A Study of Index Assignments

Jen-der Day

Thesis presented for the degree of

Doctor of Philosophy

University of Edinburgh

1999

Contents

Declaration

Except where specific reference is made to other sources, the work presented in this thesis is the original work of the author. The work has not been submitted, in whole or in part, for any other degree.

Jen-der Day Tsai

Acknowledgments

Praise the Lord and make everything possible.

I would like to express my gratitute to my supervisors, Professor Lyn Thomas, whose advice and enthusiasm made this work possible.

I also would like to thank the support and encourgement from my wonderful husband, H.T. Tsai and two lovely children, Esther and Paul over the years.

Thanks are also due to the staff of the departments of Business Studies for providing a friendly working environment.

Finally I would like to thank the University of Edinburgh for the use of their resources.

Glossary

APDS – Antipodal Direct Sum

BER – Bit Error Rate

BSA – Bainary Switching Algorithm

BSC – Binary Symmetric Channel

CCITT – Voice Digitization in North American Telephone Systems

dB - Decibel

DMC – Discrete Memoryless Channel

ECD- Expected Channel Distortion

EIA – Eigenspace Index Assignment Algorithm

FBC – Folded Binary Code

FLSA – Full Linear Search Algorithm

GLA – General Lloyd Algorithm

IA – Index Assignment

LISA – Linearity Increasing Swap Algorithm

LBG – Lindo, Buzo and Gray

MDC – Minimum Distance Code

MSE – Mean-squared-error

NBC – Natural Binary Code

RC – Random Code

SEIA – Sequential Eigenspace Index Assignment Algorithm

SNR – Signal-to-noise Ratio

VECD - Expected Channel Distortion for Vector Quantizer

VEIA – Eigenspace Index Assignment Algorithm for Vector Quantizer

Abstract

A nonredundant quantization system is extensively used in a noisy communication system. However, the removal of redundancy can introduce a great deal of sensitivity to the noise engendered by transmission and this can cause performance to degrade. Index assignment is a way of combating this degradation of performance. A good index assignment algorithm will minimize the channel distortion caused by channel errors, but since the error rate affects the channel distortion, the performance of an optimal index assignment will vary as the error rate varies. Therefore, it is important to develop an index assignment algorithm that minimizes the channel distortion and is robust against variation in error rate as well.

The thesis will look at robust index assignment algorithms which minimize channel distortion for scalar and vector quantizations. An index assignment algorithm EIA for a scalar quantization model when the error rate is fixed is proposed. The idea behind the EIA is to use the Hadamard transformation and to rearrange the quantizer as close to the linear eigenspace as possible. Technically, the EIA depends on a regression calculation and sorting algorithm. Secondly, a modification of EIA, SEIA is presented which is independent of the initial index assignment. Thirdly, an algorithm VEIA which extends the EIA for scalar quantization to vector quantization is proposed. As well as existing criterion, a new criterion, the expected channel distortion (ECD) is defined on a beta-type prior density function for error rate which is appropriate for situations where the error rate varies over time. The above three algorithms are measured under this and the criterion is shown to perform well.

To illustrate the performances of signal-to-noise ratio, efficiency and robustness, the EIA is compared with the well-known algorithm BSA by using real data - a voice digitization in North American Telephone Systems CCITT for scalar

quantization. For vector quantization, VEIA is compared with the algorithm BSA by using first-order Gauss-Markov data.

The signal-to-noise ratio performance between EIA and BSA or VEIA and BSA are shown to be indistinguishable. The calculations involved in EIA, SEIA and VEIA are very simple since only a sorting algorithm is required. EIA takes only 0.9 seconds for a 256-point real world scalar quantizer while BSA takes hours. VEIA requires only 8 seconds for a two-dimension vector quantizer of size 256 when BSA again takes hours. Also, EIA and VEIA are robust when the error rate changes. They give the same optimal index assignment for error rate varying from 0.0001 to 0.1, while the BSA algorithm leads to different assignments.

Chapter 1

Introduction

1.1 Background

A standard communication system can be thought of a source which sends messages consisting of units of input, a transmitter which transmits the input message, a channel through which the message is sent, with a noise being added to the message in the channel, a receiver which receives the contaminated received message, and the received message. The message at the source may be anything in conventional alphabetic or numerical forms such as a notice and a report, or in continuous forms such as voice or image. Figure1.1 shows the standard communication system.

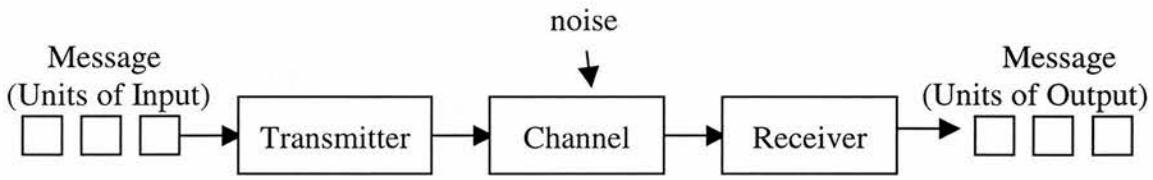

Figure 1.1: Process of a Communication System

This thesis deals with problems in such standard communication system. The transmitter encoding the message is called an encoder. The receiver decoding the message is called a decoder. The encoder is composed of a quantizer and an index assignment. A quantizer is defined as follows.

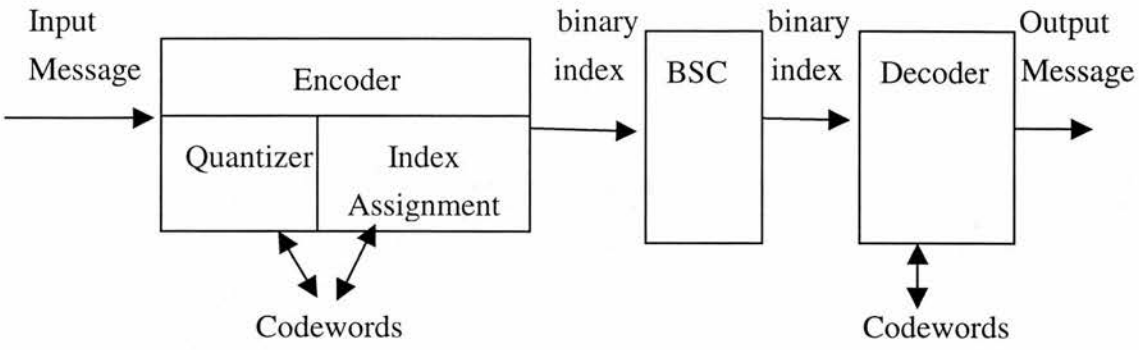

Figure 1.2: Process of Quantization System on a Binary Symmetric Channel (BSC)

Define an n-point quantizer to be a mapping from units of input into a finite set C containing n points which are called codewords. The set C is called a codebook. The index of a codeword is one of the integers from 0 to n-1. The binary representation of an index is a sequence of 1's or 0's with length $\log_2 n$ which is the binary representation of the corresponding number. The index assignment is a one-to-one mapping which assigns the indices to the codewords. In a quantization system, the input message is converted to its selected nearest codeword at the encoder. The binary representation of the assigned index of the selected codeword is then transmitted through a channel to a decoder. The corresponding received message can be decoded back to the input message by looking up the codewords for that index in the codebook. When a channel is noisy, there is a probability q that a transmitted 0 is interpreted as a 1 by the receiver, and vice versa. The q remains unchanged during the course of a transmission. Such a discrete memoryless channel (DMC) used for transmission of binary indices is called a binary symmetric channel (BSC) with bit error rate (BER) q. If the chance of 0 being represented as 1 is not the same as 1 being represented as 0, the channel is said to be non-symmetric. Figure 1.2 shows the quantization system of a binary symmetric channel.

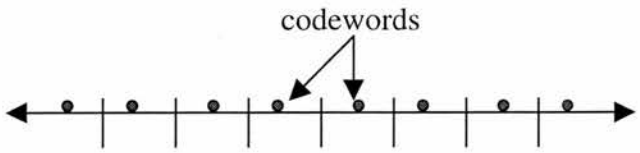

Figure 1.3 : Scalar Quantizer

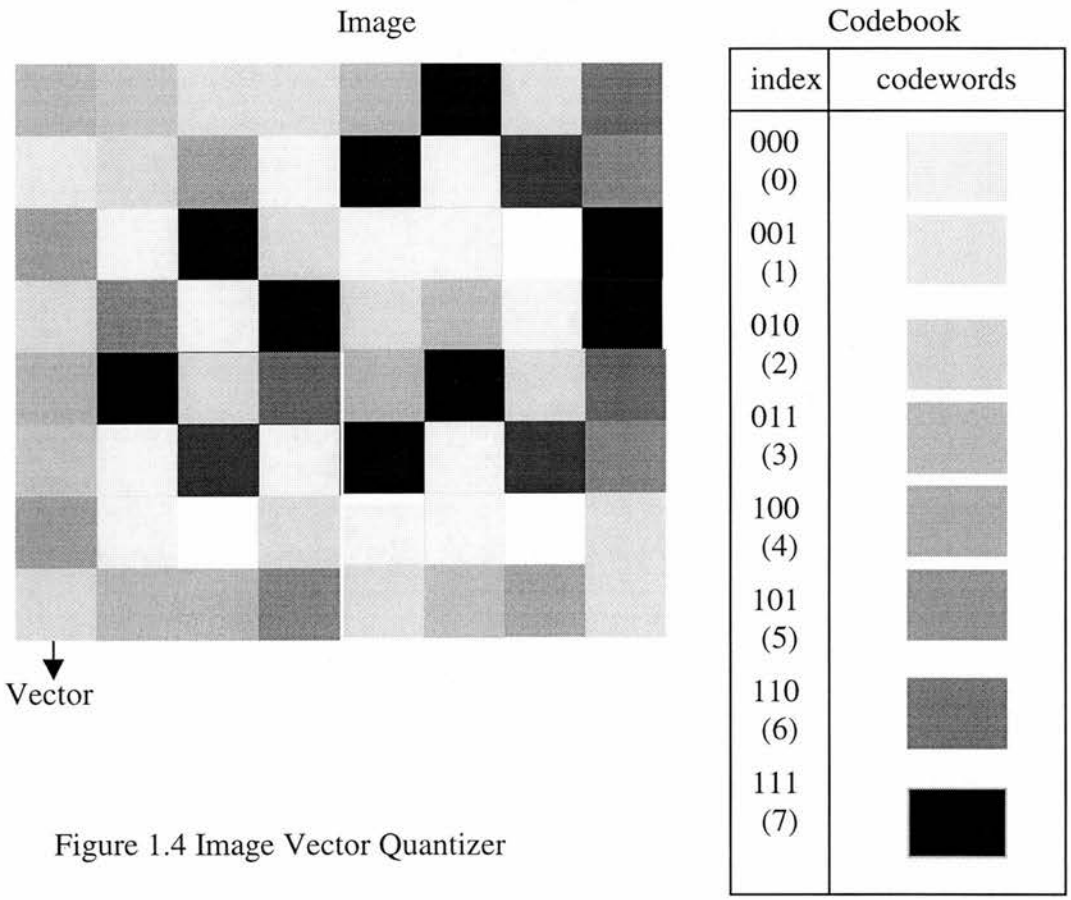

Figure 1.4 Image Vector Quantizer

Figure 1.3 shows an example of a one-dimensional (scalar) quantizer which maps each input value from a real line into one of eight codewords. To apply basic vector quantization to images, a natural way is to decompose a image into rectangular blocks of fixed size and then use these blocks as the vectors [Gersho and Gray, 1992]. Figure 1.4 shows an example of an image vector quantizer which maps each input image vector into one of eight codewords. For instance, in a colour picture we need the colour

consisting of red, yellow, blue, so the colour is a three-dimensional vector. On the picture image of Figure 1.4, the degree of brightness for each primary colour is measured on a scale of 0 to 7 for each square of the grid.

1.2 Purpose

On a noisy binary communication channel, it can happen that a binary index transmitted as a 0 is interpreted as a 1 by the receiver, and vice versa. The received binary index differs from the transmitted one. There is thus a distortion between the input message and the reconstructed output message and consequently degradation in performance occurs.

An important question in quantization theory is how to overcome performance degradation caused by noisy channels. One approach is to use redundant bits for error control coding. That is, the transmitted bits sequence consists of more than the necessary bits. The other approach is to improve the assigning of indices to codewords for a nonredundant quantized codebook. This is the index assignment (IA) problem discussed in this thesis.

In a nonredundant quantizer, the removal of redundancy can introduce a great deal of sensitivity to the transmission. An index assignment is a rearrangement of the order in which codewords appear in a quantizer codebook. This index assignment is the way that we can combat the degradation of performance caused by channel noise. For a given quantizer and a given channel, the object of an IA algorithm is to obtain an optimal index assignment which minimizes the channel distortion between input message and output message by rearranging the positions of codewords in a given codebook. Since the bit error rate (BER) will affect the channel distortion, the performance of an index assignment will vary as the BER varies. Therefore, it is important to develop an index assignment algorithm, which is robust against variation in the BER.

The thesis will look at three robust index assignment algorithms that minimize the effect of transmission errors on a discrete memoryless channel (DMC).

1.3 Organization of the Thesis

The rest of this thesis consists of three parts. Chapter 2 describes the notations of quantization system used in the thesis and reviews the literature on the index assignment. Chapters 3 and 4 propose three new index assignment algorithms for situations where the bit error rate remains unchanged. Chapter 5 proposes a new measure of performance as the bit error rate varies and describes how modifications of these new algorithms perform under this measure. Finally, Chapter 6 draws some conclusions. The details are organized as follows.

In Chapter 2, the different types of quantizers are reviewed. A quantizer design algorithm which generates optimal codebooks and minimizes the distortion caused by the quantization is reviewed. Based on the optimal codebook, index assignment algorithms which minimize the distortion caused by noisy channels are discussed. The most important of these is the Binary Switching Algorithm (BSA) which will be used to benchmark the algorithms developed in Chapters 3 and 4. Other index assignment algorithms are also reviewed. In Chapter 3, an efficient and robust eigenspace index assignment algorithm, EIA, is proposed for the scalar quantization model. The sequential eigenspace index assignment algorithm SEIA, a modification of EIA, is shown to be independent of the initial index assignment. The results of CPU time, channel distortion measure, and signal–to-noise performance measure are obtained for EIA, SEIA, and BSA applied to real data - a voice digitization in North American Telephone Systems CCITT. Also a comparison of EIA and the Linearity Increasing Swap Algorithm (LISA), a commonly used algorithm, on simulated data is undertaken. Chapter 4 extends the EIA for scalar quantization to VEIA for the vector quantization problem. Comparisons between VEIA and BSA are made using a first-order Gauss-Markov codebook and

VEIA turns out to be robust and efficient. Chapter 5 considers the channel mismatch problem where the bit error rate (BER) in the communication channel varies over time. A new criterion, the expected channel distortion (ECD) is defined using beta prior distributions. The above three algorithms are measured under this and other criterion are shown to perform well. Discussions and Conclusions are presented in Chapter 6.

Chapter 2

Literature Review

In the chapter we recall the mathematical background, outline the notations of quantization systems, and review the relevant literatures. Measures of performance, probability density functions of input to quantization systems, various choices of codebooks, a quantizer design algorithm, various choices of initial index assignments, and the existing optimal index assignment algorithms are defined.

2.1 Mathematical Notations

Definitions 2.1 and 2.2 [Lint, 1992] recall some aspect of combinatorial theory.

Definition 2.1 Let $\mathbf{I}_n$ be the identity matrix of order n. A square matrix $\mathbf{M}$ of order n with elements +1 and –1, such that $\mathbf{M}\mathbf{M}^T = n\mathbf{I}_n$, is called a Hadamard matrix.

For instance, a square Hadamard matrix M of order 2 is $\mathbf{M}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}$, and $\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$.

Definition 2.2 If $\mathbf{A}$ is a $m \times m$ matrix with entries a_{ij} and $\mathbf{B}$ is a $n \times n$ matrix then the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ is the $mn \times mn$ matrix given by

$$\mathbf{A}\otimes\mathbf{B}=\begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1m}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2m}\mathbf{B} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \cdots & a_{mm}\mathbf{B} \end{bmatrix}. \tag{2.1}$$

Note that the Kronecker product of Hadamard matrices is again a Hadamard matrix [Lint and Wilson, 1992]. Starting from $\mathbf{M}_2=\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ we can find the sequence $\mathbf{M}_{2^n}$, for instance, $\mathbf{M}_{2^2}=\mathbf{M}_2\otimes\mathbf{M}_2=\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$ etc.

2.2 Probability Density Functions of Input

Two probability density functions (pdf) used in the thesis for input $x\in R$ are listed below.

Uniform pdf with parameters a, $b\in R$, a<b:

$$p(x)=1/(b-a), \text{ for } x\in [a,b]\in R.$$

Gaussian pdf with parameters μ, $\sigma\in R$, $\sigma>0$, denoted by $N(\mu,\sigma^2)$:

$$p(x)=(2\pi\sigma^2)^{-1/2}\exp(-(x-\mu)^2/2\sigma^2),\ x\in R.$$

A k-dimensional random vector $\underline{\mathbf{x}}=[x_1, x_2, \ldots, x_k]\in R^k$ is independent and identically distributed (iid) Gaussian input, if it has independent components x_i with identical $N(\mu,\sigma^2)$. Thus the k-dimensional pdf is given by

$$p(\underline{\mathbf{x}})=\prod_{i=1}^{k} p(x_i)=(2\pi\sigma^2)^{-k/2}\exp(-\sum_{i=1}^{k}(x_i-\mu)^2/2\sigma^2).$$

2.3 Quantization Systems

A typical quantization system contains an encoder, a decoder, and a distortion

measure. The notations used in scalar quantization system and vector quantization system are described first in subsections 2.3.1 and 2.3.2, respectively, and performance measures are discussed in Section 2.4.

2.3.1 Scalar Quantization System

The scalar quantization system on a noisy channel and the notations used in this thesis are shown in Figure 2.1. Basically, the system consists of an encoder and a decoder. An encoder consists of a quantizer and an index assignment.

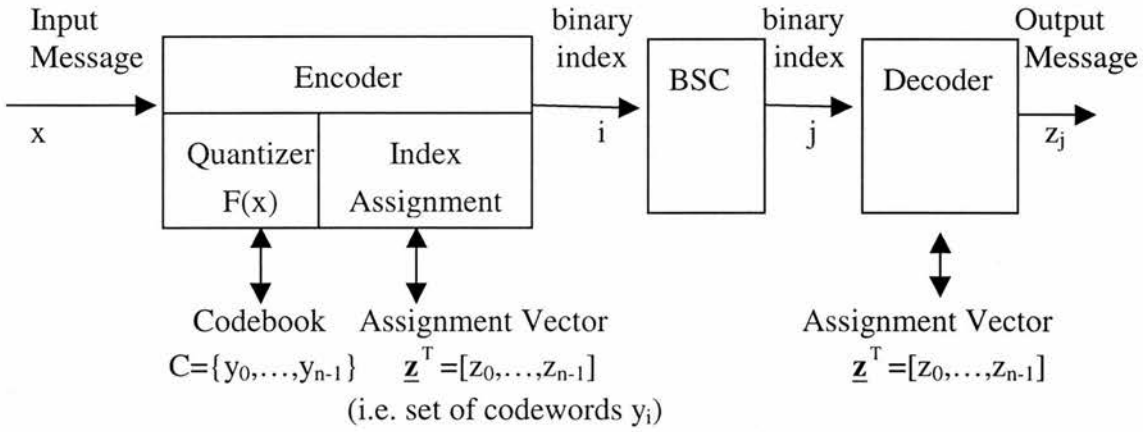

Figure 2.1: Process of Scalar Quantization on a Binary Symmetric Channel (BSC)

A scalar quantizer F of size n is a mapping from units of input $x \in R$ into a finite set C of real number containing n points $y_i \in R$. That is $F: R \rightarrow C$, where $C = \{y_0, y_1, ..., y_{n-1}\}$ is called a codebook and $y_i \in R$ for each $i \in I=\{0,1,\ldots,n-1\}$ is called a codeword. In a non-redundant binary system, the size n is a power of 2, say $n = 2^b$ and b is a positive integer. A quantizing partition $S = \{S_0, S_1, ..., S_{n-1}\}$ of R has n cells associated with n points such that $S_i = \{x \in R : F(x) = y_i\}$. Hence, the scalar quantizer F is completely described by the codebook C together with the partition S. Let p(x) denote

the probability density function of an input x, then $p_i = \int_{S_i} p(x)dx$ is the occupancy probability of the i^{th} cell. If every codeword satisfies $y_i = E[X \mid X \in S_i] = \int_{S_i} xp(x)dx / p_i$, for i=0,1,...n-1, then the centroid condition holds. We will show that the centroid condition is a necessary condition for an optimal codebook [Gersho and Gray, 1992]. The details are described in subsection 2.5.4.

In the thesis, we assume the input and the quantizer form a matched equiprobable source-quantizer pair in which all cells are equiprobable and all codewords are cell centroids, that is p_i=1/n and $y_i = E[X \mid X \in S_i]$ for all i. A quantizer of this type is also called a maximum-output entropy quantizer (defined in detail in 2.3.3). Without loss of generality, we also standardize the elements in a codebook by changing the definition of the origin point 0 in R such that $\sum_{i=0}^{n-1} y_i = 0$ without affecting performance.

The codewords in a codebook are mapped one-to-one onto the indices $i \in I = \{0,1,...,n-1\}$, each of which is a b-bit binary representation of the corresponding number i. This is called the index assignment problem. Given an input $x \in S_i$, an encoder converts it to the selected nearest codeword and delivers its b-bit binary index i to a decoder through a noise, memoryless, binary symmetric channel (BSC). However, due to the effect of channel noise during transmission, the received binary index j is not necessary the same as the transmitted index i and cause a significant performance degradation in decoded codewords. The received codeword becomes y_j for an input $x \in S_i$, and its distortion is measured by $d(x,y_j)$.

An index assignment can combat performance degradation caused by channel noise. Let $\underline{z}^T = [z_0, z_1, ..., z_{n-1}]$ be an index assignment vector, where each z_i is one of the codewords in C and $i \in I$ is its associated index. Thus, for each permutation of codewords in a codebook, there is a corresponding assignment vector $\underline{z}$. The choice of

permutation of codewords in a codebook is referred to as the index assignment of a quantizer. Thus, given an input signal $x \in S_i$, the resulting distortion is then measured by a given distortion measurement $d(x, z_j)$, where z_j is the codeword in a decoder corresponding to x.

Since rearranging the positions of codewords in a given codebook C results in different distortion between x and z_j, the object of an index assignment algorithm is to minimize the distortion $d(x, z_j)$ between the quantizer input x and the received codeword z_j.

2.3.2 Vector Quantization System

In a vector quantization system, the notation generalizes straightforwardly from scalar quantization. Define a vector quantizer $\mathbf{F}: R^k \rightarrow \mathbf{C}$, where $\mathbf{C} = \{\underline{\mathbf{y}}_0, \underline{\mathbf{y}}_1, ..., \underline{\mathbf{y}}_{n-1}\}$ is called a codebook and $\underline{\mathbf{y}}_i \in R^k$ for each $i \in \mathbf{I} = \{0,1,..., n-1\}$ is called a codeword. A quantizing partition $\mathbf{S} = \{\mathbf{S}_0, \mathbf{S}_1, ..., \mathbf{S}_{n-1}\}$ of R^k has n cells associated with n points such that $\mathbf{S}_i = \{\underline{\mathbf{x}} \in R^k : F(\underline{\mathbf{x}}) = \underline{\mathbf{y}}_i\}$, where $\underline{\mathbf{x}}^T = [x_1, x_2, \cdots, x_k]$ denotes an input vector. As in scalar quantizer, the vector quantizer **F** is completely described by the codebook **C** together with the partition **S**. Let $p(\underline{\mathbf{x}})$ denote the probability density function of $\underline{\mathbf{x}}$, then $p_i = \int_{S_i} p(\underline{\mathbf{x}}) d\underline{\mathbf{x}}$ is the occupancy probability of the i^{th} cell. As in the case of scalar quantization, if the centroid condition holds, then $\underline{\mathbf{y}}_i = E[\underline{\mathbf{X}} \mid \underline{\mathbf{X}} \in S_i] = \int_{S_i} \underline{\mathbf{x}} p(\underline{\mathbf{x}}) d\underline{\mathbf{x}} / p_i$. The zero-mean codebook assumption $\sum_{i=0}^{n-1} \underline{\mathbf{y}}_i = \underline{\mathbf{0}}$ is also assumed.

Let **Z** denote a $n \times k$ assignment matrix with distinct ordering of $\underline{\mathbf{y}}_i$**'s** in **C** as

$$\mathbf{Z} = [\underline{\mathbf{z}}_1, \underline{\mathbf{z}}_2, ..., \underline{\mathbf{z}}_k] = \left[\underline{\xi}_0^T, \underline{\xi}_1^T, ..., \underline{\xi}_{n-1}^T\right]^T, \qquad (2.2)$$

where $\underline{\xi}_i$ is the k-dimensional codeword (i.e. one of the $\underline{\mathbf{y}}_i$'s in **C**) that is assigned the b-bit binary expansion of i. As in scalar quantizer, given an input vector $\underline{\mathbf{x}} \in \mathbf{S}_i$, an encoder then converts it to the selected vector $\underline{\xi}_i$ and delivers its b-bit binary index i through a binary symmetric channel (BSC) to a decoder. Let $\underline{\xi}_j$ be the codeword that arises at the decoder when $\underline{\mathbf{x}}$ is an input vector. The resulting distortion is then measured by a distortion measure $d(\underline{\mathbf{x}}, \underline{\xi}_j)$. This measure of performance is further discussed in the Section 2.4.

2.3.3 Maximum-output Entropy Quantizer

One way of describing the structure of the output of a quantizer is to use an entropy function $\Phi(.)$. Messerschmitt [Messerschmitt, 1971] stated that the entropy at the output of a quantizer is equal to the average mutual information between unquantized and quantized random variables which is made precise in definition 2.3. Thus, for a fixed number of codewords, output entropy is a reasonable information-theoretic criterion of quantizer fidelity. Messerschmitt [Messerschmitt, 1971] also showed that, for a class of input distributions, the quantizers with maximum-output entropy and minimum quantization error are the same up to a multiplicative constant.

A quantizer is represented by a partition of n cells and a codebook with n codewords. Let Y denotes the random output from the quantizer. The concept of the entropy of a quantizer is obtained by measuring the average amount of information per codeword y_i of the quantizer output Y as follows.

Definition 2.3: The entropy of a quantizer output Y is defined by

$$\Phi(Y) = \sum_{i=1}^{n} p(y_i) \log \frac{1}{p(y_i)}, \tag{2.3}$$

where $p(y_i)$ is the probability of codeword y_i at i^{th} partition cell occurring from a random

input and n is the number of codewords.

As noted, the entropy function has the following properties [Hamming, 1980]:

(1) $0 \leq \Phi(Y) \leq \log_2 n$.

(2) $\Phi(Y)=0$, if only one codeword.

(3) $\Phi(Y)=\log_2 n$ is maximal, if all the codewords are equally likely. That is, $p(y_i)=\frac{1}{n}$, for all i.

Therefore, a maximum-output entropy quantizer is one which is equiprobable, i.e. in which all codewords are used with equal probability, say $p(y_i)= \frac{1}{n}$. So the assumption in the thesis that all the cells are equiprobable is equivalent to saying that the quantizer has maximum-output entropy.

2.4 Measures of Performance

2.4.1 Mean-squared Distortion

The distortion between an input message and its output message is defined as their Euclidean distance from each other. The Euclidean length and distance are recalled first as follows. For convenience, k-dimensional vectors are assumed.

Definition 2.4: The Euclidean length of a vector $\underline{\mathbf{x}} = (x_1, x_2, \cdots, x_k)$ in k-dimensional real Cartesian space R^k is defined by $\|\underline{\mathbf{x}}\| = (\underline{\mathbf{x}}^T \underline{\mathbf{x}})^{1/2} = \sqrt{x_1^2 + x_2^2 + \ldots + x_k^2}$.

Definition 2.5: The squared Euclidean distance between two k-dimensional vectors $\underline{\mathbf{x}} = (x_1, x_2, \cdots, x_k)$ and $\underline{\mathbf{y}} = (y_1, y_2, \cdots, y_k)$ is defined by

$$d(\underline{\mathbf{x}}, \underline{\mathbf{y}}) = \|\underline{\mathbf{x}} - \underline{\mathbf{y}}\|^2 = \sum_{i=1}^{k} (x_i - y_i)^2 . \qquad (2.4)$$

For statistical averaging of the distortion, the input probability density function as well as the specific quantizer characteristic are taken into account. One of the most popular measurements in use is the Mean-squared-error (MSE) as follows:

Definition 2.6: The mean-squared distortion D of assignment matrix **Z** is defined as follows. Suppose the input is a k-dimensional vector $\underline{\mathbf{x}} = (x_1, x_2, \cdots, x_k)$ and the corresponding codeword of the quantizer is $\mathbf{F}(\underline{\mathbf{x}}) = \underline{\xi}_i = (\xi_{i1}, \xi_{i2}, \cdots, \xi_{ik})$, and the corresponding received codeword at the decoder is $\underline{\xi}_j = (\xi_{j1}, \xi_{j2}, \cdots, \xi_{jk})$, then

$$D(\mathbf{Z}) = E[d(\underline{\mathbf{X}}, \underline{\xi}_j)] = E[\| \underline{\mathbf{X}} - \underline{\xi}_j \|^2] = \frac{1}{k} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} q_{ij} \int_{s_i} \left\| \underline{\mathbf{x}} - \underline{\xi}_j \right\|^2 p(\underline{\mathbf{x}}) d\underline{\mathbf{x}}, \qquad (2.5)$$

where $p(\underline{\mathbf{x}})$ is the probability density function of $\underline{\mathbf{x}}$; q_{ij} is the transition probability that the channel output is j given that its input is i. For scalar quantizations, a mean squared distortion is $D(\underline{\mathbf{z}}) = E[(x - z_j)^2]$, where x is the input symbol and z_j is the received codeword at the decoder corresponding to x.

Since the mean-squared-error (MSE) criterion is used, (2.5) can be decomposed into the sum of a quantization distortion (D_s) due to quantization effects only, a channel distortion (D_c) caused by misinterpretation of the received codewords and the mixed term (D_M). Let $p_i = \int_{s_i} p(\underline{\mathbf{x}}) d\underline{\mathbf{x}}$, then

$$\begin{aligned} D(\mathbf{Z}) &= \frac{1}{k} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} q_{ij} \int_{s_i} \left\| \underline{\mathbf{x}} - \underline{\xi}_j \right\|^2 p(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \qquad (2.6) \\ &= \frac{1}{k} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} q_{ij} \int_{s_i} \left\| \underline{\mathbf{x}} - \underline{\xi}_i + \underline{\xi}_i - \underline{\xi}_j \right\|^2 p(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \\ &= \frac{1}{k} \sum_{i=0}^{n-1} \int_{s_i} \left\| \underline{\mathbf{x}} - \underline{\xi}_i \right\|^2 p(\underline{\mathbf{x}}) d\underline{\mathbf{x}} + \frac{1}{k} \sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij} \left\| \underline{\xi}_i - \underline{\xi}_j \right\|^2 + \frac{2}{k} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} q_{ij} \int_{s_i} (\underline{\mathbf{x}} - \underline{\xi}_i)^T (\underline{\xi}_i - \underline{\xi}_j) p(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \\ &= D_s + D_c(\mathbf{Z}) + D_M \end{aligned}$$

or in the format of expectation, we have

$$\begin{aligned} D &= E[\| \underline{\mathbf{X}} - \underline{\xi}_j \|^2] = E[\| \underline{\mathbf{X}} - \underline{\xi}_i + \underline{\xi}_i - \underline{\xi}_j \|^2] \\ &= E[\| \underline{\mathbf{X}} - \underline{\xi}_i \|^2] + E[\| \underline{\xi}_i - \underline{\xi}_j \|^2] + 2E[(\underline{\mathbf{X}} - \underline{\xi}_i)^T (\underline{\xi}_i - \underline{\xi}_j)] \\ &= D_s + D_c + D_M \end{aligned} \quad . \qquad (2.7)$$

Since the expectation of a constant, c, is that constant, and for a random variable X, E[cX]=cE[X], $(\underline{\xi}_i - \underline{\xi}_j)$ can be moved out of the expectation so that

$$D_M = 2E[(\underline{\mathbf{X}} - \underline{\xi}_i)^T (\underline{\xi}_i - \underline{\xi}_j)] = \frac{2}{k} \sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij} (\underline{\xi}_i - \underline{\xi}_j)^T E[(\underline{\mathbf{X}} - \underline{\xi}_i) \mid \underline{\mathbf{X}} \in \mathbf{S}_i] . \qquad (2.8)$$

If the codeword $\underline{\xi}_i$ is the centroid of the partition cell $\mathbf{S}_i$ for all i, i.e. $\underline{\xi}_i = E[\underline{\mathbf{X}} \mid \underline{\mathbf{X}} \in \mathbf{S}_i]$, then

$$E[(\underline{\mathbf{X}} - \underline{\xi}_i) \mid \underline{\mathbf{X}} \in \mathbf{S}_i] = (\underline{\xi}_i - \underline{\xi}_i) = 0 \qquad (2.9)$$

Therefore, The mixed term (D_M) is zero if the centroid condition holds and

$$D(\mathbf{Z}) = D_s + D_c(\mathbf{Z}) . \qquad (2.10)$$

Totty and Clark [Totty and Clark, 1967] also showed the same result in the scalar quantization case. They also emphasized the noise in the transmission is completely arbitrary. No assumption regarding the independence of the quantization error and the channel noise is made nor required. Rydbeck and Sundberg [Rydbeck and Sundberg, 1976] also showed the same result as (2.10) for scalar quantization with arbitrary channels. For vector quantization, Farvardin [Farvardin, 1990] and Messerschmitt [Messerschmitt, 1979] also found the same result as (2.10) for discrete memoryless channels. Zeger and Gersho [Zeger and Gersho, 1990] summarized that the total distortion with MSE criterion under a noisy channel can be written as $D=D_s+D_c$ if the centroid condition is satisfied. Zeger and Manzella [Zeger and Manzella, 1994] proved that the mixed term D_M decays to zero as the size of codebook n approaches infinity even when the codewords are not positioned in the centroids for a binary symmetric

channel.

In the thesis, we construct optimal codebooks which minimize D_s and in which the centroid condition holds. Then the mean-squared distortion D is composed of D_s and D_c with D_M=0. However our focus is on index assignment algorithms to minimize channel distortion D_c for such optimal codebooks. The optimal codebook generated by the quantizer design algorithm is discussed in Section 2.5.4.

2.4.2 Signal-to-noise Ratio(SNR)

The measure of performance is a real number which indicates the overall quality degradation or distortion of a quantizer system. It should reflect the subjective assessment of the message quality that is a human perception of the aural or visual signals. The signal-to-noise ratio (SNR) is one of the most common measure, which tries to do this.

The SNR is an assessment of performance, which is defined by normalizing the input variance Var($\underline{\mathbf{X}}$) by the noise distortion D and taking a scaled logarithm:

$$\mathrm{SNR} = 10\log_{10}\frac{\mathrm{Var}(\underline{\mathbf{X}})}{\mathrm{D}} = 10\log_{10}\frac{\mathrm{E}(\underline{\mathbf{X}}-\mathrm{E}(\underline{\mathbf{X}}))^2}{\mathrm{D}}, \tag{2.11}$$

which is measured in units of decibels (dB). For a zero-mean assumption of $\mathrm{E}(\underline{\mathbf{X}}) = 0$, the input variance Var($\underline{\mathbf{X}}$) is the same as the $\mathrm{E}(\underline{\mathbf{X}}^2)$. When the total distortion is much less than the input variance, i.e. D<<Var($\underline{\mathbf{X}}$), the term high resolution is used, where resolution corresponds to the degree of discernible detail. Certainly, the real performance measurement for a communication system is the subjective quality of the received signal. In image processing, it is the quality of received picture that matters and high dots per inch means high resolution. For a voice system, it is the quality of the received speech that matters. Therefore, it is necessary to compare the theoretical results with subjective

measurements [Yan and Donaldson, 1972]. In general, the borderline between high and low resolutions is taken to be a SNR value of 10 dB [Gersho and Gray 1992].

2.5 Choices of Quantizers

In this section, we introduce some quantizers that will serve as examples later for verifying the performances of index assignment algorithms. The uniform quantizer and the antipodal direct sum codebooks are used since their optimal index assignments are known and thus can serve as benchmark examples. Also, a real world symmetric piecewise uniform scalar quantizer of the voice digitization in North American Telephone Systems (called CCITT) and a first-order Gauss-Markov quantizer are included.

2.5.1 Uniform Quantizer

A uniform quantizer is a quantizer which satisfies two conditions: (1) the partitions are equally spaced and (2) every codeword is the centroid of its associated partition if that cell is bounded. In the case of scalar quantization, let $C=\{y_0, y_1, ..., y_{n-1}\}$, where $y_i \in R$, x_i and x_{i+1} are the endpoints of the i^{th} partition cell. The first condition implies that with step size Δ, $y_i-y_{i-1}=\Delta$ for i=1,2,3,...,n-1 and the second condition implies that $y_i=(x_i+x_{i+1})/2$ for i=1,2,3,...,n-2. For unbounded inputs, the quantizer has overload cells $(-\infty, x_1]$ and $[x_{n-1}, +\infty)$ with $y_0=x_1-\Delta/2$ and $y_{n-1}=x_{n-1}+\Delta/2$.

2.5.2 Antipodal Direct Sum Codebook

An antipodal direct sum codebook (APDS) with size 2^b is the direct sum of the b antipodal codebooks $\{-c_0, c_0\}$, $\{-c_1, c_1\}, \cdots, \{-c_{b-1}, c_{b-1}\}$ [Barnes and Frost, 1993]. That is, each codeword is a signed combination of the c_i's in the form of $\{\pm c_0 \pm c_1 \pm \cdots \pm c_{b-1}\}$, where c_i's are distinct real numbers so that no c_i can be obtained

by adding or subtracting others [McLaughlin et al., 1995]. Note that each codeword has a unique decomposition as a signed combination of c_i's so all 2^b codewords are distinct.

For instance, a uniform scalar codebook is a special case of APDS codebook. For a uniform scalar codebook with size $n=2^3=8$ and step size Δ, the codebook is

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -\frac{\Delta}{2} \\ -(\frac{\Delta}{2})2 \\ -(\frac{\Delta}{2})2^2 \end{bmatrix} = \begin{bmatrix} -3.5\Delta \\ -2.5\Delta \\ -1.5\Delta \\ -0.5\Delta \\ 0.5\Delta \\ 1.5\Delta \\ 2.5\Delta \\ 3.5\Delta \end{bmatrix}.$$

Therefore, in general, $C=\left\{\frac{-n+1}{2}\Delta,\frac{-n+3}{2}\Delta,\cdots,\frac{n-1}{2}\Delta\right\}$ with $\{c_0,c_1,...,c_{b-1}\}$ $=\{-(\Delta/2),-(\Delta/2)2,...,-(\Delta/2)2^{b-1}\}$.

2.5.3 Piecewise Uniform Quantizer and CCITT

A piecewise uniform quantizer is a quantizer whose range consists of several segments. Each segment contains several partition cells with codewords corresponding to a uniform quantizer. Thus if (a_i,b_i) is the i^{th} segment with $\nu>2$ steps, then a_i and b_i are the endpoints of the i^{th} segment, in which each partition cell has size $\Delta_i=(b_i-a_i)/\nu$, and each codeword is the midpoint of the associated partition cell. Note that each segment may have a different number of steps ν and a different value of step size Δ_i.

In the current North American standard for digital telephony, the Voice Digitization Systems (CCITT G.711) [Gersho and Gray, 1992] is a symmetric piecewise uniform quantizer with 8 bits. The CCITT is symmetric around zero with 8 positive and 8 negative segments, increasing in length by a factor of 2 for each successive segment in order of increasing amplitude, respectively, and with $\nu=16$ steps on each segment. It has codeword values given by the following formula

$$y(P,S,B) = (1-2P)[(2B+33)(2^S)-33], \qquad (2.12)$$

where the parameters can take on the values of P=0,1, S=0,1,2,...,7, and B=0,1,2,...15.

The negative codeword values of the CCITT codebook are listed in Table 2.1. The possible range of codeword values y_i is from -8031 to 8031 and its variance is 6423572. There are two zero's codeword values for both positive and negative segment, so it has 255 distinct codewords in 255 cells. The quantization distortion $D_s = (\sum_{i=1}^{m} \frac{1}{m} \Delta_i^2)/12 = 910$, where m=16 denote the number of segments.

Table 2.1: The negative codeword values of CCITT Codebook.

