Sample Adaptive Product Quantization for Memoryless Noisy Channels

by

Zahir Raza

A thesis submitted to the
Department of Mathematics and Statistics
in conformity with the requirements
for the degree of Master of Science (Engineering)

Queen's University
Kingston, Ontario, Canada
November, 2002

Abstract

Channel optimized vector quantization (COVQ), as a joint source-channel coding scheme, has proven to perform well in compressing a source and making the resulting quantizer codebook robust to channel noise. Unfortunately like its counterpart in the noiseless channel case, the vector quantizer (VQ), the COVQ encoding complexity is inherently high. Sample adaptive product quantization was recently introduced by Kim and Shroff to reduce the complexity of the VQ while achieving comparable distortions, even for moderate quantization dimensions. In this thesis, we investigate the SAPQ for the case of noisy memoryless channels and employ the joint source-channel approach of optimizing the quantizer design by taking into account both source and channel statistics. It is shown that, like its counterpart in the noiseless case, the channel optimized SAPQ achieves comparable performance results to the COVQ (within 0.2-0.8 dB), while maintaining considerably lower encoding complexity (half of that of COVQ) and storage requirements.

Acknowledgments

I would foremost like to thank God for everything that He has granted me with His mercy. I would like to thank Dr. Fady Alajaji and Dr. Tamás Linder for all their patience, guidance and valuable help without which I would not have been able to complete this thesis. I would like to thank all my fellow colleagues from the Queen's University Math & Stats Communications Lab namely: Firouz Behnamfar, Ziad Rached, András György, Guangchong Zhu and all the others. I would also like to thank my family and friends. To all that were mentioned, implicitly or directly, I couldn't have done it without you. Thanks !

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Joint Source-Channel Coding

Recently, Kim and Shroff introduced in [11, 12] a constrained vector quantizer structure called the **sample adaptive product quantizer** (SAPQ) that achieves comparable performances to the **vector quantizer** (VQ) [25, 1, 2, 3, 4] while maintaining a lower encoding complexity (refer also to [13, 14, 23, 24, 29] for previous related work).

Yet, as with most data compression schemes that solely remove source redundancy, the compressed source tends to be more sensitive to channel noise. Traditionally, **tandem source-channel coding** was used to achieve reliable transmission of information by separately designing the source and channel codes. Independently designing the source and channel encoder is justified by **Shannon's Separation principles**, [35, 36]. Shannon's separation principles state that with a noisy channel of **capacity** C it is possible to obtain a reconstruction signal with fidelity, or **distortion**, D, provided that the capacity C is greater than R(D), i.e C > R(D); where R(D) is **the rate-distortion function**.

However the above result assumes the availability of unlimited coding/decoding delay and complexity. It is known that when there are delay and complexity constraints, it is more advantageous to employ **joint source-channel coding** where the source and channel codes are designed in cohesion (e.g., [5], [15]–[17],[20, 22], [30, 31, 37]). There are three main approaches to joint source-channel coding: the **unequal error protection** approach, the **zero-redundancy channel coding** approach, and the **combined source-channel coding** approach.

In the unequal error protection approach, the output of the source encoder is given unequal protection based on the effect of their error in the reconstructed sequence. Work related to this method includes that of [28], where Modestino and Daut use 2D-PCM as the source encoder and provide selective error control protection on those bits which contribute most significantly to the image reconstruction. Unequal error protection is also used in digital wireless communication systems such as the Global System for Mobile (GSM) communications [32].

In zero-redundancy channel coding, channel coding is removed and instead residual redundancy of the source encoder output is used to combat noise. In [33] residual redundancy in DPCM systems was studied and used to propose a zero-redundancy channel coding system that significantly improved gains over other tandem systems with the expense of higher complexity at the decoder.

In this thesis, we focus on the third approach, where both channel noise and source statistics are included in the design of the source coders. VQ's designed in such a way are labeled **channel optimized vector quantizers** (COVQ's). COVQ has received a considerable amount of attention due to its improvement in performance over VQ in the presence of channel noise (e.g., [16, 30]). However, COVQ still incurs high encoding complexity. In this thesis, we study the design of SAPQ for noisy memoryless channels, or channel optimized SAPQ (CO-SAPQ), in order to find a less complex alternative to COVQ.

1.2 Contributions

The contributions of this thesis are as follows:

1. The sample adaptive product quantizer (SAPQ) design of Kim and Shroff [11] is generalized to include channel statistics. In this way, this thesis is intended to channel optimize the design of the SAPQ resulting in the so called channel optimized sample adaptive product quantizer (COSAPQ). Countering [11] we will design and implement two channel optimized SAPQs, namely the COm-SAPQ and the CO1-SAPQ.

2. This thesis includes numerical results produced in order to compare the performances of the COSAPQ against the COVQ and the channel optimized product quantizer (COPQ). Through the numerical results we illustrate the advantages of using a COm-SAPQ for memoryless Gaussian sources, and a CO1-SAPQ for Gauss-Markov (correlated) sources.

1.3 Thesis Outline

The rest of this thesis is organized as follows:

In Chapter 2, we study traditional quantizers such as vector quantizer (VQ) and product quantizer (PQ). Furthermore we study their channel optimized extensions, namely channel optimized vector quantizer (COVQ) and channel optimized product quantizer (COPQ). Necessary conditions for optimality, and design algorithms are derived for each of the quantizers.

In Chapter 3, we introduce the sample adaptive product quantizer (SAPQ). We study the model of SAPQ and derive its necessary conditions for optimality. The encoding complexity and storage requirements are then studied for SAPQ, VQ, and PQ. These encoding complexities and storage requirements of SAPQ, VQ and PQ are then compared. The advantages of SAPQ are highlighted in the numerical results section.

In Chapter 4, we introduce the channel optimized sample adaptive product quantizer (COSAPQ), and study its model and necessary conditions for optimality in detail. We investigate techniques to further reduce the encoding complexity of COSAPQ using encoding simplifications illustrated in [16]. Then the encoding complexities and storage requirements of COSAPQ, COVQ, and COPQ are studied and compared too, with their performances under various channel conditions.

Finally we conclude the thesis in Chapter 5 with a summary of the work done and discussion on future related work.

Chapter 2

Channel Optimized Quantization

2.1 Vector Quantizers (VQ)

A vector quantizer (VQ) is a lossy data compression system based on the principle of block coding. In 1980, Linde, Buzo, and Gray (LBG) [25] proposed a VQ design algorithm based on two necessary conditions for quantizer optimality. In this section we will review the basic concepts of vector quantization and the LBG algorithm. Note that, throughout this thesis we will use the following notation: given any natural number $n \in \mathbb{N}$ then J_n is defined as the set

$$\mathrm{J}_n = \{1, \ldots, n\}.$$

2.1.1 VQ Model

A (k,N) vector quantizer (VQ) is a mapping from the Euclidean space $\mathbb{R}^k$ to a finite set $\mathrm{C} = \{\underline{c}_i\}_{i=1}^{N}$, of N elements. The set C is referred to as the codebook and its elements $\underline{c}_i$ are called codevectors, hence

$$\mathrm{VQ} : \mathbb{R}^k \rightarrow \mathrm{C} \quad \text{where} \quad \mathrm{C} = \{\underline{c}_i\}_{i=1}^{N} \subset \mathbb{R}^k. \tag{2.1}$$

A source sample vector $\underline{\mathrm{x}} \in \mathbb{R}^k$ is quantized, or approximated, by a (k,N) vector quantizer by mapping the source sample vector $\underline{\mathrm{x}}$ into one of the elements in the codebook C. The mapping of source sample vectors $\underline{\mathrm{x}} \in \mathbb{R}^k$ into codevectors is done using the N partition regions of $\mathbb{R}^k$ formed by the (k,N) VQ. A partition region S_i of the (k,N) VQ is defined as

$$\mathrm{S}_i = \{\underline{\mathrm{x}} \in \mathbb{R}^k : \mathrm{VQ}(\underline{\mathrm{x}}) = \underline{c}_i\}$$

for $i = 1, \ldots, N$. These partition regions $\{\mathrm{S}_i\}$ are also referred to as encoding regions since we can define an encoding function (E) as

$$\mathrm{E}(\underline{\mathrm{x}}) = i \quad \text{if and only if} \quad \underline{\mathrm{x}} \in \mathrm{S}_i$$

for $i = 1, \ldots, N$. Then a corresponding decoding function (G) needs to be defined as

$$\mathrm{G}(i) = \underline{c}_i \quad \text{for} \quad i \in \mathrm{J}_N = \{1, \ldots, N\}.$$

Hence the VQ mapping of sample vectors $\underline{\mathrm{x}} \in \mathbb{R}^k$ into codevectors can be broken down into a composite of functions: the encoding function (E) and decoding function (G) as

$$\mathrm{VQ}(\underline{\mathrm{x}}) = \mathrm{G}\left(\mathrm{E}(\underline{\mathrm{x}})\right).$$

The encoding is governed by the fidelity criteria or distortion measure used. A distortion measure is a function that assigns a non-negative number to any two elements $\underline{\mathrm{x}}$ and $\underline{\hat{\mathrm{x}}}$ of $\mathbb{R}^k$

$$d(\underline{\mathrm{x}}, \underline{\hat{\mathrm{x}}}) \geq 0 \quad \text{where} \quad \underline{\mathrm{x}}, \underline{\hat{\mathrm{x}}} \in \mathbb{R}^k.$$

In this thesis we will only concentrate on the square-error distortion measure

$$d(\underline{\mathrm{x}}, \hat{\underline{\mathrm{x}}}) = ||\underline{\mathrm{x}} - \hat{\underline{\mathrm{x}}}||^2 \quad \text{where} \quad \underline{\mathrm{x}}, \hat{\underline{\mathrm{x}}} \in \mathbb{R}^k.$$

With a (k,N) VQ, each sample vector $\underline{\mathrm{x}} \in \mathbb{R}^k$ can be represented by one of the N codevectors in codebook C, hence the rate or the resolution of a (k,N) VQ is

$$R = \frac{\log_2 N}{k} \text{ bits/source sample.} \tag{2.2}$$

2.1.2 VQ Necessary Conditions for Optimality

An optimal VQ is one that minimizes the expected distortion subject to a rate constraint.

Distortion: For a source $\underline{\mathrm{X}}$ with a probability density function $\mathrm{p}(\underline{\mathrm{x}})$, the expected squared-error distortion incurred by a (k,N) VQ with codebook $\mathrm{C} = \{\underline{c}_i\}_{i=1}^N$ and partition regions $\{\mathrm{S}_i\}_{i=1}^N$ is

$$\begin{aligned} \mathrm{D}_{VQ} &= E\{d(\underline{\mathrm{X}}, \underline{\mathbf{c}})\} && (2.3) \\ &= \sum_{i=1}^{N} E\{d(\underline{\mathrm{X}}, \underline{c}_i)|\underline{\mathrm{X}} \in \mathrm{S}_i\}\mathrm{P}(\underline{\mathrm{X}} \in \mathrm{S}_i) && (2.4) \\ &= \sum_{i=1}^{N} \int_{\mathrm{S}_i} ||\underline{\mathrm{x}} - \underline{c}_i||^2 \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}}. && (2.5) \end{aligned}$$

From the expected squared-error distortion (2.5) of the VQ, the optimal codevectors of the codebook $\mathrm{C} = \{\underline{c}_i\}_{i=1}^N$ and the optimal encoding regions $\{\mathrm{S}_i\}_{i=1}^N$ need to be determined. Unfortunately optimality conditions for a VQ are not known. The necessary conditions for optimality are, on the other hand, known. The necessary conditions are divided into the optimal encoding condition and the optimal decoding condition.

Optimal Encoding: To find the optimal encoding condition one has to consider the following: Given the optimal codebook $\mathrm{C} = \{\underline{c}_i\}_{i=1}^N$, how can we determine the optimal encoding regions $\{\mathrm{S}_i\}_{i=1}^N$, of a (k,N) VQ, so as to minimize the expected square-error distortion (2.5)?

Consider a (k,N) VQ with codebook C and a source sample $\underline{\mathrm{x}} \in \mathbb{R}^k$ and

$$\mathrm{VQ}(\underline{\mathrm{x}}) = \underline{\mathrm{y}} \;\in \mathrm{C}.$$

Then

$$\begin{aligned} d(\underline{\mathrm{x}}, \underline{\mathrm{y}}) &= \|\underline{\mathrm{x}} - \underline{\mathrm{y}}\|^2 \\ &\geq \min_{\underline{\mathrm{y}} \in \mathrm{C}} \|\underline{\mathrm{x}} - \underline{\mathrm{y}}\|^2. \end{aligned}$$

In other words the VQ mapping that minimizes (2.5) is

$$\mathrm{VQ}(\underline{\mathrm{x}}) = \arg\min_{\underline{\mathrm{y}} \in \mathrm{C}} \|\underline{\mathrm{x}} - \underline{\mathrm{y}}\|^2.$$

Hence the optimal encoding regions can be derived to be

$$\mathrm{S}_i = \{\underline{\mathrm{x}} \in \mathbb{R}^k : \|\underline{\mathrm{x}} - \underline{c}_i\|^2 \leq \|\underline{\mathrm{x}} - \underline{c}_j\|^2 \quad \forall j \in \mathrm{J}_N\} \tag{2.6}$$

for $i = 1, \ldots, N$, and thus the optimal encoding function is

$$\mathrm{E}(\underline{\mathrm{x}}) = \arg\min_{i \in \mathrm{J}_N} \|\underline{\mathrm{x}} - \underline{c}_i\|^2. \tag{2.7}$$

Optimal Decoding: Now assuming the set of encoding regions $\{\mathrm{S}_i\}_{i=1}^N$ of a (k,N) VQ are given, then the set of optimal codevectors of the codebook C can be derived from the distortion (2.5). This is done by separating the l^{th} codevector $\underline{c}_l$ of the codebook C, from the expected squared-error distortion of the (k,N) VQ

$$\begin{aligned} \mathrm{D}_{VQ} &= \sum_{i=1}^N \int_{\mathrm{S}_i} \|\underline{\mathrm{x}} - \underline{c}_i\|^2 \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}} \\ &= \int_{\mathrm{S}_l} \|\underline{\mathrm{x}} - \underline{c}_l\|^2 \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}} + \sum_{i \neq l} \int_{\mathrm{S}_i} \|\underline{\mathrm{x}} - \underline{c}_i\|^2 \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}}. \end{aligned}$$

Taking the partial derivatives with respect to $\underline{c}_l$ and setting the result to zero, we get

$$0 = \int_{S_l} \{-\underline{x} + \underline{c}_l\} p(\underline{x}) d\underline{x}.$$

Solving for $\underline{c}_l$ we get the optimal decoding condition

$$\underline{c}_l = \frac{\int_{S_l} \underline{x} p(\underline{x}) d\underline{x}}{\int_{S_l} p(\underline{x}) d\underline{x}}. \tag{2.8}$$

The optimal codevectors $\{\underline{c}_l\}_{l=1}^N$ defined by (2.8) are referred to as centroids since

$$\underline{c}_l = \arg \min_{\underline{y} \in \mathbb{R}^k} E\{\|\underline{X} - \underline{y}\|^2 | \underline{X} \in S_l\} \quad \text{for } l = 1, \ldots, N.$$

2.1.3 LBGVQ Design Algorithm

The obvious approach to designing a suboptimal (k,N) VQ is to iterate through the necessary conditions of optimality until the distortion of the resulting (k,N) VQ converges to a value within a prescribed threshold. Almost all quantizers are designed using such an iterative algorithm. Their difference lies in their starting points, or initializations. In [25] Linde, Buzo, and Gray (LBG) studied the VQ and proposed an algorithm to design a (k,N) VQ, with a particular starting point. This LBG algorithm will be used throughout this thesis.

(k,N) LBG-VQ algorithm

1. Set parameters k, N, the stopping threshold δ, the splitting constant ϵ, the maximum number of iterations $Maxiter$, and M the total number of training vectors $\{\underline{x}_f\}_{f=1}^M$. Start off with $\tau = 1$, $\rho = 0$, and

$$\underline{c}_1^{(0)} = \frac{1}{M} \sum_{f=1}^{M} \underline{x}_f$$

as the initial codevector and

$$\mathcal{D}^{(0)}[1] = \frac{1}{kM}\sum_{f=1}^{M} \|\underline{\mathrm{x}}_f - \underline{c}_1^{(0)}\|^2$$

as the initial mean squared-error distortion.

2. If $\tau \geq N$ stop otherwise split each codevector using

$$\underline{c}_i^{(\rho)} = \underline{c}_i^{(\rho)}(1+\epsilon) \quad \text{and} \quad \underline{c}_{i+\tau}^{(\rho)} = \underline{c}_i^{(\rho)}(1-\epsilon)$$

for $i = 1, \ldots, \tau$, then double $\tau = \tau * 2$ and set $\rho = 0$. Note that ρ is a counter for the iterations and τ is a counter for the number of codevectors. At this point we have τ codevectors $\mathrm{C}^{(\rho)} = \{\underline{c}_i^{(\rho)}\}_{i=1}^{\tau}$.

3. For each f, encode $\underline{\mathrm{x}}_f$ into an index using (2.7) as

$$\mathrm{E}^{(\rho)}(\underline{\mathrm{x}}_f) = \arg\min_{i \in \mathrm{J}_\tau} \|\underline{\mathrm{x}} - \underline{c}_i^{(\rho)}\|^2$$

where $\mathrm{J}_\tau = \{1, \ldots, \tau\}$.

4. Once $\underline{\mathrm{x}}_f$ has been encoded, put $\underline{\mathrm{x}}_f$ into the appropriate partition cell (2.6). So if $\underline{\mathrm{x}}_f$ was encoded into $\mathrm{E}^{(\rho)}(\underline{\mathrm{x}}_f) = i^* \in \mathrm{J}_\tau$, then

$$\underline{\mathrm{x}}_f \in \mathrm{S}_{i^*}^{(\rho)}$$

and the resulting distortion is then

$$D^{(\rho)}[\underline{\mathrm{x}}_f, \tau] = \|\underline{\mathrm{x}}_f - \underline{c}_{i^*}^{(\rho)}\|^2.$$

5. Repeat steps 3 and 4 for all $f = 1 \ldots, M$. Once all the partitions have been made, update the codebook $\mathrm{C}^{(\rho)}$ using

$$\underline{c}_l^{(\rho)} = \frac{\sum_{\underline{\mathrm{x}}_f : \underline{\mathrm{x}}_f \in \mathrm{S}_l^{(\rho)}} \underline{\mathrm{x}}_f}{\sum_{\underline{\mathrm{x}}_f : \underline{\mathrm{x}}_f \in \mathrm{S}_l^{(\rho)}}}, \quad \text{for } l = 1, \ldots, N.$$

Finally calculate the overall distortion using

$$\mathcal{D}^{(\rho)}[\tau] = \frac{1}{kM}\sum_{f=1}^{M} D^{(\rho)}[\underline{\mathrm{x}}_f, \tau].$$

6. Check $\frac{\mathcal{D}^{(\rho-1)}[\tau]-\mathcal{D}^{(\rho)}[\tau]}{\mathcal{D}^{(\rho)}[\tau]} \leq \delta$ or $\rho \geq Maxiter$, if so then go to step 2; otherwise $\rho = \rho + 1$ and go to step 3.

2.2 Channel Optimized Vector Quantizer (COVQ)

A channel optimized vector quantizer (COVQ) is a quantizer that also uses block coding and is designed under the assumption of a noisy channel. A COVQ is modeled and structured in similar ways to the vector quantizer, with the exception of including a noisy channel into the design, instead of assuming a noiseless channel. In this section we study the (k,N) COVQ.

2.2.1 COVQ Model

Codebook: A (k,N) channel optimized vector quantizer (COVQ) produces a set of N codevectors $\underline{c}_i$ called the codebook C

$$\mathrm{C} = \{\underline{c}_i\}_{i=1}^{N} \quad \text{where} \quad \underline{c}_i \in \mathbb{R}^k \text{ for } i = 1, \ldots, N. \tag{2.9}$$

Structure: Figure 2.1 depicts how a source sample vector $\underline{\mathrm{x}} \in \mathbb{R}^k$ is quantized by a (k,N) COVQ into a codevector $\underline{c}_j$. The COVQ is broken down into an encoder function (E), index assignment function (b) and the decoder function (G). The COVQ encoder E encodes a source sample vector $\underline{\mathrm{x}}$ into an index l, that is then transmitted over a noisy channel using the index assignment function b(). At the decoder G the received index j is decoded into $\underline{c}_j$.

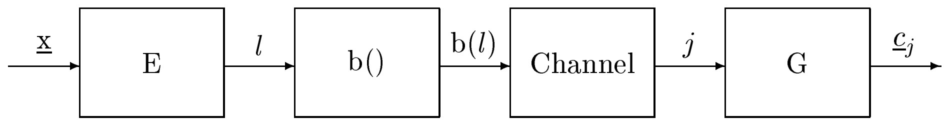

Figure 2.1: Model of a (k,N) COVQ.

Encoder: The encoder function E of the COVQ encodes the source $\underline{\mathrm{x}}$ into an index $l \in \mathrm{J}_N = \{1, \ldots, N\}$. This encoding is done using the encoding regions, or partition cells, S_l for $l = 1, \ldots, N$ of the COVQ, such that

$$\mathrm{E}(\underline{\mathrm{x}}) = l \quad \text{if and only if} \quad \underline{\mathrm{x}} \in \mathrm{S}_l$$

where $\bigcup_{l=1}^{N} \mathrm{S}_l = \mathbb{R}^k$.

Channel: After encoding the source sample $\underline{\mathrm{x}}$ into an index l, the index l is transmitted over the noisy channel using the index assignment function b, where $\mathrm{b} : \mathrm{J}_N \to \mathrm{J}_N$. The index assignment function rearranges the indices associated with the encoding regions. After applying the index assignment function on the encoder output of the source sample $\underline{\mathrm{x}}$, $\mathrm{b}(l)$ is transmitted over the channel. The channel considered in this thesis is the simplest form of a discrete memoryless channel, that is the binary symmetric channel (BSC). Using a BSC with a cross over probability ϵ, the transition probabilities of the channel are, [7],

$$\mathrm{P}(j|\mathrm{b}(l)) = (1-\epsilon)^{n-d_H(j,\mathrm{b}(l))}(\epsilon)^{d_H(j,\mathrm{b}(l))} \tag{2.10}$$

where $d_H(j, \mathrm{b}(l))$ is the Hamming distance between the $\log_2 N$-bit binary codewords of j and $\mathrm{b}(l)$ and $n = \log_2 N$.

Decoder: The decoder function G simply decodes the received index j into the appropriate index vector $\underline{c}_j$

$$\mathrm{G}(j) = \underline{c}_j \quad \text{for} \quad j \in \mathrm{J}_N.$$

Note that the decoder function inverts the combined encoder function and the index assignment function

$$
\begin{aligned}
\mathrm{E} &: \mathbb{R}^k \to \mathrm{J}_N \\
\mathrm{b} &: \mathrm{J}_N \to \mathrm{J}_N \\
\mathrm{G} &: \mathrm{J}_N \to \mathrm{C} \subset \mathbb{R}^k.
\end{aligned}
$$

Rate: Each source sample vector $\underline{\mathrm{x}} \in \mathbb{R}^k$ is encoded into an index $l \in \mathrm{J}_N$, and the binary representation of the index l requires $\log_2 N$ bits. Hence the rate of a (k,N) COVQ is

$$\mathrm{R} = \frac{\log_2 N}{k} \text{ bits/source sample.}$$

2.2.2 COVQ Necessary Conditions for Optimality

Distortion: Given a source $\underline{\mathrm{X}}$ with probability density function $\mathrm{p}(\underline{\mathrm{x}})$, let $\underline{\mathrm{Y}}$ be the output of the decoder when $\underline{\mathrm{X}}$ is quantized by a COVQ. Using the definition of an encoding region S_l

$$\mathrm{S}_l = \{\underline{\mathrm{x}} \in \mathbb{R}^k : \mathrm{E}(\underline{\mathrm{x}}) = l\} \quad \text{for } l = 1, \ldots, N$$

the expected squared-error distortion of a (k,N) COVQ, given the set of encoding regions $\{\mathrm{S}_l\}_{l=1}^N$ and the codebook $\mathrm{C} = \{\underline{c}_j\}_{j=1}^N$, can be found to be

$$
\begin{aligned}
\mathrm{D}_{COVQ} &= E\{d(\underline{\mathrm{X}}, \underline{\mathrm{Y}})\} \\
&= \sum_{l=1}^{N} \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(l)) E\{d(\underline{\mathrm{X}}, \underline{c}_j) | \underline{\mathbf{X}} \in \mathrm{S}_l\} \mathrm{P}(\underline{\mathbf{X}} \in \mathrm{S}_l) \\
&= \sum_{l=1}^{N} \int_{\mathrm{S}_l} \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(l)) \left\| \underline{\mathrm{x}} - \underline{c}_j \right\|^2 \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}}.
\end{aligned}
$$

Optimal Encoding: Given the optimal codebook $\mathrm{C} = \{\underline{c}_i\}_{i=1}^N$, how can we determine the optimal encoding function so as to minimize the expected squared-error distortion when a source sample $\underline{\mathbf{x}}$ is encoded into an index l, using a (k,N) COVQ encoder?

Let $\underline{\mathrm{Y}}$ be the reproduction, or the output of the decoder, of a source vector $\underline{\mathrm{x}}$ and let $\underline{\mathrm{x}} \in \mathrm{S}_l$ then

$$\begin{aligned} E\{d(\underline{\mathrm{x}},\underline{\mathrm{Y}})\} &= \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(l)) \left\|\underline{\mathrm{x}} - \underline{c}_j\right\|^2 \\ &\geq \min_l \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(l)) \left\|\underline{\mathrm{x}} - \underline{c}_j\right\|^2 . \end{aligned}$$

Hence the optimal encoding function is

$$\mathrm{E}(\underline{\mathrm{x}}) = \arg\min_l \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(l)) \left\|\underline{\mathrm{x}} - \underline{c}_j\right\|^2 .$$

The distortion incurred by mapping the source $\underline{\mathrm{x}}$ into the index l is

$$\mathrm{D}_l(\underline{\mathrm{x}}) = \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(l)) \left\|\underline{\mathrm{x}} - \underline{c}_j\right\|^2$$

and the optimal encoding regions become

$$\begin{aligned} \mathrm{S}_i &= \{\underline{\mathrm{x}} \in \mathbb{R}^k : i = \arg\min_l \mathrm{D}_l(\underline{\mathrm{x}})\} && (2.11) \\ &= \{\underline{\mathrm{x}} \in \mathbb{R}^k : \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(i)) \left\|\underline{\mathrm{x}} - \underline{c}_j\right\|^2 \leq \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(l)) \left\|\underline{\mathrm{x}} - \underline{c}_j\right\|^2 \; \forall \; l \in \mathrm{J}_N\} && (2.12) \end{aligned}$$

for $i = 1, \ldots, N$.

Optimal Decoding: Now assume that the optimal encoding regions $\{\mathrm{S}_i\}_{i=1}^N$ defined by (2.12) are given, then the optimal codevectors $\{\underline{c}_l\}_{l=1}^N$ of codebook C need to be derived. This is done by using the distortion

$$\mathrm{D}_{COVQ} = \sum_{i=1}^{N} \int_{\mathrm{S}_i} \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(i)) \left\|\underline{\mathrm{x}} - \underline{c}_j\right\|^2 \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}}.$$

Take the partial derivative of the above with respect to $\underline{c}_l$ by separating the l^{th} codevector in the above as follows

$$\mathrm{D}_{COVQ} = \sum_{i=1}^{N} \int_{\mathrm{S}_i} \left\{ \mathrm{P}(l|\mathrm{b}(i)) \, \|\underline{\mathrm{x}} - \underline{c}_l\|^2 + \sum_{j \neq l} \mathrm{P}(j|\mathrm{b}(i)) \|\underline{\mathrm{x}} - \underline{c}_j\|^2 \right\} \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}}.$$

Setting the derivative to zero we get

$$0 = \sum_{i=1}^{N} \int_{\mathrm{S}_i} \mathrm{P}(l|\mathrm{b}(i)) \, \{-\underline{\mathrm{x}} + \underline{c}_l\} \, \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}}.$$

Solving for $\underline{c}_l$ we get

$$\underline{c}_l = \frac{\sum_{i=1}^{N} \int_{\mathrm{S}_i} \mathrm{P}(l|\mathrm{b}(i)) \quad \underline{\mathrm{x}}\mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}}}{\sum_{i=1}^{N} \int_{\mathrm{S}_i} \mathrm{P}(l|\mathrm{b}(i)) \quad \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}}}. \tag{2.13}$$

The optimal codevectors $\{\underline{c}_l\}_{l=1}^{N}$ defined by (2.13) are also referred to as centroids since

$$\underline{c}_l = \arg \min_{\underline{\mathrm{y}} \in \mathbb{R}^k} E\{\|\underline{\mathrm{X}} - \underline{\mathrm{y}}\|^2 | \mathrm{V} = l\}$$

for $l = 1, \ldots, N$ and V is the random index output of the channel.

2.2.3 COVQ Encoding Simplifications

In [16] Farvardin and Vaishampayan study the performance of the channel optimized vector quantizer and illustrate how the encoding complexity of the COVQ can be reduced. Consider the optimal encoder function of the (k,N) COVQ

$$\begin{aligned} \mathrm{E}(\underline{\mathrm{x}}) &= \arg\min_{l} \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(l)) \left\|\underline{\mathrm{x}} - \underline{c}_j\right\|^2 \\ &= \arg\min_{l} \sum_{j=1}^{N} \mathrm{P}(j|\mathrm{b}(l)) \left\{\|\underline{\mathrm{x}}\|^2 - 2<\underline{\mathrm{x}}, \underline{c}_j> + \|\underline{c}_j\|^2\right\} \end{aligned}$$

where $<\underline{x}, \underline{y}>$ is the inner product over $\mathbb{R}^k$. This function can be simplified by introducing the functions

$$\mathrm{y}(\gamma) = \sum_{j=1}^{N} P(j|\mathrm{b}(\gamma))\underline{c}_j \quad \text{and} \quad \alpha(\gamma) = \sum_{j=1}^{N} P(j|\mathrm{b}(\gamma)) \left\|\underline{c}_j\right\|^2. \tag{2.14}$$

Substituting the above functions into the encoder function we get

$$\mathrm{E}(\underline{\mathrm{x}}) = \arg\min_{l}\{\alpha(l) - 2<\underline{\mathrm{x}}, \mathrm{y}(l)>\}.$$

Given the set of vectors $\{\mathrm{y}(\gamma)\}_{\gamma=1}^{N}$ and the set of scalars $\{\alpha(\gamma)\}_{\gamma=1}^{N}$, the encoding complexity of a (k,N) COVQ is now proportional to N.

2.2.4 COVQ Initial Codebook Design

Let $\underline{\mathrm{X}}$ and $\underline{\hat{\mathrm{X}}}$ be the input and output, respectively, of a quantizer and let $\underline{\mathbf{c}}$ be the output of the decoder as in Figure 2.2. For such a setup the total end-to-end expected squared-error distortion can be found to be

$$\mathrm{D}_{total} = E\|\underline{\mathrm{X}} - \underline{\mathbf{c}}\|^2 \tag{2.15}$$

$$= E\|(\underline{\mathrm{X}} - \underline{\hat{\mathrm{X}}} + \underline{\hat{\mathrm{X}}} - \underline{\mathbf{c}})\|^2 \tag{2.16}$$

$$= \underbrace{E\|\underline{\mathrm{X}} - \underline{\hat{\mathrm{X}}}\|^2}_{\epsilon_q^2} + \underbrace{E\|\underline{\hat{\mathrm{X}}} - \underline{\mathbf{c}}\|^2}_{\epsilon_c^2} + 2\underbrace{E\{<\underline{\mathrm{X}} - \underline{\hat{\mathrm{X}}}, \underline{\hat{\mathrm{X}}} - \underline{\mathbf{c}}>\}}_{\text{cross-term}}. \tag{2.17}$$

In a related work [38], Totty and Clark studied the reconstruction error in waveform transmission for scalar quantizers. The authors showed that if the codevectors of the quantizer, above, satisfied the centroids condition (2.8), then the cross-term in equation (2.17) can be eliminated. A detailed proof is also outlined in [9] by Cheng. Hence, assuming the centroids condition is satisfied, the total end-to-end distortion can be reduced to the sum of only the quantization distortion ϵ_q^2 and the channel distortion ϵ_c^2.

Considering a particular quantizer with codebook $\mathrm{C} = \{\underline{c}_i\}_{i=1}^{N}$ and encoding regions $\{\mathrm{S}_i\}_{i=1}^{N}$, the above quantization distortion can be formulated to

$$\epsilon_q^2 = \sum_{i=1}^{N} \int_{\mathrm{S}_i} \|\underline{\mathrm{x}} - \underline{c}_i\|^2 \mathrm{p}(\underline{\mathrm{x}}) d\underline{\mathrm{x}}$$

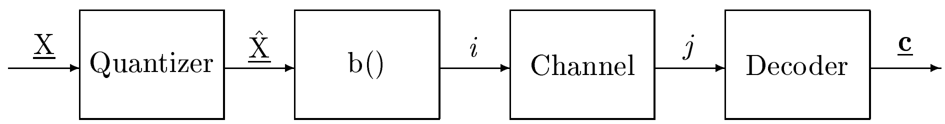

Figure 2.2: Model of a general quantizer and a noisy channel.

where p($\underline{\mathrm{x}}$) is the probability density function of $\underline{\mathrm{X}}$. Furthermore considering the binary assignment function b(), where $b : \mathrm{C} \to \mathrm{J}_N$ and $\mathrm{J}_N = \{1, \ldots, N\}$, the channel distortion can be formulated to be

$$\epsilon_c^2 = \sum_{j=1}^{N} \sum_{i=1}^{N} \mathrm{P}(\underline{c}_i)\mathrm{P}(b(\underline{c}_j)|b(\underline{c}_i)) \left\{ \|\underline{c}_i - \underline{c}_j\|^2 \right\}$$

where $\mathrm{P}(\underline{c}_i)$ is the a priori probability of the codevector $\underline{c}_i$; $\mathrm{P}(\underline{c}_i) = \mathrm{P}(\underline{\mathrm{X}} \in \mathrm{S}_i)$. In view of this analysis and as suggested by Farvardin in [15], we can focus on minimizing the channel distortion ϵ_c^2 by appropriately choosing the index assignment function b(). To choose an appropriate index assignment function b(), we use a technique called simulated annealing.

Simulated annealing belongs to a class of randomized algorithms, [34], where the next state configuration is generated randomly and "hill climbing" is allowed. "Hill climbing" is a move that results in a state with higher energy or cost (in this case the cost is $\epsilon_c^2(\underline{\mathrm{b}})$) than the current one accepted, such a move is used to avoid local minimums. In the field of Information Theory, simulated annealing has been also used to find good channel codes [18]. The simulated annealing algorithm tuned for the purpose of minimizing the channel distortion $\epsilon_c^2(\underline{\mathrm{b}})$

$$\epsilon_c^2(\underline{\mathrm{b}}) = \sum_{j=1}^{N} \sum_{i=1}^{N} \mathrm{p}(\underline{c}_i)\mathrm{P}(b(\underline{c}_j)|b(\underline{c}_i)) \{ \|\underline{c}_i - \underline{c}_j\|^2$$

uses states defined as $\underline{\mathrm{b}} = (\mathrm{b}(\underline{c}_1), \ldots, \mathrm{b}(\underline{c}_N))$, for a N codevector quantizer, and a temperature schedule given by

$$\mathrm{T}_k = \alpha \mathrm{T}_{k-1} \quad \text{where} \quad 0 < \alpha < 1. \tag{2.18}$$

The algorithm is described below.

Simulated Annealing Algorithm

1. Set *Maxper*, the maximum number of perturbations, and the effective temperature to an initial T_0. Randomly choose an initial state $\underline{\mathrm{b}}$.

2. Choose the next state $\underline{\mathrm{b}}'$ randomly and calculate the change in 'energy' $\delta\epsilon_c^2 = \epsilon_c^2(\underline{\mathrm{b}}') - \epsilon_c^2(\underline{\mathrm{b}})$. If $\delta\epsilon_c^2(\underline{\mathrm{b}}) < 0$, replace $\underline{\mathrm{b}}$ with $\underline{\mathrm{b}}'$, and goto step 3, else replace $\underline{\mathrm{b}}$ by $\underline{\mathrm{b}}'$ with probability $\exp\{-\delta\epsilon_c^2/\mathrm{T}_k\}$ and goto step 3. Note that as the temperature decreases the probability of replacing a state $\underline{\mathrm{b}}$ with one that has a higher cost or energy $\epsilon_c^2(\underline{\mathrm{b}})$, becomes lower.

3. If after *Maxper* number of perturbations, no energy drop occurs, goto step 4. Otherwise goto step 2.

4. Lower the effective temperature according to (2.18). If the temperature T is below a prescribed freezing temperature T_f or the system appears to be stable, stop with $\underline{\mathrm{b}}$ as the final state. Otherwise goto step 2.

Table 2.1, tabulates the set of prescribed parameters used in the simulated annealing algorithm. These parameters were suggested in [9], [18] and [15].

