Channel Optimized Vector Quantization over Communication Channels with Memory and Feedback

by

Timothy Liu

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

Queen's University
Kingston, Ontario, Canada
August 2025

Abstract

This thesis investigates joint source-channel coding schemes (JSCC) for noisy discrete channels with memory and noiseless feedback. While feedback does not increase the capacity of discrete memoryless channels or additive-noise channels with memory, Amanullah and Salehi have shown that feedback-adapted schemes can outperform non-adaptive schemes under the same rate in terms of mean square error distortion or signal-to-noise ratio (SNR). Building on this result, this thesis explores alternative ways in which channel feedback can enhance such adaptive coding methods. We begin by examining channel-matched tree structured vector quantization (CM-TSVQ) and generalize its necessary conditions for optimality when adapted to noiseless feedback. We then prove that these conditions are equivalent to necessary conditions for optimality for the adaptive channel optimized vector quantization (ACOVQ) scheme introduced by Amanullah and Salehi. Leveraging the tree-like structure of ACOVQ, we study pruning and growing tree algorithms for TSVQ and generalize a growing algorithm for ACOVQ. Simulation results demonstrate that under the same average rate constraints, variable-rate ACOVQ outperforms fixed-rate ACOVQ in terms of SNR. Furthermore, in general the performance SNR gap increases with higher

average rates and a more concentrated source distribution.

Acknowledgments

This endeavor would not be possible without the support of my supervisors Professor Fady Alajaji and Professor Tamás Linder. I am deeply grateful for their invaluable feedback, patience, and guidance throughout the past 2 years, and for giving me the opportunity to pursue this degree.

Many of the simulation results in this thesis would not have been obtained without the Center of Advanced Computing (CAC). I would like to thank the CAC for allowing me to use their resources and for their technical assistance.

I would also like to thank friends and colleagues for their encouragement, our academic discussions, and for making my time at Queen's an exciting and fun experience.

Words cannot express my gratitude to my mother, father, and brother for their love and support throughout my academic journey. Your constant encouragements and prayers have sustained me throughout the ups and downs along the way. I could not reach this point without you.

Above all, I want to thank God for His never-ending love and for giving me the strength and perseverance to finish this work.

Contents

Chapter 1

Introduction

1.1 Joint Source-Channel Coding

A fundamental problem in communication engineering is the reliable transmission of information — such as text, speech, or images — over noisy communication channels, such as satellite or wireless links. In the 1940s, Claude E. Shannon pioneered the foundations of Information Theory in his paper, “A Mathematical Theory of Communication” [30], by introducing mathematical models for data sources and communication channels. Shannon modeled the source as a stochastic process and mathematically quantified the intrinsic information the source contained as “source entropy,” and quantified the maximum achievable rate at which information can be reliably transferred over a channel as its “channel capacity.” Two cornerstone theorems emerged from this work: the source coding theorem and the noisy-channel coding theorem. The source coding theorem states that it is possible for a source to be losslessly compressed at any rate above its entropy; compression below this limit results in information loss. The channel coding theorem states that it is possible for a source to be reliably transmitted over a noisy channel at a rate below the channel's capacity and conversely that any attempt to transmit a message at a rate above the channel's capacity will lead to completely unreliable transmissions. Together, the source channel and the noisy-channel coding theorems support what is know as Shannon's source-channel separation principle, which state that source compression and channel coding can be done independently without loss of optimality — provided sufficient delay and computational resources are available.

The separation principle supports the optimality of tandem coding, a coding scheme comprising of two independent and separately designed components: the source codes and channel codes. The source codes removes statistical redundancies in the source. This however makes the source's encoded representation susceptible to channel noise. The channel codes then adds controlled redundancies to the output of the source encoder, to protect it from channel errors. Tandem coding provides flexibility in designing source and channel codes as they can be interchangeably replaced without affecting reliability. However, this scheme's reliability depends on source and channel codes with arbitrarily large block lengths. Thus, optimal performance is only guaranteed with infinite delay and computational resources. This can make tandem encoding infeasible to implement in practical situations. As a result, joint source-channel coding (JSCC) has emerged as an alternative approach that jointly designs source and channel encoders and decoders — or combines each pair into a single operation — to achieve better performance under practical constraints.

1.2 Literature Review

Various JSCC schemes have been introduced in literature in the past decades. One of the earliest and well-known JSCC techniques is channel optimized vector quantization (COVQ), which jointly designs a vector quantizer and channel code to minimize the expected distortion at the decoder [11]. However, a drawback of COVQ is scalability: as the codebook size increases, the search complexity and storage requirements increase, making the COVQ suffer from complexity constraints at high rates [13].