Segment i	**Values**							
$\Delta_i=(b_i-a_i)/16$	**y_i**							
1	**-8031**	**-7775**	**-7519**	**-7263**	**-7007**	**-6751**	**-6495**	**-6239**
256= (8159-4063)/16	**-5983**	**-5727**	**-5471**	**-5215**	**-4959**	**-4703**	**-4447**	**-4191**
2	-3999	-3871	-3743	-3615	-3487	-3359	-3231	-3103
128= (4063-2015)/16	-2975	-2847	-2719	-2591	-2463	-2335	-2207	-2079
3	**-1983**	**-1919**	**-1855**	**-1791**	**-1727**	**-1663**	**-1599**	**-1535**
64= (2015-991)/16	**-1471**	**-1407**	**-1343**	**-1279**	**-1215**	**-1151**	**-1087**	**-1023**
4	-975	-943	-911	-879	-847	-815	-783	-751
32= (991-479)/16	-719	-687	-655	-623	-591	-559	-527	-495
5	**-471**	**-455**	**-439**	**-423**	**-407**	**-391**	**-375**	**-359**
16= (479-223)/16	**-343**	**-327**	**-311**	**-295**	**-279**	**-263**	**-247**	**-231**
6	-219	-211	-203	-195	-187	-179	-171	-163
8=(223-95)/16	-155	-147	-139	-131	-123	-115	-107	-99
7	**-93**	**-89**	**-85**	**-81**	**-77**	**-73**	**-69**	**-65**
4=(95-31)/16	**-61**	**-57**	**-53**	**-49**	**-45**	**-41**	**-37**	**-33**
8	-30	-28	-26	-24	-22	-20	-18	-16
2=(31-(-1))/16	-14	-12	-10	-8	-6	-4	-2	0

2.5.4 Quantizer Design and GLA for Vector Quantization

In this subsection, the quantizer design algorithm and the well-known generalized Lloyd algorithm (GLA) are stated in details. For a given set of data, the quantizer design algorithm seeks an optimal quantizer which will minimize the quantization distortion D_s. In a vector quantizer **F** of size n, a k-dimensional input symbol $\underline{\mathbf{x}} = (x_1, x_2, \cdots, x_k)$ has the corresponding codeword $\mathbf{F}(\underline{\mathbf{x}})$, then the quantization distortion is given by

$$D_s = E[d(\underline{\mathbf{X}}, F(\underline{\mathbf{X}}))], \tag{2.13}$$

where $p(\underline{\mathbf{x}})$ is the probability density function of $\underline{\mathbf{x}}$. An optimal quantizer design is defined as follows.

Definition 2.7: For given input message and a given size of quantizer n with k dimension, a vector quantizer **F*** is said to be optimum if $D_s(\mathbf{F}^*) \leq D_s(\mathbf{F})$, for all other quantizers **F**. (When k=1, it is the case of a scalar quantizer.)

Since a quantizer consists of a partition and a codebook, the principal goal of quantizer design is to select the codewords and the partition cells so as to minimize the quantization distortion D_s. Gersho and Gray [Gersho and Gray, 1992] summarized two necessary conditions for an optimal quantizer as follows:

(1) Find the optimal partition for a given codebook (nearest neighbor condition)

(2) Find the optimal codebook for a given partition (centroid condition).

These two conditions and their proofs of optimality are summarized as follows.

Definition 2.8 (nearest neighbor condition): For a given codebook in vector quantizations, the codeword $\underline{\xi}_i$ is the nearest neighbor of an input point $\underline{\mathbf{x}}$ if $\underline{\mathbf{x}}$ is closer to codeword $\underline{\xi}_i$ than to any other codewords. For scalar quantization, $\underline{\mathbf{x}}$ is replaced by x and $\underline{\xi}_i$ is replaced by y_i.

Theorem 2.9: [Gersho and Gray, 1992] Given the codebook $\mathbf{C} = \{\underline{\xi}_0, \underline{\xi}_1, ..., \underline{\xi}_{n-1}\}$ for a vector quantizer **F**, the optimal partition $\mathbf{S} = \{\mathbf{S}_i; i = 0,..., n-1\}$ satisfies the nearest neighbor condition

$$\mathbf{S}_i = \{\underline{\mathbf{x}} : d(\underline{\mathbf{x}}, \underline{\xi}_i) \le d(\underline{\mathbf{x}}, \underline{\xi}_j); \forall j \ne i\}. \tag{2.14}$$

Proof of Theorem 2.9:

Given the given codebook $\mathbf{C} = \{\underline{\xi}_0, \underline{\xi}_1, ..., \underline{\xi}_{n-1}\}$, the nearest neighbor condition in (2.14) is equivalent to $\mathbf{F}(\underline{\mathbf{x}}) = \underline{\xi}_i$, only if $\underline{\mathbf{x}} \in \mathbf{S}_i = \{\underline{\mathbf{x}} : d(\underline{\mathbf{x}}, \underline{\xi}_i) \le d(\underline{\mathbf{x}}, \underline{\xi}_j), \forall j \ne i\}$, or $d(\underline{\mathbf{x}}, \mathbf{F}(\underline{\mathbf{x}})) = \min_{\underline{\xi} \in \mathbf{C}} (d(\underline{\mathbf{x}}, \underline{\xi}))$. Thus, the partition **S** has a minimum distortion as

$$\begin{aligned} D_s &= E[d(\underline{\mathbf{X}}, \mathbf{F}(\underline{\mathbf{X}}))] = \sum_{i=0}^{n-1} \int_{\mathbf{S}_i} d(\underline{\mathbf{x}}, \underline{\xi}_i) p(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \\ &\ge \sum_{i=0}^{n-1} \int_{\mathbf{S}_i} \min_{\underline{\xi} \in \mathbf{C}} (d(\underline{\mathbf{x}}, \underline{\xi})) p(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \end{aligned}$$

This lower bound is attained when the nearest neighbor condition is satisfied. □

Definition 2.10 (centroid condition): For a given set $\Psi \in R^k$ with nonzero probability, the centroid of Ψ, denoted by cent(Ψ), is defined as the vector $\underline{\mathbf{y}}^*$ which minimizes the distortion between a point $\underline{\mathbf{x}} \in \Psi$ and $\underline{\mathbf{y}}^*$ averaged over the probability distribution of $\underline{\mathbf{x}}$ given that $\underline{\mathbf{x}}$ lies in Ψ. Thus, $\underline{\mathbf{y}}^*$=cent(Ψ) if $E[d(\underline{\mathbf{X}}, \underline{\mathbf{y}}^*) \mid \underline{\mathbf{X}} \in \Psi] \le E[d(\underline{\mathbf{X}}, \underline{\mathbf{y}}) \mid \underline{\mathbf{X}} \in \Psi]$.

Theorem 2.11: [Gersho and Gray, 1992] Given a partition $\mathbf{S} = \{\mathbf{S}_i; i = 0,..., n-1\}$ for a vector quantizer, the codebook which minimizes the distortion D_s is given by $\mathbf{C} = \{\text{cent}(\mathbf{S}_i); i = 0,..., n-1\}$.

Proof of Theorem 2.11:

Let $\mathbf{F}(\underline{\mathbf{x}}) = \mathbf{C}$ denote the optimal quantizer such that $\mathbf{F}(\underline{\mathbf{x}}) = \text{cent}(\mathbf{S}_i)$, where $\underline{\mathbf{x}} \in \mathbf{S}_i$. Considering any other quantizer $\mathbf{F}'(\underline{\mathbf{x}}) = \mathbf{C}' = \{\underline{\xi}_i'; i = 0,..., n-1\}$ such that

$\mathbf{F}'(\underline{\mathbf{x}}) = \underline{\xi}_i'$, where $\underline{\mathbf{x}} \in \mathbf{S}_i$. By Definition 2.10,

$$\begin{aligned} D_s &= E[d(\underline{\mathbf{X}}, \mathbf{F}'(\mathbf{X}))] = \sum_{i=0}^{n-1} p_i E[d(\underline{\mathbf{X}}, \underline{\xi}_i') \mid \underline{\mathbf{X}} \in \mathbf{S}_i] \\ &\geq \sum_{i=0}^{n-1} p_i E[d(\underline{\mathbf{X}}, \text{cent}(\mathbf{S}_i)) \mid \underline{\mathbf{X}} \in \mathbf{S}_i] = E[d(\underline{\mathbf{X}}, \mathbf{F}(\mathbf{X}))]. \end{aligned} \tag{2.15}$$

Therefore, **C** is an optimal codebook given a partition **S**. □

For the squared error measure the centroid reduces to the mean of $\underline{\mathbf{x}}$ as follows.

Corollary 2.12: [Gersho and Gray, 1992] The mean-squared error distortion measure, D_s is minimized for a quantizer when the codeword for the i^{th} cell $\mathbf{S}_i$ is simply the mean of $\underline{\mathbf{x}}$ given that $\underline{\mathbf{x}}$ in $\mathbf{S}_i$,

$$\text{cent}(\mathbf{S}_i) = E[\underline{\mathbf{X}} \mid \underline{\mathbf{X}} \in \mathbf{S}_i] = \underline{\xi}_i. \tag{2.16}$$

Proof of Corollary 2.12:

Let $\mathbf{F}(\underline{\mathbf{x}}) = \mathbf{C} = \{\underline{\xi}_i; i = 0,...,n-1\}$ denote the quantizer such that $\mathbf{F}(\underline{\mathbf{x}}) = \underline{\xi}_i$, where $\underline{\mathbf{x}} \in \mathbf{S}_i$. Consider any other quantizer $\mathbf{F}'(\underline{\mathbf{x}}) = \mathbf{C}' = \{\underline{\xi}_i'; i = 0,...,n-1\}$ such that $\mathbf{F}'(\underline{\mathbf{x}}) = \underline{\xi}_i'$, where $\underline{\mathbf{x}} \in \mathbf{S}_i$. By (2.16), $E[\underline{\mathbf{X}} - \underline{\xi}_i \mid \underline{\mathbf{X}} \in S_i] = \underline{0}$. Then, we have

$$E[d(\underline{\mathbf{X}}, \mathbf{F}'(\underline{\mathbf{X}})] = \sum_{i=0}^{n-1} p_i E[\| \underline{\mathbf{X}} - \underline{\xi}_i' \|^2 \mid \underline{\mathbf{X}} \in \mathbf{S}_i] = \sum_{i=0}^{n-1} p_i E[\| \underline{\mathbf{X}} - \underline{\xi}_i + \underline{\xi}_i - \underline{\xi}_i' \|^2 \mid \underline{\mathbf{X}} \in \mathbf{S}_i]$$

$$= E[\| \underline{\mathbf{X}} - \mathbf{F}(\underline{\mathbf{X}}) \|^2] + \sum_{i=0}^{n-1} p_i E[\| \underline{\xi}_i - \underline{\xi}_i' \|^2] + 2\sum_{i=0}^{n-1} p_i (\underline{\xi}_i - \underline{\xi}_i')^T E[(\underline{\mathbf{X}} - \underline{\xi}_i) \mid \underline{\mathbf{X}} \in \mathbf{S}_i] \quad ,$$

$$\geq E[\| \underline{\mathbf{X}} - \mathbf{F}(\underline{\mathbf{X}}) \|^2]$$

since the mixed term is equal to zero and the second term is positive. Thus, $E[\underline{\mathbf{X}} \mid \underline{\mathbf{X}} \in S_i] = \underline{\xi}_i$ is the centroid of the i^{th} cell $\mathbf{S}_i$. □

Lloyd [Lloyd, 1957] proposed two methods for scalar quantizer design. The one which involves a non-variational approach is known as Lloyd's Method I, and the other one with a variational approach is called as Lloyd's Method II. The technique used in Lloyd's Method II was also independently developed by Max [Max, 1960] and so is

known as the Lloyd-Max quantizer. Max used it when he calculated the optimum non-uniform and the optimum uniform (equally spaced) quantizers for a noiseless channel. The General Lloyd Algorithm (GLA) proposed by Lindo, Buzo and Gray [Lindo, et al., 1980] is a direct generalization of Lloyd's Method I, which provides an efficient algorithm for designing good vector quantizers that overcomes the problems of the variational approach. The GLA algorithm is also known as the LBG algorithm after Lindo, Buzo and Gray.

The GLA is used in this thesis because it can be used as an add-on to any other design algorithm. That is, the GLA can improve (or at worst leave unchanged) the performance of any given initial codebook. So any alternative vector quantizer design algorithm can always be regarded as a way to generate an initial codebook. [Gersho and Gray, 1992].

The GLA improves the codebook by iterating between partitions that satisfy the nearest neighbor condition and codewords that satisfy the centroid conditions alternatively. Define the training set to be a set of observations of the input to be quantized. It is expected the training set has enough data to estimate the true probability density function of the input effectively. The GLA initially randomly chooses n codewords from the training set. The Lloyd iteration is performed by assigning each vector in the training set to its nearest neighbor codeword using exhaustive search method. Thus, the vectors assigned to the same codeword form a nearest neighbor cell. Then, a calculation is performed which computes the centroids of these cells as the new codewords [Gersho and Gray, 1992]. The algorithm proceeds by performing this Lloyd iteration repeatedly until the overall distortion error changes by a small enough fraction.

Since for a given set of codewords $\mathbf{C}$, the optimal partition cells $\mathbf{S}_i$ satisfy the nearest neighbor condition (i.e. any $\underline{\mathbf{x}}$ in the training set has $\underline{\xi}_i$ as its unique nearest neighbor among the set of codewords. So, if $d(\underline{\mathbf{x}},\underline{\xi}_i) \leq d(\underline{\mathbf{x}},\underline{\xi}_j)$, for all j, then $\underline{\mathbf{x}} \in \mathbf{S}_i$). If

the nearest neighbor is not unique, the convention chooses the codeword with the smallest index among all the nearest neighbor codewords. If there is no training vector in a partition cell, the cell with the highest number of training vectors is assigned to a second code vector by splitting its centroid into two codewords and the empty cell is deleted. The GLA is summarized as follows.

Generalized Lloyd Algorithm (GLA)

Input: A training set $T=\{\underline{\mathbf{x}}_1,\underline{\mathbf{x}}_2,\cdots,\underline{\mathbf{x}}_m\}$.

Output: final locally optimal codebook $\mathbf{C}^{(j)}$.

Step 0: Initialization:

(1) Choose codewords from the training set randomly as an initial codebook $\mathbf{C}^{(0)}$ with size n.

(2) Set the distortion threshold to be $\varphi>0$. Set j=1.

Step 1: Given a codebook $\mathbf{C}^{(j-1)}=\{\underline{\xi}_i;i=0,...,n-1\}$, determine the optimal partition $\mathbf{S}^{(j)}=\{\mathbf{S}_i;i=0,1,...,n-1\}$ based on the nearest neighbour condition, where $\mathbf{S}_i=\{\underline{\mathbf{x}}\in\mathbf{T}:d(\underline{\mathbf{x}},\underline{\xi}_i)<d(\underline{\mathbf{x}},\underline{\xi}_{i'});\forall i'\neq i\}$, for i=0,1,…,n-1.

Step 2: Compute the centroid of $\mathbf{S}_i$ found in Step 1 to obtain the new improved codebook $\mathbf{C}^{(j)}$, namely, set $\mathbf{C}^{(j)}=\{cent(\mathbf{S}_i);\ i=0,...,n-1\}$, where $cent(\mathbf{S}_i)=E(\underline{\mathbf{X}}\mid\underline{\mathbf{X}}\in\mathbf{S}_i)=\underline{\xi}_i$.

Step 3: Test the stopping rule:

(1) Compute the average distortion $D^{(j)}=\frac{1}{m}\sum_{k=1}^{m}d(\underline{\mathbf{x}}_k,\mathbf{F}(\underline{\mathbf{x}}_k))$, where $\mathbf{F}(\underline{\mathbf{x}}_k)\in\mathbf{C}^{(j)}$ is the nearest neighbour of $\underline{\mathbf{x}}_k$.

(2) If $\left|\frac{D^{(j-1)}-D^{(j)}}{D^{(j-1)}}\right|\leq\varphi$, then output the final $\mathbf{C}^{(j)}$ and stop;

otherwise set j=j+1 and Goto Step 1.

Note that each iteration in the GLA always reduces (or at least never increases) quantization distortion.

2.5.5 Gauss-Markov Quantizer

In later chapters, some first-order Gauss-Markov quantizers will be generated from the generalized Lloyd algorithm (GLA) using a training set. The training set is a set of observations of the input to be quantized. The generating procedure for obtaining the first-order Gauss-Markov quantizer is summarized as follows [Zeger and Gersho, 1990].

Step 1: Generate a training set of size 7500 with k dimensions:

(1) Generate an initial random input x_1 from N(0,1).

(2) Generate the first-order Gauss-Markov inputs $\{x_j\}$ of the form $x_{j+1}= \rho\, x_j+w_j$, for j=1,...,7500k-1, where $\{w_j\}$ are identical independent distribution (iid) Gaussian input and are generated from $N(0, \sigma_w^2=1)$. The correlation coefficient ρ is between –1 and 1.

(3) Generate k-dimensional input vectors such as $\underline{\mathbf{x}}_1=[x_1,\ldots,x_k]$, $\underline{\mathbf{x}}_2=[x_{k+1},\ldots,x_{2k}]$, ... , $\underline{\mathbf{x}}_{7500}=[x_{7500-k+1},\ldots,x_{7500}]$. This first-order Gauss-Markov training set has the correlation matrix as

$$\begin{bmatrix} 1 & \rho & \rho^2 & . & \rho^{k-1} \\ \rho & 1 & \rho & . & \rho^{k-2} \\ \rho^2 & \rho & . & . & . \\ . & . & . & 1 & \rho \\ \rho^{k-1} & \rho^{k-2} & . & \rho & 1 \end{bmatrix}.$$

Step 2: Perform the GLA by minimizing the D_s to obtain a locally optimal codebook.

2.6 Choices of Initial Index Assignments

The performances of optimization algorithms often depend upon the choice of a

good initial index assignment. Some well-known codes are introduced in this section, and they will be served as initial index assignments in later chapters. Firstly, the Random Code (RC) is a popular code in which indices are randomly assigned to the codewords. Secondly, Rydbeck and Sundberg [Rydbeck and Sundberg, 1976] looked at the importance of good index assignments by comparing the performances of the Natural Binary Code, the Folded Binary Code, and the Minimum Distance Code in scalar quantizations. These three popular codes are defined as follows.

(1) Natural Binary Code (NBC): Assume codewords are in ascending order so that $y_0 < y_1 < ... < y_{n-1}$. They are then assigned the respective indices $I_{NBC}=\{0,1,...,n-1\}$. This is called the Natural Binary Code. The corresponding assignment vector is $\underline{\mathbf{z}}^T = [z_0, z_1, ..., z_{n-1}] = [y_0, y_1, ..., y_{n-1}]$. Table 2.2 shows the 4-bit NBC and its corresponding assignment vector.

Crimmins et al. [Crimmins et al., 1969] showed that the NBC is globally optimal for a uniform input and a uniform scalar quantizer. McLaughlin et al. [McLaughlin et al., 1995] provided a simpler proof of the results of Crimmins et al. We also give a proof in Chapter 3.

Therefore, we will use uniform scalar quantizers as an example to verify the performances of the proposed index assignment algorithms by comparing them with NBC what is known to be optimal.

(2) Folded Binary Code (FBC): Assume the codewords are in ascending order such as $y_0 < y_1 < ... < y_{n-1}$. If the first half of the indices are in reversed order so that $I_{FBC}=\{\frac{n}{2}-1,...,0,\frac{n}{2},...,n-1\}$ then it is called Folded Binary Code. The corresponding assignment vector is $\underline{\mathbf{z}}^T = [z_0, z_1, ..., z_{n-1}] = [y_{(n/2)-1}, ..., y_0, y_{(n/2)}, ..., y_{n-1}]$. Table 2.2 shows the 4-bit FBC and its corresponding assignment vector.

Table2.2: The 4-bit NBC, FBC, MDC and Their Corresponding Assignment Vectors.

C	NBC			FBC			MDC		
y_i	z_i	decimal	binary	z_i	decimal	binary	z_i	decimal	binary
y_0	y_0	0	0000	y_7	7	0111	y_7	7	0111
y_1	y_1	1	0001	y_6	6	0110	y_6	6	0110
y_2	y_2	2	0010	y_5	5	0101	y_5	5	0101
y_3	y_3	3	0011	y_4	4	0100	y_3	3	0011
y_4	y_4	4	0100	y_3	3	0011	y_4	4	0100
y_5	y_5	5	0101	y_2	2	0010	y_2	2	0010
y_6	y_6	6	0110	y_1	1	0001	y_1	1	0001
y_7	y_7	7	0111	y_0	0	0000	y_0	0	0000
y_8	y_8	8	1000	y_8	8	1000	y_8	8	1000
y_9	y_9	9	1001	y_9	9	1001	y_9	9	1001
y_{10}	y_{10}	10	1010	y_{10}	10	1010	y_{10}	10	1010
y_{11}	y_{11}	11	1011	y_{11}	11	1011	y_{12}	12	1100
y_{12}	y_{12}	12	1100	y_{12}	12	1100	y_{11}	11	1011
y_{13}	y_{13}	13	1101	y_{13}	13	1101	y_{13}	13	1101
y_{14}	y_{14}	14	1110	y_{14}	14	1110	y_{14}	14	1110
y_{15}	y_{15}	15	1111	y_{15}	15	1111	y_{15}	15	1111

(3) Minimum Distance Code (MDC): The first half of the numbers start with a zero in the index and the second half start with a one in the index. The code is symmetric around the median. The median numbers have zeros in all places except the first. The number outside these have a 1 in the last place, and zeros in all other places except possibly the first place. Then come numbers with 1 in the last but one place and zeros elsewhere. Keep moving the 1 forward until it is in the second position with a 0 or a 1 in the first position and 0's elsewhere. Outside these are numbers with two 1's in the last b-1 places. Start with the two 1's as close to the end of the code as possible and moving the 1's lexicographically forward as the numbers move away from the median. Repeat the same procedure until b-1 1's are filled. Table 2.3 shows the MDC of 2-bit, 3-bit, 4-bit, and 5-bit. Rydbeck and Sundberg [Rydbeck and Sundberg, 1976] also presented MDC, NBC, and FBC for n=64.

Table 2.3: Minimum Distance Codes for two bits, three bits, four bits, and five bits.

two-bit	three-bit	four-bit	five-bit
01	011	0111	01111
Zero 00	010	0110	01110
10	001	0101	01101
11	Zero 000	0011	01011
	100	0100	00111
	101	0010	01100
	110	0001	01010
	111	Zero 0000	00110
		1000	01001
		1001	00101
		1010	00011
		1100	01000
		1011	00100
		1101	00010
		1110	00001
		1111	Zero 00000
			10000
			10001
			⋮

2.7 Literature Review on Index Assignments

In this section we review the literature of the index assignment problem for scalar and vector quantizations. Crimmins et al. [Crimmins et al., 1969] showed that the NBC is globally optimal for a uniform input and a uniform scalar quantizer if there is a binary symmetric noisy channel. Rydbeck and Sundberg [Rydbeck and Sundberg, 1976] showed that different index assignments such as NBC, FBC, and MDC do affect the distortion of quantization if there are channel errors. Thereafter, there were many articles devoted to finding optimal index assignment algorithms. Suppose one has devised an optimal quantizer **F** which minimizes D_s. An index assignment algorithm is designed to minimize D_c by rearranging the positions of the codewords.

Definition 2.13: For a fixed vector quantizer **F** and a given bit error rate, an assignment matrix $\mathbf{Z}^*$ is said to be optimum if $D_c(\mathbf{Z}^*) \leq D_c(\mathbf{Z})$, for all **Z** defined in (2.2). Also, for a fixed scalar quantizer F and a given bit error rate, an assignment vector $\underline{\mathbf{z}}^*$ is said to be

optimum if $D_c(\underline{z}^*) \leq D_c(\underline{z})$, for all $\underline{z}$.

The assignment of the codeword indices is referred to as the index assignment, it can effectively control performance degradation caused by channel noise. The index assignment problem is a NP-hard problem [Chiang and Potter, 1995] and there are n! assignments of indices for n codewords, an exhaustive search for the globally optimal ordering is not feasible.

2.7.1 Index Assignment Algorithms

In this subsection, index assignment algorithms are described as binary switch, simulated annealing, and genetic. Another important approach using Hadamard transformation is presented in the next subsection.

The problem of index assignment affects on the performance of vector quantization system has received much attention in the last 10 years. DeMarca and Jayant [DeMarca and Jayant, 1987] showed that a substantial reduction in channel distortion can be obtained through a judicious assignment of indices rather than a random assignment. Cheng and Kingsbury [Cheng and Kingsbury, 1989] presented an algorithm which sought to minimize the Euclidean distance between all pairs of codewords with a relative Hamming distance of one. Zeger and Gersho [Zeger and Gersho, 1987] published an algorithm which operated by continually swapping codewords until those was convergence to a local minimum but proved no convergence results. The corresponding convergence result was published later [Zeger and Gersho, 1990] and was called the Binary Switch Algorithm (BSA). To our knowledge, no other work has disputed that the BSA is the best performance in minimizing distortion. Therefore, it will be used to benchmark the algorithms proposed in Chapter 3 and 4. The detailed properties of BSA will be discussed further later in subsection 2.7.3.

Some heuristic algorithms such as simulated annealing algorithm and genetic algorithm have also been applied to the index assignment problem. Simulated annealing

[Kirkpatrick et al., 1983] is a random search algorithm in which (1) the next configuration is generated randomly, and (2) "hill climbing" is allowed. Thus, configurations of higher distortion than the present one are accepted according to a certain criterion that takes into consideration the state of the search process. Goodman and Moulsley [Goodman and Moulsley, 1988] introduced simulated annealing techniques to help find index assignment functions of vector quantizers. Farvardin [Farvardin, 1990] proposed an simulated annealing algorithm (SAA) for the index assignment problem with the same bit error rate on each bit and gave a modification of SAA for the case of unequal bit error rates on different bits.

The genetic algorithms discussed by Goldberg [Goldberg, 1989] and Davis [Davis, 1991] are local search algorithms in which the fittest population of codewords is obtained by the following three steps: (1) select the best parents, (2) crossover the parents to produce the next generation, (3) randomly mutate some genetic codes of the existing populations to add genetic diversity. Pan et al. [Pan et al., 1996] applied genetic algorithms to index assignment. Ostrowski and Ruoppila [Ostrowski and Ruoppila, 1997] applied a combination of simulated annealing and genetic algorithms to the index assignment for vector quantizations.

Other index assignment algorithms have also been developed. Wu and Barba [Wu and Barba, 1993] proposed a multilevel structure for index assignment, which outperforms the average results of the random assignment. Hall [Hall, 1970] proposed a method based on the reciprocal of the Euclidean distance between codewords. Cawley and Talbot [Cawley and Talbot, 1996] presented a fast index assignment algorithm based on this method which was comparable to the simulated annealing algorithm but with reduced computation time.

2.7.2 Hadamard Transformation Approach

Some authors have applied Hadamard transformations to solve the index assignment problem. Knagenhjelm [Knagenhjelm, 1993] suggested the quantizer can be expressed as a linear transform of a hypercube to minimize channel distortion for equiprobable quantizers on a binary symmetric channel. He suggested that the Hadamard transformation could give an objective measure called the linearity index, which indicates the dominance the linear transformation of the general nonlinear transformation. The linearity index is a way of measuring the goodness of an index assignment.

McLaughlin et al. [McLaughlin et al., 1995] used Hadamard transformation and the E_1 eigenspace to derive the globally optimal index assignments for the class of equiprobable scalar and vector quantizers called antipodal direct sum codebooks (APDS). Since a uniform source and uniform scalar quantizer is a special case of APDS, the Natural Binary Code (NBC) is globally optimal on binary symmetric noisy channel. For non-APDS codebooks, McLaughlin et al. [McLaughlin et al., 1995] suggested an idea to rearrange the index assignment matrix so that its columns are as nearly in E_1 as possible. In other words, the linearity index is maximized.

Using the Hadamard transformation and maximizing the linearity index, Knagenhjelm and Agrell [Knagenhjelm and Agrell, 1996] presented two index assignment algorithms for equiprobable scalar and vector quantizers: the Full Linear Search Algorithm (FLSA) and the Linearly Increasing Swap Algorithm (LISA). The FLSA performs a full search among $L=(n-3)!\Big/\prod_{i=2}^{b-1}(n-2^i)$ classes with special characteristics to find the index assignment with maximal linearity. However, it is of more theoretical than practical interest because of the rapid increase in L, as n increases. By modifying FLSA, LISA examines just a subset of the L classes and performs a special

type of pairwise swaps. In summary, LISA repeatedly applies the Hamming-1 routine and Remaining routine until the linearity index converges. The Hamming-1 subroutine picks out all pairwise swaps for the $b*2^{b-1}$ neighbors with Hamming distance one. (There are 12 such pairwise swaps for b=3.) The Remaining subroutine is applied to the remaining $2^{b-1}(2^b-b-1)$ pairwise swaps. (There are 16 such pairwise swaps for b=3.) Overall, there are $\binom{n}{2}=2^{b-1}(2^b-1)$ possible pairwise swaps for each cycle of Hamming-1 and Remaining subroutines. Note that a swap is effective only when it results in increasing the linearity index; otherwise, no swap is made. An example based on 8 codewords will be given in Section 3.7 to demonstrate the detailed binary swaps of the LISA.

Knagenhjelm and Agrell [Knagenhjelm and Agrell, 1996] undertook a computational comparison of LISA, SAA and BSA. All three algorithms are based on pairwise swaps to improve a given index assignment and their results depend upon the choice of the initial index assignment. For a small codebook with b=6 and k=6, the SNR performance between the three algorithms is almost indistinguishable, their differences being less than 0.13dB. However, for larger codebook with b=9, k=3, LISA does not compare well with BSA since the difference is about 0.6dB. Overall, in term of SNR performances, BSA is shown to be the best, LISA is the second best and SAA is the third best.

Using the idea of arranging an index assignment vector to be as nearly in E_1 as possible (or maximizing the linearity index), we propose an eigenspace index assignment algorithm (called EIA) for the case of scalar quantization by using regression method and sorting algorithm in Chapter 3. The EIA is simple and efficient since multiple swaps are allowed. Using the Hadamard transformation, the EIA is extended in Chapter 4 to the case of vector quantization. This extension is called eigenspace index assignment algorithm for vector quantizer (VEIA).

2.7.3 Binary Switching Algorithm (BSA)

The Binary Switching Algorithm (BSA) proposed by Zegar and Gersho [Zegar and Gersho, 1990] is designed for general vector quantizers (non-equiprobable or non-APDS) and has the best performance in minimizing channel distortion. Since it will be used to benchmark the algorithms proposed in Chapter 3 and 4, its procedure is stated here for reference. The main idea of BSA is to iteratively switch the positions of paired codevectors to reduce the channel distortion and to stop whenever one reaches a local minimum. Each such switch constitutes a change in the index assignment, i.e. a new assignment matrix. The BSA involves four major steps as follows.

Binary Switching Algorithm (BSA)

Input: An initial index assignment for codebook **C** (or assignment matrix **Z**)

Output: An Optimal index assignment for codebook **C** (or assignment matrix **Z**)

Step 1: Sort codewords based on a cost function:

(1) Assign value Cost($\underline{\xi}_i$) for all $\underline{\xi}_i$, i = 0,...,n-1, where $\text{Cost}(\underline{\xi}_i)=\frac{1}{n}\sum_{j=0}^{n-1} q_{ij} d(\underline{\xi}_i, \underline{\xi}_j)$

and q_{ij} is the transition probability that the channel output is j given that its input is i.

(2) Sort the codevectors $\underline{\xi}_i$ in decreasing order of their cost value Cost($\underline{\xi}_i$).

Step 2: New switches are attempted.

Select $y_{candidate}$ vector with the largest cost as the candidate, say $\underline{\xi}_0$, to be switched first.

Step 3:

(1) Switch ($y_{candidate}$, $\underline{\xi}_i$) temporarily for all i.

(2) Determine the channel distortion $D_c = E[\|\underline{\xi}_i - \underline{\xi}_j\|^2] = \sum_{i=0}^{n-1} \text{Cost}(\underline{\xi}_i)$ following each switch.

(3) Swap $y_{candidate}$ and $\underline{\xi}_i$ with the greatest decrease in D_c or

set new $y_{candidate}$= the second highest cost vector when there is an unsuccessful switching attempt which cannot decrease D_c.

Step 4: If every codeword is such that when switched with every other codeword no decrease in D_c occurs, then the algorithm halts in a locally optimal state
else go to Step 1.

Although the BSA has the best performance of channel distortion, it is a time consuming algorithm. Zegar and Gersho [Zegar and Gersho, 1990] reported it requires 25 CPU hours on a SUN-3/180 machine for a codebook of size 256 with the single bit error assumption. In other words, the BSA requires many swaps in order to minimize the distortion. This motivated us to develop an efficient and robust index assignment algorithm.

Chapter 3

Scalar Quantization Model

A scalar quantizer F of size n is a mapping from units of input $x \in R$ into a codebook containing $n=2^b$ codewords, namely, $F: R \rightarrow C$, where $C = \{y_0, y_1, ..., y_{n-1}\}$ is a codebook and $y_i \in R$ is a codeword. A quantizing partition $S = \{S_0, S_1, ..., S_{n-1}\}$ of R has n cells associated with the n codewords such that $S_i = \{x \in R : F(x) = y_i\}$. Let p(x) denote the probability density function of the quantizer input x, then $p_i = \int_{S_i} p(x)dx$ is the occupancy probability of the i^{th} partition.

In the thesis, we will study the matched equiprobable standardized source-quantizer pair in which all cells are equiprobable, all codewords are cell centroids and the codewords are standardized, that is $p_i=1/n$, $y_i = E[X \mid X \in S_i]$ for all i, and $\sum_{i=0}^{n-1} y_i = 0$. (The latter condition can be obtained by changing the definition of the origin point 0 in R.)

The operation of a quantization system is decomposed into a successive encoding and decoding. The codewords in a codebook are mapped one-to-one onto the indices $i \in I = \{0,1,...,n-1\}$, which can be denoted by b-bit binary sequences corresponding to their representation as binary numbers. This is called the index assignment problem. Given an input $x \in S_i$, an encoder converts it to the selected nearest codeword y_i and delivers its b-bit binary index i to a decoder through a noisy, memoryless, binary symmetric channel (BSC). However, due to effect of channel

noise during the transmission, the received binary index j is not necessary the same as the transmitted index i. Consequently the received codeword becomes y_j for an input $x \in S_i$, and its distortion is measured by $d(x,y_j)$. Regarding channel noise, it is called a single-error pattern if there is at most one bit error per transmitted index, otherwise it is a multiple-error pattern. It is the latter we are concerned with in the thesis.

An index assignment can combat the performance degradation caused by channel noise. Let $\underline{\mathbf{z}}^T = [z_0, z_1, ..., z_{n-1}]$ be an index assignment vector in which each z_i is one of the codewords in C and $i \in I$ is its associated index. Let $z_j(X)$ be the received codeword in a decoder when $z_i(X)$ is the chosen codeword for a random input X, the resulting channel distortion is then measured by the mean squared-error distortion $E[(z_i(X) - z_j(X))^2]$. However, we will drop the dependence of X from our notation. The object of an index assignment algorithm is to minimize the mean squared-error distortion by rearranging the positions of codewords in a given codebook.

This chapter is organized as follows. Section 3.1 presents the properties of channel distortion. Sections 3.2 and 3.3 introduce the ideas of linear eigenspace and type of codebooks, respectively. Then two efficient index assignment algorithms, EIA and SEIA, are proposed in Sections 3.4 and 3.5, respectively. Section 3.6 presents some numerical results to illustrate the properties of the proposed algorithms. There, EIA is compared to two index assignment algorithms, Linearity Increasing Swap Algorithm (LISA) and the Binary Switching Algorithm (BSA), to show the efficiency and robustness of its performance in Sections 3.7 and 3.8, respectively.

3.1 Channel Distortion

For an optimal quantizer with the centroid condition, the mean-squared distortion resulting from an assignment vector $\underline{\mathbf{z}}$ on a binary symmetric channel can

be decomposed into the sum of a quantization distortion (D_s) without channel noise and the channel distortion (D_c or $D_c(\underline{z};q)$) due to channel errors. The details are discussed in Chapter 2. Therefore, for a given channel bit error rate q (0<q<0.5) on a binary symmetric channel, the mean-squared distortion resulting from using the assignment vector $\underline{z}$ representing the random input X is given as

$$\begin{aligned} D(\underline{z},q) &= E(X-z_j)^2 = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} q_{ij}\int_{s_i}(x-z_j)^2 p(x)dx \\ &= \sum_{i=0}^{n-1}\int_{s_i}(x-z_i)^2 p(x)dx + \sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij}(z_i-z_j)^2 = D_s + D_c(\underline{z},q) \end{aligned}, \qquad (3.1)$$

where $q_{ij} = q^{h(i,j)}(1-q)^{b-h(i,j)}$ is the transition probability that the binary symmetric channel output is j given that its input is i and the Hamming distance h(i,j) is the number of bit positions in which i and j are different. Let p=1-q, for the case of b=3, Table 3.1.1 and 3.1.2 show the Hamming distances and the corresponding transition probabilities, respectively.

Table 3.1.1: Hamming distance h(i,j)

h	000	001	010	011	100	101	110	111
000	0	**1**	**1**	2	**1**	2	2	3
001	**1**	0	2	**1**	2	**1**	3	2
010	**1**	2	0	**1**	2	3	**1**	2
011	2	**1**	**1**	0	3	2	2	**1**
100	**1**	2	2	3	0	**1**	**1**	2
101	2	**1**	3	2	**1**	0	2	**1**
110	2	3	**1**	2	**1**	2	0	**1**
111	3	2	2	**1**	2	**1**	**1**	0

Table 3.1.2: Transition Probability q_{ij}

q_{ij}	000	001	010	011	100	101	110	111
000	p^3	$\mathbf{p^2q}$	$\mathbf{p^2q}$	pq^2	$\mathbf{p^2q}$	pq^2	pq^2	q^3
001	$\mathbf{p^2q}$	p^3	pq^2	$\mathbf{p^2q}$	pq^2	$\mathbf{p^2q}$	q^3	pq^2
010	$\mathbf{p^2q}$	pq^2	p^3	$\mathbf{p^2q}$	pq^2	q^3	$\mathbf{p^2q}$	pq^2
011	pq^2	$\mathbf{p^2q}$	$\mathbf{p^2q}$	p^3	q^3	pq^2	pq^2	$\mathbf{p^2q}$
100	$\mathbf{p^2q}$	pq^2	pq^2	q^3	p^3	$\mathbf{p^2q}$	$\mathbf{p^2q}$	pq^2
101	pq^2	$\mathbf{p^2q}$	q^3	pq^2	$\mathbf{p^2q}$	p^3	pq^2	$\mathbf{p^2q}$
110	pq^2	q^3	$\mathbf{p^2q}$	pq^2	$\mathbf{p^2q}$	pq^2	p^3	$\mathbf{p^2q}$
111	q^3	pq^2	pq^2	$\mathbf{p^2q}$	pq^2	$\mathbf{p^2q}$	$\mathbf{p^2q}$	p^3

Since the first term D_s is the distortion caused by quantization and is independent of both $\underline{z}$ and q, we wish to minimize channel distortion $D_c(\underline{z};q)$ caused by misinterpretation of the received codeword by choosing a good index assignment for a given codebook. Please note that minimizing $D_c(\underline{z};q)$ depends

upon q for the case where multiple bit errors may occur on each transmission. However, if a single-error pattern is assumed, the value of q will not affect the minimization of D_c. For a fixed i, we have

$$\begin{aligned}\sum_{j=0}^{n-1} q_{ij} &= \sum_{j=0}^{n-1} q^{h(i,j)}(1-q)^{b-h(i,j)} = \sum_{r=0}^{b} q^r (1-q)^{b-r} \sum_{j\in\tau_i^r} 1 \\ &= \sum_{r=0}^{b} \binom{b}{r} q^r (1-q)^{b-r} = 1\end{aligned}, \tag{3.2}$$

where τ_i^r is the set of all integers j satisfying h(i,j)=r and q_{ij} has a binomial distribution. Its size $|\tau_i^r| = \binom{b}{r}$ because there are r bits that can change and b-r that can not. The set of all $n=2^b$ output codewords are partitioned into b sets such that each set consists of all codewords whose indexes are a fixed Hamming distance from the index i, then Lemma 3.1 is derived as follows.

Lemma 3.1: [Zeger and Gersho, 1990] For single-error patterns, the index that minimizes D_c does not depend upon the value of the bit error rate q.

Proof of Lemma 3.1: For single-error patterns, only a single bit error can occur on each transmission, namely $q_{ij}=0$ for $h(i,j)\geq 2$. Hence the distribution of q_{ij} is

$$q_{ij} = \begin{cases} k(1-q)^b, & \text{if} \quad h(i,j)=0 \\ kq(1-q)^{b-1}, & \text{if} \quad h(i,j)=1 \end{cases}, \text{ where } k=\frac{1}{(1-q)^{b-1}(1+(b-1)q)} \text{ because}$$

$k(1-q)^b+kbq(1-q)^{b-1}=1$. The channel distortion in (3.1) can be rewritten as follows:

$$\begin{aligned} D_c(\underline{z},q) &= \sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij}(z_i - z_j)^2 \\ &= \sum_{i=0}^{n-1} p_i [k(1-q)^b 0 + kq^1(1-q)^{b-1} \sum_{j\in\tau_i^1} (z_i - z_j)]^2 \\ &= kq(1-q)^{b-1}[\sum_{i=0}^{n-1} p_i \sum_{j\in\tau_i^1} (z_i - z_j)^2] = kq(1-q)^{b-1} D_1 \end{aligned}. \tag{3.3}$$

Clearly, for a given q, minimizing $D_c(\underline{z};q)$ is the same as minimizing D_1, which is not a function of q. Therefore, the Lemma is proved. □

Although it is less computationally demanding if the single-error pattern is assumed, we would like to consider the general case of multiple-error pattern throughout the thesis.

For an equiprobable scalar quantizer with $p_i = 1/n$, $D_c(\underline{z};q)$ in (3.1) can be written in a matrix form as below:

$$D_c(\underline{z},q) = \frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} q_{ij}(z_i^2 + z_j^2 - 2z_i z_j) = \frac{2}{n}\left(\sum_{i=0}^{n-1} z_i^2 - \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} q_{ij} z_i z_j\right)$$
$$= \frac{2}{n}\|\underline{z}\|^2 - \frac{2}{n}\underline{z}^T \mathbf{Q}_n \underline{z} \quad , \qquad (3.4)$$

where $\|\underline{z}\|$ is the Euclidean length of $\underline{z}$ and $\mathbf{Q}_n = [q_{ij}]$ is the $n \times n$ positive definite matrix of channel transition probabilities with $\sum_{i=0}^{n-1} q_{ij} = \sum_{j=0}^{n-1} q_{ij} = 1$. Since $\|\underline{z}\|^2$ is independent of permuting the elements in $\underline{z}$, minimizing $D_c(\underline{z};q)$ in (3.4) is the same as maximizing the quadratic form of $\underline{z}^T \mathbf{Q}_n \underline{z}$.

