survivors

to form the index set G_2. The remaining encoding process is the same until it gets to the final stage where there is only one survivor corresponding to the least of the $N_{P-1}2^{b_P}$ distortion values. Summing up the last-stage residual codevector together with the ancestor stage codevectors on its survivor path synthesize the reproduction $\hat{x}$.

In terms of total number of distortion computations, the encoding complexity is $\sum_{i=1}^{s} N_{i-1}2^{b_i}$ where $N_0 = 1$. In most applications, the number of survivors for each stage is the same, that is $N_i = N \forall i$, so that the complexity becomes $N\sum_{i=2}^{s} 2^{b_i} + 2^{b_1}$. When $M = 1$, this reduces to the encoding complexity of a strict sequential search product code. Normally, by keeping a small number of survivors, a substantial drop in distortion is achieved.

Since $N\sum_{i=1}^{s} 2^{b_i} \ll \prod_{i=1}^{s} 2^{b_i} = 2^b$ for small N, SPCs can achieve a low and tractable encoding complexity and a comparatively low distortion level. When the maximum number of survivors is allowed to keep from stage to stage, sequential search attains its best possible performance.

Chapter 4

Joint Codebook Design For Summation Product Codes (SPCs)

The codebook design algorithm often impacts the performance of a quantizer. Traditionally, SPC quantizers are designed using the generalized Lloyd algorithm (GLA) [13]. The stage codebooks are populated sequentially to minimize the average squared error distortion at each stage. This method does not optimize the SPC as a whole. Joint-codebook design, on the other hand, implies designing/optimizing the stage codebooks as a whole to minimize the overall average distortion. In this chapter, a joint design algorithm will be elaborated. Application of the joint design method to MSVQ will also be presented.

4.1 Greedy Design Using GLA

The generalized Lloyd's algorithm was first employed for designing unstructured VQ codebooks. The GLA is based on the nearest neighbour and centroid optimality conditions, specified in Eq(3.7) and Eq(3.8), to iteratively improve the codebook. The equa-

tions Eq(3.7) and Eq(3.8) apply only to a known source distribution. However, most practical sources do not have known distributions. This difficulty may be circumvented by using a long training sequence of data. The training sequence, $[x_i; i = 1, \ldots, N]$, where N is the number of training vectors, may be used to form the time-averaged distortion D, defined as

$$D = \frac{1}{N} \sum_{i=1}^{N} d(x_i, \hat{x}_i). \tag{4.1}$$

where $\hat{x}_i$ is the quantized vector. This approximates the expected distortion $E\left[d(x_i, \hat{x}_i)\right]$ with respect to the training sequence, provided the number of training vectors is large.

The generalized Lloyd algorithm for designing a codebook based on a set of training vectors can then be described as follows.

1) The initialization step involves finding an initial codebook.

2) Given a set of training vectors, assign each of the vectors to the codevector which minimizes the distortion measure. Each of the codevectors is then replaced by the average of the training vectors currently assigned to it. If the change in average distortion between one iteration and the next is small enough, then the codebook is considered to be determined.

The execution time of this algorithm is variable because the required number of iterations cannot be predicted ahead of time. Experience indicates that execution time grows quickly as the training set gets larger, as the number of code words increases, and as the vector dimension increases. In this algorithm, by far the most complicated task is to come up with an acceptable initial codebook. A splitting algorithm is used for codebook initialization. This splitting algorithm utilizes a binary tree initialization which generates a large codebook from a small one. For a training sequence $X = [x_0, x_1, \ldots, x_N]$ where N equals the number of training vector, the splitting algorithm starts by taking the centroid y_0 of the entire sequence and then splitting it into two codewords $y_0 + \sigma, y_0 - \sigma$. One choice of σ is to make it proportional to the eigenvector

corresponding to the largest eigenvalue of the covariance matrix of the training set. Each of these two codewords can split again in exactly the same manner. The process continues until it produces the required codebook size. Since the GLA can only design one codebook at a time, SPC quantizers the GLA has been applied to, treat each of the stage codebooks as a single codebook. The residuals generated in each stage are pooled together to form the training source for the succeeding stage. Thus, each codebook is trained independently and sequentially by using the residuals generated from the previous stage. As a result, the codebook generated in each stage is not optimal with respect to the overall distortion measure. we term the GLA algorithm as a greedy design method since it finds each stage codebook as if there were only one stage.

4.2 Joint Codebook Design Algorithm

The goal of joint codebook design for SPC quantizers is to design all the stage codebooks simultaneously so that they are optimal for a given overall distortion criterion. Suppose a vector source X of dimension k is to be encoded at a rate of r bits per vector component. Each of the 2^{kr} encoder partition regions for unstructured VQ corresponds to one codevector, hence a one-to-one correspondence between the encoder partition regions and the codevectors. On the other hand, every SPC has an equivalent product codebook. From a SPC with P stages of codebooks $C_1, C_2, \ldots C_P$ with codebook sizes $L_1, L_2, \ldots, L_P$ respectively, the equivalent product codebook C_S with codebook size $L_S = \prod_{i=1}^{P} L_i$ can be produced by summing up the codevectors orderly. That is

$$\begin{aligned} C_S &= C_1 \cup C_2 \cup \ldots C_P \\ &= y_{1,1} + y_{2,1} + \ldots + y_{P,1} \\ &\quad\; y_{1,1} + y_{2,1} + \ldots + y_{P,2} \\ &\quad\; \vdots \\ &\quad\; y_{1,L_1} + y_{2,L_2} + \ldots y_{P,L_P}. \end{aligned} \tag{4.2}$$

where $y_{i,j}$ represents the j^{th} codevector in the i^{th} codebook. Corresponding to this, product codebook is a nearest-neighbour partition that minimizes the overall distortion and is the best the SPC encoder can use. Unfortunately, the encoding complexity is equal to that of an exhaustive search unstructured VQ.

For a conventional sequential search encoder, which makes a nearest neighbour encoding decision at each stage, an equivalent partition with 2^{kr} regions is induced. Sequential search only induces a suboptimal partition. However, this suboptimality can be improved by increasing the number of survivors maintained in the search. Given any partition, optimal or not, a "reference product codebook " can be formed by taking the centroid of each of the partition regions. If the SPC stage codebooks can be made to have an equivalent product codebook that approximates this reference codebook, then the stage codebooks can be improved. Based on this idea, a joint codebook design method was developed [34].

4.2.1 A necessary condition for optimal encoding

Assume that the decoder is fixed. The encoder partition that minimizes distortion is the nearest-neighbour partition with 2^{kr} partition regions. The partition regions are denoted $S_i, i = 1, \ldots, 2^{kr}$.

Let X be an k-dimensional random vector source. A P-stage SPC quantizer consists of a sequence of P stage quantizers $Q^i; 1 \leq i \leq P$. Inside the stage s quantizer, there are M_s codebooks. And within each of the M_s codebooks, there are $L_i = 2^{b_i}$ codevectors, where $\sum_{i=1}^{P} b_i = kr$. The address of the codebook to be used in stage s is determined by a codebook pointer $\mu_{m,n}$ (where $m \in \{1, \ldots M_{s-1}\}$ and $n \in \{1, \ldots, L_{s-1}\}$) associated with the previous stage codevector. The overall distortion of a SPC quantizer using the mean squared distortion criterion is

$$D = E\|X - Q^1(e_1) - Q^2(e_2)) - \ldots - Q^P(e_P)\|^2 \tag{4.3}$$

where $e_s = e_{s-1} - Q^{s-1}(e_{s-1})$,$s = 2, \ldots P-1$, is the residual vector generated by the stage $s-1$ quantizer and $e_1 = X$. The necessary condition for this encoder at stage s is

$$d(e_s, y_k) <= d(e_s, y_j) \tag{4.4}$$

where y_k, y_j $k, j = 1, \ldots, L_s$ and $k \neq j$, are the codevectors in one of the M_s codebooks. The address is provided by the codebook pointer $\mu_{m,n}$ associated with the previous stage codevector.

4.2.2 A necessary condition for optimal decoding

Assume the encoder is fixed. Regardless of the degree of fanout, bit allocations and number of survivors maintained during encoding, there will be 2^{kr} partition regions $S_i, i = 1, \ldots, 2^{kr}$ and 2^{kr} equivalent reference codevectors $c_i, i = 1, \ldots, 2^{kr}$. These reference codevectors are simply the centroids of the 2^{kr} regions. Let X be an n-dimensional random vector source. The reference codevectors will be equal to $c_i = E[X|X \in S^i]$. And the overall distortion of Eq(4.3) can be rewritten as

$$D = \sum_{i=1}^{2^{kr}} p_i E\left[\|X - \hat{x} + c_i - c_i\|^2 | S_i\right] \tag{4.5}$$

where p_i is the probability of the region S_i and $\hat{x}$ is the decoder reproduction for that region. Expanding this quadratic to

$$D = \sum_{i=1}^{2^{kr}} p_i E\left[\|X - c_i | X \in S_i\|^2\right] + \sum_{i=1}^{2^{kr}} p_i E\left[\|c_i - \hat{x} | X \in S_i\|^2\right] \tag{4.6}$$

$$+ \sum_{i=1}^{2^{kr}} p_i E\left[(X - c_i | X \in S_i)^T (c_i - \hat{x} | X \in S_i)\right] \tag{4.7}$$

The last term is equal to zero since $c_i = E[X|X \in S_i]$. Therefore the overall distortion is equal to

$$D = D_{min} + D_{ex} \tag{4.8}$$

where $D_{min} = \sum_{i=1}^{2^{kr}} p_i E[\|X - c_i|X \in S_i\|^2]$ and $D_{ex} = \sum_{i=1}^{2^{kr}} p_i E[\|c_i - \hat{x}|X \in S_i\|^2]$. Thus, for a given encoder, the optimal decoder should use equivalent product code vectors that minimized the excess distortion.

4.2.3 Joint codebook design algorithm for MSVQ

As an illustration of how the joint design algorithm operates, a two stage MSVQ with unity fanout is used. A two stage MSVQ, with the size of the first and second stage codebook equals to $L_1 = 2^{b_1}$ and $L_2 = 2^{b_2}$ respectively, have a total of $L_1 + L_2$ codevectors, where $b_1 + b_2 = kr$. In addition, there are 2^{kr} equivalent product codevectors $c_{ij}, i = 1, \ldots, L_1, j = 1, \ldots, L_2$ and 2^{kr} encoding partition regions $S_{ij}, i = 1, \ldots, L_1, j = 1, \ldots, L_2$. Let $Y = \{y_1, y_2, \ldots, y_{L_1}\}$ be the first stage codevectors and $Z = \{z_1, z_2, \ldots, z_{L_2}\}$ be the second stage codevectors. The overall MSE distortion can then be written as

$$\begin{aligned} D &= \sum_{i=1}^{L_1}\sum_{j=1}^{L_2} p_{ij} E\left[\|X - y_i - z_j\|^2 | S_{ij}\right] \\ &= \sum_{i=1}^{L_1}\sum_{j=1}^{L_2} p_{ij} E\left[\|X - c_{ij} + h_{ij}\|^2 | S_{ij}\right] \\ &= \underbrace{\sum_{i=1}^{L_1}\sum_{j=1}^{L_2} p_{ij} E\left[\|X - c_{ij}\|^2 | S_{ij}\right]}_{D_{\min}} + \underbrace{\sum_{i=1}^{L_1}\sum_{j=1}^{L_2} p_{ij} E\left[\|h_{ij}\|^2 | S_{ij}\right]}_{D_{ex}} \\ &\quad \underbrace{+ 2\sum_{i=1}^{L_1}\sum_{j=1}^{L_2} p_{ij} E\left[(X - c_{ij})^T h_{ij} | S_{ij}\right]}_{=0} \end{aligned} \tag{4.9}$$

where p_{ij} is the probability of cell S_{ij}, $h_{ij} = c_{ij} - y_i - z_j$ is the difference between the reference codevector and the reproduction synthesized from the stage vectors y_i and z_j. Since MSE is a per dimension distortion measure, the excess distortion can be summed over all the dimension of the error vector ex_{ij}.