To address this drawback, channel optimized sample adaptive product quantizers (COSAPQ) [24, 25], channel matched tree structured vector quantization (CM-TSVQ) [21, 23], and channel matched multi-stage vector quantization (CM-MSVQ) [21, 23] were proposed as alternative schemes to COVQ aimed to reduce the complexity of COVQ at the cost of acceptable reductions in performance. COSAPQ reduces the complexity of COVQ by taking a high dimensional source and quantizing subvectors of the source with multiple, lower dimensional COVQs. The COSAPQ codebook would then be the cartesian product of these lower dimensional COVQ codebooks. Simulation results show that COSAPQ performs similarly to COVQ (within 0.2-0.8 dB) despite having an encoding complexity half of COVQ [24]. CM-TSVQ and CM-MSVQ are multi-stage successive refinement quantization schemes. An initial approximation is made in the first stage, with subsequent stages refining the residual *expected* error [21, 23]. CM-TSVQ reduces the encoding complexity by imposing a tree structure in its codebook. CM-MSVQ is a more constricted scheme: the scheme is equivalent to CM-TSVQ with the added constraint that all codebooks for a given stage are the same. In both cases, the resulting designs are not optimal and in general perform worse than a full-search COVQ. Simulation results in [21, 23] show that CM-TSVQ outperforms CM-MSVQ (under the same dimension, rate, and number of stages) in noiseless channels and that the CM-MSVQ outperforms CM-TSVQ in channels with high noise. This suggests that CM-TSVQ is more prone to channel noise as a single decoding error can propagate into further decoding errors for subsequent stages.

Another JSCC is maximum a posteriori (MAP) decoding. In MAP decoding JSCC schemes, channel and source properties are given to the decoder. The decoder observes a sequence of channel outputs and determines the most probable source sequence. Although the MAP detection can be computationally expensive to implement, in some cases the MAP decoder can be simplified down to simple rules such as "believe what you see" or "guess zero (or one) regardless of what you see." In [4], the authors derive necessary and sufficient conditions under which these simple rules are true for a binary symmetric Markov source over a Polya contagion channel with memory, introduced in [3]. Simulation results in [4] show that the performance of MAP detection improves with higher source redundancy and channel noise correlation. In [29], MAP decoding for non-binary noise discrete channels with finite queue based correlated noise was studied and conditions were derived for which a sequential MAP decoder, with large delay, can be reduced to an instantaneous symbol-by-symbol MAP decoder.

One challenge in analyzing JSCC systems — particular those using COVQ — is the difficulty in finding an analytical expression for expected distortion, due to the nonuniform transition probability between channel inputs and outputs. In [34], a tandem VQ and channel coding system is considered with random index assignment. By taking the expected distortion over all possible index assignments, the expected distortion of a tandem VQ and channel coding system is decomposed into three components: distortion from the quantizer, from the source variance, and from the "scatter factor" of the codebook. Necessary conditions for the optimality of jointly designed VQs and channel coders under random index assignment are derived in [34]. Also a high-rate analysis shows a gain of 4.77 dB using the joint source-channel scheme over the tandem scheme for a two-dimensional Gaussian source.

The study of channels with noiseless feedback is motivated by the fact that such models capture realistic communication scenarios. For example, uplink communication between a mobile cell user and a base tower can be paired with a downlink feedback channel. The base tower is assumed to have significantly more transmission power and resources, making the assumption of noiseless downlink transmission reasonable, since strong error-correcting codes and/or higher transmission power can be used. One of the earliest results on the effects of feedback on channel capacity was derived by Shannon, who proved that the presence of feedback, even if noiseless, does not increase the capacity of a discrete memoryless channel (DMC) [31]. This result was then extended in [1] to show that feedback does not increase the capacity of discrete channels with additive random noise in the most general case. Note that the Polya contagion channel, which will be considered for the simulations of this thesis, falls into this class of channels with memory. But under input cost constraints, feedback can increase capacity for both discrete additive noise channels [32] and additive Gaussian noise channels [10]. It can also be used to increase the channel's zero-error capacity [31]. Note however that channel capacity provides a theoretical limit on the performance of the best error correcting codes under sufficiently long block lengths. In practice, coding schemes use finite-block lengths. Thus even if feedback cannot increase capacity for certain channels, it can still yield significant performance improvements for finite coding lengths. Indeed, results in [5] show that JSCC schemes that use feedback perform better than schemes that do not over the memoryless binary symmetric channel. Simulation results in [26] show that the adaptive scheme introduced in [5] outperforms channel optimized scalar quantization without feedback when transmitting over the Polya contagion channel with a noiseless feedback link. Although channel memory is generally seen as an unfavorable condition, since currently deployed error-correcting codes cannot properly handle long error bursts that typically occur in fading channels, these simulations show that the average distortion decreases the more correlated the channel noise is. This indicates that the adaptive scheme effectively exploits channel memory.

1.3 Thesis Contributions

The contributions of this thesis are as follows:

1. The necessary conditions for optimality of the CM-TSVQ scheme of [21] are generalized to adapt to noiseless feedback from a discrete channel with memory. We then show that these necessary conditions are equivalent to those of the adaptive COVQ scheme in [5] and show that, with equivalent initializations, the performance of the adaptive COVQ scheme in [5] match the performance of

the derived adaptive CM-TSVQ scheme. This result helps unify two previously distinct JSCC approaches.