3.2 Linear Eigenspace

To facilitate the development of the proposed index assignment algorithms, some properties of linear eigenspace [Noble and Daniel, 1988] and the alternative expression of channel distortion are explored in details in this section. Johnson and Wichern [Johnson and Wichern, 1988] defined the spectral decomposition for a square symmetric matrix as follow:

The spectral decomposition of a $n \times n$ symmetric matrix $\mathbf{A}_n$ can be factorized as $\mathbf{A}_n = \sum_{i=0}^{n-1} \upsilon_i \underline{\mathbf{p}}_i \underline{\mathbf{p}}_i^T = \mathbf{P}_n \mathbf{V}_n \mathbf{P}_n^T$, where $\mathbf{V}_n = \mathrm{diag}(\upsilon_0, \upsilon_1, \cdots, \upsilon_{n-1})$ whose elements are the eigenvalues of $\mathbf{A}_n$, and the orthogonal matrix $\mathbf{P}_n = \{\underline{\mathbf{p}}_0, \cdots, \underline{\mathbf{p}}_{n-1}\}$ has columns which are the associated normalized eigenvectors, i.e., $\underline{\mathbf{p}}_i^T \underline{\mathbf{p}}_i = 1$, for i=0,1,…,n-1 and $\underline{\mathbf{p}}_i^T \underline{\mathbf{p}}_j = 0$, for $i \neq j$. Note that $\underline{\mathbf{p}}_i^T \mathbf{A}_n \underline{\mathbf{p}}_i = \upsilon_i > 0$, for all i=0,1,…,n-1.

Since the n×n matrix $\mathbf{Q}_n$ of channel transition probability is symmetric, the spectral decomposition of an n×n positive definite channel transition probability matrix $\mathbf{Q}_n$ can be factorized as

$$\mathbf{Q}_n = \sum_{i=0}^{n-1} \lambda_i \underline{\mathbf{e}}_i \underline{\mathbf{e}}_i^T = \mathbf{M}_n \mathbf{\Lambda}_n \mathbf{M}_n^T , \tag{3.5}$$

where $\mathbf{\Lambda}_n = \mathrm{diag}(\lambda_0, \lambda_1, \cdots, \lambda_{n-1})$ whose elements are the eigenvalues of $\mathbf{Q}_n$ and the normalized Hadamard $\mathbf{M}_n = \{\underline{\mathbf{e}}_0, \cdots, \underline{\mathbf{e}}_{n-1}\}$ whose columns are the associated orthonormal eigenvectors of $\mathbf{Q}_n$. It is well known that the transition probability matrix as well as its eigenvalues and eigenvectors have a simple recursive structure [McLaughlin, et. al., 1995]. A size $2^b \times 2^b$ matrix can be easily expressed by using b successive Kronecker products as defined in Chapter 2.

When b=1, $\mathbf{Q}_2 = \begin{bmatrix} p & q \\ q & p \end{bmatrix}$, p=1-q. Since $|\mathbf{Q}_2 - \lambda \mathbf{I}| = 0$, we have λ_0=1 and λ_1=1-2q>0 for 0<q<0.5, that is $\mathbf{\Lambda}_2 = \mathrm{diag}(1, \ (1-2q))$. Then solving for the associated eigenvectors in the equation $\mathbf{Q}_2\underline{\mathbf{e}}_i=\lambda_i\underline{\mathbf{e}}_i$, for i=0,1, we have $\mathbf{M}_2 = [\underline{\mathbf{e}}_0 \quad \underline{\mathbf{e}}_1] = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} / \sqrt{2}$, thus $\mathbf{Q}_2\mathbf{M}_2 = \mathbf{M}_2\mathbf{\Lambda}_2$. Therefore, we have $\mathbf{Q}_2 = \mathbf{M}_2\Lambda_2\mathbf{M}_2^T$. Besides, $\underline{\mathbf{e}}_0^T\mathbf{Q}_2\underline{\mathbf{e}}_0 = 1 = \lambda_0$, $\underline{\mathbf{e}}_1^T\mathbf{Q}_2\underline{\mathbf{e}}_1 = \frac{1}{2}\begin{bmatrix} 1 & -1 \end{bmatrix}\begin{bmatrix} p & q \\ q & p \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} =$ $1-2q = \lambda_1$. There are $\binom{1}{j}$ eigenvalues with the same value $\lambda_j = (1-2q)^j$, j=0,1.

When b=2, the transition matrix and its eigenvalues and eigenvectors can be obtained by Kronecker product such as:

$$\mathbf{Q}_{2^2} = \mathbf{Q}_2 \otimes \mathbf{Q}_2 = \begin{bmatrix} p\mathbf{Q}_2 & q\mathbf{Q}_2 \\ q\mathbf{Q}_2 & p\mathbf{Q}_2 \end{bmatrix} = \begin{bmatrix} p\begin{bmatrix} p & q \\ q & p \end{bmatrix} & q\begin{bmatrix} p & q \\ q & p \end{bmatrix} \\ q\begin{bmatrix} p & q \\ q & p \end{bmatrix} & p\begin{bmatrix} p & q \\ q & p \end{bmatrix} \end{bmatrix} = \begin{bmatrix} p^2 & pq & qp & q^2 \\ pq & p^2 & q^2 & qp \\ qp & q^2 & p^2 & pq \\ q^2 & qp & pq & p^2 \end{bmatrix},$$

$$\mathbf{\Lambda}_{2^2} = \mathbf{\Lambda}_2 \otimes \mathbf{\Lambda}_2 = \begin{bmatrix} \mathbf{\Lambda}_2 & 0 \\ 0 & (1-2q)\mathbf{\Lambda}_2 \end{bmatrix}$$

$$= \begin{bmatrix} 1\begin{bmatrix} 1 & 0 \\ 0 & 1-2q \end{bmatrix} & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} & (1-2q)\begin{bmatrix} 1 & 0 \\ 0 & 1-2q \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-2q & 0 & 0 \\ 0 & 0 & 1-2q & 0 \\ 0 & 0 & 0 & (1-2q)^2 \end{bmatrix},$$

$$\mathbf{M}_{2^2} = \mathbf{M}_2 \otimes \mathbf{M}_2 = \begin{bmatrix} \mathbf{M}_2 & \mathbf{M}_2 \\ \mathbf{M}_2 & -\mathbf{M}_2 \end{bmatrix} / \sqrt{2} = \begin{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & -1\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \end{bmatrix} / \left(\sqrt{2}\right)^2$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} / 2$$

then it is easy to verify that $\mathbf{Q}_4 = \mathbf{M}_4 \Lambda_4 \mathbf{M}_4^T$ and $\underline{\mathbf{e}}_0^T \mathbf{Q}_4 \underline{\mathbf{e}}_0 = 1 = \lambda_0$, $\underline{\mathbf{e}}_1^T \mathbf{Q}_4 \underline{\mathbf{e}}_1 = \underline{\mathbf{e}}_2^T \mathbf{Q}_4 \underline{\mathbf{e}}_2 = 1-2q = \lambda_1$, $\underline{\mathbf{e}}_3^T \mathbf{Q}_4 \underline{\mathbf{e}}_3 = (1-2q)^2 = \lambda_2$. There are $\binom{2}{j}$ eigenvalues with the same value $\lambda_j = (1-2q)^j$, j=0,1,2.

In general, we have

$$\mathbf{Q}_{2^{k+1}} = \mathbf{Q}_{2^k} \otimes \mathbf{Q}_2 = \begin{bmatrix} p\mathbf{Q}_{2^k} & q\mathbf{Q}_{2^k} \\ q\mathbf{Q}_{2^k} & p\mathbf{Q}_2 \end{bmatrix},$$

$$\mathbf{\Lambda}_{2^{k+1}} = \mathbf{\Lambda}_{2^k} \otimes \mathbf{\Lambda}_2 = \begin{bmatrix} \mathbf{\Lambda}_{2^k} & 0 \\ 0 & (1-2q)\mathbf{\Lambda}_{2^k} \end{bmatrix},$$

$$\mathbf{M}_{2^{k+1}} = \mathbf{M}_{2^k} \otimes \mathbf{M}_2 = \begin{bmatrix} \mathbf{M}_{2^k} & \mathbf{M}_{2^k} \\ \mathbf{M}_{2^k} & -\mathbf{M}_{2^k} \end{bmatrix} / \sqrt{2}$$

Thus, $\mathbf{Q}_{2^{k+1}} = \mathbf{M}_{2^{k+1}} \mathbf{\Lambda}_{2^{k+1}} \mathbf{M}_{2^{k+1}}^T$, and $\underline{\mathbf{e}}_i^T \mathbf{Q}_{2^{k+1}} \underline{\mathbf{e}}_i = \lambda_j > 0$, i=0,1,…,$2^{k+1}$-1, j=0,1,..,k+1.

If $(\lambda, \underline{\mathbf{e}})$ is an eigenvalue-eigenvector pair of $\mathbf{Q}_{2^k}$, then $\left[\lambda, \begin{pmatrix} \underline{\mathbf{e}} \\ \underline{\mathbf{e}} \end{pmatrix}\right]$ and

$\left[(1-2q)\lambda, \begin{pmatrix} \underline{\mathbf{e}} \\ -\underline{\mathbf{e}} \end{pmatrix}\right]$ are eigenvalue-eigenvector pairs of $\mathbf{Q}_{2^{k+1}}$. There are $\binom{k+1}{j}$ eigenvalues with the same value $\lambda_j = (1-2q)^j$, j=0,1,...,k+1. Therefore, the following Lemma is easily deduced.

Lemma 3.2: [McLaughlin, et. al., 1995] The spectral decomposition of an n×n positive definite channel transition probability matrix $\mathbf{Q}_n$ can be factorized as

$$\mathbf{Q}_n = \sum_{i=0}^{n-1} \lambda_i \underline{\mathbf{e}}_i \underline{\mathbf{e}}_i^T = \mathbf{M}_n \mathbf{\Lambda}_n \mathbf{M}_n^T ,$$

where $\mathbf{\Lambda}_n = \text{diag}(\lambda_0, \lambda_1, \cdots, \lambda_{n-1})$ whose elements are the eigenvalues of $\mathbf{Q}_n$ and the normalized Hadamard $\mathbf{M}_n = \{\underline{\mathbf{e}}_0, \cdots, \underline{\mathbf{e}}_{n-1}\}$ whose columns are the associated orthonormal eigenvectors of $\mathbf{Q}_n$. Note that $\underline{\mathbf{e}}_i^T \mathbf{Q}_n \underline{\mathbf{e}}_i = \lambda_j > 0$, i=0,1,…,$2^b$-1, j=0,1,..,b. There are $b_j = \binom{b}{j}$ eigenvalues with the same value $\lambda_j = (1-2q)^j$, j=0,1,...,b.

Hence the associated eigenspace E_j of $\mathbf{Q}_n$ is spanned by those b_j eigenvectors corresponding to λ_j. Let $\mathbf{H}_b = [\underline{\mathbf{h}}_0, ..., \underline{\mathbf{h}}_{b-1}]$ denote the $n \times b$ linear matrix whose columns are the b eigenvectors corresponding to $\lambda_1 = (1-2q)^1$ multiplied by $\sqrt{n}$, then the linear eigenspace E_1 is spanned by $\mathbf{H}_b$. For instance, $\mathbf{H}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$,

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix}, \qquad \mathbf{H}_3 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix},$$

and in general,

$$\mathbf{H}_b = [\underline{\mathbf{h}}_0, \underline{\mathbf{h}}_1, ..., \underline{\mathbf{h}}_{b-1}] = \begin{bmatrix} \underbrace{1}_{2^0} & \underbrace{-1}_{2^0} & \cdots & \underbrace{1}_{2^0} & \underbrace{-1}_{2^0} \\ \underbrace{1\ 1}_{2^1} \underbrace{-1}_{2^1} & \underbrace{-1}_{2^1} & \cdots & \underbrace{1\ 1}_{2^1} \underbrace{-1}_{2^1} & \underbrace{-1}_{2^1} \\ \vdots & & & & \\ \underbrace{1 \cdots 1}_{2^{b-1}} & \underbrace{-1 \cdots -1}_{2^{b-1}} & & & \end{bmatrix}^T .$$

Lemma 3.3: [McLaughlin, et. al., 1995] For a matched equiprobable standardized scalar source-quantizer pair, the channel distortion can be written as

$$D_c(\underline{\mathbf{z}}, q) = \frac{2}{n} \sum_{j=1}^{b} [1 - (1-2q)^j] \left\| \text{Proj}_j \underline{\mathbf{z}} \right\|^2, \text{for} \quad 0 < q < 0.5 . \tag{3.6}$$

Proof of Lemma 3.3:

The space of $\mathbf{Q}_n$ can be decomposed into b eigenspaces, say E_j, j=0,1,…,b and the squared length of the projection of $\underline{\mathbf{z}}$ onto the eigenspace E_j is given by $\left\| \text{Proj}_j \underline{\mathbf{z}} \right\|^2 = \sum_{k=1}^{b_j} (\underline{\mathbf{z}}^T \underline{\mathbf{e}}_k)^2$, where $b_j = \binom{b}{j}$. Since $\| \text{Proj}_0 \underline{\mathbf{z}} \| = \underline{\mathbf{z}}^T \underline{\mathbf{e}}_0 = [z_0, z_1, ..., z_{n-1}][1,1,...,1]^T = \sum_{i=0}^{n-1} z_i$, the zero-mean codebook assumption $\sum_{i=0}^{n-1} z_i = 0$ implies that $\left\| \text{Proj}_0 \underline{\mathbf{z}} \right\| = 0$ and the j=0 term can be omitted. Therefore, from $\mathbf{Q}_n = \mathbf{M}_n \Lambda_n \mathbf{M}_n^T$ in (3.5),

$$\begin{aligned} \underline{\mathbf{z}}^T \mathbf{Q}_n \underline{\mathbf{z}} &= \underline{\mathbf{z}}^T \mathbf{M}_n \Lambda_n \mathbf{M}_n^T \underline{\mathbf{z}} = \sum_{i=0}^{n-1} \lambda_i (\underline{\mathbf{z}}^T \underline{\mathbf{e}}_i)^2 = \sum_{j=0}^{b} \lambda_j \sum_{k=1}^{b_j} (\underline{\mathbf{z}}^T \underline{\mathbf{e}}_k)^2 \\ &= \sum_{j=0}^{b} \lambda_j \left\| \text{Proj}_j \underline{\mathbf{z}} \right\|^2 = \sum_{j=1}^{b} (1-2q)^j \left\| \text{Proj}_j \underline{\mathbf{z}} \right\|^2 . \end{aligned} \tag{3.7}$$

By the fact of $\|\underline{\mathbf{z}}\|^2 = \sum_{j=1}^{b} \left\| \text{Proj}_j \underline{\mathbf{z}} \right\|^2$ and (3.7), $D_c(\underline{\mathbf{z}}, q) = \frac{2}{n} \|\underline{\mathbf{z}}\|^2 - \frac{2}{n} \underline{\mathbf{z}}^T \mathbf{Q}_n \underline{\mathbf{z}}$ in (3.4) is rewritten as

$$D_c(\underline{\mathbf{z}}, q) = \frac{2}{n} \sum_{j=1}^{b} [1 - (1-2q)^j] \left\| \text{Proj}_j \underline{\mathbf{z}} \right\|^2, \text{for} \quad 0 < q < 0.5 . \qquad \square$$

Note that $D_c(\underline{z},q)$ is equal to 0 when q=0; $D_c(\underline{z},q)$ is equal to $2\|\underline{z}\|^2/n$ when q=0.5, and is increasing in q for 0<q<0.5. For the case of 0.5<q<1, intuitively we can convert 0 (or 1) to 1 (or 0) in the decoding process, consequently the new bit error rate becomes q'=1-q which can be directly applied into (3.6). Therefore, the following Corollary can be easily obtained.

Corollary 3.4: For a matched equiprobable standardized scalar source-quantizer pair, the channel distortion can be written as

$$D_c(\underline{z},q)=\frac{2}{n}\sum_{j=1}^{b}[1-(2q-1)^j]\|Proj_j\underline{z}\|^2, \text{for} \quad 0.5<q<1. \tag{3.8}$$

Furthermore, minimizing D_c in (3.6) is the same as maximizing $\sum_{j=1}^{b}(1-2q)^j\|Proj_j\underline{z}\|^2$. One tries to find an index assignment of $\underline{z}$ such that the projection of $\underline{z}$ onto the eigenspace corresponding to the largest eigenvalue is as great as possible. The zero-mean codebook assumption implies $\|Proj_0\underline{z}\|=0$ that any choice of $\underline{z}$ has zero projection onto eigenspace E_0 corresponding to the largest eigenvalue $\lambda_0=1$. Thus we try to find an index assignment of $\underline{z}$ such that the projection of $\underline{z}$ onto eigenspace E_1 corresponding to the second largest eigenvalue $\lambda_1=1-2q$ is as great as possible. Lemma 3.5 is then deduced as follows.

Lemma 3.5: [McLaughlin, et. al., 1995] For an equiprobable standardized scalar source-quantizer pair, if an assignment vector $\underline{z}$ lies entirely in the linear eigenspace E_1, then the index assignment implied by $\underline{z}$ is globally optimal and its channel distortion is $D_{cmin}=4q\|\underline{z}\|^2/n$.

Proof of Lemma 3.5:

Since $[1-(1-2q)^b]>[1-(1-2q)^{b-1}]>...>[1-(1-2q)]$ for a given q, then

$$D_c(\underline{z},q)=\frac{2}{n}\sum_{j=1}^{b}[1-(1-2q)^j]\|Proj_j\underline{z}\|^2 \geq \frac{2}{n}\sum_{j=1}^{b}[1-(1-2q)]\|Proj_j\underline{z}\|^2$$

$$= \frac{4q}{n} \sum_{j=1}^{b} \left\| \mathrm{Proj}_j \underline{z} \right\|^2 = \frac{4q}{n} \left\| \underline{z} \right\|^2 .$$

If an assignment vector $\underline{z}$ lies entirely in the linear eigenspace E_1, we have $\left\| \mathrm{Proj}_j \underline{z} \right\| = 0$, for $j \geq 2$ and $\left\| \underline{z} \right\|^2 = \left\| \mathrm{Proj}_1 \underline{z} \right\|^2$, then the channel distortion is given by $D_c(\underline{z}, q) = 4q \left\| \underline{z} \right\|^2 / n$, which is the lower bound of channel distortion. Therefore, the Lemma is proved. □

Knagenhjelm [Knagenhjelm, 1993] defined a "linearity index" to measure the closeness between $\underline{z}$ and the linear eigenspace E_1 as follow:

$$\eta = \left\| \mathrm{Proj}_1 \underline{z} \right\|^2 / \left\| \underline{z} \right\|^2 . \tag{3.9}$$

Lemma 3.5 implies that if η=1, then an assignment vector $\underline{z}$ lies in E_1 and is globally optimal. Therefore, this linearity index indicates how good is an index assignment.

For any two index assignment vectors with the same linearity index, they are not necessarily identical since their projections on the j^{th} eigenspace for $j \geq 2$ might be different, and the difference of their channel distortions is given in the following lemma.

Lemma 3.6: For any two index assignment vectors with the same linearity index, the absolute difference of their channel distortions is bounded by $2[1-(1-2q)^b](1-\eta)\left\| \underline{z} \right\|^2 / n$.

Proof of Lemma 3.6:

Let $\underline{z}_1$ and $\underline{z}_2$ denote any two index assignment vectors with the same value of η, then we have $\left\| \mathrm{Proj}_1 \underline{z}_i \right\|^2 / \left\| \underline{z} \right\|^2 = \eta$ and $\sum_{j=2}^{b} \left\| \mathrm{Proj}_j \underline{z}_i \right\|^2 / \left\| \underline{z} \right\|^2 = 1-\eta$ so that

$$
\begin{aligned}
&| D_c(\underline{z}_1, q) - D_c(\underline{z}_2, q) | / \|\underline{z}\|^2 \\
&= | \frac{2}{n} \sum_{j=2}^{b} [1-(1-2q)^j] [\|\mathrm{Proj}_j \underline{z}_1\|^2 - \|\mathrm{Proj}_j \underline{z}_2\|^2] | / \|\underline{z}\|^2 \\
&\le \frac{2}{n} [1-(1-2q)^b] | \sum_{j=2}^{b} \|\mathrm{Proj}_j \underline{z}_1\|^2 - \sum_{j=2}^{b} \|\mathrm{Proj}_j \underline{z}_2\|^2 | / \|\underline{z}\|^2 \le \frac{2}{n} [1-(1-2q)^b](1-\eta)
\end{aligned}
$$

□

3.3 Types of Codebooks

Various codebooks will be used to demonstrate the proposed index assignment algorithms in later sections. Codebooks can be categorized into antipodal direct sum (APDS) and non-APDS codebooks by using the idea of the linear eigenspace E_1. Their properties are discussed in this section.

McLaughlin, Neuhoff and Ashley [McLaughlin, et. al., 1995] proposed an antipodal direct sum codebook (APDS) whose codewords have the format of combinations of $\{\pm c_0 \pm c_1 \pm \cdots \pm c_{b-1}\}$, where c_i's are distinct real numbers where no c_i can be obtained by adding or subtracting values of others. Namely, codewords are generated from all combinations of plus and minus signs of c_i. Equivalently, APDS codebooks can be expressed by using Hadamard matrices. Recall that the linear eigenspace E_1 is spanned by b eigenvectors whose corresponding eigenvalues are equal to (1-2q), and $\mathbf{H}_b = [\underline{\mathbf{h}}_0, \ldots, \underline{\mathbf{h}}_{b-1}]$ is the $n \times b$ linear matrix whose columns are the above eigenvectors multiplied by $\sqrt{n}$. For instance, the normalized Hadamard matrix $\mathbf{M}_8$ for n=8 is given by Kronecker products as follow:

$$
\mathbf{M}_8 = [\underline{\mathbf{e}}_0, \ldots, \underline{\mathbf{e}}_7] = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1
\end{bmatrix} / \sqrt{8} \quad , \qquad (3.10)
$$

and the corresponding eigenvalues are $(\lambda_0,...,\lambda_7)=(1,\delta,\delta,\delta^2,\delta,\delta^2,\delta^2,\delta^3)$, where $\delta=1-2q$. Therefore, E_1 is spanned by the linear matrix $\mathbf{H}_3$ such that

$$\mathbf{H}_3=[\underline{\mathbf{h}}_0,\underline{\mathbf{h}}_1,\underline{\mathbf{h}}_2]=\sqrt{8}[\underline{\mathbf{e}}_1,\underline{\mathbf{e}}_2,\underline{\mathbf{e}}_4]=\begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{3.11}$$

Now, the codewords in an APDS codebook can be rearranged in a matrix format as $\underline{\mathbf{z}}=\mathbf{H}_b\underline{\mathbf{c}}=\sum_{j=0}^{b-1}c_j\underline{\mathbf{h}}_j$, where $\underline{\mathbf{h}}_j$'s are the eigenvectors spanning E_1. Since $\underline{\mathbf{z}}$ is a linear combination of $\underline{\mathbf{h}}_j$'s, clearly it is a member of E_1 and is a globally optimal assignment vector by Lemma 3.5. Hence, we have the following lemma.

Lemma 3.7: [McLaughlin, et. al., 1995] If a matched equiprobable source-quantizer pair has an antipodal direct-sum codebook, there exists a globally optimal assignment vector $\underline{\mathbf{z}}$, which lies entirely in the linear eigenspace E_1, and its associated linearity index is equal to one.

Consider a special case of antipodal direct-sum codebook, a uniform scalar codebook of size n with step size $\Delta=z_i-z_{i-1}$, $C=\left\{\frac{-n+1}{2}\Delta,\frac{-n+3}{2}\Delta,\cdots,\frac{n-1}{2}\Delta\right\}$. Arrange codewords in ascending order so that $y_0<y_1<\ldots<y_{n-1}$. The corresponding assignment vector $\underline{\mathbf{z}}=[z_0,z_1,\ldots,z_{n-1}]=[y_0,y_1,\ldots,y_{n-1}]=\left[\frac{-n+1}{2}\Delta,\frac{-n+3}{2}\Delta,\cdots,\frac{n-1}{2}\Delta\right]^T$ is a linear combination of eigenvectors $\underline{\mathbf{h}}_j$'s spanning eigenspace E_1, denoted as $\underline{\mathbf{z}}=\mathbf{H}_b\underline{\mathbf{c}}=\sum_{j=0}^{b-1}c_j\underline{\mathbf{h}}_j$, where $\underline{\mathbf{c}}=[c_0,c_1,...,c_{b-1}]^T=[-(\Delta/2),-(\Delta/2)2,...,-(\Delta/2)2^{b-1}]^T$

$$\text{and } \mathbf{H}_b = [\underline{\mathbf{h}}_0, \underline{\mathbf{h}}_1, ..., \underline{\mathbf{h}}_{b-1}] = \begin{bmatrix} \underbrace{1}_{2^0} & \underbrace{-1}_{2^0} & \cdots & \underbrace{1}_{2^0} & \underbrace{-1}_{2^0} \\ \underbrace{1}_{2^1} \underbrace{1-1}_{2^1} & -1 & \cdots & \underbrace{1}_{2^1} \underbrace{1-1}_{2^1} & -1 \\ \vdots & & & & \\ \underbrace{1 \cdots 1}_{2^{b-1}} & \underbrace{1-1 \cdots -1}_{2^{b-1}} & & & \end{bmatrix}^T . \qquad (3.12)$$

Since $\underline{\mathbf{z}}$ is a linear combination of $\underline{\mathbf{h}}_j$'s, clearly it is a member of E_1 and is a globally optimal assignment vector by Lemma 3.7. Also, the codewords $y_0<y_1<...<y_{n-1}$ is ordered in the same way as the indices 0,...,n-1, NBC is an optimal index assignment.

Hence, we have the following Corollary.

Corollary 3.8: [McLaughlin, et. al., 1995] For a uniform source on $\left[\frac{-n}{2}\Delta, \frac{n}{2}\Delta\right]$ matched to an n-point uniform scalar codebook and a binary symmetric channel with bit error rate q<0.5, the NBC is a globally optimal index assignment.

Given any index assignment vector for a non-APDS codebook, its higher order eigenspace projections are not all equal to zero's, namely, at least one $\left\|\mathrm{Proj}_j \underline{\mathbf{z}}\right\|^2 > 0$, for j=2,3,...,b. Hence, the optimal index assignment for a non-APDS codebook does not lie entirely in E_1. In other word, the associated η is not equal to one when $\underline{\mathbf{z}}$ is globally optimal. Table 3.2 summaries the relationship between codebook and linearity index η when $\underline{\mathbf{z}}$ is globally optimal.

Table 3.2: The Maximal Value of Linearity Index for Different Types of Codebook

Codebook		Linearity Index η	Optimal index assignment
APDS	Uniform	η=1	NBC
	Non-uniform	η=1	-
Non-APDS		0<η<1	-

3.4 Eigenspace Index Assignment Algorithm (EIA)

The main purpose of the remainder of the chapter is to find the optimal index assignment which minimizes channel distortion D_c for the case of a matched equiprobable standardized scalar source-quantizer pair. Note that the optimal index assignment which minimizes D_c depends on the choice of q, the designated bit error rate q_d.

In this section, we proceed to develop an eigenspace index assignment (EIA) algorithm in which $\underline{\mathbf{z}}$ is permuted so that its projection onto the linear eigenspace E_1 is as large as possible while D_c is kept as small as possible.

Regressing $\underline{\mathbf{z}}$ on the linear matrix $\mathbf{H}_b$, we have the regression equation as:

$$\underline{\mathbf{z}} = \mathbf{H}_b \underline{\boldsymbol{\beta}} + \underline{\boldsymbol{\varepsilon}}, \tag{3.13}$$

where $\underline{\boldsymbol{\varepsilon}}$ is an error term including the projections onto all subspaces other than E_1. Since the columns in $\mathbf{H}_b$ are mutually orthogonal, the statistical estimation of β_i can be simplified as

$$\hat{\beta}_i = (\underline{\mathbf{h}}_i{}^T \underline{\mathbf{z}})/n, \tag{3.14}$$

and the predicted value of $\underline{\mathbf{z}}$ is given by

$$\underline{\hat{\mathbf{z}}} = \mathbf{H}_b \underline{\hat{\boldsymbol{\beta}}}. \tag{3.15}$$

It is clear that $\underline{\hat{\mathbf{z}}}$ is a linear combination of columns in $\mathbf{H}_b$, hence it is a member of E_1. Thus, the squared length of the projection of $\underline{\mathbf{z}}$ onto E_1 is equal to

$$\left\| \mathrm{Proj}_1 \underline{\mathbf{z}} \right\|^2 = \underline{\hat{\mathbf{z}}}^T \underline{\hat{\mathbf{z}}} = \underline{\hat{\boldsymbol{\beta}}}^T \mathbf{H}_b^T \mathbf{H}_b \underline{\hat{\boldsymbol{\beta}}} = n \sum_{j=0}^{b-1} \hat{\beta}_j^2 . \tag{3.16}$$

If the elements in $\underline{\mathbf{z}}$ can be permuted as close to $\underline{\hat{\mathbf{z}}}$ as possible, then the permuted $\underline{\mathbf{z}}$ will lie nearly in E_1. To facilitate the permutation method, let Rank($\underline{\mathbf{x}}$) denote the ascending rank order of a given vector $\underline{\mathbf{x}} = (x_0, x_1, ..., x_{n-1})$, for instance,

$\text{Rank}(\underline{\mathbf{x}}) = \{0,1,...,n-1\}$ when $x_0<x_1<x_2<\ldots<x_{n-1}$, then the following lemma holds.

Lemma 3.9: Maximization of Inner Product: Let $\underline{\mathbf{a}} = (a_0, a_1, ..., a_{n-1})$ and $\underline{\mathbf{b}} = (b_0, b_1, ..., b_{n-1})$ be any two given vectors. The sufficient condition for maximizing the inner product of $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ without changing the values of their elements is to sort $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ so that $\text{Rank}(\underline{\mathbf{a}}) = \text{Rank}(\underline{\mathbf{b}})$.

Proof of Lemma 3.9: Let $J = \underline{\mathbf{a}}^T \underline{\mathbf{b}} = \sum_{k=0}^{n-1} a_k b_k$ denote the inner product of $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$. It is invariant if (a_i,b_i) and (a_j,b_j) are pairwisely swapped. Hence, without loss of generality, we assume that $a_0 < a_1 < \ldots < a_i \ldots < a_j < \ldots < a_{n-1}$ and $b_0 < b_1 < \ldots < b_i < \ldots < b_j < \ldots < b_{n-1}$, i.e. $\text{Rank}(\underline{\mathbf{a}}) = \text{Rank}(\underline{\mathbf{b}}) = \{0,1,2,\ldots,n-1\}$. Now b_i and b_j are swapped each other while a_i's are kept the same, and let $J^{(0)}$ and $J^{(1)}$ denote the inner product before and after this swap, respectively. Then

$$J^{(0)} - J^{(1)} = (a_i b_i + a_j b_j) - (a_i b_j + a_j b_i) = (a_i - a_j)(b_i - b_j) > 0. \qquad (3.17)$$

This proves that $J^{(1)}$ is always less than $J^{(0)}$ if any two elements in $\underline{\mathbf{a}}$ or $\underline{\mathbf{b}}$ are swapped each other. In other words, the inner product of $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ is maximized when $\text{Rank}(\underline{\mathbf{a}}) = \text{Rank}(\underline{\mathbf{b}})$. It is quite easy to extend the result to the case of multiple swaps. □

Since the inner product of $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ can be expressed as $\underline{\mathbf{a}}^T \underline{\mathbf{b}} = \|\underline{\mathbf{a}}\| \|\underline{\mathbf{b}}\| \cos(\theta)$, maximizing the inner product is equivalent to minimizing the angle θ. This implies that $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ are as close as possible to each other. For instance, consider $\underline{\mathbf{a}} = (1,2,3)$ and $\underline{\mathbf{b}} = (5,6,4)$, then $\underline{\mathbf{a}}^T \underline{\mathbf{b}} = 29$ and $\theta = \pi / 6.4$. Now, rearrange the elements in each vector such as $\underline{\mathbf{a}} = (1,2,3)$ and $\underline{\mathbf{b}} = (4,5,6)$ so $\text{Rank}(\underline{\mathbf{a}}) = \text{Rank}(\underline{\mathbf{b}})$, then the maximal inner product is obtained as 32 and the minimal angle is equal to $\theta = \pi / 13.7$.

When $\text{Rank}(\underline{\mathbf{z}}) \neq \text{Rank}(\hat{\underline{\mathbf{z}}})$, by applying Lemma 3.9 the elements in $\underline{\mathbf{z}}$ can be sorted into new order according to $\text{Rank}(\hat{\underline{\mathbf{z}}})$. After sorting, the angle between the

new $\underline{\mathbf{z}}$ and $\hat{\underline{\mathbf{z}}}$ decreases, hence the new $\underline{\mathbf{z}}$ is closer to $\hat{\underline{\mathbf{z}}}$ which lies in E_1, and consequently η must increase. However, by the regression model in (3.13) the new $\underline{\mathbf{z}}$ can produce a new $\hat{\underline{\mathbf{z}}}$ which again is another member of E_1, thus another sorting may be performed so that another new $\underline{\mathbf{z}}$ will be more closer to E_1. Therefore the sorting can be repeated until $\text{Rank}(\underline{\mathbf{z}}) = \text{Rank}(\hat{\underline{\mathbf{z}}})$, or equivalently η stops increasing or converges. This fact leads to the following Corollary.

Corollary 3.10: Sorting elements in $\underline{\mathbf{z}}$ into new order according to $\text{Rank}(\hat{\underline{\mathbf{z}}})$ minimizes the angle between new $\underline{\mathbf{z}}$ and $\hat{\underline{\mathbf{z}}}$ which is a member of E_1. This strictly increases the value of the linearity index η.

This result can be incorporated into an iterative procedure to obtain the optimal solution by maximizing η. In the proof of Lemma 3.5, when $\underline{\mathbf{z}}$ is close enough to E_1, η is large and D_c is small. However, D_c is not necessarily strictly decreasing in η but Knagenhjelm and Agrell [Knagenhjelm and Agrell, 1996] showed that D_c and η have a high correlation. They showed the correlation between D_c and η is –0.846 based on 100,000 samples for a 4-bit codebook with q=0.01. Thus maximizing η does not guarantee minimizing D_c. So, the strategy in the proposed EIA algorithm is to maximize η iteratively until $\text{Rank}(\underline{\mathbf{z}}) = \text{Rank}(\hat{\underline{\mathbf{z}}})$, and then choose the smallest D_c among iterations. Alternatively, the algorithm can be stopped whenever D_c increases.

The stopping rule for the EIA algorithm is to stop when the linearity index is maximized or $\text{Rank}(\underline{\mathbf{z}}) = \text{Rank}(\hat{\underline{\mathbf{z}}})$ while D_c is kept as small as possible. The EIA algorithm can be summarized as follows:

Eigenspace Index Assignment Algorithm (EIA):

Input: An initial index assignment vector $\underline{\mathbf{z}}^{(0)}$.

Output: A locally optimal index assignment vector, also named $\underline{\mathbf{z}}^{(i)}$.

Step 0: Set an iteration indicator i=0. Choose an initial assignment vector $\underline{\mathbf{z}}^{(i)}$, and choose a designated bit error rate q_d.

Step 1: Regress $\underline{\mathbf{z}}^{(i)}$ on the linear matrix $\mathbf{H}_b$, then compute η, $D_c(\underline{\mathbf{z}}^{(i)}, q_d)$, and the predicted value $\hat{\underline{\mathbf{z}}}^{(i)}$.

Step 2: Record the minimal distortion of $D_c(\underline{\mathbf{z}}^{(i)}, q_d)$ among all iterations, say $D_c(\underline{\mathbf{z}}^{*}, q_d)$.

Step 3: If $\text{Rank}(\underline{\mathbf{z}}^{(i)}) = \text{Rank}(\hat{\underline{\mathbf{z}}}^{(i)})$ then go to Step 4; else sort the elements in $\underline{\mathbf{z}}^{(i)}$ into new $\underline{\mathbf{z}}^{(i+1)}$ according to $\text{Rank}(\hat{\underline{\mathbf{z}}}^{(i)})$, set i=i+1, and go to Step1.

Step 4: Obtain the locally optimal $\underline{\mathbf{z}}^{*}$ with the minimal distortion among all iterations; the algorithm is stopped.

The sorting algorithm required in Step 3 results in multiple swaps rather than binary swaps, so EIA is relatively easy and fast. The program for implementing this algorithm is to be found in Appendix B.

Example 3.1: Consider a uniform codebook $\left\{\frac{-n+1}{2}\Delta, \frac{-n+3}{2}\Delta, \cdots, \frac{n-1}{2}\Delta\right\}$ = {-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5} with n=8, step size Δ=1, $\sum_{i=0}^{n-1} z_i = 0$, and $\sum_{i=0}^{n-1} z_i^2 = 42$. The detailed steps in each iteration for the EIA algorithm are presented as below:

Iteration 0:

Step 0: Set i=0. An initial index assignment $\underline{\mathbf{z}}^{(0)} = [3.5,0.5,-1.5,1.5,-0.5,-2.5,-3.5,2.5]^T$ is randomly selected, and its associated rank order is given by $\text{Rank}(\underline{\mathbf{z}}^{T(0)}) = \{7,4,2,5,3,1,0,6\}$. Choose the designated bit error rate q_d=0.01.

Step 1: Regress $\underline{\mathbf{z}}^{(0)}$ on $\mathbf{H}_b$, so $\underline{\mathbf{z}}^{(0)} = \mathbf{H}_b \underline{\boldsymbol{\beta}}^{(0)} + \underline{\boldsymbol{\varepsilon}}$, and compute

i) $\hat{\beta}_i^{(0)} = (\underline{\mathbf{h}}_i^T \underline{\mathbf{z}}^{(0)})/n$,

$$\underline{\hat{\boldsymbol{\beta}}}^{(0)} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 3.5 \\ 0.5 \\ -1.5 \\ 1.5 \\ -0.5 \\ -2.5 \\ -3.5 \\ 2.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.25 \\ 1 \end{bmatrix}.$$

ii) By (3.9) and (3.16), compute $\eta=8*[(-0.5)^2+0.25^2+1^2]=10.5/42=0.25$.

iii) By (3.6), compute $D_c=0.25*(42-(1-0.02)(10.5)-(1-0.02)^2(31)-(1-0.02)^3(0.5))$ $=0.3667$ with $q_d=0.01$.

iv) By (3.15), compute $\underline{\hat{\mathbf{z}}}^{(0)} = \mathbf{H}_b \underline{\hat{\boldsymbol{\beta}}}^{(0)}$,

$$\underline{\hat{\mathbf{z}}}^{(0)} = \begin{bmatrix} \hat{z}_0 \\ \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \\ \hat{z}_5 \\ \hat{z}_6 \\ \hat{z}_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -0.5 \\ 0.25 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 1.75 \\ 0.25 \\ 1.25 \\ -1.25 \\ -0.25 \\ -1.75 \\ -0.75 \end{bmatrix},$$

and its associated rank order is $\text{Rank}(\underline{\hat{\mathbf{z}}}^{(0)})=\{5,7,4,6,1,3,0,2\}$.

Step 2: Record the minimal $D_c=0.3667$ and the associated $\underline{\mathbf{z}}^*=\underline{\mathbf{z}}^{(0)}$.

Step 3: Since $\text{Rank}(\underline{\mathbf{z}}^{(0)}) \neq \text{Rank}(\underline{\hat{\mathbf{z}}}^{(0)})$, i.e. $\{7,4,2,5,3,1,0,6\} \neq \{5,7,4,6,1,3,0,2\}$, sort the elements in $\underline{\mathbf{z}}^{(0)}$ according to $\text{Rank}(\underline{\hat{\mathbf{z}}}^{(0)})=\{5,7,4,6,1,3,0,2\}$, then we obtain the new $\underline{\mathbf{z}}^{(1)}=[1.5,3.5,0.5,2.5,-2.5,-0.5,-3.5,-1.5]^T$. Set i=i+1 and go to Step1.

Iteration 1:

Step 1: Regress $\underline{\mathbf{z}}^{(1)}$ on $\mathbf{H}_b$, and compute

i) $\hat{\beta}_i^{(1)} = (\underline{\mathbf{h}}_i^T \underline{\mathbf{z}}^{(1)})/n$,

$$\underline{\hat{\boldsymbol{\beta}}}^{(1)} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 3.5 \\ 0.5 \\ 2.5 \\ -2.5 \\ -0.5 \\ -3.5 \\ -1.5 \end{bmatrix} = \begin{bmatrix} -1 \\ 0.5 \\ 2 \end{bmatrix}.$$

ii) By (3.9) and (3.16), $\eta = 8*[(-1)^2+0.5^2+2^2] = 42/42 = 1$.

iii) By (3.6), D_c is equal to 0.21

iv) By (3.15), compute $\underline{\hat{\mathbf{z}}}^{(1)} = \mathbf{H}_b \underline{\hat{\boldsymbol{\beta}}}^{(1)}$,

$$\underline{\hat{\mathbf{z}}}^{(1)} = \begin{bmatrix} \hat{z}_0 \\ \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \\ \hat{z}_5 \\ \hat{z}_6 \\ \hat{z}_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ 0.5 \\ 2 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 3.5 \\ 0.5 \\ 2.5 \\ -2.5 \\ -0.5 \\ -3.5 \\ -1.5 \end{bmatrix},$$

and its associated rank order is $\text{Rank}(\underline{\hat{\mathbf{z}}}^{(1)}) = \{5,7,4,6,1,3,0,2\}$.

Step 2: Since 0.3667>0.21, record the minimal D_c=0.21 and the associated $\underline{\mathbf{z}}^* = \underline{\mathbf{z}}^{(1)}$.