$$D_{ex} = \sum_{l=1}^{k}\sum_{i=1}^{L_1}\sum_{j=1}^{L_2} p_{ij} E\left[\|ex_{ij}\|^2 | S_{ij}\right] \tag{4.10}$$

Obviously each dimension can be optimized independently. The scalar representation of the excess distortion of Eq(4.10) for one dimension becomes

$$D_{ex} = \sum_{i=1}^{L_1}\sum_{j=1}^{L_2} p_{ij}(c_{ij} - y_i - z_j)^2 \tag{4.11}$$

We aim to minimize the excess distortion between the reference and the equivalent product codebook. By letting the vector v be the L_1+L_2 codevectors $v = [y_1, \ldots, y_{L_1}, z_1, \ldots, z_{L_2}]^T$,

and vector b be the 2^{kr} reference codevector $b = [c_{11}, c_{12}, \ldots, c_{1,L_2}, c_{21}, \ldots, c_{L_1 L_2}]^T$, Q be an $(L_1 + L_2)$x2^{kr} structured matrix with the square root probability $q_{ij} = (p_{ij})^{1/2}$ in the diagonal, and A be an $(L_1 + L_2)$x2^{kr} structure matrix

$$A = \begin{bmatrix} u & 0 & \ldots & 0 & I \\ 0 & u & \ldots & 0 & I \\ 0 & 0 & u & \ldots & I \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & u & I \end{bmatrix}. \tag{4.12}$$

Eq(4.11) can be rewritten as

$$D_{ex}(l) = \|Q(Av - b)\|^2 \tag{4.13}$$

The u and 0 are L_1-dimensional column vectors of 1's and 0's respectively, and I is an L_1 x L_1 identity matrix. Therefore, by differentiating D_{ex} and set it equal to zero, a solution to the minimization is formed.

$$(QA)^T QAv = (QA)^T Qb \tag{4.14}$$

This can be reduced to

$$Bv = d \tag{4.15}$$

where

$$d = \begin{bmatrix} \sum_{j=1}^{L_2} p_{1j} c_{1j} & \cdots & \sum_{j=1}^{L_2} p_{L_1 j} c_{L_1 j} & \sum_{i=1}^{L_1} p_{i1} c_{i1} & \cdots & \sum_{i=1}^{L_1} p_{i L_2} c_{i L_2} \end{bmatrix}^T \tag{4.16}$$

$$B = \begin{bmatrix} \sum_{j=1}^{L_2} p_{1j} & 0 & \cdots & p_{11} & \cdots & p_{1L_2} \\ 0 & \ddots & 0 & \vdots & & \vdots \\ 0 & \cdots & \sum_{j=1}^{L_2} p_{L_1 j} & p_{L_1 1} & \cdots & p_{L_1 L_2} \\ p_{11} & \cdots & p_{L_1 1} & \sum_{i=1}^{L_1} p_{i1} & 0 & \cdots \\ \vdots & & \vdots & 0 & \ddots & 0 \\ p_{1L_2} & \cdots & p_{L_1} & 0 & \cdots & \sum_{i=1}^{L_1} p_{iL_2} \end{bmatrix} \tag{4.17}$$

By solving this matrix equation, the first and second stage codevectors $\begin{bmatrix} y_1 & y_2 & \cdots & y_{L_1} \end{bmatrix}$ and $\begin{bmatrix} z_1 & z_2 & \cdots & z_{L_2} \end{bmatrix}$ can be determined.

4.2.4 The joint design framework

The algorithm for SPCs follows:

1) The initial codebooks are obtained using the conventional GLA. In addition, the pointer function $\mu_{m,n}$ is obtained using the CSVQ algorithm and the counter a is set to 1.

2)Encoder:

Find for each vector $x_t, t = 1, \cdots, N$ in the training set the corresponding partition S_i using the nearest neighbour condition. This encoding process may use a sequential search, possibly with multiple survivors, to approximate the exact nearest neighbour condition.

3)Let the distortion measure $d = \frac{1}{N}\sum_{t=1}^{N} \|x_t - \hat{x}_t\|^2$, where $\hat{x}_t$ is the reproduced vector. If the quotient $\|d^a - d^{a+1}\|/d^a$ is less than a predetermined small positive threshold δ or if a has reached a predetermined upper limit, the algorithm terminates.

4)Decoder:

By minimizing the excess distortion defined in Eq(4.8), and solving the matrix equation as in Eq(4.15). The new codevectors for all the stages can be obtained.

Increment a by 1 and Goto step 2.

Chapter 5

Simulation and Results

We design quantizers to achieve transparent coding of speech LPC parameters. Transparent quantization means no audible distortion. The reconstructed speech using the unquantized LPC parameters should be perceptually indistinguishable from the version obtained by using the quantized parameters. An average of spectral distortion 1dB has been traditionally considered to be the threshold for transparent quantization. However, outliers corresponding to speech frames with spectral distortion greater than 1dB, can severely degrade the perceptual quality of the coded speech, even if average spectral distortion is below 1dB. An additional requirement along with the average spectral distortion will therefore be the minimization of the number of outliers. The formal characterization of "transparent quantization" is [16]:

(a) average spectral distortion of about 1dB,

(b) no outlier frames with spectral distortion larger than 4dB,

(c) less than 2% of the number of frames with spectral distortion in the range 2-4dB.

For scalar quantization of LSF parameters [11], it was found that 32 to 40 bits

per frame were needed to achieve transparent quantization. This rate is too high for low bit rate transmission. The alternative for bit rate reduction is vector quantization(VQ). The average spectral distortion of a 10 bit/frame unstructured VQ scheme is about 3.3dB[12], clearly insufficient for high quality speech coding. Using more bits implies a larger codebook. A larger set of training data is also required which will then increase the complexity of the training process, notwithstanding the prohibitively expensive storage and computational requirement in coding the parameters. Transparent quantization has to be obtained using suboptimal structured VQ schemes. The performance of a structured VQ depends on its storage complexity, encoding complexity, the design algorithm, etc.. Since any good design algorithm will only affect the computational complexity during the design phase and will not change the encoding and storage complexity of the VQ structure, any improvement by the design algorithm will be a free gain. Thus, the first part of this chapter is a comparative study of the conventional GLA versus joint design of SPC for first order Gaussian Markov source. This is followed by a section examining a novel SPC quantizer, namely the predictive split VQ designed using both the GLA and joint design codebook algorithm.

5.1 The Performance of Summation Product Codes Using A Joint Design Algorithm

The Joint design Algorithm, discussed in the previous chapter, updates the 2-stage MSVQ codebooks by solving a matrix equation (4.15). The elements of the matrix A are the probability values p_{ij}. With the property that the sum of all the probability values p_{ij} is always 1, the matrix A is rank deficient. Thus, the matrix equation can be solved only if one or more of the unknowns is fixed. Our first approach to solving the equation is an iterative procedure. Let M be the total number of codevectors to be updated and $L = 2^{kr}$ be the size of the reference product codebook. First, set N of the M unknown

N0. of variables kept constant	The indexes kept constant	Joint Design(SNR dB)	Lloyld's Method
2	1,2	14.25	13.38
2	2,3	14.20	13.38
2	3,4	14.25	13.38
2	4,5	14.24	13.38
2	8,9	14.19	13.38
3	1-3	14.22	13.38
3	2-4	14.24	13.38
3	4-6	14.24	13.38
3	7-9	14.26	13.38

Table 5.1: Joint design vs GLA algorithm for 2-stage MSVQ. Number of bits/stage equalt to 3. AR(1) Gauss-Markov with correlation factor=.95

vector v to their initial codebook values. The matrix now becomes full rank and the remaining $M - N$ codevectors can be solved using for example Gaussian Elimination. In the next iteration, the N codevectors can be updated while the remaining $M - N$ codevectors fixed to their previous values.

Table (5.1) summarizes different arbitrary subsets. The result is based on a number of training vectors of 25000, the dimension is 2, and the number of bits/dim is 3. The first column of the table indicates the number of codevectors in the first stage fixed. The second column indicates the index of the variables fixed during the first iteration. Since the simulation is using a MSVQ quantizer, the indexing of the codevector starts with the first stage codebook and then the second stage codebook. The table compiles SNR results for joint design method and the GLA algorithm. It should be noted that the results for the various choices of subsets are comparatively close. As a result, one can conclude that any subset is likely to be equally effective.

By restructuring the matrix A,

$$\begin{bmatrix} F & P \\ P^T & B \end{bmatrix} \begin{bmatrix} M \\ N \end{bmatrix} = \begin{bmatrix} C \\ D \end{bmatrix} \tag{5.1}$$

with F and B are diagonal matrices, and M and N respectively contains the codevectors of the first and second codebook. By dividing the matrix into blocks, two equations are generated as follows

$$FM + PN = C \quad \rightarrow M = F^{-1}[C - PN] \tag{5.2}$$

$$P^T M + BN = D \quad \rightarrow N = B^{-1}\left[D - P^T M\right] \tag{5.3}$$

These two equations can be solved iteratively when one of the codebooks is fixed. This method is related to the joint design method proposed in [35]. The rest of the simulation results are based on solving the above matrix equation.

5.1.1 Variation on storage and encoding complexity

Since the quantizer design algorithm is an iterative procedure, design time is determined partly by the number of iterations. The maximum number of iterations is chosen to be 15 in the subsequent simulations.

We tested the joint design algorithm on a simple MSVQ structure. At each step, different complexity variations were tested. The search complexity, degree of fanout and the number of stages were varied. The test were performed with a first order Gauss-Markov source, with 25000 training vectors and 5000 test vectors.

Fig(5.1) shows the gain of the joint design over the GLA design for various correlation values and number of survivors. The joint design gain over conventional stage-by-stage design increases with source correlation. The design gain for an i.i.d Gaussian source is very small. As the source becomes more correlated, the design gain increases.

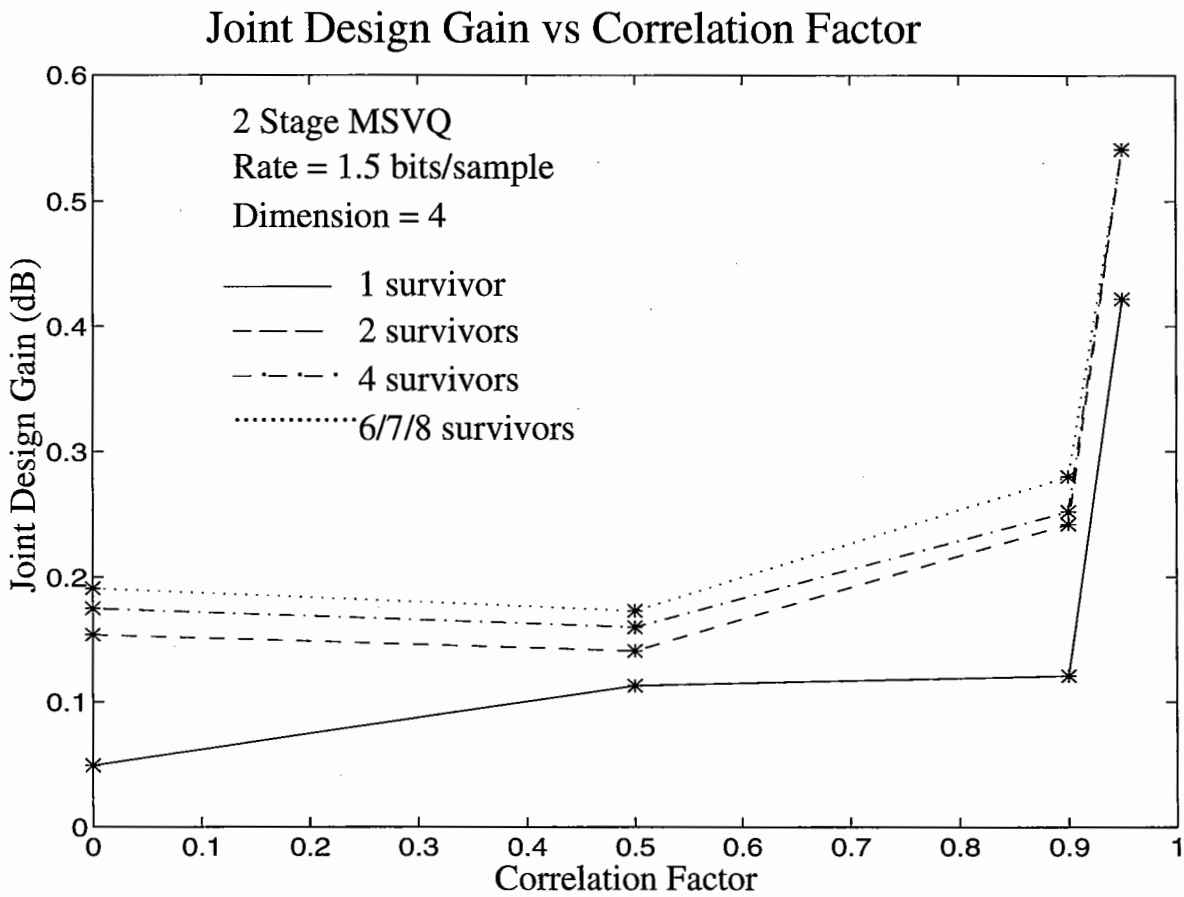


Figure 5.1: Joint design gain vs Gauss-Makov correlation source; no. of stages=2;