2. We introduce a variable-rate version of the ACOVQ, by generalizing a tree pruning algorithm introduced in [27] to the ACOVQ scheme. Simulations are then systematically carried to compare the performance of the two schemes under the same average rate constraints. The simulation code is available at: `https://github.com/timothyliutl/VR-ACOVQ.git`. We show a significant increase in performance for the variable-rate ACOVQ compared to its fixed-rate counterpart comes, albeit with an increased encoding complexity. We also show that as the channel noise becomes more correlated, the performance for variable-rate ACOVQ increases, indicating that the scheme is able to effectively exploit channel memory.

1.4 Thesis Outline

The remainder of the thesis is organized as follows:

- Chapter 2 introduces various discrete channel models with memory. We then study vector quantization (VQ) and the necessary conditions for optimal VQs as well as an algorithm for designing locally optimal VQs. Afterwards, we introduce various discrete noise communication channel models and JSCC schemes — COVQ, CM-TSVQ, and ACOVQ — as well as their respective necessary conditions for optimality.

- Chapter 3 presents generalized necessary conditions for optimality for the CM-TSVQ when adapted to noiseless feedback. We then prove that the conditions are equivalent to those of ACOVQ and validate this result through simulation.
- In Chapter 4, we study various variable rate tree structured schemes for VQs. We then extend the pruning method introduced in [27] for the design of variable rate ACOVQ and compare the encoding complexity and performance of fixed-rate and variable rate ACOVQ.
- Chapter 5 concludes the thesis with a summary of main results and a discussion of possible directions for future work.

Chapter 2

Preliminaries

2.1 Communication Channel Models

2.1.1 Discrete Channels

A discrete communication channel can be characterized by input X with finite alphabet $\mathcal{X}$, output Y with finite alphabet $\mathcal{Y}$, and a sequence of i-dimensional transition probabilities

$$\{P(Y^i = y^i | X^i = x^i)\}_{i=1}^{\infty} \tag{2.1}$$

for $i \geq 1$, where $x^i = (x_1, \ldots, x_i) \in \mathcal{X}^i$ and $y^i = (y_1, \ldots, y_i) \in \mathcal{Y}^i$. A discrete channel is called a discrete memoryless channel (DMC) if the following property holds:

$$P(Y^n = y^n | X^n = x^n) = \prod_{i=1}^{n} P(Y_i = y_i | X_i = x_i) \tag{2.2}$$

for all $x^n \in \mathcal{X}^n$ and $y^n \in \mathcal{Y}^n$. This property implies that the transition at time i only depends on the input x_i and does not depend on the previous input and output sequences. A simple example of a discrete memoryless channel is the binary symmetric channel (BSC). In this model we have that $\mathcal{X} = \{0, 1\}, \mathcal{Y} = \{0, 1\}$ and $P(Y = 1|X = 0) = P(Y = 0|X = 1) = \epsilon$, $P(Y = 1|X = 1) = P(Y = 0|X = 0) = 1 - \epsilon$, where $0 \leq \epsilon \leq 1$ is the cross-over probability.

2.1.2 Discrete Channels with Memory

The memoryless assumption allows for a simple transition probability calculations; however, it does not accurately reflect how errors in real-world communication channel often come in bursts (such as in wireless fading channels). Here we describe a few widely used discrete channel models with memory.

The Gilbert-Elliot channel (GEC) extends the BSC by incorporating a time varying cross-over probability governed by a first-order Markov process. The Markov process contains 2 states: a “good” state, in which ϵ is low, and a “bad” state, in which ϵ is high. This model does account for the burst-like behaviour of real-world models. However, the resulting noise process is a hidden Markov process and hence hard to analyze since its transition distribution and entropy rate (and resulting GEC capacity) do not admit simple analytical expressions.

Another model is the binary additive Markov noise channel model whose output Y_i at time $i \geq 1$ is given by $Y_i = X_i \oplus Z_i$, where $\oplus$ denotes addition modulo-2 and $\{Z_i\}_{i=1}^{\infty}$ is a binary (i.e., with alphabet $\mathcal{Z} = \{0, 1\}$) Markov noise process of memory order M. While this model has a closed form capacity formula, its noise transition matrix, has size $2^M \times 2^M$, which makes it computationally infeasible for large memory values M.

The Polya contagion model with finite memory, proposed in [3], is a specific case of the binary additive Markov noise channel model. Here we have that for Markov memory M, cross-over probability ϵ, and memory δ,

$$P(Z_i = 1 | Z_{i-1} = e_{i-1}, \ldots, Z_1 = e_1) \tag{2.3}$$

$$= P(Z_i = 1 | Z_{i-1} = e_{i-1}, \ldots, Z_{i-M} = e_{i-M}) \tag{2.4}$$

$$= \begin{cases} \frac{\epsilon + \sum_{j=i-M}^{i-1} e_j \delta}{1 + M\delta}, & i \geq M+1 \\ \frac{\epsilon + \sum_{j=1}^{i-1} e_j \delta}{1 + (i-1)\delta}, & i \leq M \end{cases} \tag{2.5}$$

where $e_i \in \{0, 1\}$. In this model, Z_i only depends on the sum of the previous M noise samples. Here δ denotes the correlation in $\{Z_i\}_{i=1}^{\infty}$ — the higher δ is the more likely the noise will occur in bursts. Note that when there is no noise correlation ($\delta = 0$) the model reduces down to the memoryless BSC with crossover probability ϵ.