Step 3: Since $\text{Rank}(\underline{\mathbf{z}}^{(1)}) = \text{Rank}(\underline{\hat{\mathbf{z}}}^{(1)})$, i.e. $\{5,7,4,6,1,3,0,2\} = \{5,7,4,6,1,3,0,2\}$, STOP in Step 4.

Step 4: Obtain the optimal index assignment with minimal distortion among all iterations, $\underline{\mathbf{z}}^* = [1.5,3.5,0.5,2.5,-2.5,-0.5,-3.5,-1.5]^T$ with D_c=0.21. Then, STOP the algorithm.

Referring to the hamming distance preserving index assignments discussed in Appendix A and the results in Table A.2, the optimal $\underline{\mathbf{z}}^* = \underline{\mathbf{z}}^{(1)}$ with $\text{Rank}(\underline{\mathbf{z}}^{(1)}) = \{5,7,4,6,1,3,0,2\}$ is hamming distance preserving isomorphism to the optimal $\underline{\mathbf{z}}$ with NBC $= \{0,1,2,3,4,5,6,7\}$. This confirms the result proved by McLaughlin, Neuhoff and Ashley [McLaughlin, et. al., 1995], that NBC is an

optimal index assignment for a uniform codebook.

Lemma 3.11: The EIA algorithm must stop in a finite number of iterations.

Proof of Lemma 3.11:

The stopping rule of the EIA algorithm is that the linearity index η converges when $\text{Rank}(\underline{\mathbf{z}}) = \text{Rank}(\hat{\underline{\mathbf{z}}})$. In the Step 3 of the EIA algorithm, a sorting procedure is invoked whenever $\text{Rank}(\underline{\mathbf{z}}) \neq \text{Rank}(\hat{\underline{\mathbf{z}}})$, by Corollary 3.10, η always strictly increases. By equation (3.9), the possible value of η is between 0 and 1. Since there are only a finite number of values of Rank(.) there certainly be a finite number of values of η and η always increases and has an upper bound of 1. Then the EIA algorithm must stop in a finite number of iterations. □

3.5 Sequential Eigenspace Index Assignment Algorithm (SEIA)

For the existing index assignment algorithms, their results depend upon the initial index assignments [Zeger and Gersho, 1990][Knagenhjelm and Agrell, 1996]. Similarly, the EIA algorithm also leads to different locally optimal index assignments for different initial index assignments. We observe that Step 1 in the EIA algorithm uses the $n \times b$ linear matrix $\mathbf{H}_b = [\underline{\mathbf{h}}_0, \cdots, \underline{\mathbf{h}}_{b-1}]$ in the regression. However, due to the special structure of $\mathbf{H}_b$, the EIA algorithm may be sequentially performed in such a way that its final result will be independent of the initial index assignment. Consider the regression model,

$$\underline{\mathbf{z}} = \mathbf{R}\underline{\boldsymbol{\beta}} + \underline{\boldsymbol{\varepsilon}}, \tag{3.18}$$

where $\mathbf{R}$ denotes the regressor matrix. Initially, set the b-th column of $\mathbf{H}_b$ as the initial regressor, say $\mathbf{R} = \underline{\mathbf{h}}_{b-1}$. Then add vectors $\underline{\mathbf{h}}_i$, i=b-2,b-3,...,0 one at a time into $\mathbf{R}$. This modified version of EIA is called the sequential EIA algorithm (or SEIA) and is summarized as follows:

Sequential Eigenspace Index Assignment Algorithm (SEIA):

Input: An initial index assignment vector $\underline{\mathbf{z}}^{(0)}$.

Output: A locally optimal index assignment vector, also named $\underline{\mathbf{z}}^{(i)}$.

Step 0: Set i=0 and j=0. Choose an initial assignment vector $\underline{\mathbf{z}}^{(i)}$; choose a designated bit error rate q_d. Set the initial regressor matrix to be the $(b-j)^{th}$ column of the linear matrix $\mathbf{H}_b$, say $\mathbf{R}^{(j)} = \underline{\mathbf{h}}_{b-j-1}$.

Step 1: Use linear matrix $\mathbf{H}_b$ to compute η, $D_c(\underline{\mathbf{z}}^{(i)},q_d)$. Regress $\underline{\mathbf{z}}^{(i)}$ on the regressor matrix $\mathbf{R}^{(j)}$, and compute the predicted value $\hat{\underline{\mathbf{z}}}^{(i)}$.

Step 2: Record the minimal distortion of $D_c(\underline{\mathbf{z}}^{(i)},q_d)$ among all iterations, say $D_c(\underline{\mathbf{z}}^{*},q_d)$.

Step 3: If $\text{Rank}(\underline{\mathbf{z}}^{(i)}) = \text{Rank}(\hat{\underline{\mathbf{z}}}^{(i)})$ then go to Step 4, else sort the elements in $\underline{\mathbf{z}}^{(i)}$ into new $\underline{\mathbf{z}}^{(i+1)}$ according to $\text{Rank}(\hat{\underline{\mathbf{z}}}^{(i)})$, set i=i+1, and go to Step1.

Step 4: If $\mathbf{H}_b$ is not completely used, then j=j+1 and add vectors $\underline{\mathbf{h}}_{b-j-1}$ into the current regressor matrix $\mathbf{R}^{(j)}$, i.e. $\mathbf{R}^{(j)}=[\ \underline{\mathbf{h}}_{b-j-1}, \mathbf{R}^{(j-1)}]$; set i=i+1, let $\underline{\mathbf{z}}^{(i)} = \underline{\mathbf{z}}^{*}$ and go to Step 1,

else the algorithm is stopped and $\underline{\mathbf{z}}^{*}$ is locally optimal.

The program for implementing this algorithm is to be found in Appendix B.

Example 3.2: The same example as in Example 3.1 is used here to demonstrate the SEIA algorithm. Consider a uniform codebook $\left\{\frac{-n+1}{2}\Delta,\frac{-n+3}{2}\Delta,\cdots,\frac{n-1}{2}\Delta\right\}$ = {-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5} with n=8, step size Δ=1, $\sum_{i=0}^{n-1} z_i = 0$, and $\sum_{i=0}^{n-1} z_i^2 = 42$, the detailed steps of each iteration are summarized as below.

Iteration 0:

Step 0: i=0. j=0. An initial index assignment $\underline{\mathbf{z}}^{(0)}$=[3.5,0.5,-1.5,1.5,-0.5,-2.5, -3.5,2.5]^T is randomly selected, and its associated rank order is given by

Rank($\underline{\mathbf{z}}^{(0)}$)={7,4,2,5,3,1,0,6}. Choose the designated bit error rate q_d=0.01. Set the initial regressor matrix to be $\mathbf{R}^{(j)} = \underline{\mathbf{h}}_{b-j-1}$.

Step 1: Regress $\underline{\mathbf{z}}^{(0)}$ on the regressor matrix $\mathbf{R}^{(0)}$, $\underline{\mathbf{z}}^{(0)} = \mathbf{R}^{(0)}\underline{\boldsymbol{\beta}}^{(0)} + \underline{\boldsymbol{\varepsilon}}$, where $\mathbf{R}^{(0)} = \underline{\mathbf{h}}_{b-1} = \underline{\mathbf{h}}_2$ initially. Compute

i) $\hat{\underline{\boldsymbol{\beta}}}^{(0)} = (\mathbf{H}_b^T \underline{\mathbf{z}}^{(0)})/n$,

$$\hat{\underline{\boldsymbol{\beta}}}^{(0)} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}\begin{bmatrix} 3.5 \\ 0.5 \\ -1.5 \\ 1.5 \\ -0.5 \\ -2.5 \\ -3.5 \\ 2.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.25 \\ 1 \end{bmatrix},$$

ii) By (3.9) and (3.16), η=10.5/42=0.25.

iii) By (3.6), D_c=0.25*[42-((1-0.02)*10.5+(1-0.02)^2*31+(1-0.02)^3*(0.5)] =0.3667 with q_d=0.01.

iv) By (3.15), compute $\hat{\underline{\mathbf{z}}}^{(0)} = \mathbf{R}^{(0)}\hat{\beta}_2^{(0)}$,

$$\hat{\underline{\mathbf{z}}}^{(0)} = \begin{bmatrix} \hat{z}_0 \\ \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \\ \hat{z}_5 \\ \hat{z}_6 \\ \hat{z}_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}[1] = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

and the associated rank order is Rank($\hat{\underline{\mathbf{z}}}^{(0)}$)={4,5,6,7,0,1,2,3} with increasing order used when there are ties.

Step 2: Since D_c=0.3667<initial D_c=∞, record the minimal distortion $D_c(\underline{\mathbf{z}}^*, q_d)$=0.3667 with the associated $\underline{\mathbf{z}}^* = \underline{\mathbf{z}}^{(0)}$.

Step 3: Since Rank($\underline{\mathbf{z}}^{(0)}$) $\neq$ Rank($\hat{\underline{\mathbf{z}}}^{(0)}$), i.e. {7,4,2,5,3,1,0,6} $\neq$ {4,5,6,7,0,1,2,3}, sort

the elements in $\underline{\mathbf{z}}^{(0)}$ according to the Rank($\underline{\hat{\mathbf{z}}}^{(0)}$)={4,5,6,7,0,1,2,3}, then we obtain the new $\underline{\mathbf{z}}^{T(1)}$ =[0.5,1.5,2.5,3.5,-3.5,-2.5,-1.5,-0.5]. Set i=i+1 and go to Step1.

Iteration 1:

Step 1: Regress $\underline{\mathbf{z}}^{(1)}$ on the regressor matrix $\mathbf{R}^{(0)}$, $\underline{\mathbf{z}}^{(1)} = \mathbf{R}^{(0)}\underline{\boldsymbol{\beta}}^{(1)} + \underline{\boldsymbol{\varepsilon}}$, where $\mathbf{R}^{(0)} = \underline{\mathbf{h}}_2$. Compute

i) $\underline{\hat{\boldsymbol{\beta}}}^{(1)} = (\mathbf{H}_b^T \underline{\mathbf{z}}^{(1)})/n$,

$$\underline{\hat{\boldsymbol{\beta}}}^{(1)} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}\begin{bmatrix} 0.5 \\ 1.5 \\ 2.5 \\ 3.5 \\ -3.5 \\ -2.5 \\ -1.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1 \\ 2 \end{bmatrix}$$

ii) By (3.9) and (3.16), η=42/42.

iii) By (3.6), D_c=0.25*[42-((1-0.02)*42+(1-0.02)^2*0+(1-0.02)^3*0]=0.21.

iv) By (3.15), compute $\underline{\hat{\mathbf{z}}}^{(1)} = \mathbf{R}^{(0)}\hat{\beta}_2^{(1)}$,

$$\underline{\hat{\mathbf{z}}}^{(1)} = \begin{bmatrix} \hat{z}_0 \\ \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \\ \hat{z}_5 \\ \hat{z}_6 \\ \hat{z}_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}[2] = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \\ -2 \\ -2 \\ -2 \\ -2 \end{bmatrix},$$

and the associated rank order is Rank($\underline{\hat{\mathbf{z}}}^{(1)}$)={4,5,6,7,0,1,2,3}.

Step 2: Since D_c=0.21<D_c=0.3667, record the minimal distortion $D_c(\underline{\mathbf{z}}^*, q_d)$=0.3667 with the associated $\underline{\mathbf{z}}^* = \underline{\mathbf{z}}^{(1)}$.

Step 3: Since $\text{Rank}(\underline{z}^{(1)}) = \text{Rank}(\underline{\hat{z}}^{(1)})$, i.e. {4,5,6,7,0,1,2,3}={4,5,6,7,0,1,2,3}, go to Step 4.

Step 4: Since $\mathbf{H}_b$ is not completely used, add the vector $\underline{h}_1$ into the current regressor matrix $\mathbf{R}^{(0)}$, then $\mathbf{R}^{(1)} = [\underline{h}_1, \ \underline{h}_2]$; i=i+1; $\underline{z}^{(2)} = \underline{z}^*$; go to Step 1.

Iteration 2:

Step 1: Regress $\underline{z}^{(2)}$ on the regressor matrix $\mathbf{R}^{(1)}$, and compute

i) $\underline{\hat{\beta}}^{(2)} = (\mathbf{H}_b^T \underline{z}^{(2)})/n$

$$\underline{\hat{\beta}}^{(2)} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}\begin{bmatrix} 0.5 \\ 1.5 \\ 2.5 \\ 3.5 \\ -3.5 \\ -2.5 \\ -1.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1 \\ 2 \end{bmatrix},$$

ii) η=42/42.

iii) D_c=0.25*[42-((1-0.02)*42+(1-0.02)^2*0+(1-0.02)^3*0]=0.21.

iv) Compute $\underline{\hat{z}}^{(2)} = \mathbf{R}^{(1)}\underline{\hat{\beta}}^{(2)}$,

$$\underline{\hat{z}}^{(2)} = \begin{bmatrix} \hat{z}_0 \\ \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \\ \hat{z}_5 \\ \hat{z}_6 \\ \hat{z}_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ -1 & -1 \\ -1 & -1 \end{bmatrix}\begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 3 \\ 3 \\ -3 \\ -3 \\ -1 \\ -1 \end{bmatrix},$$

and the associated ranking is $\text{Rank}(\underline{\hat{z}}^{(2)})$={4,5,6,7,0,1,2,3}.

Step 2: Since minimal distortion $D_c(\underline{z}^*, q_d)$=0.21 does not change, the associated $\underline{z}^* = \underline{z}^{(1)}$ does not change.

Step 3: Since $\text{Rank}(\underline{\mathbf{z}}^{(2)}) = \text{Rank}(\underline{\hat{\mathbf{z}}}^{(2)})$, i.e. {4,5,6,7,0,1,2,3}={4,5,6,7,0,1,2,3}, go to Step 4.

Step 4: Since $\mathbf{H}_b$ is not completely used, add the vector $\underline{\mathbf{h}}_0$ into the current regressor matrix $\mathbf{R}^{(2)}$, then $\mathbf{R}^{(2)} = [\underline{\mathbf{h}}_1, \underline{\mathbf{h}}_1, \underline{\mathbf{h}}_2]$; i=i+1; $\underline{\mathbf{z}}^{(3)} = \underline{\mathbf{z}}^*$; go to Step 1.

Iteration 3:

Step 1: Regress $\underline{\mathbf{z}}^{(3)}$ on the regressor matrix $\mathbf{R}^{(2)}$, and compute

i) $\underline{\hat{\boldsymbol{\beta}}}^{(3)} = (\mathbf{H}_b^T \underline{\mathbf{z}}^{(3)})/n$

$$\underline{\hat{\boldsymbol{\beta}}}^{(3)} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}\begin{bmatrix} 0.5 \\ 1.5 \\ 2.5 \\ 3.5 \\ -3.5 \\ -2.5 \\ -1.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1 \\ 2 \end{bmatrix},$$

ii) η=42/42.

iii) D_c=0.25*[42-((1-0.02)*42+(1-0.02)^2*0+(1-0.02)^3*0]=0.21.

iv) Compute $\underline{\hat{\mathbf{z}}}^{(3)} = \mathbf{R}^{(2)}\underline{\hat{\boldsymbol{\beta}}}^{(3)}$,

$$\underline{\hat{\mathbf{z}}}^{(3)} = \begin{bmatrix} \hat{z}_0 \\ \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \\ \hat{z}_5 \\ \hat{z}_6 \\ \hat{z}_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}\begin{bmatrix} -0.5 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1.5 \\ 2.5 \\ 3.5 \\ -3.5 \\ -2.5 \\ -1.5 \\ -0.5 \end{bmatrix},$$

and the associated ranking is $\text{Rank}(\underline{\hat{\mathbf{z}}}^{(3)}) = \{4,5,6,7,0,1,2,3\}$.

Step 2: Since minimal distortion $D_c(\underline{\mathbf{z}}^*, q_d)$=0.21 does not change, the associated

$\underline{\mathbf{z}}^* = \underline{\mathbf{z}}^{(1)}$ does not change.

Step 3: Since $\text{Rank}(\underline{\mathbf{z}}^{(3)}) = \text{Rank}(\underline{\hat{\mathbf{z}}}^{(3)})$, i.e. {4,5,6,7,0,1,2,3}={4,5,6,7,0,1,2,3}, go to Step 4.

Step 4: Since $\mathbf{R}^{(2)}=\mathbf{H}_b$ is completely used, STOP. The optimal $\underline{\mathbf{z}}^* = \underline{\mathbf{z}}^{(1)}=[0.5,1.5,2.5,3.5,-3.5,-2.5,-1.5,-0.5]^T$ and the minimum of D_c is 0.21.

Although the optimal assignment vector with index set {4,5,6,7,0,1,2,3} obtained from SEIA differs from that with index set {5,7,4,6,1,3,0,2} obtained from EIA, they are indeed hamming distance preserving isomorphism. Note that hamming distance preserving index assignment is discussed in Appendix A and the results are in Table A.2. In general SEIA needs more iterations to obtain an optimal assignment vector than EIA does, but it is independent of an initial index assignment. The proof is in the following Lemma.

Lemma 3.12: The SEIA algorithm is independent of an initial index assignment.

Proof of Lemma 3.12: In the SEIA algorithm, the initial regressor is $\underline{\mathbf{h}}_{b-1}^T = [1,...,1,-1,...,-1]$ as in (3.12), in which the first half entries are 1's, and all the others are –1's. For any initial index assignment vector $\underline{\mathbf{z}}$, the regression model in (3.18) has $\hat{\beta}_{b-1} = (\underline{\mathbf{h}}_{b-1}^T \underline{\mathbf{z}})/n$, and $\underline{\hat{\mathbf{z}}}^T = \underline{\mathbf{h}}_{b-1}\hat{\beta}_{b-1} = [\hat{\beta}_{b-1},...,\hat{\beta}_{b-1},-\hat{\beta}_{b-1},...,-\hat{\beta}_{b-1}]$. After sorting the elements of $\underline{\mathbf{z}}$ in accordance with the ranking in $\underline{\hat{\mathbf{z}}}$, the new $\underline{\mathbf{z}}$ will be in one of the two following cases:

(1) When $\hat{\beta}_{b-1} < 0$, then the ranking in $\underline{\hat{\mathbf{z}}}$ is in ascending order, and the ranking of the new $\underline{\mathbf{z}}$ is given by $\text{Rank}(\underline{\mathbf{z}})=\{0,1,\ldots,n-1\}$ which is the same as that of natural binary code (NBC). The associated rank order is in increasing order when they are ties. From now on, all the permutations give new rank followed the NBC are represented as $\text{Rank}(\underline{\mathbf{z}}_{NBC})=\{r_0,r_1,..,r_{(n/2)-1},r_{n/2},r_{(n/2)+1},\ldots,r_{n-1}\}$, $r_i=0,\ldots,n-1$. $\hat{\beta}_i$ is represented as $\hat{\beta}_{i(NBC)}$, $i=0,1,\ldots,b-1$. $\underline{\hat{\mathbf{z}}}$ is represented as $\underline{\hat{\mathbf{z}}}_{(NBC)}$.

(2) When $\hat{\beta}_{b-1} > 0$, then the ranking in $\underline{\hat{\mathbf{z}}}$ is in descending order, and the ranking of new $\underline{\mathbf{z}}$ is given by $\text{Rank}(\underline{\mathbf{z}})=\{n/2,\ldots,n-1,0,\ldots,n/2-1\}$ whose binary codes is equal

to NBC by adding (1 0 ... 0) bit-wisely. We call this code NNBC. From now on, all the permutations give new rank followed the NNBC are represented as Rank($\underline{z}_{NNBC}$). $\hat{\beta}_i$ is represented as $\hat{\beta}_{i(NNBC)}$, i=0,1,...,b-1. $\hat{\underline{z}}$ is represented as $\hat{\underline{z}}_{(NBC)}$.

Therefore, for any initial index assignment vector $\underline{z}$, the new $\underline{z}$ is either NBC or NNBC after one iteration. In SEIA, $\hat{\underline{\beta}} = (\mathbf{H}_b^T \underline{z})/n$ in (3.13) and

$$\mathbf{H}_b = [\underline{\mathbf{h}}_0, ..., \underline{\mathbf{h}}_{b-2}, \underline{\mathbf{h}}_{b-1}] = \begin{bmatrix} \mathbf{1} & . & . & \mathbf{1} & \mathbf{1} \\ \mathbf{-1} & . & . & . & . \\ . & . & . & . & . \\ . & . & . & \mathbf{1} & . \\ . & . & . & \mathbf{-1} & . \\ . & . & . & . & . \\ \mathbf{1} & . & . & . & . \\ \mathbf{-1} & . & . & \mathbf{-1} & \mathbf{1} \\ 1 & . & . & 1 & -1 \\ -1 & . & . & . & . \\ . & . & . & . & . \\ . & . & . & 1 & . \\ . & . & . & -1 & . \\ . & . & . & . & . \\ 1 & . & . & . & . \\ -1 & . & . & -1 & -1 \end{bmatrix}.$$

$\hat{\underline{z}} = \mathbf{R}^{(j)} \hat{\underline{\beta}}$ in (3.18), where $\mathbf{R}^{(j)}$=[$\underline{\mathbf{h}}_{b-j-1}$, $\mathbf{R}^{(j-1)}$] with $\mathbf{R}^{(0)} = \underline{\mathbf{h}}_{b-1}$ and $\mathbf{R}^{(b-1)} = \mathbf{H}_b$. Sort the elements of $\underline{z}$ in accordance with the ranking in $\hat{\underline{z}}$.

As $\mathbf{R}^{(1)} = [\underline{\mathbf{h}}_{b-2}, \underline{\mathbf{h}}_{b-1}]$, since $\hat{\underline{\beta}} = (\mathbf{R}^{T(1)} \underline{z})/n$ and let $\underline{z}_{NBC} = [z_0, z_1, ..., z_{n-1}]$,

$$\hat{\beta}_{b-1(NBC)} = (z_0+z_1+...+z_{(n/2)-2}+z_{(n/2)-1}) - (z_{n/2}+z_{(n/2)+1}+...+z_{n-1}) \text{ and}$$

$$\hat{\beta}_{b-1(NNBC)} = (z_{n/2}+z_{(n/2)+1}+...+z_{n-1}) - (z_0+z_1+...+z_{(n/2)-2}+z_{(n/2)-1}).$$

Thus, $\hat{\beta}_{b-1(NNBC)} = -\hat{\beta}_{b-1(NBC)}$. Also,

$$\hat{\beta}_{b-2(NBC)}=(z_0+\ldots+z_{(n/4)-1})-(z_{(n/4)}\ldots+\quad z_{(n/2)-1})+(z_{(n/2)}+\ldots+z_{(3n/4)-1})-(z_{3n/4}+\ldots+z_{n-1})$$

and $\hat{\beta}_{b-2(NNBC)}=(z_{(n/2)}+\ldots+z_{(3n/4)-1})-(z_{3n/4}+\ldots+z_{n-1})+(z_0+\ldots+z_{(n/4)-1})-(z_{(n/4)}\ldots+z_{(n/2)-1})$.

Thus, $\hat{\beta}_{b-2(NNBC)}=\hat{\beta}_{b-2(NBC)}$.

Let $\hat{\beta}_{b-2(NBC)}=a_{b-2}\in R$ and $\hat{\beta}_{b-1(NBC)}=a_{b-1}\in R$.

$$\underline{\hat{\mathbf{z}}}^{T}_{(NBC)}=\begin{bmatrix}1 & 1\\ \vdots & \vdots\\ 1 & 1\\ -1 & 1\\ \vdots & \vdots\\ -1 & 1\\ 1 & -1\\ \vdots & \vdots\\ 1 & -1\\ -1 & -1\\ \vdots & \vdots\\ -1 & -1\end{bmatrix}\begin{bmatrix}a_{b-2}\\ a_{b-1}\end{bmatrix}=\begin{bmatrix}a_{b-2}+a_{b-1}\\ \vdots\\ a_{b-2}+a_{b-1}\\ -a_{b-2}+a_{b-1}\\ \vdots\\ -a_{b-2}+a_{b-1}\\ a_{b-2}-a_{b-1}\\ \vdots\\ a_{b-2}-a_{b-1}\\ -a_{b-2}-a_{b-1}\\ \vdots\\ -a_{b-2}-a_{b-1}\end{bmatrix},\ \text{and}$$

$$\underline{\hat{\mathbf{z}}}^{T}_{(NNBC)}=\begin{bmatrix}1 & 1\\ \vdots & \vdots\\ 1 & 1\\ -1 & 1\\ \vdots & \vdots\\ -1 & 1\\ 1 & -1\\ \vdots & \vdots\\ 1 & -1\\ -1 & -1\\ \vdots & \vdots\\ -1 & -1\end{bmatrix}\begin{bmatrix}a_{b-2}\\ -a_{b-1}\end{bmatrix}=\begin{bmatrix}a_{b-2}-a_{b-1}\\ \vdots\\ a_{b-2}-a_{b-1}\\ -a_{b-2}-a_{b-1}\\ \vdots\\ -a_{b-2}-a_{b-1}\\ a_{b-2}+a_{b-1}\\ \vdots\\ a_{b-2}+a_{b-1}\\ -a_{b-2}+a_{b-1}\\ \vdots\\ -a_{b-2}+a_{b-1}\end{bmatrix}$$

Sort the elements of $\underline{\mathbf{z}}$ in accordance with the ranking in $\underline{\hat{\mathbf{z}}}$. Let the rank order of $\underline{\mathbf{z}}$ with NBC is $\text{Rank}(\underline{\mathbf{z}}_{NBC})=\{r_0,r_1,..,r_{(n/2)-1},r_{n/2},r_{(n/2)+1},\ldots,r_{n-1}\}$, it is easy to see

$\text{Rank}(\underline{\mathbf{z}}_{\text{NNBC}}) = \{r_{n/2}, r_{(n/2)+1}, \ldots, r_{n-1}, r_0, r_1, .., r_{(n/2)-1}\}$.

In general, $\text{Rank}(\underline{\mathbf{z}}_{\text{NBC}}) = \{r_0, r_1, .., r_{(n/2)-1}, r_{n/2}, r_{(n/2)+1}, \ldots, r_{n-1}\}$ and $\text{Rank}(\underline{\mathbf{z}}_{\text{NNBC}}) = \{r_{n/2}, r_{(n/2)+1}, \ldots, r_{n-1}, r_0, r_1, .., r_{(n/2)-1}\}$ is true for all $\mathbf{R}^{(j)} = [\ \underline{\mathbf{h}}_{b-j-1}, \mathbf{R}^{(j-1)}]$, j=1,…,b-1. Since the first half entries of $\underline{\mathbf{h}}_i$ is the same as the second half entries of $\underline{\mathbf{h}}_i$ for all i=0,…,b-2 and $\hat{\underline{\boldsymbol{\beta}}} = (\mathbf{H}_b^T \underline{\mathbf{z}})/n$ in (3.13), $\hat{\beta}_{i(\text{NNBC})} = \hat{\beta}_{i(\text{NBC})}$ for all i=0,..,b-2. Since the second half entries of $\underline{\mathbf{h}}_{b-1}$ are the same as the second half entries of $\underline{\mathbf{h}}_{b-1}$ with sign changes and $\hat{\underline{\boldsymbol{\beta}}} = (\mathbf{H}_b^T \underline{\mathbf{z}})/n$, $\hat{\beta}_{b\text{-}1(\text{NNBC})} = -\hat{\beta}_{b\text{-}1(\text{NBC})}$. The values of all components in $\hat{\underline{\mathbf{z}}}_{(\text{NBC})} = \mathbf{R}^{(j)} \hat{\underline{\boldsymbol{\beta}}}_{(\text{NBC})}$ are equal to the values of all components in $\hat{\underline{\mathbf{z}}}_{(\text{NNBC})} = \mathbf{R}^{(j)} \hat{\underline{\boldsymbol{\beta}}}_{(\text{NNBC})}$ but with the first half and the second half exchanged. This is the initial regressor $\underline{\mathbf{h}}_{b-1}^T = [1,...,1,-1,...,-1]$, in which the first half entries are 1's, and all the others are –1's. Thus, the index sets followed NNBC are equal to the index sets followed NBC for all iterations when (1 0 … 0) is added bit-wisely. This addition is hamming distance preserving index for all iterations. This proves that SEIA is independent of an initial index assignment. □

3.6 Numerical Results

In this section, two codebooks are used to show the performances of EIA and SEIA algorithms. Firstly, APDS codebooks are used to show the multiple swaps and nearly global optimality of the EIA algorithm (Examples 3.3 and 3.4). Secondly, Examples 3.5 to 3.8 take the real world case of the voice digitization in North American Telephone Systems (CCITT) [Gersho and Gray, 1992] to illustrate the properties of both EIA and SEIA algorithms.

An antipodal direct sum codebook (APDS) introduced in Chapter 2 [McLaughlin, et al., 1995] has codewords in the format of $\{\pm c_0 \pm c_1 \pm ... \pm c_{b-1}\}$, where c_i's are distinct real numbers. The codewords can be rearranged in a format

of $\underline{z} = \mathbf{H}_b \underline{c} = \sum_{j=0}^{b-1} c_j \underline{h}_j$, where $\underline{h}_j$'s are eigenvectors spanning eigenspace E_l. Lemma 3.7 proved that, for an APDS, there is an optimal index assignment whose linearity index $\eta = \|Proj_l \underline{z}\|^2 / \|\underline{z}\|^2$ is equal to one. Also, from Lemma 3.5, the global minimum channel distortion is equal to $D_{c\min} = 4q\|\underline{z}\|^2 / n$. Since the optimal solution is known for an APDS, it will serve as a nice example to check the performance of the proposed algorithms.

Example 3.3: (Multiple Swaps)

The same antipodal direct sum codebook as in Example 3.1 is taken here. The initial index assignment vector used there was randomly selected and its performance is not good, however it reaches the global optimal index assignment after one iteration. In Table 3.3, comparing iteration 0 and 1, we notice that -3.5 is the only element which remains in the same position, and all the others are moved to different positions. It shows that the EIA algorithm results in multiple swaps rather than binary swaps, and hence is very efficient.

Table 3.3: The Multiple Swaps of EIA for an Antipodal Direct Sum Codebook.

Itrn	$\underline{z}$								η
0	3.5	0.5	-1.5	1.5	-0.5	-2.5	-3.5	2.5	0.25
1	1.5	3.5	0.5	2.5	-2.5	-0.5	-3.5	-1.5	1.00

Example 3.4: (Nearly Global Optimum)

Now consider an arbitrary APDS of size 256 with the coefficients c_i's being equal to (14.77,6.01,4.24,5.49,4.61,2.51,-2.32,12.39). A natural binary code (NBC) and a random code (RC) are used as the initial index assignment vectors. Let the designated bit error rate q_d=0.001, then D_{cmin}=4*0.001*488.837=1.9553. The EIA algorithm is then applied to find a locally optimal index assignment. Table 3.4 lists η's and D_c's for all iterations. The details of the EIA algorithm are summarized as below:

(1) Both NBC and RC initial codes give good performances, i.e., both

$\eta_1 = 0.999191$ and $\eta_2 = 0.999254$ are very close to the global maximum value 1; and the channel distortions $D_{c1} = 1.959794$ and $D_{c2} = 1.95909$ are also very close to the global minimum value D_{cmin}=1.9553. In term of channel distortion, NBC and RC initial codes are only 0.2% and 0.19% worse than the global minimum value, respectively.

(2) The assignment vector will be very close to E_1 after the first iteration, but only minor improvement is obtained after the second iteration. For example, the initial assignment vector for random code has large $D_{c2} = 6.1712$, and it is reduced to $D_{c2} = 1.9616$ in iteration 1. Only minor reductions for channel distortion are observed from iterations 2 to 12.

Table 3.4: The Performances of EIA Using Natural Binary Code and Random Code as Initials Choices for an Antipodal Direct Sum Codebook with n=256 and q_d=0.001.

Natural Binary Code			Random Code		
Itrn	η_1	D_{c1}	Itrn	η_2	D_{c2}
0	0.98781	2.0046	0	0.29410	6.1712
1	0.99214	1.9873	1	0.99866	1.9616
2	0.99455	1.9778	2	0.99881	1.9610
3	0.99600	1.9723	3	0.99891	1.9606
4	0.99703	1.9683	4	0.99899	1.9603
5	0.99772	1.9656	5	0.99908	1.9598
6	0.99814	1.9638	6	0.99911	1.9597
7	0.99842	1.9627	7	0.99912	1.95968
8	0.99860	1.9619	8	0.99917	1.9595
9	0.99876	1.9613	9	0.99922	1.9593
10	0.99892	1.9609	10	0.99924	1.9592
11	0.99902	1.9605	11	0.999251	1.95911
12	0.99905	1.9604	12	0.999254	1.95909
13	0.99909	1.9602			
14	0.99912	1.9601			
15	0.99915	1.96000			
16	0.99916	1.95995			
17	0.999175	1.95990			
18	0.999184	1.95986			
19	0.999191	1.959794			
20	0.999193	1.959799			

The remaining examples use a different real data set. The Voice Digitization in North American Telephone Systems (CCITT) [Gersho and Gray, 1992] is used to show some properties of both the EIA and SEIA algorithms in examples 3.5 to 3.8. Recall from Chapter 2, CCITT is a symmetric piecewise uniform quantizer with 8 bits, 8 positive segments, increasing in length by a factor of 2 for each successive segment in order of increasing amplitude, and with 16 steps on each segment. The output formula and levels are listed in the Table 2.1.

Example 3.5.1: (EIA may be terminated earlier)

Knagenhjelm and Agrell [Knagenhjelm and Agrell, 1996] pointed out the strong negative correlation –0.846 between D_c and η from 100,000 samples for 4-bit codebook with q=0.01. However, D_c may not be strictly monotonic decreasing in η. Empirically, Table 3.5.1 shows a typical pattern such that the values of D_c decrease initially and thereafter increase for large values of η with a few minor exceptions (for instances, iterations 17 and 37). In other words, D_c is almost concave in η and D_c=33848 is minimum when η=0.8438. Therefore, in order to save computational effort for this particular case, the EIA algorithm can be easily modified so that it terminates whenever D_c increases.

TABLE 3.5.1: The Performances of EIA using NBC for CCITT Example with q_d=0.001.

itrn	η	Dc	itrn	η	Dc
0	0.8177	35055		:	:
1	0.8360	34108	16	0.8537	34799
2	0.8415	33880	17	0.8538	34794
3	**0.8438**	**33848**	.	:	:
4	0.8454	33900	36	0.857703	35368.4
5	0.8471	33979	37	0.857705	35368.2

Example 3.5.2: (EIA depends upon an initial index assignment)

The EIA algorithm is initialized by five codeword mappings: MDC, FBC, NBC, and two random codes RC_1 and RC_2. Let the designated bit error rate be equal to q_d=0.001. Table 3.5.2 shows the performances of the five different initial codes for

the EIA algorithm, and the findings are summarized as below:

(1) Comparing different codes, the initial channel distortion D_c of NBC (35055) is better than that of FBC (47507) or MDC (53891).

(2) At the first iteration, both MDC and FBC reduce to the same D_c=35055 and η=0.8177 as the initial NBC does. Thus, these three initial codes lead to the same optimal results of D_c=33848 and η=0.8438.

(3) For the index assignment which is locally a minimum, the distortion is reduced to D_{c4}=34330 for RC_1 and to D_{c5}=33816 for RC_2. Since D_{c5}=33816<33848< D_{c4}=34330, it shows that neither NBC nor random codes will outperform each other. Hence, the EIA algorithm depends upon the initial code. Moreover, the EIA significantly improves the performance of channel distortion. For instance, EIA reduces the distortion by 58% and 65% compared with that of RC_1 and RC_2 initial codes, respectively.

(4) Since only a few iterations are required for getting the optimal index assignment, the EIA algorithm is shown to be a very efficient algorithm. In this example, no more than 5 iterations were required.

TABLE 3.5.2: The Performances of EIA using Five Different Initial Codes for CCITT Example with q_d=0.001.

	MDC		FBC		NBC		RC_1		RC_2	
Itrn	η_1	D_{c1}	η_2	D_{c2}	η_3	D_{c3}	η_4	D_{c4}	η_5	D_{c5}
0	0.3489	53891	0.3489	47507	*0.8177*	*35055*	0.2721	81453	0.1161	96045
1	*0.8177*	*35055*	*0.8177*	*35055*	*0.8360*	*34108*	**0.8481**	**34330**	**0.8434**	**33816**
2	*0.8360*	*34108*	*0.8360*	*34108*	*0.8415*	*33880*	0.8500	34432	0.8452	33852
3	*0.8415*	*33880*	*0.8415*	*33880*	***0.8438***	***33848***				
4	***0.8438***	***33848***	***0.8438***	***33848***	*0.8454*	*33900*				
5	*0.8454*	*33900*	*0.8454*	*33900*						

Example 3.6: (SEIA is independent of an initial index assignment)

The SEIA algorithm is also initialized by five different codeword mappings: MDC, FBC, NBC, and two random codes RC_3 and RC_4. Let the designated bit error

rate be equal to q_d=0.001. Table 3.6 shows the performances of these five initial codes, and the findings are summarized as below:

(1) In the first iteration, all the other four initial codes except NBC reach η=0.8177 and D_c=35054, which is the result of using NBC. All five initial codes have the same final locally optimal index assignment with η=0.8435 and D_c=33823. This supports Lemma 3.12 that the SEIA algorithm is independent of the initial code.

(2) The resulting D_c (33823) is almost the same as the best one (33816) which is the best result of 52 simulation studies with different random codes in Example 3.5. Hence the performance of the SEIA algorithm is quite good.

(3) Although more iterations are required for SEIA compared to EIA, it is still very efficient. It typically takes only 0.91 CPU seconds for EIA and 1.9 CPU seconds for SEIA on a SUN Sparc 20 Workstation.

TABLE 3.6: The Performances of SEIA Using Five Different Initial Codes for CCITT Example and q_d=0.001.

	MDC		FBC		NBC		RC_3		RC_4	
itrn	η_1	D_{c1}	η_2	D_{c2}	η_3	D_{c3}	η_4	D_{c4}	η_5	D_{c5}
0	0.3489	53891	0.3489	47507	*0.8177*	*35055*	0.2721	81453	0.1970	90618
1	*0.8177*	*35055*	*0.8177*	*35055*	*0.8225*	*34803*	*0.8177*	*35055*	*0.8177*	*35055*
2	*0.8225*	*34803*	*0.8225*	*34803*	:	:	*0.8225*	*34803*	*0.8225*	*34803*
:	:	:	:	:	:	:	:	:	:	:
13	:	:	:	:	***0.8435***	***33823***	:	:	:	:
14	***0.8435***	***33823***	***0.8435***	***33823***	*0.8448*	*33862*	***0.8435***	***33823***	***0.8435***	***33823***
15	*0.8448*	*33862*	*0.8448*	*33862*			*0.8448*	*33862*	*0.8448*	*33862*

Example 3.7: (Comparisons among EIA, SEIA, and NBC)

Since NBC is the most popular code, we are interested in comparing performances of both EIA and SEIA against NBC by using the CCITT example with q_d=0.01. Table 3.7.1 lists the channel distortions of the locally optimal index assignments obtained from EIA and SEIA and that for NBC for actual q's ranging from 0 to 0.5. The results are summarized as below:

(1) In term of channel distortion, both EIA and SEIA algorithms outperform NBC for $0 < q < 0.5$. For instance, when q=0.001, SEIA outperforms NBC by

$\delta_1 = 3.458\%$ while EIA outperforms NBC by $\delta_2 = 3.441\%$, where δ_1=100%*(NBC-SEIA)/NBC and δ_2=100%*(NBC-EIA)/NBC.

(2) SEIA slightly outperforms EIA when $q \leq 0.05$. For example, the channel distortion of SEIA is 80, or 0.0005% less than that of EIA when q=0.05. In other words, the difference between SEIA and EIA is almost indistinguishable.

Table 3.7.1: The Comparisons of Channel Distortions among NBC, SEIA and EIA for CCIT example with q_d=0.01.

q	SEIA	EIA	SEIA-EIA	NBC	δ_1	δ_2
5.00E-01	1.28471E+07	1.28471E+07	0.00	1.28471E+07	0.000	0.000
4.00E-01	1.06640E+07	1.06634E+07	687.00	1.07274E+07	0.590	0.597
3.00E-01	8.38567E+06	8.38468E+06	994.00	8.49541E+06	1.292	1.303
2.00E-01	5.91551E+06	5.91477E+06	736.00	6.03884E+06	2.042	2.054
1.00E-01	3.15454E+06	3.15441E+06	126.00	3.24485E+06	2.783	2.787
5.00E-02	1.63307E+06	1.63315E+06	-80.00	1.68590E+06	3.134	3.129
1.00E-02	3.36190E+05	3.36239E+05	-49.00	3.48022E+05	3.400	3.386
5.00E-03	1.68714E+05	1.68741E+05	-27.00	1.74710E+05	3.432	3.417
1.00E-03	3.38426E+04	3.38484E+04	-5.90	3.50546E+04	3.458	3.441
7.50E-04	2.53866E+04	2.53910E+04	-4.40	2.62962E+04	3.459	3.442
5.00E-04	1.69275E+04	1.69305E+04	-2.90	1.75344E+04	3.461	3.444
2.50E-04	8.46533E+03	8.46681E+03	-1.48	8.76894E+03	3.462	3.445
1.00E-04	3.38651E+03	3.38710E+03	-0.59	3.50800E+03	3.463	3.446
7.50E-05	2.53992E+03	2.54037E+03	-0.45	2.63105E+03	3.464	3.447
5.00E-05	1.69330E+03	1.69361E+03	-0.30	1.75407E+03	3.465	3.447
2.50E-05	8.46673E+02	8.46822E+02	-0.15	8.77053E+02	3.464	3.447
1.00E-05	3.38673E+02	3.38733E+02	-0.06	3.50825E+02	3.464	3.447
7.50E-06	2.54005E+02	2.54050E+02	-0.05	2.63119E+02	3.464	3.447
5.00E-06	1.69337E+02	1.69367E+02	-0.03	1.75413E+02	3.464	3.447
2.50E-06	8.46687E+01	8.46837E+01	-0.01	8.77068E+01	3.464	3.447
1.00E-06	3.38675E+01	3.38735E+01	-0.01	3.50828E+01	3.464	3.447

Note: δ_1=100%*(NBC-SEIA)/NBC, δ_2=100%*(NBC-EIA)/NBC.