No. of sur	Correlation factor	storage Complexity	Joint design SNR	Joint design gain
1	.1	16	6.71dB	0.05dB
2	.1	24	7.02dB	0.09dB
6	.1	56	7.24dB	0.24dB
1	.9	16	12.47dB	0.12dB
2	.9	24	12.70dB	0.24dB
4	.9	40	12.72dB	0.25dB
6	.9	56	12.73dB	0.28dB

Table 5.2: SNR and the design gain of Joint-design algorithm vs number of survivors.

By increasing the number of survivors, the joint design algorithm can yield a better performance even for an i.i.d Gaussian source. However, only a small number of survivors are needed. Additional survivors increase the encoding complexity while more or less the same SNR performance is obtained as seen in Table(5.2). For a fixed design gain level, the number of survivors appears to be inversely related to the correlation factor. That is, the number of survivors required to achieve the best performance decreases as correlation increases.

The next complexity factor varied was the number of stages. With conventional GLA design the distortion tends to increase as the number of stages increases for the same bit rate. However, it may be desirable to increase the number of stages so as to lower the search and storage complexity of the quantizer. Fig(5.2) shows the joint design gain versus the number of stages. It shows that joint design has a small but steady gain over GLA design for an i.i.d Gaussian source as the number of stages increases. On the other hand, the design gain increases rapidly for a high correlation factor. Moreover, in Table(5.3), the joint design method narrows the SNR performance gap between the various configuration with different number of stages. For low correlation values, (Fig(5.2))

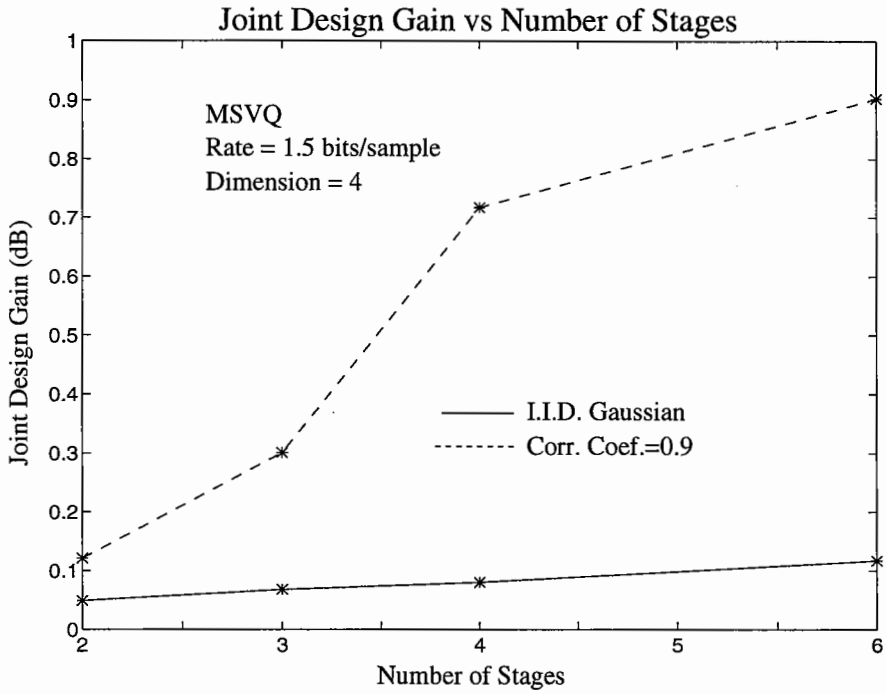


Figure 5.2: Joint design gain vs number of stages GLA algorithm.

the joint design gain remains relatively constant as the number of stages increases. On the other hand, for a high correlation factor, the joint design gain increases more or less linearly as the number of stages increases.

With the number of stages increases for an iid. source, a better design gain can be achieved by keeping multiple survivors (Fig(5.3)).

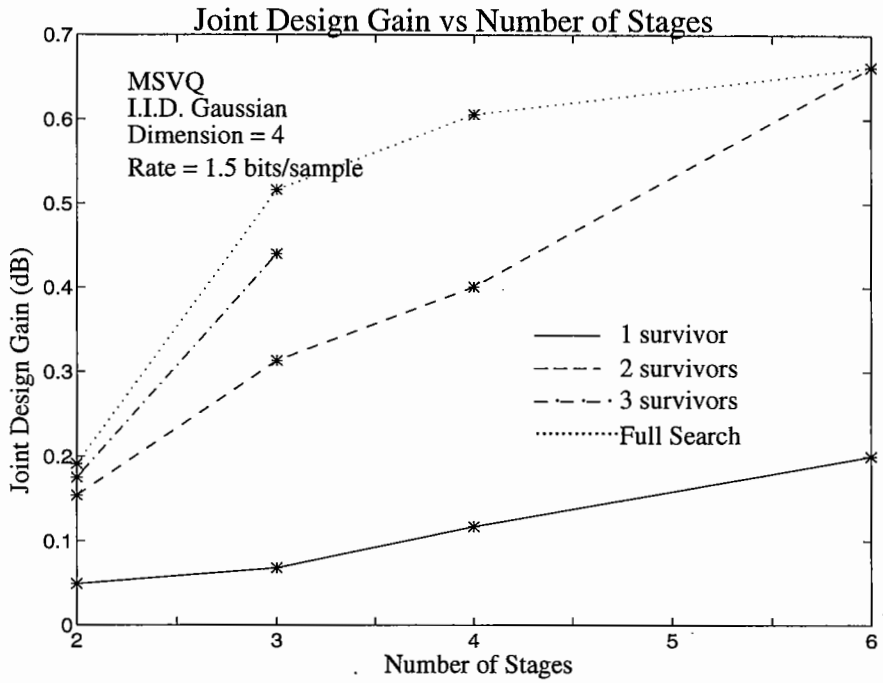


Figure 5.3: Joint design gain vs number of stages GLA algorithm;

N0. of bit/stage	No. of stage	Correlation factor	Joint design Method	GLA Method	Gain
1	6	.1	4.11dB	4.04dB	0.07dB
2	3	.1	4.20dB	4.18dB	0.02dB
3	2	.1	4.37dB	4.36dB	0.00dB
1	6	.5	5.37dB	5.11dB	0.25dB
2	3	.5	5.40dB	5.21dB	0.18dB
3	2	.5	5.45dB	5.38dB	0.07dB
1	6	.9	10.18dB	9.64dB	0.54dB
2	3	.9	10.38dB	10.17dB	0.21dB
3	2	.9	10.62dB	10.54dB	0.08dB

Table 5.3: Signal to noise ratio (dB) vs number of stages. Dimension =6.

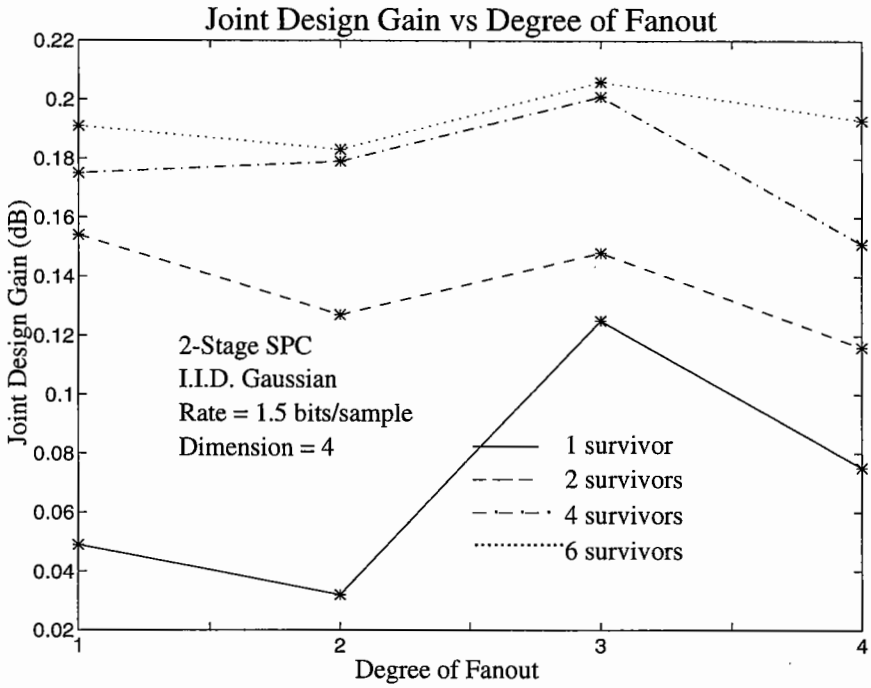


Figure 5.4: Joint design gain vs degree of fanout; i.i.d Gaussian.

Constrained storage VQ

For a 2-stage MSVQ structure with various degrees of fanout, the initial stage codebooks and the pointer function were designed using the GLA algorithm [35]. Fig(5.4) and Fig(5.5) show joint design gain versus degree of fanout. As the degree of fanout increases, the design gain decreases. Fig(5.6) and Fig(5.7) show SNR performance against the number of survivors. For an i.i.d. Gaussian source, with multiple survivor search, the SNR performance of the joint design MSVQ with degree of fanout equal to 2 is slightly better than the joint design MSVQ with unity fanout. Only for high correlated source, the CSVQ has a higher SNR performance than MSVQ. Moreover, the joint design MSVQ always outperform the modified GLA-design CSVQ (with fanout equal to two). Table(5.4) is a simulation results of the design gain for different correlation factor and number of survivors. The design gain increases as the number of survivors increases and/or the correlation factor increases.