Let $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_n)$ be the input and output block, respectively, such that $\mathbf{Z} = \mathbf{X} \oplus \mathbf{Y} = (X_1 \oplus Y_1, \ldots, X_n \oplus Y_n)$. For cases where $n \leq M$, the channel block transition probability is [3]

$$P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) = P(\mathbf{Z} = (e_1, \ldots, e_n)) \tag{2.6}$$

$$= P(Z_1 = e_1) \prod_{i=2}^{n} P(Z_n = e_n | Z_{n-1} = e_{n-1}, \ldots, Z_1 = e_1) \tag{2.7}$$

$$= \frac{\prod_{i=0}^{s_n - 1} (\epsilon + i\delta) \prod_{j=0}^{n - s_n - 1} (1 - \epsilon + j\delta)}{\prod_{l=1}^{n-1} (1 + l\delta)}, \tag{2.8}$$

where $e_i = x_i \oplus y_i$ and $s_n = e_1 + e_2 + \ldots + e_n$. For cases where $n \geq M+1$ the channel block transition probability is [3]

$$P(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) = L \prod_{i=M+1}^{n} \left(\frac{\epsilon + \tilde{s}_{i-1}\delta}{1 + M\delta} \right)^{e_i} \left(\frac{1 - \epsilon + (M - \tilde{s}_{i-1})\delta}{1 + M\delta} \right)^{1-e_i} \tag{2.9}$$

where $\tilde{s}_{i-1} = e_{i-M} + \ldots + e_{i-1} + e_{i-1}$, and

$$L = \frac{\prod_{i=0}^{s_M - 1} (\epsilon + i\delta) \prod_{j=0}^{n - s_M - 1} (1 - \epsilon + j\delta)}{\prod_{l=1}^{M-1} (1 + l\delta)}. \tag{2.10}$$

Throughout this thesis, only Polya contagion channels with Markov memory of 1 (*i.e.*, $M = 1$) will be considered. For $M = 1$, the one-step transition probability matrix of $\{Z_i\}_{i=1}^{\infty}$ is given by

$$\begin{bmatrix} P(Z_i = 0 | Z_{i-1} = 0) & P(Z_i = 1 | Z_{i-1} = 0) \\ P(Z_i = 0 | Z_{i-1} = 1) & P(Z_i = 1 | Z_{i-1} = 1) \end{bmatrix} = \frac{1}{1 + \delta} \begin{bmatrix} 1 - \epsilon + \delta & \epsilon \\ 1 - \epsilon & \epsilon + \delta \end{bmatrix}. \tag{2.11}$$

The capacity of the Polya contagion channel with Markov memory of one (i.e., $M = 1$) is shown in [3] to be given by

$$\begin{aligned} C &= 1 - H(Z_2 | Z_1) & (2.12) \\ &= 1 - \left(\epsilon \times h_b \left(\frac{\epsilon + \delta}{1 + \delta} \right) + (1 - \epsilon) \times h_b \left(\frac{\epsilon}{1 + \delta} \right) \right) & (2.13) \end{aligned}$$

where $h_b(p) = -p \log(p) - (1 - p) \log(1 - p)$ is the binary entropy function and $H(Z_2 | Z_1) = -\sum_{a \in \mathcal{Z}} \sum_{b \in \mathcal{Z}} P(Z_1 = a, Z_2 = b) \log_2 P(Z_2 = b | Z_1 = a)$ is the conditional entropy of Z_2 given Z_1.

2.2 Source and Channel Encoding

The goal of source encoding is to represent source samples using fewer bits than their original representation, before transmission through a channel. There are two types of source encoding: lossless encoding and lossy encoding. In lossless encoding, statistical redundancies embodied by the source entropy are removed, allowing the original sample to be perfectly recovered from its compressed representation described at a rate above entropy. This redundancy can come from memory in the source and from the non-uniformity of the source distribution and can be mathematically quantified. In lossy encoding, both non-statistical and statistical redundancies are reduced so that the compressed representation is described at a rate below entropy. In this case, the original sample cannot be perfectly recovered from the compressed representation, but this process typically results in smaller compression rates compared to lossless encoding (and hence higher compression efficiency).

We will first mathematically quantify the redundancies in lossless encoding. Shannon quantified the information gained when observing an random variable as *entropy*. For a random variable X with a discrete finite alphabet $\mathcal{X}$, the entropy of X is defined as

$$H(X) := -\sum_{x \in \mathcal{X}} P(X = x) \cdot \log_2 P(X = x) \quad \text{(bits)}. \tag{2.14}$$

Note that entropy is maximized when the distribution of X is uniform. For a stochastic process $\{X_i\}_{i=1}^{\infty}$, the *entropy rate* is defined as

$$H(\mathcal{X}) := \lim_{n\to\infty} \frac{1}{n} H(X_n, \ldots, X_1), \tag{2.15}$$

where

$$H(X_n, \ldots, X_1) := \\ -\sum_{(x_1,\ldots,x_n)\in\mathcal{X}^n} P(X_n = x_n, \ldots, X_1 = x_1) \cdot \log_2 P(X_n = x_n, \ldots, X_1 = x_1). \tag{2.16}$$