Tables 3.7.2 and 3.7.3 list the 256 codewords of locally optimal index assignment vectors for EIA and SEIA algorithms using NBC as an initial code, respectively. There the bold case highlights differences in the order between EIA and SEIA. Both tables list codewords z_i by rows with index i=0 to 255 in the binary form of i_1 denoting the first 5 digits and i_2 denoting the last 3 digits. In Table 3.7.2, –8031 is the value of codeword z_0, -7775 is the value of codeword z_1, and 8031 is the value

of codeword z_{255}, etc.

Table 3.7.2: The Locally Optimal Index Assignment of EIA for CCITT in Which Codewords Are Arranged by Rows from z_0 to z_{255}. (i_1 - First 5 Digits of Binary Index of z_i; i_2 - Last 3 Digits of Binary Index of z_i.)

z_i: i_2 / i_1	000	001	010	011	100	101	110	111
00000	-8031	-7775	-7519	-7263	-7007	-6751	-6495	-6239
00001	-5983	-5727	-5471	-5215	-4959	-4703	-4447	**-3999**
00010	**-4191**	-3871	-3743	-3615	-3487	-3359	-3231	-3103
00011	-2975	**-2719**	**-2463**	**-2207**	-1983	**-1855**	**-1663**	**-1407**
00100	**-2847**	**-2591**	**-2335**	-2079	**-1919**	**-1791**	**-1599**	**-1343**
00101	**-1151**	**-975**	**-879**	**-783**	**-687**	**-591**	**-495**	-423
00110	**-439**	**-375**	**-327**	**-279**	**-231**	**-203**	**-171**	**-139**
00111	**-115**	**-93**	**-81**	**-69**	**-57**	**-45**	**-30**	**-20**
01000	-1727	**-1471**	**-1215**	**-1023**	**-911**	**-815**	**-719**	**-623**
01001	**-559**	**-455**	**-391**	**-343**	**-295**	**-247**	**-211**	**-179**
01010	**-195**	**-155**	**-123**	**-99**	**-85**	**-73**	**-61**	**-49**
01011	**-41**	**-24**	**-14**	**-6**	**0**	**8**	**16**	**26**
01100	**-37**	**-22**	**-12**	**-4**	**2**	**10**	**18**	**28**
01101	**33**	**53**	**65**	**77**	**89**	**107**	**131**	**163**
01110	**147**	**187**	**219**	**263**	**311**	**359**	**407**	**471**
01111	**527**	**655**	**751**	847	**943**	1087	**1279**	**1535**
10000	**-1535**	**-1279**	-1087	**-943**	-847	**-751**	**-655**	**-527**
10001	**-471**	**-407**	**-359**	**-311**	**-263**	**-219**	**-187**	**-147**
10010	**-163**	**-131**	-107	**-89**	-77	**-65**	**-53**	**-33**
10011	**-28**	**-18**	**-10**	**-2**	**4**	**12**	**22**	**37**
10100	**-26**	**-16**	**-8**	**0**	**6**	**14**	**24**	**41**
10101	**49**	**61**	**73**	**85**	**99**	**123**	**155**	**195**
10110	**179**	**211**	**247**	**295**	**343**	**391**	**455**	**559**
10111	**623**	**719**	**815**	**911**	**1023**	**1215**	**1471**	1727
11000	**20**	**30**	**45**	**57**	**69**	**81**	**93**	**115**
11001	**139**	**171**	**203**	**231**	**279**	**327**	**375**	**439**
11010	423	**495**	**591**	**687**	**783**	**879**	**975**	**1151**
11011	**1343**	**1599**	**1791**	**1919**	2079	**2335**	**2591**	**2847**
11100	**1407**	**1663**	**1855**	1983	**2207**	**2463**	**2719**	2975
11101	3103	3231	3359	3487	3615	3743	3871	**4191**
11110	**3999**	4447	4703	4959	5215	5471	5727	5983
11111	6239	6495	6751	7007	7263	7519	7775	8031

*Bold case highlights differences in the order between EIA and SEIA.

Table 3.7.3: The Locally Optimal Index Assignment of SEIA for CCITT in Which Codewords Are Arranged by Rows from z_0 to z_{255}. (i_1 - First 5 Digits of Binary Index of z_i; i_2 - Last 3 Digits of Binary Index of z_i.)

i_1 \ z_i \ i_2	000	001	010	011	100	101	110	111
00000	-8031	-7775	-7519	-7263	-7007	-6751	-6495	-6239
00001	-5983	-5727	-5471	-5215	-4959	-4703	-4447	**-4191**
00010	**-3999**	-3871	-3743	-3615	-3487	-3359	-3231	-3103
00011	-2975	**-2847**	**-2719**	**-2591**	-1983	**-1919**	**-1855**	**-1791**
00100	**-2463**	**-2335**	**-2207**	-2079	**-1471**	**-1407**	**-1343**	**-1279**
00101	**-719**	**-687**	**-655**	**-623**	**-471**	**-455**	**-439**	-423
00110	**-219**	**-211**	**-203**	**-195**	**-155**	**-147**	**-139**	**-131**
00111	**-61**	**-57**	**-53**	**-49**	**-30**	**-28**	**-26**	**-24**
01000	-1727	**-1663**	**-1599**	**-1535**	**-975**	**-943**	**-911**	**-879**
01001	**-591**	**-559**	**-527**	**-495**	**-34**	**-327**	**-311**	**-295**
01010	**-187**	**-179**	**-171**	**-163**	**-93**	**-89**	**-85**	**-81**
01011	**-45**	**-41**	**-37**	**-33**	**-6**	**-4**	**-2**	**0**
01100	**-14**	**-12**	**-10**	**-8**	**16**	**18**	**20**	**22**
01101	**65**	**69**	**73**	**77**	**99**	**107**	**115**	**123**
01110	**231**	**247**	**263**	**279**	**359**	**375**	**391**	**407**
01111	**751**	**783**	**815**	847	**1023**	1087	**1151**	**1215**
10000	**-1215**	**-1151**	-1087	**-1023**	-847	**-815**	**-783**	**-751**
10001	**-407**	**-391**	**-375**	**-359**	**-279**	**-263**	**-247**	**-231**
10010	**-123**	**-115**	-107	**-99**	-77	**-73**	**-69**	**-65**
10011	**-22**	**-20**	**-18**	**-16**	**8**	**10**	**12**	**14**
10100	**0**	**2**	**4**	**6**	**33**	**37**	**41**	**45**
10101	**81**	**85**	**89**	**93**	**163**	**171**	**179**	**187**
10110	**295**	**311**	**327**	**343**	**495**	**527**	**559**	**591**
10111	**879**	**911**	**943**	**975**	**1535**	**1599**	**1663**	1727
11000	**24**	**26**	**28**	**30**	**49**	**53**	**57**	**61**
11001	**131**	**139**	**147**	**155**	**195**	**203**	**211**	**219**
11010	423	**439**	**455**	**471**	**623**	**655**	**687**	**719**
11011	**1279**	**1343**	**1407**	**1471**	2079	**2207**	**2335**	**2463**
11100	**1791**	**1855**	**1919**	1983	**2591**	**2719**	**2847**	2975
11101	3103	3231	3359	3487	3615	3743	3871	**3999**
11110	**4191**	4447	4703	4959	5215	5471	5727	5983
11111	6239	6495	6751	7007	7263	7519	7775	8031

*Bold case highlights differences in the order between EIA and SEIA.

Example 3.8: (Robustness)

Both EIA and SEIA depend upon the designated bit error rate q_d, thus the choice of q_d deserves attentions. The values of q_d are tested from 10^{-7} to 0.1 for both EIA and SEIA algorithms using NBC initial code for CCITT example.

For the CCITT codebook, Table 3.8.1 shows the iterations of EIA for various q_d=0.11, 0.01, 0.001 using NBC as an initial code. All three cases follow the same pattern of iterations since EIA seeks to maximize η. However, the local optimal solutions will depend upon the values of q_d. For instance, the two cases of q_d=0.01 and q_d =0.001 lead to the same index assignment with η=0.8438; the case of q_d=0.11 has different index assignment with η=0.8454. It is also observed that two D_c's differ by only a small amount when their η's are close to each other. For instance between iterations 3 and 4, the difference of D_c is only 72 for q_d=0.11, 458 for q_d=0.01, and 52 for q_d =0.001, which are relatively small with respect to their final D_c 3446356, 336239, and 33848. Note that Lemma 3.6 gave a upper bound on the difference between two channel distortions when their η's are the same. In general, the bound is small when η is large. For instance between iterations 36 and 37, the difference of D_c is only 11 for q_d=0.11, 2 for q_d=0.01, and 0.2 for q_d =0.001, which are relatively small with respect to their D_c 3495927, 350246, and 35368 in iteration 37.

Table 3.8.2 shows that the EIA algorithm results in the same index assignment with η_1=0.8438 for q_d running from 0 to 0.1, which demonstrates a high degree of robustness from a very noisy channel (q>0.01) to a low noisy channel. Also, the SEIA algorithm shows its robustness and has the same index assignment with η_2=0.84352 for q_d running from 0 to 0.005. The other cases have similar results, with η_2 between 0.84348 and 0.8448. In summary, both algorithms are quite robust against the varying of designated bit error rate q_d.

Table 3.8.1:The Iterations of EIA with Initial NBC for Various q_d's Using CCITT.

Iteration	η	$D_c(\underline{z},q_d=0.11)$	$D_c(\underline{z},q_d=0.01)$	$D_c(\underline{z},q_d=0.001)$
0	0.8177	3542655	348022	35055
1	0.8360	3470384	338809	34108
2	0.8415	3451401	336567	33880
3	**0.8438**	3446428	**336239**	**33848**
4	**0.8454**	**3446356**	336697	33900
5	0.8471	3447522	337409	33979
6	0.8483	3448695	338002	34044
7	0.8493	3451529	338881	34140
8	0.8499	3453492	339433	34200
9	0.8506	3457176	340385	34303
10	0.8511	3459598	341001	34369
11	0.8516	3463187	341869	34463
12	0.8521	3466650	342701	34552
13	0.8526	3468650	343256	34613
14	0.8529	3469783	343578	34648
15	0.8533	3472298	344183	34713
16	0.8537	3475663	344980	34798
17	0.8538	3475330	344933	34793
18	0.8539	3476236	345130	34815
19	0.8540	3477833	345499	34854
20	0.8542	3479041	345797	34886
21	0.8544	3480463	346129	34922
22	0.8545	3481895	346469	34959
23	0.8546	3483503	346828	34997
24	0.8547	3484990	347160	35033
25	0.8549	3485838	347364	35055
26	0.8551	3488153	347888	35111
27	0.8553	3489983	348333	35159
28	0.8557	3490513	348500	35177
29	0.8557	3490978	348628	35191
30	0.8560	3491089	348756	35205
31	0.8563	3491165	348854	35216
32	0.8567	3493346	349404	35276
33	0.8572	3495225	349939	35334
34	0.8575	3495956	350182	35360
35	0.8576	3495983	350244	35367
36	0.857703	3495938	350248	35368.4
37	0.857705	3495927	350246	35368.2

NOTES: η denotes linearity index; bold cases are locally optimal.

TABLE 3.8.2: Robustness of EIA and SEIA using NBC Initial Code for Various q_d's.

EIA			SEIA		
q_d	η_1	D_{c1}	q_d	η_2	D_{c2}
0-0.1	0.8438	0-3154410	0.09-0.12	0.8447	2858341-3733957
			0.08	0.8448	2558407
			0.006-0.07	0.84348	202308-2254295
			0-0.005	0.84352	0-168619

3.7 Comparison between EIA and LISA

In this section the EIA algorithm is compared to the Linearity Increasing Swap Algorithm (LISA) proposed by Knagenhjelm and Agrell [Knagenhjelm and Agrell, 1996]. Theoretically, both algorithms try to increase the value of the linearity index η that is derived from the Hadamard matrix and its linear eigenspace. However, LISA attempts to maximize the linearity index η alone, while EIA maximizes η and keep channel distortion D_c as small as possible. Since D_c is not monotone decreasing in η, maximizing η can not guarantee a minimal D_c. Technically, EIA applies a sorting algorithm which always increase the value of η, while LISA uses a binary swap technique in which some binary swap do not increase the value of η.

The Linearity Increasing Swap Algorithm (LISA), which repeats the routines of Hamming-1 Butterflies and of Remaining Butterflies until η is maximized, is stated as follows.

Linearity Increasing Swap Algorithm (LISA)

Input: An initial codebook C(or assignment vector $\underline{\mathbf{z}}$)

Output: An Optimal index assignment for C(or assignment vector $\underline{\mathbf{z}}$)

Step 1: Compute the Hadamard transform of $\underline{\mathbf{z}}$, $(\mathbf{M}_{n*n}\underline{\mathbf{z}})/n$.

Step 2: Repeat following routines

Hamming-1 Butterflies

Remaining Butterflies

Until convergence

The Hamming-1 Butterflies routine picks out $b2^{b-1}$ hamming-1 neighbours to perform pair-wise swaps. In the Remaining Butterflies routine, the remaining $(2^{b-1})(2^b\text{-}b\text{-}1)$ pair-wise swaps are considered. Figure 3.1 shows all the 12 Hamming-1 pair-wise swaps and 16 other pair-wise swaps for each cycle in LISA with b=3.

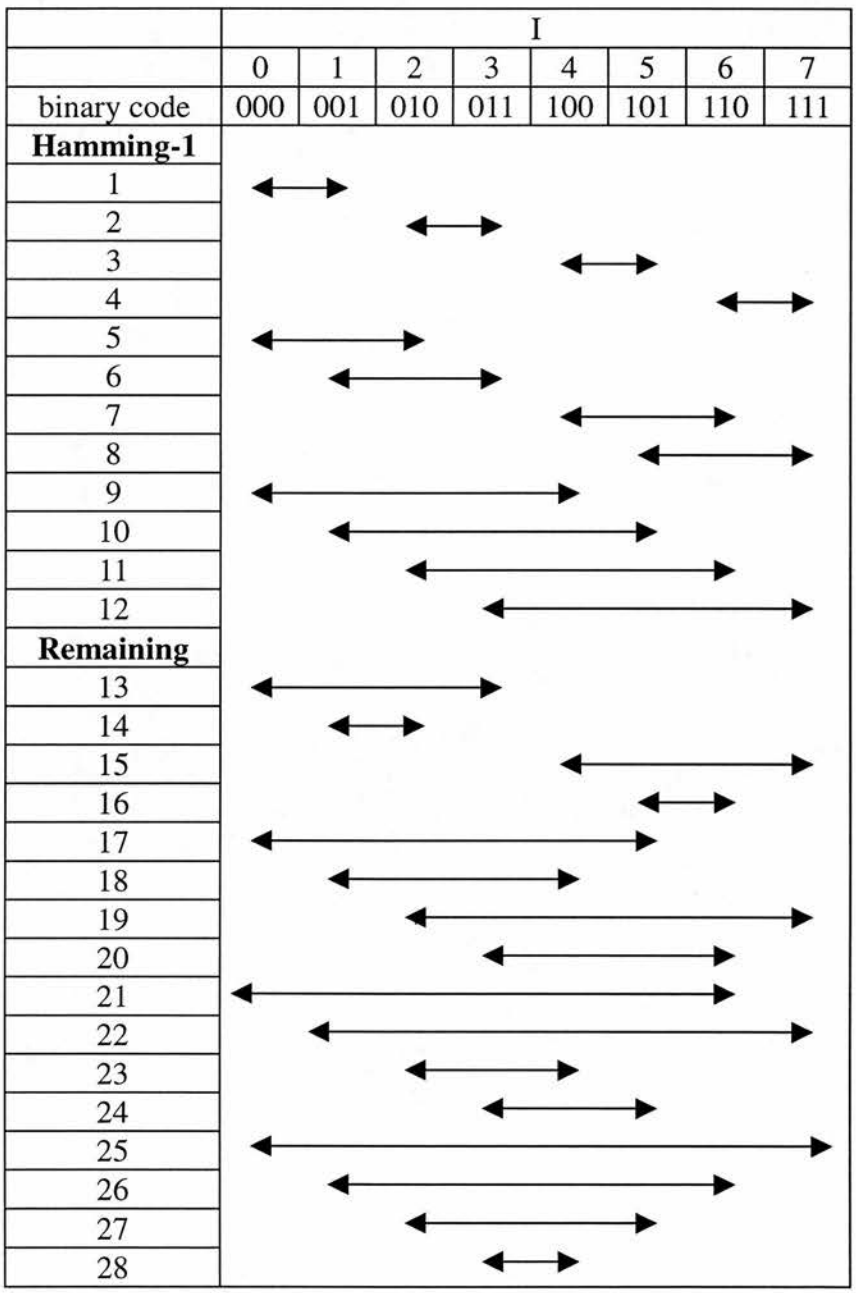

Figure 3.1: All possible binary swaps in LISA for a 3-bit codebook.

For comparison, Example 3.1 is used again to illustrate the difference between the multiple swaps of EIA and the binary swaps of LISA. Table 3.9 shows the multiple swaps in each iteration for EIA and that only one iteration is required to reach the global optimal index assignment. Moreover, every iteration is effective since it always increases the value of η. Table 3.10 shows the binary swaps in each iteration for LISA, where the bold case indicates the permuted codewords in each binary swap that increases the value of η and underline indicates the permuted codewords in each binary swap that does not increase the value of η. It is noteworthy that 25 iterations are required to reach the global optimal index assignment. However only 5 out of the 25 iterations increase η. Furthermore, among the 25 iterations, the first 12 iterations use the Hamming-1 Butterflies routine and the next 13 iterations use the Remaining Butterflies routine and convergence occurs on the 25th iteration. Note that the final codebook with index set {5,7,4,6,1,3,0,2} in Table 3.9 is hamming distance preserving index set to that index set {3,7,1,5,2,6,0,4} in Table 3.10. Hamming distance preserving index assignments are discussed in Appendix A and the results are in Table A.2. In summary, each swap in EIA is effective, hence it takes less iterations to obtain an locally optimal index assignment than LISA does.

Table 3.9: The Multiple Swaps in EIA for an Antipodal Direct Sum Codebook.

iteration	$\underline{z}$								η
0	3.5	0.5	-1.5	1.5	-0.5	-2.5	-3.5	2.5	0.25
1	1.5	3.5	0.5	2.5	-2.5	-0.5	-3.5	-1.5	1.00

Table 3.10: The Binary Swaps in LISA for an Antipodal Direct Sum Codebook.

effective swaps	z								η
0	3.5	0.5	-1.5	1.5	-0.5	-2.5	-3.5	2.5	0.25
1	**0.5**	**3.5**	-1.5	1.5	-0.5	-2.5	-3.5	2.5	**0.5**
2	0.5	3.5	-1.5	1.5	-0.5	-2.5	-3.5	2.5	0.25
3	0.5	3.5	-1.5	1.5	**-2.5**	**-0.5**	-3.5	2.5	**0.78**
4	0.5	3.5	-1.5	1.5	-2.5	-0.5	-3.5	2.5	0.21
5	0.5	3.5	-1.5	1.5	-2.5	-0.5	-3.5	2.5	0.78
6	0.5	3.5	-1.5	1.5	-2.5	-0.5	-3.5	2.5	0.78
7	0.5	3.5	-1.5	1.5	-2.5	-0.5	-3.5	2.5	0.77
8	0.5	3.5	-1.5	1.5	-2.5	**2.5**	-3.5	**-0.5**	**0.96**
9	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.78
10	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.88
11	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.82
12	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.82
13	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.91
14	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.25
15	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.91
16	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.25
17	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.91
18	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.25
19	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.91
20	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.25
21	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.58
22	0.5	3.5	-1.5	1.5	-2.5	2.5	-3.5	-0.5	0.58
23	0.5	3.5	**-2.5**	1.5	**-1.5**	2.5	-3.5	-0.5	**0.98**
24	0.5	3.5	-2.5	1.5	-1.5	2.5	-3.5	-0.5	0.96
25	**-0.5**	3.5	-2.5	1.5	-1.5	2.5	-3.5	**0.5**	**1.00**

Note: **Bold** case indicates the permuted codewords in each binary swap that increases the value of η. Underline indicates the permuted codewords in each binary swap that does not increase the value of η.

3.8 Comparison between EIA and BSA

Two examples of APDS codebook with known optimal solution and a real word case CCITT are considered in this section to compare EIA with BSA (refer to Chapter 2 where BSA was defined).

Example 3.9: (APDS)

An antipodal direct sum codebook can serve as a good example to evaluate the performances of EIA and BSA since its global minimum channel distortion is known as $D_{cmin} = 4q\|\underline{z}\|^2/n$. Consider an arbitrary APDS of size 256 with the coefficients c_i's being equal to (14.77,6.01,4.24,5.49,4.61,2.51,-2.32,12.39). Table 3.11 compares the channel distortion of EIA with that of BSA while the designated bit error rate q_d varying from 0.0001 to 0.1. In the last two columns, "over ratio" indicates how much worse an algorithm will perform than the global optimal solution. The over ratio is between 0.4% to 0.6% for BSA, while it stays within 0.2% for EIA. In practical situation, one may consider the difference between BSA and EIA is almost indistinguishable. However, EIA takes much less CPU time than BSA does. On the Sun Sparc 20 Workstation, the average CPU time for obtaining an optimal solution given q_d=0.0001 to 0.1 is 0.70 seconds with standard deviation 0.21 seconds for EIA. However it is 64 hours (230,400 seconds) with standard deviation 47 hours (169,200 seconds) for BSA!

Table 3.11: The Performances of EIA and BSA Compared to Minimal Channel Distortion for an Antipodal Direct Sum Codebook.

	D_c		D_{cmin}	Over ratio	
q_d	EIA*	BSA**	OPT	(EIA-OPT)/OPT	(BSA-OPT)/OPT
0.0001	0.1959	0.1968	0.1955	0.2%	0.6%
0.001	1.9598	1.9659	1.9553	0.2%	0.5%
0.01	19.5960	19.6303	19.5535	0.2%	0.4%
0.1	195.8196	196.3362	195.5348	0.1%	0.4%

* about 0.70 seconds on the average CPU time; ** about 230,400 seconds on the average CPU time.

Example 3.10: (CCITT)

Consider the voice digitization North American Telephone Systems (CCITT) with n=256 and variance 6,423,572. We compare the SNR performance, robustness, and CPU time between BSA and EIA. Recall from Chapter 2, the quantization distortion D_s=910 is derived from $\frac{1}{12}\sum_{i=1}^{16}\Delta_i^2/16$, where Δ_i is the width of i^{th}

quantization cell. Since the variance of the input signal is not known, we have estimated it by taking the variance of the codebook namely 6423572. SNR is computed as $10\log_{10}\frac{6423572}{910+D_c}$ (dB) by (2.11).

Let $\underline{\mathbf{z}}^*(q_d)$ denote the locally optimal index assignment vector obtained from EIA or BSA using a prespecified q_d, and let $D_c(\underline{\mathbf{z}}^*(q_d),q)$ denote the channel distortion for a given q and $\underline{\mathbf{z}}^*(q_d)$. Results are reported in Table 3.12:

(1) EIA is robust with respect to q_d since the same index assignment is obtained for $10^{-7} \le q_d \le 0.1$.

(2) For BSA, different channel distortions in Columns 3-6 indicate that locally optimal index assignments are different for various values of q_d. Even a slight change in q_d, for instance from 0.0001 to 0.0002, will lead to different results. Therefore, we conclude that BSA is more sensitive to q_d than EIA.

(3) When q_d=0.0001, $D_c(\underline{\mathbf{z}}^*(q_d), 0.0001) = 3408$ is greater than $D_c(\underline{\mathbf{z}}^*(q_d),0.0002)$=3398, $D_c(\underline{\mathbf{z}}^*(q_d),0.001)$=3389 or $D_c(\underline{\mathbf{z}}^*(q_d),0.01)$=3387. However, $D_c(\underline{\mathbf{z}}^*(q_d),q)$ should attain its minimum when $q=q_d$. Thus BSA can only obtain a locally optimal index assignment. Unfortunately, EIA has the same fault.

(4) EIA takes much less CPU time than BSA does. For instance, on a Sun Sparc 20 Workstation BSA requires average 268 hours (964,800 seconds) CPU time with standard deviation 108 hours (388,800 seconds) for while EIA requires only 0.91 seconds with standard deviation 0.19 seconds!

SNR is often used to measure channel distortion. Table 3.13 shows the SNR performance between EIA and BSA is almost indistinguishable, their differences is less than 0.02dB for high resolution. For a high noise $q \ge 0.05$, the performance is quite bad for both algorithms since their SNR's are less than 10.

Table 3.12: The Channel Distortions of Locally Optimal Index Assignments Obtained from EIA and BSA Using Various Designated Bit Error Rates.

$D_c(\underline{z}^*(q_d),q)$	EIA	BSA			
q_d / q	0.0-0.1	0.0001	0.0002	0.001	0.01
0.00001	339	341	340	339	339
0.0001	3387	3408	3398	3389	3387
0.001	33848	34057	33954	33864	33851
0.01	336239	338390	337354	336500	336325
0.1	3154411	3180667	3169584	3164413	3159652
0.25	7179746	7246502	7220214	7213819	7198095
0.5	12847145	12847145	12847145	12847145	12847145

Table 3.13: The Comparison of SNR's Between BSA and EIA for CCITT Example.

	SNR		
q_d	EIA*	BSA**	EIA-BSA
0.0001	31.746	31.725	0.021
0.001	22.667	22.665	0.002
0.01	12.800	12.798	0.002
0.05	5.945	5.944	0.001
0.1	3.087	3.090	-0.002
0.2	0.363	-0.001	0.364
0.3	-1.145	-1.379	0.233
0.4	-2.189	-2.317	0.127
0.5	-3.011	-3.011	0

about 0.91 seconds on the average CPU time; ** about 964,800 seconds on the average CPU time.

In summary, EIA is robust with respect to q_d, but BSA is not. EIA takes much less CPU time than BSA does for obtaining a locally optimal index assignment. Moreover, EIA even has a slightly better performance than BSA has for both the APDS codebook and the CCITT codebook. However the difference is indistinguishable.

3.9 Conclusions

The eigenspace index assignment (EIA) algorithm and its extended version called sequential EIA (SEIA) have been developed for obtaining an optimal index

assignment. Numerical examples of the APDS codebook and a real world case of the CCITT illustrate the proposed algorithms and compare their results with the existing algorithms. The results are summarized as follows:

1. For both EIA and SEIA algorithms:
   a. Multiple bit errors in each transmission are considered in both algorithms.
   b. The object is to minimize channel distortion D_c.
   c. The central idea of the algorithms is to rearrange an assignment vector $\underline{\mathbf{z}}$ as close to the linear eigenspace E_1 as possible, while keeping D_c is as small as possible. Technically, $\underline{\mathbf{z}}$ is regressed on the linear matrix $\mathbf{H}_b$, then $\underline{\mathbf{z}}$ is sorted in accordance with Rank($\underline{\hat{\mathbf{z}}}$) (Corollary 3.10). Overall, the computational complexity of the algorithm relies only on a sorting algorithm.
   d. Both algorithms result in multiple swaps rather than binary swaps (see example 3.3), hence they are very fast and efficient.
   e. In the first iteration, the resulting D_c is improved considerably since the resulting vector is close to E_1. Only minor improvements are obtained after the first iteration (see example 3.4).
   f. For an antipodal direct sum codebook, its linearity index η is equal to one and both algorithms can obtain nearly global optimal index assignments (see example 3.4).
   g. For the CCITT example, η is less than one and the overall effect of D_c is almost concave in η. Thus in order to save computation time, the algorithms can be terminated whenever D_c increases. (see examples 3.5 and 3.6).
2. Comparisons among EIA, SEIA and NBC:
   a. EIA depends on an initial index assignment vector since different initial random codes yield different optimal index assignments. However, in the EIA algorithm, MDC and FBC become NBC after the first iteration. Thus MDC, FBC, and NBC give the same optimal index assignment (see example 3.5).
   b. SEIA is independent of an initial index assignment vector. The index assignment vector in the first iteration becomes the NBC (see Lemma 3.12 and

example 3.6).

c. SEIA needs more iterations than EIA does. Using different initial random codes, EIA occasionally has a better result than SEIA but normally these two algorithms have indistinguishable results (see examples 3.5, 3.6, and 3.7).

d. Both EIA and SEIA outperform NBC code by about 3.4% for small q (see example 3.7) and outperform random code by about 62% on the average (see example 3.5).

3. Comparisons between EIA and LISA:

a. They both try to increase the value of the linearity index for obtaining a locally optimal index assignment. However, LISA maximizes the value of linearity index, but EIA minimizes the D_c for large value of linearity index.

b. All multiple swaps of EIA are effective to increase the value of linearity index while LISA need two complicated formulae to test if a binary swap is effective.

c. EIA needs less iteration to obtain an optimal index assignment than LISA does (see tables 3.9 and 3.10).

d. EIA gives the same result for different initial index assignments such as NBC, FBC, and MDC (see table 3.5.2). LISA depends upon the initial index assignment [Knagenhjelm and Agrell, 1996].

4. Comparisons between EIA and BSA:

a. The D_c or SNR performance of EIA is slightly better than BSA. However the difference is indistinguishable (see examples 3.9 and 3.10).

b. EIA is robust against the designated bit error rates q_d while BSA is more sensitive to q_d (see example 3.10).

c. EIA takes much less CPU time than BSA does (see table 3.13).

d. BSA depends upon the initial index assignment [Zeger and Gersho, 1990] but EIA gives the same result for different initial index assignments such as NBC, FBC, and MDC (see table 3.5.2).

In summary, both EIA and SEIA are very simple, efficient, and robust algorithms for obtaining an optimal index assignment.

Chapter 4

Vector Quantization Model

In this chapter the scalar case will be extended to the vector case, thus the basic definitions and distortion measures are generalized straightforwardly from scalar quantization to vector quantization. A k-dimensional vector quantizer **F** of size $n=2^b$ is a mapping from units of input vector $\underline{\mathbf{x}}^T = [x_1, x_2, \cdots, x_k] \in R^k$ into a codebook **C** containing n codewords $\underline{\mathbf{y}}_i \in R^k$. A quantizing partition $\mathbf{S} = \{\mathbf{S}_0, \mathbf{S}_1, ..., \mathbf{S}_{n-1}\}$ of R^k has n cells associated with the n codewords $\{\underline{\mathbf{y}}_0, \underline{\mathbf{y}}_1, \cdots, \underline{\mathbf{y}}_{n-1}\}$ such that $\mathbf{S}_i = \{\underline{\mathbf{x}} \in R^k : \mathbf{F}(\underline{\mathbf{x}}) = \underline{\mathbf{y}}_i\}$. Let $p(\underline{\mathbf{x}})$ denote the probability density function of $\underline{\mathbf{x}}$, then $p_i = \int_{S_i} p(\underline{\mathbf{x}}) d\underline{\mathbf{x}}$ is the occupancy probability of i^{th} cell. The codeword chosen is a random output of the quantizing partition depending on the random message into input. As in the scalar quantization, in the thesis, we will study the matched equiprobable standardized vector source-quantizer pairs in which all cells are equiprobable, all codewords are cell centroids and the codewords are standardized. This means $p_i=1/n$, $\underline{\mathbf{y}}_i = \int_{S_i} \underline{\mathbf{x}} p(\underline{\mathbf{x}}) d\underline{\mathbf{x}} / p_i$, and $\sum_{i=0}^{n-1} y_{ir} = 0, r = 1,2,...,k$.

Let **Z** denote an $n \times k$ assignment matrix which represents the ordering of the n codewords $\underline{\mathbf{y}}_i$'s, i=0,…,n-1 in **C**. **Z** has k dimensions, so it has k variables $\underline{\mathbf{z}}_r$, r=1,…,k where $\underline{\mathbf{z}}_r$ is the r^{th} component of each of the codewords.

$$\mathbf{Z} = [\underline{\mathbf{z}}_1, \underline{\mathbf{z}}_2, ..., \underline{\mathbf{z}}_k] = \begin{bmatrix} z_{01} & z_{02} & . & z_{0k} \\ z_{11} & z_{12} & . & z_{1k} \\ . & . & . & . \\ . & . & . & . \\ z_{(n-1)1} & z_{(n-1)2} & . & z_{(n-1)k} \end{bmatrix} = \begin{bmatrix} \underline{\xi}_0^T \\ \underline{\xi}_1^T \\ . \\ . \\ \underline{\xi}_{n-1}^T \end{bmatrix}, \qquad (4.1)$$

where codeword $\underline{\xi}_i = [z_{i1}, z_{i2}, \ldots, z_{ik}]^T$, i=0,…,n-1 (i.e. one of the codewords y_i) is the k-dimensional quantizer point that is assigned the b-bit binary expansion of i. From now we will use $\underline{\xi}_i$ to represent the codeword chosen. As in the scalar quantization system, given a unit vector of input $\underline{\mathbf{x}} \in \mathbf{S}_i$, an encoder then converts it to the selected vector $\underline{\xi}_i$ and delivers its b-bit binary index through a binary symmetric channel (BSC) to a decoder. Let $\underline{\xi}_j$ be the received codeword corresponding to $\underline{\mathbf{x}}$ in a decoder, then the resulting mean squared error distortion is measured by $E[\| \underline{\mathbf{X}} - \underline{\xi}_j \|^2]$.

This chapter is organized as follows. Section 4.1 presents the properties of channel distortion. Then a simple, efficient, and robust algorithm VEIA is proposed for the matched equiprobable standardized vector source-quantizer pair in Section 4.2. Section 4.3 presents some numerical results to illustrate the properties of VEIA, and VEIA is compared with the well-known index assignment algorithm, Binary Switching Algorithm (BSA) to show its good performance of efficiency and robustness. Finally, discussions are given in Section 4.4.

4.1 Channel Distortion

For a codebook with centroid condition, the mean squared error distortion (Refer to Chapter 2.4.1) that results from using an assignment matrix **Z** on a binary symmetric channel with channel bit error rate (BER) q (0<q<0.5) is given as

$$
\begin{aligned}
D(\mathbf{Z},q) &= E[\| \underline{X} - \underline{\xi}_j \|^2] = \frac{1}{k}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} q_{ij}\int_{S_i} \| \underline{x} - \underline{\xi}_j \|^2 p(\underline{x})d\underline{x} \\
&= \frac{1}{k}\sum_{i=0}^{n-1}\int_{S_i} \| \underline{x} - \underline{\xi}_i \|^2 p(\underline{x})d\underline{x} + \frac{1}{k}\sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij} \| \underline{\xi}_i - \underline{\xi}_j \|^2 \\
&= D_s + D_c(\mathbf{Z},q),
\end{aligned}
\tag{4.2}
$$

where $q_{ij} = q^{h(i,j)}(1-q)^{b-h(i,j)}$ is the transition probability that the channel output is j given that its input is i, and the hamming distance h(i,j) is the number of bit positions in which i and j are different. Since the first term D_s is the distortion caused by quantization and is independent of both assignment matrix $\mathbf{Z}$ and bit error rate q, we wish to minimize channel distortion $D_c(\mathbf{Z},q)$ by choosing a good index assignment for a given codebook.

For an equiprobable quantizer problem, where the occupancy probabilities are $p_i = 1/n$, i=0,...,n-1. $D_c(\mathbf{Z},q)$ in (4.2) can then be simplified as follows

$$
\begin{aligned}
D_c(\mathbf{Z},q) &= \frac{1}{k}\sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij} \left\| \underline{\xi}_i - \underline{\xi}_j \right\|^2 \\
&= \frac{1}{kn}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} q_{ij} \left\| \underline{\xi}_i - \underline{\xi}_j \right\|^2 \\
&= \frac{2}{kn}\left(\sum_{i=0}^{n-1}\sum_{r=1}^{k} z_{ir}^2 - \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{r=1}^{k} q_{ij} z_{ir} z_{jr}\right) \\
&= \frac{2}{kn}\sum_{r=1}^{k} \| \underline{z}_r \|^2 - \frac{2}{kn}\sum_{r=1}^{k} \underline{z}_r^T \mathbf{Q}_n \underline{z}_r \quad ,
\end{aligned}
\tag{4.3}
$$

where $\| \underline{z}_r \|$ is the Euclidean length of $\underline{z}_r$ and $\mathbf{Q}_n = [q_{ij}]$ is a $n \times n$ positive definite channel transition matrix with $\sum_{i=0}^{n-1} q_{ij} = \sum_{j=0}^{n-1} q_{ij} = 1$.

Similar to the Hadamard transformation of the scalar case in Lemma 3.2, the spectral decomposition of a nxn positive definite channel transition probability matrix $\mathbf{Q}_n$ can be factorized as $\mathbf{Q}_n = \sum_{i=0}^{n-1} \lambda_i \underline{e}_i \underline{e}_i^T$, where λ_i's are the eigenvalues of $\mathbf{Q}_n$ and

$\underline{e}_i$'s are the associated orthonormal eigenvectors of $\mathbf{Q}_n$. There are $b_j=\binom{b}{j}$ eigenvalues with the same value $\lambda_j=(1-2q)^j$, j=0,1,...,b, and hence its associated eigenspace E_j of $\mathbf{Q}_n$ is spanned by those b_j eigenvectors corresponding to λ_j. As the scalar case in Lemma 3.3, the squared length of the projection of $\underline{z}_r$ onto the eigenspace E_j can be computed as $\left\|\mathrm{Proj}_j \underline{z}_r\right\|^2=\sum_{s=1}^{b_j}(\underline{z}_r{}^T\underline{e}_s)^2$, where $\underline{e}_s$, s=1,..,b_j are an orthogonal basis for E_j. Since the whole space of $\mathbf{Q}_n$ consists of E_j, j=0,..,b, where E_j's are mutually orthogonal, the second term of $D_c(\mathbf{Z},q)$ in equation (4.3) can be rewritten as the format of (3.7)

$$\underline{z}_r^T\mathbf{Q}_n\underline{z}_r=\sum_{j=1}^{b}(1-2q)^j\,\|\mathrm{Proj}_j\underline{z}_r\|^2. \tag{4.4}$$

and the first term in (4.3) is $\|\underline{z}_r\|^2=\sum_{j=1}^{b}\|\mathrm{Proj}_j\underline{z}_r\|^2$. The j=0 term is omitted due to the zero-mean codebook assumption. Therefore, $D_c(\mathbf{Z},q)$ in equation (4.3) can be rewritten as

$$D_c(\mathbf{Z},q)=\frac{2}{kn}\sum_{r=1}^{k}\sum_{j=1}^{b}[1-(1-2q)^j]\,\|\mathrm{Proj}_j\underline{z}_r\|^2,\text{for}\quad 0<q<0.5, \tag{4.5}$$

Note that $D_c(\mathbf{Z},q)=0$ when q=0, and is equal to $2\sum_{r=1}^{k}\|\underline{z}_r\|^2/kn$ when q=0.5. It is increasing in q for 0<q<0.5. Moreover,

$$D_c(\mathbf{Z},q)=\frac{2}{kn}\sum_{r=1}^{k}\sum_{j=1}^{b}[1-(2q-1)^j]\,\|\mathrm{Proj}_j\underline{z}_r\|^2,\text{for}\quad 0.5<q<1. \tag{4.6}$$

For a given designated bit error rate q_d, the main purpose of the chapter is to find the locally optimal index assignment $\mathbf{Z}^*(q_d)$ which minimizes the channel distortion $D_c(\mathbf{Z},q_d)$ for a matched equiprobable standardized vector source-quantizer pair.

4.2 EIA for Vector Quantizer (VEIA)

In this section, we proceed to develop an index assignment algorithm (called VEIA) for a k-dimensional vector quantizer based on the scalar EIA. The key idea of the VEIA algorithm is that the assignment matrix $\mathbf{Z} = [\underline{\mathbf{z}}_1, ..., \underline{\mathbf{z}}_k]$ is orthogonally transformed into $\mathbf{T} = [\underline{\mathbf{t}}_1, ..., \underline{\mathbf{t}}_k]$ so that $\underline{\mathbf{t}}_1$ has the largest variance and $\underline{\mathbf{t}}_i$'s are mutually orthogonal, then $\underline{\mathbf{t}}_1$ is treated as a scalar quantizer and the EIA can be directly applied to obtain a proper index assignment.