In view of the result (2.17)

$$\mathrm{D}_{total} = \epsilon_q^2 + \epsilon_c^2$$

one can suggest a 'channel optimized' tandem source-channel coding system where the codebooks of a quantizer are designed using the LBG algorithm of section 2.1.3

T_0	10.0
T_f	0.00025
α	0.97
$Maxper$	200

Table 2.1: Suggested parameters for Simulated Annealing Algorithm.

and the index assignment function designed using the simulated annealing algorithm. In [15] Farvardin illustrates the advantages of using an initial codebook thus designed as initial points to the design of a COVQ. Such a method of initializing the design of a COVQ was shown to be superior, in achieving lower distortions, than designs that used the splitting method or random initialization method.

2.2.5 Design Algorithm of a (k,N) COVQ

Now that the initial codebook $\mathrm{C}^{(0)}$ and index assignment function b() can be found using the analysis of Section 2.2.4, the (k,N) COVQ can be designed using a simple algorithm that iterates through the necessary conditions (2.12) and (2.13).

(k,N) COVQ Algorithm

1. Set parameters k, N, the design BSC error crossover probability ϵ_d, the stopping threshold δ, the maximum number of iterations $Maxiter$, and M the total number of training vectors $\{\underline{\mathrm{x}}_f\}_{f=1}^{M}$. Initialize $\rho = 0$, and initialize the codebook $\mathrm{C}^{(0)} = \{\underline{c}_i^{(0)}\}_{i=1}^{N}$ and index assignment function b(), from Section 2.2.4.

2. Calculate the N vectors $\{\mathrm{y}^{(\rho)}(\gamma)\}_{\gamma=1}^{N}$ and the N scalars $\{\alpha^{(\rho)}(\gamma)\}_{\gamma=1}^{N}$ using codebook $\mathrm{C}^{(\rho)}$ and functions (2.14).

3. For each f, encode $\underline{\mathrm{x}}_f$ using

$$\mathrm{E}^{(\rho)}(\underline{\mathrm{x}}_f) = \arg\min_{i \in \mathrm{J}_N} \left\{ \alpha^{(\rho)}(i) - 2<\underline{\mathrm{x}}_f, \mathrm{y}^{(\rho)}(i)> \right\}$$

where $\mathrm{J}_N = \{1, \ldots, N\}$.

4. Put $\underline{\mathrm{x}}_f$ into the appropriate encoding regions, (2.12). So if $\mathrm{E}^{(\rho)}(\underline{\mathrm{x}}_f) = i^*$ then

$$\underline{\mathrm{x}}_f \in \mathrm{S}_{i^*}^{(\rho)}$$

with the distortion

$$D^{(\rho)}(\underline{\mathrm{x}}_f) = \sum_{l=1}^{N} \mathrm{P}(l|\mathrm{b}(i^*)) \left\| \underline{\mathrm{x}}_f - \underline{c}_l^{(\rho)} \right\|^2$$

where the transition probabilities are calculated using (2.10) and ϵ_d.

5. Repeat steps 3 and 4 for $f = 1, \ldots, M$. Then update the codebook to $\mathrm{C}^{(\rho+1)}$ using

$$\underline{c}_l^{(\rho+1)} = \frac{\sum_{i=1}^{N} \mathrm{P}(l|\mathrm{b}(i)) \sum_{\underline{\mathrm{x}}_f : \underline{\mathrm{x}}_f \in \mathrm{S}_i^{(\rho)}} \underline{\mathrm{x}}_f}{\sum_{i=1}^{N} \mathrm{P}(l|\mathrm{b}(i)) \sum_{\underline{\mathrm{x}}_f : \underline{x}_f \in \mathrm{S}_i^{(\rho)}}}.$$

Finally calculate the overall distortion

$$\mathcal{D}^{(\rho)} = \frac{1}{kM} \sum_{f=1}^{M} D^{(\rho)}(\underline{\mathrm{x}}_f).$$

6. Check $\frac{\mathcal{D}^{(\rho-1)} - \mathcal{D}^{(\rho)}}{\mathcal{D}^{(\rho)}} \leq \delta$ or $\rho \geq Maxiter$, if so then stop, otherwise $\rho = \rho + 1$ and go to step 2.

2.3 Product Quantizer (PQ)

The sample adaptive product quantizer (SAPQ), to be described in Chapter 3, is based on the product quantizer (PQ). The PQ is an alternative to the vector quantizer (VQ), that achieves a lower encoding complexity at the expense of a loss in

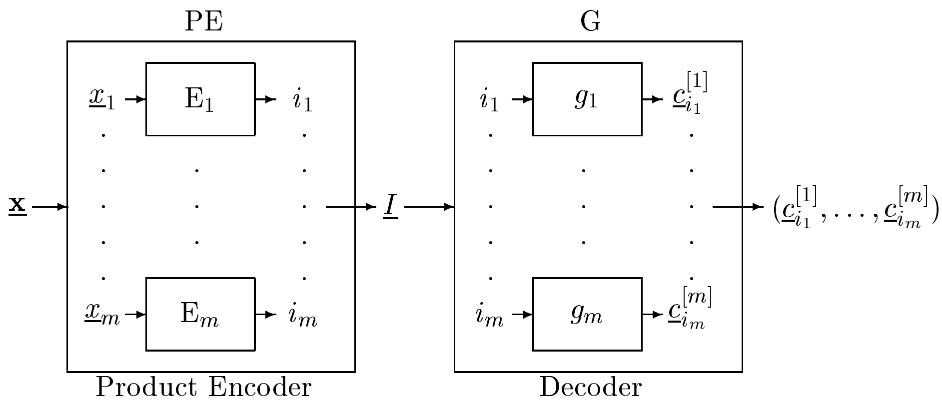

Figure 2.3: Model of a (k,m,N) PQ.

performance (lower signal-to-distortion ratio). In this section the (k,m,N) PQ will be described and studied.

2.3.1 PQ Model

Codebook: The codebook $\mathbf{C}$ of a (k,m,N) product quantizer (PQ) is a product of a set of m codebooks $\{\mathrm{C}_j\}_{j=1}^{m}$. Each codebook C_j is a set of N codevectors $\{\underline{c}_i^{[j]}\}_{i=1}^{N}$ and a subset of $\mathbb{R}^k$ such that

$$\mathbf{C} = \mathrm{C}_1 \times \ldots \times \mathrm{C}_m \text{ where } \mathrm{C}_j = \{\underline{c}_i^{[j]}\}_{i=1}^{N} \text{ and } \underline{c}_i^{[j]} \in \mathbb{R}^k. \tag{2.19}$$

Structure: Figure 2.3 depicts how a source sample vector $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$, where $\underline{\mathrm{x}}_s \in \mathbb{R}^k$ for $s = 1, \ldots, m$, is quantized by a (k,m,N) PQ. As depicted the PQ is broken down into a product encoder (PE), and a decoder (G).

Encoder: A (k,m,N) PQ encoder is referenced here as a product encoder (PE). A product encoder is a vector function that encodes a source sample $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$

into a vector of indexes $\underline{I}$. The constituent function of the product encoder are encoder functions (E_s, for $s = 1, \ldots, m$) such that

$$\mathrm{PE}(\underline{\mathbf{x}}) = (\mathrm{E}_1(\underline{\mathrm{x}}_1), \ldots, \mathrm{E}_m(\underline{\mathrm{x}}_m)) = \underline{I} \quad \text{where} \quad \mathrm{E}_s(\underline{\mathrm{x}}_s) = i_s \in \mathrm{J}_N = \{1, \ldots, N\}$$
$$\text{and} \quad \underline{I} = (i_1, \ldots, i_m) \in \mathrm{J}_N^m.$$

Decoder: At the decoder the index vector $\underline{I} = (i_1, \ldots, i_m)$ produced by the product encoder (PE) is decoded into codevectors. The decoding function is also a vector function (G) that has m component functions $\{g_s\}_{s=1}^m$, such that

$$\mathrm{G}(\underline{I}) = (g_1(i_1), \ldots, g_m(i_m)) = (\underline{c}_{i_1}^{[1]}, \ldots, \underline{c}_{i_m}^{[m]}).$$

Note that encoder function E_s and decoder function g_s are related by codebook C_s, for $s = 1, \ldots, m$. In other words a (k,m,N) PQ can be thought of as a row of m (k,N) vector quantizers. Each one of the m (k,N) VQ quantizes its corresponding component source sample $\underline{\mathrm{x}}_s$ for $s = 1, \ldots, m$ of $\underline{\mathbf{x}}$.

Rate: For a source sample vector $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$, where $\underline{\mathrm{x}}_s \in \mathbb{R}^k$ for $s = 1, \ldots, m$, the product encoder function (PE) of the (k,m,N) PQ produces an index vector $\underline{I} = (i_1, \ldots, i_m)$, where $i_s \in \mathrm{J}_N$ for $s = 1, \ldots, m$, when it encodes $\underline{\mathbf{x}}$. Hence there are m indexes i_s that are produced for every km source samples $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$. Each index i_s can be represented by a $\log_2 N$-bit codeword. Hence the rate of a (k,m,N) PQ is

$$\mathrm{R} = \frac{m \log_2 N}{km} = \frac{\log_2 N}{k} \text{ bits/source sample.}$$

2.3.2 PQ Necessary Conditions for Optimality

Distortion: For a (k,m,N) PQ, the partition cells or encoding regions, are defined as

$$\mathrm{S}_i^{[j]} = \{\underline{\mathrm{x}} \in \mathbb{R}^k : \mathrm{E}_j(\underline{\mathrm{x}}) = i\} \tag{2.20}$$

where $i = 1, \ldots, N$, $j = 1, \ldots, m$ and E_j is the j^{th} encoding function of the (k,m,N) PQ. With the encoding regions defined the mean squared distortion of the (k,m,N) PQ can be found. Let $\underline{\mathbf{c}}$ be the output of the decoder when a source $\underline{\mathbf{X}} = (\underline{\mathrm{X}}_1, \ldots, \underline{\mathrm{X}}_m)$, with probability density function $\mathrm{p}(\underline{\mathbf{x}}) = \mathrm{p}(\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$, as input. Then the mean squared distortion is

$$\begin{aligned} \mathrm{D}_{PQ} &= E\{d(\underline{\mathbf{X}}, \underline{\mathbf{c}})\} \\ &= \sum_{t=1}^{m} \sum_{i_t=1}^{N} \mathrm{P}(\underline{\mathrm{X}}_t \in \mathrm{S}_{i_t}^{[t]}) E\{\|\underline{\mathrm{X}}_t - \underline{c}_{i_t}^{[t]}\|^2 | \underline{\mathrm{X}}_t \in \mathrm{S}_{i_t}^{[t]}\} \\ &= \sum_{t=1}^{m} \sum_{i_t=1}^{N} \int_{\mathrm{S}_{i_t}^{[t]}} \|\underline{\mathrm{x}} - \underline{c}_{i_t}^{[t]}\|^2 \mathrm{p}_t(\underline{\mathrm{x}}) d\underline{\mathrm{x}} \end{aligned}$$

where for $t = 1, \ldots, m$, $\mathrm{p}_t(\underline{\mathrm{x}})$ is the marginal probability density function of $\underline{\mathrm{X}}_t$,

$$\mathrm{p}_t(\underline{\mathrm{x}}) = \underbrace{\int_{\underline{\mathrm{x}}_1 \in \mathbb{R}^k} \cdots \int_{\underline{\mathrm{x}}_m \in \mathbb{R}^k}}_{\text{no } \underline{\mathrm{X}}_t} \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}.$$

Optimal Encoding: Let $\underline{\mathbf{y}}$ be the output of the decoder when source sample $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$ is quantized by a (k,m,N) PQ, and let $\underline{\mathrm{x}}_t \in \mathrm{S}_{z_t}^{[t]}$, for $t = 1, \ldots, m$ and $z_t \in \mathrm{J}_N$, then

$$\begin{aligned} d(\underline{\mathbf{x}}, \underline{\mathbf{y}}) &= \sum_{t=1}^{m} \|\underline{\mathrm{x}}_t - \underline{c}_{z_t}^{[t]}\|^2 \\ &\geq \min_{\underline{Z}=(z_1,\ldots,z_m) \in \mathrm{J}_N^m} \sum_{t=1}^{m} \|\underline{\mathrm{x}}_t - \underline{c}_{z_t}^{[t]}\|^2. \end{aligned}$$

Hence the optimal product encoding function (PE) of a (k,m,N) PQ is

$$\mathrm{PE}(\underline{\mathbf{x}}) = \arg \min_{\underline{Z}=(z_1,\ldots,z_m) \in \mathrm{J}_N^m} \sum_{t=1}^{m} \|\underline{\mathrm{x}}_t - \underline{c}_{z_t}^{[t]}\|^2$$

and naturally the optimal component encoding function is

$$\mathrm{E}_t(\underline{\mathrm{x}}) = \arg \min_{z \in \mathrm{J}_N} \|\underline{\mathrm{x}} - \underline{c}_z^{[t]}\|^2 \tag{2.21}$$

for $t = 1, \ldots, m$. From the above formulation of the optimal encoding functions, and from the definition of the encoding regions of a (k,m,N) PQ (2.20), the optimal encoding regions can be found to be

$$\mathrm{S}_i^{[j]} = \{\underline{\mathrm{x}} \in \mathbb{R}^k : \|\underline{\mathrm{x}} - \underline{c}_i^{[j]}\|^2 \leq \|\underline{\mathrm{x}} - \underline{c}_z^{[j]}\|^2 \quad \forall z \in \mathrm{J}_N\} \tag{2.22}$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, m$.

Optimal Decoding: To find the optimal set of codevectors $\{\underline{c}_i^{[j]}\}$, the expected mean squared distortion of the (k,m,N) PQ is manipulated in order to filter out codevector $\underline{c}_l^{[j]}$, the l^{th} codevector of codebook C_j where $l \in \mathrm{J}_N$ and $j \in \mathrm{J}_m$, as follows

$$\begin{aligned}
\mathrm{D}_{PQ} &= \sum_{t=1}^{m}\sum_{i_t=1}^{N}\int_{\mathrm{S}_{i_t}^{[t]}} \|\underline{\mathrm{x}} - \underline{c}_{i_t}^{[t]}\|^2 \mathrm{p}_t(\underline{\mathrm{x}})d\underline{\mathrm{x}} \\
&= \sum_{i=1}^{N}\int_{\mathrm{S}_i^{[j]}} \|\underline{\mathrm{x}} - \underline{c}_i^{[j]}\|^2 \mathrm{p}_j(\underline{\mathrm{x}})d\underline{\mathrm{x}} + \sum_{t\neq j}\sum_{i_t=1}^{N}\int_{\mathrm{S}_{i_t}^{[t]}} \|\underline{\mathrm{x}} - \underline{c}_{i_t}^{[t]}\|^2 \mathrm{p}_t(\underline{\mathrm{x}})d\underline{\mathrm{x}}
\end{aligned}$$

where $\{\mathrm{S}_i^{[t]}\}$ is given and defined as in (2.22), for $i = 1, \ldots, N$ and $t = 1, \ldots, m$. Taking the partial derivative of the above with respect to $\underline{c}_l^{[j]}$, and setting the resultant derivative to zero we get

$$0 = \int_{\mathrm{S}_l^{[j]}} \{-\underline{\mathrm{x}} + \underline{c}_l^{[j]}\}\mathrm{p}_j(\underline{\mathrm{x}})d\underline{\mathrm{x}}.$$

Solving for $\underline{c}_l^{[j]}$ we get

$$\underline{c}_l^{[j]} = \frac{\int_{\mathrm{S}_l^{[j]}} \underline{\mathrm{x}} \quad \mathrm{p}_j(\underline{\mathrm{x}})d\underline{\mathrm{x}}}{\int_{\mathrm{S}_l^{[j]}} \quad \mathrm{p}_j(\underline{\mathrm{x}})d\underline{\mathrm{x}}}. \tag{2.23}$$

Note that this is the same as finding the optimal l^{th} codevector of the j^{th} (k,N) vector quantizer, (2.24).

2.3.3 Design Algorithm of (k,m,N) PQ

As described by Gersho and Gray in [19], the (k,m,N) PQ can be thought of a combination of m parallel and independent (k,N) vector quantizers, such that

$$\mathrm{PQ}(\underline{\mathbf{x}}) = (\mathrm{VQ}_1(\underline{\mathrm{x}}_1), \ldots, \mathrm{VQ}_m(\underline{\mathrm{x}}_m)) \tag{2.24}$$

where $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$, and $\underline{\mathrm{x}}_j \in \mathbb{R}^k$ and VQ_j is a (k,N) VQ, for $j = 1, \ldots, m$. With this in mind, we can design a (k,m,N) PQ using an algorithm that designs each VQ_j, or codebook C_j, using the LBG-VQ algorithm of Section 2.1.3 for $j = 1, \ldots, m$.

(k,m,N) PQ Algorithm

1. Set parameters k, m, N, the stopping threshold δ, the splitting constant ϵ, the maximum number of iterations $Maxiter$, and M the total number of training vectors $\{\underline{\mathbf{x}}_f = (\underline{\mathrm{x}}_{1,f}, \ldots, \underline{\mathrm{x}}_{m,f})\}_{f=1}^{M}$. Initialize $j = 1$, this is the counter that keeps track of the m codebooks.

2. Start off with $\tau = 1$, $\rho = 0$, and

$$\underline{c}_1^{[j],(0)} = \frac{1}{M} \sum_{f=1}^{M} \underline{\mathrm{x}}_f$$

as the initial codevector of codebook C_j and

$$\mathcal{D}_j^{(0)}[1] = \frac{1}{kM} \sum_{f=1}^{M} \|\underline{\mathrm{x}}_{j,f} - \underline{c}_1^{[j],(0)}\|^2$$

as the initial mean squared-error distortion for the vector quantizer VQ_j.

3. If $\tau \geq N$ goto step 8 otherwise double the codevectors in codebook C_j using

$$\underline{c}_i^{[j],(\rho)} = \underline{c}_i^{[j],(\rho)}(1+\epsilon) \quad \text{and} \quad \underline{c}_{\tau+i}^{[j],(\rho)} = \underline{c}_i^{[j],(\rho)}(1-\epsilon)$$

for $i = 1, \ldots, \tau$ then double $\tau = \tau * 2$ and set $\rho = 0$. Note that ρ is a counter for the iterations and τ is a counter for the number of codevectors in codebook C_j. At this point we have τ codevectors $\mathrm{C}_j^{(\rho)} = \{\underline{c}_i^{[j],(\rho)}\}_{i=1}^{\tau}$.

4. For each f, encode $\underline{\mathrm{x}}_{j,f}$ into an index using (2.21) as

$$\mathrm{E}_j^{(\rho)}(\underline{\mathrm{x}}_{j,f}) = \arg\min_{i \in \mathrm{J}_\tau} \|\underline{\mathrm{x}}_{j,f} - \underline{c}_i^{[j],(\rho)}\|^2$$

where $\mathrm{J}_\tau = \{1, \ldots, \tau\}$.

5. Once $\underline{\mathrm{x}}_{j,f}$ has been encoded, put $\underline{\mathrm{x}}_{j,f}$ into the appropriate partition cells (2.22). So if $\underline{\mathrm{x}}_{j,f}$ was encoded into $\mathrm{E}_j^{(\rho)}(\underline{\mathrm{x}}_{j,f}) = i^*$, then

$$\underline{\mathrm{x}}_f \in \mathrm{S}_{i^*}^{[j],(\rho)}$$

and the resulting distortion is then

$$D_j^{(\rho)}[\underline{\mathrm{x}}_{j,f}, \tau] = \|\underline{\mathrm{x}}_{j,f} - \underline{c}_{i^*}^{[j],(\rho)}\|^2.$$

6. Repeat steps 4 and 5 for all $f = 1 \ldots, M$. Once all the partitions have been made, update the codebook $\mathrm{C}_\tau^{(\rho)}$ using

$$\underline{c}_l^{[j],(\rho)} = \frac{\sum_{\underline{\mathrm{x}}_{j,f}:\underline{\mathrm{x}}_{j,f} \in \mathrm{S}_l^{[j],(\rho)}} \underline{\mathrm{x}}_{j,f}}{\sum_{\underline{\mathrm{x}}_{j,f}:\underline{\mathrm{x}}_{j,f} \in \mathrm{S}_l^{[j],(\rho)}}}.$$

Finally calculate the overall distortion of VQ_j using

$$\mathcal{D}_j^{(\rho)}[\tau] = \frac{1}{kM} \sum_{f=1}^{M} D_j^{(\rho)}[\underline{\mathrm{x}}_{j,f}, \tau].$$

7. Check $\frac{\mathcal{D}_j^{(\rho-1)}[\tau] - \mathcal{D}_j^{(\rho)}[\tau]}{\mathcal{D}_j^{(\rho)}[\tau]} \leq \delta$ or $\rho \geq Maxiter$, if so then go to step 3 otherwise increment $\rho = \rho + 1$ and go to step 4.

8. If $j \geq m$ then calculate the overall distortion as

$$\mathcal{D}_{overall} = \frac{1}{m}\sum_{j=1}^{m} \mathcal{D}_j^{(Maxiter)}[N]$$

and stop, otherwise increment $j = j + 1$ and goto step 2.

2.4 Channel Optimized Product Quantizer (COPQ)

Just as the product quantizer (PQ), is a building block for the sample adaptive product quantizer (SAPQ), the channel optimized product quantizer (COPQ) serves as a building block for the channel optimized sample adaptive product quantizer (COSAPQ). In this section the (k,m,N) COPQ will be described and studied.

2.4.1 COPQ Model

Codebook: The codebook $\mathbf{C}$ of a (k,m,N) channel optimized product quantizer (COPQ) is a product of a set of m codebooks $\{\mathrm{C}_j\}_{j=1}^{m}$. Each constituent codebook C_j is a set of N codevectors $\{\underline{c}_i^{[j]}\}_{i=1}^{N}$ and a subset of $\mathbb{R}^k$ such that

$$\mathbf{C} = \mathrm{C}_1 \times \ldots \times \mathrm{C}_m \text{ where } \mathrm{C}_j = \{\underline{c}_i^{[j]}\}_{i=1}^{N} \text{ and } \underline{c}_i^{[j]} \in \mathbb{R}^k. \tag{2.25}$$

Structure: Figure 2.4 depicts how a source sample vector $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$ is quantized by a (k,m,N) COPQ. Like the COVQ the COPQ includes the noise statistics and hence the channel is included in the model of the COPQ.

Encoder: A (k,m,N) COPQ takes in a source sample $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$ and encodes it into a vector of indexes $\underline{I} = (i_1, \ldots, i_m)$ using a product encoder function (PE). The product encoder function (PE) is a vector of encoding functions $\{\mathrm{E}_s\}_{s=1}^{m}$ such that

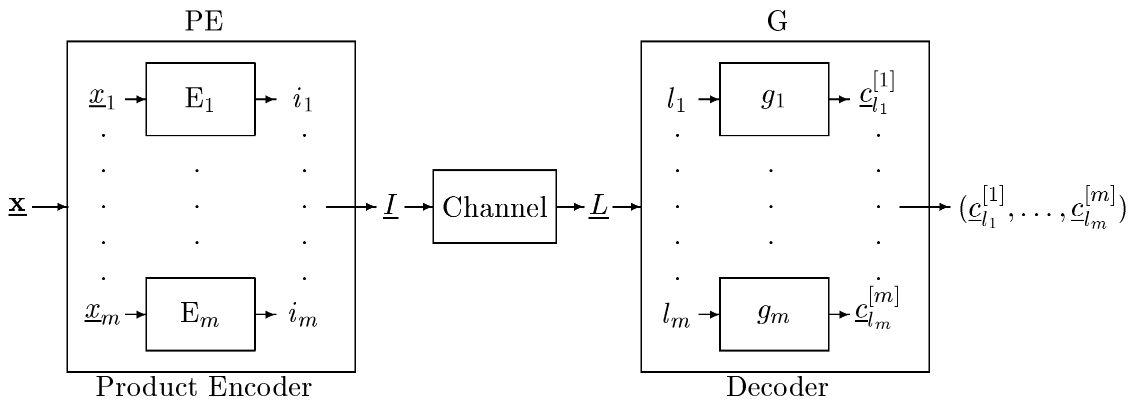

Figure 2.4: Model of a (k,m,N) COPQ.

$$\mathrm{PE}(\underline{\mathbf{x}}) = (\mathrm{E}_1(\underline{\mathrm{x}}_1), \ldots, \mathrm{E}_m(\underline{\mathrm{x}}_m)) = \underline{I} \quad \text{where} \quad \mathrm{E}_s(\underline{\mathrm{x}}_s) = i_s \in \mathrm{J}_N = \{1, \ldots, N\}$$
$$\text{and} \quad \underline{I} = (i_1, \ldots, i_m) \in \mathrm{J}_N^m.$$

Note that although the product encoder (PE) of a (k,m,N) COPQ shares the same name and description as that of a (k,m,N) PQ, their formulations are different. The PE of the COPQ takes into consideration the channel noise whereas the PE of the PQ does not. However both are a mapping from the Euclidean space $\mathbb{R}^{km}$ to the set $\mathrm{J}_N^m = \{1, \ldots, N\}^m$, $\mathrm{PE} : \mathbb{R}^{km} \to \mathrm{J}_N^m$.

Channel: Transmission of the index vector $\underline{I} = (i_1, \ldots, i_m)$ over a noisy channel is realized by converting the indexes $i_1, \ldots, i_m$, of index vector $\underline{I}$, where $i_s \in \mathrm{J}_N$, into $\log_2 N$-bit codewords and transmitting each binary codeword one at a time. Thus the channel is used independently by each transmitted index $i_1, \ldots, i_m$. Hence the probability of receiving $\underline{L} = (l_1, \ldots, l_m)$ given that $\underline{I}$ was transmitted is

$$\mathrm{P}(\underline{L}|\underline{I}) = \prod_{s=1}^{m} \mathrm{P}(l_s|i_s). \tag{2.26}$$

In this thesis the natural binary codeword (NBC) assignment will be used for the

COPQ. So each of the indexes $i_1, \ldots, i_m$, of index vector $\underline{I}$, are encoded into their n-bit binary codeword equivalent, where $n = \log_2 N$. These codewords are then transmitted over a BSC with cross over probability ϵ, such that the channel transition probabilities resolve to

$$\mathrm{P}(l_s|i_s) = (1-\epsilon)^{n-d_H(l_s,i_s)}(\epsilon)^{d_H(l_s,i_s)} \quad \text{for } s = 1, \ldots, m \tag{2.27}$$

where $d_H(l_s, i_s)$ is the Hamming distance between the n-bit binary codewords of i_s and l_s.

Decoder: At the decoder the index vector $\underline{L} = (l_1, \ldots, l_m)$, received from the channel output, is decoded into codevectors. The decoding function is also a vector function (G) that has m component functions $\{g_s\}_{s=1}^m$, such that

$$\mathrm{G}(\underline{I}) = (g_1(l_1), \ldots, g_m(l_m)) = (\underline{c}_{l_1}^{[1]}, \ldots, \underline{c}_{l_m}^{[m]}).$$

Just as the PQ, a (k,m,N) COPQ can be thought of as row of m (k,N) COVQ, where each (k,N) COVQ quantizes a component $\underline{\mathrm{x}}_s$ of $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$.

Rate: Like the (k,m,N) PQ, the rate of a (k,m,N) COPQ can be derived to be

$$\mathrm{R} = \frac{\log_2 N}{k} \text{ bits/source sample.}$$

2.4.2 COPQ Necessary Conditions for Optimality

Distortion: The mean squared distortion of a (k,m,N) COPQ can be derived by using the definition of a partition cell or encoding region

$$\mathrm{S}_i^{[j]} = \{\underline{\mathrm{x}} \in \mathbb{R}^k : \mathrm{E}_j(\underline{\mathrm{x}}) = i\} \tag{2.28}$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, m$. Let $\underline{\mathbf{c}}$ be the output of a (k,m,N) COPQ for a source $\underline{\mathbf{X}} = (\underline{\mathrm{X}}_1, \ldots, \underline{\mathrm{X}}_m)$ with probability density function $\mathrm{p}(\underline{\mathbf{x}}) = \mathrm{p}(\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$, then

$$\begin{aligned}\mathrm{D}_{COPQ} &= E\{d(\underline{\mathbf{X}},\underline{\mathbf{c}})\}\\ &= \sum_{t=1}^{m}\sum_{i_t=1}^{N}\mathrm{P}(\underline{\mathrm{X}}_t\in \mathrm{S}_{i_t}^{[t]})E\{\sum_{l_t=1}^{N}\mathrm{P}(l_t|i_t)\left\|\underline{\mathrm{X}}_t-\underline{c}_{l_t}^{[t]}\right\|^2|\underline{\mathrm{X}}_t\in \mathrm{S}_{i_t}^{[t]}\}\\ &= \sum_{t=1}^{m}\sum_{i_t=1}^{N}\int_{\mathrm{S}_{i_t}^{[t]}}\sum_{l_t=1}^{N}\mathrm{P}(l_t|i_t)\left\|\underline{\mathrm{x}}-\underline{c}_{l_t}^{[t]}\right\|^2\mathrm{p}_t(\underline{\mathrm{x}})d\underline{\mathrm{x}}\end{aligned}$$

where for $t=1,\ldots,m$,$\mathrm{p}_t(\underline{\mathrm{x}})$ is the marginal probability density function of $\underline{\mathrm{X}}_t$,

$$\mathrm{p}_t(\underline{\mathrm{x}})=\underbrace{\int_{\underline{\mathrm{x}}_1\in\mathbb{R}^k}\cdots\int_{\underline{\mathrm{x}}_m\in\mathbb{R}^k}}_{\text{no }\underline{\mathrm{X}}_t}\mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}}.$$

Optimal Encoding: Again let $\underline{\mathbf{Y}}$ be the output of the decoder when the source $\underline{\mathbf{x}}=(\underline{\mathrm{x}}_1,\ldots,\underline{\mathrm{x}}_m)$ is quantized by a (k,m,N) COPQ. Let $\underline{\mathrm{x}}_t\in\mathrm{S}_{z_t}^{[t]}$ for $t=1,\ldots,m$ then

$$\begin{aligned}E\{d(\underline{\mathbf{x}},\underline{\mathbf{Y}})\} &= \sum_{t=1}^{m}\sum_{l_t=1}^{N}\mathrm{P}(l_t|z_t)\left\|\underline{\mathrm{x}}_t-\underline{c}_{l_t}^{[t]}\right\|^2\\ &\geq \min_{(z_1,\ldots,z_m)\in\mathrm{J}_N^m}\sum_{t=1}^{m}\sum_{l_t=1}^{N}\mathrm{P}(l_t|z_t)\left\|\underline{\mathrm{x}}_t-\underline{c}_{l_t}^{[t]}\right\|^2.\end{aligned}$$

Hence the optimal product encoder function (PE) is

$$\mathrm{PE}(\underline{\mathbf{x}})=\arg\min_{(z_1,\ldots,z_m)\in\mathrm{J}_N^m}\sum_{t=1}^{m}\sum_{l_t=1}^{N}\mathrm{P}(l_t|z_t)\left\|\underline{\mathrm{x}}-\underline{c}_{l_t}^{[t]}\right\|^2$$

and the optimal component encoding function is

$$\mathrm{E}_t(\underline{\mathrm{x}})=\arg\min_{z\in\mathrm{J}_N}\sum_{l=1}^{N}\mathrm{P}(l|z)\left\|\underline{\mathrm{x}}-\underline{c}_{l}^{[t]}\right\|^2.$$

Thus the optimal encoding regions are

$$\mathrm{S}_i^{[j]}=\{\underline{\mathrm{x}}\in\mathbb{R}^k:\sum_{l=1}^{N}\mathrm{P}(l|i)\left\|\underline{\mathrm{x}}-\underline{c}_{l}^{[j]}\right\|^2\leq\sum_{l=1}^{N}\mathrm{P}(l|z)\left\|\underline{\mathrm{x}}-\underline{c}_{l}^{[j]}\right\|^2\quad\forall z\in\mathrm{J}_N\}\qquad(2.29)$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, m$.

Optimal Decoding: Separate the i^{th} codevector $\underline{c}_i^{[j]}$ of the j^{th} codebook C_j, from the expected squared-error distortion of a (k,m,N) COPQ as follows

$$\begin{aligned}
\mathrm{D}_{COPQ} &= \sum_{t=1}^{m} \sum_{z_t=1}^{N} \int_{\mathrm{S}_{z_t}^{[t]}} \sum_{l_t=1}^{N} \mathrm{P}(l_t|z_t) \left\| \underline{\mathrm{x}} - \underline{c}_{l_t}^{[t]} \right\|^2 \mathrm{p}_t(\underline{\mathrm{x}}) d\underline{\mathrm{x}} \\
&= \sum_{z=1}^{N} \int_{\mathrm{S}_z^{[j]}} \sum_{l=1}^{N} \mathrm{P}(l|z) \left\| \underline{\mathrm{x}} - \underline{c}_l^{[j]} \right\|^2 \mathrm{p}_j(\underline{\mathrm{x}}) d\underline{\mathrm{x}} \\
&\qquad + \sum_{t \neq j} \sum_{z_t=1}^{N} \int_{\mathrm{S}_{z_t}^{[t]}} \sum_{l_t=1}^{N} \mathrm{P}(l_t|z_t) \left\| \underline{\mathrm{x}} - \underline{c}_{l_t}^{[t]} \right\|^2 \mathrm{p}_t(\underline{\mathrm{x}}) d\underline{\mathrm{x}}
\end{aligned}$$

and take the derivative of the above with respect to $\underline{c}_i^{[j]}$. Setting the resultant to zero we get

$$0 = \sum_{z=1}^{N} \int_{\mathrm{S}_z^{[j]}} \mathrm{P}(i|z) \left\{ -\underline{\mathrm{x}} + \underline{c}_i^{[j]} \right\} \mathrm{p}_j(\underline{\mathrm{x}}) d\underline{\mathrm{x}}.$$

Solving for $\underline{c}_i^{[j]}$ we get

$$\underline{c}_i^{[j]} = \frac{\sum_{z=1}^{N} \mathrm{P}(i|z) \int_{\mathrm{S}_z^{[j]}} \underline{\mathrm{x}} \;\; \mathrm{p}_j(\underline{\mathrm{x}}) d\underline{\mathrm{x}}}{\sum_{z=1}^{N} \mathrm{P}(i|z) \int_{\mathrm{S}_z^{[j]}} \mathrm{p}_j(\underline{\mathrm{x}}) d\underline{\mathrm{x}}}.$$

2.4.3 COPQ Encoding Simplifications

As in the theme of section 2.2.3, the encoding function

$$\begin{aligned}
\mathrm{E}_t(\underline{\mathrm{x}}) &= \arg\min_{z \in \mathrm{J}_N} \sum_{l=1}^{N} \mathrm{P}(l|z) \left\| \underline{\mathrm{x}} - \underline{c}_l^{[t]} \right\|^2 \\
&= \arg\min_{z \in \mathrm{J}_N} \sum_{l=1}^{N} \mathrm{P}(l|z) \{ \|\underline{\mathrm{x}}\|^2 - 2 < \underline{\mathrm{x}}, \underline{c}_l^{[t]} > + \|\underline{c}_l^{[t]}\|^2 \}
\end{aligned}$$

for $t = 1, \ldots, m$, of a (k,m,N) COPQ can be reduced by using the functions

$$\mathrm{y}_j(\gamma) = \sum_{l=1}^{N} P(l|\gamma) \underline{c}_l^{[j]} \quad \text{and} \quad \alpha_j(\gamma) = \sum_{l=1}^{N} P(l|\gamma) \left\| \underline{c}_l^{[j]} \right\|^2 \tag{2.30}$$

for $j = 1, \ldots, m$. The resulting simplified encoding function with a reduced complexity is then

$$\mathrm{E}_t(\underline{\mathrm{x}}) = \arg\min_{z \in \mathrm{J}_N} \{\alpha_t(z) - 2 < \underline{\mathrm{x}}, \mathrm{y}_t(z) > \}.$$

2.4.4 Design Algorithm of a (k,m,N) COPQ

Just as in the case of the (k,m,N) PQ, the (k,m,N) COPQ can be thought of as a combination of m (k,N) channel optimized vector quantizers

$$\mathrm{COPQ}(\underline{\mathbf{x}}) = (\mathrm{COVQ}_1(\underline{\mathrm{x}}_1), \ldots, \mathrm{COVQ}_m(\underline{\mathrm{x}}_m))$$

where $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$, and $\underline{\mathrm{x}}_j \in \mathbb{R}^k$ and COVQ_j is a (k,N) COVQ, for $j = 1, \ldots, m$. Hence a (k,m,N) COPQ can be designed in the same way as a (k,m,N) PQ, where the individual (k,N) channel optimized vector quantizers are designed independently, using the algorithm of Section 2.2.5. The initial codebooks used in this algorithm is the codebook of a (k,m,N) PQ designed using the algorithm in Section 2.3.3.

Chapter 3

Sample Adaptive Product Quantizer

The sample adaptive product quantizer (SAPQ) [11] is based on adaptive quantization. For every source sample vector, the SAPQ employs a codebook from a previously designed set of 2^η codebooks, available at both the encoder and the decoder. The 2^η codebooks are actually codebooks of 2^η product quantizers. Note, the codebook used can change, or adapt, for every source sample vector. So when transmitting the indexes, representing the codevectors or reconstruction vectors, the encoder must also transmit an additional index indicating the codebook used for that source sample vector. This chapter describes the SAPQ, and through numerical results we illustrate the advantages of the SAPQ.