When $\{X\}_{i=1}^{\infty}$ is an independently and identically distributed process, $H(\mathcal{X}) = H(X_1)$. A uniform and memoryless source will achieve the highest entropy rate of $H(\mathcal{X}) = \log_2 |\mathcal{X}|$, where $|\mathcal{X}|$ denotes the number of elements in $\mathcal{X}$. According to Shannon's source coding theorem, a stationary ergodic source $\{X\}_{i=1}^{\infty}$ can be encoded with a rate of $H(\mathcal{X})$ bits per source symbol losslessly for a sufficiently long block-length. Further, the theorem states that any block codes with a rate less than $H(\mathcal{X})$ bits per source symbol cannot encode the source with an arbitrarily small probability of decoding error. The redundancy can then be quantified as the difference between $\log_2 |\mathcal{X}|$, the rate for uncompressed source representation, and $H(\mathcal{X})$. We then have the following,

$$\rho_t := \log_2 |\mathcal{X}| - H(\mathcal{X}) \tag{2.17}$$

$$\rho_d := \log_2 |\mathcal{X}| - H(X_1) \tag{2.18}$$

$$\rho_m := H(X_1) - H(\mathcal{X}), \tag{2.19}$$

where ρ_t represents the total redundancy, ρ_d represents the redundancy due to the nonuniformity of the distribution of X, and ρ_m represents the redundancy due to the statistical memory of the source. Note that $\rho_t = \rho_d + \rho_m$.

2.3 Joint Source Channel Encoding

In his renowned paper [30], Shannon, proved that channel coding does not depend on the source it is transmitting, as long as the rate of the channel code transmitted does not exceed the channel's capacity. Therefore, channel codes can be created independently from source codes and still achieve asymptotically error-free performances. However, this proof assumes block-lengths of infinite lengths in order for this to be achieved, introducing large delay and requiring large processing power to encode and decode. In cases where modest amounts of distortion is allowed and minimal delay is needed, it is favorable to combine channel and source encoding in a low-delay joint source encoder. We next examine different lossy joint source-coding schemes for effective quantization and transmission of real-valued sources over noisy channels. Before doing so, we first describe the vector quantization lossy source coding method.

2.4 Vector Quantization

Conceptually, a vector quantizer (VQ) is a function that takes in a vector of source samples and outputs the nearest approximation from a finite set of predetermined vectors. This finite set is called the codebook and the elements in this set are called reproduction codevectors. In general, the input space is partitioned into encoding regions, in which values within the same encoding region share the same vector quantizer output. Let Q represent a vector quantizer of dimension k and size N. Then Q is the function,

$$Q : \mathbb{R}^k \to \mathcal{C}, \tag{2.20}$$

where $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_N\} \subseteq \mathbb{R}^k$, which assumes that the support of the source vector $\mathbf{u}$ is $\mathbb{R}^k$. Here $\mathcal{C}$ is called the codebook and $\mathbf{c}_i \in \mathbb{R}^k$, for $i \in \{0, \ldots, N-1\}$, are called codevectors. Every codevector $\mathbf{c}_i$ has a corresponding encoding region given by

$$S_i := \left\{\mathbf{u} \in \mathbb{R}^k : Q(\mathbf{u}) = \mathbf{c}_i\right\}, \tag{2.21}$$

such that $\bigcup_{i=0}^{N-1} S_i = \mathbb{R}^k$ and $S_i \cap S_j = \emptyset$ for $i \neq j \in \{0, \ldots, N-1\}$. We will denote the set of encoding regions or partition by $\mathcal{S} = \{S_0, \ldots, S_{N-1}\}$. The quantizer Q can be represented as the composition of two functions: the encoder and decoder, denoted by $\mathcal{E}$ and $\mathcal{D}$, respectively. The encoder and decoder are defined as:

$$\mathcal{E} : \mathbb{R}^k \to \{0, \ldots, N-1\} \quad \text{s.t. } \mathcal{E}(\mathbf{u}) = i \iff \mathbf{u} \in S_i \tag{2.22}$$

$$\mathcal{D} : \{0, \ldots, N-1\} \to \mathcal{C} \quad \text{s.t. } \mathcal{D}(i) = \mathbf{c}_i, \tag{2.23}$$

such that $Q(\mathbf{u}) = \mathcal{D} \circ \mathcal{E}(\mathbf{u})$, for all $\mathbf{u} \in \mathbb{R}^k$. The encoder can be defined by the partition $\mathcal{S}$ and the decoder likewise can be defined by the codebook $\mathcal{C}$. In general, the encoding regions and codevectors are selected to minimize the expected distortion of Q defined as

$$E[d(\mathbf{U}, Q(\mathbf{U}))], \tag{2.24}$$

where $d : \mathbb{R}^k \times \mathbb{R}^k \to [0, \infty)$ is called a distortion measure. For this thesis, the *square error distortion* measure will be used, which is given by the squared Euclidean distance $\|\cdot\|^2$:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2 \tag{2.25}$$