Since the zero-mean and equiprobable codebook, the k by k positive definite variance-covariance matrix of $\mathbf{Z}$ be

$$\Sigma_{\mathbf{Z}} = \mathrm{Cov}(\mathbf{Z}) = \mathrm{E}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})$$

$$= \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdot & \sigma_{1k} \\ \sigma_{21} & \sigma_{22} & \cdot & \sigma_{2k} \\ \cdot & \cdot & \cdot & \cdot \\ \sigma_{k1} & \sigma_{k2} & \cdot & \sigma_{kk} \end{bmatrix} = \begin{bmatrix} \sigma_{11} & \rho_{12}\sqrt{\sigma_{11}\sigma_{22}} & \cdot & \rho_{1k}\sqrt{\sigma_{11}\sigma_{kk}} \\ \rho_{12}\sqrt{\sigma_{11}\sigma_{22}} & \sigma_{22} & \cdot & \rho_{2k}\sqrt{\sigma_{22}\sigma_{kk}} \\ \cdot & \cdot & \cdot & \cdot \\ \rho_{1k}\sqrt{\sigma_{11}\sigma_{kk}} & \rho_{2k}\sqrt{\sigma_{22}\sigma_{kk}} & \cdot & \sigma_{kk} \end{bmatrix}, \quad (4.7)$$

where $\sigma_{jj} = \mathrm{E}(\underline{\mathbf{z}}_j^{\mathrm{T}}\underline{\mathbf{z}}_j) = \frac{1}{n}\sum_{i=0}^{n-1} z_{ij}^2$, j=1, .. ,k is the variance of $\underline{\mathbf{z}}_j$;

$\sigma_{jj'} = \mathrm{E}(\underline{\mathbf{z}}_j^{\mathrm{T}}\underline{\mathbf{z}}_{j'}) = \frac{1}{n}\sum_{i=0}^{n-1} z_{ij}z_{ij'}$, j,j'=1,...,k and $\rho_{jj'} = \frac{\sigma_{jj'}}{\sqrt{\sigma_{jj}\sigma_{j'j'}}}$ are the covariance and correlation of $\underline{\mathbf{z}}_j$ and $\underline{\mathbf{z}}_{j'}$ respectively. The orthogonal transformation matrix $\mathbf{G}$ is then derived from $\Sigma_{\mathbf{Z}}$ by using spectral decomposition [Johnson and Wichern, 1988]. That is

$$\Sigma_{\mathbf{Z}} = \mathbf{G}\Omega\mathbf{G}^{\mathrm{T}}, \quad (4.8)$$

where $\Omega = \mathrm{diag}(\omega_1, ..., \omega_k)$ with $\omega_1 \geq \omega_2 \geq ... \geq \omega_k > 0$, and $\mathbf{G} = [\underline{\mathbf{g}}_1, \underline{\mathbf{g}}_2, ..., \underline{\mathbf{g}}_k]$ is the orthogonal transformation matrix and $\underline{\mathbf{g}}_i$ is the associated standardized eigenvector with respect to eigenvalue ω_i. Now let us consider the orthogonal transformation of an assignment matrix $\mathbf{Z}$ such that,

$$\mathbf{T}_{n\times k} = \mathbf{Z}_{n\times k}\mathbf{G}_{k\times k} = [\underline{\mathbf{t}}_1,....,\underline{\mathbf{t}}_k] = \begin{bmatrix} \underline{\mathbf{u}}_0^T \\ \underline{\mathbf{u}}_1^T \\ . \\ \underline{\mathbf{u}}_{n-1}^T \end{bmatrix}, \tag{4.9}$$

where $\underline{\mathbf{t}}_i$'s are mutually orthogonal and $\underline{\mathbf{t}}_1$ has the largest variance and each codeword $\underline{\mathbf{u}}_i$, i=0,…,n-1 is the k-dimensional quantizer point that is assigned the b-bit binary expansion of i. By the property of an orthogonal matrix, we have $\mathbf{G}^T = \mathbf{G}^{-1}$. Consequently, the variance-covariance matrix of $\mathbf{T}$ can be obtained as

$$\text{Cov}(\mathbf{T}) = \text{Cov}(\mathbf{ZG}) = \mathbf{G}^T\,\text{Cov}(\mathbf{Z})\mathbf{G} = \mathbf{G}^T\,\Sigma_{\mathbf{Z}}\mathbf{G} = \mathbf{G}^T\,\mathbf{G}\Omega\mathbf{G}^T\mathbf{G} = \Omega, \tag{4.10}$$

that is $\text{Var}(\underline{\mathbf{t}}_j) = \omega_j$, $j = 1,2...,k$ with $\omega_1 \geq \omega_2 \geq ... \geq \omega_k > 0$ and $\text{Cov}(\underline{\mathbf{t}}_i, \underline{\mathbf{t}}_j) = 0$ for all $i \neq j$. $\underline{\mathbf{t}}_1$ has the largest variance being equal to the largest eigenvalue ω_1.

If $\underline{\mathbf{z}}$'s are highly correlated, i.e. the correlation is close to one, then the data points form a thin ellipsoid. Example 4.1 gives a clear example for correlation and largest eigenvalue.

Example 4.1: (Two-dimensional quantizer with equal variance)

Assume that a two-dimensional quantizer has equal variance for each dimension, say $\sigma_{11} = \sigma_{22}$, so the variance-covariance of $\mathbf{Z}$ is given by $\Sigma_Z = \begin{pmatrix} \sigma_{11} & \rho_{12}\sigma_{11} \\ \rho_{12}\sigma_{11} & \sigma_{11} \end{pmatrix}$. Consequently, the eigenvalues are $\omega_1 = \sigma_{11}(1+\rho_{12})$ and $\omega_2 = \sigma_{11}(1-\rho_{12})$, and their associated eigenvectors are $\underline{\mathbf{g}}_1 = [1/\sqrt{2}, 1/\sqrt{2}]$ and $\underline{\mathbf{g}}_2 = [1/\sqrt{2}, -1/\sqrt{2}]$, respectively. Now, if the correlation is close to one, then ω_2 is close to zero while ω_1 is relatively large. This indicates that the data points form a thin ellipsoid, which lies along the major axis of $\underline{\mathbf{g}}_1$, or the two-dimensional quantizer can be treated as a scalar quantizer.

Therefore, in general, the largest eigenvalue ω_1 is relatively larger than the others if the $\underline{z}$'s are highly correlated. Also, as $\omega_1 = \mathrm{Var}(\underline{t}_1)$, this means that after the orthogonal transformation, the elements of **T** lie along a thin ellipsoid in the k-dimensional Euclidean space. In that case $\underline{t}_1$ is a good approximation for the original data and the problem can be treated simply as a scalar quantizer. Furthermore, geometrically an orthogonal transformation rotates the axes without changing the distance between any two points, and hence the channel distortion remains the same. This leads to the following Lemma.

Lemma 4.1: Let **Z** denote a n by k assignment matrix which represents the ordering of the n codewords $\underline{\xi}_i$, i=0,…,n-1 and $\mathbf{T} = [\underline{t}_1,...,\underline{t}_k]$ be the n by k matrix considering of codewords $\underline{u}_i$, i=0,…,n-1, which is an orthogonal transformation of **Z**. For an orthogonal transformation, the channel distortion remains the same, namely $D_c(\mathbf{Z},q) = D_c(\mathbf{T},q)$.

Proof of Lemma 4.1: Consider an orthogonal transformation **G** in (4.9) such that $\underline{u}_i^T = \underline{\xi}_i^T \mathbf{G}$, then by (4.2) and $\mathbf{G}\mathbf{G}^T = \mathbf{I}$, we have

$$
\begin{aligned}
D_c(\mathbf{T},q) &= \frac{1}{k}\sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij} \left\| \underline{u}_i - \underline{u}_j \right\|^2 = \frac{1}{k}\sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij} (\underline{u}_i - \underline{u}_j)^T (\underline{u}_i - \underline{u}_j) \\
&= \frac{1}{k}\sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij} (\underline{\xi}_i - \underline{\xi}_j)^T \mathbf{G}\mathbf{G}^T (\underline{\xi}_i - \underline{\xi}_j) = \frac{1}{k}\sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij} (\underline{\xi}_i - \underline{\xi}_j)^T (\underline{\xi}_i - \underline{\xi}_j) \\
&= \frac{1}{k}\sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij} \left\| \underline{\xi}_i - \underline{\xi}_j \right\|^2 = D_c(\mathbf{Z},q)
\end{aligned}
$$

Thus, the channel distortion remains the same. □

Therefore, for a matched equiprobable standardized vector source-quantizer pair, the proposed VEIA algorithm can be summarized as follows:

Eigenspace Index Assignment Algorithm for VQ (VEIA)

Input: An initial index assignment assignment matrix **Z**

Output: An Optimal index assignment assignment matrix **Z**

Step 0: Find the orthogonal transformation matrix **G** of a given assignment matrix **Z**.

Step 1: Perform the orthogonal transformation of **T=ZG** to obtain the $\underline{\mathbf{t}}_1$ with the largest eigenvalue ω_1. Sort the rows of **T** and **Z** according to the entries of $\underline{\mathbf{t}}_1$ in accending order, namely using the natural binary code as the initial code.

Step 2: Choose a designated bit error rate q_d and apply the EIA algorithm to $\underline{\mathbf{t}}_1$, namely, to permute the rows of **T**, say k-dimensional codewords $\underline{\mathbf{u}}_i, i = 0,...,n-1$ so that the channel distortion $D_c(\mathbf{T},q_d)=D_c(\mathbf{Z},q_d)$ is locally minimized.

Since the VEIA algorithm is applying the EIA algorithm directly to the $\underline{\mathbf{t}}_1$ vector, it must converge in a finite number of iterations. The program for implementing this algorithm is to be found in Appendix C.

4.3 Numerical Results

Several examples in this section illustrate the performances of the proposed VEIA algorithm versus the well-known Binary Switching Algorithm (BSA) at q_d=0.0001, 0.001, 0.01, and 0.1 in terms of robustness, CPU time, and signal-to-noise ratio (SNR). In general, the borderline between high and low resolutions is a SNR value of 10 dB [Gersho and Gray, 1992]. Higher SNR values means higher resolution. Note that the case of multiple-bit errors is considered throughout the thesis.

All codebooks were produced by the GLA algorithm with a training set with size 7500 which was randomly generated with an initial seed 99999 from the first-order Gauss-Markov input $\{x_j\}$ of the form [Zeger and Gersho, 1990]

$$x_{j+1}= \rho\, x_j+w_j, \tag{4.11}$$

where w_j is iid Gaussian input $N(0,\sigma_w^2)$, and variance σ_w^2 is often chosen as unity for convenience. Table 4.1 summarizes the detailed information of each codebook adopted in this section.

Table 4.1: The list of codebooks produced by GLA algorithm with training set with size 7500 generated randomly with an initial seed 99999 from the first-order Gauss-Markov units of input.

Codebook	b	k	ρ	Ds	MSE(x)	$\omega_i, i = 1,2,...,k$
C669	6	6	0.9	0.395	5.485	(30.9,3.18,0.61,0.05,0.02,0.004)
C660	6	6	0	0.302	0.993	(1.179,0.835,0.709,0.662,0.615,0.357)
C639	6	3	0.9	0.162	5.481	(14.98,1.04,0.21)
C630	6	3	0	0.094	0.995	(1.203,1.158,1.074)
C629	6	2	0.9	0.064	5.598	(12.77,0.66)
C620	6	2	0	0.028	0.997	(1.389,1.355)
C729	7	2	0.9	0.034	5.598	(11.89,0.69)
C829	8	2	0.9	0.017	5.598	(11.04,0.68)
C929	9	2	0.9	0.009	5.598	(11.11,0.62)
C1029	10	2	0.9	0.004	5.598	(10.52,0.56)

Note: b:number of bits; k:number of dimensions;ρ:correlation of first order Gauss-Markov input; D_s:quantization distortion; ω_i :eigenvalues of orthogaonal transformation (or Variance of $\underline{\mathbf{t}}_i$).

Table 4.2 shows the iterations of VEIA for three cases of designated bit error rate (q_d) using the codebook C660, and three interesting facts are observed:

(1) It is clear that the channel distortion $D_c(\mathbf{Z},q)$ is not monotonic decreasing in the linearity index η for all three cases. Knagenhjelm and Agrell [Knagenhjelm and Agrell, 1996] also pointed out that the channel distortion is not a monotonic function of the linearity index, but they do have very strong negative correlation. It indicates that the linearity index is a potent parameter in the search for a good index assignment.

(2) The linearity index η in the VEIA is independent of q_d, in other words, the iterations are identical for different values of q_d.

(3) The locally optimal index assignment depends upon a given q_d. The locally optimal index assignment occurs in the third iteration for q_d=0.1 while in the

first iteration for both q_d=0.01 and 0.0001. Since a locally optimal index assignment depends on q_d as shown in Table 4.2, the channel mismatch problem pointed out by Farvadin [Farvadin, 1990] will occur when the bit error rate is dynamic.

Table 4.2:The Iterations of VEIA for Various q_d's Using Codebook C660.

Iteration	η	$D_c(\mathbf{Z}, q_d = 0.1)$	$D_c(\mathbf{Z}, q_d = 0.01)$	$D_c(\mathbf{Z}, q_d = 0.0001)$
0	0.985448	0.6507	0.07531	0.000772
1	0.988923	0.5853	**0.07146**	**0.000731**
2	0.989517	0.6102	0.07578	0.000777
3	0.990197	**0.5837**	0.07149	0.000732
4	0.990274	0.5966	0.07307	0.000748

NOTES: η denotes linearity index; bold cases are locally optimal.

Table 4.3: The Channel Mismatch Patterns for Various q_d in VEIA Using Codebook C660. (The entries are $D_c(\mathbf{Z}^*(q_d), q)$'s .)

$D_c(\mathbf{Z}^*(q_d),q)$: q_d / q	0.01	0.1	sign
	(a)	(b)	(a)-(b)
1.0E-6	0.7312E-5	0.7317E-5	-
0.5E-5	0.3656E-4	0.3658E-4	-
1.0E-5	0.7312E-4	0.7317E-4	-
0.5E-4	0.3655E-3	0.3658E-3	-
1.0E-4	0.7310E-3	0.7315E-3	-
0.5E-3	0.3652E-2	0.3654E-2	-
1.0E-3	0.7295E-2	0.7300E-2	-
0.5E-2	0.3614E-1	0.3616E-1	-
1.0E-2	0.7146E-1	0.7148E-1	-
0.5E-1	0.3265E+0	0.3261E+0	+
0.1	0.5853E+0	0.5836E+0	+
0.15	0.7903E+0	0.7871E+0	+
0.20	0.9529E+0	0.9482E+0	+
0.25	0.1082E+1	0.1076E+1	+
0.30	0.1186E+1	0.1180E+1	+
0.35	0.1271E+1	0.1265E+1	+
0.40	0.1341E+2	0.1336E+2	+
0.45	0.1400E+2	0.1397E+2	+
0.50	0.1452E+2	0.1452E+2	+

Furthermore, Table 4.3 shows the channel mismatch problem for both q_d=0.1 and q_d=0.01 in the VEIA using the codebook C660, where the entries are $D_c(\mathbf{Z}^*(q_d),q)$'s. We notice that two patterns intersects each other at q between 0.01and 0.05. For q<0.01, the channel distortions for q_d=0.01 are less than that for q_d=0.1, and vice verse for q>0.05. Namely, the index assignment for q_d=0.01 outperforms that for q_d=0.1 when q is small, and vice verse when q is large. Since all index assignment algorithms using $D_c(\mathbf{Z},q)$ as a measure have channel mismatch problem in dynamic channel, a new measure will be proposed in Chapter 5 to solve the channel mismatch problem.

However, the VEIA algorithm is quite robust as a designated bit error rate varies, namely, the same index assignment will be obtained when q_d varies at some range. For the C660 codebook VEIA reaches the same index assignment when q_d varies from 0.0001 to 0.01. Table 4.4 shows the comparison of robustness between VEIA and BSA algorithms. VEIA is quite robust as q_d varies from 0.0001 to 0.1 for C669, C639, C630, C629, and C620 codebooks while BSA is only robust against low noisy channel (10^{-4}-10^{-3}) for C669, C660, C629, and C620 codebooks. As in the scalar case, BSA is more sensitive to q_d than VEIA is. For instance the results of BSA for C639 and C630 are all different when q_d varies between 10^{-4}-10^{-1}.

Table 4.4: The Comparison of Robustness Between VEIA and BSA for q_d within 10^{-4} to 10^{-1} in different codebooks.

q_d	C669	C660	C639	C630	C629	C620
VEIA	10^{-4}-10^{-1}	10^{-4}-10^{-2}	10^{-4}-10^{-1}	10^{-4}-10^{-1}	10^{-4}-10^{-1}	10^{-4}-10^{-1}
BSA	10^{-4}-10^{-3}	10^{-4}-10^{-3}	No	No	10^{-4}-10^{-3}	10^{-4}-10^{-3}

Note: No- BSA is not robust from 10^{-4} to 10^{-1}.

Table 4.5 compares the SNR's between BSA and VEIA for different values of k and ρ. Recall from Chapter 2, higher SNR values imply better resolution and high resolution is taken to be a SNR value greater than 10dB [Gersho and Gray 1992]. For those codebooks with high correlation ρ=0.9 (C669,C639,and C629), the SNR's of VEIA and BSA are almost indistinguishable. The differences between these two algorithms become slightly larger as k decreases and b is fixed. For instance the differences increases from 0.036 to 0.107 dB when k decreases from 6 to 2 for b=6. The results are similar for those codebooks with no correlation ρ=0.0 (C660,C630,and C620). In general, the differences are magnified when correlation decreases, for instance the difference is 0.047dB for k=3 and ρ=0.9 while it is 0.200dB for k=3 and ρ=0.0.

Table 4.5: The Comparison of SNR's (in dB) Between BSA and VEIA for Various k and ρ. (Given b=6, and q_d =0.001.)

Codebook	dimension (k)	ρ	BSA	VEIA	BSA-VEIA
C669	6	0.9	11.159	11.123	0.036
C639	3	0.9	14.715	14.668	0.047
C629	2	0.9	17.873	17.766	0.107
C660	6	0.0	5.100	5.066	0.034
C630	3	0.0	9.992	9.792	0.200
C620	2	0.0	14.660	13.992	0.668

In Table 4.6, the SNR performance is given as a function of bit error rate for codebook C669 using q_d=0.01. The results indicate the difference between BSA and VEIA is less than 0.13797dB for high resolution performance. For the same example, Figure 4.1 shows the results of BSA and VEIA are almost indistinguishable.

Table 4.6:The Comparison of SNR between BSA and VEIA for Codebook C669 and q_d=0.01.

q	BSA	VEIA	BSA-VEIA
1.0E-6	11.42552	11.42548	0.00004
0.5E-5	11.42442	11.42423	0.00019
1.0E-5	11.42304	11.42266	0.00038
0.5E-4	11.41203	11.41016	0.00187
1.0E-4	11.39831	11.39458	0.00373
0.5E-3	11.29009	11.27194	0.01814
1.0E-3	11.15852	11.12344	0.03508
0.5E-2	10.22966	10.09169	0.13797
1.0E-2	9.29478	9.07818	0.21660
0.5E-1	5.25674	4.89395	0.36278
0.1	2.84421	2.50134	0.34287
0.15	1.32596	1.03160	0.29435
0.20	0.21898	-0.02252	0.24150
0.25	-0.65125	-0.84146	0.19021
0.30	-1.36778	-1.51037	0.14259
0.35	-1.97663	-2.07615	0.09951
0.40	-2.50598	-2.56732	0.06134
0.45	-2.97427	-3.00246	0.02818
0.50	-3.39427	-3.39427	0.00000

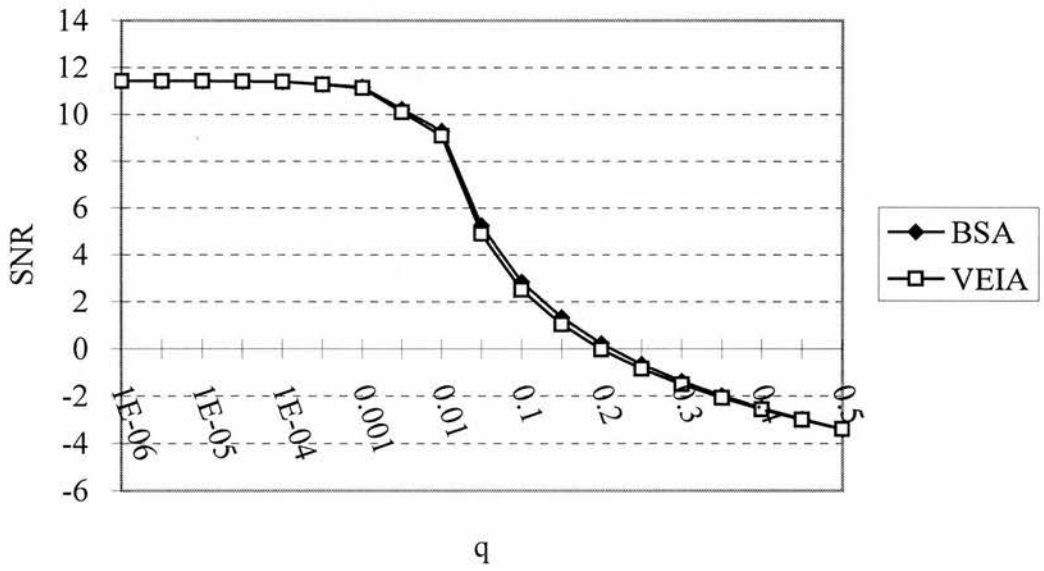

Figure 4.1: The Comparison of SNR between BSA and VEIA for Codebook C669 and q_d=0.01.

Table 4.7 compares the SNR`s and CPU time's between BSA and VEIA given that k=2 and q_d=0.001 to show the effect of large number of bits. It is obvious that VEIA gives similar SNR to BSA and is much faster to find an assignment than BSA in all cases. The only time consuming part in VEIA is the sorting algorithm, hence, the CPU of VEIA depends on which sorting algorithm is adopted, for instances, $O(n^2)$ for the bubble sort; O(nlogn) for the binary sort, where n is the size of a codebook. Bubble sort reorders the sequence so that the elements are in nondecreasing order by performing a series of passes over the sequence. In each pass, the elements are scanned by increasing rank, from rank 0 to the end of the sequence of unordered items. At each position in a pass, an element is compared with its neighbour, and if the preceding element is larger than the succeeding one, then the two elements are swaped. In Binary sort a binary search tree is constructed and its elements visited in a nondecreasing order. In a binary search tree, each internal node ϖ stores an element ε such that the elements stored in the left subtree of ϖ are less than or equal to ε, and the elements stored in the right subtree of ϖ are greater than ε[Goodrich and Tamassia, 1998]. Note that 62 CPU hours (223,200 seconds) are required for BSA when b=8 (or n=256) on a SUN Sparc 20 Workstation with clock speed 75Mhz compared with 8.4 seconds for VEIA. Although VEIA is very fast, its SNR`s are still close enough to that of BSA. The difference of SNR is only 0.327 dB when b=8 and that is almost indistinguishable. Note that SNR increases as b increases. For the high bit rate such that b=10, only 461 seconds are required for VEIA and its SNR is reasonable as large as 22.552 dB.

Table 4.7: The Comparison of SNR's and CPU Time's Between BSA and VEIA. (Given k=2, ρ=0.9, q_d =0.001.)

Codebook	b	CPU		SNR(dB)		
		BSA	VEIA	BSA	VEIA	BSA-VEIA
C629	6	168s	0.3s	17.873	17.766	0.107
C729	7	1.8h	1.5s	19.734	19.537	0.197
C829	8	62h	8.4s	21.334	21.007	0.327
C929	9	>260h	38s		22.065	
C1029	10	>260h	461s		22.552	

Note: s-seconds, h-hours, $SNR = 10\log_{10}[MSE(\underline{\mathbf{x}})/(D_s + D_c(\mathbf{Z}^*(q_d), q_d)]$.

4.4 Conclusions

For a vector quantization with centroid and equiprobable properties, the VEIA algorithm has been developed to obtain a locally optimal index assignment. The results are summarized as follows:

1. The case of multiple bit errors in each transmission is considered in the VEIA.
2. The key idea of the VEIA algorithm is to apply the EIA algorithm on the vector $\underline{\mathbf{t}}_1$ which is the first dimension with the largest variance after an orthogonal transformation is performed. In the VEIA algorithm, record D_c while maximizing the linearity index, and the minimum D_c is selected at the end.
3. Technically speaking, only a sorting algorithm is required in term of computation complexity.
4. The $D_c(\mathbf{Z}, q)$ is not monotonic decreasing in the linearity index η, however, the two means have high negative correlation (see Table 4.2).
5. The linearity index η in the VEIA is independent of q_d (see Table 4.2).
6. A locally optimal index assignment of VEIA depends upon a given q_d (see Table 4.2).
7. It is possible that two performance curves in terms of $D_c(\mathbf{Z}^*(q_d),q)$ for different values of q_d will intersect each, thus a channel mismatch problem occurs (see Table 4.3). This problem will be addressed in Chapter 5.

8. Both VEIA and BSA obtain a locally optimal index assignment depending upon a given q_d, however, VEIA is quite robust for a certain range of q, namely, the same index assignment will be obtained for q_d ranging from 0.0001 to 0.1 for most codebooks. While BSA is only robust at low noise ranging from 0.0001 to 0.001 for some codebooks (see Table 4.4).
9. The SNR performances for both VEIA and BSA are almost indistinguishable. The differences between these two algorithms become slightly larger for low correlations and low dimensions with fixed b (see Table 4.5 - 4.7, and Figure 4.1).
10. The VEIA is much faster than the BSA is in all cases (see Table 4.7).

In summary, for a matched equiprobable standardized vector source-quantizer pair, the VEIA is very simple, efficient, and robust while its performance of distortion is comparable with that of the BSA.

Chapter 5

Channel Mismatch in Dynamic Channels

In many practical situations noisy communication channels are dynamic, thus the actual value of the bit error rate q varies with time. Recall that the channel distortion of scalar or vector quantizer ($D_c(\underline{z},q)$ or $D_c(\underline{Z},q)$) is a function of q. Therefore, because the bit error rate q varies with time, so does the channel distortion. Up to now, all existing index assignment algorithms such as BSA, LISA, EIA, etc. aimed to minimize $D_c(\underline{z},q)$ for a prespecified q (called the designated bit error rate q_d). Hence it was important to know q_d in order to design the optimal index assignment, or equivalently the optimal assignment vector $\underline{z}^*$. The channel mismatch problem occurs when the actual value of the bit error rate is not the same as the designated bit error rate q_d. There are difficulties in finding an optimal index assignment under channel mismatch conditions since the distortion measure of performance depends upon q. Hence, in this chapter, a new distortion measure of performance which is independent of q is proposed which avoids the channel mismatch problem. Let the dynamic bit error rate in a noisy dynamic communication channels be taken as a statistical random variable denoted by Q. The choice of a proper prior distribution for Q becomes a key issue for developing a new distortion measure.

This chapter introduces the channel mismatch problem. In Section 5.2 we suggest that a beta distribution gives a good description of the behavior of Q and because it is the conjugate prior of binomial distributions it allows exact calculation of the performance of different index assignments. The expected channel distortion (ECD) based on a beta distribution is then developed and gives a new distortion

measure of performance which is independent of q. By using ECD instead of $D_c(\underline{z},q)$, two mismatch-free index assignment algorithms called EIA-ECD and SEIA-ECD are proposed in Section 5.3. Numerical results are discussed in Section 5.4. Finally, conclusions are presented and in addition, the expected channel distortion based on a beta distribution is presented for a vector quantizer.

5.1 Channel Mismatch Problem

Let Q denote the dynamic bit error rate in a noisy dynamic communication channel. Let q denote the actual bit error rate, and q_d denote the designated bit error rate for which $D_c(\underline{z},q_d)$ was minimized to obtain an optimal assignment vector $\underline{z}^*$. While the actual bit error rate q varies with time, q is not necessary the same as q_d, and the resulting optimal assignment vector is not guaranteed to be optimal for the actual q. This demonstrates the channel mismatch problem, namely, that an optimal assignment vector at q_d, say $\underline{z}^*(q_d)$, may not be the optimal vector uniformly within $0<q<0.5$. Recall from Chapters 3 and 4, that the distortion performance of one index assignment is not consistently better than the other index assignments for different q_d's when q varies from 0 to 0.5. For the real world case of the voice digitization in North American Telephone Systems (CCITT), Table 5.1 lists the $D_c(\underline{z}^*(q_d),q)$ performances of the SEIA algorithm using $q_d=0.01$ and $q_d=0.1$, and their comparisons for actual q ranging from 0 to 0.5. The result shows that the two performance curves for $q_d=0.01$ and $q_d=0.1$ intersect when q is between 0.01 and 0.1. Knagenhjelm [Knagenhjelm, 1993] also reported the channel mismatch problem when the bit error rate q varies from 0 to 0.05. Moreover, Farvardin [Farvardin, 1990] showed the channel mismatch problem by comparing different encoding schemes when the actual bit error rate q varies between 0.001 and 0.1. He also suggested that the value of q_d can be set as the average dynamic bit error rate $\overline{q}$ under following assumptions: (1) the bit error rate q is very small, say $bq<<1$, where b is the number of bits, (2) there is only a single-error pattern, and (3) q follows a uniform distribution. However, we prefer to relax the above assumptions and

consider a more general case where q has a multiple-error pattern, and is not necessary uniformly distributed or small enough.

Table 5.1: The Comparison of $D_c(\underline{z}^*(q_d),q)$ Performances of SEIA Algorithm Using CCITT Codebook for Different q_d's to Show Channel Mismatch Problem.

$D_c(\underline{z}^*(q_d),q)$: q_d / q	0.1	0.01	(a)-(b)
	(a)	(b)	
0.5	12847144.000	12847144.000	0.000
0.4	10661192.000	10664000.000	-2808.000
0.3	8381580.000	8385672.000	-4092.000
0.2	5912551.000	5915508.000	-2957.000
0.1	3154127.000	3154536.000	-409.000
0.01	336414.600	336190.230	224.370
0.001	33868.800	33842.559	26.241
0.0001	3389.170	3386.507	2.663
0.00001	338.940	338.673	0.267
0.000001	33.894	33.868	0.026

5.2 Beta Distribution for Bit Error Rate

To avoid the channel mismatch problem, the key issue is to find a new distortion measure which is independent of q. For a given assignment vector $\underline{z}$, the channel distortion as defined in the Equation (3.1) is

$$D_c(\underline{z},q)=\sum_{i=0}^{n-1} p_i \sum_{j=0}^{n-1} q_{ij}(z_i - z_j)^2 , \qquad (5.1)$$

where $q_{ij} = q^{h(i,j)}(1-q)^{b-h(i,j)}$ is the transition probability that the channel output is j given that its input is i and the hamming distance h(i,j) is the number of bit positions in which i and j are different. Note that q_{ij} has a binomial distribution with parameter q.

A beta distribution is chosen to fit the dynamic bit error rate Q for the following three reasons: (a) In mathematical statistics, a beta distribution is the conjugate prior distribution for a binomial distribution [Lindgren, 1976]. Therefore, if the dynamic

bit error rate has a beta type distribution, then the expected channel distortion (ECD) has a closed form formula. (b) A beta distribution with parameters α and β can fit many types of data including uniform, linear, L-shape, J-shape, uni-mode and bath tub shapes. In general, a dynamic bit error rate tends to small values with larger probability. Therefore, a beta distribution [Johnson and Kotz, 1970] with L-shape density function, say $\alpha=1$ and $\beta>1$, is a good description of the dynamic bit error rate. Also, the choice of $\alpha=1$ can simplify the computation. (c) Since the expected value of the dynamic bit error rate Q has the closed form of $\alpha/[2(\alpha+\beta)]$ for a beta distribution, it would be easy to choose the value of β to fit the average value of Q. Lemma 5.1, Lemma 5.2 and Theorem 5.3, present the underpinning for this new distortion measure, ECD.

Lemma 5.1: Let the random variable Q denote the dynamic bit error rate, and 2Q have a beta distribution with parameters α and β, then the probability density function of Q is given by

$$f(q) = \frac{2}{B(\alpha,\beta)}(2q)^{\alpha-1}(1-2q)^{\beta-1}, \quad 0 < q < 0.5, \tag{5.2}$$

where $B(\alpha,\beta) = \Gamma(\alpha)\Gamma(\beta) / \Gamma(\alpha+\beta)$ is a beta function with parameters α and β, and $\Gamma(\alpha) = (\alpha-1)!$.

Proof of Lemma 5.1:

Let Y have a beta distribution, then the probability density function of Y is as follows:

$$g(y) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{\alpha-1}(1-y)^{\beta-1}, \qquad 0 < y < 1. \tag{5.3}$$

Replace Y=2Q in (5.3), then the probability density function of Q is derived as

$$\begin{aligned} f(q) &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(2q)^{\alpha-1}(1-2q)^{\beta-1}(\frac{dy}{dq}), \qquad 0 < q < 0.5 \\ &= 2\cdot\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(2q)^{\alpha-1}(1-2q)^{\beta-1} \\ &= \frac{2}{B(\alpha,\beta)}(2q)^{\alpha-1}(1-2q)^{\beta-1} \end{aligned}$$

□

Lemma 5.2: Let the random variable Q denote the dynamic bit error rate with a density function given in (5.2), then the r^{th} moment of Q is equal to

$$E(Q^r) = 2^{-r} \frac{\prod_{i=\alpha}^{\alpha+(r-1)} i}{\prod_{i=\alpha+\beta}^{\alpha+\beta+(r-1)} i}. \tag{5.4}$$

Proof of Lemma 5.2:

$$\begin{aligned} E(Q^r) &= \int_0^{0.5} q^r f(q)dq = \int_0^{0.5} q^r \cdot 2\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(2q)^{\alpha-1}(1-2q)^{\beta-1}dq \\ &= 2^{-r}\int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(2q)^{\alpha+r-1}(1-2q)^{\beta-1}d(2q) \\ &= 2^{-r}\frac{\Gamma(\alpha+\beta)\Gamma(\alpha+r)}{\Gamma(\alpha)\Gamma(\alpha+\beta+r)}\int_0^1 \frac{\Gamma(\alpha+\beta+r)}{\Gamma(\alpha+r)\Gamma(\beta)}(2q)^{\alpha+r-1}(1-2q)^{\beta-1}d(2q) \\ &= 2^{-r}\frac{(\alpha+\beta-1)!(\alpha+r-1)!}{(\alpha-1)!(\alpha+\beta+r-1)!} = 2^{-r}\frac{\prod_{i=\alpha}^{\alpha+(r-1)} i}{\prod_{i=\alpha+\beta}^{\alpha+\beta+(r-1)} i} \end{aligned}$$

□

Let r=1 in Lemma 5.2, the expected value of Q is equal to

$$E(Q)=\alpha/[2(\alpha+\beta)]. \tag{5.5}$$

The variance of Q is equal to $Var(Q)=E(Q^2)-E^2(Q)= \alpha\beta/[4(\alpha+\beta)^2(\alpha+\beta+1)]$ and the standard deviation of Q, $\sigma_Q=\sqrt{Var(Q)}$. The value of the β-parameter can be varied according to the actual situation. Note that increasing the value of β will decrease the expected value of Q, thus a larger value of β is appropriate for the case where Q tends to have a small average value. Figure 5.1 shows the density function of Q with parameters α=1 and β=20 within the interval of (0,0.5). Its expected value is equal to 1/42.

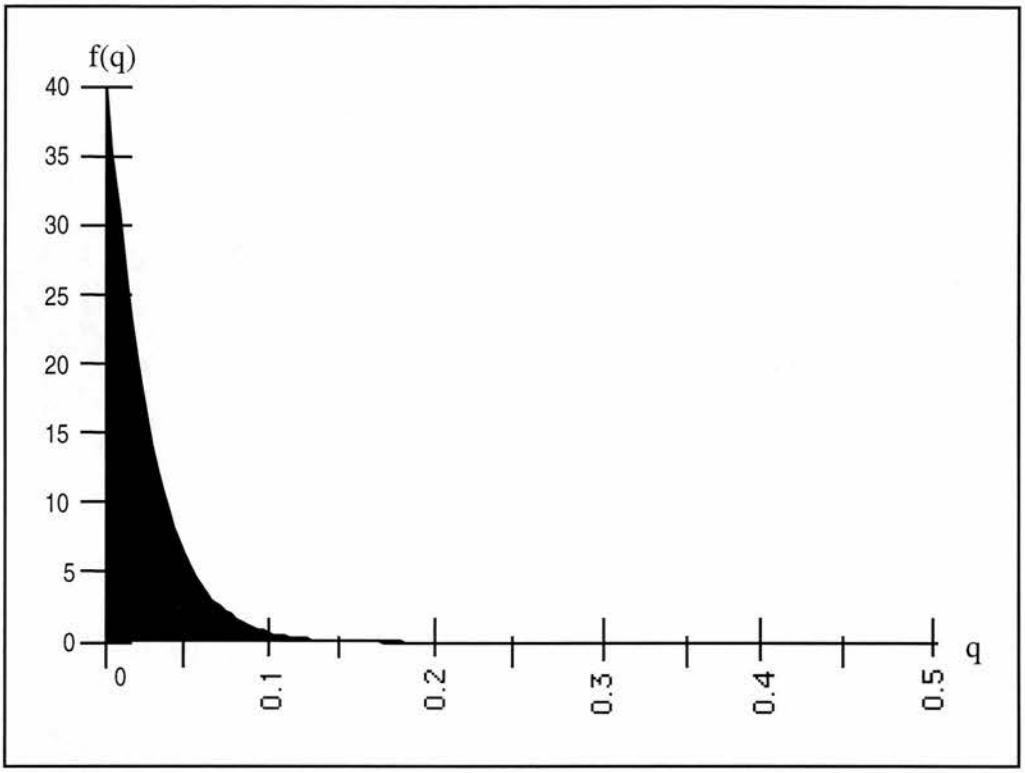

Figure 5.1: The Density Function of Q with Parameters α=1 and β=20, and Its Expected Value is Equal to 1/42.

Adopting a beta distribution with parameters α and β as a prior distribution for 2Q as in (5.3), the expected channel distortion (called ECD) is then calculated in Theorem 5.3.

Theorem 5.3: Let the random variable Q denote the dynamic bit error rate, and 2Q have a prior beta distribution with parameters α and β, For a matched equiprobable standardized scalar source-quantizer pair, the expected channel distortion (ECD) is given by

$$E(D_c(\underline{z},Q)) = \frac{2}{n} \sum_{j=1}^{b} \left(1 - \frac{B(\alpha,\beta+j)}{B(\alpha,\beta)}\right) \left\| Proj_j \underline{z} \right\|^2 \qquad (5.6)$$

Proof of Theorem 5.3:

Recalled from Chapter 3, the channel distortion from Hadamard transformation for a matched equiprobable standardized scalar source-quantizer pair can be rewritten

as a function of bit error rate:

$$D_c(\underline{z},q)=\frac{2}{n}\sum_{j=1}^{b}[1-(1-2q)^j]\left\|\mathrm{Proj}_j\underline{z}\right\|^2,\ \text{for } 0<q<0.5 \qquad (5.7)$$

The expected channel distortion (ECD) with respect to Q is derived using f(q) in (5.2) and $D_c(\underline{z}, q)$ in (5.7):

$$E(D_c(\underline{z},Q))=E(\frac{2}{n}\sum_{j=1}^{b}\left\|\mathrm{Proj}_j\underline{z}\right\|^2(1-(1-2Q)^j))=\frac{2}{n}\sum_{j=1}^{b}\left\|\mathrm{Proj}_j\underline{z}\right\|^2(1-E(1-2Q)^j)$$

$$=\frac{2}{n}\sum_{j=1}^{b}\left\|\mathrm{Proj}_j\underline{z}\right\|^2(1-\int_0^{0.5}(1-2q)^j\cdot f(q)dq)$$

$$=\frac{2}{n}\sum_{j=1}^{b}\left\|\mathrm{Proj}_j\underline{z}\right\|^2(1-\int_0^{0.5}(1-2q)^j\cdot 2\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(2q)^{\alpha-1}(1-2q)^{\beta-1}dq)$$

$$=\frac{2}{n}\sum_{j=1}^{b}\left\|\mathrm{Proj}_j\underline{z}\right\|^2(1-\frac{\Gamma(\alpha+\beta)\Gamma(\beta+j)}{\Gamma(\alpha+\beta+j)\Gamma(\beta)}\int_0^{0.5}2\frac{\Gamma(\alpha+\beta+j)}{\Gamma(\alpha)\Gamma(\beta+j)}(2q)^{\alpha-1}(1-2q)^{\beta+j-1}dq)$$

$$=\frac{2}{n}\sum_{j=1}^{b}\left\|\mathrm{Proj}_j\underline{z}\right\|^2(1-\frac{B(\alpha,\beta+j)}{B(\alpha,\beta)})$$

□

Note that ECD is a real number and is not a function of q. Therefore, an index assignment algorithm using ECD as the distortion measure is independent of bit error rate q and thus can avoid the channel mismatch problem. However, ECD depends upon the prior distribution and its parameters of α and β. Section 5.4 will discuss the properties of ECD as β varies.

5.3 Index Assignment Algorithms Using ECD

In this section, the index assignment algorithms proposed in Chapter 3 are modified by using the distortion measure ECD instead of $D_c(\underline{z},q)$. The corresponding algorithms are the eigenspace index assignment algorithm based on ECD (called EIA-ECD) and the sequential eigenspace index assignment algorithm based on ECD (called SEIA-ECD), respectively. The purpose of these two algorithms is to minimize the ECD and obtain an optimal index assignment for a matched equiprobable standardized scalar source-quantizer pair.

5.3.1 EIA-ECD

We proceed to develop the EIA-ECD algorithm in which $\underline{\mathbf{z}}$ is permuted as nearly into the linear eigenspace E_1 as possible while keeping ECD is as small as possible. The EIA-ECD algorithm is summarized as follows:

Eigenspace Index Assignment Algorithm using ECD (EIA-ECD):

Input: An initial index assignment vector $\underline{\mathbf{z}}^{(0)}$.

Output: A locally optimal index assignment vector, also named $\underline{\mathbf{z}}^{(i)}$.

Step 0: Set an iteration indicator i=0. Choose an initial assignment vector $\underline{\mathbf{z}}^{(i)}$.

Step 1: Regress $\underline{\mathbf{z}}^{(i)}$ on the linear matrix $\mathbf{H}_b$, then compute the linearity index η defined in (3.9), ECD, and the predicted value $\hat{\underline{\mathbf{z}}}^{(i)}$.

Step 2: Whenever ECD increases, the algorithm is stopped and $\underline{\mathbf{z}}^{(i)}$ is a local optimum; Otherwise go to step 3.

Step 3: If $\text{Rank}(\underline{\mathbf{z}}^{(i)}) = \text{Rank}(\hat{\underline{\mathbf{z}}}^{(i)})$ then the algorithm is stopped and $\underline{\mathbf{z}}^{(i)}$ is locally optimal; else sort the elements in $\underline{\mathbf{z}}^{(i)}$ into new $\underline{\mathbf{z}}^{(i+1)}$ according to $\text{Rank}(\hat{\underline{\mathbf{z}}}^{(i)})$, set i=i+1, and go to Step1.