3.1 m-SAPQ Model

Codebook: A (k,m,N,η) m-SAPQ is constructed from a set of 2^η codebooks $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$. Each codebook $\mathbf{C}_j$ is a product of m codebooks $\{\mathrm{C}_{s,j}\}_{s=1}^{m}$ that are subsets of $\mathbb{R}^k$ with cardinality N. In other words, each codebook $\mathrm{C}_{s,j}$ contains N codevectors $\underline{c}_i^{[s,j]}$, where $i = 1, \ldots, N$, that belong to $\mathbb{R}^k$

$$\mathbf{C}_j = \mathrm{C}_{1,j} \times \ldots \times \mathrm{C}_{m,j} \text{ such that } \mathrm{C}_{s,j} = \{\underline{c}_i^{[s,j]}\}_{i=1}^{N} \text{ and } \underline{c}_i^{[s,j]} \in \mathbb{R}^k. \tag{3.1}$$

For $\underline{I} \in \mathrm{J}_N^m$ and $j \in \mathrm{J}_{2^\eta}$, define $\underline{\mathbf{c}}_{\underline{I}}^{[j]}$ to be a vector of codevectors $\underline{c}_{i_s}^{[s,j]}$ that are ordered as follows

$$\underline{\mathbf{c}}_{\underline{I}}^{[j]} = (\underline{c}_{i_1}^{[1,j]}, \ldots, \underline{c}_{i_m}^{[m,j]}) \text{ where } \underline{I} = (i_1, \ldots, i_m) \text{ and } i_s \in \mathrm{J}_N = 1, \ldots, N \text{ for } s = 1, \ldots, m$$

Note that $\underline{\mathbf{c}}_{\underline{I}}^{[j]} \in \mathbf{C}_j$, and $\underline{\mathbf{c}}_{\underline{I}}^{[j]}$ is referred to as a product codevector.

Structure: Figure 3.1 depicts how a source vector $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$, where $\underline{\mathrm{x}}_s \in \mathbb{R}^k$ for $s = 1, \ldots, m$ is quantized by a (k,m,N,η) m-SAPQ. The m-SAPQ encoder encodes the source sample $\underline{\mathbf{x}}$ into an index vector $\underline{I} \in \mathrm{J}_N^m$ and an overhead index $j^* \in \mathrm{J}_{2^\eta}$. Then the decoder decodes the index vector $\underline{I}$ and overhead index j^* into a product codevector $\underline{\mathbf{c}}_{\underline{I}}^{[j^*]}$.

Encoder: At the m-SAPQ encoder the encoding of a source vector $\underline{\mathbf{x}}$ is processed by a set of 2^η vector functions called product encoders (PE), $\{\mathrm{PE}_j\}_{j=1}^{2^\eta}$. Each vector function PE_j takes in a copy of the source vector $\underline{\mathbf{x}}$ and encodes it using the product codebook $\mathbf{C}_j$. Furthermore each PE_j has m component encoding functions, $\{\mathrm{E}_{s,j}\}_{s=1}^{m}$, such that each function $\mathrm{E}_{s,j}$ encodes subvector $\underline{\mathrm{x}}_s$ into an index $i_{s,j}$ using codebook $\mathrm{C}_{s,j}$, for $s = 1, \ldots, m$. The concatenation of all the indexes $i_{s,j}$ for $s = 1, \ldots, m$, forms the index vector $\underline{I}_j$ which is the output of PE_j

$$\mathrm{PE}_j(\underline{\mathbf{x}}) = (\mathrm{E}_{1,j}(\underline{x}_1), \ldots, \mathrm{E}_{m,j}(\underline{x}_m)) = \underline{I}_j \quad \text{where} \quad \mathrm{E}_{s,j}(\underline{x}_s) = i_{s,j} \in \mathrm{J}_N$$

$$\text{and} \qquad \underline{I}_j = (i_{1,j}, \ldots, i_{m,j}) \in \mathrm{J}_N^m.$$

Hence the m-SAPQ encoder internally produces 2^η index vectors $\{\underline{I}_j\}_{j=1}^{2^\eta}$. However only one index vector $\underline{I} \in \{\underline{I}_j\}_{j=1}^{2^\eta}$, is chosen to be transmitted to the decoder along with the index j^* representing PE_{j^*} (the PE that encoded $\underline{\mathbf{x}}$ into $\underline{I}$). Details of the encoding process and the choosing of index vector $\underline{I}$ and overhead index j^* are described in Section 3.2.2.

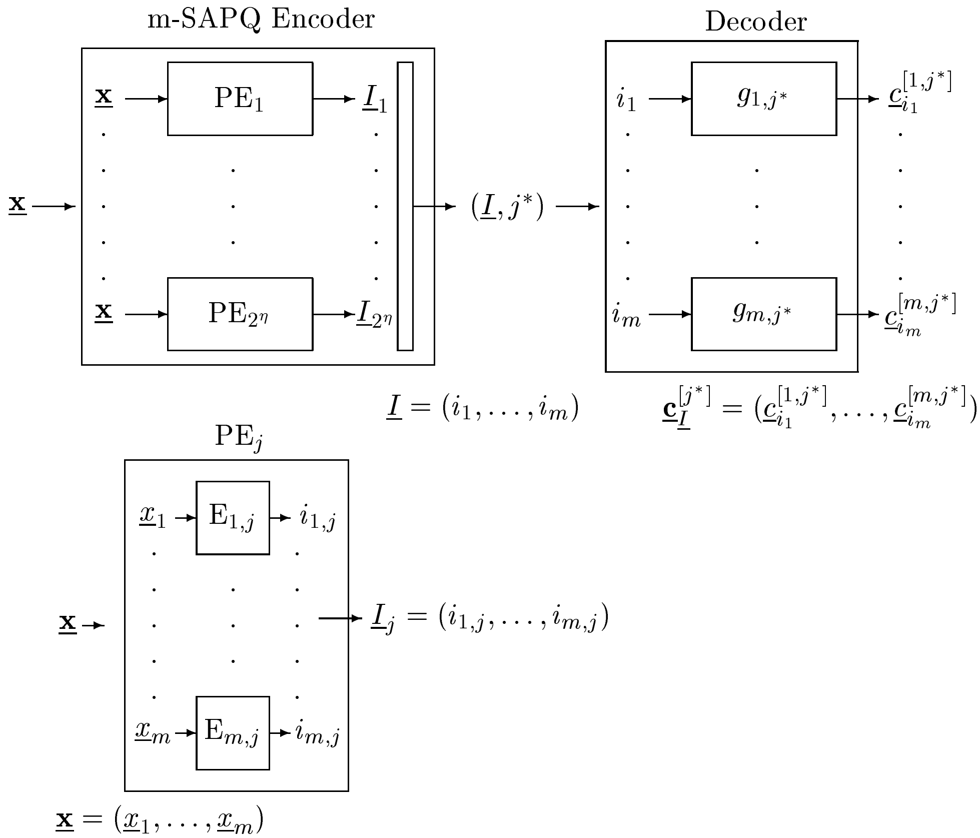

Figure 3.1: Figure of a (k,m,N,η) m-SAPQ and the j^{th} Product Encoder PE_j, where, $j = 1, \ldots, 2^\eta$, $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m) \in \mathbb{R}^{km}$, and $\underline{I} = (i_1, \ldots, i_m) \in \mathrm{J}_N^m$.

Decoder: The decoder consists of 2^η vector decoding functions $\{G_j\}_{j=1}^{2^\eta}$. The vector decoding function G_j consists of m component decoding functions, $\{g_{s,j}\}_{s=1}^{m}$. Each decoding function $g_{s,j}$ decodes an index i into the codevector $\underline{c}_i^{[s,j]}$ ($\underline{c}_i^{[s,j]} \in \mathrm{C}_{s,j}$). At the decoder the choice of which vector decoding function G_j, out of the set $\{G_j\}_{j=1}^{2^\eta}$, is determined by the received overhead index. When index vector $\underline{I}$ and index j^* are received, the decoder decodes index vector $\underline{I}$ using the vector decoding function G_{j^*}

$$\mathrm{Decoder}(\underline{I}, j^*) = G_{j^*}(\underline{I}) = (g_{1,j^*}(i_1), \ldots, g_{m,j^*}(i_m)) = (\underline{c}_{i_1}^{[1,j^*]}, \ldots, \underline{c}_{i_m}^{[m,j^*]}) = \underline{\mathbf{c}}_{\underline{I}}^{[j^*]}.$$

Naturally the decoding functions invert the encoding functions

$$\mathrm{PE}_j : \mathbb{R}^{km} \to \mathrm{J}_N^m \quad \text{and} \quad G_j : \mathrm{J}_N^m \to \mathbf{C}_j \subset \mathbb{R}^{km}$$
$$\mathrm{E}_{s,j} : \mathbb{R}^{k} \to \mathrm{J}_N \quad \text{and} \quad g_{s,j} : \mathrm{J}_N \to \mathrm{C}_{s,j} \subset \mathbb{R}^{k}$$

for $j = 1, \ldots, 2^\eta$ and $s = 1, \ldots, m$.

Rate: The source input vector $\underline{\mathbf{x}} = (x_1, \ldots, x_m)$, where $x_s \in \mathbb{R}^k$ for $s = 1, \ldots, m$, has a total of km source samples. The m-SAPQ encoder output for source vector $\underline{\mathbf{x}}$ is $\underline{I} = (i_1, \ldots, i_m)$ and j^*. So each $i_s \in \mathrm{J}_N$, for $s = 1, \ldots, m$, can be represented by a $\log_2 N$-bit codeword and $j^* \in \mathrm{J}_{2^\eta}$ can be represented by a η-bit codeword. In total there are $m \log_2 N + \eta$ bits needed to represent $\underline{\mathbf{x}}$. Hence the rate of a m-SAPQ is

$$R = \frac{\log_2 N}{k} + \frac{\eta}{km} \quad \text{bits/source sample.} \tag{3.2}$$

3.2 m-SAPQ Necessary Conditions for Optimality

3.2.1 m-SAPQ Distortion

To find necessary conditions for optimality of a (k,m,N,η) m-SAPQ, the expected mean square distortion of the (k,m,N,η) m-SAPQ needs to be calculated. To simplify the notation we define the following terms for $s = 1, \ldots, m$

$$\begin{aligned} \mathrm{v}_s(\underline{I}) &= \text{the } s^{th} \text{ index component of } \underline{I} = i_s \text{ ; } \ \mathrm{v}_s : \mathrm{J}_N^m \to \mathrm{J}_N & (3.3) \\ \mathrm{u}_s(\underline{\mathbf{x}}) &= \text{the } s^{th} \text{ vector component of } \underline{\mathbf{x}} = \underline{x}_s \text{ ; } \mathrm{u}_s : \mathbb{R}^{km} \to \mathbb{R}^k. & (3.4) \end{aligned}$$

Furthermore let $\mathbf{S}_{\underline{Z}}^{[j]}$ be the encoding region for index vector $\underline{Z}$ and index j of a (k,m,N,η) m-SAPQ, i.e,

$$\mathbf{S}_{\underline{Z}}^{[j]} = \{\underline{\mathbf{x}} \in \mathbb{R}^{km} : \text{m-SAPQ Encoder}(\underline{\mathbf{x}}) = (\underline{Z}, j)\} \tag{3.5}$$

where $\underline{Z} \in \mathrm{J}_N^m$, and $j \in \mathrm{J}_{2^\eta}$. In other words $\mathbf{S}_{\underline{Z}}^{[j]}$ is the set of source vectors $\underline{\mathbf{x}}$ such that $\underline{\mathbf{x}}$ is encoded into index vector $\underline{Z}$ and overhead index j. In total there are $2^\eta N^m$ encoding regions $\mathbf{S}_{\underline{Z}}^{[j]}$.

Let $\underline{\mathbf{c}}$ represent the reproduction, or the output of the decoder, of the m-SAPQ for source $\underline{\mathbf{X}}$ with a probability density function $\mathrm{p}(\underline{\mathbf{x}})$. Given the $2^\eta N^m$ encoding regions of a m-SAPQ $\mathbf{S}_{\underline{Z}}^{[j]}$, and the codebooks $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$, the expected mean square distortion of a (k,m,N,η) m-SAPQ can be found to be

$$\begin{aligned} \mathrm{D}_{m\text{-}SAPQ} &= E\{d(\underline{\mathbf{X}}, \underline{\mathbf{c}})\} & (3.6) \\ &= \sum_{j=1}^{2^\eta} \sum_{\underline{Z} \in \mathrm{J}_N^m} E\{d(\underline{\mathbf{X}}, \underline{\mathbf{c}}_{\underline{Z}}^{[j]}) | \underline{\mathbf{X}} \in \mathbf{S}_{\underline{Z}}^{[j]}\} \mathrm{P}(\underline{\mathbf{X}} \in \mathbf{S}_{\underline{Z}}^{[j]}) & (3.7) \\ &= \sum_{j=1}^{2^\eta} \sum_{\underline{Z} \in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{Z}}^{[j]}} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_s(\underline{Z})}^{[s,j]}\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}. & (3.8) \end{aligned}$$

Note that $\underline{\mathbf{x}}$ and $\underline{\mathbf{c}}$ are vectors of subvectors with their distortion defined to be the sum of the squared distance between their corresponding subvectors. So if $\underline{\mathbf{x}} \in \mathbf{S}_{\underline{Z}}^{[j]}$ then

$$d(\underline{\mathbf{x}}, \underline{\mathbf{c}}_{\underline{Z}}^{[j]}) = \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_s(\underline{Z})}^{[s,j]}\|^2.$$

3.2.2 m-SAPQ Optimal Encoding

We next consider the following: given product codebooks $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$, how can we determine the optimal encoding function so as to minimize the mean square distortion when a source sample $\underline{\mathbf{x}}$ is encoded into an index vector $\underline{I}$ and index j^*, using a (k,m,N,η) m-SAPQ encoder?

Let $\underline{\mathbf{c}}$ be the reproduction, or output of the m-SAPQ decoder, of $\underline{\mathbf{x}}$ and let $\underline{\mathbf{x}} \in \mathbf{S}_{\underline{Z}}^{[j]}$, then

$$\begin{aligned} d(\underline{\mathbf{x}}, \underline{\mathbf{c}}) &= \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_s(\underline{Z})}^{[s,j]}\|^2 \\ &\geq \min_{\underline{Z}\in \mathrm{J}_N^m} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_s(\underline{Z})}^{[s,j]}\|^2 \\ &\geq \min_{j} \min_{\underline{Z}\in \mathrm{J}_N^m} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_s(\underline{Z})}^{[s,j]}\|^2. \end{aligned}$$

Hence there are two optimizations to be implemented. As a consequence of the structure of a m-SAPQ, the first optimization is done by each product encoder,for $j = 1, \ldots, 2^\eta$ PE_j

$$\begin{aligned} \mathrm{PE}_j(\underline{\mathbf{x}}) &= \arg \min_{\underline{Z}\in \mathrm{J}_N^m} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_s(\underline{Z})}^{[s,j]}\|^2 = \underline{I}_j && (3.9) \\ &= (\arg \min_{z_1 \in \mathrm{J}_N} \|\mathrm{u}_1(\underline{\mathbf{x}}) - \underline{c}_{z_1}^{[1,j]}\|^2, \ldots, \arg \min_{z_m \in \mathrm{J}_N} \|\mathrm{u}_m(\underline{\mathbf{x}}) - \underline{c}_{z_m}^{[m,j]}\|^2) && (3.10) \end{aligned}$$

where $\underline{Z} = (z_1, \ldots, z_m)$, and each constituent encoder $\mathrm{E}_{s,j}$ of PE_j produces the s^{th} index component of $\underline{I}_j$

$$\mathrm{E}_{s,j}(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg \min_{z_s \in \mathrm{J}_N} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{z_s}^{[s,j]}\|^2 = \mathrm{v}_s(\underline{I}_j) \tag{3.11}$$

independently of each other. The independence in encoding is again a consequence of the structure of the m-SAPQ. The mean squared distortion incurred by each product encoder PE_j, and associated to $\underline{I}_j$, is

$$\mathrm{D}_j(\underline{\mathbf{x}}) = \min_{\underline{Z}\in \mathrm{J}_N^m} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j]}_{\mathrm{v}_s(\underline{Z})}\|^2. \tag{3.12}$$

Within the encoder we have 2^η index vectors $\{\mathrm{PE}_j(\underline{\mathbf{x}}) = \underline{I}_j\}_{j=1}^{2^\eta}$ each with an associated distortion of $\{(\underline{I}_j, \mathrm{D}_j(\underline{\mathbf{x}}))\}_{j=1}^{2^\eta}$, for $j = 1, \ldots, 2^\eta$. Out of the 2^η index vectors $\{\underline{I}_j\}_{j=1}^{2^\eta}$, the index vector $\underline{I}$ with the minimum distortion $\mathrm{D}_{j^*}(\underline{\mathbf{x}}) = \min_j \mathrm{D}_j$, is chosen to be transmitted to the decoder. The PE that produced the index vector $\underline{I}$ is distinguished by transmitting the overhead index vector j^* along with $\underline{I}$. Thus the optimal encoding regions are given by

$$\mathbf{S}_{\underline{I}}^{[j^*]} = \left\{ \underline{\mathbf{x}} \in \mathbb{R}^{km} : j^* = \arg\min_{j\in \mathrm{J}_{2^\eta}} D_j(\underline{\mathbf{x}}) \text{ and } \underline{I} = \underline{I}_{j^*} = \mathrm{PE}_{j^*}(\underline{\mathbf{x}}) \right\}. \tag{3.13}$$

With $\underline{I}$ and j^* defined above, the optimal distortion given the codebooks $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$ is

$$\mathrm{D}_{m\text{-}SAPQ} = E\{\min_j \mathrm{D}_j(\underline{\mathbf{x}})\} = \sum_{j^*=1}^{2^\eta} \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j^*]}} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j^*]}_{\mathrm{v}_s(\underline{I})}\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}. \tag{3.14}$$

3.2.3 m-SAPQ Optimal Decoding

In the previous section we assumed that the set $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$ was given, in other words the set of all codevectors $\{\underline{c}_l^{[s,j]}\}$ was assumed to be given. We will now find the set of all optimal codevectors assuming that the set of $2^\eta N^m$ optimal encoding regions $\{\mathbf{S}_{\underline{I}}^{[j^*]}\}$ are given. The distortion with the encoding regions $\{\mathbf{S}_{\underline{I}}^{[j^*]}\}$ defined as in (3.13) is

$$\mathrm{D}_{m\text{-}SAPQ} \;=\; \sum_{j^*=1}^{2^\eta} \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j^*]}} \sum_{t=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[t,j^*]}_{\mathrm{v}_t(\underline{I})}\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}.$$

Take the partial derivative with respect to the codevector $\underline{c}_i^{[s,j]}$ by separating the i^{th} codevector of codebook $\mathrm{C}_{s,j}$, where $s \in \mathrm{J}_m$, $j \in \mathrm{J}_{2^\eta}$ and $i \in \mathrm{J}_N$, in the above distortion as follows

$$\begin{aligned}
\mathrm{D}_{m\text{-}SAPQ} &= \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j]}} \sum_{t=1}^{m} \|\mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{I})}^{[t,j]}\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \\
&\quad + \sum_{j^*:j^*\neq j} \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j^*]}} \sum_{t=1}^{m} \|\mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{I})}^{[t,j^*]}\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \\
&= \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j]}} \left\{ \|\mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_s(\underline{I})}^{[s,j]}\|^2 + \sum_{t:t\neq s} \|\mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{I})}^{[t,j]}\|^2 \right\} \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \\
&\quad + \sum_{j^*:j^*\neq j} \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j^*]}} \sum_{t=1}^{m} \|\mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{I})}^{[t,j^*]}\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}.
\end{aligned}$$

Setting the resultant derivative to zero we get

$$0 = \sum_{\underline{I}:\mathrm{v}_s(\underline{I})=i} \int_{\mathbf{S}_{\underline{I}}^{[j]}} -\mathrm{u}_s(\underline{\mathbf{x}}) + \underline{c}_i^{[s,j]} \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}.$$

Solving for $\underline{c}_i^{[s,j]}$ we get the centroids

$$\underline{c}_i^{[s,j]} = \frac{\int_{S_i^{[s,j]}} \mathrm{u}_s(\underline{\mathbf{x}}) \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}}{\int_{S_i^{[s,j]}} \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}} \tag{3.15}$$

where the partition cells $S_i^{[s,j]}$ are defined to be

$$\begin{aligned}
S_i^{[s,j]} &= \bigcup_{\underline{I}:\mathrm{v}_s(\underline{I})=i} \mathbf{S}_{\underline{I}}^{[j]} && (3.16) \\
&= \{\underline{\mathbf{x}} \in \mathbb{R}^{km} : \text{m-SAPQ Encoder}(\underline{\mathbf{x}}) = ((i_1, \ldots, i_m), j) \text{ and } i_s = i\}. && (3.17)
\end{aligned}$$

Note that there are $2^\eta Nm$ partition cells $S_i^{[s,j]}$, one for each codevector $\underline{c}_i^{[s,j]}$, and there are $2^\eta N^m$ encoding regions $\mathbf{S}_{\underline{I}}^{[j^*]}$. Due the inherent nature of the m-SAPQ, although there are $2^\eta Nm$ codevectors, when encoding we have a choice of $2^\eta N^m$ encoding regions $\mathbf{S}_{\underline{I}}^{[j^*]}$ or product codevectors $\underline{\mathbf{c}}_{\underline{I}}^{[j^*]}$.

3.3 Example of a (k,m,N,η) m-SAPQ

This example is intended to better illustrate the notation, and the encoding and decoding process of a m-SAPQ. Consider a (1,2,2,1) m-SAPQ with a codebook as depicted in Figure 3.2. Consider a sample of the source vector $\underline{\mathbf{x}} = (\mathrm{x}_1, \mathrm{x}_2)$ located as shown in Figure 3.2. When this source sample $\underline{\mathbf{x}}$ is encoded by the (1,2,2,1) m-SAPQ with codebooks $\{\mathbf{C}_1, \mathbf{C_2}\}$, the output of the product encoders is:

$$\begin{aligned} \mathrm{PE}_1(\underline{\mathbf{x}}) &= (\mathrm{E}_{1,1}(\mathrm{x}_1), \mathrm{E}_{2,1}(\mathrm{x}_2)) = (1,1) = \underline{\mathrm{I}}_1 \\ \mathrm{PE}_2(\underline{\mathbf{x}}) &= (\mathrm{E}_{1,2}(\mathrm{x}_1), \mathrm{E}_{2,2}(\mathrm{x}_2)) = (1,1) = \underline{\mathrm{I}}_2. \end{aligned}$$

The set of product encoder functions $\{\mathrm{PE}_1, \mathrm{PE}_2\}$ and the set of encoder functions $\{\mathrm{E}_{1,1}, \mathrm{E}_{2,1}, \mathrm{E}_{1,2}, \mathrm{E}_{2,2}\}$ are formulated by (3.10) and (3.11) respectively. The resulting distortions of each product encoder is

$$\begin{aligned} \mathrm{D}_1(\underline{\mathbf{x}}) &= (\mathrm{x}_1 - \underline{c}_1^{[1,1]})^2 + (\mathrm{x}_2 - \underline{c}_1^{[2,1]})^2 = \|\underline{\mathbf{x}} - \underline{\mathbf{c}}_{(1,1)}^{[1]}\|^2 \\ \mathrm{D}_2(\underline{\mathbf{x}}) &= (\mathrm{x}_1 - \underline{c}_1^{[1,2]})^2 + (\mathrm{x}_2 - \underline{c}_1^{[2,2]})^2 = \|\underline{\mathbf{x}} - \underline{\mathbf{c}}_{(1,1)}^{[2]}\|^2. \end{aligned}$$

Clearly $\underline{\mathbf{x}}$ is closer to product codevector $\underline{\mathbf{c}}_{(1,1)}^{[1]}$ then product codevector $\underline{\mathbf{c}}_{(1,1)}^{[2]}$, hence the output of this m-SAPQ encoder is

$$\text{m-SAPQ Encoder}(\underline{\mathbf{x}}) = ((1,1),1) = (\underline{I}_1, 1).$$

At the decoder, there are two decoding functions $\{G_1, G_2\}$. Since in this case the overhead index is 1, the decoder output is

$$\mathrm{Decoder}((1,1),1) = G_1(1,1) = (g_1(1), g_1(1)) = (c_1^{[1,1]}, c_1^{(2,1)}).$$

Hence the m-SAPQ takes in the source sample $\underline{\mathbf{x}} = (\mathrm{x}_1\mathrm{x}_2)$ and reproduces it as $(c_1^{[1,1]}, c_1^{(2,1)})$.

$$\mathbf{C}_1 = \mathrm{C}_{1,1} \times \mathrm{C}_{2,1} = \{\underline{\mathbf{c}}^{[1]}_{(1,1)}, \underline{\mathbf{c}}^{[1]}_{(1,2)}, \underline{\mathbf{c}}^{[1]}_{(2,1)}, \underline{\mathbf{c}}^{[1]}_{(2,2)}\}$$

$$\mathbf{C}_2 = \mathrm{C}_{1,2} \times \mathrm{C}_{2,2} = \{\underline{\mathbf{c}}^{[2]}_{(1,1)}, \underline{\mathbf{c}}^{[2]}_{(1,2)}, \underline{\mathbf{c}}^{[2]}_{(2,1)}, \underline{\mathbf{c}}^{[2]}_{(2,2)}\}$$

$$\mathrm{C}_{1,1} = \{c_1^{[1,1]}, c_2^{[1,1]}\}$$

$$\mathrm{C}_{2,1} = \{c_1^{[2,1]}, c_2^{[2,1]}\}$$

$$\mathrm{C}_{1,2} = \{c_1^{[1,2]}, c_2^{[1,2]}\}$$

$$\mathrm{C}_{2,2} = \{c_1^{[2,2]}, c_2^{[2,2]}\}$$

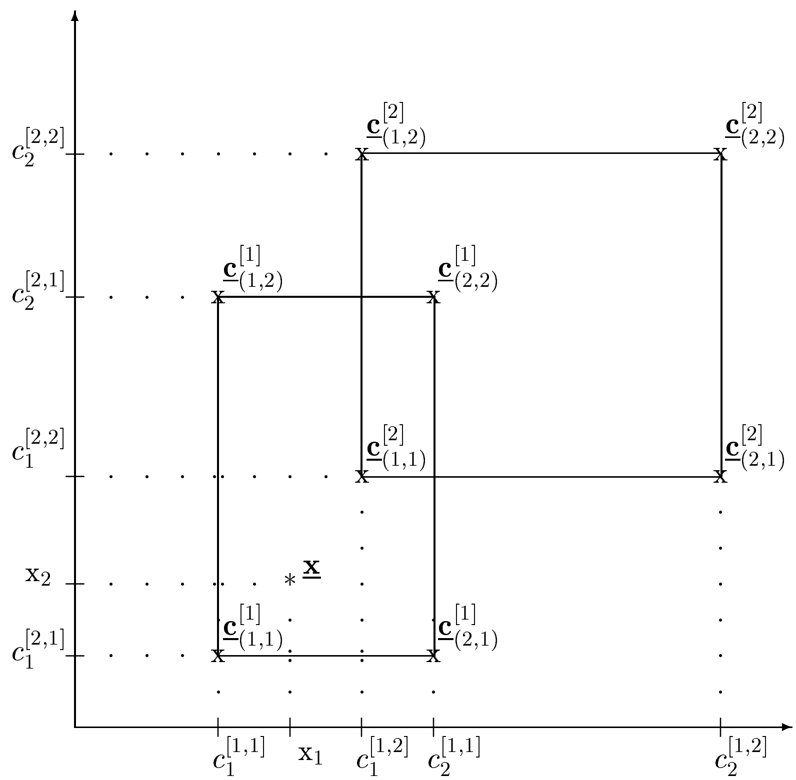

Figure 3.2: Example of a (1,2,2,1) m-SAPQ.

3.4 1-SAPQ

The 1-SAPQ is a particular case of a m-SAPQ. In the case of the (k,m,N,η) m-SAPQ, the 2^η codebooks of (3.1), were products of m individual codebooks that can be different from each other. For the (k,m,N,η) 1-SAPQ, the 2^η codebooks are products of the same codebook. The (k,m,N,η) 1-SAPQ only requires a set of codebooks $\{\mathrm{C}_j\}_{j=1}^{2^\eta}$, where each codebook C_j is a subset of $\mathbb{R}^k$ of cardinality N; i.e.

$$\mathrm{C}_j = \{\underline{c}_i^{[j]}\}_{i=1}^{N} \text{ such that } \underline{c}_i^{[j]} \in \mathbb{R}^k.$$

The structure of a (k,m,N,η) 1-SAPQ is depicted in Figure 3.3. The source sample vector $\underline{\mathbf{x}} = (x_1, \ldots, x_m)$, is encoded into an index vector by a set of 2^η vector functions called repeated encoders (RE), $\{\mathrm{RE}_j\}_{j=1}^{2^\eta}$. Each repeated encoder RE_j encodes $\underline{\mathbf{x}}$ into an index vector $\underline{I}_j$, by using the same encoder function E_j on each subvector x_s of $\underline{\mathbf{x}}$, for $s = 1, \ldots, m$. Out of the set $\{\underline{I}_j\}_{j=1}^{2^\eta}$, produced by the 2^η repeated encoders, only one index vector $\underline{I}_{j^*}$ with the minimum distortion is chosen and transmitted to the decoder, along with an index indicating which of the 2^η repeated encoders produced $\underline{I}_{j^*}$. The decoder then simply inverts the encoding functions.

Distortion: Given the codebooks $\{\mathrm{C}_j\}_{j=1}^{2^\eta}$ and encoding regions $\{\mathbf{S}_{\underline{Z}}^{[j]}\}$, the distortion of a (k,m,N,η) 1-SAPQ can be derived to be

$$\begin{aligned}
\mathrm{D}_{\textit{1-SAPQ}} &= E\{d(\underline{\mathbf{X}}, \underline{\mathbf{c}})\} && (3.18)\\
&= \sum_{j=1}^{2^\eta} \sum_{\underline{Z}\in \mathrm{J}_N^m} E\{d(\underline{\mathbf{X}}, \underline{\mathbf{c}}_{\underline{Z}}^{[j]}) | \underline{\mathbf{X}} \in \mathbf{S}_{\underline{Z}}^{[j]}\} P(\underline{\mathbf{X}} \in \mathbf{S}_{\underline{Z}}^{[j]}) && (3.19)\\
&= \sum_{j=1}^{2^\eta} \sum_{\underline{Z}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{Z}}^{[j]}} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{V}_s(\underline{Z})}^{[j]}\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}. && (3.20)
\end{aligned}$$

Optimal Encoding: Given the codebooks $\{\mathrm{C}_j\}_{j=1}^{2^\eta}$, the residual distortion incurred

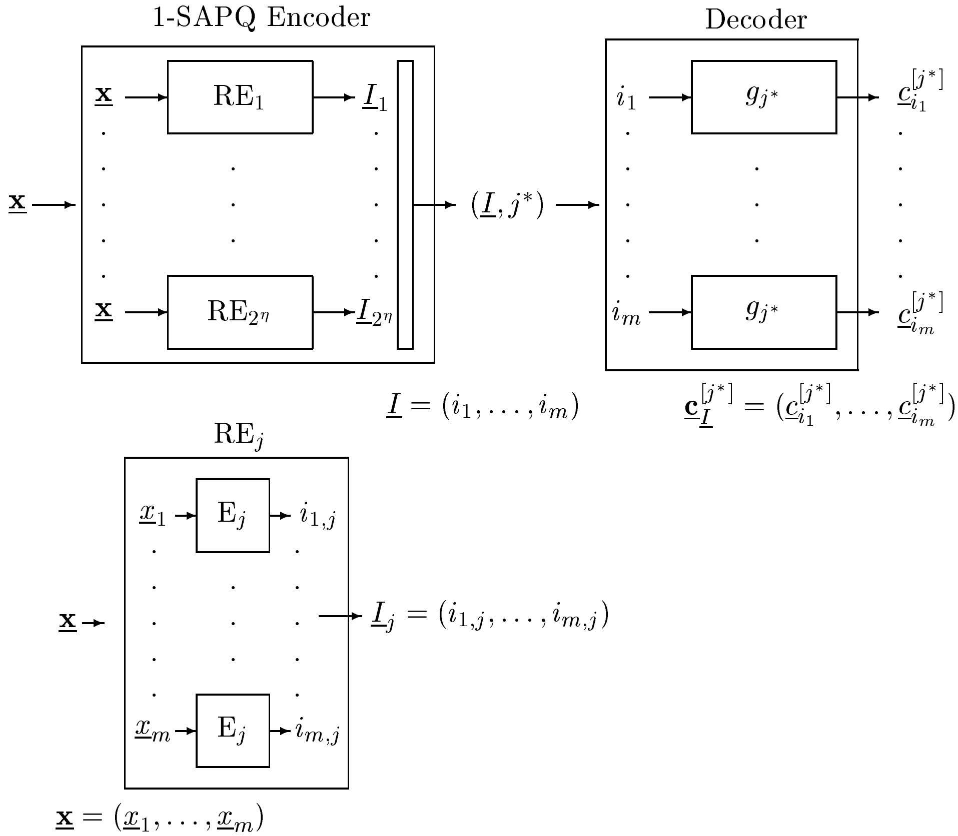

Figure 3.3: Figure of a (k,m,N,η) 1-SAPQ and the j^{th} Repeated Encoder RE_j, where, $j \in \mathrm{J}_{2^\eta}$, $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m) \in \mathbb{R}^{km}$, and $\underline{I} = (i_1, \ldots, i_m) \in \mathrm{J}_N^m$

by each repeated encoder RE_j is given by

$$\mathrm{D}_j(\underline{\mathbf{x}}) = \min_{\underline{Z} \in \mathrm{J}_N^m} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[j]}_{\mathrm{v}_s(\underline{Z})}\|^2. \tag{3.21}$$

Then the optimal encoding regions of a (k,m,N,η) 1-SAPQ can be derived to be

$$\mathbf{S}^{[j^*]}_{\underline{I}} = \left\{ \underline{\mathbf{x}} \in \mathbb{R}^{km} : j^* = \arg\min_{j \in \mathrm{J}_{2^\eta}} \mathrm{D}_j(\underline{\mathbf{x}}) \text{ and } \underline{I} = \underline{I}_{j^*} = \mathrm{RE}_{j^*}(\underline{\mathbf{x}}) \right\} \tag{3.22}$$

where $\underline{I} \in \mathrm{J}_N^m$, $j^* = 1, \ldots, 2^\eta$,

$$\mathrm{RE}_j(\underline{\mathbf{x}}) = \arg\min_{\underline{Z} \in \mathrm{J}_N^m} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[j]}_{\mathrm{v}_s(\underline{Z})}\|^2 = \underline{I}_j \tag{3.23}$$

$$= (\arg\min_{z_1 \in \mathrm{J}_N} \|\mathrm{u}_1(\underline{\mathbf{x}}) - \underline{c}^{[j]}_{z_1}\|^2, \ldots, \arg\min_{z_m \in \mathrm{J}_N} \|\mathrm{u}_m(\underline{\mathbf{x}}) - \underline{c}^{[j]}_{z_m}\|^2) \tag{3.24}$$

, and

$$\mathrm{E}_j(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg\min_{z_s \in \mathrm{J}_N} \|\mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{z_s}^{[j]}\|^2 = \mathrm{v}_s(\underline{I}). \tag{3.25}$$

Optimal Decoding: Now given the encoding regions $\{\mathbf{S}_{\underline{I}}^{[j^*]}\}$ defined as in (3.22), the optimal codevectors can be found, as in Section 3.2.3, to be

$$\underline{c}_i^{[j]} = \frac{\sum_{t=1}^m \int_{S_i^{[t,j]}} \mathrm{u}_t(\underline{\mathbf{x}})\mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}}}{\sum_{t=1}^m \int_{S_i^{[t,j]}} \mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}}} \tag{3.26}$$

where $i = 1, \ldots, N$, $j = 1, \ldots, 2^\eta$, and for $t = 1, \ldots, m$

$$\begin{aligned} S_i^{[t,j]} &= \bigcup_{\underline{I}:\mathrm{v}_t(\underline{I})=i} \mathbf{S}_{\underline{I}}^{[j]} && (3.27) \\ &= \{\underline{\mathbf{x}} \in \mathbb{R}^{km} : \text{1-SAPQ Encoder}(\underline{\mathbf{x}}) = ((i_1, \ldots, i_m), j) \text{ and } i_t = i\}. && (3.28) \end{aligned}$$

3.5 Encoding Complexity and Storage Requirements

Let us now compare the encoding complexity and storage requirements for each of the following quantizers: VQ, PQ, m-SAPQ, and 1-SAPQ. We define the encoding complexity to be the total amount of multiplications required to encode a source sample (complexity=total multiplications required for encoding/source sample). The storage requirements are measured as the total number of scalar values that are required to be stored at the encoder and decoder, in order to implement the quantizer in question (storage=total scalars required for implementation).The results are summarized in Table 3.1.