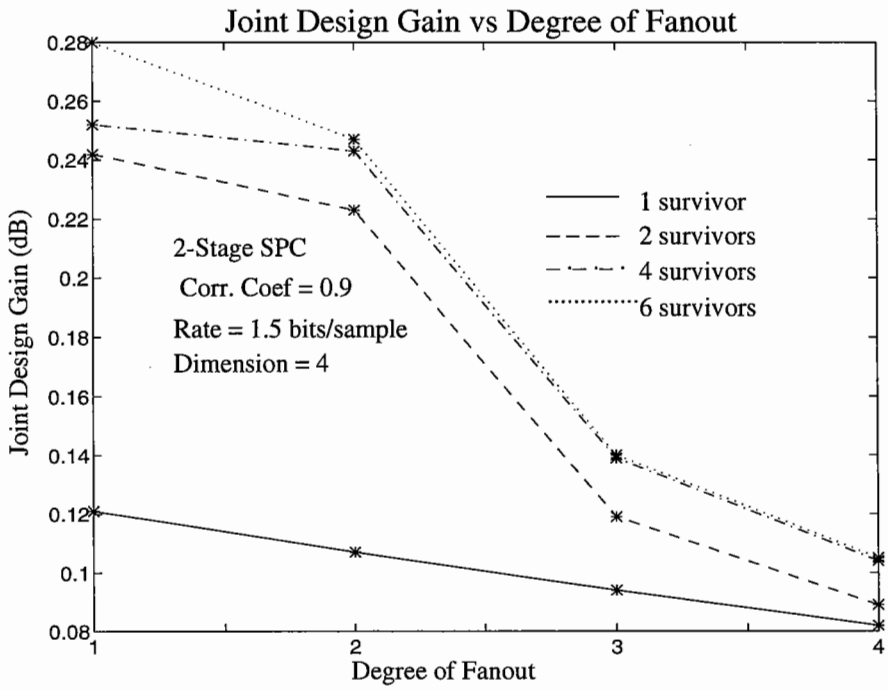


Figure 5.5: Joint design gain vs degree of fanout; Gaussain AR(1), correlation factor =.9.

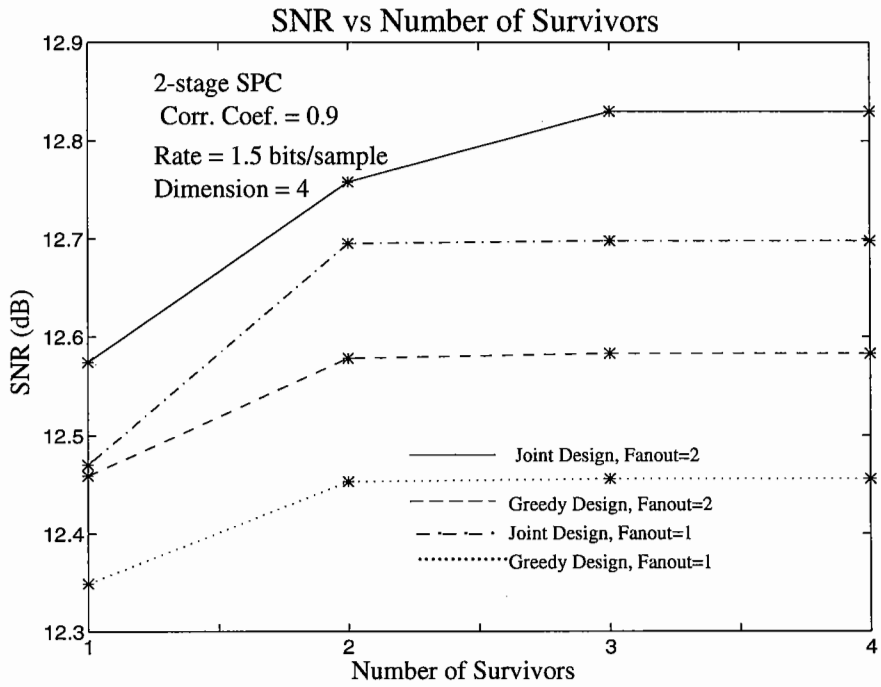


Figure 5.6: Signal to noise ratio vs number of survivors; number of stages=2; dimension of vectors =4; Gaussian AR(1), correlation factor=.9.

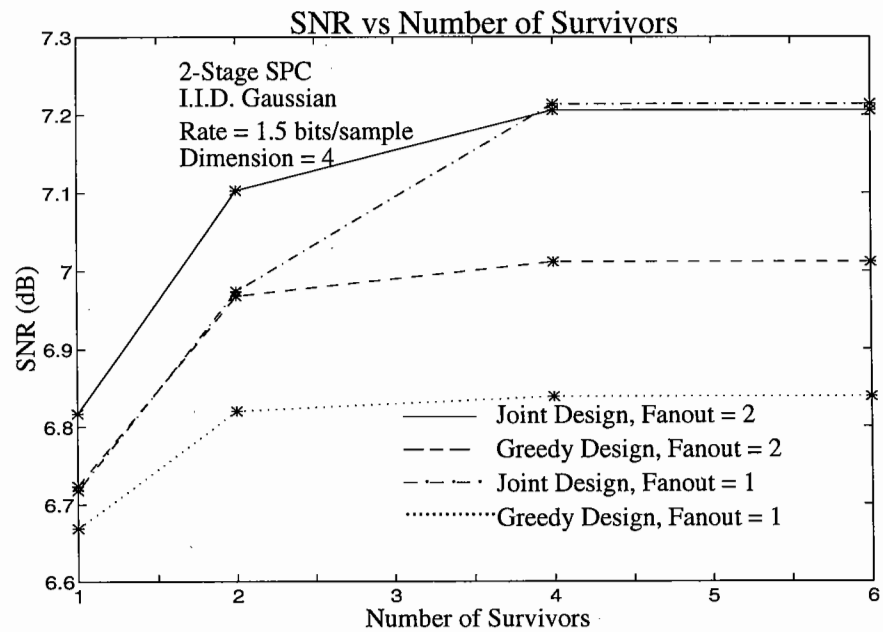


Figure 5.7: Signal to noise ratio vs number of survivors; i.i.d. Gaussian source.

Correlation factor	No. of sur	Encod Complex.	Joint design SNR(dB)	Design gain
.1	1	16	6.77	0.03dB
.1	2	24	7.08	0.10dB
.1	4	40	7.2	0.15dB
.5	1	16	7.86	0.10dB
.5	2	24	8.14	0.14dB
.5	4	40	8.21	0.17dB
.9	1	16	12.57	0.12dB
.9	2	24	12.76	0.18dB
.9	4	40	12.83	0.25dB

Table 5.4: Design gain vs number of survivor and correlation factor. Dimension =4, number of stages=2; number of bits/stage=3; degre of fanout =2.

No. of stage	bit/ stage	Corr. fact.	No. of N sur	SNR	Reference(SNR) $N = 1, M = 1$	Gain (dB)
6	1	.1	2	6.12dB	6.40dB	(0.28)
3	2	.1	4	6.06dB	6.27dB	(0.21)
6	1	.5	2	6.47dB	6.87dB	(0.40)
3	2	.5	4	8.89dB	9.33dB	(0.44)
6	1	.9	2	11.07dB	11.92dB	(0.85dB)
3	2	.9	4	13.34dB	14.08dB	(0.74dB)
6	1	.95	2	13.35dB	13.89dB	(0.53dB)
3	2	.95	4	15.68dB	16.51dB	(0.82dB)

Table 5.5: Multi-stage joint design using N survivors for design and M survivors for actual testing, $N > M$; where $M = 1$, dimension =4.

Multiple survivor

Even though multiple-survivors help to increase joint-design gain, there is no advantage in keeping more survivors during design phase than the number actually used in encoding. It is not beneficial to a codebook for M-survivor encoding with N-survivors, where $N > M$. In fact, the performance of the codebook deteriorates if more survivors are used in design than encoding (Table(5.5)). We conclude that the number of survivors for design and actual encoding should be the same.

5.2 Predictive Split VQ (PSVQ)

As pointed out in the previous chapter, SVQ is related to MSVQ in that some components of the feature vectors in MSVQ are constrained to zero. Hence, the performance of SVQ is upper-bounded by that of MSVQ. Between MSVQ and SVQ, we formed a

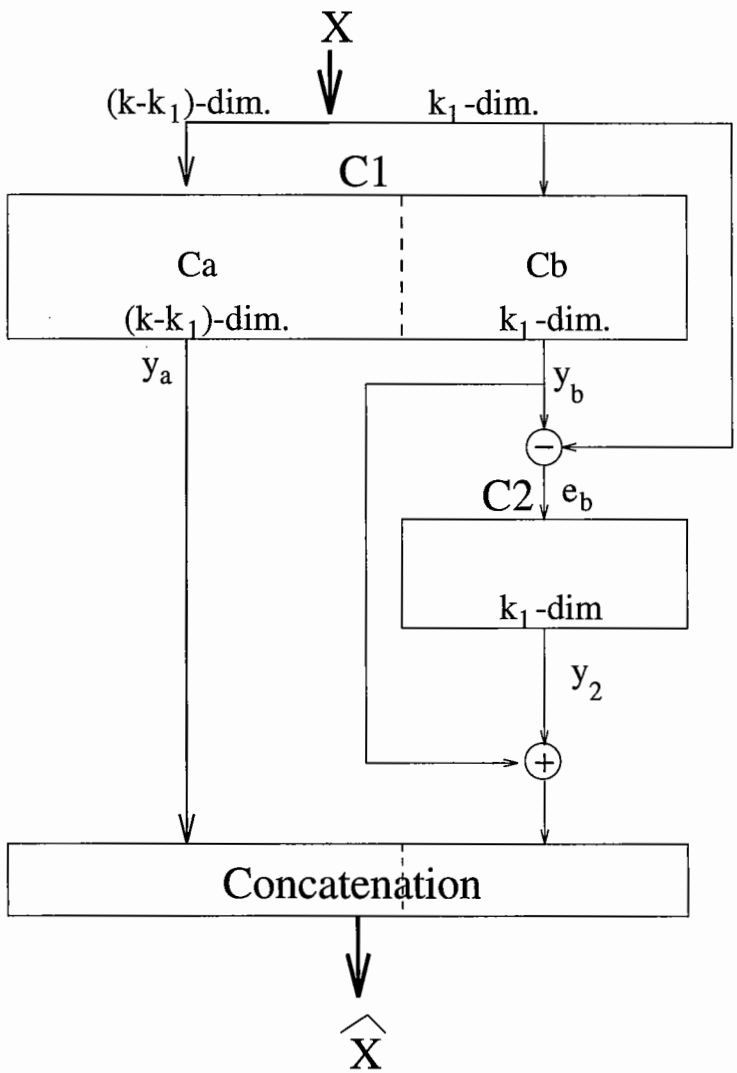


Figure 5.8: Predictive split vector quantizer

hybrid structured VQ, namely PSVQ.

Let X be a source of vector dimensions k. Suppose that there are only two stages of PSVQ. The first stage of PSVQ contains a codebooks C_1 of dimension k and size $L_1 = 2^{b_1}$. The second codebook C_2 has dimension k_1 and size $L_2 = 2^{b_2}$. Fig(5.8) shows the structure of PSVQ. For encoding, the first stage codebook can be regarded as consisting of two codebook C_a and C_b with dimensions $k-k_1$ and k_1 respectively. Correspondingly, the input vector X is divided into two subvectors X_a and X_b of dimensions $k-k_1$ and k_1 respectively. For a MSE distortion criterion, the first subvector X_a is encoded by searching through the codebook C_a to produce index i, where

$$\|X_a - y_{a_i}\|^2 \leq \|X_a - y_{a_j}\|^2 \quad i \neq j \tag{5.4}$$

y_{a_i} and y_{a_j} are the entries codewords in the codebook C_a; and $i, j \in \{1, 2, \ldots, L_1\}$. The index i is then used to obtain from a predictive codebook C_b, a codevector y_{b_i}. The initial predictive codebook C_b is obtained by first assigning the training vector into its corresponding regions using the index provided by encoding with codebook C_a and then taking the centroid of each regions as the codevector. This encoding process uses the first $k-k_1$ components to predict non-linearly the last k_1 components of the vector. A residual vector e_b which is the difference between the input vector X_b and codevector y_{b_i} is fed to the second stage quantizer which then generate output y_2. The reproduced vector $\hat{x}$ is the sum $\hat{x} = y_1 + y_2'$ where y_1 is y_{a_i} concatenated with y_{b_i}, and $y_2{}'$ is a zero vector of dimension $k-k_1$ concatenated with y_2. PSVQ encoding search has about the same search complexity as SVQ. One major difference between SVQ and PSVQ encoding is that the error vector e_b instead of the original input subvector X_b, is fed into the second stage quantizer.