$$= \sum_{i=1}^{k} (x_i - y_i)^2, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^k, \tag{2.26}$$

where x_i, y_i are the ith component of vectors $\mathbf{x}, \mathbf{y}$, respectively. For a codebook $\mathcal{C}$ and partition $\mathcal{P}$, the expected distortion of Q is given by

$$E[d(\mathbf{U}, Q(\mathbf{U}))] = \sum_{i=0}^{N-1} \int_{S_i} f_{\mathbf{U}}(\mathbf{u}) d(\mathbf{u}, \mathbf{c}_i) d\mathbf{u} \tag{2.27}$$

$$= \sum_{i=0}^{N-1} \int_{S_i} f_{\mathbf{U}}(\mathbf{u}) \|\mathbf{u} - \mathbf{c}_i\|^2 \, d\mathbf{u}, \tag{2.28}$$

where $f_{\mathbf{U}}(\mathbf{u})$ is the probability density function of the source. Let $\mathcal{Q}_N$ denote the family of all $N-$level quantizers (quantizers where $|\mathcal{C}| = N$). An N-level quantizer $Q^* \in \mathcal{Q}_N$ is called an *optimal quantizer* if it satisfies

$$Q^* = \arg \min_{Q \in \mathcal{Q}_n} E[d(\mathbf{U}, Q(\mathbf{U}))]. \tag{2.29}$$

A practical challenge is determining an optimal quantizer - that is, determining the optimal codebook and partition that define it. In general, the interdependence between the codebook (decoder) and the partition (encoder) makes simultaneous optimization difficult. As a result, it is advantageous to decompose the problem and alternately optimize the encoder or decoder while holding the other fixed. There are two necessary conditions for a quantizer to be optimal called the *nearest neighbor condition* and the *centroid condition*. The centroid condition states that for a fixed partition $\mathcal{S} = \{S_0, \ldots, S_{N-1}\}$, the optimal codevectors (that minimize expected distortion) satisfy [15]

$$\mathbf{c}_i = \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^k} E[d(\mathbf{U}, \boldsymbol{\omega}) | \mathbf{U} \in S_i], \quad i \in \{0, \ldots, N-1\}. \tag{2.30}$$

In other words, the decoder of an optimal quantizer must assign each region S_i a codevector $\mathbf{c}_i \in \mathbb{R}^k$ that minimizes the expected distortion for the source $\mathbf{U}$ conditioned on $\mathbf{U} \in S_i$. Under mean squared distortion, the centroid condition reduces down to

$$\mathbf{c}_i = E[\mathbf{U} | \mathbf{U} \in S_i] \tag{2.31}$$

$$= \frac{\int_{S_i} \mathbf{u} f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u}}{\int_{S_i} f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u}}, \quad i \in \{0, \ldots, N-1\}. \tag{2.32}$$

The nearest neighbor condition states that for a fixed codebook $\mathcal{C} = \{\mathbf{c}_0, \ldots, \mathbf{c}_{N-1}\}$ the optimal encoding regions satisfy [15]

$$S_i = \left\{\mathbf{u} \in \mathbb{R}^k : d(\mathbf{u}, \mathbf{c}_i) \leq d(\mathbf{u}, \mathbf{c}_j), \quad j \in \{0, \ldots, N-1\}\right\}, \\ i \in \{0, \ldots, N-1\}. \tag{2.33}$$

In other words, the encoder of an optimal quantizer maps each source vector to the codeword that minimizes distortion. Equivalently the optimal encoder $\mathcal{E}^*$ is given by

$$\mathcal{E}^*(\mathbf{u}) = i \iff d(\mathbf{u}, \mathbf{c}_i) \leq d(\mathbf{u}, \mathbf{c}_j), \quad j \in \{0, \ldots, N-1\}, \tag{2.34}$$

for fixed codebook $\mathcal{C}$, where $i \in \{0, 1, \ldots, N-1\}$. Note that for a given source vector, the indices that correspond to the minimum distortion may not be unique (i.e., a source vector may be equidistant from 2 different centroids). In these cases, the source vector can be encoded arbitrarily from the minima without changing the expected distortion of the quantizer [15].

While these conditions are defined for a continuous random variable with a density function, in practice, the density function of the source is not always available. In these cases, a training set can be used with slight modifications to the nearest neighbor and centroid conditions. Let $A = (\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_\eta) \subseteq \mathbb{R}^k$ be a training set of size η that are independenty and identically drawn from the random variable $\mathbf{U}$. Here we approximate the distribution of $\mathbf{U}$ with discrete random variable with probability $\frac{1}{\eta}$ for each value in the sequence A. Thus, the nearest neighbor condition can be expressed as

$$S_i = \{\mathbf{a} \in A : d(\mathbf{a}, \mathbf{c}_i) \leq d(\mathbf{a}, \mathbf{c}_j), \quad j \in \{0, \ldots, N-1\}\}, \quad i \in \{0, \ldots, N-1\}. \tag{2.35}$$