3.5.1 (k_1,N_1) VQ

Encoding Complexity: From Section 2.1.2, the (k_1,N_1) VQ encoding function is

$$\mathrm{E}(\underline{\mathrm{x}}) = \arg\min_{i\in \mathrm{J}_{N_1}} \|\underline{\mathrm{x}} - \underline{c}_i\|^2 \tag{3.29}$$

where $\mathrm{J}_{N_1} = \{1,\ldots,N_1\}$ and $\underline{\mathrm{x}} \in \mathbb{R}^{k_1}$. This encoding function (3.29) requires k_1 multiplications to be performed N_1 times for a source sample $\underline{\mathrm{x}}$ of dimension k_1. Thus the encoding complexity of a (k_1,N_1) VQ is

$$(k_1,N_1) \text{ VQ Complexity} = \frac{k_1 N_1}{k_1} = N_1.$$

Storage Requirements: In order to implement the (k_1,N_1) VQ only the codebook,(2.1) , of the quantizer needs to be stored, this is a total of $k_1 N_1$ scalars, bringing the storage requirements to

$$(k_1,N_1) \text{ VQ Storage} = k_1 N_1.$$

3.5.2 (k_2,m_2,N_2) PQ

Encoding Complexity: From Section 2.3.2, the constituent encoding function of a (k_2,m_2,N_2) PQ is

$$\mathrm{E}_t(\underline{\mathrm{x}}) = \arg\min_{z\in \mathrm{J}_{N_2}} \|\underline{\mathrm{x}} - \underline{c}_z^{[t]}\|^2 \tag{3.30}$$

where $\mathrm{J}_{N_2} = \{1,\ldots,N_2\}$, $\underline{\mathrm{x}} \in \mathbb{R}^{k_2}$ and $t = 1,\ldots,m_2$. This operation (3.30) requires k_2 multiplications to be performed N_2 times. There are m_2 such encoding functions $\mathrm{E}_t(\underline{\mathrm{x}})$, each encoding a component of the source sample $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1,\ldots,\underline{\mathrm{x}}_2)$, where $\underline{\mathrm{x}}_t \in \mathbb{R}^{k_2}$. This is a total of $k_2 m_2 N_2$ multiplications for $k_2 m_2$ scalar source samples. Hence the encoding complexity is

$$(k_2,m_2,N_2) \text{ PQ Complexity} = \frac{k_2 m_2 N_2}{k_2 m_2} = N_2.$$

Storage Requirements: Only the codebook of the (k_2,m_2,N_2) PQ needs to be stored to implement it. This is a total of $k_2 m_2 N_2$ scalars according to (2.19). Thus the storage requirements are

$$(k_2,m_2,N_2) \text{ PQ Storage} = k_2 m_2 N_2.$$

3.5.3 (k_3,m_3,N_3,η_3) m-SAPQ

Encoding Complexity: From Section 3.2.2, the s^{th} encoder function of the j^{th} product encoder PE_j for a (k_3,m_3,N_3,η_3) m-SAPQ is

$$\text{E}_{s,j}(\text{u}_s(\underline{\mathbf{x}})) = \arg \min_{z_s \in \text{J}_{N_3}} \|\text{u}_s(\underline{\mathbf{x}}) - \underline{c}_{z_s}^{[s,j]}\|^2 \tag{3.31}$$

where $\text{J}_{N_3} = \{1, \ldots, N_3\}$, $\underline{\mathbf{x}} \in \mathbb{R}^{k_3 m_3}$, and $s = 1, \ldots, m_3$ and $j = 1, \ldots, 2^{\eta_3}$. The operation (3.31) requires k_3 multiplications done N_3 times to encode k_3 source scalar samples ($\text{u}_s(\underline{\mathbf{x}}) \in \mathbb{R}^{k_3}$). There are m_3 such encoding functions in each of the 2^{η_3} product encoders PE_j. Thus each $\text{PE}_j(\underline{\mathbf{x}})$ requires $k_3 m_3 N_3$ multiplications to encode $k_3 m_3$ source samples ($\underline{\mathbf{x}} \in \mathbb{R}^{k_3 m_3}$). There are 2^{η_3} of these product encoders $\text{PE}_j(\underline{\mathbf{x}})$, in a ($k_3$,$m_3$,$N_3$,$\eta_3$) m-SAPQ. Hence the encoding complexity is

$$(k_3,m_3,N_3,\eta_3) \text{ m-SAPQ Complexity} = 2^{\eta_3} \frac{k_3 m_3 N_3}{k_3 m_3} = 2^{\eta_3} N_3.$$

Storage Requirements: To store the codebook of a (k_3,m_3,N_3,η_3) m-SAPQ we require $k_3 m_3 N_3 2^{\eta_3}$ scalars. Thus the storage requirements are

$$(k_3,m_3,N_3,\eta_3) \text{ m-SAPQ Storage} = k_3 m_3 N_3 2^{\eta_3}.$$

To keep the complexity of a (k_3,m_3,N_3,η_3) m-SAPQ less than that of a (k_1,N_1) VQ, of the same rate, the parameters (k_3,m_3,N_3,η_3) must be chosen so that $2^{\eta_3} N_3 < N_1$ while keeping $R = \frac{\log_2 N_3}{k_3} + \frac{\eta_3}{k_3 m_3} = \frac{\log_2 N_1}{k_1}$ equal. In [11], it is also further advised to keep $\beta = \frac{m_3}{N_3}$ as high as possible.

Quantizer	Complexity	Storage
(k_1,N_1) VQ	N_1	$k_1 N_1$
(k_2,m_2,N_2) PQ	N_2	$k_2 m_2 N_2$
(k_3,m_3,N_3,η_3) m-SAPQ	$2^{\eta_3} N_3$	$k_3 m_3 N_3 2^{\eta_3}$
(k_4,m_4,N_4,η_4) 1-SAPQ	$2^{\eta_4} N_4$	$k_4 N_4 2^{\eta_4}$

Table 3.1: Table of Encoding Complexity and Storage Requirements for Quantizers designed under noiseless conditions.

3.5.4 (k_4,m_4,N_4,η_4) 1-SAPQ

Encoding Complexity: The encoding complexity of the (k_4,m_4,N_4,η_4) 1-SAPQ is derived in the same way as the (k_3,m_3,N_3,η_3) m-SAPQ to be

$$(k_4,m_4,N_4,\eta_4) \text{ 1-SAPQ Complexity} = 2^{\eta_4}\frac{k_4 m_4 N_4}{k_4 m_4} = 2^{\eta_4} N_4.$$

This is the same as a (k_4,m_4,N_4,η_4) m-SAPQ.

Storage Requirements: The advantage of the (k_4,m_4,N_4,η_4) 1-SAPQ is a lesser storage requirement than that of a (k_4,m_4,N_4,η_4) m-SAPQ. This is because the same codebook C_j, for $j = 1, \ldots, 2^{\eta_4}$, is repeated within each repeated encoder RE_j. The storage requirements are then

$$(k_4,m_4,N_4,\eta_4) \text{ 1-SAPQ Storage} = k_4 N_4 2^{\eta_4}$$

which is $\frac{1}{m_4}$ less than that of a (k_4,m_4,N_4,η_4) m-SAPQ.

3.6 Design Algorithm for SAPQs

The design algorithm for the m-SAPQ is next described. This design with be based on the derivations of the necessary conditions for optimality (3.13) and (3.15). In this algorithm we iterate over the necessary conditions for optimality in the similar fashion to the Lind-Buzo-Gray (LBG) algorithm.

(k,m,N,η) m-SAPQ Algorithm

1. Set parameters k, m, N, η, the stopping threshold δ, the splitting constant k-dimensional vector $\underline{\epsilon} = (\epsilon, \ldots, \epsilon)$, the maximum number of iterations $Maxiter$, and M the total number of training vectors $\{\underline{\mathbf{x}}_f = (\underline{x}_{1,f}, \ldots, \underline{x}_{m,f})\}_{f=1}^{M}$. Start off with $\tau = 1$, $\rho = 0$, and initial codebooks $\{\mathrm{C}_{s,\tau}^{(0)}\}_{s=1}^{m}$.

2. If $\tau \geq 2^\eta$ stop otherwise split the codebooks using

$$\mathrm{C}_{s,j}^{(\rho)} = \mathrm{C}_{s,j}^{(\rho)} - \underline{\epsilon} \quad \text{and} \quad \mathrm{C}_{s,j+\tau}^{(\rho)} = \mathrm{C}_{s,j}^{(\rho)} + \underline{\epsilon}$$

   for $s = 1, \ldots, m$ and $j = 1, \ldots, \tau$ then double $\tau = \tau * 2$ and set $\rho = 0$. Note that ρ is a counter for the iterations and τ is a counter for the codebooks . At this point we have τ codebooks $\{\mathbf{C}_j^{(0)}\}_{j=1}^{\tau}$ where $\mathbf{C}_j^{(0)} = \mathrm{C}_{1,j}^{(0)} \times \ldots \times \mathrm{C}_{m,j}^{(0)}$.

3. For each f, encode $\underline{\mathbf{x}}_f$ into an index vector $\underline{I}$ and index j^* using the product encoders $\{PE_j^{(\rho)}\}_{j=1}^{\tau}$. This is done by encoding $\underline{\mathbf{x}}_f$ with each $PE_j^{(\rho)}$

$$PE_j^{(\rho)}(\underline{\mathbf{x}}_f) = \underline{I}_j = \arg\min_{\underline{Z} \in \mathrm{J}_N^m} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}_f) - \underline{c}_{\mathrm{V}_s(\underline{Z})}^{[s,j],(\rho)}\|^2.$$

   Then the optimal index vector $\underline{I}$ and overhead index j^* is chosen to be

$$\begin{aligned} j^* &= \arg\min_{j \in \mathrm{J}_\tau} \min_{\underline{Z} \in \mathrm{J}_N^m} \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}_f) - \underline{c}_{\mathrm{V}_s(\underline{Z})}^{[s,j],(\rho)}\|^2 \\ \underline{I} &= \underline{I}_{j^*} = PE_{j^*}^{(\tau)}(\underline{\mathbf{x}}_f). \end{aligned}$$

4. Once $\underline{\mathbf{x}}_f$ is encoded, $\underline{\mathbf{x}}_f$ can be put into the appropriate cells, (3.17). So if $\underline{\mathbf{x}}_f$ is encoded into $((i_1, \ldots, i_m), j^*)$ then

$$\underline{\mathbf{x}}_f \in S_{i_1}^{[1,j^*],(\rho)}, \ldots, \underline{\mathbf{x}}_f \in S_{i_m}^{[m,j^*],(\rho)}.$$

The resulting distortion is

$$D^{(\rho)}[\underline{\mathbf{x}}_f,\tau] = \sum_{s=1}^{m} \|\mathrm{u}_s(\underline{\mathbf{x}}_f) - \underline{c}^{[s,j^*],(\rho)}_{\mathrm{V}_s(\underline{I})}\|^2.$$

5. Repeat steps 3 and 4 for all $f = 1\ldots,M$. Once all the partitions have been made, calculate the centroids using

$$\underline{c}_i^{[s,j],(\rho+1)} = \frac{\sum_{\underline{\mathbf{x}}:\underline{\mathbf{x}}\in S_i^{[s,j^*],(\rho)}} \mathrm{u}_s(\underline{\mathbf{x}})}{\sum_{\underline{\mathbf{x}}:\underline{\mathbf{x}}\in S_i^{[s,j^*],(\rho)}}}$$

and update the codebooks to $\{\mathbf{C}_j^{(\rho+1)}\}_{j=1}^{\tau}$ using the new centroids. Finally calculate the overall distortion using

$$\mathcal{D}^{(\rho)}[\tau] = \frac{1}{kmM}\sum_{f=1}^{M} D^{(\rho)}[\underline{\mathbf{x}}_f,\tau].$$

6. Check $\frac{\mathcal{D}^{(\rho-1)}[\tau]-\mathcal{D}^{(\rho)}[\tau]}{\mathcal{D}^{(\rho)}[\tau]} \leq \delta$ or $\rho \geq Maxiter$, if so then go to step 2 otherwise $\rho = \rho + 1$ and go to step 3.

This algorithm assumes an initial set of codebooks $\mathbf{C}_1^{(0)}$ for the (k,m,N,η) m-SAPQ which is obtained from a (k,m,N) PQ, using the design algorithm of Section 2.3.3. Similarly the (k,m,N,η) 1-SAPQ algorithm starts off with only one codebook $\mathrm{C}_1^{(0)}$which can be obtained from a (k,N) VQ, designed using the LBG algorithm of Section 2.1.3. The algorithm for the design of the (k,m,N,η) 1-SAPQ codebooks is easily deduced from the above design.

3.7 Numerical Results

In order to illustrate the advantages of the sample adaptive product quantizers (SAPQ) over the generic quantizers, such as the vector quantizer (VQ) and the product quantizer (PQ), numerical results were produced. The design algorithms of Sections 2.1.3, 2.3.3 and 3.6, were used to design the VQ, PQ, m-SAPQ and 1-SAPQ. In all cases 200,000 training source samples were used to design each codebook. The sources considered were the unit variance and zero mean Gauss-Markov source and the memoryless Gaussian source. A Gauss-Markov source is a sequence $\{\mathbf{X}_i\}$ described by the recursion

$$\mathbf{X}_i = \rho \mathbf{X}_{i-1} + \mathbf{U}_i$$

where $\{\mathbf{U}_i\}$ is a sequence of independent and identically distributed (i.i.d) Gaussian random variables, and ρ is the correlation coefficient. A memoryless Gaussian sequence and a Gauss-Markov sequence was generated by setting the correlation coefficient to $\rho = 0.0$ and $\rho = 0.9$, respectively. The designed codebooks of the VQ, PQ, m-SAPQ and 1-SAPQ were tested by implementing the quantizers shown in Figure 3.4:

1. The quantizer is designed using the algorithms of Sections 2.1.3,2.3.3, and 3.6 with the design parameters: splitting constant $\epsilon = 0.001$, stopping threshold $\delta = 0.001$ and maximum number of iterations $Maxiter = 200$. Note that all the quantizers are designed using the same design parameters, and the same training sequence $\{\underline{\mathbf{x}}_f\}$. The final distortion of the quantizers is $\mathcal{D}[final]$.

2. The codebook $\mathbf{C}$ produced by the design algorithm is then used to implement the encoder. The encoder is then used to encode the testing sequence $\{\hat{\underline{\mathbf{x}}}_f\} \neq \{\underline{\mathbf{x}}_f\}$; a sequence that is entirely different from the training sequence.

3. The codebook produced from Step 1 is also used to implement a decoder. The resultant reconstruction of the testing sequence is $\{\underline{\mathbf{y}}_f\}$. The distortion of the

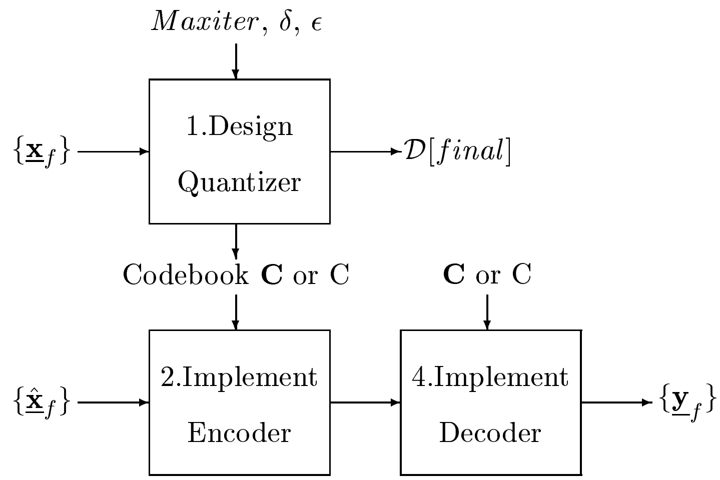

Figure 3.4: Block Diagram illustrating the validation of quantizers.

implemented quantizer can then be calculated as

$$\mathbf{D} = \sum_f \|\hat{\underline{\mathbf{x}}}_f - \underline{\mathbf{y}}_f\|^2.$$

The performance of the channel optimized quantizers is measured using the signal-to-distortion ratio (SDR) which is equal to SDR $= -10\log_{10}(\mathcal{D}[final])$ or SDR $= -10\log_{10}(\mathbf{D})$. When comparing the VQ, PQ and SAPQ of the same rate R, the criteria used for comparison is three-fold: namely the performance (SDR), the encoding complexity (complexity=total multiplications required for encoding/source sample), and the storage requirements (storage=total scalars required for implementation).

3.7.1 Comparing the VQ, m-SAPQ, and 1-SAPQ

Tables 3.6, and 3.7, show the performances (SDR $= -10\log_{10}(\mathcal{D}[final])$) of the VQ, the m-SAPQ and the 1-SAPQ at rates $R = 1.5, 2.5$ and 3.5, for memoryless Gaussian and Gauss-Markov sources, respectively. The encoding complexities and storage requirements are calculated using Table 3.1. These results are then tested as in Figure 3.4, and their validated performances (SDR $= -10\log_{10}(\mathbf{D})$) are then tabulated in Tables 3.8, and 3.9.

Memoryless Gaussian Sources: In Table 3.6 we see that for rates $R = 1.5, 2.5, 3.5$, for every (2,2^{2R}) VQ, we can find a 1-SAPQ and a m-SAPQ of comparable performance and less encoding complexity and storage requirements: compare the (2,8) VQ with the (1,2,2,1) 1-SAPQ and m-SAPQ, compare the (2,32) VQ with the (1,4,4,2) 1-SAPQ and m-SAPQ, compare the (2,128) VQ with the (1,4,8,2) 1-SAPQ and m-SAPQ. Note that as the rate gets higher the advantage of the 1-SAPQ and m-SAPQ over VQ increases. This is especially true for m-SAPQ. In order to attain an advantage of using the m-SAPQ over the VQ and 1-SAPQ, the dimension km must be high enough ($km > 2$).

Gauss-Markov Sources: For Gauss-Markov sources, Table 3.7 illustrates that for every rate R and (2,2^{2R}) VQ, we can find a 1-SAPQ of the same complexity and lower storage requirements, that outperforms the (2,2^{2R}) VQ. Compare the (2,8) VQ with the (1,4,2,2) 1-SAPQ, compare the (2,32) VQ with the (1,6,4,3) 1-SAPQ, compare the (2,128) VQ with the (1,8,8,4) 1-SAPQ. Note that the dimensions km of these 1-SAPQ's are higher than that of the VQ's, of the same rate, but their encoding complexity is the same and their storage requirements are lower.

3.7.2 Comparing PQ, m-SAPQ and 1-SAPQ

Tables 3.2 and 3.3 tabulate the performances of the PQ and the least complex m-SAPQ and 1-SAPQ of rates $R = 2.0, 3.0, 4.0$ and 5.0. These results are validated in Tables 3.4 and 3.5. From Tables 3.2, and 3.3 we see that in all cases the m-SAPQ outperforms the PQ. The performance gain of the m-SAPQ over the PQ increases as the rate R increases (0.77-0.24 dB) for memoryless Gaussian sources, Tables 3.2. For memoryless Gaussian sources the advantage of the 1-SAPQ over the PQ is realized as the rate increases ($R \geq 4.0$). But when the source is Gauss-Markov the 1-SAPQ has an advantage over the PQ when the rate is low ($R < 4.0$).

3.7.3 Comparing m-SAPQ and 1-SAPQ

In Tables 3.6, 3.7, 3.2, and 3.3, we see that the (k,m,N,η) m-SAPQ always performs equal to, or greater than, the (k,m,N,η) 1-SAPQ. The m-SAPQ outperforms the 1-SAPQ. Though the (k,m,N,η) m-SAPQ always has the disadvantages of having m times the storage requirement than the (k,m,N,η) 1-SAPQ. Note that the above conclusion is only contradicted by one result: namely in Table 3.7 compare (1,8,8,4) 1-SAPQ and the (1,8,8,4) m-SAPQ. The poor performance of the (1,8,8,4) m-SAPQ may have been due to the limitations in the size of the training sequence (200,000 samples). This hypothesis is supported by comparing Tables 3.7 and 3.9, where the difference in the performance of the (1,8,8,4) m-SAPQ with the testing sequence over the training sequence is exceptionally great (0.22 dB). A longer training sequence may have been required in order to design the codebook of the (1,8,8,4) m-SAPQ.

3.7.4 The Effect of $\beta = m/N$

At a given rate R we can clearly see from Tables 3.6 and 3.7, that the higher $\beta = m/N$ is, the better are the performances of the m-SAPQ and the 1-SAPQ. Of course the increase of $\beta = m/N$ increases the encoding complexity and storage requirements of the m-SAPQ and 1-SAPQ. Though just as a guideline, when trying to determine the best choice of parameters k, m, N and η of a SAPQ, one should always aim for a high $\beta = m/N$.

km	Quantizer		$R = 2.0$	$R = 3.0$	$R = 4.0$	$R = 5.0$
2	PQ	Codebook (N)	4	8	16	32
	$k = 1$	SNR (dB)	9.27	14.57	20.18	25.95
		Complexity	4	8	16	32
		Storage	8	16	32	64
2	1-SAPQ	Codebook (N)	2	4	8	16
	$k = 1, \eta = 1$	SNR (dB)	8.72	14.49	20.28	26.24
		Complexity	4	8	16	32
		Storage	4	8	16	32
		$\beta = m/N$	1	1/2	1/4	1/8
2	m-SAPQ	Codebook (N)	2	4	8	16
	$k = 1, \eta = 1$	SNR (dB)	9.51	15.03	20.84	26.72
		Complexity	4	8	16	32
		Storage	8	16	32	64
		$\beta = m/N$	1	1/2	1/4	1/8

Table 3.2: SDR (dB) performances comparison of the (k,m,N) PQ, the (k,m,N,η) 1-SAPQ, and the (k,m,N,η) m-SAPQ, at rates R, designed using 200,000 memoryless Gaussian training samples.

km	Quantizer		$R = 2.0$	$R = 3.0$	$R = 4.0$	$R = 5.0$
2	PQ	Codebook (N)	4	8	16	32
	$k = 1$	SNR (dB)	9.28	14.57	20.18	25.98
		Complexity	4	8	16	32
		Storage	8	16	32	64
2	1-SAPQ	Codebook (N)	2	4	8	16
	$k = 1$,$\eta = 1$	SNR (dB)	12.50	15.22	19.84	25.40
		Complexity	4	8	16	32
		Storage	4	8	16	32
		$\beta = m/N$	1	1/2	1/4	1/8
2	m-SAPQ	Codebook (N)	2	4	8	16
	$k = 1$,$\eta = 1$	SNR (dB)	12.50	15.36	20.61	26.22
		Complexity	4	8	16	32
		Storage	8	16	32	64
		$\beta = m/N$	1	1/2	1/4	1/8

Table 3.3: SDR (dB) performances comparison of the (k,m,N) PQ, the (k,m,N,η) 1-SAPQ, and the (k,m,N,η) m-SAPQ, at rates R, designed using 200,000 Gauss-Markov training samples.

km	Quantizer		$R=2.0$	$R=3.0$	$R=4.0$	$R=5.0$
2	PQ	Codebook (N)	4	8	16	32
	$k=1$	SNR (dB)	9.29	14.60	20.21	25.94
2	1-SAPQ	Codebook (N)	2	4	8	16
	$k=1,\eta=1$	SNR (dB)	8.74	14.50	20.25	26.20
2	m-SAPQ	Codebook (N)	2	4	8	16
	$k=1,\eta=1$	SNR (dB)	9.52	15.02	20.81	26.67

Table 3.4: SDR (dB) performances testing designs of the (k,m,N) PQ, the (k,m,N,η) 1-SAPQ, and the (k,m,N,η) m-SAPQ, at rates R, using 200,000 memoryless Gaussian testing samples.

km	Quantizer		$R=2.0$	$R=3.0$	$R=4.0$	$R=5.0$
2	PQ	Codebook (N)	4	8	16	32
	$k=1$	SNR (dB)	9.32	14.63	20.26	26.00
2	1-SAPQ	Codebook (N)	2	4	8	16
	$k=1,\eta=1$	SNR (dB)	12.54	15.27	19.87	25.47
2	m-SAPQ	Codebook (N)	2	4	8	16
	$k=1,\eta=1$	SNR (dB)	12.54	15.41	20.63	26.23

Table 3.5: SDR (dB) performances testing designs of the (k,m,N) PQ, the (k,m,N,η) 1-SAPQ, and the (k,m,N,η) m-SAPQ, at rates R, using 200,000 Gauss-Markov testing samples.

km	Quantizer		$R = 1.5$	$R = 2.5$	$R = 3.5$
2	VQ	Codebook (N)	8	32	128
		SNR (dB)	6.94	12.40	18.14
		Complexity	8	32	128
		Storage	16	64	256
2	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 1$	SNR (dB)	6.86	12.25	17.87
		Complexity	4	8	16
		Storage	4	8	16
		$\beta = m/N$	1	1/2	1/4
2	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 1$	SNR (dB)	6.86	12.25	17.87
		Complexity	4	8	16
		Storage	8	16	32
		$\beta = m/N$	1	1/2	1/4
4	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 2$	SNR (dB)	6.95	12.57	18.25
		Complexity	8	16	32
		Storage	8	16	32
		$\beta = m/N$	2	1	1/2
4	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 2$	SNR (dB)	7.08	12.60	18.42
		Complexity	8	16	32
		Storage	32	64	128
		$\beta = m/N$	2	1	1/2
6	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 3$	SNR (dB)	6.61	12.79	18.64
		Complexity	16	32	64
		Storage	16	32	64
		$\beta = m/N$	3	3/2	3/4
6	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 3$	SNR (dB)	7.34	12.90	18.80
		Complexity	16	32	64
		Storage	96	192	384
		$\beta = m/N$	3	3/2	3/4
8	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 4$	SNR (dB)	6.95	12.86	18.85
		Complexity	32	64	128
		Storage	32	64	128
		$\beta = m/N$	4	2	1
8	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 4$	SNR (dB)	7.52	13.15	19.13
		Complexity	32	64	128
		Storage	256	512	1024
		$\beta = m/N$	4	2	1

Table 3.6: SDR (dB) performances comparing the (k, N) VQ, the (k, m, N, η) 1-SAPQ, and the (k, m, N, η) m-SAPQ, at rates R, designed using 200,000 memoryless Gaussian training samples.

km	Quantizer		$R=1.5$	$R=2.5$	$R=3.5$
2	VQ	Codebook (N)	8	32	128
		SNR (dB)	10.79	16.25	21.89
		Complexity	8	32	128
		Storage	16	64	256
2	1-SAPQ	Codebook (N)	2	4	8
	$k=1,\eta=1$	SNR (dB)	8.01	12.43	17.63
		Complexity	4	8	16
		Storage	4	8	16
		$\beta=m/N$	1	1/2	1/4
2	m-SAPQ	Codebook (N)	2	4	8
	$k=1,\eta=1$	SNR (dB)	8.01	13.67	17.63
		Complexity	4	8	16
		Storage	8	16	32
		$\beta=m/N$	1	1/2	1/4
4	1-SAPQ	Codebook (N)	2	4	8
	$k=1,\eta=2$	SNR (dB)	10.90	13.91	18.65
		Complexity	8	16	32
		Storage	8	16	32
		$\beta=m/N$	2	1	1/2
4	m-SAPQ	Codebook (N)	2	4	8
	$k=1,\eta=2$	SNR (dB)	11.00	15.81	19.12
		Complexity	8	16	32
		Storage	32	64	128
		$\beta=m/N$	2	1	1/2
6	1-SAPQ	Codebook (N)	2	4	8
	$k=1,\eta=3$	SNR (dB)	11.56	16.55	19.57
		Complexity	16	32	64
		Storage	16	32	64
		$\beta=m/N$	3	3/2	3/4
6	m-SAPQ	Codebook (N)	2	4	8
	$k=1,\eta=3$	SNR (dB)	11.78	16.82	19.70
		Complexity	16	32	64
		Storage	96	192	384
		$\beta=m/N$	3	3/2	3/4
8	1-SAPQ	Codebook (N)	2	4	8
	$k=1,\eta=4$	SNR (dB)	11.55	16.85	22.29
		Complexity	32	64	128
		Storage	32	64	128
		$\beta=m/N$	4	2	1
8	m-SAPQ	Codebook (N)	2	4	8
	$k=1,\eta=4$	SNR (dB)	12.46	17.43	22.27
		Complexity	32	64	128
		Storage	256	512	1024
		$\beta=m/N$	4	2	1

Table 3.7: SDR (dB) performances comparing the (k,N) VQ, the (k,m,N,η) 1-SAPQ, and the (k,m,N,η) m-SAPQ, at rates R, designed using 200,000 Gauss-Markov training samples.

km	Quantizer		$R = 1.5$	$R = 2.5$	$R = 3.5$
2	VQ	Codebook (N)	8	32	128
		SNR (dB)	6.96	12.42	18.07
2	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 1$	SNR (dB)	6.87	12.42	17.90
2	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 1$	SNR (dB)	6.87	12.27	17.90
4	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 2$	SNR (dB)	6.83	12.27	18.22
4	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 2$	SNR (dB)	7.07	12.58	18.40
6	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 3$	SNR (dB)	6.59	12.77	18.62
6	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 3$	SNR (dB)	7.31	12.83	18.65
8	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 4$	SNR (dB)	6.30	12.82	18.81
8	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 4$	SNR (dB)	7.45	12.98	18.84

Table 3.8: SDR (dB) performances testing designs of the (k,N) VQ, the (k,m,N,η) 1-SAPQ, and the (k,m,N,η) m-SAPQ, at rates R using 200,000 memoryless Gaussian testing samples.

km	Quantizer		$R = 1.5$	$R = 2.5$	$R = 3.5$
2	VQ	Codebook (N)	8	32	128
		SNR (dB)	10.82	16.24	21.84
2	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 1$	SNR (dB)	8.04	13.74	17.70
2	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 1$	SNR (dB)	8.04	13.74	17.70
4	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 2$	SNR (dB)	10.94	16.01	18.70
4	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 2$	SNR (dB)	11.04	16.18	19.13
6	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 3$	SNR (dB)	11.61	16.60	18.62
6	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 3$	SNR (dB)	11.80	16.81	19.62
8	1-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 4$	SNR (dB)	11.57	16.86	22.26
8	m-SAPQ	Codebook (N)	2	4	8
	$k = 1, \eta = 4$	SNR (dB)	12.45	17.32	22.05

Table 3.9: SDR (dB) performances testing designs of the (k,N) VQ, the (k,m,N,η) 1-SAPQ, and the (k,m,N,η) m-SAPQ, at rates R, using 200,000 Gauss-Markov testing samples.

Chapter 4

Channel Optimized Sample Adaptive Product Quantizer

In this chapter the concepts of a sample adaptive product quantizer are extended to a channel optimized sample adaptive product quantizer (COSAPQ), where the channel statistics are included in the design of the quantizer. Both the m-SAPQ and 1-SAPQ are extended to a COm-SAPQ and CO1-SAPQ, respectively. In this chapter the channel considered is the binary symmetric channel (BSC).

4.1 COm-SAPQ Model

Codebook: A (k,m,N,η) COm-SAPQ is constructed from a set of 2^η codebooks $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$. Each codebook $\mathbf{C}_j$ is a product of m codebooks $\{\mathrm{C}_{s,j}\}_{s=1}^{m}$ that are subsets of $\mathbb{R}^k$, of cardinality N. In other words, each codebook $\mathrm{C}_{s,j}$ contains N codevectors

$\underline{c}_i^{[s,j]}$, where $i = 1, \ldots, N$, that belong to $\mathbb{R}^k$:

$$\mathbf{C}_j = \mathrm{C}_{1,j} \times \ldots \times \mathrm{C}_{m,j} \text{ such that } \mathrm{C}_{s,j} = \{\underline{c}_i^{[s,j]}\}_{i=1}^{N} \text{ and } \underline{c}_i^{[s,j]} \in \mathbb{R}^k. \tag{4.1}$$

For $\underline{I} \in \mathrm{J}_N^m$ and $j \in \mathrm{J}_{2^\eta}$, define $\underline{\mathbf{c}}_{\underline{I}}^{[j]}$ to be a vector of codevectors $\underline{c}_{i_s}^{[s,j]}$ that are ordered as follows

$$\underline{\mathbf{c}}_{\underline{I}}^{[j]} = (\underline{c}_{i_1}^{[1,j]}, \ldots, \underline{c}_{i_m}^{[m,j]}) \text{ where } \underline{I} = (i_1, \ldots, i_m),\ i_s \in \mathrm{J}_N = \{1, \ldots, N\} \text{ for } s = 1, \ldots, m.$$

Note that $\underline{\mathbf{c}}_{\underline{I}}^{[j]} \in \mathbf{C}_j$ for $j = 1, \ldots, 2^\eta$, and $\underline{\mathbf{c}}_{\underline{I}}^{[j]}$ is referred to as a product codevector.

Structure: Figure 4.1 depicts how a source vector $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$, where $\underline{\mathrm{x}}_s \in \mathbb{R}^k$ for $s = 1, \ldots, m$, is quantized by a (k,m,N,η) COm-SAPQ. The COm-SAPQ encoder encodes the source sample $\underline{\mathbf{x}}$ into an index vector $\underline{I} \in \mathrm{J}_N^m$ and an overhead index $j^* \in \mathrm{J}_{2^\eta}$. The index vector $\underline{I}$ and index j^* are then transmitted over a noisy channel. At the decoder, the received index vector $\underline{L} \in \mathrm{J}_N^m$ and index $j' \in \mathrm{J}_{2^\eta}$ are decoded into a product codevector $\underline{\mathbf{c}}_{\underline{L}}^{[j']}$.

Encoder: At the COm-SAPQ encoder the encoding of a source vector $\underline{\mathbf{x}}$ is processed by a set of 2^η vector functions called product encoders (PE), $\{\mathrm{PE}_j\}_{j=1}^{2^\eta}$. Each vector function PE_j takes in a copy of the source vector $\underline{\mathbf{x}}$ and encodes it using the codebook $\mathbf{C}_j$. Furthermore each PE_j has m component encoding functions, $\{\mathrm{E}_{s,j}\}_{s=1}^{m}$, such that each function $\mathrm{E}_{s,j}$ encodes subvector $\underline{\mathrm{x}}_s$ into an index $i_{s,j}$, for $s = 1, \ldots, m$. The concatenation of all the indexes $i_{s,j}$ for $s = 1, \ldots, m$, forms the index vector $\underline{I}_j$ which is the output of PE_j

$$\mathrm{PE}_j(\underline{\mathbf{x}}) = (\mathrm{E}_{1,j}(\underline{x}_1), \ldots, \mathrm{E}_{m,j}(\underline{x}_m)) = \underline{I}_j \quad \text{where} \quad \mathrm{E}_{s,j}(\underline{x}_s) = i_{s,j} \in \mathrm{J}_N$$
$$\text{and} \quad \underline{I}_j = (i_{1,j}, \ldots, i_{m,j}) \in \mathrm{J}_N^m.$$

Hence the COm-SAPQ encoder internally produces 2^η index vectors $\{\mathrm{PE}_j(\underline{\mathbf{x}}) = \underline{I}_j\}_{j=1}^{2^\eta}$. However only one index vector $\underline{I} \in \{\underline{I}_j\}_{j=1}^{2^\eta}$, is chosen to be transmitted over the channel along with the index $j^* \in \mathrm{J}_{2^\eta}$ representing PE_{j^*} (the PE that encoded $\underline{\mathbf{x}}$ into $\underline{I}$). Details of the encoding process and the choosing of index vector $\underline{I}$ and overhead index j^* are described in Section 4.2.2.