Three properties deserve attentions: (1) Since there are only a finite number of values of η and η is increasing with an upper bound of 1, the algorithm must stop. (2) The EIA-ECD algorithm is independent of the bit error rate q and thus is a mismatch-free algorithm. (3) Empirically, ECD is observed to be almost concave in η, hence a good heuristic is that if ECD starts to increase the algorithm can be stopped. An example of the almost concavity is given in Table 5.2.1.

5.3.2 SEIA-ECD

Step 1 in the EIA-ECD algorithm uses the nxb linear matrix $\mathbf{H}_b = [\underline{\mathbf{h}}_0, \cdots, \underline{\mathbf{h}}_{b-1}]$ in the regression. However, due to the special structure of $\mathbf{H}_b$, the EIA-ECD

algorithm may be performed sequentially by using the b^{th} column of $\mathbf{H}_b$ as the initial regressor, i.e., the regressor matrix $\mathbf{R}$ contains $\underline{\mathbf{h}}_{b-1}$. Then insert vectors $\underline{\mathbf{h}}_i$, i=b-2, b-3, ..., 0 one at a time into $\mathbf{R}$ until the linear matrix $\mathbf{H}_b$ is completely used. This modified version of EIA-ECD is called the sequential EIA-ECD (or SEIA-ECD) and is summarized as follows:

Sequential Eigenspace Index Assignment Algorithm Using ECD (SEIA_ECD):

Input: An initial index assignment vector $\underline{\mathbf{z}}^{(0)}$.

Output: A locally optimal index assignment vector, also named $\underline{\mathbf{z}}^{(i)}$.

Step 0: Set i=0 and j=0. Choose an initial assignment vector $\underline{\mathbf{z}}^{(i)}$; Set the initial regressor matrix to be the $(b-j)^{th}$ column of the linear matrix $\mathbf{H}_b$, say $\mathbf{R}^{(j)} = \underline{\mathbf{h}}_{b-j-1}$.

Step 1: Use linear matrix $\mathbf{H}_b$ to compute η, ECD. Regress $\underline{\mathbf{z}}^{(i)}$ on the regressor matrix $\mathbf{R}^{(j)}$, and compute the predicted value $\hat{\underline{\mathbf{z}}}^{(i)}$.

Step 2: Whenever ECD increases, go to Step 4; Otherwise go to Step 3.

Step 3: If $\text{Rank}(\underline{\mathbf{z}}^{(i)}) = \text{Rank}(\hat{\underline{\mathbf{z}}}^{(i)})$ then go to Step 4; else sort the elements in $\underline{\mathbf{z}}^{(i)}$ into new $\underline{\mathbf{z}}^{(i+1)}$ according to $\text{Rank}(\hat{\underline{\mathbf{z}}}^{(i)})$, set i=i+1, and go to Step1.

Step 4: If $\mathbf{H}_b$ is not completely used, then j=j+1 and add vectors $\underline{\mathbf{h}}_{b-j-1}$ into the current regressor matrix $\mathbf{R}^{(j)}$, i.e. $\mathbf{R}^{(j)}=[\ \underline{\mathbf{h}}_{b-j-1}, \mathbf{R}^{(j-1)}]$ and set i=i+1, go to Step 1; else the algorithm is stopped and $\underline{\mathbf{z}}^{(i)}$ is locally optimal.

As in the EIA-ECD algorithm, the bit error rate q will not affect the result of optimal index assignment for the SEIA-ECD because the distortion measure of ECD is adopted. Again, SEIA-ECD is independent of the initial index assignment vector. Refer to previous proof of Lemma 3.12 in Chapter 3.

5.4 Numerical Results

The voice digitization in North American Telephone Systems (CCITT) is adopted to illustrate the proposed algorithms. The nature binary code (NBC) is used as an initial code. Since a beta distribution in (5.2) with L-shaped density function describes the true behavior of Q, α=1 and β>1 are taken as the possible values of parameters throughout this chapter. Four points emerge from the calculations:

(1) The linearity index η is not necessary "the-bigger-the-better".

(2) Both EIA-ECD and SEIA-ECD algorithms are robust with respect to β.

(3) The same algorithm using the two different criterion, D_c and ECD, for stopping, stop at the same value of η for certain ranges of q_d and β.

(4) A good estimate of β is $\hat{\beta} = \frac{1}{2\bar{q}} - 1$, where $\bar{q}$ denotes the average dynamic bit error rates.

(1) The linearity index η is not necessary "the-bigger-the-better".

As was the case when D_c was used to measure EIA and SEIA algorithms, η is not necessary "the-bigger-the-better" although the algorithm is designed to maximize η. Table 5.2.1 shows the performance of the EIA-ECD in which β=50 and the algorithm is stopped when ECD increases. The values of ECD is almost concave in η (except iterations 17 and 37) and ECD=327452 is minimum when η=0.8438. Therefore, in order to save computational effort, the algorithm can be stopped heuristically whenever ECD begins to increase. Table 5.2.2 shows the performance of SEIA-ECD with β=50. Again ECD is almost concave in η with some minor fluctuation, for example, ECD=332911 in iteration 27 is less than ECD=332930 in iteration 26. However, ECD=327412 is the minimum when η=0.84348 and that indicates the algorithm can be stopped whenever ECD begins to increase. SEIA-ECD gives a slightly better value of ECD than EIA-ECD does. These are similar results to those in Chapter 3, EIA-ECD outperforms NBC by 3.33% and SEIA-ECD

outperforms NBC by 3.34%. Compared to random codes, they both reduce the distortion over 55%.

TABLE 5.2.1: The Performance of EIA-ECD using CCITT Codebook with β=50.

itrn	η	ECD	itrn	η	ECD
0(NBC)	0.8177	338724	:	:	:
1	0.8360	329944	16	0.8537	335334
2	0.8415	327799	17	0.8538	335289
3	**0.8438**	**327452**	:	:	:
4	0.8454	327847	36	0.8577	340113
5	0.8471	328476	37	0.8577	340111
6	0.8483	329000			

TABLE 5.2.2: The Performance of SEIA-ECD using CCITT Codebook with β=50.

itrn	η	ECD	itrn	η	ECD
0(NBC)	0.8177	338724	:	:	:
1	0.8225	336399	26	0.8513	332930
:	:	:	27	0.8514	332911
11	**0.84348**	**327412**	:	:	:
12	0.8445	327734	53	0.8577	340113
:	:	:	54	0.8577	340111

(2) Both EIA-ECD and SEIA-ECD algorithms are robust with respect to β.

It is clear that the designated bit error rate q_d is no longer needed for both EIA-ECD and SEIA-ECD algorithms, instead a final index assignment now depends upon the values of parameters α and β of a prior beta distribution. Therefore, we are interested in how the parameter β affects the result of EIA-ECD or SEIA-ECD when α is fixed as one.

For the CCITT codebook, Table 5.3.1 shows the iterations of EIA-ECD for various β=5,20,50 using NBC as an initial code. All three cases follow the same pattern of iterations since EIA-ECD seeks to maximize η. However, the local optimal solutions will depend upon the values of β. For instance, the two cases of β=20 and

β=50 lead to the same index assignment with η=0.8438; the case of β=5 has different index assignment with η=0.8454. It is also observed that two ECD's differ by only a small amount when their η's are large and close to each other. For instance between iterations 3 and 4, the difference of ECD is only 102 for β=5, 637 for β=20, and 395 for β=50, which are relatively small with respect to their final ECD 2563917, 780609, and 327452. Note that Lemma 3.6 gave a upper bound on the difference between two channel distortions when their η's are the same. In general, the bound is small when η is large.

Table 5.3.1: The Iterations of EIA-ECD for Various β Using CCITT and Initial NBC.

Iteration	η	ECD(β=5)	ECD(β=20)	ECD(β=50)
0(NBC)	0.8177	2629728	806077	338724
1	0.8360	2580224	786439	329944
2	0.8415	2567424	781542	327799
3	**0.8438**	2564019	**780609**	**327452**
4	**0.8454**	**2563917**	781246	327847
5	0.8471	2564608		

NOTES: η denotes linearity index; bold cases are locally optimal.

For a given α=1 the values of β are tested from 4 to 70000 for both EIA-ECD and SEIA-ECD algorithms. In the left part of Table 5.3.2, for β=4 or 5 EIA-ECD results in the same value of η=0.8454; it also results in the same η=0.8438 when β varies from 6 to 70000. The right part of Table 5.3.2 shows that SEIA-ECD results in the same value of η for β varying from 5 to 8, and then for β from 10 to 184, and finally for β from 185 to 70000. Note that all the resulting η's are close to each other for both EIA-ECD and SEIA-ECD, i.e. 0.84348 to 0.8469. These results show that two algorithms are quite robust with respect to the values of parameter β. The advantage of this robustness is that if even the parameter β is estimated with bias, the final index assignment will almost be insensitive to the bias.

TABLE 5.3.2: Robustness of Parameter β for EIA-ECD and SEIA-ECD Using CCITT and Initial NBC.

EIA-ECD			SEIA-ECD		
β	η	ECD	β	η	ECD
4-5	0.8454	3032505-2563840	4	0.8469	3032818
6-70000	0.8438	2222173-0	5-8	0.8447	2563692-1756039
			9	0.8448	1589809
			10-184	0.84348	1452489-91172
			185-70000	0.84352	90633-0

(3) The same value of η for certain ranges of q_d and β.

The channel distortion $D_c(\underline{z},q)$ and the designated bit error rate q_d are required for both EIA and SEIA algorithms while the ECD and the parameter β are needed for both EIA-ECD and SEIA-ECD algorithms when α is fixed. Although it is meaningless to directly compare $D_c(\underline{z},q)$ with ECD, it is interesting that the same algorithm using the two different criterion for stopping, stop at the same value of η for certain ranges of q_d and β.

TABLE 5.4: Robustness of Parameter β for EIA-ECD, SEIA-ECD and Designated Bit Error Rate q_d for EIA, SEIA Using Initial NBC.

EIA	EIA-ECD		SEIA	SEIA-ECD	
q_d	β	η	q_d	β	η
0.11	4-5	0.8454	0.13	4	0.8469
0-0.1	6-70000	0.8438	0.09-0.12	5-8	0.8447
			0.08	9	0.8448
			0.006-0.07	10-184	0.84348
			0-0.005	185-70000	0.84352

Using NBC as an initial code, Table 5.4 illustrates that EIA with q_d varying from 0 to 0.1 has the same η as EIA-ECD with β between 6 and 70000. In addition, SEIA-ECD for β between 5 and 8 has the same η as SEIA for q_d varying from 0.09 to 0.12. Similarly as β varies from 10 to 184, SEIA-ECD has the same η as SEIA for

q_d varying from 0.006 to 0.07. Similarly as β varies from 185 to 70000, SEIA-ECD has the same η as SEIA for q_d varying from 0 to 0.005. These facts also show that both EIA and SEIA are robust with respect to q_d and they are compatible with EIA-ECD and SEIA-ECD. Overall, the resulting η's are all large and similar, i.e., from 0.84348 to 0.8469 such that the resulting channel distortions are different in small amount.

(4) Empirically, the value of β can be chosen as $\hat{\beta} = \frac{1}{2\bar{q}} - 1$, where $\bar{q}$ denotes the average value of the dynamic bit error rates.

In order to apply the proposed index assignment algorithms EIA-ECD and SEIA-ECD, the parameter β for the beta distribution has to be determined when α is fixed. The parameter β can be chosen as any positive value due to the property of robustness. However using a standard statistical estimation method called the method of moments, let $E(Q) = \frac{\alpha}{2(\alpha+\beta)}$ be equal to $\bar{q}$, where $\bar{q}$ is computed by the average value of dynamic bit error rates. Then for a fixed α, β can be estimated empirically by

$$\hat{\beta} = \frac{\alpha}{2\bar{q}} - \alpha . \tag{5.8}$$

For example, if $\bar{q}$ is 0.01, then $\hat{\beta}$ is equal to 49 for α=1.

In Table 5.3, for the CCITT example EIA-ECD has the same index assignment as β varies between 6 and 70000, while from (5.8) $\bar{q}$ is less than 1/14=0.0714. In other words, for all $\bar{q} < 0.0714$ EIA-ECD reaches the same index assignment. In practice, $\bar{q}$ should be small and consequently the EIA-ECD is quite robust. In Table 5.3, β exceeding 10 is equivalent to $\bar{q} < 0.045$, this again shows that SEIA-ECD is quite robust since $\bar{q}$ is small in general.

5.5 Conclusions

This chapter solved the channel mismatch problem with multiple-error pattern in which the bit error rate varies with time. The key idea and approach are summarized as follows:

1. In the channel mismatch problem, the actual bit error rate q may be different from the designated bit error rate q_d and thus its channel distortion $D_c(\underline{\mathbf{z}},q)$ or $D_c(\mathbf{Z},q)$ is not necessarily optimal.
2. The dynamic bit error rate is denoted by a random variable Q, and its behavior can be described by the prior beta distribution in (5.2). Theorem 5.3 developed the expected channel distortion (ECD), a new distortion measure which avoids the channel mismatch problem.
3. Using the ECD instead of $D_c(\underline{\mathbf{z}},q)$, the EIA and SEIA index assignment algorithms in Chapter 3 are modified to be the EIA-ECD and SEIA-ECD mismatch-free algorithms. The key idea is to rearrange the assignment vector as close to linear eigenspace E_1 as possible, while keeping ECD as small as possible (see Table 5.2.1 and Table 5.2.2).
4. For the CCITT example, both EIA-ECD and SEIA-ECD algorithms are robust with respect to the values of parameter β when α is fixed (see Table 5.3.1 and Table 5.3.2).
5. For the CCITT example, EIA-ECD, SEIA-ECD, EIA and SEIA all have similar results (see Table 5.4).
6. Empirically, the value of β can be estimated by $\frac{1}{2\overline{q}}-1$, where $\overline{q}$ is the average value of the dynamic bit error rates.

Finally, the results for the scalar quantizer in this chapter can be easily extended to the case of vector quantizer. Let $\mathbf{Z}=[\underline{\mathbf{z}}_1,\underline{\mathbf{z}}_2,...,\underline{\mathbf{z}}_k]$ denote a k-dimensional equiprobable vector quantizer, the expected channel distortion for vector quantizer (VECD) is then given as

$$E(D_c(\mathbf{Z},Q)) = \frac{2}{kn}\sum_{j=1}^{b}\sum_{r=1}^{k}\left(1-\frac{B(\alpha,\beta+j)}{B(\alpha,\beta)}\right)\left\|Proj_{j}\underline{z}_{r}\right\|^{2}. \qquad (5.9)$$

As in the case of scalar quantizer, a mismatch-free algorithm can be obtained by modifying the optimal index assignment algorithm VEIA proposed in Chapter 4. This uses distortion measure VECD instead of $D_c(\mathbf{Z},q)$.

Chapter 6

Conclusions

Index assignment rearranges the order of codewords appearing in a nonredundant quantized codebook in a way that can combat the degradation of performance caused by channel noise. The binary switch algorithm (BSA) has the best performance in minimizing channel distortion D_c in the comparisons made up to date. However, this algorithm is time consuming and is sensitive to the bit error rate. We propose three fast and robust index assignment algorithms with indistinguishable signal to noise ratio performance to BSA for the binary symmetric channel. The channel mismatch problem is also addressed for dynamic channels where we propose a new measure on performance.

In Chapter 3, the thesis introduces the scalar index assignment algorithm EIA which arranges an index assignment vector to project as much as possible into the linear eigenspace of the largest eigenvalue. Its D_c and SNR performance are shown to be indistinguishable to the BSA on real scalar data – a voice digitization in North American Telephone Systems (CCITT). EIA takes only 0.9 seconds to get an optimal result for CCITT and is also shown to be robust against designated bit error rates varying from low noise 0.0001 to high noise 0.1. Technically, the EIA only requires a regression method and a sorting algorithm, consequently it is very simple, efficient, and robust. Then a sequential eigenspace index assignment algorithm (SEIA), a modification of EIA is proved to be independent of the initial index assignment, and its performance is shown to be similar to that of EIA.

In Chapter 4, the EIA for scalar quantization is extended to a vector eigenspace index assignmnet algorithm (VEIA) for the vector quantization problem. The key idea is that the k-dimensional assignment matrix is transformed orthogonally such

that the first dimension has the largest variance, which is then treated as a scalar quantizer and the EIA is directly applied to obtain a proper index assignment. The VEIA again shows indistinguishable SNR performance to BSA and uses much less computational time in an example involving a first-order Markov-Gauss codebook. The VEIA also shows its robustness in that the same D_c performance is obtained for designated bit error rate from low noise to high noise.

In the dynamic noisy communication channels, the actual value of the bit error rate varies with time and thus the channel mismatrch problem occurs when the measure of performance depends on the bit error rate. The conventional measure D_c depends on the bit error rate, so Chapter 5 proposes a new measure expected channel distortion (ECD) which is independent of the bit error rate so that any index assignment algorithm using ECD is mismatch-free. An ECD measure based on a beta-type density function for channel bit error rate is proposed and tested with the proposed EIA and SEIA algorithms, which are shown to perform well under this measure.

The proposed algorithms and the new measure provide a potentially useful approach for further developments in the source-channel quantization system. Integrating VEIA into quantization systems specifically designed when noisy, memoryless, channels is given by a Beta distribution is a task of the future. Another future study for index assignment will remove the memoryless assumption on channel errors.

Appendix A

Hamming Distance Preserving Index Assignments

For a scalar index assignment problem in Chapter 3, we permute the elements in a codebook into different assignment vectors $\underline{\mathbf{z}}$. Hence there are n! possible assignment vectors in arranging a set of n codewords. Alternatively, we may fix the order of codewords in a codebook, say $C=\{y_0, y_1,..., y_{n-1}\}$, and assign an index to each codeword and forms an index set for a codebook. Hence, there are also n! different index sets. Let $I_A=\{a(i),i=0,1,\ldots,n-1\}$ denote an index set, where a(i) takes on the value of 0, 1,..., n-1. Each index may be denoted by a b-bit binary word, or by a decimal integer for notational brevity. For instance, $I_A=\{11,01,10,00\}=\{3,1,2,0\}$, where a(0)=3(11), a(1)=1(01), a(2)=2(10), a(3)=0(00). Let h(a(i),a(j)) be the hamming distance between a(i) and a(j), which means the number of bit positions in which a(i) and a(j) differ. Let **HD**(I_A)=[h(a(i),a(j)), i,j=0,1,...,n-1] denote a Hamming distance matrix. For a codebook $C=\{y_i, i=0,1,...,n-1\}$, each transmitted codeword y_i is assigned to a(i) and the received signal is decoded to obtain codeword y_j. Therefore, the channel distortion $D_c(\underline{\mathbf{z}},q)$ being a function of $\underline{\mathbf{z}}$ in (3.4) can be rewritten as a function of $\underline{\mathbf{y}}$ and I_A:

$$D_c(\underline{\mathbf{y}},I_A,q)=\frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}q^{h(a(i),a(j))}(1-q)^{b-h(a(i),a(j))}(y_i-y_j)^2=\frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}q_{ij}d(y_i,y_j). \qquad (A.1)$$

From (A.1) we see that for a given codebook $\underline{\mathbf{y}}$ and a given bit error rate q, the channel

distortion is not changed if the hamming distance matrices for different index sets remain the same. Thus, if two index sets preserve the hamming distance matrix, then the channel distortion level is also preserved.

Definition A.1: Two index sets are hamming distance preserving if their associated Hamming distance matrices are the same.

Lemma A.2: Two hamming distance preserving index sets have the same channel distortion.

Proof of Lemma A.2: For any index set, $d(y_i,y_j)$ in (A.1) remains the same for any i and j since the order of the codewords in a codebook is fixed. By Definition A.1, two hamming distance preserving index sets have the same Hamming distance matrix for all i and j. Thus, by (A.1), it is easy to observe that q_{ij} is not changed and consequently the channel distortions remain the same for the hamming distance preserving index sets. □

Lemma A.3: There are n*b! hamming distance preserving index sets which can be obtained by applying two operations: (a) bit permutation (b) bit-wise modulo 2 addition.

Proof of Lemma A.3:

(a) Let the element in an index set I_A be denoted by a b-bit binary word such as $a(i)=(a_{i1},a_{i2},\ldots,a_{ib})$, $a_{ij}=\{0,1\}$, i=0,…,n-1. Let π_{sk} denote the operation of permuting s^{th} and k^{th} binary bits for an index set, say $\pi_{sk}[a(i)]= \pi_{sk}[(\ldots a_{is},\ldots,a_{ik}\ldots)]=(\ldots a_{ik},\ldots,a_{is}\ldots)$, where s,k=1,2,…,b. For instance, $\pi_{23}[(010)]=001$. Now, it is easy to observe that the hamming distance between any two indices remains the same by performing a bit permutation. For instance, $h(\pi_{23}[(010)], \pi_{23}[(001)])=h(001,010)=2=h(010,001)$. In general, for all $(i,j)^{th}$ elements in a Hamming distance matrix, we have $h(a(i),a(j))=h(\pi_{sk}[a(i)], \pi_{sk}[a(j)])$, namely, $HD(I_A) = HD(\pi_{sk}(I_A))$ for s,k=1,2,…,b. There are b! possible bit permutations.

(b) A bit-wise modulo 2 addition is denoted by $\oplus$, for instance, $(001)\oplus(101)=(100)$.

APPENDIX A

The hamming distance between any two indices remains the same by performing a bit-wise modulo 2 addition. For example, h[010⊕(101),001⊕(101)] = h[111,100] = 2 = h[010,001]. In general, for all $(i,j)^{th}$ elements in a Hamming distance matrix, we have h(a(i),a(j))=h(a(i)⊕k,a(j)⊕k), namely, $\mathbf{HD}(I_A) = \mathbf{HD}(I_A \oplus k)$ for k=0,1,…,n-1. There are n possible bit-wise modulo 2 additions.

Consider the representation of the possible indices as vertices of a hypercube in b dimensions. For example the index (0100) is mapped into the vertex (0,1,0,0). The edges of the hypercube connect vertices whose hamming distance is equal to 1. Thus any transformation between two hamming distance preserving, index sets must correspond to transforming the corresponding hypercubes into each other. Thus it is enough to find the number of transformations of a hypercube into itself.

Take any vertex. It can be transformed into any of the n vertices of the hypercube. There are b edges adjacent to this vertex and they can be mapped in b! ways into the b edges adjacent to the transformed vertex. Once one vertex and all its edges are fixed, this defines the hypercube exactly. Hence there are n*b! ways of transforming a hypercube into itself. These are n*b! distinct hamming distance preserving index sets.

Therefore, the above two operations of π and ⊕ will not change Hamming distance matrix, and hence there are n*b! hamming distance preserving index sets for a given codebook. □

Table A.1 lists all eight possible hamming distance preserving index sets for I_A={0,1,2,3} with b=2 and n=4, namely $I_{A'}$={0,1,2,3}, {1,0,3,2}, {2,3,0,1}, {3,2,1,0}, {0,2,1,3}, {1,3,0,2}, {2,0,3,1}, and {3,1,2,0} are all hamming distance preserving index sets, and they all have the same hamming distance matrix $\mathbf{HD}(I_A) = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{bmatrix}$.

Table A.1: All Eight Possible Hamming Distance Preserving Index Sets for $I_A=\{0,1,2,3\}$ with n=4.

⊕(00)		⊕(01)		⊕(10)		⊕(11)	
	Binary Bit		Binary Bit		Binary Bit		Binary Bit
i	**(1,2)***	**i**	**(1,2)**	**i**	**(1,2)**	**i**	**(1,2)**
0	00	1	01	2	10	3	11
1	01	0	00	3	11	2	10
2	10	3	11	0	00	1	01
3	11	2	10	1	01	0	00
i	**(2,1)**	**i**	**(2,1)**	**i**	**(2,1)**	**i**	**(2,1)**
0	00	1	01	2	10	3	11
2	10	3	11	0	00	1	01
1	01	0	00	3	11	2	10
3	11	2	10	1	01	0	00

Note: * numbers in the parenthesis denote bit orders.

Therefore, for a codebook of size n, the n! possible index assignments in an unconstrained search can be reduced by identify n*b! assignments yielding identical distortion. Hence, there exits n!/(n*b!) possible index assignments which confirms the conjecture of full-search index assignment by Knagenhjelm and Agrell [Knagenhjelm and Agrell, 1996]. Table A.2 gives another hamming distance preserving examples of NBC with size n=8, which are obtained from various index assignment algorithms described in Chapter 3. An optimal index assignment with $I_{EIA}=\{5,7,4,6,1,3,0,2\}$ obtained from EIA, $I_{SEIA}=\{4,5,6,7,0,1,2,3\}$ obtained from SEIA, $I_{LISA}=\{3,7,1,5,2,6,0,4\}$ obtained from LISA, and $I_{NBC}=\{0,1,2,3,4,5,6,7\}$ for NBC are mutually hamming distance preserving because they have the same hamming distance matrix $\mathbf{HD}(I_{EIA})=\mathbf{HD}(I_{SEIA})=\mathbf{HD}(I_{LISA})=\mathbf{HD}(I_{NBC})$.

APPENDIX A

Table A.2: Four Hamming Distance Preserving Index Sets and Their Hamming distance Matirx of Size n=8.

HD					j							
	EIA				101	111	100	110	001	011	000	010
		SEIA			100	101	110	111	000	001	010	011
			LISA		011	111	001	101	010	110	000	100
				NBC	000	001	010	011	100	101	110	111
	101	100	011	000	0	1	1	2	1	2	2	3
	111	101	111	001	1	0	2	1	2	1	3	2
	100	110	001	010	1	2	0	1	2	3	1	2
i	110	111	101	011	2	1	1	0	3	2	2	1
	001	000	010	100	1	2	2	3	0	1	1	2
	011	001	110	101	2	1	3	2	1	0	2	1
	000	010	000	110	2	3	1	2	1	2	0	1
	010	011	100	111	3	2	2	1	2	1	1	0

Appendix B

EIA/SEIA

B.1 EIA/SEIA Program

The following program is used in the EIA, SEIA, EIA-ECD, SEIA-ECD algorithms presented in chapters 3, 4, and 5.

```
C EIA/SEIA, DC/EDC, INDEX ASSIGNMENT ALGORITHM USING EIGENSPACE E1
C REGRESS Y ON E1 TO PRODUCE YHAT, AND SWAP CODEBOOK Y
C FILENAME NIND10.F, NATEIA.F  UPDATE 22/7/97, 8/DEC/97

      PROGRAM INDEX_ASSGN
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (KBIT=8,NMAX=2**KBIT,NIN=5,NOUT=6)
      DIMENSION HDM(NMAX,NMAX),INDX(NMAX)
      DIMENSION Y(NMAX),YMEAN(NMAX),PROB(NMAX),YHAT(NMAX)
      DIMENSION SY(NMAX),SYMEAN(NMAX),SPROB(NMAX)
      DIMENSION IY(NMAX),IOPTY(NMAX),IYOPT(NMAX)
      DIMENSION COEF(KBIT),IE1(KBIT),EJ(KBIT)
      DOUBLE PRECISION G05DAF,G01FAF,G01EAF
      DOUBLE PRECISION M01DAF
      EXTERNAL G05DDF   !NORMAL DSTN / G01FAF  !INVERSE NORMAL
      EXTERNAL M01DAF           !RANK
      EXTERNAL M01CAF           !SORT

      IR=0     !IR=0 - DEFAULT/ =1 - RANDOM APDS
      IA=1     !IA=0 - NO IA  / =1 - YES IA
      IDIST=1   !=1 - GAUSSIAN(0,1) / =2 LAPLACIAN

      NBIT=0
      DO 1000 L=8,8     !@@@ NBIT AND N
```

```
      NBIT=NBIT+1
      IBIT=L
      ISEQ=1       !@@@ ISEQ=1 - SEQUENTIAL EIA / =IBIT EIA
      N=2**IBIT
      WRITE(NOUT,5553)
      IF(ISEQ .EQ. 1) THEN
        WRITE(NOUT,5554)L,N
      ELSE
        WRITE(NOUT,5555)L,N
      ENDIF
      CALL HADAMA(IBIT,N,HDM,INDX,IE1)
c     CALL BETADIST(IBIT,COEF)
C     CALL APDS(IBIT,N,E1,IR,Y,YMEAN,PROB,DQ)
      CALL NAT(N,Y,YMEAN,PROB,DQ)
      CALL NORM(N,Y,YMEAN,PROB,YVAR,SY,SYMEAN,SPROB,IY)

      IF(IA .EQ. 0) GOTO 1000
      SDC=1.0D+08
C
      DO 500 INI=1, 2   !@@@ INI=1 -MDC /=2 -FBC /=3 -NBC /=4 -RANDOM
      CALL INITIAL(IBIT,N,Y,IY,INI)
      ITRN=0
      WRITE(NOUT,9998)
      SUBDC=1.0D+08
      DO 450 IREG = ISEQ,IBIT
  100 IFNL=0
      CALL REG(IREG,IBIT,N,HDM,INDX,IE1,COEF,IE1,SY,Y,SYMEAN,
     - YMEAN,YHAT,YVAR,ITRN,SUBDC,IYOPT,IY,ISTOP,IFNL)
      IF(ISTOP .EQ. 0)GOTO 450   !STOP IF NO IMPROVEMENT OF SUBDC
      CALL SWAP(N,Y,YHAT,IY,ISW)
      ITRN=ITRN+1
      IF(ISW .EQ. 1) GOTO 100
  450 CONTINUE                          !IREG LOOP
      CALL UPDATE(N,SY,Y,SDC,IYOPT,SUBDC,IOPTY) !UPDATE OPTIMAL Y
  500 CONTINUE                          !INITIAL LOOP

      ITRN=99                 !FINAL OPTIMAL CODEBOOK
      WRITE(NOUT,7777)
      WRITE(NOUT,6666)(Y(I),I=1,N)
      WRITE(NOUT,9998)
      IFNL=1                  !PRINT FINAL REPORT
```

APPENDIX B

```
      CALL REG(IREG,IBIT,N,HDM,INDX,IE1,COEF,IE1,SY,Y,SYMEAN,
     -  YMEAN,YHAT,YVAR,ITRN,SUBDC,IYOPT,IY,ISTOP,IFNL)

 1000  CONTINUE    !//IBIT LOOP

 5553  FORMAT(/1X,'@@@ INDEX ASSIGNMENT:NIND10.F UPDATE 22/7/97 @@@')
 5554  FORMAT(/1X,'SEQUENTIAL EIA ALGORITHM CASE NUMBER',I3,
     -  ' WITH N =',I4)
 5555  FORMAT(/1X,'EIA ALGORITHM CASE NUMBER',I3,' WITH N =',I4)
 6666  FORMAT(/1X,'FINAL CODEBOOK WITH INDEX ASSIGNMENT FROM 1 TO N'
     -  /(8F9.0))
 7777  FORMAT(/1X,T15,'*** FINAL CODEBOOK REPORT ***')
 9998  FORMAT(/3X,'ITRN',T14,'E1 RATIO',T33,'DC',T47,'IBIT')
      STOP
      END



      SUBROUTINE BETADIST(IBIT,COEF)
C     COMPUTE COEFFICIENTS OF EXPECTED DISTORTION FOR BETA
C     DISTRIBUTION
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION COEF(IBIT)
      ALPHA=1.0
      BETA=10.0
      COEF(1)=BETA/(ALPHA+BETA)
      DO 10 I=1,IBIT
      COEF(I+1)=COEF(I)*(BETA+I)/(ALPHA+BETA+I)
   10  CONTINUE
      WRITE(6,7777)ALPHA,BETA,(COEF(I),I=1,IBIT)
 7777  FORMAT(/1X,'TRANSITION ERROR HAS A BETA(ALPHA,BETA) DISTRIBUTION
     -, WHERE'/' ALPHA=',F5.1,' BETA=',F5.1,3X,
     -'COEFFICIENTS OF EXPECTED CHANNEL DISTORTION ARE',/
     -  (8F10.4))
      RETURN
      END


      SUBROUTINE HADAMA(IBIT,N,HDM,INDX,IE1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION HDM(N,N),INDX(N),IE1(IBIT)
```

APPENDIX B

```
      M = 2              !GENERATE HADAMARD MATRIX
      HDM(1,1) = 1.0
      HDM(1,2) = 1.0
      HDM(2,1) = 1.0
      HDM(2,2) = -1.0
    INDX(1)=0
    INDX(2)=1
    DO 5 K=1,IBIT
     DO 10 I=1,M
      INDX(I+M)=INDX(I)+1
        DO 20 J = 1,M
          HDM(I,J+M) = HDM(I,J)
          HDM(I+M,J ) = HDM(I,J)
          HDM(I+M,J+M) = -HDM(I,J)
20      CONTINUE
10      CONTINUE
     M=M*2
5    CONTINUE
    K=0
    DO 30 J=1,N
    IF(INDX(J) .EQ. 1) THEN
    K=K+1
    IE1(K)=J
    ENDIF
30   CONTINUE
    RETURN
    END


    SUBROUTINE REG(IREG,IBIT,N,HDM,INDX,IE1,COEF,EJ,
   -  Y,SY,YMEAN,SYMEAN,YHAT,YVAR,ITRN,SDC,IY,IYOPT,
   -    ISTOP,IFNL)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    DIMENSION HDM(N,N),INDX(N),IY(N),IYOPT(N)
    DIMENSION Y(N),SY(N),YMEAN(N),SYMEAN(N),YHAT(N)
    DIMENSION EJ(IBIT),COEF(IBIT),IE1(IBIT)
    DIMENSION QE(37),BT(N)
    DATA(QE(I),I=1,37)/0.5,0.475,0.45,0.425,0.4,0.375,0.35,
   - 0.325,0.3,0.275,0.25,0.225,0.2,0.175,0.15,0.125,0.1,
   - 0.075,0.05,0.025,0.01,0.0075,0.005,0.0025,0.001,
   - 0.00075,0.0005,0.00025,0.0001,0.000075,0.00005,
```

```
     - 0.000025,0.00001,0.0000075,0.000005,0.0000025,0.000001/
       QERR=0.13
       DO 3 I=1,N
       Y(I)=SY(IY(I))
       YMEAN(I)=SYMEAN(IY(I))
     3 CONTINUE
       DO 5 J=1,N
       BT(J)=0.0
       DO 10 I=1,N
       BT(J)=BT(J)+Y(I)*HDM(I,J)
    10 CONTINUE
       BT(J)=BT(J)/N
     5 CONTINUE
       DO 15 I=1,IBIT
       EJ(I)=0.0
    15 CONTINUE
       DO 25 I=1,N
       EJ(INDX(I))=EJ(INDX(I))+BT(I)*BT(I)
    25 CONTINUE
       DC=0.0
       DO 35 I=1,IBIT
       DC=DC+2.0*(1.0-(1.0-2.0*QERR)**I)*EJ(I)
    35 CONTINUE
       ETA=EJ(1)/YVAR
       WRITE(6,7777)ITRN,ETA,DC,IREG

       IF(IFNL .EQ. 0)THEN
       DO 45 I=1,N
       YHAT(I)=0.0
       DO 45 K=1,IREG
       KJ=IBIT-K
       YHAT(I)=YHAT(I)+HDM(I,IE1(KJ))*BT(IE1(KJ))
    45 CONTINUE
       IF(DC .LE. SDC)THEN
        DO 55 I=1,N
        IYOPT(I)=IY(I)
    55   CONTINUE
        SDC=DC
        ISTOP=1
       ELSE
        DO 65 I=1,N
```

```
      IY(I)=IYOPT(I)
 65    CONTINUE
      ISTOP=0
     ENDIF
     ELSE
      WRITE(6,8887)          !FINAL REPORT IFNL=1
      TEN=10.0
      D10=DLOG(TEN)
      NQ=37
      DO 75 I=1,NQ           !COMPUTE DC FOR EACH Q
      DC=0.0
      DO 85 J=1,IBIT
      DC=DC+2.0*(1.0-(1.0-2.0*QE(I))**J)*EJ(J)
 85    CONTINUE
      SNR = -10.0*DLOG(DC)/D10
      WRITE(6,8888)I,QE(I),DC,SNR
 75    CONTINUE
     ENDIF
7777   FORMAT(1X,I5,F15.6,D20.8,I8)
8887   FORMAT(/5X,'I',T13,'QERR',T30,'DC',T45)
8888   FORMAT(1X,I5,D12.3,D20.8,F10.2)
     RETURN
     END


     SUBROUTINE SWAP(N,YHAT,Y,IY,ISW)
C     SWAP ELEMENTS IN Y ACCORDING TO THE THEIR RANKS IN YHAT
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     DIMENSION YHAT(N),Y(N),IY(N)
     DIMENSION IRANK(1024),OY(1024)
     DO 5 I=1,N
     IRANK(I)=I
     OY(I)=Y(I)
 5   CONTINUE
     CALL SORT2(N,YHAT,IRANK)
     CALL SORT2(N,Y,IY)
     CALL SORT1(N,IRANK,Y,IY)
     ISW=0
     DO 15 I=1,N
     IF(Y(I) .NE. OY(I))THEN
      ISW=1
```

```
      GOTO 500
     ENDIF
 15  CONTINUE
500  CONTINUE
     RETURN
     END


     SUBROUTINE SORT1(N,IRANK,Y,IY)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     DIMENSION Y(N),IRANK(N),IY(N)
     DO 15 I=1,N-1
     IND=0
     DO 25 J=1,N-I
     IF(IRANK(J) .GT. IRANK(J+1)) THEN
      ITEMP=IRANK(J)
      IRANK(J)=IRANK(J+1)
      IRANK(J+1)=ITEMP
      TEMP=Y(J)
      Y(J)=Y(J+1)
      Y(J+1)=TEMP
      ITEMP=IY(J)
      IY(J)=IY(J+1)
      IY(J+1)=ITEMP
      IND=1
     ENDIF
 25  CONTINUE
     IF(IND .EQ. 0) GOTO 500
 15  CONTINUE
500  CONTINUE
     RETURN
     END


     SUBROUTINE SORT2(N,X,IDX)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     DIMENSION X(N),IDX(N)
     DO 35 I=1,N-1
     IND=0
     DO 45 J=1,N-I
     IF(X(J) .GT. X(J+1)) THEN
```

```
      TEMP=X(J)
      X(J)=X(J+1)
      X(J+1)=TEMP
      ITEMP=IDX(J)
      IDX(J)=IDX(J+1)
      IDX(J+1)=ITEMP
      IND=1
     ENDIF
  45  CONTINUE
      IF(IND .EQ. 0) GOTO 500
  35  CONTINUE
 500  CONTINUE
     RETURN
     END


     SUBROUTINE UPDATE(N,Y,SY,SUBDC,IYOPT,SDC,IOPTY)
C    UPDATE OPTIMAL Y AFTER EACH NEW INITIAL
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     DIMENSION Y(N),SY(N),IYOPT(N),IOPTY(N)
     IF(SUBDC .LT. SDC)THEN
      DO 5 I=1,N
      IOPTY(I)=IYOPT(I)
   5  CONTINUE
     SDC=SUBDC
     ENDIF
     DO 15 I=1,N
     Y(I)=SY(IOPTY(I))
  15 CONTINUE
     RETURN
     END


     SUBROUTINE INITIAL(IBIT,N,Y,IY,INI)
C    GENERATE INITIAL Y FOR EIA
C    INI=1 - MDC / 2 FDC / 3 - NBC / >4 - RANDOM CODE
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     DIMENSION Y(N),IY(N),R1(1),R2(1)
     DIMENSION SY(1024),IYS(1024),IRANK(1024)
     DO 5 I=1,N
      IRANK(I)=I
```

```
      SY(I)=Y(I)
      IYS(I)=IY(I)
    5 CONTINUE
      CALL SORT2(N,SY,IYS)
      IF (INI .EQ. 1)THEN
        CALL MDC(IBIT,N,IRANK)
        WRITE(6,555)
      ELSE
       IF(INI .EQ. 2) THEN
        M=N/2
        DO 10 I=1,M/2
        ITEMP=IRANK(I)
        IRANK(I)=IRANK(M+1-I)
        IRANK(M+1-I)=ITEMP
   10    CONTINUE
        WRITE(6,444)
       ELSE
        IF(INI .EQ. 3) THEN
        WRITE(6,333)
       ELSE
c       A=0.0
c       B=1.0*N
       NSW=N/3
       DO 15 I=1,NSW
       R1=G05DAF(A,B)+1.0
       R2=G05DAF(A,B)+1.0
       K1=R1(1)*N+1.0
       K2=R2(1)*N+1.0
       ITEMP=IRANK(K1)
       IRANK(K1)=IRANK(K2)
       IRANK(K2)=ITEMP
   15  CONTINUE
       WRITE(6,666)
       ENDIF
       ENDIF
      ENDIF

      DO 25 I=1,N              !OBTAIN Y AND IY FROM IRANK
      Y(I)=SY(IRANK(I))
      IY(I)=IYS(IRANK(I))
   25 CONTINUE
```

```
 333  FORMAT(/1X,'------ INI=3, INITIAL IS NATURE BINARY CODE')
 444  FORMAT(/1X,'------ INI=2, INITIAL IS FOLDED BINARY CODE')
 555  FORMAT(/1X,'------ INI=1, INITIAL IS MINIMUM DISTANCE CODE')
 666  FORMAT(/1X,'------ INI=4, INITIAL IS RANDOM CODE')
      RETURN
      END



      SUBROUTINE NORM(N,Y,YMEAN,PROB,YVAR,SY,SYMEAN,SPROB,IY)
C     NORMALIZE Y AND YMEAN, COMPUTE VAR(Y), AND MINIMUM
C     CHANNEL DISTORTION GIVEN A QERR AS ZERO MEAN
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Y(N),YMEAN(N),PROB(N),IY(N)
      DIMENSION SY(N),SYMEAN(N),SPROB(N)
      AVE=0.0          !COMPUTE MEAN AND VARIANCE OF CODEBOOK
      YVAR=0.0
      DO 5 I=1,N
      AVE=AVE+Y(I)
      YVAR=YVAR+Y(I)*Y(I)
    5 CONTINUE
      AVE=AVE/N
      YVAR=(YVAR-AVE*AVE*N)/N
      DO 10 I=1,N              !NORMALIZE CODEBOOK TO ZERO MEAN
      SY(I)=Y(I)-AVE
      SYMEAN(I)=YMEAN(I)-AVE
      SPROB(I)=PROB(I)
      IY(I)=I
   10 CONTINUE
      WRITE(6,7777)YVAR
 7777 FORMAT(/1X,'THE VARIANCE OF CODEBOOK =',D20.8)
      RETURN
      END



      SUBROUTINE MDC(IBIT,N,IMDC)
C     GENERATE MINIMUM DISTANCE CODE, USING HADAMARD INDX
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION IMDC(N),ICOUNT(8)

      IB2=IBIT-2
```

```
      IB3=IBIT-3
      IB4=IBIT-4
      IB5=IBIT-5
      N=2**IBIT
      M=N/2
      IMDC(M)=0
      IMDC(M-1)=1
      I=2
      P=2
      DO  5 K=1,IB2            !**** ONE 1
       IMDC(M-I)=P
        I=I+1
        P=P*2
 5    CONTINUE
      ISUM=0
      II=IB2
      DO 25 ID=0,IB2
         ISUM=ISUM+II    !ICOUNT()=C(IB2,1)....C(1,1)
         II=II-1
         J=I-ISUM
         DO 35 K=1,IB2-ID
           IMDC(M-I)=IMDC(M-J)+2**ID
           I=I+1
           J=J+1
35       CONTINUE
25    CONTINUE
      ICOUNT(0)=0              !** GENERATE FROM TWO 1'S +2**()
      DO 40 J=1, IB3      !ICOUNT(1)=C(IB2-2,2)=1, ICOUNT(2)=C(IB2-1,2)=3
        ICOUNT(J)=ICOUNT(J-1)+J  !ICOUNT(3)=C(IB2,2)
40    CONTINUE
      ISCOUNT=0
      DO 42 J=IB3, 1,-1
        ISCOUNT=ISCOUNT+ICOUNT(J)
      DO 45 K= 1, ICOUNT(J)
        IMDC(M-I)=IMDC(M-I+ISCOUNT)+2**(IB3-J)
        I=I+1
45    CONTINUE
42    CONTINUE
      ICOUNT(1)=1    !C(IB2-2,3)
      ICOUNT(2)=4    !C(IB2-1,3)
      ICOUNT(3)=10   !C(IB2,3)
```

```
      ICOUNT(4)=20
      ISCOUNT=0
      DO 52 J=IB4,1,-1
        ISCOUNT=ISCOUNT+ICOUNT(J)
      DO 55 K=1,ICOUNT(J)
        IMDC(M-I)=IMDC(M-I+ISCOUNT)+2**(IB4-J)
        I=I+1
   55 CONTINUE
   52  CONTINUE
      ICOUNT(1)=1    !C(IB2-*,4)
      ICOUNT(2)=5
      ICOUNT(3)=15
      ISCOUNT=0
      DO 62 J=IB5,1,-1
        ISCOUNT=ISCOUNT+ICOUNT(J)
      DO 65 K=1,ICOUNT(J)
        IMDC(M-I)=IMDC(M-I+ISCOUNT)+2**(IB5-J)
        I=I+1
   65 CONTINUE
   62 CONTINUE

      DO 88 K= 1, IBIT-1              !*** IBIT-2 1'S
      IMDC(K+1)=M-1-2**(K-1)
   88 CONTINUE
      IMDC(1)=M-1                     !*** IBIT-1 1'S
      DO 90 I=1,M
        IMDC(I)=IMDC(I)+1
   90 CONTINUE
      DO 95 I=1,M
        IMDC(M+I)=IMDC(M-I+1)+128
   95 CONTINUE
      RETURN
      END

      SUBROUTINE SORT3(IS,N,INDX,IMDC)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION INDX(N),IMDC(N)
      M=N/2
      IE=IS+M-1
      K=0
      DO 15 I=IS,IE-1
```

```
      IND=0
      K=K+1
      DO 25 J=IS,IE-K
      IF(INDX(J) .GT. INDX(J+1)) THEN
      ITEMP=INDX(J)
      INDX(J)=INDX(J+1)
      INDX(J+1)=ITEMP
      ITEMP=IMDC(J)
      IMDC(J)=IMDC(J+1)
      IMDC(J+1)=ITEMP
      IND=1
      ENDIF
   25 CONTINUE
      IF(IND .EQ. 0) GOTO 500
   15 CONTINUE
  500 CONTINUE
      RETURN
      END


      SUBROUTINE APDS(IBIT,N,HDM,IE1,IR,Y,YMEAN,PROB,DQ)
C     GENERATE ANTI-PODAL-DIRECT-SUM CODEBOOK
C     IR=0 - DEFAULT APDS /  IR=1 - RANDOM APDS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION HDM(N,N),IE1(IBIT),Y(N),YMEAN(N),PROB(N)
      DIMENSION ALPHA(10),RANDA(10),TEMP(1024)
      DATA(ALPHA(I),I=1,8)
     - /14.77,6.01,4.24,5.49,4.61,2.51,-2.32,12.39/
      ITRN=0
      IF(IR .EQ. 0) GOTO 200
      TOL=0.1
      DO 5 I=1,IBIT
      RANDA(I)=ALPHA(I)
   5  CONTINUE

  100  ITRN=ITRN+1
      IF(ITRN .GT. 50) THEN
      PRINT *
      PRINT *,'WARNING: ITRN>50,A DEFAULT QUANTIZER IS ASSUMED'
      GOTO 200
      ENDIF
```

```
      A=-3.0D-00
      B= 3.0D-00
      DO 7 I=1,IBIT/2
      RAND=G05DAF(A,B)
      K=DABS(RAND)/B*IBIT+1.0
      RANDA(K)=RANDA(K)+RAND
    7 CONTINUE

      DO 10 I=1,N
      Y(I)=0.
      DO 15 J=1,IBIT
      Y(I)=Y(I)+HDM(I,IE1(J))*RANDA(J)
   15 CONTINUE
      TEMP(I)=Y(I)
   10 CONTINUE

      IFAIL=0
      DO 30 I=1,N-1
      IF(TEMP(I+1)-TEMP(I) .LT. TOL) GOTO 100
   30 CONTINUE
      WRITE(6,666)ITRN,(RANDA(I),I=1,IBIT)
      WRITE(6,444),TOL
      GOTO 300

  200 DO 35 I=1,N
      Y(I)=0.0
      DO 34 J=1,IBIT
      Y(I)=Y(I)+HDM(I,IE1(J))*ALPHA(J)
   34 CONTINUE
      TEMP(I)=Y(I)
   35 CONTINUE
      IFAIL=0
      SMLD=1.0D+08
      DO 40 I=1,N-1
      DD=TEMP(I+1)-TEMP(I)
       IF(DD .LT. SMLD) THEN
        SMLD=DD
      ENDIF
   40 CONTINUE
      WRITE(6,555)(ALPHA(I),I=1,IBIT)
      WRITE(6,444)SMLD
```

```
      IF(SMLD .LT. 1.0D-06) STOP
  300 EQP=1.0/N
      DO 45 I=1,N
      YMEAN(I)=Y(I)
      PROB(I)=EQP
   45 CONTINUE
      DQ=0.0
      WRITE(6,777)
  444 FORMAT(1X,'MINIMUM DISTANCE BETWEEN CODEWORDS IS .GT. TOL='
     - ,F8.4)
  555 FORMAT(/1X,'IR=0, A DEFAULT APDS WITH COEFFICIENTS'/(5D13.4))
  666 FORMAT(/1X,'IR=1, AN APDS WITH RANDOM COEFFICIENTS','(ITRN=',
     - I2,')'/(5D13.4))
  777 FORMAT(1X,'*** DQ IS ASSUMED TO BE ZERO ***')
      RETURN
      END

      SUBROUTINE NAT(N,Y,YMEAN,PROB,DQ)
C     GENERATE PIECEWISE UNIFORM CODEBOOK USED BY NORTH AMERICAN
C     TELEPHONE COMPANY
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Y(N),YMEAN(N),PROB(N)
      EQP=1.0/256.0
      K=128
      DO 5 IS=1,8
      DO 5 IB=1,16
      K=K+1
      Y(K)=(2.0**(IS-1)*(2.0*(IB-1)+33.0)-33.0)
      Y(257-K)=-Y(K)
    5 CONTINUE
      DO 15 I=1,N
      YMEAN(I)=Y(I)
      PROB(I)=EQP
   15 CONTINUE
      WRITE(6,444)
  444 FORMAT(/1X,'THIS IS THE CODEBOOK USED BY NORTH AMERICAN TELEPHONE
     -COMPANY ')
C     WRITE(6,555)(Y(I),I=1,N)
C 555 FORMAT(8F9.0)
      RETURN
      END
```