As shown in Fig(5.9),(5.10),(5.11), PSVQ designed using the GLA performs worse than SVQ for a low correlation source. GLA-designed PSVQ performs better than SVQ only for a high correlation source. Thus, the greedy GLA design is not suitable for generating PSVQ codebooks. For subsequent simulations, PSVQ is designed using the

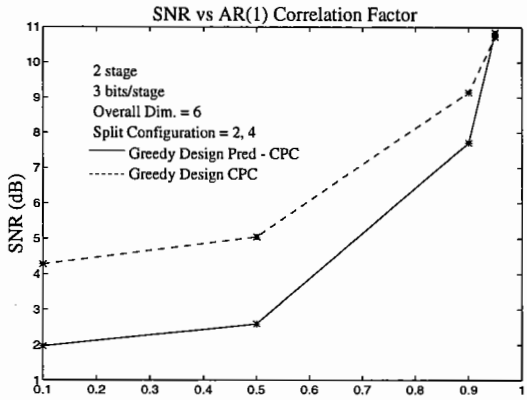


Figure 5.9: Signal to noise ratio vs correlation factor. $k - k_1 = 2, k_1 = 4$

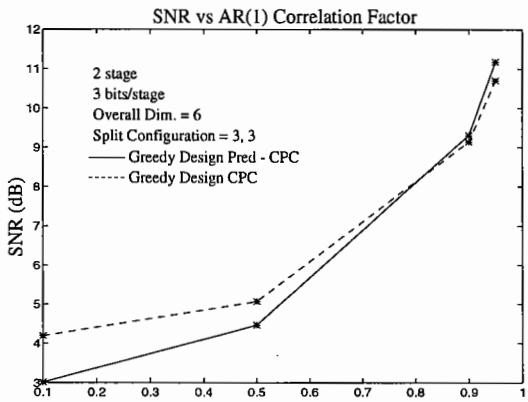


Figure 5.10: Signal to noise ratio vs correlation factor. $k - k_1 = 3, k_3 = 3$

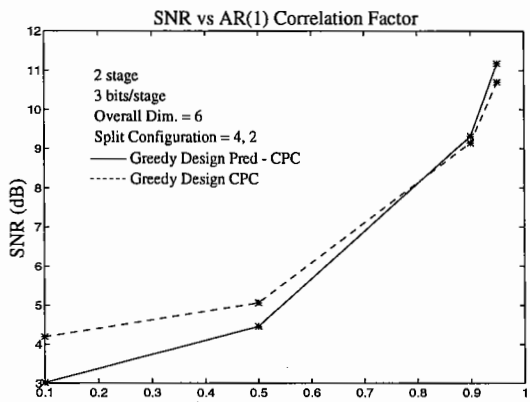


Figure 5.11: Signal to noise ratio vs correlation factor. $k - k_1 = 4, k_1 = 2$

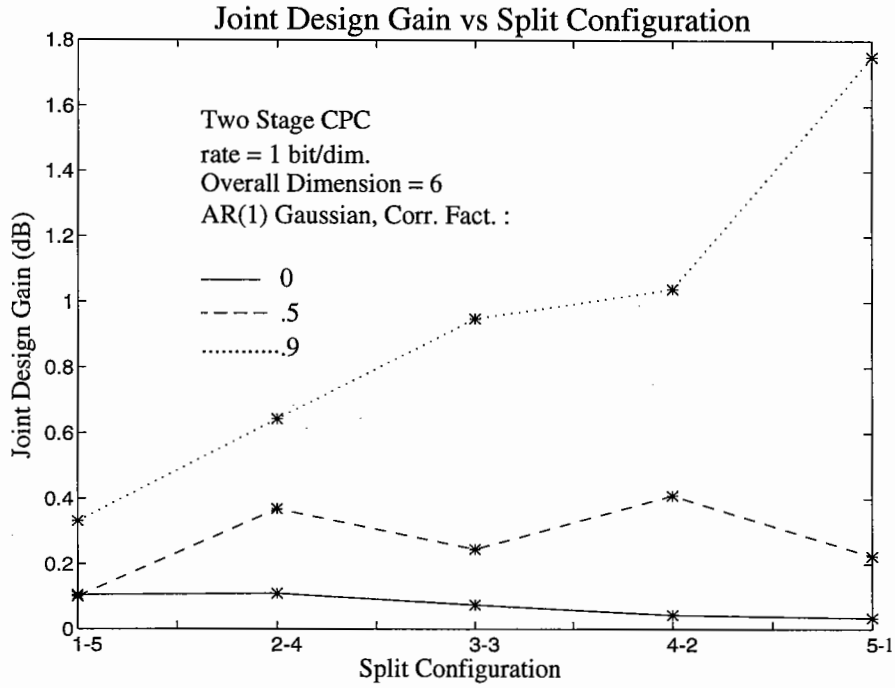


Figure 5.12: Joint design gain vs split configuration. Split configuration : $k - k_1, k_1$.

joint design algorithm, and the initial codebooks are generated using the GLA algorithm. The term "joint design gain" used in the remaining section is the gain of joint design PSVQ over SVQ.

Fig(5.12) and Table(5.6) show the joint design gain versus different split configurations for 1 bit/dim. The joint design gain of PSVQ over SVQ increases as the number of bits assigned to the first stage decreases (i.e. k_1 increases) for an i.i.d Gaussian source. On the other hand, by increasing the number of bits for the first stage for the high-correlated source, we increase the joint-design gain. Fig(5.13) shows that for low correlation, assigning more bits/dim to the second stage residual quantizer improves the overall performance. However, for high correlation source, assigning more bits/dim to the second stage quantizer degrades the performance. Thus, the optimal split configuration is source dependent. For low correlation, there is only a moderate gain of PSVQ over SVQ as seen in Fig(5.14). However, as correlation increases, the performance of joint-design PSVQ approaches the performance of joint design MSVQ. Moreover, with

Split Config-uration	Corr. factor	PSVQ Storage for Complexity	SVQ Storage Complexity	PSVQ (SNR)	Gain Over SVQ
1-5	ind	172	162	4.67dB	0.11dB
2-4	ind	88	72	4.51dB	0.11dB
3-3	ind	72	48	4.47dB	0.07dB
4-2	ind	104	72	4.55dB	0.04dB
5-1	ind	194	162	4.62dB	0.03dB
1-5	.9	172	162	9.11dB	0.33dB
2-4	.9	88	72	9.91dB	0.64dB
3-3	.9	72	48	10.22dB	0.95dB
4-2	.9	104	72	10.33dB	1.04dB
5-1	.9	194	162	10.57dB	1.75db

Table 5.6: PSVQ vs SVQ, with bit/dim=1. Split configuration : $k - k_1, k_1$

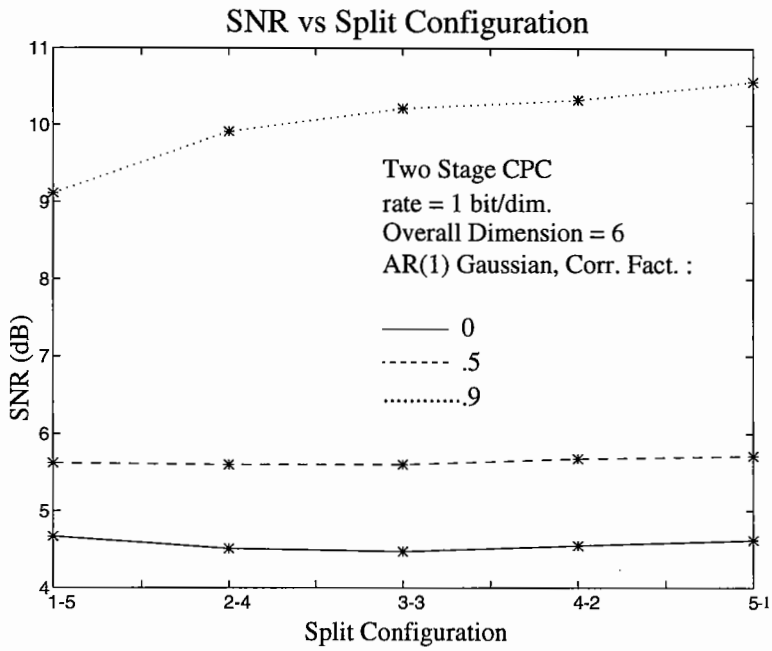


Figure 5.13: Signal to noise ratio vs split configuration. Split configuration: $k - k_1, k_1$

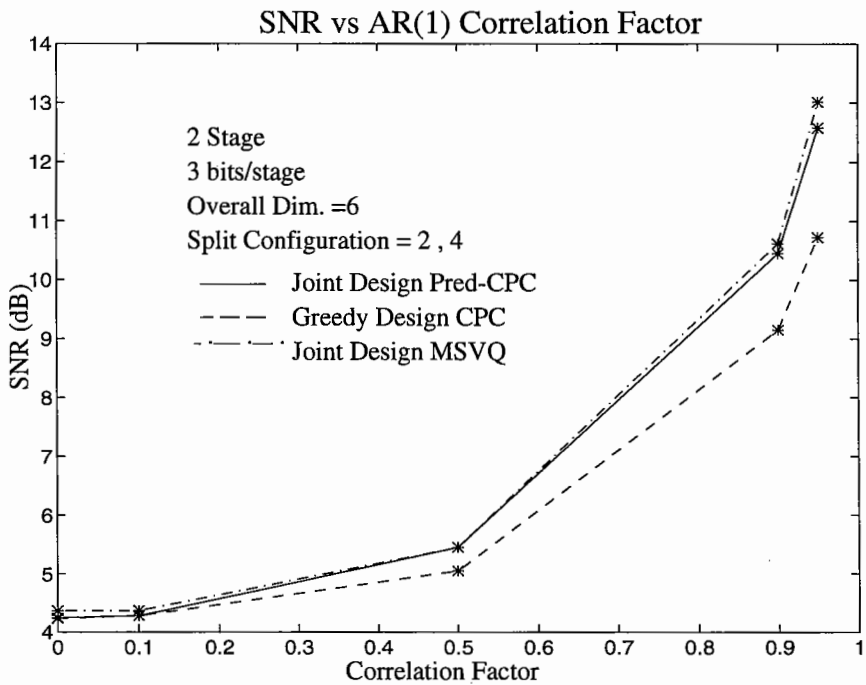


Figure 5.14: Signal to noise ratio vs AR(1) Gaussian. Split configuration : $k - k_1, k_1$

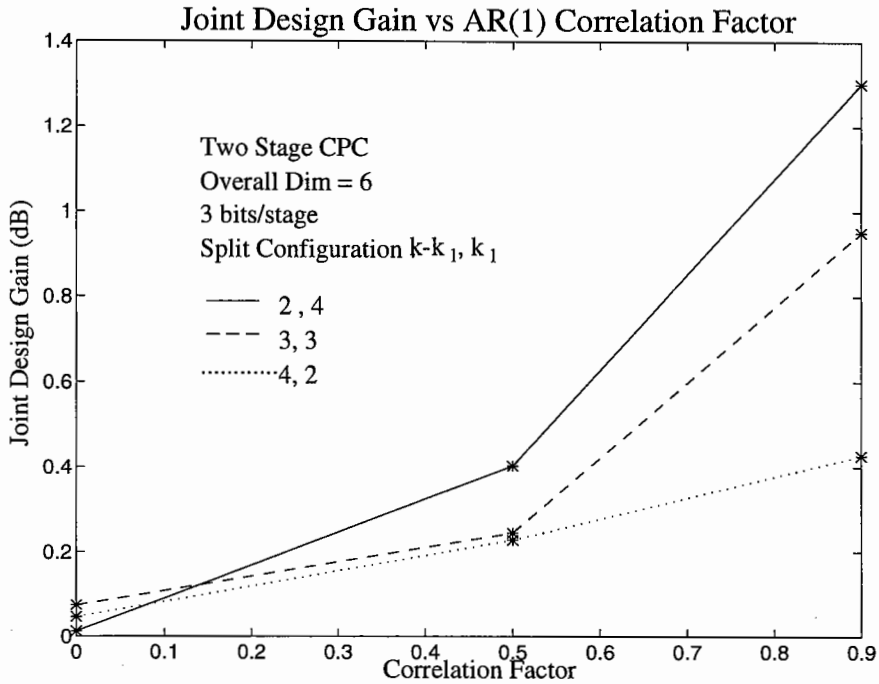


Figure 5.15: Joint design gain vs AR(1) Gaussian.

multiple survivor search, (Fig(5.17)), we can improve the joint design gain of PSVQ over SVQ. We conclude that the number of components $k - k_1$ needed for prediction is proportional to the correlation source.