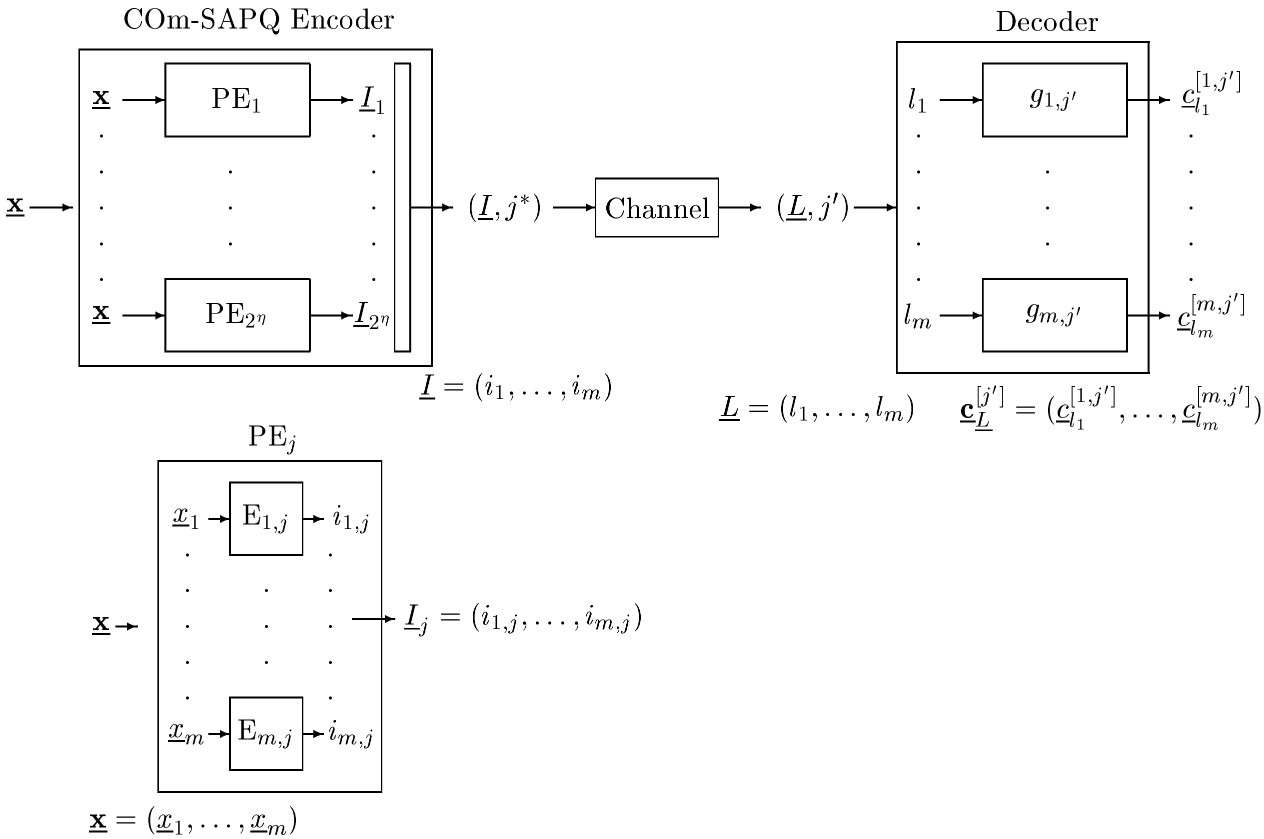

Figure 4.1: Figure of a (k,m,N,η) COm-SAPQ and the j^{th} Product Encoder PE_j where, $j = 1, \ldots, 2^\eta$, $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m) \in \mathbb{R}^{km}$, $\underline{I} = (i_1, \ldots, i_m) \in \mathrm{J}_N^m$, $\underline{L} = (l_1, \ldots, l_m) \in \mathrm{J}_N^m$, and $j^* \in \mathrm{J}_{2^\eta}$.

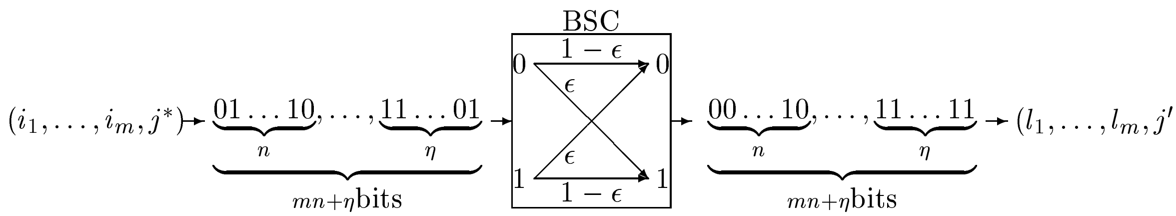

Figure 4.2: Figure of the transmission of binary codewords over the BSC.

Channel: Transmission of the index vector $\underline{I} = (i_1, \ldots, i_m)$ and index j^* over a noisy channel is realized by converting the indexes $i_1, \ldots, i_m$, of index vector $\underline{I}$, and j^* into binary codewords and transmitting each binary codeword one at a time. Thus the channel is used independently by each transmitted index $i_1, \ldots, i_m$ and j^*. Hence the probability of receiving $\underline{L} = (l_1, \ldots, l_m)$ and j' given that $\underline{I}$ and j^* were transmitted, formulates to

$$\mathrm{P}(\underline{L}, j'|\underline{I}, j^*) = \mathrm{P}(j'|j^*)\mathrm{P}(\underline{L}|\underline{I}) = \mathrm{P}(j'|j^*)\prod_{s=1}^{m}\mathrm{P}(l_s|i_s). \tag{4.2}$$

In this chapter the natural binary codeword (NBC) assignment is used. So each of the indexes $i_1, \ldots, i_m$, of index vector $\underline{I}$, are encoded into their n-bit binary codeword equivalent, where $i_s \in \mathrm{J}_N$ and $n = \log_2 N$, and j^* is encoded into its η-bit binary codeword equivalent. These codewords are then transmitted over a BSC with cross over probability ϵ as in Figure 4.2 such that the channel transition probabilities resolve to

$$\begin{aligned} \mathrm{P}(l_s|i_s) &= (1-\epsilon)^{n-d_H(l_s,i_s)}(\epsilon)^{d_H(l_s,i_s)} \quad \text{for } s = 1, \ldots, m & (4.3)\\ \mathrm{P}(j'|j^*) &= (1-\epsilon)^{\eta-d_H(j',j^*)}(\epsilon)^{d_H(j',j^*)} & (4.4)\end{aligned}$$

where $d_H(l_s, i_s)$ is the Hamming distance between the n-bit binary codewords of i_s

and l_s, and similarly $d_H(j', j^*)$ is the Hamming distance between the η-bit binary codewords of j^* and j'.

Decoder: The decoder consists of 2^η vector decoding functions $\{G_j\}_{j=1}^{2^\eta}$. Each vector decoding function G_j consists of m component decoding functions, $\{g_{s,j}\}_{s=1}^{m}$. Each decoding function $g_{s,j}$ decodes an index i into the codevector $\underline{c}_i^{[s,j]}$ ($\underline{c}_i^{[s,j]} \in \mathrm{C}_{s,j}$). At the decoder the choice of which vector decoding function G_j to use, out of the set $\{G_j\}_{j=1}^{2^\eta}$, is determined by the received overhead index j'. When index vector $\underline{L}$ and index j' are received, the decoder decodes index vector $\underline{L}$ using the vector decoding function $G_{j'}$

$$\text{Decoder}(\underline{L}, j') = G_{j'}(\underline{L}) = (g_{1,j'}(l_1), \ldots, g_{m,j'}(l_m)) = (\underline{c}_{l_1}^{[1,j']}, \ldots, \underline{c}_{l_m}^{[m,j']}) = \underline{\mathbf{c}}_{\underline{L}}^{[j']}.$$

Naturally the decoding functions invert the encoding functions

$$\begin{aligned} \mathrm{PE}_j : \mathbb{R}^{km} \to \mathrm{J}_N^m \quad &\text{and} \quad G_j : \mathrm{J}_N^m \to \mathbf{C}_j \subset \mathbb{R}^{km} \\ \mathrm{E}_{s,j} : \mathbb{R}^{k} \to \mathrm{J}_N \quad &\text{and} \quad g_{s,j} : \mathrm{J}_N \to \mathrm{C}_{s,j} \subset \mathbb{R}^{k} \end{aligned}$$

for $j = 1, \ldots, 2^\eta$ and $s = 1, \ldots, m$.

Rate: The source input vector $\underline{\mathbf{x}} = (x_1, \ldots, x_m)$, where $x_s \in \mathbb{R}^k$ for $s = 1, \ldots, m$, has a total of km source samples. The COm-SAPQ encoder output for source vector $\underline{\mathbf{x}}$ is $\underline{I} = (i_1, \ldots, i_m)$ and j^*. So each $i_s \in \mathrm{J}_N$, for $s = 1, \ldots, m$, can be represented by a $\log_2 N$-bit codeword and $j^* \in \mathrm{J}_{2^\eta}$ can be represented by a η-bit codeword. In total there are $m \log_2 N + \eta$ bits needed to represent $\underline{\mathbf{x}}$. Hence the rate of a COm-SAPQ is

$$R = \frac{\log_2 N}{k} + \frac{\eta}{km} \quad \text{bits/source sample.} \tag{4.5}$$

4.2 COm-SAPQ Necessary Conditions for Optimality

4.2.1 COm-SAPQ Distortion

To find necessary conditions for optimality of a (k,m,N,η) COm-SAPQ, the expected mean square distortion of the (k,m,N,η) COm-SAPQ needs to be calculated. To simplify notation we reuse (3.3) and (3.4).

Furthermore let $\mathbf{S}_{\underline{Z}}^{[j]}$ be the encoding region for index vector $\underline{Z}$ and index j of a (k,m,N,η) COm-SAPQ, i.e

$$\mathbf{S}_{\underline{Z}}^{[j]} = \{\underline{\mathbf{x}} \in \mathbb{R}^{km} : \text{COm-SAPQ Encoder}(\underline{\mathbf{x}}) = (\underline{Z}, j)\} \tag{4.6}$$

where $\underline{Z} \in \mathrm{J}_N^m$, and $j \in \mathrm{J}_{2^\eta}$. In total there are $2^\eta N^m$ encoding regions $\mathbf{S}_{\underline{Z}}^{[j]}$.

Let $\underline{\mathbf{c}}$ represent the reproduction, or the output of the decoder, of the COm-SAPQ for source $\underline{\mathbf{X}}$ with a probability density function $\mathrm{p}(\underline{\mathbf{x}})$. Given the set $\mathbf{S}_{\underline{Z}}^{[j]}$ of $2^\eta N^m$ encoding regions of a COm-SAPQ, and the codebooks $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$, the expected mean square distortion of a (k,m,N,η) COm-SAPQ can be found to be

$$\begin{aligned}
\mathrm{D}_{\scriptscriptstyle COm\text{-}SAPQ} &= E\{d(\underline{\mathbf{X}}, \underline{\mathbf{c}})\} \\
&= \sum_{j=1}^{2^\eta} \sum_{\underline{Z}\in \mathrm{J}_N^m} \sum_{j'=1}^{2^\eta} \sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(j'|j)\mathrm{P}(\underline{L}|\underline{Z}) E\{d(\underline{\mathbf{X}}, \underline{\mathbf{c}}_{\underline{L}}^{[j']})|\underline{\mathbf{X}} \in \mathbf{S}_{\underline{Z}}^{[j]}\}\mathrm{P}(\underline{\mathbf{X}} \in \mathbf{S}_{\underline{Z}}^{[j]}) \\
&= \sum_{j=1}^{2^\eta} \sum_{\underline{Z}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{Z}}^{[j]}} \sum_{j'=1}^{2^\eta} \sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(j'|j)\mathrm{P}(\underline{L}|\underline{Z}) \sum_{t=1}^{m} \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}.
\end{aligned}$$

The above can be simplified using

$$\sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(\underline{L}|\underline{Z}) \sum_{t=1}^{m} \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2 = \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2.$$

from Appendix A, to be

$$\mathrm{D}_{COm\text{-}SAPQ} = \sum_{j=1}^{2^\eta} \sum_{\underline{Z}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{Z}}^{[j]}} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}^{[t,j']}_{\mathrm{v}_t(\underline{L})} \right\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}.$$

4.2.2 COm-SAPQ Optimal Encoding

We next consider the following: given product codebooks $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$, how can we determine the optimal encoding function so as to minimize the mean square distortion when a source sample $\underline{\mathbf{x}}$ is encoded into an index vector $\underline{I}$ and index j^*, using a (k,m,N,η) COm-SAPQ encoder?

Let $\underline{\mathbf{c}}$ be the reproduction, or the output of the decoder, of a source vector $\underline{\mathbf{x}}$ and let $\underline{\mathbf{x}} \in \mathbf{S}_{\underline{Z}}^{[j]}$, then

$$\begin{aligned} E\{d(\underline{\mathbf{x}},\underline{\mathbf{c}})\} &= \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}^{[t,j']}_{\mathrm{v}_t(\underline{L})} \right\|^2 \\ &\geq \min_{\underline{Z}\in \mathrm{J}_N^m} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}^{[t,j']}_{\mathrm{v}_t(\underline{L})} \right\|^2 \\ &\geq \min_{j} \min_{\underline{Z}\in \mathrm{J}_N^m} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_t(\underline{L})} \right\|^2 . \end{aligned}$$

Hence there are two optimizations to be implemented. As a consequence of the structure of a COm-SAPQ, the first optimization is done by each product encoder PE_j

$$\mathrm{PE}_j(\underline{\mathbf{x}}) = \arg \min_{\underline{Z}\in \mathrm{J}_N^m} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}^{[t,j']}_{\mathrm{v}_t(\underline{L})} \right\|^2 = \underline{I}_j$$

for $j = 1, \ldots, 2^\eta$ and each constituent encoder $\mathrm{E}_{s,j}$ of PE_j produces the s^{th} index component of $\underline{I}_j$

$$\mathrm{E}_{s,j}(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg\min_{z_s \in \mathrm{J}_N} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{\mathrm{v}_s(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_s(\underline{L})|z_s) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2 = \mathrm{v}_s(\underline{I}_j) \tag{4.7}$$

independent of each other, for $s = 1, \ldots, m$ and where $\underline{Z} = (z_1, \ldots, z_m)$. The mean squared distortion incurred by each product encoder PE_j that is associated to $\underline{I}_j$ is

$$\mathrm{D}_j(\underline{\mathbf{x}}) = \min_{\underline{Z} \in \mathrm{J}_N^m} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}^{[t,j']}_{\mathrm{v}_t(\underline{L})} \right\|^2. \tag{4.8}$$

Within the encoder we have 2^η index vectors $\{\mathrm{PE}_j(\underline{\mathbf{x}}) = \underline{I}_j\}_{j=1}^{2^\eta}$ each with an associated distortion of $\{(\underline{I}_j, \mathrm{D}_j(\underline{\mathbf{x}}))\}_{j=1}^{2^\eta}$, for $j = 1, \ldots, 2^\eta$. Out of the 2^η index vectors $\{\underline{I}_j\}_{j=1}^{2^\eta}$, the index vector $\underline{I}$, $\underline{I} \in \{\underline{I}_j\}_{j=1}^{2^\eta}$, with the minimum distortion $\mathrm{D}_{j^*}(\underline{\mathbf{x}}) = \min_j \mathrm{D}_j$, is chosen to be transmitted over the channel. The PE that produced the index vector $\underline{I}$ is distinguished by transmitting the overhead index vector j^* over the channel along with $\underline{I}$. Thus the optimal encoding regions are given by

$$\mathbf{S}^{[j^*]}_{\underline{I}} = \left\{ \underline{\mathbf{x}} \in \mathbb{R}^{km} : j^* = \arg\min_{j \in \mathrm{J}_{2^\eta}} D_j(\underline{\mathbf{x}}) \text{ and } \underline{I} = \underline{I}_{j^*} = \mathrm{PE}_{j^*}(\underline{\mathbf{x}}) \right\}. \tag{4.9}$$

With $\underline{I}$ and j^* defined above, the optimal distortion given the codebooks $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$ is

$$\begin{aligned} \mathrm{D}_{\mathit{COm\text{-}SAPQ}} &= E\{\min_j \mathrm{D}_j(\underline{\mathbf{X}})\} \\ &= \sum_{j^*=1}^{2^\eta} \sum_{\underline{I} \in \mathrm{J}_N^m} \int_{\mathbf{S}^{[j^*]}_{\underline{I}}} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j^*) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{I})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}^{[t,j']}_{\mathrm{v}_t(\underline{L})} \right\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}. \end{aligned}$$

4.2.3 COm-SAPQ Optimal Decoding

In the previous sections we assumed that the set of codebooks $\{\mathbf{C}_j\}_{j=1}^{2^\eta}$ were given, in other words the set of all codevectors $\{\underline{c}_i^{[s,j]}\}$ were assumed to be given. We will now find the set of all optimal codevectors $\{\underline{c}_i^{[s,j]}\}$ assuming that the set of $2^\eta N^m$ optimal encoding regions $\{\mathbf{S}_{\underline{I}}^{[j^*]}\}$ are given. The distortion with the encoding regions $\{\mathbf{S}_{\underline{I}}^{[j^*]}\}$ defined as in (4.9) is

$$\mathrm{D}_{COm\text{-}SAPQ} = \sum_{j^*=1}^{2^\eta} \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j^*]}} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j^*) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{I})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2 \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}.$$

Take the partial derivative with respect to the codevector $\underline{c}_l^{[s,j]}$ by filtering out the l^{th} codevector of codebook $\mathrm{C}_{s,j}$, where $s \in \mathrm{J}_m$, $j \in \mathrm{J}_{2^\eta}$ and $l \in \mathrm{J}_N$, in the above distortion as follows

$$\begin{aligned}
\mathrm{D}_{COm\text{-}SAPQ} &= \sum_{j^*=1}^{2^\eta} \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j^*]}} \Bigg\{ \mathrm{P}(j|j^*) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{I})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j]} \right\|^2 + \\
&\qquad \sum_{j':j'\neq j} \mathrm{P}(j'|j^*) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{I})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2 \Bigg\} \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \\
&= \sum_{j^*=1}^{2^\eta} \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j^*]}} \Bigg\{ \mathrm{P}(j|j^*) \sum_{\mathrm{v}_s(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_s(\underline{L})|\mathrm{v}_s(\underline{I})) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_s(\underline{L})}^{[s,j]} \right\|^2 + \\
&\qquad \mathrm{P}(j|j^*) \sum_{t:t\neq s} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{I})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j]} \right\|^2 + \\
&\qquad \sum_{j':j'\neq j} \mathrm{P}(j'|j^*) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{I})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2 \Bigg\} \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}.
\end{aligned}$$

Setting the resultant derivative to zero we get

$$\begin{aligned}
0 &= \sum_{j*=1}^{2^\eta} \sum_{\underline{I}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{I}}^{[j^*]}} \mathrm{P}(j|j^*)\mathrm{P}(\mathrm{v}_s(\underline{L}) = l|\mathrm{v}_s(\underline{I})) \left\{ -\mathrm{u}_s(\underline{\mathbf{x}}) + \underline{c}_l^{[s,j]} \right\} \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \\
&= \sum_{j^*=1}^{2^\eta} \sum_{i_1=1}^{N} \cdots \sum_{i_m=1}^{N} \int_{\mathbf{S}_{(i_1,\ldots,i_m)}^{[j^*]}} \mathrm{P}(j|j^*)\mathrm{P}(l|i_s) \left\{ -\mathrm{u}_s(\underline{\mathbf{x}}) + \underline{c}_l^{[s,j]} \right\} \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}} \\
&= \sum_{j^*=1}^{2^\eta} \sum_{i=1}^{N} \int_{S_i^{[s,j^*]}} \mathrm{P}(j|j^*)\mathrm{P}(l|i_s = i) \left\{ -\mathrm{u}_s(\underline{\mathbf{x}}) + \underline{c}_l^{[s,j]} \right\} \mathrm{p}(\underline{\mathbf{x}}) d\underline{\mathbf{x}}
\end{aligned}$$

where $\underline{I} = (i_1, \ldots, i_m)$ and

$$S_i^{[s,j^*]} = \bigcup_{\underline{I}\ :\ \mathrm{v}_s(\underline{I})=i} \mathbf{S}_{\underline{I}}^{[j^*]} \tag{4.11}$$

$$= \{\underline{\mathbf{x}} \in \mathbb{R}^{km} : \text{COm-SAPQ Encoder}(\underline{\mathbf{x}}) = ((i_1, \ldots, i_m), j^*) \text{ and } i_s = i\}. \tag{4.12}$$

Solving for $\underline{c}_l^{[s,j]}$ we get

$$\underline{c}_l^{[s,j]} = \frac{\sum_{j^*=1}^{2^\eta} \sum_{i=1}^{N} \mathrm{P}(j|j^*)\mathrm{P}(l|i) \int_{S_i^{[s,j^*]}} \mathrm{u}_s(\underline{\mathbf{x}})\mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}}}{\sum_{j^*=1}^{2^\eta} \sum_{i=1}^{N} \mathrm{P}(j|j^*)\mathrm{P}(l|i) \int_{S_i^{[s,j^*]}} \mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}}}. \tag{4.13}$$

4.3 CO1-SAPQ

Just as the 1-SAPQ, the CO1-SAPQ is a particular case of the COm-SAPQ. A (k,m,N,η) CO1-SAPQ only requires 2^η codebooks $\{\mathrm{C}_j\}_{j=1}^{2^\eta}$, where

$$\mathrm{C}_j = \{\underline{c}_i^{[j]}\}_{i=1}^{N} \text{ and } \underline{c}_i^{[s,j]} \in \mathbb{R}^k. \tag{4.14}$$

Figure 4.3 depicts how a source vector $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m)$, where $\underline{\mathrm{x}}_s \in \mathbb{R}^k$ for $s = 1, \ldots, m$, is quantized by a (k,m,N,η) CO1-SAPQ. Note the the repeated encoders (RE) of a CO1-SAPQ, are still vector functions with the same constituent encoding functions, E_j for each RE_j, where $j = 1, \ldots, 2^\eta$. The difference between the repeated encoders of a CO1-SAPQ and that of Section 3.4, is the inclusion of channel statistics (4.4) and (3.3) in the RE function of the CO1-SAPQ.

Distortion: Given the codebooks $\{\mathrm{C}_j\}_{j=1}^{2^\eta}$ and encoding regions $\{\mathbf{S}_{\underline{Z}}^{[j]}\}$, the distortion of a ($k$,$m$,$N$,$\eta$) CO1-SAPQ can be derived to be

$$\begin{aligned}
\mathrm{D}_{\textit{CO1-SAPQ}} &= E\{d(\underline{\mathbf{X}}, \underline{\mathbf{c}})\} \\
&= \sum_{j=1}^{2^\eta} \sum_{\underline{Z} \in \mathrm{J}_N^m} E\{\sum_{j'=1}^{2^\eta} \sum_{\underline{L} \in \mathrm{J}_N^m} \mathrm{P}(j'|j)\mathrm{P}(\underline{L}|\underline{Z})d(\underline{\mathbf{X}}, \underline{\mathbf{c}}_{\underline{L}}^{[j']})|\underline{\mathbf{X}} \in \mathbf{S}_{\underline{Z}}^{[j]}\}\mathrm{P}(\underline{\mathbf{X}} \in \mathbf{S}_{\underline{Z}}^{[j]})
\end{aligned}$$

$$= \sum_{j=1}^{2^\eta} \sum_{\underline{Z}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{Z}}^{[j]}} \sum_{j'=1}^{2^\eta} \sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(j'|j)\mathrm{P}(\underline{L}|\underline{Z}) \sum_{t=1}^{m} \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[j']} \right\|^2 \mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}}.$$

Optimal Encoding: Given the codebooks $\{\mathrm{C}_j\}_{j=1}^{2^\eta}$, the mean squared distortion incurred by each repeated encoder RE_j is given by

$$\mathrm{D}_j(\underline{\mathbf{x}}) = \min_{\underline{Z}\in \mathrm{J}_N^m} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[j']} \right\|^2 . \tag{4.15}$$

Hence the optimal encoding regions of a (k,m,N,η) CO1-SAPQ are defined by

$$\mathbf{S}_{\underline{I}}^{[j^*]} = \left\{ \underline{\mathbf{x}} \in \mathbb{R}^{km} : j^* = \arg\min_{j\in \mathrm{J}_{2^\eta}} D_j(\underline{\mathbf{x}}) \text{ and } \underline{I} = \underline{I}_{j^*} = \mathrm{RE}_{j^*}(\underline{\mathbf{x}}) \right\}. \tag{4.16}$$

where $\underline{I} \in \mathrm{J}_N^m$, $j^* = 1, \ldots, 2^\eta$,

$$\mathrm{RE}_j(\underline{\mathbf{x}}) = \arg\min_{\underline{Z}\in \mathrm{J}_N^m} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[j']} \right\|^2 = \underline{I}_j$$

, and

$$\mathrm{E}_j(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg\min_{z_s\in \mathrm{J}_N} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{\mathrm{v}_s(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_s(\underline{L})|z_s) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_s(\underline{L})}^{[j']} \right\|^2 = \mathrm{v}_s(\underline{I}_j). \tag{4.17}$$

Optimal Decoding: Given the encoding regions $\{\mathbf{S}_{\underline{I}}^{[j^*]}\}$ defined as in (4.16), the optimal codevectors of a (k,m,N,η) CO1-SAPQ can be derived to be

$$\underline{c}_l^{[j]} = \frac{\sum_{j^*=1}^{2^\eta} \sum_{i=1}^{N} \sum_{t=1}^{m} \mathrm{P}(j|j^*)\mathrm{P}(l|i) \int_{S_i^{[t,j^*]}} \mathrm{u}_s(\underline{\mathbf{x}})\mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}}}{\sum_{j^*=1}^{2^\eta} \sum_{i=1}^{N} \sum_{t=1}^{m} \mathrm{P}(j|j^*)\mathrm{P}(l|i) \int_{S_i^{[t,j^*]}} \mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}}}. \tag{4.18}$$

where $l = 1, \ldots, N$, $j = 1, \ldots, 2^\eta$, and for $t = 1, \ldots, m$

$$S_i^{[t,j^*]} = \bigcup_{\underline{I}\,:\, \mathrm{v}_t(\underline{I})=i} \mathbf{S}_{\underline{I}}^{[j^*]} \tag{4.19}$$

$$= \{\underline{\mathbf{x}} \in \mathbb{R}^{km} : \text{CO1-SAPQ Encoder}(\underline{\mathbf{x}}) = ((i_1, \ldots, i_m), j^*) \text{ and } i_t = i\}. \tag{4.20}$$

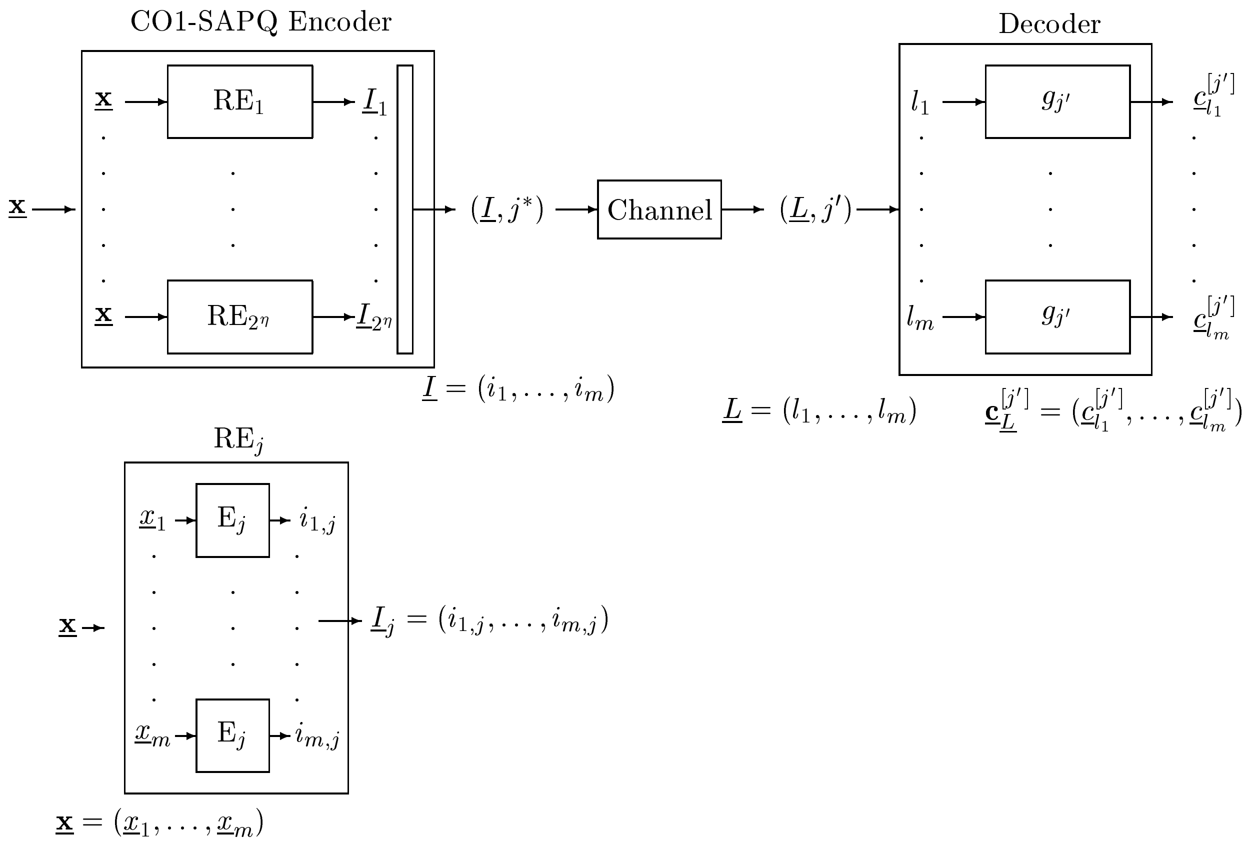

Figure 4.3: Figure of a (k,m,N,η) CO1-SAPQ and the j^{th} Repeated Encoder RE$_j$ where, $j = 1, \ldots, 2^\eta$, $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1, \ldots, \underline{\mathrm{x}}_m) \in \mathbb{R}^{km}$, $\underline{I} = (i_1, \ldots, i_m) \in \mathrm{J}_N^m$, $\underline{L} = (l_1, \ldots, l_m) \in \mathrm{J}_N^m$, and $j^* \in \mathrm{J}_{2^\eta}$.

4.4 Encoding Simplifications

Encoding complexity is measured as the total number of multiplications required to encode a scalar source sample. In processing, it is far simpler to add then to multiply and hence multiplication is taken as a reasonable measure of complexity.

4.4.1 COm-SAPQ Encoding Simplifications

Consider a (k,m,N,η) COm-SAPQ with source sample $\underline{\mathbf{x}}$, the s^{th} encoder function of j^{th} product encoder PE_j is

$$\mathrm{E}_{s,j}(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg\min_{z_s \in \mathrm{J}_N} \sum_{j'=1}^{2^\eta} \sum_{\mathrm{v}_s(\underline{L})=1}^{N} \mathrm{P}(j'|j)\mathrm{P}(\mathrm{v}_s(\underline{L})|z_s) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2 = \mathrm{v}_s(\underline{I}_j)$$

where $\mathrm{PE}_j(\underline{\mathbf{x}}) = \underline{I}_j$. The term

$$\mathrm{P}(j'|j)\mathrm{P}(\mathrm{v}_s(\underline{L})|z_s) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2$$

requires $(k+2)$ multiplications which must be carried out for $\mathrm{v}_s(\underline{L}) = 1, \ldots, N$, $j' = 1, \ldots, 2^\eta$ and $z_s \in \mathrm{J}_N$. Hence a total of $N(2^\eta N(k+2))$ multiplications are required to encode k source samples ($\mathrm{u}_s(\underline{\mathbf{x}}) \in \mathbb{R}^k$) into an index $\mathrm{v}_s(\underline{I}_j)$. So the encoding complexity of the optimal encoder function is $\frac{N(2^\eta N(k+2))}{k}$ per sample. This encoding complexity can be reduced by taking the encoder function

$$\mathrm{E}_{s,j}(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg\min_{z_s \in \mathrm{J}_N} \sum_{j'=1}^{2^\eta} \sum_{l_s=1}^{N} \mathrm{P}(j'|j)\mathrm{P}(l_s|z_s) \left\{ \|\mathrm{u}_s(\underline{\mathbf{x}})\|^2 + \|\underline{c}^{[s,j']}_{l_s}\|^2 - 2 < \mathrm{u}_s(\underline{\mathbf{x}}), \underline{c}^{[s,j']}_{l_s} > \right\}$$

where $\mathrm{v}_s(\underline{L}) = l_s$ and where $<\underline{x}, \underline{y}>$ is the inner product over $\mathbb{R}^k$ for $\underline{x}, \underline{y} \in \mathbb{R}^k$, and simplifying it using functions

$$\mathrm{y}_{s,j}(\gamma) = \sum_{j'=1}^{2^\eta} \sum_{l=1}^{N} P(j'|j)P(l|\gamma)\underline{c}^{[s,j']}_l \quad \text{and} \quad \alpha_{s,j}(\gamma) = \sum_{j'=1}^{2^\eta} \sum_{l=1}^{N} P(j'|j)P(l|\gamma)\|\underline{c}^{[s,j']}_l\|^2 \tag{4.21}$$

for $\gamma \in \mathrm{J}_N$, so that

$$\mathrm{E}_{s,j}(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg\min_{z_s \in \mathrm{J}_N} \{\alpha_{s,j}(z_s) - 2 < \mathrm{u}_s(\underline{\mathbf{x}}), \mathrm{y}_{s,j}(z_s) > \} = \mathrm{v}_s(\underline{I}_j). \tag{4.22}$$

Now given $\{\alpha_{s,j}(\gamma)\}_{\gamma=1}^N$ and $\{\mathrm{y}_{s,j}(\gamma)\}_{\gamma=1}^N$, the function

$$\{\alpha_{s,j}(z_s) - 2 < \mathrm{u}_s(\underline{\mathbf{x}}), \mathrm{y}_{s,j}(z_s) > \}$$

requires only k multiplications per $z_s \in \mathrm{J}_N$. So a total of Nk multiplications are required to encode k source samples $\mathrm{u}_s(\underline{\mathbf{x}})$ into an index $\mathrm{v}_s(\underline{I}_j)$ using (4.22). This brings the encoding complexity of the encoding function $\mathrm{E}_{s,j}$ down to $\frac{Nk}{k}$ per sample. Since for $j = 1, \ldots, 2^\eta$,

$$\mathrm{PE}_j(\underline{\mathbf{x}}) = (\mathrm{E}_{1,j}(\mathrm{u}_1(\underline{\mathbf{x}})), \ldots, \mathrm{E}_{m,j}(\mathrm{u}_m(\underline{\mathbf{x}}))) = \underline{I}_j$$

the encoding complexity of the (k,m,N,η) COm-SAPQ is reduced as a whole. The product encoder function PE_j, for $j = 1, \ldots, 2^\eta$, reduces to

$$\begin{aligned} \mathrm{PE}_j(\underline{\mathbf{x}}) &= \arg \min_{\underline{Z} \in \mathrm{J}_N^m} \sum_{t=1}^{m} \sum_{j'=1}^{2^\eta} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(j'|j)\mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2 \quad (4.23) \\ &= \arg \min_{\underline{Z} \in \mathrm{J}_N^m} \sum_{t=1}^{m} \left\{ \alpha_{t,j}(\mathrm{v}_t(\underline{Z})) - 2<\mathrm{u}_t(\underline{\mathbf{x}}), \mathrm{y}_{t,j}(\mathrm{v}_t(\underline{Z}))> \right\} = \underline{I}_j. \quad (4.24) \end{aligned}$$

When encoding a source sample $\underline{\mathbf{x}}$, within the (k,m,N,η) COm-SAPQ encoder there are 2^η index vectors $\{\underline{I}_j\}_{j=1}^{2^\eta}$ produced by the 2^η product encoders, $\{\mathrm{PE}_j(\underline{\mathbf{x}}) = \underline{I}_j\}_{j=1}^{2^\eta}$ each with an associated distortion, $\{(\underline{I}_j, \mathrm{D}_j(\underline{\mathbf{x}}))\}_{j=1}^{2^\eta}$ where

$$\mathrm{D}_j(\underline{\mathbf{x}}) = \min_{\underline{Z} \in \mathrm{J}_N^m} \sum_{t=1}^{m} \sum_{j'=1}^{2^\eta} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(j'|j)\mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2 .$$

The index $j^* \in \mathrm{J}_{2^\eta}$ representing the index vector $\underline{I} = \underline{I}_{j^*}$ with the minimum distortion is then chosen by

$$j^* = \arg \min_{j \in \mathrm{J}_{2^\eta}} \mathrm{D}_j(\underline{\mathbf{x}}) = \arg \min_{j \in \mathrm{J}_{2^\eta}} \left\{ \min_{\underline{Z} \in \mathrm{J}_N^m} \sum_{t=1}^{m} \left\{ \alpha_{t,j}(\mathrm{v}_t(\underline{Z})) - 2<\mathrm{u}_t(\underline{\mathbf{x}}), \mathrm{y}_{t,j}(\mathrm{v}_t(\underline{Z}))> \right\} \right\}. \quad (4.25)$$

4.4.2 CO1-SAPQ Encoding Simplifications

Consider a (k,m,N,η) CO1-SAPQ with source sample $\underline{\mathbf{x}}$, the constituent encoder function of repeated encoder RE_j is

$$\mathrm{E}_j(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg\min_{z_s \in \mathrm{J}_N} \sum_{j'=1}^{2^\eta} \sum_{l_s=1}^{N} \mathrm{P}(j'|j)\mathrm{P}(l_s|z_s) \left\{ \|\mathrm{u}_s(\underline{\mathbf{x}})\|^2 - 2{<}\mathrm{u}_s(\underline{\mathbf{x}}), \underline{c}_{l_s}^{[s,j']}{>} + \|\underline{c}_{l_s}^{[j']}\|^2 \right\}$$

where $\mathrm{v}_s(\underline{L}) = l_s$ and where ${<}\underline{x}, \underline{y}{>}$ is the inner product over $\mathbb{R}^k$ for $\underline{x}, \underline{y} \in \mathbb{R}^k$, and simplifying it using functions

$$\mathrm{y}_j(\gamma) = \sum_{j'=1}^{2^\eta} \sum_{l=1}^{N} \mathrm{P}(j'|j)\mathrm{P}(l|\gamma)\underline{c}_l^{[j']} \quad \text{and} \quad \alpha_j(\gamma) = \sum_{j'=1}^{2^\eta} \sum_{l=1}^{N} \mathrm{P}(j'|j)\mathrm{P}(l|\gamma)\|\underline{c}_l^{[j']}\|^2 \quad (4.26)$$

for $\gamma \in \mathrm{J}_N$, so that

$$\mathrm{E}_j(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg\min_{z_s \in \mathrm{J}_N} \{\alpha_j(z_s) - 2{<}\mathrm{u}_s(\underline{\mathbf{x}}), \mathrm{y}_j(z_s){>}\} = \mathrm{v}_s(\underline{I}_j). \quad (4.27)$$

Now given $\{\alpha_j(\gamma)\}_{\gamma=1}^{N}$ and $\{\mathrm{y}_j(\gamma)\}_{\gamma=1}^{N}$, the function

$$\{\alpha_j(z_s) - 2{<}\mathrm{u}_s(\underline{\mathbf{x}}), \mathrm{y}_j(z_s){>}\}$$

requires only k multiplications per $z_s \in \mathrm{J}_N$. So a total of Nk multiplications are required to encode k source samples $\mathrm{u}_s(\underline{\mathbf{x}})$ into an index $\mathrm{v}_s(\underline{I}_j)$ using (4.27). This brings the encoding complexity of the encoding function E_j down to $\frac{Nk}{k}$ per sample.