The centroid condition can also be modified by replacing the integral with summations as follows:

$$E[d(\mathbf{U}, Q(\mathbf{U}))] = \frac{1}{\eta} \sum_{i=0}^{N-1} \sum_{\mathbf{a} \in S_i} \|\mathbf{a} - \mathbf{c}_i\|^2 . \tag{2.36}$$

A vector quantizer that satisfies both the *centroid* and *nearest neighbor* condition is called a *Lloyd-Max quantizer*. Note that every optimal quantizer (i.e., a quantizer that minimizes the distortion for a given N) is a Lloyd-Max quantizer, but a Lloyd-Max quantizer is not necessarily optimal. A natural question that follows is how do we find a quantizer with encoding regions and a codebook that satisfy these necessary optimality conditions. In general, it is difficult to jointly optimize the encoder and decoder; however, if either the encoder or decoder is fixed, the other component can be easily optimized by applying the conditions in (2.30) or (2.33). Given an initial encoder and decoder, a simple algorithm to find a Lloyd-Max quantizer is to successively fix the encoder and optimize the decoder and vise-versa until the expected distortion of the quantizer converges. This algorithm is called the *LBG-Algorithm* [18], which is the generalization of *Lloyd's algorithm* [19] from scalar quantization (when $k = 1$) to vector quantization (when $k > 1$). Details of the LBG-algorithm are shown below.

1. Let $D^{(m)}, Q^{(m)}$ represent the expected distortion and quantizer at the m-th iteration, respectively. Set $m = 0$ and $D^{(m)} = \infty$. Choose an initial $\mathcal{S}^{(m)} = \{S_0^{(m)}, \ldots, S_{N-1}^{(m)}\}$, $\mathcal{C}^{(m)} = \{\mathbf{c}_0^{(m)}, \ldots, \mathbf{c}_{N-1}^{(m)}\}$, and T. Here $\mathcal{S}^{(m)}$, $\mathcal{C}^{(m)}$ represents the m-th iteration of the encoding regions and codebook, respectively.

2. Using the fixed codebook $\mathcal{C}^{(m)}$, for all $i \in \{0, \ldots, N-1\}$, set $S_i^{(m+1)} = \{\mathbf{u} \in \mathbb{R}^k : d(\mathbf{u}, \mathbf{c}_i^{(m)}) \leq d(\mathbf{u}, \mathbf{c}_j^{(m)}), \quad j \in \{0, \ldots, N-1\}\}$.

3. Using the fixed encoding regions $\mathcal{S}^{(m+1)}$, set $\mathbf{c}_i^{(m+1)} = E[\mathbf{U}|\mathbf{U} \in \mathcal{S}_i]$.

4. Using $\mathcal{C}^{(m+1)}$ and $\mathcal{S}^{(m+1)}$, calculate $D^{(m+1)} = E[d(\mathbf{U}, Q^{(m+1)}(\mathbf{U}))]$. If

$$\frac{D^{(m)} - D^{(m+1)}}{D^{(m)}} < T,$$

then return $\mathcal{C}^{(m+1)}$ and $\mathcal{S}^{(m+1)}$. Otherwise, set $m = m + 1$ and go to step 2.

The distortion in the LBG-algorithm is monotonically decreasing for each iteration and eventually converges. The produced quantizer will satisfy the centroid and nearest neighbor conditions of an optimal quantizer, but it is not necessarily globally optimal. However, [14] shows that in cases where the source probability density function is log concave, these conditions become *sufficient* for an optimal quantizer. Therefore, Lloyd-Max quantizers are optimal for distributions such as Gaussian, Laplacian and uniform distributions. Further, the initial choice of the codebook and encoding regions plays a large role in whether the LBG-algorithm converges to a “bad” local minimum. The LBG-algorithm assumes that all source values are encoded without noise. In the presence of channel noise, the performance of the LBG quantizer degrades significantly with increasing noise. However, the nearest neighbor and centroid conditions can be modified to make the quantizer more robust to channel noise.

2.5 Channel Optimized Vector Quantization

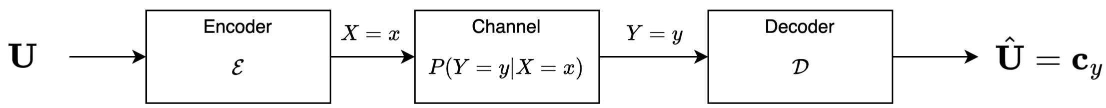

Figure 2.1: A block diagram for a COVQ communication system.

The channel optimized vector quantization (COVQ) scheme, introduced among others in [11] [13], generalizes the nearest neighbor and centroid condition of the LBG-algorithm to account for channel noise. Consider the communication system in Figure 2.1. A COVQ is a VQ that accounts for a noisy channel between the encoder and decoder. In this scheme, knowledge of the channel's block transition probability and memory is assumed. However, when the memory in the channel is not known, an interleaver can be used to effectively turn the channel into a memoryless channel. In [22], it was shown that the channel memory can be exploited to improve the performance of a COVQ and that in general the COVQ outperforms a COVQ with an interleaver.