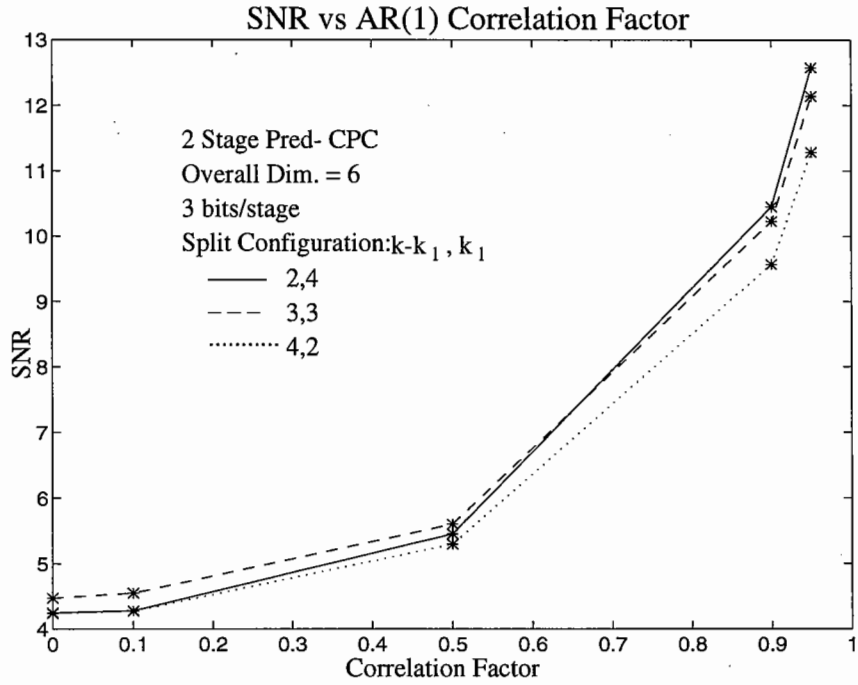


Figure 5.16: Signal to noise ratio vs AR(1) Gaussian; 3 bits/stage.

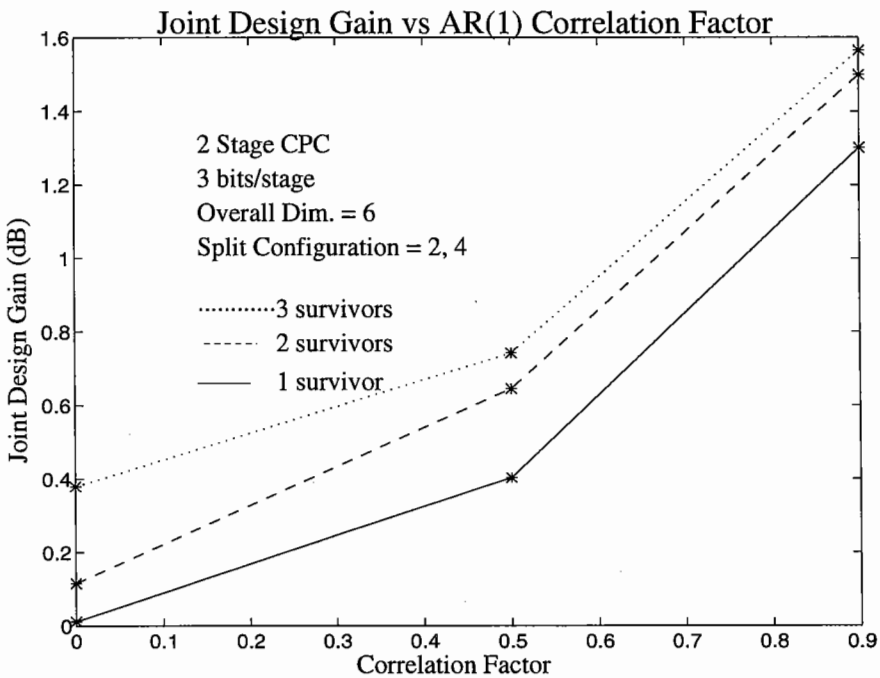


Figure 5.17: Joint design gain vs AR(1) Gaussian, Split config: $k - k_1 = 2, k_1 = 4$, multiple survivors.

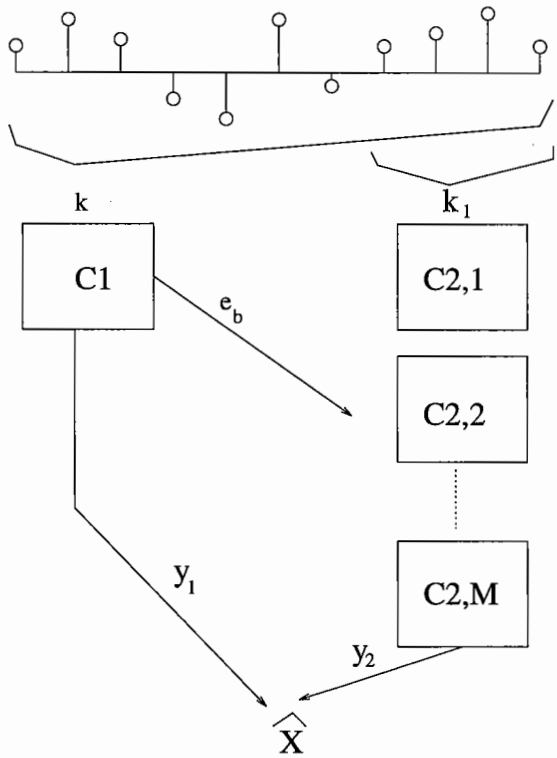


Figure 5.18: Constrained storage predictive split vector quantizer

5.3 Constrained Storage Predictive Split VQ (CPSVQ)

As we increase the degree of fanout as shown in Fig(5.18), another new SPC structure is obtained: constrained storage predictive split VQ (CSPVQ). The way to construct the multiple codebooks is as in regular constrained storage VQ [34].

Let X be a source of dimension k. Let us assume there are only two stages. The first stage contains a single codebook C_1 dimension k and size $L_1 = 2^{b_1}$. The second stage quantizer contains M codebooks $\{C_{2,1}, \ldots, C_{2,M}\}$. Each codebook $C_{2,i}$ has a dimension k_1 and size $L_2 = 2^{b_2}$. Fig(5.18) shows the structure of CPSVQ. Encoding in CPSVQ is the same as in PSVQ. The first stage quantizer encodes the input vector X with the modified nearest neighbour condition Eq(5.4). The decision is based on the first $k - k_1$ components of the input vector X using codebook C_1. The quantizer produces an index

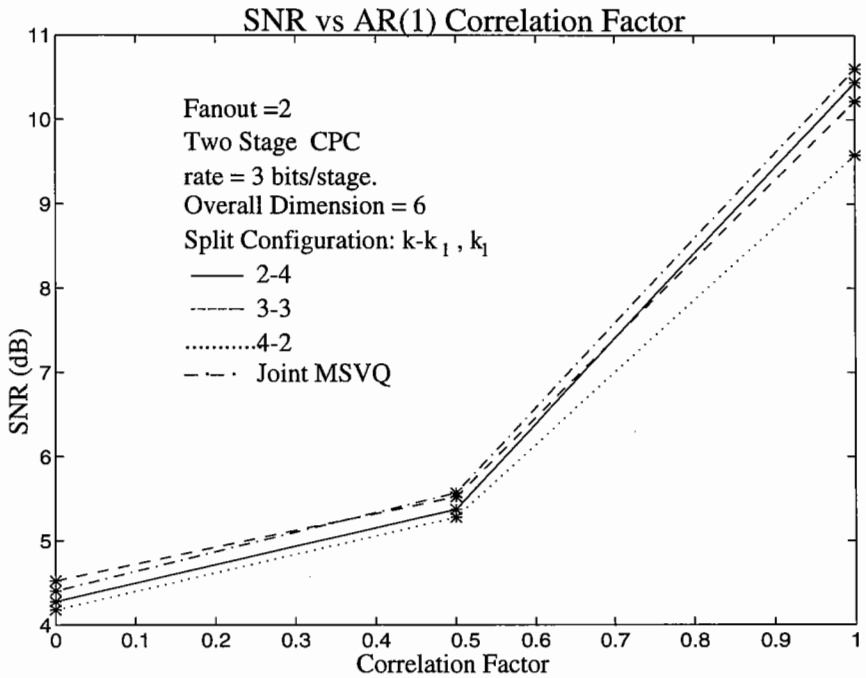


Figure 5.19: Signal to noise ratio vs AR(1) Gaussian. Split configuration: $k - k_1, k_1$

i, a residual vector e_b which is the difference between the last k components of X and y_1, and the value of the pointer μ_i giving the address of the second stage codebook. The residual vector e_b is then encoded by a second stage codebook whose address is given by μ_i and quantized to the output vector y_2. The reproduced vector $\hat{X} = y_1 + y_2$ Table(5.7) for PSVQ with fanout equal to two, an even split configuration which consists of the lowest storage complexity, has a better distortion-performance tradeoff than the rest. Fig(5.20) and Fig(5.21) show the SNR performance of the CPSVQ versus number of survivors. For the high correlation source, a small number of survivors is needed to achieve the best performance a CPSVQ quantizer can furnish. By assigning more bits/dim to the first stage, we get a better quantizer performance. Fig(5.22) shows that the performance of the quantizer improves when the degree of fanout is increased to two and then levels off when the fanout is further increased.