4.5 Encoding Complexity and Storage Requirements

Let us now compare the encoding complexity and storage requirements for each of the following channel optimized quantizers: COVQ, COPQ, COm-SAPQ, and CO1-SAPQ. We define the encoding complexity to be the total amount of multiplications required to encode a source sample (complexity=total multiplications required for encoding/source sample). The storage requirements are measured as the total number of scalar values that are required to be stored at the encoder and decoder, in order to implement the quantizer in question (storage=total scalars required for implementation). The results are summarized in Table 4.1.

4.5.1 (k_1,N_1) COVQ

Encoding Complexity: From Section 2.2.3 a (k_1,N_1) COVQ encoder, encodes a source sample $\underline{\mathrm{x}} \in \mathbb{R}^{k_1}$, using the following function

$$\mathrm{E}(\underline{\mathrm{x}}) = \arg\min_{l \in \mathrm{J}_{N_1}} \{\alpha(l) - 2<\underline{\mathrm{x}}, \mathrm{y}(l)>\} \tag{4.28}$$

where $\mathrm{J}_{N_1} = \{1, \ldots, N_1\}$, and functions y() and $\alpha()$ are defined as in (2.14). This operation (4.28) requires k_1 multiplications done N_1 times to encode $\underline{\mathrm{x}} \in \mathbb{R}^{k_1}$ into an index in J_{N_1}. Hence a total of $k_1 N_1$ multiplications are required for source $\underline{\mathrm{x}} \in \mathbb{R}^{k_1}$, which brings the complexity to

$$(k_1,N_1) \text{ COVQ Complexity} = \frac{k_1 N_1}{k_1} = N_1.$$

This is the same as a the complexity of a (k_1,N_1) VQ.

Storage Requirements: In order to implement (4.28) the sets $\{\mathrm{y}(l)\}_{l=1}^{N_1}$ and $\{\alpha(l)\}_{l=1}^{N_1}$, need to be pre-calculated, and stored at the encoder. Calculating the vectors of $\mathrm{y}(l)$ for $l = 1, \ldots, N_1$ gives us N_1 vectors, each of dimension k_1 using (2.14). Calculating the scalars $\alpha(l)$ for $l = 1, \ldots, N_1$ gives us N_1 scalars to be stored. Hence from both sets $\{\mathrm{y}(l)\}_{l=1}^{N_1}$ and $\{\alpha(l)\}_{l=1}^{N_1}$ we have a total of $k_1 N_1 + N_1$ scalars to store at the encoder. Then at the decoder we require $k_1 N_1$ scalars to be stored for the codebook C, (2.9). This brings the storage requirements of the (k_1,N_1) COVQ to be

$$(k_1,N_1)\text{ COVQ Storage} = 2k_1N_1 + N_1.$$

4.5.2 (k_2,m_2,N_2) COPQ

Encoding Complexity: From Section 2.4.3, the product encoder (PE) of a (k_2,m_2,N_2) COPQ is a vector function of m_2 encoder functions that perform the following operation

$$\mathrm{E}_t(\underline{\mathrm{x}}) = \arg\min_{z\in \mathrm{J}_{N_2}} \{\alpha_t(z) - 2<\underline{\mathrm{x}}, \mathrm{y}_t(z)>\} \tag{4.29}$$

where $\mathrm{J}_{N_2} = \{1,\ldots,N_2\}$, $\underline{\mathrm{x}} \in \mathbb{R}^{k_2}$, $t = 1,\ldots,m_2$, and functions $\mathrm{y}_t()$ and $\alpha_t()$ are defined as in (2.30). This operation (4.29) requires k_2N_2 multiplications for each source subvector $\underline{\mathrm{x}} \in \mathbb{R}^{k_2}$. There are m_2 such encoders $\mathrm{E}_t(\underline{\mathrm{x}})$, each encoding a source subvector $\underline{\mathrm{x}}_t$ from vector $\underline{\mathbf{x}} = (\underline{\mathrm{x}}_1,\ldots,\underline{\mathrm{x}}_{m_2}) \in \mathbb{R}^{k_2m_2}$. Hence a total of $m_2k_2N_2$ multiplications are required to encode m_2 source vectors of dimension k_2, bringing the complexity to

$$(k_2,m_2,N_2)\text{ COPQ Complexity} = \frac{m_2k_2N_2}{m_2k_2} = N_2.$$

Storage Requirements: Just as in the case of the (k_1,N_1) COVQ, the sets $\{\mathrm{y}_t(z)\}_{z=1}^{N_2}$ and $\{\alpha_t(z)\}_{z=1}^{N_2}$, need to be calculated for $t = 1,\ldots,m_2$. For each t, $\mathrm{y}_t(z)$ when precalculated for $z = 1,\ldots,N_2$ produces N_2 vectors of dimension k_2. Hence there are $m_2k_2N_2$ scalars needed to store the results of the m_2 sets $\{\mathrm{y}_t(z)\}_{z=1}^{N_2}$. Similarly the function $\alpha_t(z)$ produces N_2 scalars for $z = 1,\ldots,N_2$. There are m_2 sets $\{\alpha_t(z)\}_{z=1}^{N_2}$, hence a total of m_2N_2 scalars need to be stored. All the $m_2k_2N_2$ scalars and m_2N_2 scalars for $\mathrm{y}_t()$ and $\alpha_t()$, respectively, are stored at the encoder. At the decoder we need to store an additional $k_2m_2N_2$ scalars for the codebook (2.25), bringing the total to

$$(k_2, m_2, N_2) \text{ COPQ Storage} = 2k_2 m_2 N_2 + m_2 N_2.$$

4.5.3 (k_3,m_3,N_3,η_3) COm-SAPQ

Encoding Complexity: In Section 4.4, we clearly describe the encoding complexity for the s^{th} encoder function of j^{th} product encoder PE_j

$$\mathrm{E}_{s,j}(\mathrm{u}_s(\underline{\mathbf{x}})) = \arg \min_{z_s \in \mathrm{J}_{N_3}} \{\alpha_{s,j}(z_s) - 2 < \mathrm{u}_s(\underline{\mathbf{x}}), \mathrm{y}_{s,j}(z_s) > \} \tag{4.30}$$

where now for the (k_3,m_3,N_3,η_3) COm-SAPQ $\underline{\mathbf{x}} \in \mathbb{R}^{k_3 m_3}$, $\mathrm{J}_{N_3} = \{1, \ldots, N_3\}$, $s = 1, \ldots, m_3$, $j = 1, \ldots, \eta_3$ and functions $\alpha_{s,j}()$ and $\mathrm{y}_{s,j}()$ are defined in (4.21) with a source sample $\underline{\mathbf{x}} \in \mathbb{R}^{k_3 m_3}$. We showed the encoding complexity to be $\frac{k_3 N_3}{k_3}$, given the set $\{\alpha_{s,j}(\gamma)\}_{\gamma=1}^{N_3}$ and $\{\mathrm{y}_{s,j}(\gamma)\}_{\gamma=1}^{N_3}$. In each product encoder PE_j we have m_3 such encoding functions processing source samples of dimension k_3, $\mathrm{u}_s(\underline{\mathbf{x}}) \in \mathbb{R}^{k_3}$. Each PE_j processes $k_3 m_3$ source samples $\underline{\mathbf{x}} \in \mathbb{R}^{k_3 m_3}$ using these m_3 encoding functions hence the total number of multiplications per source sample for each PE_j is $\frac{k_3 m_3 N_3}{k_3 m_3}$. In a (k_3,m_3,N_3,η_3) COm-SAPQ there a 2^{η_3} such PE_j, bringing the complexity to

$$(k_3,m_3,N_3,\eta_3) \text{ COm-SAPQ Complexity} = 2^{\eta_3} \frac{k_3 m_3 N_3}{k_3 m_3} = 2^{\eta_3} N_3.$$

Storage Requirements: As described above, in order for the complexity of equation (4.30) to be low we need to pre-calculate and store the sets $\{\alpha_{s,j}(\gamma)\}_{\gamma=1}^{N_3}$ and $\{\mathrm{y}_{s,j}(\gamma)\}_{\gamma=1}^{N_3}$ for $s = 1, \ldots, m_3$ and $j = 1, \ldots, 2^{\eta_3}$. Each set $\{\mathrm{y}_{s,j}(\gamma)\}_{\gamma=1}^{N_3}$ is a set of N_3 vectors of dimension k_3. In total we need $k_3 m_3 N_3 2^{\eta_3}$ scalars for sets $\{\mathrm{y}_{s,j}(\gamma)\}_{\gamma=1}^{N_3}$ where $s = 1, \ldots, m_3$ and $j = 1, \ldots, 2^{\eta_3}$. Similarly the sets $\{\alpha_{s,j}(\gamma)\}_{\gamma=1}^{N_3}$ contain N_3 scalars for $s = 1, \ldots, m_3$ and $j = 1, \ldots, 2^{\eta_3}$. Hence at the encoder we require

$k_3 m_3 N_3 2^{\eta_3} + m_3 N_3 2^{\eta_3}$ scalars. At the decoder we need to store the codebook (4.1) which is a total of $k_3 m_3 N_3 2^{\eta_3}$ scalars, bringing the storage requirements to

$$(k_3,m_3,N_3,\eta_3) \text{ COm-SAPQ Storage} = 2k_3 m_3 N_3 2^{\eta_3} + m_3 N_3 2^{\eta_3}.$$

So to keep the encoding complexity of a (k_3,m_3,N_3,η_3) COm-SAPQ lower then that of a (k_1,N_1) COVQ of the same rate, we need $2^{\eta_3} N_3 < N_1$. This is accomplished by carefully choosing the parameters of the (k_3,m_3,N_3,η_3) COm-SAPQ. When choosing the parameters of (k_3,m_3,N_3,η_3) COm-SAPQ, a balance must be kept between keeping the rate $R = \frac{\log_2 N_3}{k_3} + \frac{\eta_3}{k_3 m_3} = \frac{\log_2 N_1}{k_1}$ equal and trying to achieve the inequality $2^{\eta_3} N_3 < N_1$.

4.5.4 (k_4,m_4,N_4,η_4) CO1-SAPQ

Encoding Complexity: The derivation of the encoding complexity of the (k_4,m_4,N_4,η_4) CO1-SAPQ is exactly the same as that of the (k_3,m_3,N_3,η_3) COm-SAPQ, using section 4.4. It can be derived to be

$$(k_4,m_4,N_4,\eta_4) \text{ CO1-SAPQ Complexity} = 2^{\eta_4} \frac{k_4 m_4 N_4}{k_4 m_4} = 2^{\eta_4} N_4$$

which is the same encoding complexity as a (k_4,m_4,N_4,η_4) COm-SAPQ.

Storage Requirements: The advantage of the (k_4,m_4,N_4,η_4) CO1-SAPQ over the (k_3,m_3,N_3,η_3) COm-SAPQ is in the storage requirements. The encoding function (4.27) requires the pre-calculation of the sets $\{\alpha_j(\gamma)\}_{\gamma=1}^{N_4}$ and $\{\mathrm{y}_j(\gamma)\}_{\gamma=1}^{N_4}$ for $j = 1, \ldots, 2^{\eta_4}$. This is N_4 vectors of dimension k_4 for the set $\{\mathrm{y}_j(\gamma)\}_{\gamma=1}^{N_4}$ and N_4 scalars for the set $\{\alpha_j(\gamma)\}_{\gamma=1}^{N_4}$. The total at the encoder is hence $k_4 N_4 2^{\eta_4} + N_4 2^{\eta_4}$ stored scalars. At the decoder the codebook (4.14) requires only $k_4 N_4 2^{\eta_4}$ scalars, bringing

Quantizer	Complexity	Storage
(k_1,N_1) COVQ	N_1	$2k_1N_1 + N_1$
(k_2,m_2,N_2) COPQ	N_2	$2k_2m_2N_2 + m_2N_2$
(k_3,m_3,N_3,η_3) COm-SAPQ	$2^{\eta_3}N_3$	$2k_3m_3N_32^{\eta_3} + m_3N_32^{\eta_3}$
(k_4,m_4,N_4,η_4) CO1-SAPQ	$2^{\eta_4}N_4$	$2k_4N_42^{\eta_4} + N_42^{\eta_4}$

Table 4.1: Table of Encoding Complexity and Storage Requirements for Quantizers designed with noisy memoryless channels (BSC).

the storage to

$$(k_4,m_4,N_4,\eta_4)\text{ CO1-SAPQ Storage} = 2k_4N_42^{\eta_4} + N_42^{\eta_4}$$

which is $\frac{1}{m_4}$ less than the storage requirements of a (k_4,m_4,N_4,η_4) COm-SAPQ.

4.6 Design Algorithm for CO-SAPQs

The design algorithm for the COm-SAPQ is next described. This design with be based on the derivations of the necessary conditions for optimality (4.9) and (4.13). In this algorithm we iterate over the necessary conditions for optimality in the similar fashion to the Lind-Buzo-Gray (LBG) algorithm. This design is derived in order to attain a local minimum using a designed initial codebook.

(k,m,N,η) COm-SAPQ Algorithm

1. Set parameters k, m, N, η, the design BSC error crossover probability ϵ_d, the stopping threshold δ, the splitting constant k-dimensional vector $\underline{\epsilon} = (\epsilon, \ldots, \epsilon)$,

the maximum number of iterations $Maxiter$, and M the total number of training vectors $\{\underline{\mathbf{x}}_f = (\underline{x}_{1,f}, \ldots, \underline{x}_{m,f})\}_{f=1}^{M}$. Initialize $\tau = 1$, $\rho = 0$, and the initial set of codebooks $\mathbf{C}_1^{(0)} = C_{1,1}^{(0)} \times \cdots \times C_{m,1}^{(0)}$.

2. If $\tau \geq 2^\eta$ stop; otherwise split the codebooks using

$$C_{s,j}^{(\rho)} = C_{s,j}^{(\rho)} - \underline{\epsilon} \quad \text{and} \quad C_{s,j+\tau}^{(\rho)} = C_{s,j}^{(\rho)} + \underline{\epsilon}$$

for $s = 1, \ldots, m$ and $j = 1, \ldots, \tau$, then double $\tau = \tau * 2$ and set $\rho = 0$. At this point we have τ sets of codebooks $\{\mathbf{C}_j^{(\rho)}\}_{j=1}^{\tau}$.

3. For $s = 1, \ldots, m$, $j = 1, \ldots, \tau$ and $\gamma = 1, \ldots, N$ calculate the $\tau N m$ vectors $\{\underline{y}_{s,j}^{(\rho)}(\gamma)\}$ and values $\{\alpha_{s,j}^{(\rho)}(\gamma)\}$ as in (4.21), using the codebooks $\{\mathbf{C}_j^{(\rho)}\}_{j=1}^{\tau}$. For each $\underline{\mathbf{x}}_f$, encode $\underline{\mathbf{x}}_f$ with each $\{\mathrm{PE}_j^{(\rho)}\}_{j=1}^{\tau}$ as in (4.24). This will give us the set of index vectors $\{\underline{I}_j\}_{j=1}^{\tau}$. The $\mathrm{PE}_{j^*}^{(\rho)}$ that produces the index vector $\underline{I}$ with minimum distortion is chosen using (4.25).

4. Once $\underline{\mathbf{x}}_f$ is encoded, $\underline{\mathbf{x}}_f$ can be put into the appropriate cells, (4.12). So if $\underline{\mathbf{x}}_f$ is encoded into $((i_1, \ldots, i_m), j^*)$ then for

$$\underline{\mathbf{x}}_f \in S_{i_1}^{[1,j^*],(\rho)}, \ldots, \underline{\mathbf{x}}_f \in S_{i_m}^{[m,j^*],(\rho)}.$$

The resulting distortion is

$$D^{(\rho)}[\underline{\mathbf{x}}_f, \tau] = \sum_{j'=1}^{\tau} \mathrm{P}(j'|j^*) \sum_{s=1}^{m} \sum_{l=1}^{N} \mathrm{P}(l|i_s) \|\mathrm{u}_s(\underline{\mathbf{x}}_f) - \underline{c}_l^{[s,j'],(\rho)}\|^2.$$

5. Repeat Steps 3 and 4 for $f = 1, \ldots, M$. Then calculate the centroids, using

$$\underline{c}_l^{[s,j],(\rho+1)} = \frac{\sum_{j^*=1}^{2^\eta} \sum_{i=1}^{N} \mathrm{P}(j|j^*)\mathrm{P}(l|i) \sum_{\underline{\mathbf{x}}:\underline{\mathbf{x}} \in S_i^{[s,j^*],(\rho)}} \mathrm{u}_s(\underline{\mathbf{x}})}{\sum_{j^*=1}^{2^\eta} \sum_{i=1}^{N} \mathrm{P}(j|j^*)\mathrm{P}(l|i) \sum_{\underline{\mathbf{x}}:\underline{\mathbf{x}} \in S_i^{[s,j^*],(\rho)}}}$$

and update the set of codebooks to $\{\mathbf{C}_j^{(\rho+1)}\}_{j=1}^{\tau}$ using the new centroids. Finally calculate the overall distortion

$$\mathcal{D}^{(\rho)}[\tau] = \frac{1}{kmM} \sum_{f=1}^{M} D^{(\rho)}[\underline{\mathbf{x}}_f, \tau].$$

.

6. Check $\frac{\mathcal{D}^{(\rho-1)}[\tau]-\mathcal{D}^{(\rho)}[\tau]}{\mathcal{D}^{(\rho)}[\tau]} \leq \delta$ or $\rho \geq Maxiter$, if so then go to Step 2; otherwise set $\rho = \rho + 1$ and go to Step 3.

This algorithm assumes an initial set of codebooks $\mathbf{C}_1^{(0)}$ for the (k,m,N,η) COm-SAPQ which is obtained from a (k,m,N) COPQ designed for the same ϵ_d using the algorithm of Section 2.4.4. The algorithm for the design of the (k,m,N,η) CO1-SAPQ codebooks is easily deducible from the above design. Similarly the (k,m,N,η) CO1-SAPQ algorithm starts off with only one codebook which can be obtained from a (k,N) COVQ, again designed with the same ϵ_d using the algorithm of Section 2.2.5.

4.7 Numerical Results

Numerical results were produced in order to compare the performances of the channel optimized sample adaptive product quantizer (COSAPQ) over other generic channel optimized quantizers such as the channel optimized vector quantizer (COVQ), of Section 2.2.5, and the channel optimized product quantizer (COPQ), of Section 2.4.4. The design algorithms of Sections 2.2.5, 2.4.4, and 4.6 were used to design the COVQ, COPQ, COm-SAPQ and CO1-SAPQ. The design parameters used to generate the numerical results of this section were: a maximum number of iterations $Maxiter = 200$, a splitting constant $\epsilon = 0.001$, and a stopping threshold $\delta = 0.001$. The same

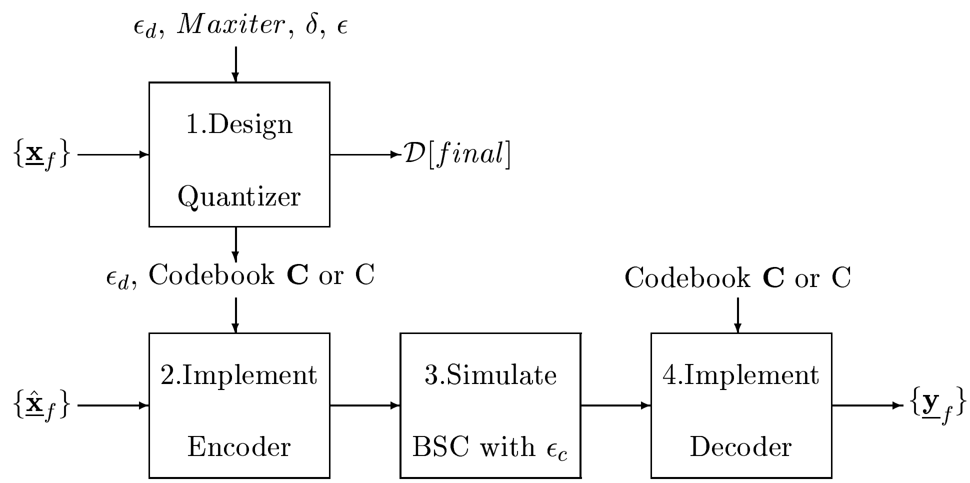

Figure 4.4: Block Diagram illustrating the validation of channel optimized quantizers.

design parameters were used for designing the codebooks of the COVQ, COPQ, COm-SAPQ and CO1-SAPQ. Furthermore 200,000 training source samples were used to design each codebook. The sources considered were the unit variance zero mean, Gauss-Markov and the memoryless Gaussian sources. A Gauss-Markov source is a sequence $\{\mathbf{X}_i\}$ described by the recursion

$$\mathbf{X}_i = \rho \mathbf{X}_{i-1} + \mathbf{U}_i$$

where $\{\mathbf{U}_i\}$ is a sequence of independent and identically distributed (i.i.d) Gaussian random variables, and ρ is the correlation coefficient. A memoryless Gaussian sequence and a Gauss-Markov sequence was generated by setting the correlation coefficient to $\rho = 0.0$ and $\rho = 0.9$, respectively. In some cases the designed codebooks of the COVQ, COPQ, COm-SAPQ and CO1-SAPQ were tested by implementing the quantizers as in Figure 4.4. Figure 4.4 depicts how

1. The channel optimized quantizer is designed using the algorithms of Section 2.2.5, Section 2.4.4, and Section 4.6 with: design BSC cross over probability ϵ_d, split-

ting constant $\epsilon = 0.001$, stopping threshold $\delta = 0.001$ and maximum number of iterations $Maxiter = 200$. The quantizers are designed using the training sequence $\{\underline{\mathbf{x}}_f\}$. The final distortion of the quantizers is $\mathcal{D}[final]$.

2. The codebook $\mathbf{C}$ produced by the design algorithm and ϵ_d is then used to implement the encoder by calculating and storing the sets: $\{\underline{y}_{s,j}^{(\rho)}(\gamma)\}$ and $\{\alpha_{s,j}^{(\rho)}(\gamma)\}$ as in (4.21), or $\{\underline{y}_j^{(\rho)}(\gamma)\}$ and $\{\alpha_j^{(\rho)}(\gamma)\}$ as in (4.26), or $\{\underline{y}_j^{(\rho)}(\gamma)\}$ and $\{\alpha_j^{(\rho)}(\gamma)\}$ as in (2.30), or $\{\mathrm{y}^{(\rho)}(\gamma)\}_{\gamma=1}^{N}$ and $\{\alpha^{(\rho)}(\gamma)\}_{\gamma=1}^{N}$ as in (2.14). The encoder is then used to encode the testing sequence $\{\underline{\hat{\mathbf{x}}}_f\} \neq \{\underline{\mathbf{x}}_f\}$; a sequence that is entirely different to the training sequence.

3. The output of the encoder is feed into a simulated BSC with cross over probability ϵ_c.

4. The codebook produced from Step 1 is also used to implement a decoder that decodes the output of the simulated BSC. The resultant reconstruction of the testing sequence is $\{\underline{\mathbf{y}}_f\}$. The distortion of the implemented quantizer can then be calculated as

$$\mathbf{D} = \sum_f \|\underline{\hat{\mathbf{x}}}_f - \underline{\mathbf{y}}_f\|^2.$$

The performance of the channel optimized quantizers is measured using the signal-to-distortion ratio (SDR) which is equal to SDR $= -10\log_{10}(\mathcal{D}[final])$ or SDR $= -10\log_{10}(\mathbf{D})$. The comparison of the channel optimized quantizers is done by comparing the performance, the encoding complexity and the storage requirement of each channel optimized quantizer at the same rate. The criteria here is three-fold that is: performance (SDR), encoding complexity (complexity=total multiplications required for encoding/source sample), and storage requirement (storage=total scalars required for implementation).

4.7.1 Comparing COVQ, COPQ, COm-SAPQ and CO1-SAPQ

Tables 4.2 and 4.3 tabulate the performances (SDR) of COVQ, COPQ, COm-SAPQ and CO1-SAPQ, respectively, at rates $R = 1.0, 2.0$ and 3.0 for memoryless Gaussian and Gauss-Markov sources. The encoding complexities and storage requirements were calculated using Table 4.1 and the performances were calculated using SDR $= -10\log_{10}(\mathcal{D}[final])$. Note that since product quantizers are being compared with quantizers, km is used to measure the dimension of the quantizers; $m = 1$ for all channel optimized vector quantizers (COVQ).

Memoryless Gaussian Source: In Table 4.2 compare COVQ and COm-SAPQ for rates $R = 1.0, 2.0$ and 3.0. For memoryless Gaussian sources and $\epsilon_d > 0$, COm-SAPQ performs within 0.2 dB of COVQ of the same rate R and design BSC cross over probability ϵ_d. The performance of COm-SAPQ can exceed that of COVQ by up to 0.19 dB; compare (2,64) COVQ and (1,2,2,4) COm-SAPQ at rate $R = 3.0$ and $\epsilon_d = 0.005$. This performance gain by COm-SAPQ over COVQ, is also matched by a degradation of up to 0.16 dB over COVQ; compare (4,64) COVQ and (2,2,2,2) COm-SAPQ at rate $R = 1.0$ and $\epsilon_d = 0.005$. Still in all cases COm-SAPQ attains a SDR within 0.2 dB of that of COVQ with an encoding complexity equal to half that of COVQ at the same rate R, and a lower ($\frac{3}{5}$ to $\frac{5}{9}$) storage requirement to that of COVQ at the same rate R.

In Table 4.2, if we compare the least complex CO1-SAPQ to COPQ of the same rate R, we see that CO1-SAPQ out-performs COPQ only at rates $R = 1.0$ and 2.0, and when $\epsilon_d \geq 0.050$. The performance gain is in the range of 0.57-0.21 dB. Otherwise, for rate $R = 3.0$ COPQ outperforms CO1-SAPQ by up to 0.46 dB.

Gauss-Markov Sources: In Table 4.3, we observe that for Gauss-Markov sources COm-SAPQ performs 1.0-0.33 dB lower than COVQ of the same rate R and dimension km. However when the dimension km of CO1-SAPQ is increased, a performance gain of up to 0.81 dB is achieved over COVQ; compare (2,64) COVQ and (1,4,4,4) CO1-SAPQ at rate $R = 3.0$ and $\epsilon_d = 0.005$. The gain over COVQ decreases as ϵ_d increases. However at times COVQ does perform up to 0.23 dB better than CO1-SAPQ. Still CO1-SAPQ attains SDRs of up to 0.81 db more than COVQ while maintaining a lower ($\frac{3}{5}$ to $\frac{1}{3}$) storage requirement then COVQ of the same rate R.

Note that in Table 4.3, the least complex CO1-SAPQ always outperforms COPQ at the same rate. The performance gains of the least complex CO1-SAPQ is in the range of 3.08-1.12 dB.

4.7.2 Comparing 'noisy' and 'noiseless' quantizers

As in [16] Tables 4.4–4.9 compare the performances of the channel optimized quantizers to the quantizers designed under noiseless conditions. Note that the LBGVQ(+sim.annl.) is the LBGVQ design as in Section 2.2.4. Thus this LBGVQ(+sim.annl.) does include some joint channel-source coding in its design. All the quantizers were designed and tested as described in Figure 4.4 with $\epsilon_c = \epsilon_d$, and the performance is measured using SDR $= -10\log_{10}(\mathbf{D})$. As in [16] the same conclusion can be derived, that is the performance gain of the channel optimized sample adaptive product quantizers over the sample adaptive product quantizers increases when the channel noise ϵ_c is increased, when the dimension km is increased, when the rate R is increased, and when the correlation ρ is increased.

4.7.3 Comparing COSQ and COSAPQ

Since the channel optimized scalar quantizer (COSQ), is considered for image coding [8, 10] we compare the performance gain of COSAPQ over COSQ in Tables 4.10 and 4.11. Note that a rate $R = 1.0$, COSAPQ cannot be compare to a COSQ, since for $R = 1.0$ if $k = 1$ of the (k,m,N,η) COSAPQ then $1.0 - \frac{\eta}{m} = \log_2 N$. In Table 4.10 there is a definite gain of COm-SAPQ over COSQ. A gain by COm-SAPQ of 1.57-0.27 dB over COSQ of the same rate R is observed. In Table 4.11 there is a definite gain by the least complex CO1-SAPQ of 3.08-1.12 dB over COSQ of the same rate R.

4.7.4 Mismatch Conditions

Tables 4.4–4.9 test the results of Tables 4.2 and 4.3 by implementing the quantizers as in Figure 4.4 with $\epsilon_c = \epsilon_d$. The condition ($\epsilon_c = \epsilon_d$) where the design BSC cross over probability perfectly matches the 'realized' BSC cross over probability is an ideal condition. Normally the codebooks of the quantizers are designed using an estimate of the cross over probability of the BSC to be used for transmission. Hence a more realistic insight to the performance of the quantizers can be obtained by implementing the quantizers as in Figure 4.4, but with $\epsilon_c \neq \epsilon_d$. This is the mismatch conditions that are depicted in Figures 4.5–4.16 for memoryless Gaussian and Gauss-Markov sources. In Figures 4.5–4.7 the $\epsilon_d = 0.005$ and the source is memoryless Gaussian. In Figures 4.8–4.10 the $\epsilon_d = 0.050$ and the source is memoryless Gaussian. Similarly for Figures 4.11–4.13 the $\epsilon_d = 0.005$ and the source is Gauss-Markov, and in Figures ,4.14–4.16 the $\epsilon_d = 0.050$ and the source is Gauss-Markov.

Memoryless Gaussian Sources: For memoryless Gaussian sources Figures 4.5–4.10 illustrate that COm-SAPQ performs within 0.2 dB of COVQ designed with the same ϵ_d and at the same rate R, when the channel noise $\epsilon_c \leq 0.001$. But as the channel gets noisier ($\epsilon_c \geq 0.050$), the performance of COm-SAPQ converges to within 0.05 dB to that of COVQ designed with the same ϵ_d and at the same rate R.

Gauss-Markov Sources: For Gauss-Markov sources Figures 4.11–4.13 illustrate an up to 0.2 dB gain by CO1-SAPQ of higher km over COVQ designed with the same ϵ_d and of the same rate R. But as the channel is anticipated to be noisier (i.e ϵ_d is greater) and as the channel does get noisier ($\epsilon_c \geq 0.050$), CO1-SAPQ performance approaches (within 0.02 dB) that of COVQ designed with the same ϵ_d and at the same rate R. Although if the channel is anticipated to be noisy and the channel in reality is not as noisy ($\epsilon_c \leq 0.01$), a gain of up to 0.5 dB is attained by CO1-SAPQ of higher km, over COVQ designed with the same ϵ_d and at the same rate R.

4.7.5 Summary of Numerical Results

Our numerical results illustrate that when the source is memoryless Gaussian, and when the channel being used is anticipated to be noisy ($\epsilon_d > 0$), COm-SAPQ will perform within 0.2 dB of the performance of a COVQ, designed with the same design parameters. As the channel gets noisier ($\epsilon_c \geq 0.050$), COm-SAPQ's performance converges closer to that of COVQ (within 0.05 dB). These performances are attained with half the encoding complexity and lower ($\frac{3}{5}$ to $\frac{5}{9}$) storage requirements then COVQ at the same rate R and dimension km. In all cases COm-SAPQ outperforms COPQ and COSQ.

Furthermore if the source is Gauss-Markov, CO1-SAPQ with a higher dimension km, but the same encoding complexity, as COVQ, at the same rate R, will outperform COVQ designed with the same design parameters by 0.2 dB. As the channel gets noisier ($\epsilon_c, \epsilon_d \geq 0.050$) CO1-SAPQ performance converges to within 0.02 dB of that of COVQ, of the same rate R and encoding complexity, but with CO1-SAPQ having the advantage of lower ($\frac{3}{5}$ to $\frac{5}{9}$) storage requirements then COVQ. But when the channel gets noisier ($\epsilon_d \geq 0.050$) and the channel actually used for transmission is not as noisy as anticipated ($\epsilon_c < 0.01$) then CO1-SAPQ will have a gain of up to 0.5 dB over COVQ of the same rate R. At all times CO1-SAPQ outperforms COPQ and COSQ when the source is Gauss-Markov.

R	km	Quantizer	ϵ_d						complexity	storage
			0.000	0.005	0.010	0.050	0.100	0.150		
3.0	2	COVQ $k = 2, N = 64$	15.23	12.19	11.07	7.35	5.11	3.78	64	320
		COPQ $k = 1, N = 8$	14.57	12.00	10.47	5.62	4.65	3.48	8	48
		COm-SAPQ $k = 1, N = 2, \eta = 4$	15.07	12.38	11.04	7.20	5.14	3.75	32	192
		CO1-SAPQ $k = 1, N = 2, \eta = 4$	13.59	11.54	10.29	6.48	4.35	3.13	32	96
	4	CO1-SAPQ $k = 1, N = 4, \eta = 4$	15.19	12.41	10.92	6.61	4.47	3.18	64	192
2.0	2	COVQ $k = 2, N = 16$	9.64	8.71	8.02	5.52	3.82	2.71	16	80
		COPQ $k = 1, N = 4$	9.27	8.50	7.86	4.85	3.04	1.99	4	24
		COm-SAPQ $k = 1, N = 2, \eta = 2$	9.51	8.71	8.10	5.48	3.86	2.79	8	48
		CO1-SAPQ $k = 1, N = 2, \eta = 2$	8.72	8.04	7.50	5.16	3.61	2.50	8	24
	3	CO1-SAPQ $k = 1, N = 2, \eta = 3$	8.92	8.16	7.58	5.06	3.45	2.45	16	48
1.0	4	COVQ $k = 4, N = 16$	4.66	4.44	4.24	3.14	2.26	1.61	16	144
		COPQ $k = 2, N = 4$	4.38	4.16	3.96	2.72	1.75	1.14	4	40
		COm-SAPQ $k = 2, N = 2, \eta = 2$	4.47	4.28	4.09	3.13	2.26	1.61	8	80
		CO1-SAPQ $k = 2, N = 2, \eta = 2$	4.41	4.15	3.95	2.81	1.96	1.44	8	40
	6	CO1-SAPQ $k = 2, N = 2, \eta = 3$	4.53	4.25	4.01	2.73	1.95	1.39	16	80

Table 4.2: SDR (dB) performances, encoding complexity and storage requirement comparisons for the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, at rates R and dimension km, designed using 200,000 memoryless Gaussian training samples and BSC cross over probability ϵ_d.

R	km	Quantizer	ϵ_d						complexity	storage
			0.000	0.005	0.010	0.050	0.100	0.150		
3.0	2	COVQ $k=2, N=64$	19.03	14.50	13.58	9.36	6.80	5.13	64	320
		COPQ $k=1, N=8$	14.57	12.00	10.47	5.60	4.63	3.46	8	48
		COm-SAPQ $k=1, N=2, \eta=4$	16.62	14.22	12.75	8.43	6.07	4.62	32	192
		CO1-SAPQ $k=1, N=2, \eta=4$	17.24	14.02	12.65	8.68	6.25	4.58	32	96
	4	CO1-SAPQ $k=1, N=4, \eta=4$	19.72	15.31	13.85	9.51	6.88	5.19	64	192
2.0	2	COVQ $k=2, N=16$	13.54	11.39	10.04	7.27	5.27	3.82	16	80
		COPQ $k=1, N=4$	9.28	8.50	7.85	4.83	3.02	1.96	4	24
		COm-SAPQ $k=1, N=2, \eta=2$	12.50	10.76	9.71	6.26	4.53	3.37	8	48
		CO1-SAPQ $k=1, N=2, \eta=2$	12.50	10.76	9.71	6.26	4.53	3.37	8	24
	3	CO1-SAPQ $k=1, N=2, \eta=3$	13.95	11.81	10.68	7.08	5.21	3.93	16	48
1.0	4	COVQ $k=4, N=16$	10.20	9.15	8.36	6.24	4.64	3.42	16	144
		COPQ $k=2, N=4$	7.89	7.35	6.87	4.41	2.81	1.84	4	40
		COm-SAPQ $k=2, N=2, \eta=2$	9.66	8.78	8.17	5.63	4.08	3.10	8	80
		CO1-SAPQ $k=2, N=2, \eta=2$	9.52	8.63	8.01	5.55	4.11	3.09	8	40
	6	CO1-SAPQ $k=2, N=2, \eta=3$	9.99	9.11	8.51	6.09	4.58	3.50	16	48

Table 4.3: SDR (dB) performances, encoding complexity and storage requirement comparisons for the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, at rates R and dimension km, designed using 200,000 Gauss-Markov training samples and BSC cross over probability ϵ_d.