B.2 EIA Printout

The following is a printout of eigenspace index assignment algorithm for CCITT with bit error rate=0.001. The program stops whenever D_c increases.

```
@@@ INDEX ASSIGNMENT:IND10.F UPDATE 22/7/97 @@@
  EIA ALGORITHM CASE NUMBER 8 WITH N = 256

THIS IS THE CODEBOOK USED BY NORTH AMERICAN TELEPHONE COMPANY

VARIANCE OF CODEBOOK = 0.6423572D+07
TRANSITION ERROR =  0.1000D-02
THE MINIMUM EXPECTED CHANNEL DISTORTION (L.B.)= 0.256943D+05

------ INI=1, INITIAL IS MINIMUM DISTANCE CODE

 ITRN   E1 RATIO       DC          IBIT
   1    0.348873D+00 53890.7672     8
   2    0.817722D+00 35054.5939     8
   3    0.836027D+00 34107.5711     8
   4    0.841486D+00 33879.8352     8
   5    0.843775D+00 33848.3999     8
   6    0.845412D+00 33900.3507     8

------ INI=2, INITIAL IS FOLDED BINARY CODE

 ITRN   E1 RATIO       DC          IBIT
   1    0.348873D+00 47506.7752     8
   2    0.817722D+00 35054.5939     8
   3    0.836027D+00 34107.5711     8
   4    0.841486D+00 33879.8352     8
   5    0.843775D+00 33848.3999     8
   6    0.845412D+00 33900.3507     8

------ INI=3, INITIAL IS NATURE BINARY CODE

 ITRN   E1 RATIO       DC          IBIT
```

APPENDIX B

1	0.817722D+00	35054.5939	8
2	0.836027D+00	34107.5711	8
3	0.841486D+00	33879.8352	8
4	0.843775D+00	33848.3999	8
5	0.845412D+00	33900.3507	8

------ INI=4, INITIAL IS RANDOM CODE

ITRN	E1 RATIO	DC	IBIT
1	0.272064D+00	81452.7170	8
2	0.848133D+00	34330.4446	8
3	0.850042D+00	34431.5380	8

*** FINAL CODEBOOK REPORT ***

0	0.843775D+00	33848.3999	8

FINAL CODEBOOK WITH INDEX ASSIGNMENT FROM 1 TO N

-8031. -7775. -7519. -7263. -7007. -6751. -6495. -6239.
-5983. -5727. -5471. -5215. -4959. -4703. -4447. -3999.
-4191. -3871. -3743. -3615. -3487. -3359. -3231. -3103.
-2975. -2719. -2463. -2207. -1983. -1855. -1663. -1407.
-2847. -2591. -2335. -2079. -1919. -1791. -1599. -1343.
-1151. -975. -879. -783. -687. -591. -495. -423.
-439. -375. -327. -279. -231. -203. -171. -139.
-115. -93. -81. -69. -57. -45. -30. -20.
-1727. -1471. -1215. -1023. -911. -815. -719. -623.
-559. -455. -391. -343. -295. -247. -211. -179.
-195. -155. -123. -99. -85. -73. -61. -49.
-41. -24. -14. -6. 0. 8. 16. 26.
-37. -22. -12. -4. 2. 10. 18. 28.
33. 53. 65. 77. 89. 107. 131. 163.
147. 187. 219. 263. 311. 359. 407. 471.
527. 655. 751. 847. 943. 1087. 1279. 1535.
-1535. -1279. -1087. -943. -847. -751. -655. -527.
-471. -407. -359. -311. -263. -219. -187. -147.
-163. -131. -107. -89. -77. -65. -53. -33.
-28. -18. -10. -2. 4. 12. 22. 37.
-26. -16. -8. 0. 6. 14. 24. 41.
49. 61. 73. 85. 99. 123. 155. 195.
179. 211. 247. 295. 343. 391. 455. 559.
623. 719. 815. 911. 1023. 1215. 1471. 1727.

APPENDIX B

20.	30.	45.	57.	69.	81.	93.	115.
139.	171.	203.	231.	279.	327.	375.	439.
423.	495.	591.	687.	783.	879.	975.	1151.
1343.	1599.	1791.	1919.	2079.	2335.	2591.	2847.
1407.	1663.	1855.	1983.	2207.	2463.	2719.	2975.
3103.	3231.	3359.	3487.	3615.	3743.	3871.	4191.
3999.	4447.	4703.	4959.	5215.	5471.	5727.	5983.
6239.	6495.	6751.	7007.	7263.	7519.	7775.	8031.

I	QERR	DC
1	0.500D+00	0.12847144D+08
2	0.475D+00	0.12304893D+08
3	0.450D+00	0.11761164D+08
4	0.425D+00	0.11214480D+08
5	0.400D+00	0.10663357D+08
6	0.375D+00	0.10106311D+08
7	0.350D+00	0.95418492D+07
8	0.325D+00	0.89684740D+07
9	0.300D+00	0.83846782D+07
10	0.275D+00	0.77889449D+07
11	0.250D+00	0.71797458D+07
12	0.225D+00	0.65555402D+07
13	0.200D+00	0.59147729D+07
14	0.175D+00	0.52558733D+07
15	0.150D+00	0.45772538D+07
16	0.125D+00	0.38773083D+07
17	0.100D+00	0.31544107D+07
18	0.750D-01	0.24069136D+07
19	0.500D-01	0.16331467D+07
20	0.250D-01	0.83141566D+06
21	0.100D-01	0.33623963D+06
22	0.750D-02	0.25264502D+06
23	0.500D-02	0.16874138D+06
24	0.250D-02	0.84526957D+05
25	0.100D-02	0.33848400D+05
26	0.750D-03	0.25391008D+05
27	0.500D-03	0.16930479D+05
28	0.250D-03	0.84668100D+04
29	0.100D-03	0.33871011D+04
30	0.750D-04	0.25403729D+04
31	0.500D-04	0.16936134D+04

32	0.250D-04	0.84682241D+03
33	0.100D-04	0.33873273D+03
34	0.750D-05	0.25405002D+03
35	0.500D-05	0.16936700D+03
36	0.250D-05	0.84683655D+02
37	0.100D-05	0.33873500D+02

B.3 SEIA Printout

The following is a printout of sequential eigenspace index assignment algorithm for CCITT with bit error rate=0.001. The program stops whenever D_c increases.

@@@ INDEX ASSIGNMENT:IND10.F UPDATE 22/7/97 @@@

SEQUENTIAL EIA ALGORITHM CASE NUMBER 8 WITH N = 256

THIS IS THE CODEBOOK USED BY NORTH AMERICAN TELEPHONECOMPANY

THE VARIANCE OF CODEBOOK = 0.64235721D+07
TRANSITION ERROR = 0.1000D-02

------ INI=1, INITIAL IS MINIMUM DISTANCE CODE

ITRN	E1 RATIO	DC	IBIT
0	0.348873	0.53890767D+05	1
1	0.817722	0.35054594D+05	1
2	0.817722	0.35054594D+05	2
3	0.817722	0.35054594D+05	3
4	0.822580	0.34803808D+05	3
5	0.822580	0.34803808D+05	4
6	0.832489	0.34282223D+05	4
7	0.836703	0.34063089D+05	4
8	0.836703	0.34063089D+05	5
9	0.840297	0.33898017D+05	5
10	0.841945	0.33841181D+05	5
11	0.841945	0.33841181D+05	6
12	0.843480	0.33842559D+05	6
12	0.841945	0.33841181D+05	7
13	0.843523	0.33823124D+05	7
14	0.844706	0.33868823D+05	7

14	0.843523	0.33823124D+05	8
15	0.844780	0.33861940D+05	8

------ INI=2, INITIAL IS FOLDED BINARY CODE

ITRN	E1 RATIO	DC	IBIT
0	0.348873	0.47506775D+05	1
1	0.817722	0.35054594D+05	1
2	0.817722	0.35054594D+05	2
3	0.817722	0.35054594D+05	3
4	0.822580	0.34803808D+05	3
5	0.822580	0.34803808D+05	4
6	0.832489	0.34282223D+05	4
7	0.836703	0.34063089D+05	4
8	0.836703	0.34063089D+05	5
9	0.840297	0.33898017D+05	5
10	0.841945	0.33841181D+05	5
11	0.841945	0.33841181D+05	6
12	0.843480	0.33842559D+05	6
12	0.841945	0.33841181D+05	7
13	0.843523	0.33823124D+05	7
14	0.844706	0.33868823D+05	7
14	0.843523	0.33823124D+05	8
15	0.844780	0.33861940D+05	8

------ INI=3, INITIAL IS NATURE BINARY CODE

ITRN	E1 RATIO	DC	IBIT
0	0.817722	0.35054594D+05	1
1	0.817722	0.35054594D+05	2
2	0.817722	0.35054594D+05	3
3	0.822580	0.34803808D+05	3
4	0.822580	0.34803808D+05	4
5	0.832489	0.34282223D+05	4
6	0.836703	0.34063089D+05	4
7	0.836703	0.34063089D+05	5
8	0.840297	0.33898017D+05	5
9	0.841945	0.33841181D+05	5
10	0.841945	0.33841181D+05	6
11	0.843480	0.33842559D+05	6
11	0.841945	0.33841181D+05	7

12	0.843523	0.33823124D+05	7
13	0.844706	0.33868823D+05	7
13	0.843523	0.33823124D+05	8
14	0.844780	0.33861940D+05	8

------ INI=4, INITIAL IS RANDOM CODE

ITRN	E1 RATIO	DC	IBIT
0	0.272064	0.81452717D+05	1
1	0.817722	0.35054594D+05	1
2	0.817722	0.35054594D+05	2
3	0.817722	0.35054594D+05	3
4	0.822580	0.34803808D+05	3
5	0.822580	0.34803808D+05	4
6	0.832489	0.34282223D+05	4
7	0.836703	0.34063089D+05	4
8	0.836703	0.34063089D+05	5
9	0.840297	0.33898017D+05	5
10	0.841945	0.33841181D+05	5
11	0.841945	0.33841181D+05	6
12	0.843480	0.33842559D+05	6
12	0.841945	0.33841181D+05	7
13	0.843523	0.33823124D+05	7
14	0.844706	0.33868823D+05	7
14	0.843523	0.33823124D+05	8
15	0.844780	0.33861940D+05	8

------ INI=4, INITIAL IS RANDOM CODE

ITRN	E1 RATIO	DC	IBIT
0	0.196989	0.90617746D+05	1
1	0.817722	0.35054594D+05	1
2	0.817722	0.35054594D+05	2
3	0.817722	0.35054594D+05	3
4	0.822580	0.34803808D+05	3
5	0.822580	0.34803808D+05	4
6	0.832489	0.34282223D+05	4
7	0.836703	0.34063089D+05	4
8	0.836703	0.34063089D+05	5
9	0.840297	0.33898017D+05	5
10	0.841945	0.33841181D+05	5

11	0.841945	0.33841181D+05	6
12	0.843480	0.33842559D+05	6
12	0.841945	0.33841181D+05	7
13	0.843523	0.33823124D+05	7
14	0.844706	0.33868823D+05	7
14	0.843523	0.33823124D+05	8
15	0.844780	0.33861940D+05	8

*** FINAL CODEBOOK REPORT ***

FINAL CODEBOOK WITH INDEX ASSIGNMENT FROM 1 TO N

-8031. -7775. -7519. -7263. -7007. -6751. -6495. -6239.
-5983. -5727. -5471. -5215. -4959. -4703. -4447. -4191.
-3999. -3871. -3743. -3615. -3487. -3359. -3231. -3103.
-2975. -2847. -2719. -2591. -2207. -2079. -1855. -1791.
-2463. -2335. -1983. -1919. -1599. -1535. -1215. -1151.
-719. -687. -655. -623. -527. -495. -407. -391.
-247. -231. -203. -195. -171. -163. -123. -115.
-61. -57. -53. -49. -37. -33. -22. -20.
-1727. -1663. -1471. -1407. -1087. -1023. -911. -879.
-591. -559. -471. -455. -375. -359. -311. -295.
-187. -179. -155. -147. -107. -99. -85. -81.
-45. -41. -30. -28. -14. -12. -2. 0.
-18. -16. -6. -4. 8. 10. 24. 26.
65. 69. 73. 77. 89. 93. 131. 139.
211. 219. 263. 279. 327. 343. 423. 439.
751. 783. 815. 847. 943. 975. 1279. 1343.
-1343. -1279. -975. -943. -847. -815. -783. -751.
-439. -423. -343. -327. -279. -263. -219. -211.
-139. -131. -93. -89. -77. -73. -69. -65.
-26. -24. -10. -8. 4. 6. 16. 18.
0. 2. 12. 14. 28. 30. 41. 45.
81. 85. 99. 107. 147. 155. 179. 187.
295. 311. 359. 375. 455. 471. 559. 591.
879. 911. 1023. 1087. 1407. 1471. 1663. 1727.
20. 22. 33. 37. 49. 53. 57. 61.
115. 123. 163. 171. 195. 203. 231. 247.
391. 407. 495. 527. 623. 655. 687. 719.
1151. 1215. 1535. 1599. 1919. 1983. 2335. 2463.
1791. 1855. 2079. 2207. 2591. 2719. 2847. 2975.
3103. 3231. 3359. 3487. 3615. 3743. 3871. 3999.

APPENDIX B

4191. 4447. 4703. 4959. 5215. 5471. 5727. 5983.
6239. 6495. 6751. 7007. 7263. 7519. 7775. 8031.

ITRN	E1 RATIO	DC	IBIT
99	0.843523	0.33823124D+05	8

I	QERR	DC
1	0.500D+00	0.12847144D+08
2	0.475D+00	0.12305054D+08
3	0.450D+00	0.11761476D+08
4	0.425D+00	0.11214924D+08
5	0.400D+00	0.10663906D+08
6	0.375D+00	0.10106930D+08
7	0.350D+00	0.95424988D+07
8	0.325D+00	0.89691089D+07
9	0.300D+00	0.83852518D+07
10	0.275D+00	0.77894109D+07
11	0.250D+00	0.71800617D+07
12	0.225D+00	0.65556697D+07
13	0.200D+00	0.59146902D+07
14	0.175D+00	0.52555669D+07
15	0.150D+00	0.45767306D+07
16	0.125D+00	0.38765985D+07
17	0.100D+00	0.31535729D+07
18	0.750D-01	0.24060401D+07
19	0.500D-01	0.16323694D+07
20	0.250D-01	0.83091207D+06
21	0.100D-01	0.33600698D+06
22	0.750D-02	0.25246642D+06
23	0.500D-02	0.16861953D+06
24	0.250D-02	0.84464624D+05
25	0.100D-02	0.33823124D+05
26	0.750D-03	0.25372008D+05
27	0.500D-03	0.16917784D+05
28	0.250D-03	0.84604480D+04
29	0.100D-03	0.33845528D+04
30	0.750D-04	0.25384613D+04
31	0.500D-04	0.16923387D+04
32	0.250D-04	0.84618491D+03
33	0.100D-04	0.33847770D+03
34	0.750D-05	0.25385874D+03

```
35  0.500D-05     0.16923947D+03
36  0.250D-05     0.84619892D+02
37  0.100D-05     0.33847994D+02
```

Appendix C

VEIA

C.1 VEIA Program

The following program is used in the VEIA algorithm presented in chapter 4.

```
C VQ:TRANSFOR TO Y, APPLY SEIA ON Y1(N), AND ROTATE THE Y(N,K)
c 1-st step markov gaussian source
c for 6 dimension scalar quantization
c FILENAME:VQIND11.F,vqgla.f ,vqgladc.f   UPDATE 2/8/97,31/10/97

      PROGRAM Vquantizer

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER  (IBIT=6,NM=2**IBIT,QERR=0.01)
      PARAMETER  (NIN=5,NOUT=6)
c     PARAMETER  (NDIM=6,LDA=IBIT,LWORK=16*IBIT) !naglib
      PARAMETER  (NDIM=6,LDA=NDIM,LDEVEC=NDIM) !MDIM=NDIM   !IMSL

c     DOUBLE PRECISION YLIST(2048),YALIST(NM),WLIST(NM)
      DOUBLE PRECISION SS(NDIM,NDIM),COR(NDIM,NDIM),COVM(LDA,NDIM)
      DOUBLE PRECISION SYVECT(NM,NDIM)
      DOUBLE PRECISION YVORI(NM,NDIM)
      DOUBLE PRECISION TYMEAN(NDIM),YVMEAN(NDIM),TYVECT(NM,NDIM)
c     DOUBLE PRECISION W(NM), WORK(LWORK)       !nag
      DOUBLE PRECISION EVAL(NDIM),EVEC(LDEVEC,NDIM)          !IMSL

      DIMENSION HDM(NM,NM),INDX(NM),IE1(IBIT),COEF(IBIT)
      DIMENSION EJ(IBIT,NDIM),YHAT(NM)
      DIMENSION IYOPT(NM),IY(NM),IOPTY(NM)
c     EXTERNAL G05DDF              !NORMAL DISTRIBUTION-nag
c     EXTERNAL G05CCF              ! NONREPEATABLE NUMBER-nag
```

APPENDIX C

```
c     EXTERNAL F02FAF, X04CAF        ! EIGENVALUES AND EIGENVECTORS
c     EXTERNAL UMACH
      EXTERNAL DEVCSF,WRRRN          !IMSL EIGENVAL AND VEC,WRITE
      EXTERNAL TDATE

C
      ISEQ= IBIT
      print*, ' first order markov gaussian: '
      print*,' N= ',NM, '  DIM: ', NDIM, ' Q(BER): ',QERR

     WRITE(NOUT,5553)              !PRINT HEADING 'EIA'
     IF ( ISEQ .EQ. 1 ) THEN
       WRITE (NOUT,5554)L,NM
     ELSE
       WRITE (NOUT,5555)L,NM
     ENDIF

C
CCCCCC     VECTOR ( MULTIPLE DIMENSION )
C
c        NM=N/NDIM        !read data from standard input file
        DO 120 I=1, NM
           READ(NIN,*)(YVORI(I,J),J=1,NDIM)
 120     CONTINUE

        DO 125 J=1, NDIM
          YVMEAN(J)=0.0
          DO 130 I= 1, NM
            YVMEAN(J)=YVMEAN(J)+YVORI(I,J)
 130       CONTINUE
          YVMEAN(J)=YVMEAN(J)/NM
 125     CONTINUE

        DO 126 J=1,NDIM    !normalize to zero mean
        DO 126 I=1,NM
          SYVECT(I,J)=YVORI(I,J)-YVMEAN(J)
 126     CONTINUE

c*****    compute covariance matrix *****

        DO 135 I= 1, NDIM  !correlation
```

```
      DO 135 J= 1, NDIM
        SS(I, J)=0.0
      DO 135 K=1, NM
        SS(I,J)=SS(I,J) + SYVECT(K,I)*SYVECT(K,J)
 135    CONTINUE
      DO 155 I=1, NDIM
      DO 155 J=1, NDIM
        COR(I,J) = SS(I,J)/ SQRT( SS(I,I)*SS(J,J) )
 155    CONTINUE
C
      DO 156 I=1, NDIM        !covariance matrix
      DO 156 J=1, NDIM        !!nag- J=1,I!!
        COVM(I,J) = SS(I,J)/NM
 156    CONTINUE
      PRINT*,' MULTIPLE DIMENSION: '
      PRINT*, ' INPUT QUANTIZER; '

      PRINT*,' COR(I,J):
      DO 220 I=1, NDIM
        WRITE (NOUT,9) (COR(I,J),J=1, I)
 220    CONTINUE
c      PRINT*,' VAR-COV MATRIX: '
c      DO 221 I=1, NDIM
c        WRITE (NOUT,9) (COVM(I,J),J=1, I)
c 221    CONTINUE
C
C*****   COMPUTE EIGENVALUES AND EIGENVECTORS

      CALL DEVCSF(NDIM,COVM,LDA,EVAL,EVEC,LDEVEC)
      CALL DWRRRN('EVAL',1,NDIM,EVAL,1,0)
      CALL DWRRRN('EVEC',NDIM,NDIM,EVEC,LDEVEC,0)
      MAXJ=1
C
C

C**************        TRANSFORMATION        *****************
C
C
      DO 310 I=1,NM
      DO 310 J=1,NDIM
        TYVECT(I,J)=0.0
```

```
      DO 310 K=1,NDIM
        TYVECT(I,J) = TYVECT(I,J) + SYVECT(I,K)*EVEC(K,J)
 310    CONTINUE
      WRITE(NOUT,9)
C*******************
      DO 425 J=1, NDIM        !correlation
        TYMEAN(J)=0.0
        DO 430 I= 1, NM
          TYMEAN(J)=TYMEAN(J)+TYVECT(I,J)
 430      CONTINUE
        TYMEAN(J)=TYMEAN(J)/NM
 425    CONTINUE

      DO 435 I= 1, NDIM       !covariance matrix
      DO 435 J= 1, NDIM
        SS(I, J)=0.0
      DO 435 K=1, NM
        SS(I,J)=SS(I,J) + TYVECT(K,I)*TYVECT(K,J)
 435    CONTINUE
      DO 437 I=1, NDIM
      DO 437 J=1, NDIM
        SS(I,J)=SS(I,J) - TYMEAN(I)*TYMEAN(J)*NM
 437    CONTINUE
      DO 455 I=1, NDIM
      DO 455 J=1, NDIM
        COR(I,J) = SS(I,J)/ SQRT( SS(I,I)*SS(J,J) )
 455    CONTINUE
C
      DO 456 I=1, NDIM
      DO 456 J=1, I
        COVM(I,J) = SS(I,J)/NM
 456    CONTINUE

c       PRINT*,' YMEAN: '
c       WRITE (NOUT,9) (TYMEAN(I),I= 1, NDIM)
c       PRINT*,'SS(I,J): '
c       DO 410 I=1, NDIM
c         WRITE (NOUT,9)(SS(I,J),J=1, NDIM)
c 410    CONTINUE
c       PRINT*,' COR(I,J): '
c       DO 420 I=1, NDIM
```

```
c          WRITE (NOUT,9) (COR(I,J),J=1, NDIM)
c 420     CONTINUE
       PRINT*,' VAR-COV MATRIX: '
       DO 421 I=1, NDIM
         WRITE (NOUT,9) (COVM(I,J),J=1, I)
 421     CONTINUE

c********** EIA ************

    CALL EIA(NM,NDIM,TYVECT,,MAXJ)

    9 FORMAT(1X,6E13.6)
5553 FORMAT(/1X,'@@@ INDEX ASSIGNMENT:VQgladc.F UPDATE 28/3/98 @@@')
5555 FORMAT(/1X,'EIA  ALGORITHM NUMBER',I3,' WITH N =',I4)
9998 FORMAT(/3X,'ITRN',T14,'ETA',T33,'DC',T47,'IBIT')
       END
```

C.2 VEIA Printout

The following is a printout of eigenspace index assignment algorithm for vector quantizer. A 6-dimensional vector quantizer of size 64 from the first Gauss-Markov Input with ρ=0 is used. Set bit error rate=0.0001.

```
first order markov gaussian:
 N=  64  DIM:  6 Q(BER):    1.0000000000000D-04

@@@ INDEX ASSIGNMENT:VQgladc.F UPDATE 28/3/98 @@@

EIA  ALGORITHM NUMBER  6 WITH N =  64
 MULTIPLE DIMENSION:
 INPUT QUANTIZER;
 COR(I,J):
 0.100000E+01
-0.303239E-01 0.100000E+01
-0.107892E-01-0.120151E-01 0.100000E+01
 0.110548E+00 0.223040E-02 0.967525E-02 0.100000E+01
-0.290008E+00 0.120146E+00 0.279278E-01-0.536969E-01 0.100000E+01
 0.311937E+00-0.339808E+00 0.882775E-01 0.652934E-02-0.537379E-01 0.100000E+01
```

APPENDIX C

EVAL

1	2	3	4	5	6
1.179	0.835	0.709	0.662	0.615	0.357

EVEC

	1	2	3	4	5	6
1	0.5460	0.3929	0.0945	-0.2407	0.4314	-0.5428
2	-0.4456	0.5517	0.2303	-0.3649	0.3517	0.4326
3	0.0529	-0.2537	0.7234	-0.4926	-0.3959	-0.1007
4	0.1322	0.3640	0.5561	0.7215	-0.1176	0.0799
5	-0.4264	-0.4604	0.3006	0.2131	0.6046	-0.3239
6	0.5488	-0.3638	0.1229	-0.0175	0.3927	0.6299

VAR-COV MATRIX:

0.117907E+01

-0.186483E-15 0.834783E+00

0.364292E-16-0.902056E-16 0.708833E+00

0.398986E-15 0.277556E-16-0.131839E-15 0.662034E+00

-0.304227E-15-0.110155E-15-0.346945E-17 0.442354E-16 0.614761E+00

0.211962E-15-0.119913E-15 0.659195E-16 0.555112E-16 0.119262E-17 0.356517E+00

ITRN	ETA	DC	IBIT
0	0.985448D+00	0.77200425D-03	6
1	0.988923D+00	0.73100426D-03	6
2	0.989517D+00	0.77677912D-03	6
3	0.990197D+00	0.73158907D-03	6
4	0.990274D+00	0.74759754D-03	6

ITRN	ETA	DC	IBIT
99	0.988923D+00	0.73100426D-03	6

I	QERR	DC
1	0.500D+00	0.14519993D+01
2	0.475D+00	0.14268250D+01
3	0.450D+00	0.14001184D+01

4	0.425D+00	0.13715505D+01
5	0.400D+00	0.13407530D+01
6	0.375D+00	0.13073142D+01
7	0.350D+00	0.12707735D+01
8	0.325D+00	0.12306153D+01
9	0.300D+00	0.11862632D+01
10	0.275D+00	0.11370728D+01
11	0.250D+00	0.10823246D+01
12	0.225D+00	0.10212160D+01
13	0.200D+00	0.95285342D+00
14	0.175D+00	0.87624289D+00
15	0.150D+00	0.79028125D+00
16	0.125D+00	0.69374611D+00
17	0.100D+00	0.58528557D+00
18	0.750D-01	0.46340746D+00
19	0.500D-01	0.32646794D+00
20	0.200D-01	0.13970265D+00
21	0.100D-01	0.71460710D-01
22	0.750D-02	0.53902779D-01
23	0.500D-02	0.36141500D-01
24	0.250D-02	0.18174653D-01
25	0.100D-02	0.72949420D-02
26	0.750D-03	0.54743495D-02
27	0.500D-03	0.36516632D-02
28	0.250D-03	0.18268808D-02
29	0.100D-03	0.73100426D-03
30	0.750D-04	0.54828470D-03
31	0.500D-04	0.36554414D-03
32	0.250D-04	0.18278257D-03
33	0.100D-04	0.73115549D-04
34	0.750D-05	0.54836977D-04
35	0.500D-05	0.36558194D-04
36	0.250D-05	0.18279202D-04
37	0.100D-05	0.73117061D-05

Bibliography

Barnes, C.F. and Frost, R.L., "Vector Quantizers with Direct Sum Codebooks," *IEEE Trans. Inform. Theory*, Mar. 1993, Vol. 36, No. 2, pp. 565-580.

Cawley, G.L. and Talbot, N.L.C., "Fast Index Assignment for Vector Quantization over Noisy Transmission Channels," *Electron. Lett.*, 1996, Vol. 32, No. 15, pp. 1343-1344.

Cheng, N.T. and Kingsbury, N.K., "Robust Zero-redundancy Vector Quantization for Noisy Channels," *Proc. IEEE Int. conf. Commun.*, Boston, MA, June 1989, pp. 1338-1342.

Chiang, D. and Potter, L.C., "Minimax Nonredundant Channel Coding," *IEEE Trans. Commun.*, 1995, Vol. 43, No. 2/3/4, pp. 804-811,

Crimmins, T.R., Horwitz, H.M., Palermo, C.J., and Palermi, R.V., "Minimization of Mean-Squared Error for Data Transmitted via Group Codes," *IEEE Trans. Inform. Theory*, Jan. 1969, Vol. IT-15, No.1, pp. 72-78.

Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

DeMarca, J., and Jayant, N., "An algorithm for assigning binary indices to the codevectors of a multi-dimensional quantizer," *Proc. IEEE Int. Conf. Commun.*, Seattle, WA, June 1987, pp.1128-1132.

Farvardin, N., "A Study of Vector Quantization for Noisy Channels," *IEEE Trans. on Inform. Theory*, 1990, Vol.36, No.4, pp. 799-809.

Gersho, A. and Gray, R.M., *Vector Quantization and Signal Compression*, Boston, MA, Kluwer, 1992.

Goldberg, D.E., *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, 1989.

Goodman, D. and Moulsley, T.J., "Using Simulated Annealing to Design Transmission Code for Analogue Sources," *Electron. Lett.,* May 1988, Vol. 24, No. 10, pp. 617-618.

Goodrich, M.T. and Tamassia, R., *Data Structures and Algorithms in Java*, John Wiley and Sons, Inc., New York, 1998

Hall, K., "An R-dimensional Quadratic Placement Algorithm," *Management Science*, 1970, Vol. 77, No. 3, pp. 219-229.

Hamming, R.W., *Coding and Information Theory*, Second Edition, Prentice-Hall, N.J., 1980

Johnson, N.L. and Kotz, S., *Distribution in Statistics*, Houghton Mifflin Company, Boston, 1970.

Johnson, R.A. and Wichern, D.W., *Applied Multivariate Statistical Analysis*, Second Edition, Prentice-Hall International, London, 1988.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by Simulated Annealing," *Science*, May 1983, Vol. 220, pp. 671-680.

Knagenhjelm, P., "How Good Is Your Index Assignment?" in *Proc. IEEE Int. Conf. On Acoustic Speech, and Signal Processing,* Minneapolis, MN, Apr. 1993, Vol. II, pp. 423-426.

Knagenhjelm, P. and Agrell, E., "The Hadamard Transform - A Tool for Index Assignment," *IEEE Trans. Inform. Theory*, July 1996, Vol. 42, No. 4, pp. 1139-1151.

Linde, Y., Buzo, A., and Gray, R.M. "An Algorithm for Vector Quantizer Design," *IEEE Trans. Commun.*, , Jan. 1980, Vol. COM-28, No. 1, pp. 84-95.

Lindgren, B.W., *Statistical Theory*, 3rd edition, Macmillan, New York, 1976.

Lint, J.H., *Introduction To Coding Theory*, 2nd edition, Springer-Verlag, London, 1992.

Lint, J.H. and Wilson, R.M., *A Course in Combinatorics*, Cambridge University Press, 1992.

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

Max, J., "Quantizing for Minimum Distortion," *IRE Trans. on Inform. Theory*, IT-6, pp. 7-12, March 1960.

McLaughlin, S.W., Neuhoff, D.L., and Ashley, J.J., "Optimal Binary Index Assignments for a Class of Equiprobable Scalar and Vector Quantizers," *IEEE Trans. Inform Theory*, Jan. 1995, Vol. 41, No. 6, pp. 2031-2037.

Messerschmitt, David G., "Quantizing for Maximum Output Entropy," *IEEE Trans. Inform. Theory*, Sep. 1971, Vol. IT-17, No.5, p. 612.

Messerschmitt, David G., "Accumulation Distortion in TANDEM Communication Links," *IEEE Trans. Inform. Theory*, Nov. 1979, Vol. IT-25, No.6, pp. 692-698.

Noble, B. and Daniel, J.W., *Applied Linear Algebra*, 3rd edition, Englewood Cliffs, N.J., Prentice Hall, Inc., 1988

Ostrowski, T and Ruoppila, V.T., " Genetic Annealing Search for Index Assignment in Vector Quantization," *Pattern Recognition Lett.*, 1997, Vol.18, No.4, pp. 311-318,

Pan, J.S., McInnes, F.R., and Jack, M.A., "Application of Parallel Genetic Algorithm and Property of Multiple Global Optima to VQ Codevector Index Assignment for Noisy Channels," *Electronics Lett.*, 1996, Vol.32, No. 4, pp. 296-297.

Rydbeck, N. and Sundberg, C.W., "Analysis of Digital Errors in Nonlinear PCM Systems," *IEEE Trans. on Commun.*, Jan. 1976, Vol. COM-24, No.1, pp. 59-65.

Totty, R.E. and Clark, G.C., "Reconstruction error in Waveform Transmission," *IEEE Trans. Inform. Theory*, Apr. 1967, Vol. IT-13, pp.336-338.

Wu, H.S. and Barba, J., "Index Allocation in Vector Quantization for Noisy Channels," *Electro. Lett.*, July 1993, Vol. 29, No. 15, pp. 1317-1319.

Yan, H. and Donaldson, R.W., "Subjective Effects of Channel Transmission Errors on PCM and DPCM Voice Communication Systems," *IEEE Trans. Commun.*, 1972, Vol. COM-20, pp.281-290.

Zeger, K.A. and Gersho, A., "Zero Redundancy Channel Coding in Vector Quantization," *Electro. Lett.*, Jun. 1987, Vol. 23, No. 12, pp. 654-655.

Zeger, K.A. and Gersho, A., "Pseudo-Gray Coding," *IEEE Trans. Commun.*, Dec. 1990, Vol. 38, No. 12, pp. 2147-2158.

Zeger, K.A. and Manzella, V., "Asymptotic Bounds on Optimal Noisy Channel Quantization Via Random Coding," *IEEE Trans. Inform. Theory*, 1994, Vol. 40, No. 6, pp.1926-1939.