Moreover, when comparing the performance of joint design CPSVQ and GLA-designed CSVQ with degree of fanout equal to two for a fixed bit allocation, Table(5.7)

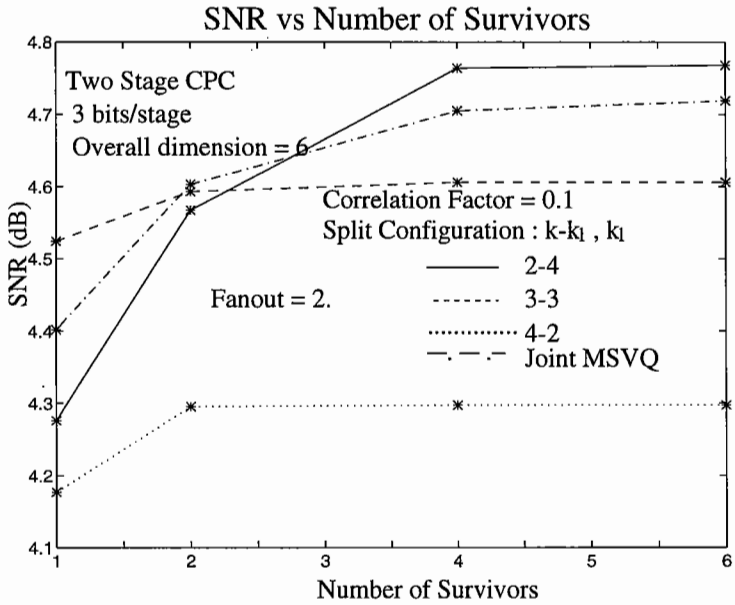


Figure 5.20: Signal to noise ratio vs number of survivors. Split configuration: $k - k_1, k_1$

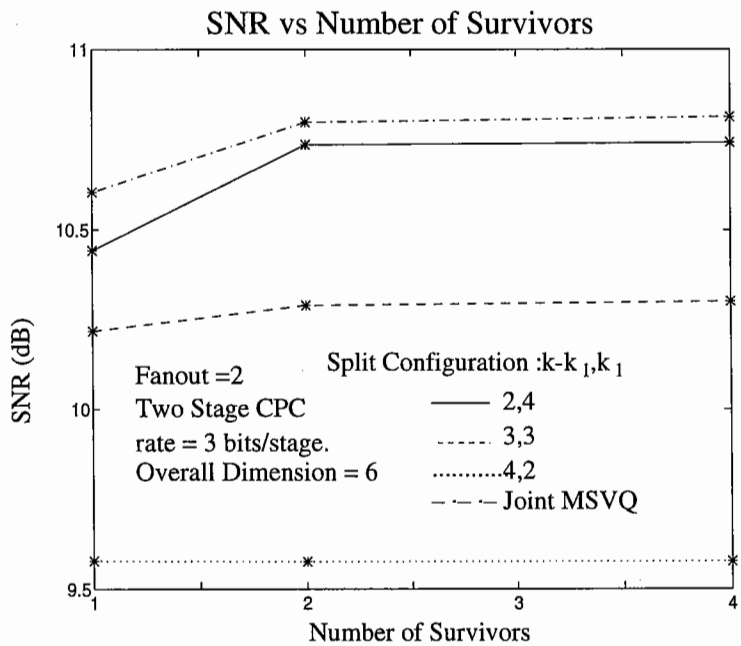


Figure 5.21: Joint design gain vs number of survivors, correlation factor=.9. Split configuration: $k - k_1, k_1$

Split Config-uration	Corr. factor	Storage Complexity for CPSVQ	Storage complexity for CSVQ	PSVQ SNR	Gain Over CSVQ
2-4	ind	152	216	4.248dB	0.016dB
3-3	ind	96	216	4.467dB	0.071dB
4-2	ind	112	216	4.238dB	0.047dB
2-4	.9	152	216	10.416dB	1.267dB
3-3	.9	96	216	10.209dB	0.935dB
4-2	.9	112	216	9.574dB	0.432dB

Table 5.7: Joint design CPSVQ vs GLA design CSVQ, with degree of fanout=2, and 3 bits/stage. Bit allocation fixed. Split configuration: $k - k_1, k_1$

shows that joint design CPSVQ actually outperforms conventional design CSVQ for the high correlation source. There is a significant decrease in both search and storage complexity. However, for a low correlation source, there is no performance gain over CSVQ, but only gaining in searching and storage complexity.

5.4 Line Spectral Frequency Parameters

The performance objective to be minimized is the average spectral distortion (SD) between quantized and unquantized LPC parameter vectors. However, due to the complex dependence between LSF's and the spectral envelope and the complexity of SD computation, there is no easy way of incorporating SD into the codebook design algorithm. For this reason, a weighted Euclidean distortion measure (defined in Section 2.7.2) is used to design the codebooks. Such a distance measure gives more weight to perceptually significant vector components. While SD is difficult to use in codebook design, it can

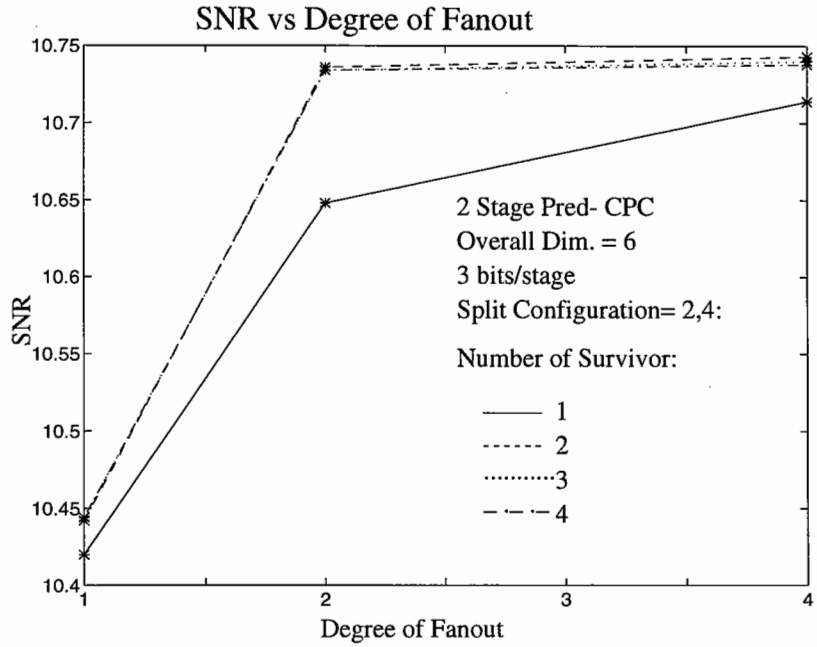


Figure 5.22: Signal to noise ratio vs degree of fanout, correlation factor =.9.

be used in encoding. However, because of the computational cost of calculating SD, it is not realistic to perform the entire codebook search using that distortion function. On the other hand, in a multiple survivor search, SD can be used to select the best candidate from the final stage survivor set. This was done in our simulation of SVQ and PSVQ, where m survivors were kept for each feature and the SD was used to select the best quantized output from the last survivor set.

One of the major advantages of the LSF representation is that as long as the ascending order of the LSF vector components is preserved in the quantization process, the quantized synthesis filter is guaranteed to be stable. When the initial codebook of SVQ are designed using the generalized Lloyd algorithm (modified for the WMSE distortion measure Eq(2.40)), the centroid for each quantization region is obtained by simply averaging all training vectors in that region, as would be done for the MSE criterion, and thereby stability is guaranteed. For PSVQ codebooks, due to its constituent stage structure, a stability check is imposed in each iteration during the design phase. If any

Split Configuration	Ave. SD.	$2-4$dB % outlier	>4dB % outlier
11-10	1.33	5.47	0.0
10-11	1.24	2.92	0.0
11-11	1.17	1.88	0.0
12-11	1.12	1.36	0.0

Table 5.8: Performance of SVQ with LSF parameters.

of the updated codevectors has an ordering problem, that particular updated codevector will be replaced by the previous updated value.

A 24.5 minutes of speech signal is first passed through a low pass filter at a cut off frequency of 3.4kHz and sampled at a sampling frequency of 8kHz. Frames with energy below a threshold were classified as "silence" and rejected of the training set. The digitalized speech signal is analyzed using a 10th order LPC and a 20ms window. Moreover, consecutive analysis frames overlap by half a frame, resulting in a training set size of 144000 vectors. Bandwidth expansion was also used: the ith prediction coefficient is multipled by γ^i, where $i = 1, \ldots, 10$ and γ is a constant equal to 0.996. The performance of the quantizer was then evaluated with a test set comprising of 7700 vectors which were generated independently from the training set.

When comparing SVQ and PSVQ, there is a steady 0.04dB SD gain for one survivor encoding. This is not a significant gain. However, when the number of survivors increases to 8, there is a 0.08dB gain, which is almost a gain of one bit. Therefore, PSVQ saves 1 bit/frame over SVQ.

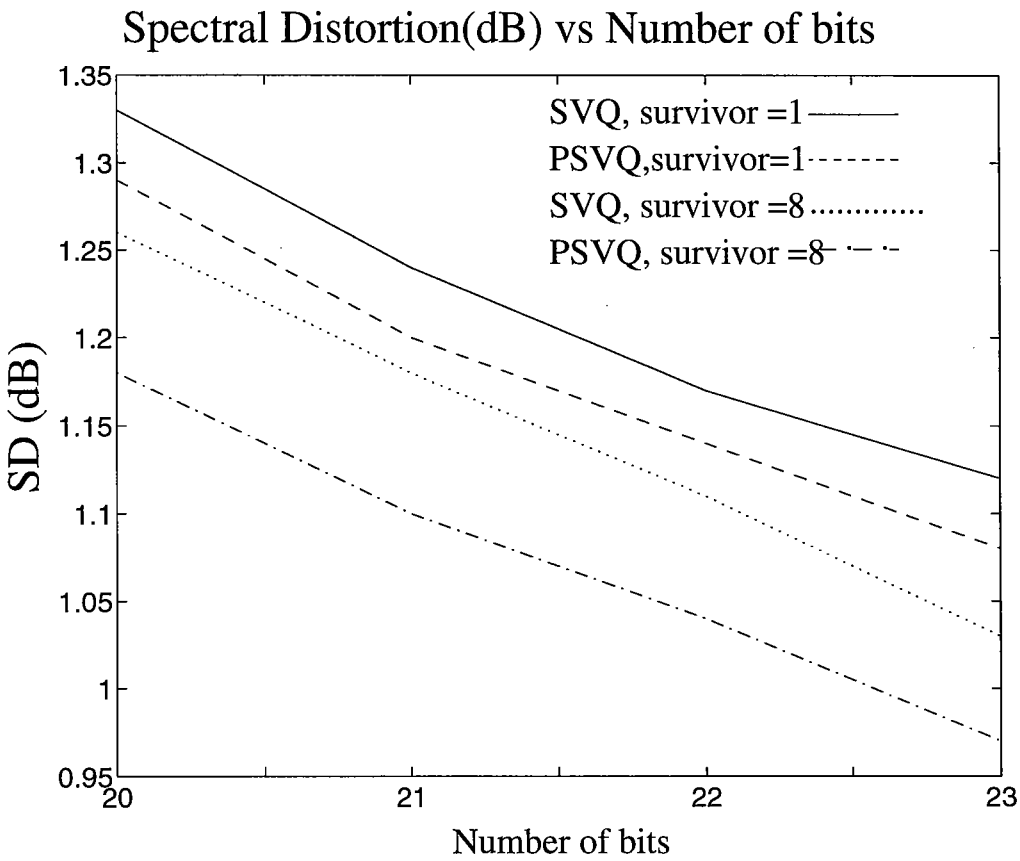


Figure 5.23: Number of survivors vs spectral distortion.

Split Configuration	Ave. SD.	$2-4$dB % outlier	>4dB % outlier
10-10	1.29	5.86	0.0
10-11	1.20	3.71	0.0
11-11	1.14	3.00	0.0
12-11	1.08	2.05	0.0

Table 5.9: Performance of PSVQ for LSF parameters.

Split Configuration	Ave. SD.	$2-4$dB % outlier	>4dB % outlier
10-10	1.26	4.01	0.0
10-11	1.18	2.56	0.0
11-11	1.11	0.82	0.0
11-12	1.03	0.35	0.0

Table 5.10: Performance of SVQ for LSF parameters, survivor=8.

Split Configuration	Ave. SD.	$4-2$dB % outlier	>4dB %outlier
10-10	1.18	1.65	0.0
10-11	1.10	1.17	0.0
11-11	1.03	0.93	0.0
11-12	0.97	0.51	0.0

Table 5.11: Performance of PSVQ for LSF parameters, survivor=8.

Chapter 6

Conclusion

The purpose of this thesis is to examine candidate vector quantization techniques for coding Linear Prediction (LP) coefficients and the efficiency of a joint design algorithm for various summation product code structured quantizers. LP coefficients are used often in low-bit rate speech coders as they provide an economical representation of the spectral envelope of speech signals. Thus, efficient representation of the LPC parameters is a central issue in speech coding. The most often used representation of the LP coefficients are reflection coefficients and the Line Spectral Frequencies (LSF). The LSF parameters have two properties which make them amenable to quantization: their ordering property which relates to the stability condition and their localization property. Due to these properties, for the same quantization scheme, quantizers which operate on LSF parameters are more efficient than quantizers which operate on many other LPC representations.