Quantizer	ϵ_c					
	0.000	0.005	0.010	0.050	0.100	0.150
LBGVQ(+sim.annl.) $k = 2, N = 64$	15.20	11.39	9.95	4.84	2.24	0.54
PQ $k = 1, m = 2, N = 8$	14.60	12.03	10.32	4.96	2.17	0.52
m-SAPQ $k = 1, m = 2, N = 2, \eta = 4$	15.09	11.97	10.17	4.58	1.86	0.28
1-SAPQ $k = 1, m = 2, N = 2, \eta = 4$	13.59	11.21	9.64	4.41	1.80	0.25
1-SAPQ $k = 1, m = 4, N = 4, \eta = 4$	15.16	12.16	10.44	4.78	2.07	0.50
COVQ $k = 2, N = 64$	15.20	12.19	11.06	7.36	5.13	3.79
COPQ $k = 1, m = 2, N = 8$	14.60	12.02	10.50	5.61	4.64	3.48
CO-m-SAPQ $k = 1, m = 2, N = 2, \eta = 4$	15.09	12.39	11.04	7.21	5.16	3.74
CO-1-SAPQ $k = 1, m = 2, N = 2, \eta = 4$	13.59	11.60	10.31	6.47	4.35	3.13
CO-1-SAPQ $k = 1, m = 4, N = 4, \eta = 4$	15.16	12.47	10.92	6.55	4.46	3.15

Table 4.4: SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, to the (k,N) LBGVQ, with simmulated annealing, the (k,m,N) PQ, the (k,m,N,η) m-SAPQ, and the (k,m,N,η) 1-SAPQ, at rate $R = 3.0$, using 200,000 memoryless Gaussian testing samples and a simmulated BSC cross over probability ϵ_c.

Quantizer	ϵ_c					
	0.000	0.005	0.010	0.050	0.100	0.150
LBGVQ(+sim.annl.) $k=2, N=16$	9.66	8.63	7.82	4.21	2.36	0.94
PQ $k=1, m=2, N=4$	9.29	8.47	7.82	4.59	2.42	1.02
m-SAPQ $k=1, m=2, N=2, \eta=2$	9.52	8.68	7.97	4.56	2.36	0.94
1-SAPQ $k=1, m=2, N=2, \eta=2$	8.73	8.01	7.40	4.26	2.16	0.77
1-SAPQ $k=1, m=3, N=2, \eta=3$	8.89	8.09	7.42	4.16	2.03	0.62
COVQ $k=2, N=16$	9.66	8.70	7.98	5.51	3.82	2.72
COPQ $k=1, m=2, N=4$	9.29	8.49	7.86	4.87	3.06	2.01
COm-SAPQ $k=1, m=2, N=2, \eta=2$	9.52	8.73	8.10	5.50	3.88	2.81
CO1-SAPQ $k=1, m=2, N=2, \eta=2$	8.73	8.05	7.51	5.18	3.63	2.60
CO1-SAPQ $k=1, m=3, N=2, \eta=3$	8.89	8.14	7.55	5.06	3.47	2.46

Table 4.5: SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, to the (k,N) LBGVQ, with simmulated annealing, the (k,m,N) PQ, the (k,m,N,η) m-SAPQ, and the (k,m,N,η) 1-SAPQ, at rate $R = 2.0$, using 200,000 memoryless Gaussian testing samples and a simmulated BSC cross over probability ϵ_c.

Quantizer	ϵ_c					
	0.000	0.005	0.010	0.050	0.100	0.150
LBGVQ(+sim.annl.) $k = 4, N = 16$	4.65	4.41	4.13	2.82	1.62	0.73
PQ $k = 2, m = 2, N = 4$	4.38	4.15	3.95	2.62	1.43	0.56
m-SAPQ $k = 2, m = 2, N = 2, \eta = 2$	4.46	4.26	4.05	2.80	1.66	0.80
1-SAPQ $k = 2, m = 2, N = 2, \eta = 2$	4.41	4.15	3.91	2.47	1.24	0.37
1-SAPQ $k = 2, m = 3, N = 2, \eta = 3$	4.52	4.23	3.95	2.33	1.03	0.14
COVQ $k = 4, N = 16$	4.65	4.42	4.15	2.99	2.27	1.62
COPQ $k = 2, m = 2, N = 4$	4.38	4.17	3.97	2.75	1.78	1.16
COm-SAPQ $k = 2, m = 2, N = 2, \eta = 2$	4.46	4.26	4.07	2.94	2.25	1.61
CO1-SAPQ $k = 2, m = 2, N = 2, \eta = 2$	4.41	4.14	3.94	2.81	1.96	1.45
CO1-SAPQ $k = 2, m = 3, N = 2, \eta = 3$	4.52	4.23	4.00	2.74	1.95	1.35

Table 4.6: SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, to the (k,N) LBGVQ, with simmulated annealing, the (k,m,N) PQ, the (k,m,N,η) m-SAPQ, and the (k,m,N,η) 1-SAPQ, at rate $R = 1.0$, using 200,000 memoryless Gaussian testing samples and a simmulated BSC cross over probability ϵ_c.

Quantizer	ϵ_c					
	0.000	0.005	0.010	0.050	0.100	0.150
LBGVQ(+sim.annl.) $k = 2, N = 64$	19.01	13.63	11.07	5.29	2.37	0.71
PQ $k = 1, m = 2, N = 8$	14.64	12.01	10.39	5.01	2.25	0.57
m-SAPQ $k = 1, m = 2, N = 2, \eta = 4$	16.65	12.82	10.74	4.85	2.09	0.48
1-SAPQ $k = 1, m = 2, N = 2, \eta = 4$	17.26	12.87	10.74	4.74	1.97	0.38
1-SAPQ $k = 1, m = 4, N = 4, \eta = 4$	19.71	13.42	10.98	4.55	1.69	0.05
COVQ $k = 2, N = 64$	19.01	14.62	13.62	9.48	6.85	5.20
COPQ $k = 1, m = 2, N = 8$	14.60	12.02	10.50	5.57	4.60	3.45
CO-m-SAPQ $k = 1, m = 2, N = 2, \eta = 4$	16.64	14.30	12.75	8.45	6.10	4.66
CO-1-SAPQ $k = 1, m = 2, N = 2, \eta = 4$	17.26	14.11	12.63	8.71	6.27	4.63
CO-1-SAPQ $k = 1, m = 4, N = 4, \eta = 4$	19.71	15.27	13.94	9.52	6.91	5.21

Table 4.7: SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, to the (k,N) LBGVQ, with simmulated annealing, the (k,m,N) PQ, the (k,m,N,η) m-SAPQ, and the (k,m,N,η) 1-SAPQ, at rate $R = 3.0$, using 200,000 Gauss-Markov testing samples and a simmulated BSC cross over probability ϵ_c.

Quantizer	ϵ_c					
	0.000	0.005	0.010	0.050	0.100	0.150
LBGVQ(+sim.annl.) $k = 2, N = 16$	13.56	11.30	9.86	4.82	2.18	0.57
PQ $k = 1, m = 2, N = 4$	9.32	8.54	7.89	4.65	2.44	1.02
m-SAPQ $k = 1, m = 2, N = 2, \eta = 2$	12.54	10.50	9.13	4.21	1.62	0.07
1-SAPQ $k = 1, m = 2, N = 2, \eta = 2$	12.54	10.52	9.14	4.23	1.65	0.09
1-SAPQ $k = 1, m = 3, N = 2, \eta = 3$	13.98	11.07	9.45	3.98	1.29	-0.26
COVQ $k = 2, N = 16$	13.56	11.38	10.05	7.31	5.30	3.90
COPQ $k = 1, m = 2, N = 4$	9.32	8.55	7.92	4.94	3.10	2.04
COm-SAPQ $k = 1, m = 2, N = 2, \eta = 2$	12.54	10.78	9.74	6.32	4.60	3.44
CO1-SAPQ $k = 1, m = 2, N = 2, \eta = 2$	12.54	10.79	9.74	6.33	4.60	3.44
CO1-SAPQ $k = 1, m = 3, N = 2, \eta = 3$	13.98	11.78	10.69	7.12	5.37	3.98

Table 4.8: SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, to the (k,N) LBGVQ, with simmulated annealing, the (k,m,N) PQ, the (k,m,N,η) m-SAPQ, and the (k,m,N,η) 1-SAPQ, at rate $R = 2.0$, using 200,000 Gauss-Markov testing samples and a simmulated BSC cross over probability ϵ_c.

Quantizer	ϵ_c					
	0.000	0.005	0.010	0.050	0.100	0.150
LBGVQ(+sim.annl.) $k = 4, N = 16$	10.21	9.10	8.26	4.50	2.13	0.59
PQ $k = 2, m = 2, N = 4$	7.93	7.37	6.91	4.26	2.31	0.99
m-SAPQ $k = 2, m = 2, N = 2, \eta = 2$	9.71	8.71	7.84	4.16	1.87	0.44
1-SAPQ $k = 2, m = 2, N = 2, \eta = 2$	9.57	8.51	7.64	3.91	1.61	0.18
1-SAPQ $k = 2, m = 3, N = 2, \eta = 3$	10.01	8.99	8.23	4.55	2.23	0.79
COVQ $k = 4, N = 16$	10.21	9.15	8.30	6.20	4.63	3.40
COPQ $k = 2, m = 2, N = 4$	7.93	7.37	6.93	4.51	2.90	1.93
COm-SAPQ $k = 2, m = 2, N = 2, \eta = 2$	9.71	8.81	8.16	5.65	4.11	3.16
CO1-SAPQ $k = 2, m = 2, N = 2, \eta = 2$	9.57	8.67	8.00	5.58	4.14	3.15
CO1-SAPQ $k = 2, m = 3, N = 2, \eta = 3$	10.01	9.12	8.52	6.12	4.61	3.54

Table 4.9: SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, to the (k,N) LBGVQ, with simmulated annealing, the (k,m,N) PQ, the (k,m,N,η) m-SAPQ, and the (k,m,N,η) 1-SAPQ, at rate $R = 1.0$, using 200,000 Gauss-Markov testing samples and a simmulated BSC cross over probability ϵ_c.

Rate	km	Quantizer	ϵ_d						complexity	storage
			0.000	0.005	0.010	0.050	0.100	0.150		
3.0	1	COSQ $k=1, N=8$	14.50	12.00	10.47	5.63	4.66	3.48	8	8
	2	COVQ $k=2, N=64$	15.23	12.19	11.07	7.35	5.11	3.78	64	320
	2	COm-SAPQ $k=1, N=2, \eta=4$	15.07	12.38	11.04	7.20	5.14	3.75	32	192
	2	CO1-SAPQ $k=1, N=2, \eta=4$	13.59	11.54	10.29	6.48	4.35	3.13	32	96
	4	CO1-SAPQ $k=1, N=4, \eta=4$	15.19	12.41	10.92	6.61	4.47	3.18	64	192
2.0	1	COSQ $k=1, N=4$	9.28	8.50	7.86	4.85	3.04	1.99	4	4
	2	COVQ $k=2, N=16$	9.64	8.71	8.02	5.52	3.82	2.71	16	80
	2	COm-SAPQ $k=1, N=2, \eta=2$	9.51	8.71	8.10	5.48	3.86	2.79	8	48
	2	CO1-SAPQ $k=1, N=2, \eta=2$	8.72	8.04	7.50	5.16	3.61	2.50	8	24
	3	CO1-SAPQ $k=1, N=2, \eta=3$	8.92	8.16	7.58	5.06	3.45	2.45	16	48

Table 4.10: SDR (dB) performances, encoding complexity and storage requirement comparisons for the $(1,N)$ COSQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, at rates $R = 2.0, 3.0$, designed using 200,000 memoryless Gaussian training samples and BSC cross over probability ϵ_d.

Rate	km	Quantizer	ϵ_d 0.000	0.005	0.010	0.050	0.100	0.150	complexity	storage
3.0	1	COSQ $k=1, N=8$	14.57	12.00	10.46	5.60	4.63	3.45	8	8
	2	COVQ $k=2, N=64$	19.03	14.50	13.58	9.36	6.80	5.13	64	320
	2	COm-SAPQ $k=1, N=2, \eta=4$	16.62	14.22	12.75	8.43	6.07	4.62	32	192
	2	CO1-SAPQ $k=1, N=2, \eta=4$	17.24	14.02	12.65	8.68	6.25	4.58	32	96
	4	CO1-SAPQ $k=1, N=4, \eta=4$	19.72	15.31	13.85	9.51	6.88	5.19	64	192
2.0	1	COSQ $k=1, N=4$	9.28	8.50	7.85	4.83	3.02	1.96	4	4
	2	COVQ $k=2, N=16$	13.54	11.39	10.04	7.27	5.27	3.82	16	80
	2	COm-SAPQ $k=1, N=2, \eta=2$	12.50	10.76	9.71	6.26	4.53	3.37	8	48
	2	CO1-SAPQ $k=1, N=2, \eta=2$	12.50	10.76	9.71	6.26	4.53	3.37	8	24
	3	CO1-SAPQ $k=1, N=2, \eta=3$	13.95	11.81	10.68	7.08	5.21	3.93	16	48

Table 4.11: SDR (dB) performances, encoding complexity and storage requirement comparisons for the (1,N) COSQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ, at rates R and dimension km, designed using 200,000 Gauss-Markov training samples and BSC cross over probability ϵ_d.

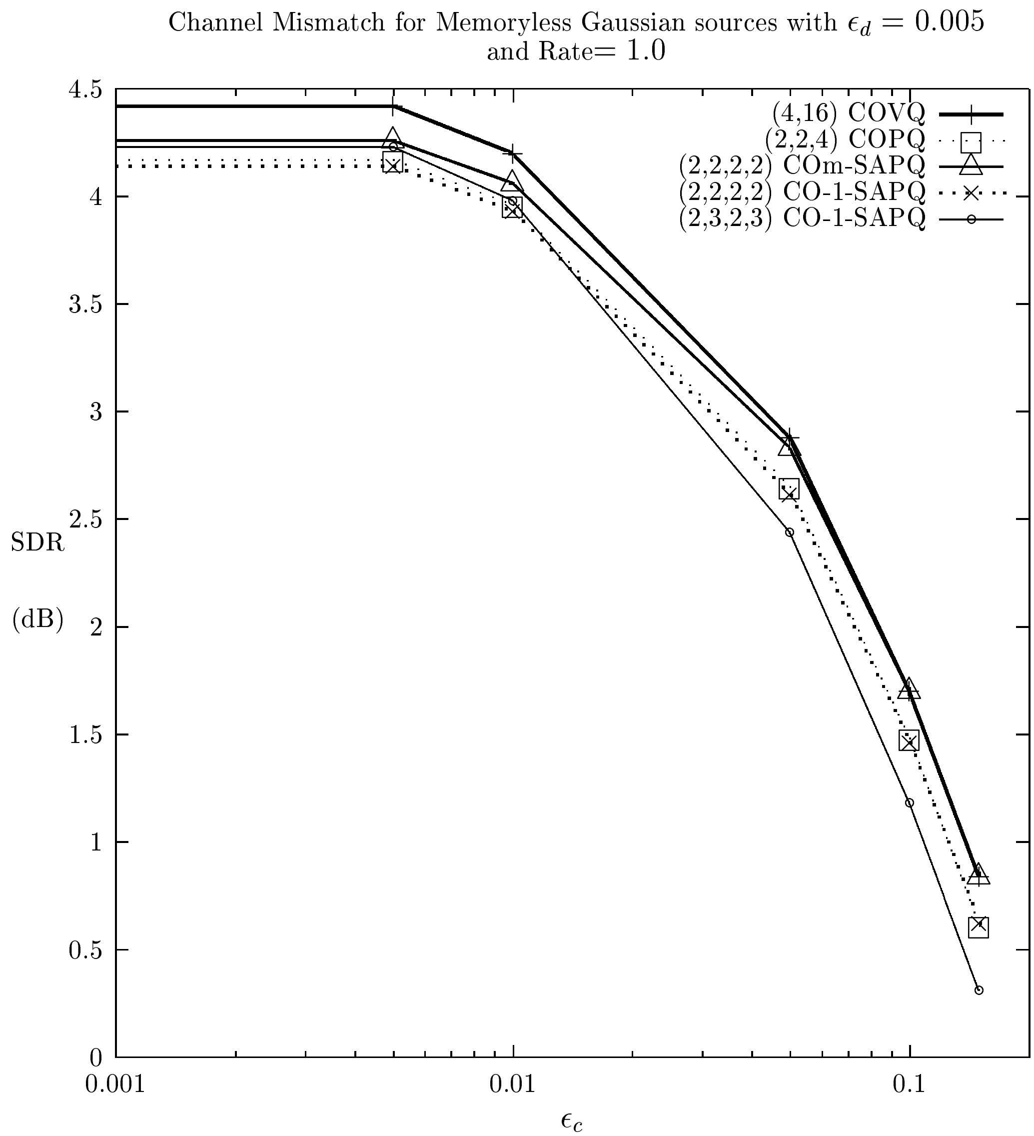

Figure 4.5: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 1.0$, using 200,000 memoryless Gaussian testing samples and a simmulated BSC cross over probability ϵ_c

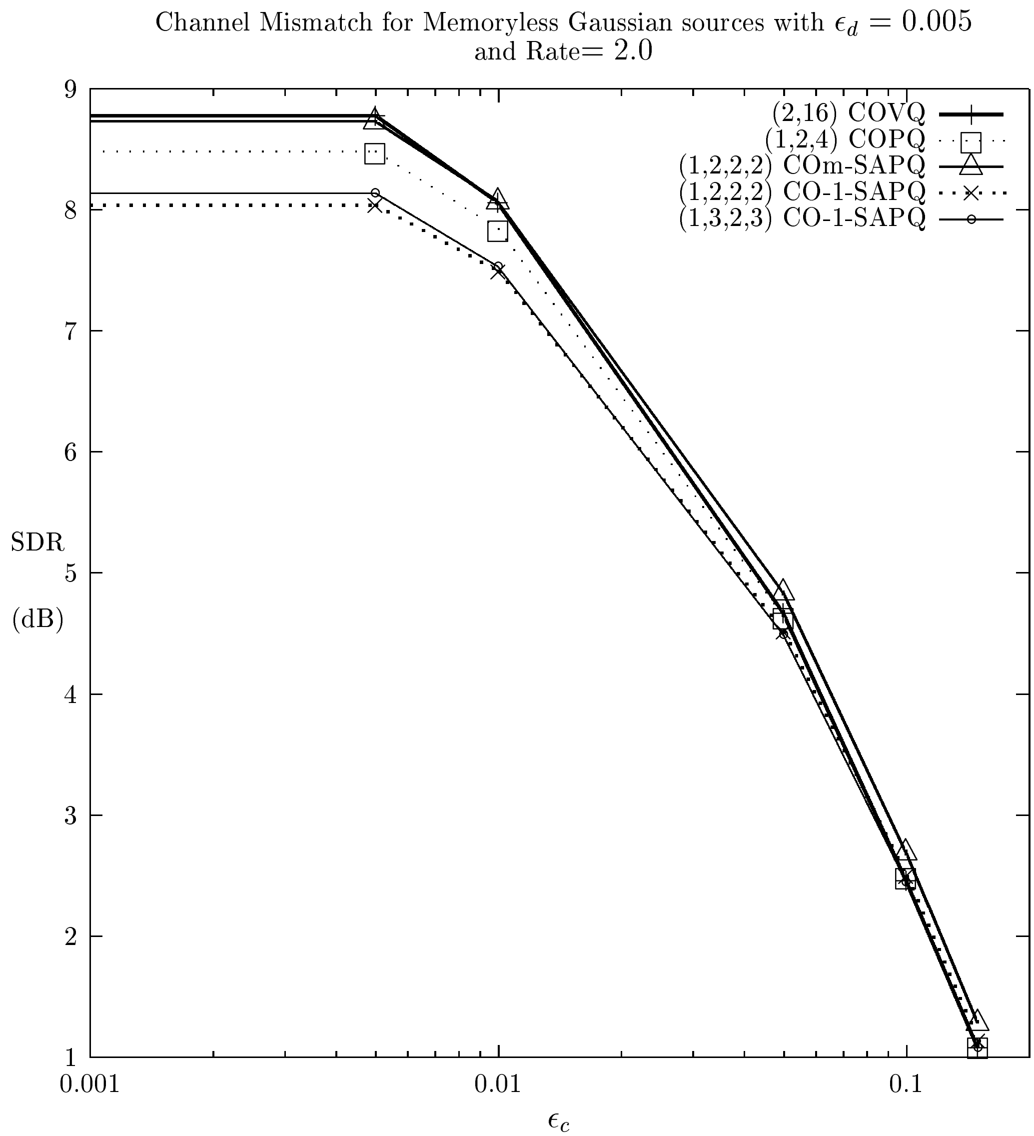

Figure 4.6: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 2.0$, using 200,000 memoryless Gaussian testing samples and a simmulated BSC cross over probability ϵ_c

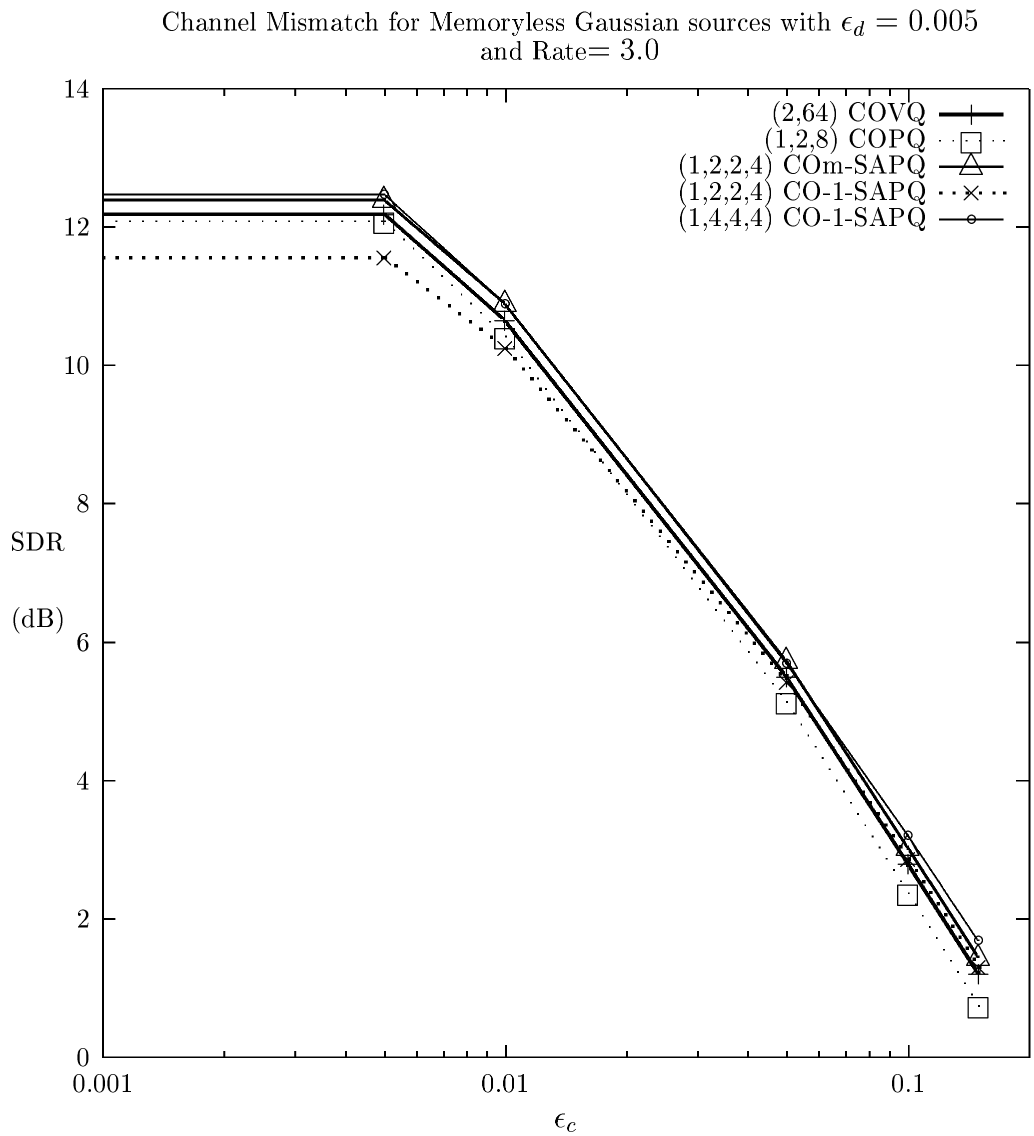

Figure 4.7: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 3.0$, using 200,000 memoryless Gaussian testing samples and a simmulated BSC cross over probability ϵ_c

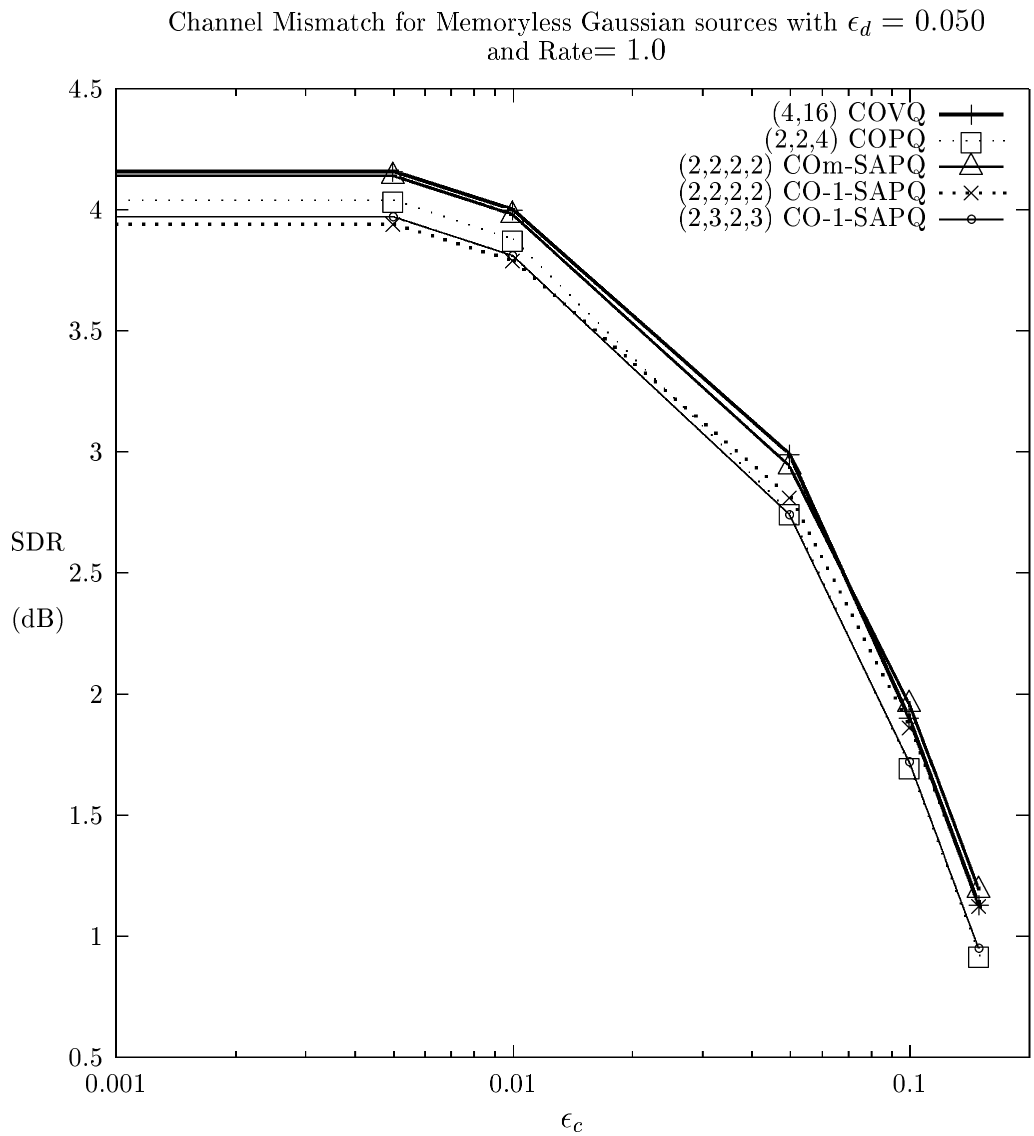

Figure 4.8: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 1.0$, using 200,000 memoryless Gaussian testing samples and a simmulated BSC cross over probability ϵ_c

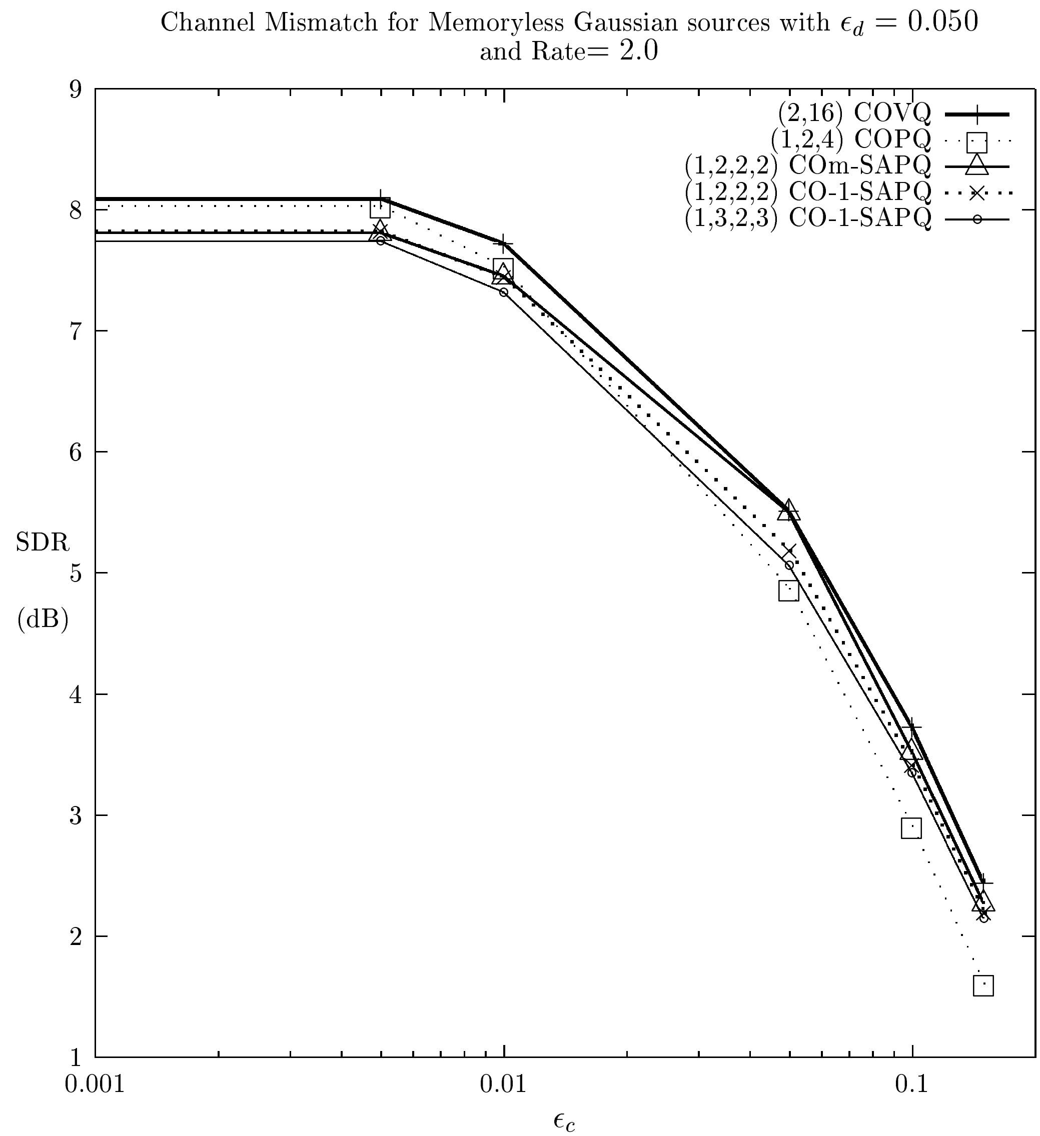

Figure 4.9: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 2.0$, using 200,000 memoryless Gaussian testing samples and a simmulated BSC cross over probability ϵ_c

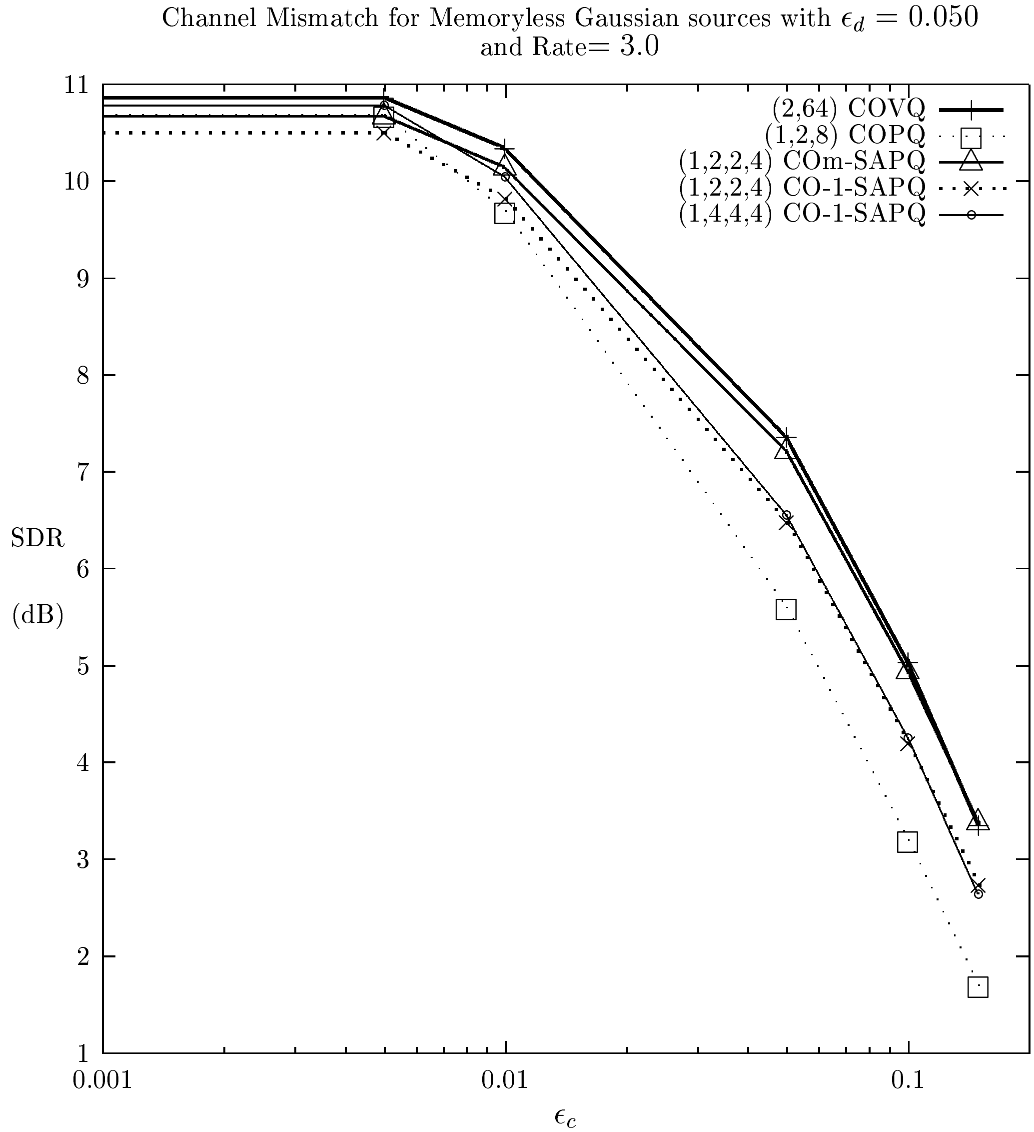

Figure 4.10: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 3.0$, using 200,000 memoryless Gaussian testing samples and a simmulated BSC cross over probability ϵ_c

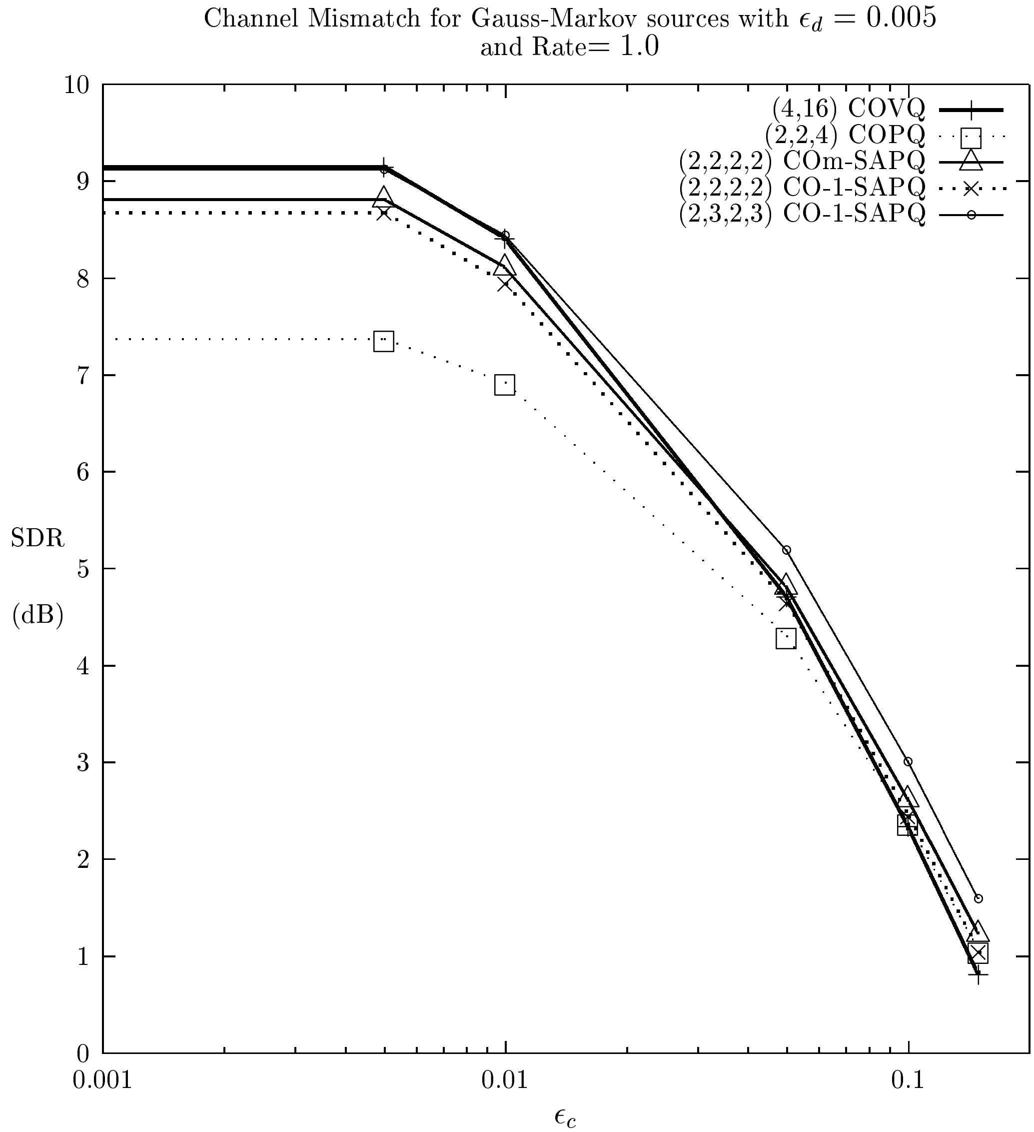

Figure 4.11: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 1.0$, using 200,000 Gauss-Markov testing samples and a simmulated BSC cross over probability ϵ_c

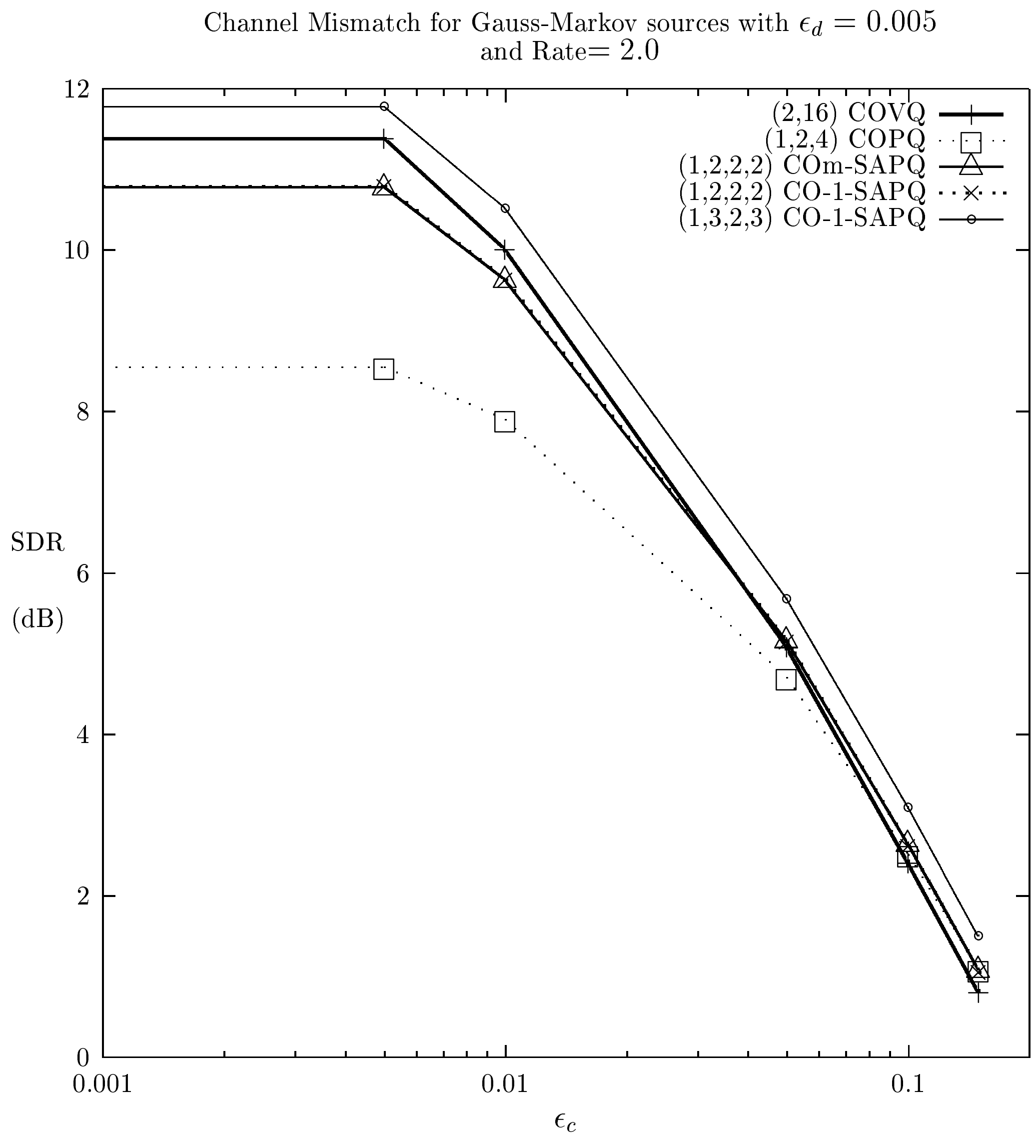

Figure 4.12: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 2.0$, using 200,000 Gauss-Markov testing samples and a simmulated BSC cross over probability ϵ_c

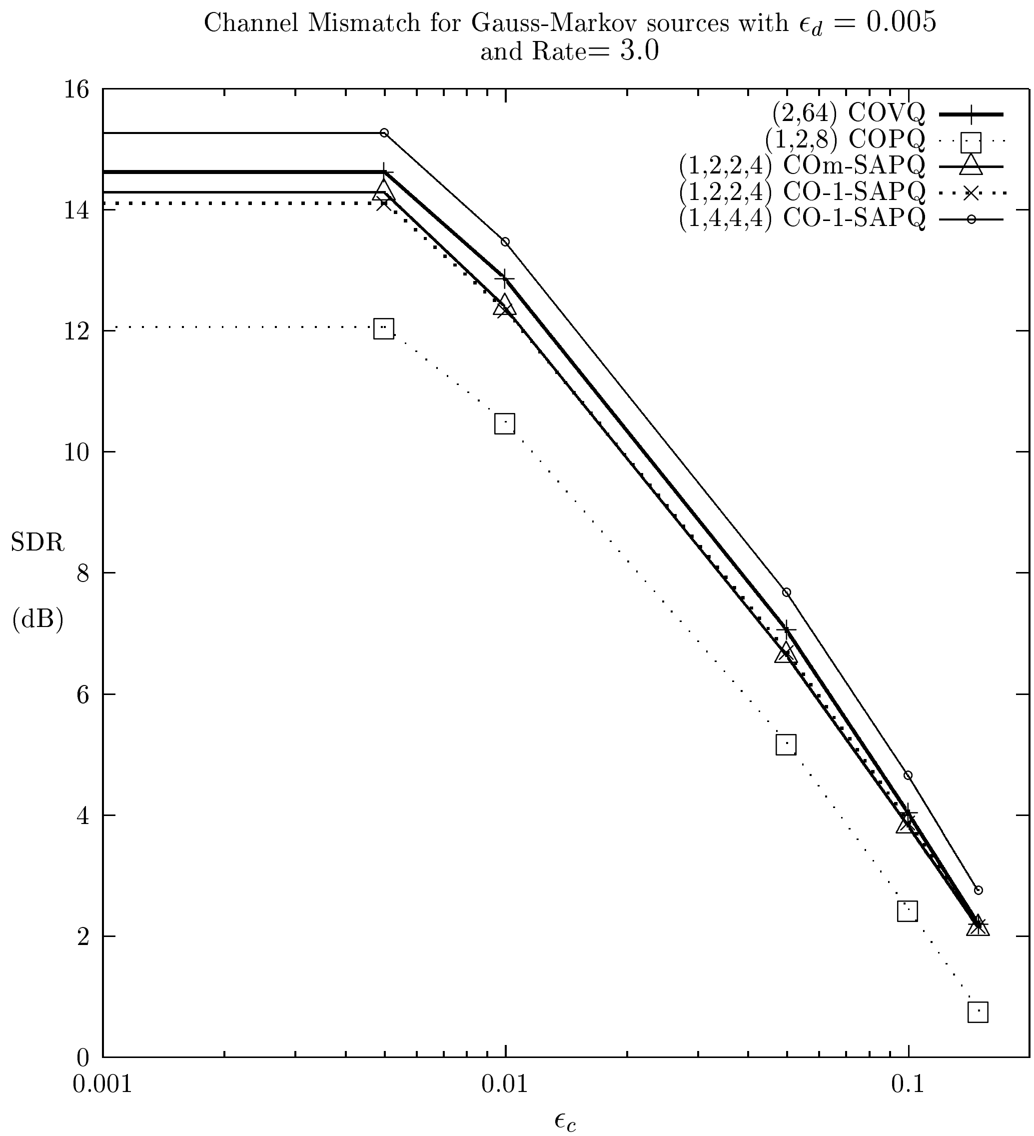

Figure 4.13: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 3.0$, using 200,000 Gauss-Markov testing samples and a simmulated BSC cross over probability ϵ_c

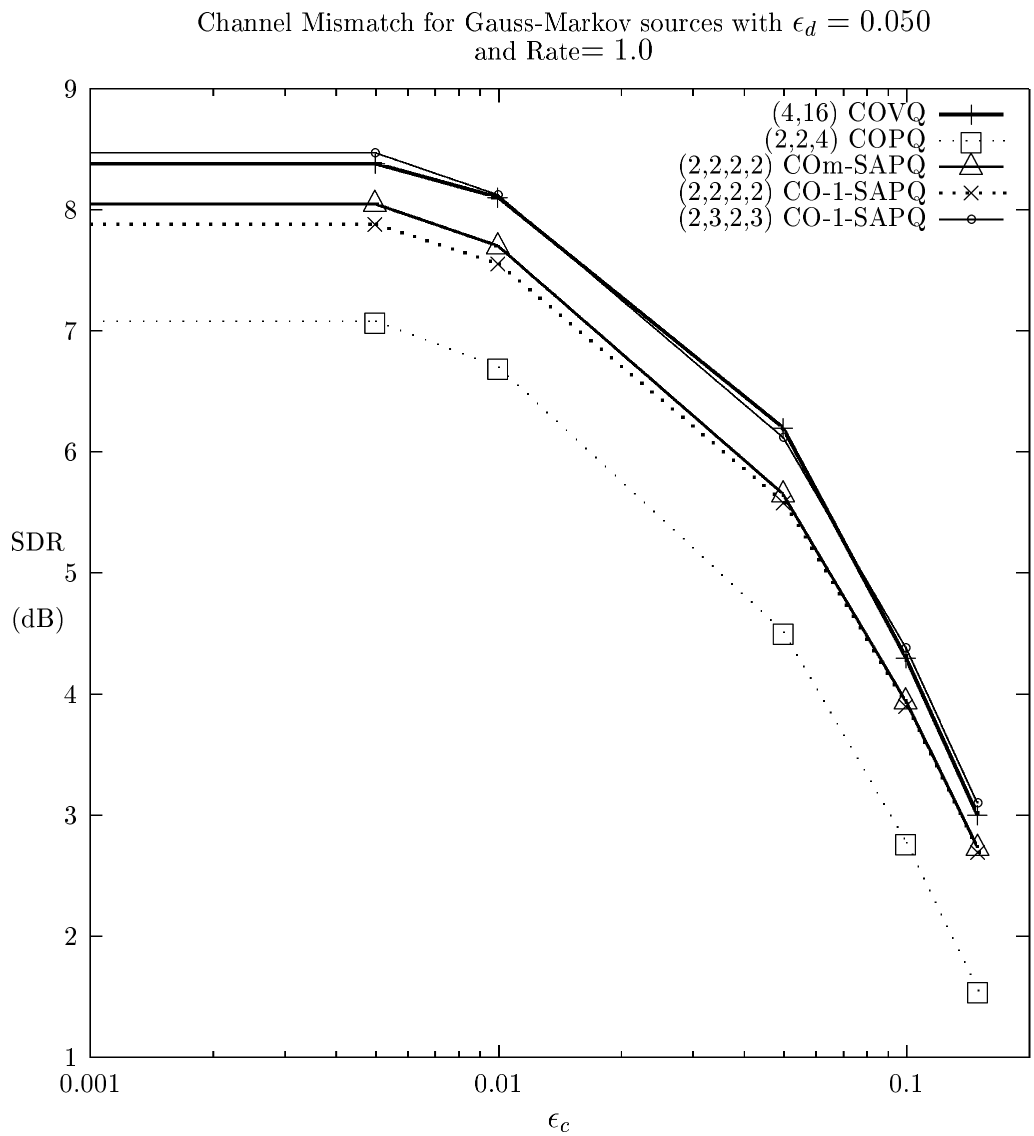

Figure 4.14: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 1.0$, using 200,000 Gauss-Markov testing samples and a simmulated BSC cross over probability ϵ_c

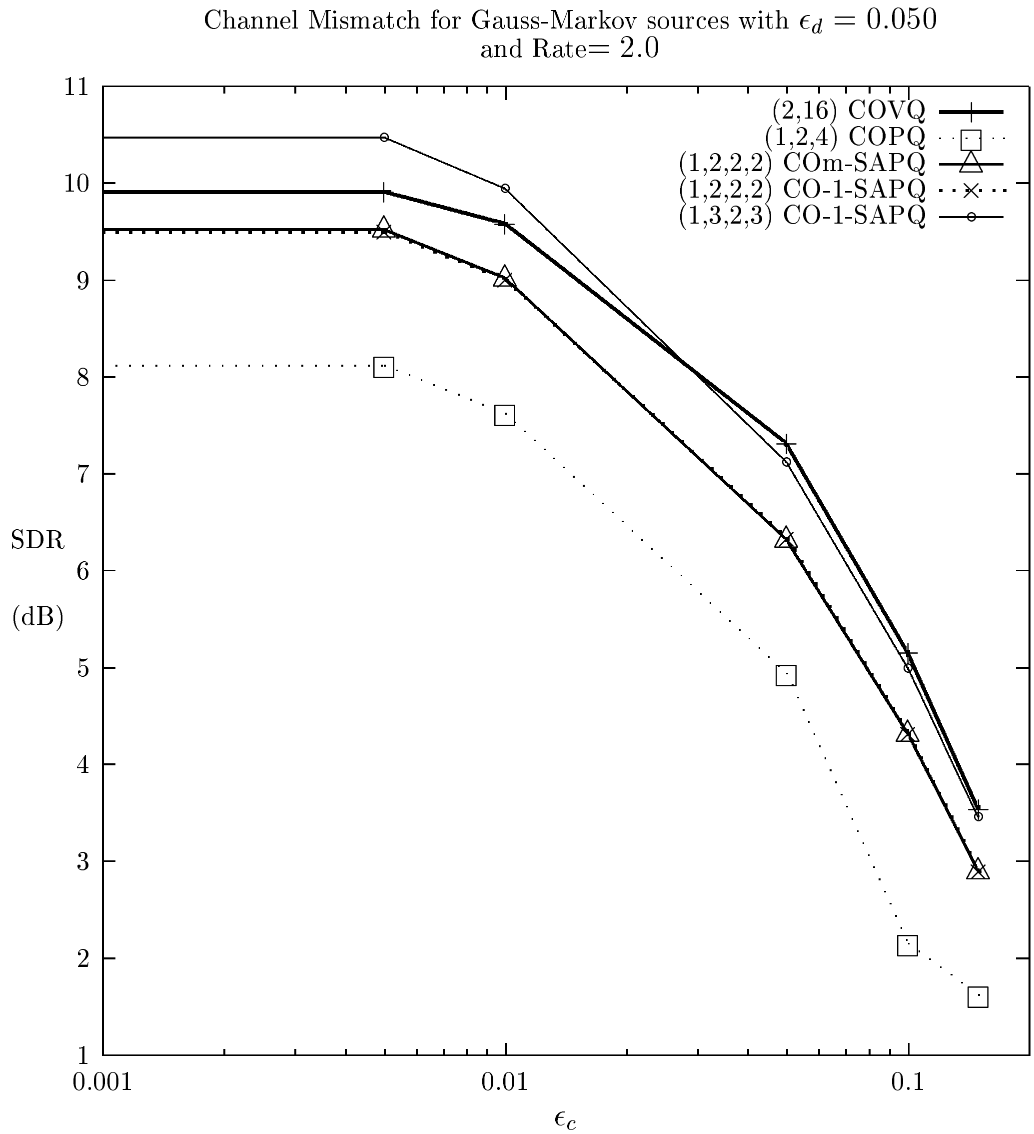

Figure 4.15: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 2.0$, using 200,000 Gauss-Markov testing samples and a simmulated BSC cross over probability ϵ_c

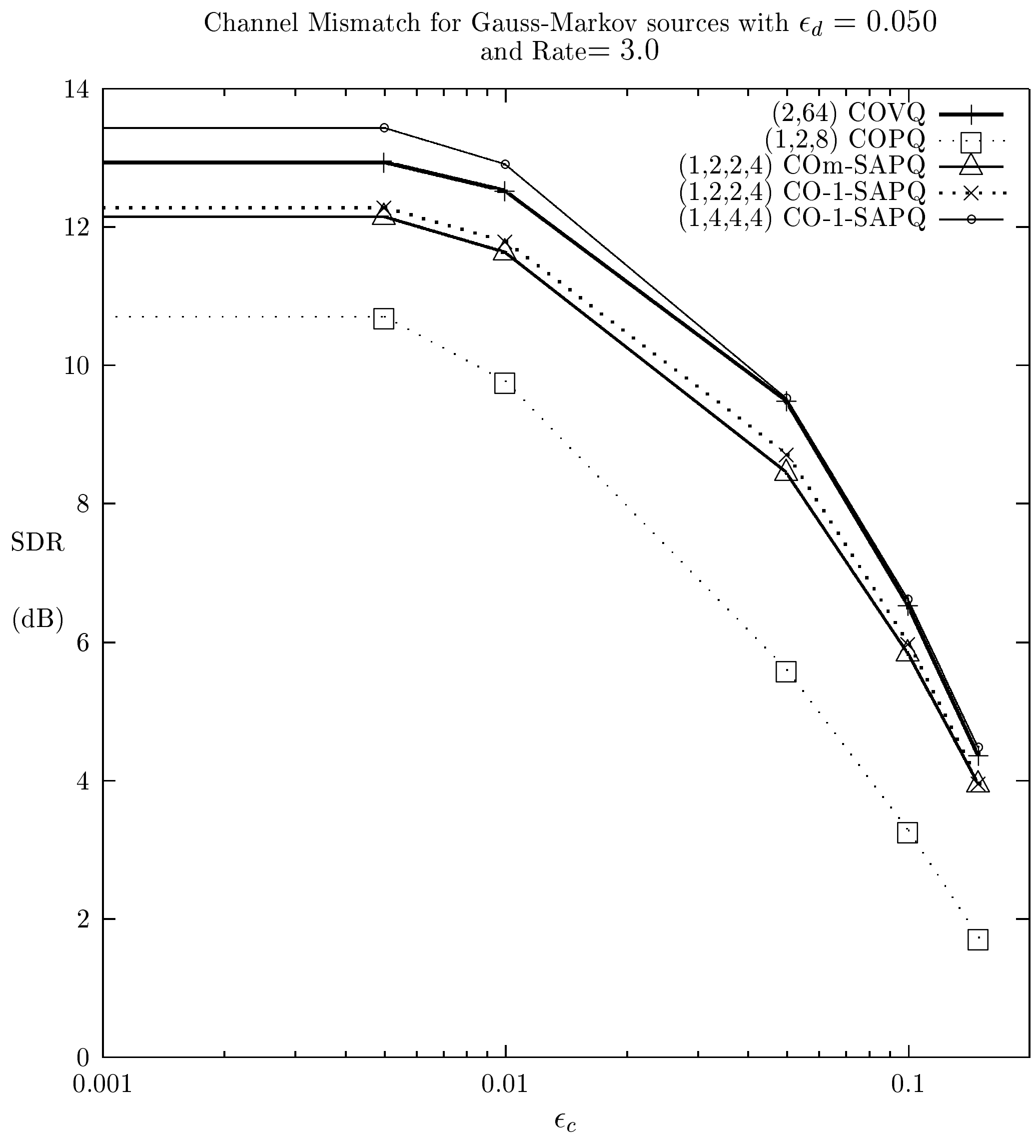

Figure 4.16: Graph of SDR (dB) performances comparing the (k,N) COVQ, the (k,m,N) COPQ, the (k,m,N,η) COm-SAPQ, and the (k,m,N,η) CO1-SAPQ at rate $R = 3.0$, using 200,000 Gauss-Markov testing samples and a simmulated BSC cross over probability ϵ_c

Chapter 5

Conclusion

5.1 Summary of Work

In Chapter 2, we illustrated the necessary conditions for optimality of the vector quantizer and presented a design algorithm for VQ, namely the LBGVQ algorithm, based on iterating over the necessary conditions for optimality. VQ design was then extended to include channel statistics, and the result was a joint source-channel coder called the channel optimized vector quantizer (COVQ). We then described the product quantizer (PQ) and its extension the channel optimized product quantizer (COPQ).

In Chapter 3 we introduced the sample adaptive product quantizer (SAPQ) as designed by Kim and Shroff. SAPQ was studied and numerical results were produced to compare SAPQ performances with that of VQ and PQ. As a result, we saw that there is a definite advantage of using m-SAPQ and 1-SAPQ over PQ and VQ, when the source is memoryless Gaussian, since SAPQ attained performances close to that of VQ with half the encoding complexity and lower storage requirements. For Gauss-Markov sources we saw that for every VQ, we could find a 1-SAPQ that outperforms the VQ, while keeping the encoding complexity equal to that of VQ, and having a storage requirement less then that of VQ.

In Chapter 4, we extended the design of SAPQ to include channel statistics. As a results we designed a channel optimized sample adaptive product quantizer (COS-APQ) which was then tested and compared to COVQ and COPQ. We discovered that for memoryless Gaussian sources COm-SAPQ performed within 0.2 dB to COVQ, and converging to 0.05 dB as the channel noise increased. This performance of COm-SAPQ was achieved with half the encoding complexity and lower storage requirements then that of COVQ of the same rate. When the source is Gauss-Markov we found that CO1-SAPQ outperformed COVQ by 0.2-0.8 dB with the same encoding complexity but lower storage requirements. As the channel got nosier CO1-SAPQ performances converged to within 0.02 dB of COVQ, but still these performances were achieved with less storage requirements then that of COVQ of the same rate.

5.2 Future Work

Future work on this thesis may include the study of other channels such as discrete channels with memory and BPSK-modulated Raleigh fading channels. Improvements in the design of the initial codebook is another possible direction of future research.

Appendix A

Distortion of COm-SAPQ

In the design of a COm-SAPQ, as formulated in (4.2), the distortion of a (k,m,N,η) COm-SAPQ is given by

$$\begin{aligned}
\mathrm{D}_{\scriptscriptstyle COm\text{-}SAPQ} &= \sum_{j=1}^{2^\eta} \sum_{\underline{Z}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{Z}}^{[j]}} \sum_{j'=1}^{2^\eta} \sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(j'|j)\mathrm{P}(\underline{L}|\underline{Z}) \sum_{t=1}^{m} \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2 \mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}} \\
&= \sum_{j=1}^{2^\eta} \sum_{\underline{Z}\in \mathrm{J}_N^m} \int_{\mathbf{S}_{\underline{Z}}^{[j]}} \sum_{j'=1}^{2^\eta} \mathrm{P}(j'|j) \sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(\underline{L}|\underline{Z}) \sum_{t=1}^{m} \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2 \mathrm{p}(\underline{\mathbf{x}})d\underline{\mathbf{x}}.
\end{aligned}$$

The above distortion is simplified by manipulating the term

$$\sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(\underline{L}|\underline{Z}) \sum_{t=1}^{m} \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}_{\mathrm{v}_t(\underline{L})}^{[t,j']} \right\|^2.$$

Note that $\underline{L} = (l_1, \ldots, l_m)$ and $\underline{Z} = (z_1, \ldots, z_m)$ so

$$\begin{aligned}
\mathrm{P}(\underline{L}|\underline{Z}) &= \mathrm{P}(l_1, \ldots, l_m | z_1, \ldots, z_m) \\
&= \mathrm{P}(l_1|z_1) \ldots \mathrm{P}(l_m|z_m) \\
&= \prod_{r=1}^{m} \mathrm{P}(\mathrm{v}_r(\underline{L})|\mathrm{v}_r(\underline{Z}))
\end{aligned}$$

and hence we can manipulate

$$\sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(\underline{L}|\underline{Z}) \left\{ \left\| \mathrm{u}_1(\underline{\mathbf{x}}) - \underline{c}^{[1,j']}_{\mathrm{v}_1(\underline{L})} \right\|^2 + \ldots + \left\| \mathrm{u}_m(\underline{\mathbf{x}}) - \underline{c}^{[m,j']}_{\mathrm{v}_m(\underline{L})} \right\|^2 \right\}$$

by considering the term $\mathbf{T}_s$

$$\mathbf{T}_s = \sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(\underline{L}|\underline{Z}) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2$$

for a fixed $s \in \{1, \ldots, m\}$. In this way we get

$$\begin{aligned}
\mathbf{T}_s &= \sum_{\underline{L}\in \mathrm{J}_N^m} \prod_{r=1}^{m} \mathrm{P}(\mathrm{v}_r(\underline{L})|\mathrm{v}_r(\underline{Z})) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2 \\
&= \sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(\mathrm{v}_s(\underline{L})|\mathrm{v}_s(\underline{Z})) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2 \prod_{r:r\neq s} \mathrm{P}(\mathrm{v}_r(\underline{L})|\mathrm{v}_r(\underline{Z})) \\
&= \sum_{\mathrm{v}_1(\underline{L})=1}^{N} \cdots \sum_{\mathrm{v}_m(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_s(\underline{L})|\mathrm{v}_s(\underline{Z})) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2 \left\{ \prod_{r:r\neq s} \mathrm{P}(\mathrm{v}_r(\underline{L})|\mathrm{v}_r(\underline{Z})) \right\} \\
&= \sum_{\mathrm{v}_s(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_s(\underline{L})|\mathrm{v}_s(\underline{Z})) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2 \left\{ \prod_{r:r\neq s} \sum_{\mathrm{v}_r(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_r(\underline{L})|\mathrm{v}_r(\underline{Z})) \right\}.
\end{aligned}$$

Clearly

$$\sum_{\mathrm{v}_r(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_r(\underline{L})|\mathrm{v}_r(\underline{Z})) = 1$$

so then

$$\sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(\underline{L}|\underline{Z}) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2 = \sum_{\mathrm{v}_s(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_s(\underline{L})|\mathrm{v}_s(\underline{Z})) \left\| \mathrm{u}_s(\underline{\mathbf{x}}) - \underline{c}^{[s,j']}_{\mathrm{v}_s(\underline{L})} \right\|^2$$

and hence we have

$$\sum_{\underline{L}\in \mathrm{J}_N^m} \mathrm{P}(\underline{L}|\underline{Z}) \sum_{t=1}^{m} \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}^{[t,j']}_{\mathrm{v}_t(\underline{L})} \right\|^2 = \sum_{t=1}^{m} \sum_{\mathrm{v}_t(\underline{L})=1}^{N} \mathrm{P}(\mathrm{v}_t(\underline{L})|\mathrm{v}_t(\underline{Z})) \left\| \mathrm{u}_t(\underline{\mathbf{x}}) - \underline{c}^{[t,j']}_{\mathrm{v}_t(\underline{L})} \right\|^2.$$

Bibliography

[1] H. Abut, *Vector Quantization*, IEEE Reprint Collection., IEEE Press, May 1990.

[2] H. Abut, R.M. Gray, and G. Rebolledo, "Vector quantization of speech and speech-like waveforms," *IEEE Acoust. Speech Signal Process.,* ASSP-30, pp 423–435, June 1982.

[3] H. Abut, and S.A. Luse, "Vector quantizers for subband coded waveforms," *Int. Conf. on Acoust., Speech, and Signal Process.,* vol. 1, pp 10.6.1–10.6.4, March 1984.

[4] J.P. Adoul, and P. Mabilleau, "4800 bps RELP vocoder using vector quantization for both filter and residual representation," *Int. Conf. on Acoust., Speech, and Signal Process.,* vol. 1, pp 601, April 1982.

[5] F.Alajaji, N. Phamdo, and T.Fuja, "Channel codes that exploit the residual redundancy in CELP-encoded speech," *IEEE Trans. Speech Audio Processing,* Vol. 4, pp. 325–336, Sept. 1996.

[6] E.Ayanoglu and R. M. Gray, "The design of joint source and channel trellis waveform coders," *IEEE Trans. Inform. Theory,* Vol. IT-33, pp. 855–865, Nov. 1987.

[7] T.M.Cover and J.A.Thomas, *Elements of Information Theory,* A Wiley Series in Telecommunications: John Wiley Sons, Inc. 1991.

[8] J.Cheng and F.Alajaji, “Channel optimized quantization of images over binary channels with memory,”Part of *The 1997 Canadian Workshop on Information Theory* , June 1997.

[9] J.Cheng , *Channel Optimized Quantization of Images over Binary Channels with Memory,* M.Sc.Eng Thesis, Department of Mathematics and Statistics, Queen's University, 1997.

[10] Q.Chen, and T.R. Fischer, “Image coding using robust quantization for noisy digital transmission,” *IEEE Trans. on Image Processing,*.

[11] D.S. Kim and N.B. Shroff , “Quantization based on a novel sample-adaptive product quantizer (SAPQ),” *IEEE Trans. Inform. Theory,* Vol. 45,No.7, pp. 2306–2320, Nov.1999.

[12] D.S. Kim and N.B. Shroff , “Sample-adaptive product quantization: Asymptotic analysis and examples,” *IEEE Trans. on Signal Processing,* Vol. 48,No. 10, pp. 2937–2947, Oct.2000.

[13] M. Effros and P. A. Chou, “Weighted universal bit allocation: Optimal multiple quantization matrix coding,” *Proc. IEEE ICASSP,* pp. 2343–2346, Detroit, May 1995.

[14] M. Effros, P. A. Chou and R. M. Gray, “Universal image compression,” *IEEE Trans. Image Processing,* Vol. 8, pp. 1317–1329, Oct. 1999.

[15] N. Farvardin , “A study of vector quantization for noisy channels,” *IEEE Trans. Inform. Theory,* Vol. 36,N0. 4, pp. 799–808, Jul. 1990.

[16] N. Farvardin and V. Vaishampayan , "On the performance and complexity of channel-optimized vector quantizers," *IEEE Trans. Inform. Theory,* Vol. 37, pp. 155–160, Jan.1991.

[17] N. Farvardin and V. Vaishampayan , "Optimal quantizer design for noisy channels: An approach to combined source-channel coding," *IEEE Trans. Inform. Theory,* Vol. IT-33, pp. 827–838, Nov.1987.

[18] A.A. Gamal, L.A. Hamachandra, I. Shperling, and V. Wei, "Using simulated annealing to design good codes,“ *IEEE Trans. Inform. Theory,* Vol. IT-33, pp. 116–123, Apr.1987.

[19] A. Gercho, and R. M. Gray, *Vector Quantization and Siganl Compression,* Norwell, MA :Kluwer,1992.

[20] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.* vol. 1, pp. 4–29, April 1984.

[21] H. Kumazawa, M. Kasahara, and T. Namekawa, "A Construction of Vector Quantizers for Noisy Channels," *Elect. and Eng. in Japan,* vol. 67-B, pp 39–47, Jan. 1984.

[22] A. Kurtenbach, and P. Wintz , "Quantizing for noisy channels," *IEEE Trans. Commun. Technol,* Vol. COM-17, pp. 291–302, Apr. 1969.

[23] R. Laroia and N. Farvardin, "A structured fixed-rate vector quantizer derived from a variable-length scalar quantizer: Part I – Memoryless sources," *IEEE Trans. Inform. Theory,* Vol. 39,pp. 851–867, May 1993.

[24] R. Laroia and N. Farvardin, “A structured fixed-rate vector quantizer derived from a variable-length scalar quantizer: Part II – Vector sources,” *IEEE Trans. Inform. Theory,* Vol. 39, pp. 868–876, May 1993.

[25] Y.Linde, A. Buzo and R.M. Gray , “An algorithm for vector quantizer design,” *IEEE Trans. Inform. Theory,* Vol. 28, pp. 84–95, Jan.1980.

[26] S.P. Lloyd, “Least squares quantization in PCM,” *IEEE Trans. Info. Theory,* vol. 28, pp 129–137, March 1982.

[27] J. Max, “Quantizing for minimum distortion,” *IRE Trans. Info. Theory,* vol. IT-6, pp 7–12, March 1960.

[28] J.W. Modestino, and D.G. Daut, “Combined source-channel coding of images,” *IEEE Trans. Comm.,* vol. COM-29, pp 1261–1274, Sept. 1981.

[29] A. Ortega and M. Vetterli, “Adaptive scalar quantization without side information,” *IEEE Trans. Image Processing,* Vol. 6, pp. 665–676, May 1997.

[30] N. Phamdo, N. Farvardin, and T. Moriya, “A unified approach to tree-structured and multistage vector quantization for noisy channels,” *IEEE Trans. Info. Theory,* Vol. 39, pp. 835–850, May 1993.

[31] N. Phamdo, F. Alajaji and N. Farvardin , “Quantization of memoryless and Gauss-Markov sources over binary Markov channels,” *IEEE Trans. on Comm.,* Vol. 45, No. 6, pp. 668–675, Jun.1997.

[32] T.S. Rappaport, *Wireless Communications Principles and Practices,* Upper Saddle River, NJ :Prentice-Hall,1996.

[33] K.Sayood and J.C. Borkenhagen, "Use of residual redundancy in the design of joint source/channel coders," *IEEE Trans. Comm.,* vol. 39, pp 838–846, June 1991.

[34] J.Schwartz, "Fast probabilistic algorithms for verification of polynomial identities", *J. Assoc. Comput.*, Vol. 27, pp. 701–717, Mar.1980.

[35] C.E. Shannon, "A Mathematical Theory of Communications," *Bell Syst. Tech. J.,* vol. 27, pp. 379–423 and 623–656, 1948.

[36] C.E. Shannon, "Coding theoroms for a discrete source with a fidelity criterion," *IRE NAt. Conv. Rec.,* pp. 142–163, Mar. 1959.

[37] M. Skoglund and P. Hedelin, "Hadamard-based soft-decoding for vector quantization over noisy channels," *IEEE Trans. Inform. Theory*, Vol. 45, pp. 515–532, Mar. 1999.

[38] R.E. Totty and G.C. Clark, "Reconstruction errors in waveform transmission", *IEEE Trans. Inform. Theory,* Vol. IT-13, pp. 336–338, Apr.1967.

Vita

Zahir Raza

Education

Queen's University	M.Eng	Mathematics and Engineering	1999-2002
Queen's University	B.Eng	Mathematics and Engineering	1995-1999

Experience

Sr. Engineer (2001) T-mobile U.S, Bellevue, Washington

Research Assistant (1999-2001) Mathematics and Statistics, Queen's University, Ontario

Teaching Assistant (1998-2001) Mathematics and Statistics, Queen's University, Ontario

Publications

Z. Raza, F. Alajaji, and T. Linder, "Channel optimized sample adaptive product quantization", *Thirty-Ninth Annual Allerton Conference on Communication, Control, and Computing*, Oct. 2001