Transparent quantization of LPC parameters is a crucial condition for achieving toll-quality. Vector quantization is one of the techniques which is capable of yielding less than 1dB spectral distortion at low bit rates. However, complexity problems quickly arise with unstructured vector quantization as the codebook size grows. The summation

product codes (SPC) are a family of structured VQ's that can circumvent this complexity obstacle. The performance of the SPC quantizer can be varied against its storage and encoding complexity. Apart from the storage and encoding complexity of the SPC VQ, the design algorithm can also affect the efficiency of the quantizer. The Generalized Lloyd's Algorithm (GLA) is traditionally used to design the constituent codebooks of a SPC on a stage-by-stage basis. However, it has been shown in this thesis that joint codebook design can construct better stage codebooks. We have studied a joint design algorithm which updates all the codebooks simultaneously by solving a weighted least squares system of equations. These equations are derived from the notion that every SPC quantizer has an equivalent reference product codebook which minimizes the overall distortion. This equivalent reference codebook can be determined by first encoding the source vectors into their corresponding partition regions and then taking the centroid of all the source vectors assigned to each particular region. By minimizing the difference between the reference product codebook and the equivalent reproduced product codebook, a matrix equation is obtained. However, this matrix equation is rank deficient. An iterative method is used in this paper to solve this rank deficient system.

Using 25000 training vectors, 5000 test vectors and a mean squared error (MSE) distortion criteria, the performances of GLA design and joint design were compared. Simulation was first performed for MSVQ, and the following conclusions were drawn. The joint design gain over conventional GLA design increases as source correlation increases. The design gain also increases with the number of stages. With GLA design, the average distortion increases when complexity is reduced by increasing the number of stages. However, joint design can result in a relatively stable signal-to-noise ratio (SNR) even the number of stages increases. This indicates that joint design can improve the complexity-distortion tradeoff significantly. Unfortunately, joint design does not improve the codebook with a single survivor search at low bit rates. In order to improve the performance at low bit rates, multiple survivors have to be maintained. Since the encoding complexity of multiple survivor search at low bit rate is relatively small,

keeping multiple survivors is justified. By increasing slightly the number of survivors, the joint design algorithm can yield significantly better performance. Also the number of survivors needed is inversely proportional to source correlation. However, there is no benefit to performing a joint design with more survivors than necessary. In fact, when more survivors than required are used in design , the SNR performance of the quantizer dropped. Thus the number of survivors for both design and actual encoding should be the same.

A further examination of the joint design algorithm was done for SPCs with greater than unity fanout. It has been found that the joint design algorithm improves the performance of CSVQ only moderately. In fact, SPC with unity fanout can perform better than the one with degree of fanout greater than one. As a result, there is no advantage to do a joint design on SPCs with fanout greater than one.

An innovative SPC structure codebook, predictive split VQ (PSVQ), is also tested in this thesis. PSVQ has the same encoding complexity as split VQ (SVQ) but with a higher storage complexity. Designing PSVQ with GLA results in performance worse than regular SVQ for a low correlation source. In fact, the PSVQ quantizer designed using the GLA algorithm only performed slightly better than SVQ for a high correlation source.

On the other hand, with the joint design algorithm, the performance of PSVQ can perform better than SVQ. The performance of PSVQ also depends on the correlation of the source. For low correlation, PSVQ has only a slight performance gain over SVQ. With increasing correlation, the performance of PSVQ approaches that of joint design MSVQ. The choice of the split configuration is also source dependent. For low correlation and a fix rate (bit/dim) for each stage, the performance of the quantizer improves as the number of bits assigned to the first stage decreases. On the other hand, with high correlation, the number of bits assigned to the first stage has to be increased in order to produce better performance. By increasing the degree of fanout in PSVQ, we achieved a

new structure: constrained storage predictive split VQ (CPSVQ). Joint-design CPSVQ can perform slightly better than CSVQ but with lower storage and search complexity.

In conclusion, the joint design algorithm can improve the performance of SPCs without inducing any increase in real time complexity. As a result, it is almost a free gain. However, there is a major disadvantage of using the joint design algorithm. The procedure of generating the codebooks using this joint design algorithm is very time consuming.

Lastly, the PSVQ quantizer was used for coding LSF parameters. The performances of PSVQ and SVQ were examined. By using the joint-design algorithm, PSVQ introduced a significantly lower average spectral distortion and fewer outlier than SVQ. A 23 bit PSVQ quantizer yielded an average spectral distortion of 1.03 dB, 0.93% outliners in the range 2-4dB, and 0.03% outliners in the range $>$ 4dB and achieved transparent quality. Comparing with SVQ, a reduction of 1 bit/frame for coding LSF parameter can be achieved by using PSVQ with eight survivors.

6.0.1 Suggestions for further investigation

One area of improvement that can be explored is in the structure of SPCs. In this thesis, each stage consists of only codebooks. However, a more sophisticated structure can be formed by considering each stage as a structured VQ by itself. One example is the Tree-MSVQ proposed by Chemla et al [37]. Another area that is worth investigating is the technique of prediction used for generating the second-stage codebook in PSVQ. It has been shown that the prediction method used in this paper is not sufficient when low-correlation source is used. One suggestion is to use linear prediction to predict the second stage of PSVQ. Finally, better results than those achieved in this work can probably be obtained by solving the matrix equations of the joint design method with a better technique .

Bibliography

[1] P. Kroon and B.S. Atal, "Predictive coding of speech using analysis-by-synthesis techniques", in *Advances in Speech Signal Processing*, S. Furui and M.M. Sondhi, Eds. (New York, NY: Marcel Dekker), 1991.

[2] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Comm.*, vol. COM-32, pp. 169-176, 1984.

[3] T. M. Cover and P.E. Hart, "Nearest neighbor pattern classification", *IEEE Trans. Inform. Theory*, vol. 13, pp. 21-27, 1967.

[4] D. Y. Cheng and A. Gersho, "A fast Codebook Search Algorithm for Nearest-Neighbor Pattern Matching," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.* , (Tokyo), pp. 265-268, 1986.

[5] A. Madisetti, R. Subramonian, and V. R. Algazi, "A radius-bucketing approach to fast vector quantization encoding," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, (Glasgow), pp. 1767-1770, 1989.

[6] Y. W. Chan,"The design of generalized product codebook vector quantizers", *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, (San Francisco), Vol. 3, pp. 389-392, 1992.

[7] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, Kluwer Academic Publishers, 1991.

[8] J. Makhoul,"Linear prediction: A tutorial review," *Proceeding of the IEEE* 63(4), pp. 561-580, 1975.

[9] P. Kabal and R. P. Ramachandran, "The computation of line spectral frequencies using Chebyshev polynomials," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, ASSP-34(3), pp. 1419-1426, 1986.

[10] F. Itakura,"Line spectrum representation of linear predictive coefficients," *J. Acoust. Soc. Am.* 57 Supplement 10, p. S.35, 1975.

[11] G. S. Kang and L. S. Fransen, "Application of line spectrum pairs to low bit rates speech encoders," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, (Tampa), pp. 7.3.1-7.3.4, 1985.

[12] B. H. Juang, D. Y. Gray and A. H. Gray Jr., "Distortion performance of vector quantization for LPC voice coding," *Proc. IEEE Int. Conf. Acoust. Speech and Signal Proc.*, ASSP-30, pp. 294-303, 1982.

[13] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantization design," *IEEE Trans. on Comm.*, COM-30, pp. 84-95, 1980.

[14] J. Grass and P. Kabal, "Quantization of predictor coefficients in speech coding,"*Rapport Technique de L'INRS-Telecommunication* no. 91-01, 1991.

[15] K. K. Paliwal and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 Bits/Frame," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.* , (Toronto), pp. 661-664, 1991.

[16] E. Paksoy, W. Y. Chan and A. Gersho, "Vector quantization of speech LSF parameters with generalized product codes," *Proc. IEEE Int. Conf. on Spoken Language*, (Banff), 1992.

[17] A. H. Gray and J. D. Markel,"Quantization and bit allocation in speech processing," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, ASSP-24, pp. 459-573, 1976.

[18] L. Rabiner and R. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, (NJ) 1979.

[19] H.Abut, R. M. Gray, and G. Rebolledo, "Vector quantization of speech waveforms,"*Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, pp. 12-15, 1981.

[20] B. S. Atal and M. R. Schroeder, "Predictive coding of speech signals and subjective error criteria," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, (Tulsa), pp. 573-576, 1978.

[21] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel," A two-step speech compression system with vector quantizing," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, pp. 52-55, 1979.

[22] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, " Speech coding based upon vector quantization," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, Vol. ASSP-28,No.5, pp. 562-574, 1980.

[23] J. H. Conway and N. J. A. Sloane, " Voronoi regions of lattices, second moments of polytopes and quantization," *IEEE Trans. on Inform. Theory*, Vol. IT-28, No. 2, pp. 211-226, 1982.

[24] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. on Inform. Theory*, Vol. IT-25, No. 4, pp. 373-380, 1979.

[25] A. Gersho, "On the structure of vector quantizers, " *IEEE Trans. on Inform. Theory*, Vol. IT-28, No. 2, pp. 157-166, 1982.

[26] R. M. Gray, A. Buzo, A. H. Gray, Jr., and Y. Matsuyama, "Distortion measures for speech processing," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, Vol. ASSP-28, No. 4, pp. 367-376, 1980.

[27] R. M. Gray and Y. Linde, "Vector quantizers and predictive quantizers for Gauss-Markov sources," *IEEE Trans. on Comm.*, Vol. COM-30, No.2, pp. 381-389, 1982.

[28] D. O'Shaughnessy, *Speech Communication*, Addison-Wesley, Don Mills, Ontario, 1987.

[29] F. K. Soong and B. H. Juang, "Line spectrum pair (LSP) and speech data compression", *Proc. IEEE Int. Conf. Acoust. Speech and Signal Proc.*, (San Diego), pp. 1.10.1-1.10.4, 1984.

[30] N. Sugamura and N. Farvardin, "Quantizer design in LSP speech analysis-synthesis", *IEEE J. on Selected Areas in Commun.*, vol. 6, 1988.

[31] R. M. Gray, "Vector quantization", *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, pp. 4-29, 1984.

[32] T. D. Lookabaugh and R. M. Gray, "High-resolution quantization theory and the vector quantizer advantage," *IEEE Trans. Inform. Theory* vol.35, pp. 1020-1033, 1989.

[33] W. Y. Chan and A. Gersho, "Constrained storage quantization of multiple vector sources by codebook sharing," *IEEE Trans. Comm.*, vol. COM-38, no.12, pp. 11-13, 1991.

[34] W. Y. Chan and A. Gersho, "Enhanced multistage vector quantization with constrained storage," *Proc. 24th Asilmar Conf. Cir., Sys., and Comp.*, pp. 659-663, 1990.

[35] W. Y. Chan, S. Gupta and A. Gersho, "Enhanced multistage vector quantization by joint codebook design," *IEEE Trans. Comm.*, November 1992.

[36] D. H. Lee, D. L. Neuhoff and K.K. Paliwal, "Cell-conditioned multistage vector quantization," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, pp. 653-656, (Toronto), 1991.

[37] D. Chemla, S. W. Soong and W. Y. Chan "Tree-structure vector quantization of speech LSF parameters", *IEEE Workshop on Speech Coding for Telecommunications*, 1993.

[38] R. Viswanathan and J. Makhoul, "Quantization properties of transmission parameters in linear predictive systems", *Proc. IEEE Int. Conf. Acoust., Speech and Signal Proc.*, vol. ASSP-23, pp. 309-321, 1975