2.5.1 Optimality Conditions

Similar to a VQ, a COVQ will have a codebook and a set of encoding regions denoted by $\mathcal{C}$ and $\mathcal{S}$, respectively. Let $\mathcal{E}$ and $\mathcal{D}$ represent the encoder and decoder of the

COVQ, respectively, and let $\mathcal{I} = \{0, \ldots, N-1\}$ represent the channel index set. In this system we have that the encoder takes in a source vector $\mathbf{u} \in \mathbb{R}^k$ and outputs an index $x \in \mathcal{I}$. Afterwards the index x is transmitted through a noisy channel and the index $y \in \mathcal{I}$ is received by the decoder. The decoder then takes in the index y and outputs the corresponding value from the codebook $\mathbf{c}_y$. Here and throughout the thesis, we assume that the channel has identical input and output alphabets (i.e., $\mathcal{X} = \mathcal{Y} = \mathcal{I}$). The expected distortion of this communication system is given by

$$D = \int_{\mathbb{R}^k} \sum_{y \in \mathcal{I}} \sum_{x \in \mathcal{I}} P(Y = y | X = x) \left\| \mathbf{u} - \mathbf{c}_y \right\|^2 1_{\{\mathbf{u} \in S_x\}} f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u} \tag{2.37}$$

$$= \sum_{x \in \mathcal{I}} \int_{S_x} \left[\sum_{y \in \mathcal{I}} P(Y = y | X = x) \left\| \mathbf{u} - \mathbf{c}_y \right\|^2 \right] f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u} \tag{2.38}$$

where 1_A is the indicator function of event A. The rate of the COVQ is defined as $R = \frac{1}{k} \log_2 N$ bits per source sample. The term in brackets is called the *modified distortion* and is denoted as

$$d' : \mathbb{R}^k \times \mathcal{I} \to \mathbb{R} \tag{2.39}$$

$$d'(\mathbf{u}, x) = \sum_{y \in \mathcal{I}} P(Y = y | X = x) \left\| \mathbf{u} - \mathbf{c}_y \right\|^2. \tag{2.40}$$

The modified distortion describes the expected distortion for a source vector $\mathbf{u}$ when encoded with a given index x. We can see in (2.38) that the distortion expression is analogous to that for VQ in (2.27) with distortion being replaced with the modified distortion. Note that the modified distortion is a generalization of the distortion in the VQ case: when the channel is noiseless, it reduces to the square error distortion. The generalized nearest neighbor condition for the COVQ states that for a fixed codebook $\mathcal{C}$ an optimal COVQ will have a partition that satisfy [11]

$$S_x = \{\mathbf{u} \in \mathbb{R}^k : d'(\mathbf{u}, x) \leq d'(\mathbf{u}, x'), \quad x' \in \mathcal{I}\}, \quad x \in \mathcal{I}. \tag{2.41}$$

The proof for this statement is as follows. Let $\mathcal{Q}$ denote the family of N-level, k-dimensional quantizers. Let $Q \in \mathcal{Q}$ be an arbitrary quantizer with partition $\mathcal{S}$ and codebook $\mathcal{C}$. We then have that

$$\begin{aligned}
E[d(\mathbf{U}, Q(\mathbf{U}))] &= \sum_{x\in\mathcal{I}} \int_{S_x} \left[\sum_{y\in\mathcal{I}} P(Y = y|X = x) \left\| \mathbf{u} - \mathbf{c}_y \right\|^2 \right] f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u} && (2.42)\\
&= \sum_{x\in\mathcal{I}} \int_{S_x} d'(\mathbf{u}, x) f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u} && (2.43)\\
&\geq \int_{\mathbb{R}^k} \arg\min_{x'\in\mathcal{I}} d'(\mathbf{u}, x') f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u} && (2.44)\\
&= E[d(\mathbf{U}, Q'(\mathbf{U}))], && (2.45)
\end{aligned}$$

where $Q' \in \mathcal{Q}$ is a quantizer that shares the same codebook as Q and satisfies the COVQ nearest neighbor condition. Hence it is necessary for an optimal COVQ to satisfy the generalized nearest neighbor condition. The centroid condition for the COVQ states that for a fixed partition $\mathcal{S}$, the corresponding optimal centroids satisfy

$$\mathbf{c}_y = \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^k} E[d(\mathbf{U}, \boldsymbol{\omega})|Y = y]. \tag{2.46}$$

The proof for this statement is as follows. Let $\mathcal{Q}$ denote the family of N-level, k-dimensional quantizers. We have that

$$E[d(\mathbf{U}, Q(\mathbf{U}))] = \sum_{y\in\mathcal{I}} E[d(\mathbf{U}, Q(\mathbf{U}))|Y = y]P(Y = y) \tag{2.47}$$

$$= \sum_{y\in\mathcal{I}} E[d(\mathbf{U}, \mathbf{c}_y)|Y = y]P(Y = y) \tag{2.48}$$

$$\geq \sum_{y\in\mathcal{I}} \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^k} E[d(\mathbf{U}, \boldsymbol{\omega})|Y = y]P(Y = y) \tag{2.49}$$

$$= E[d(\mathbf{U}, Q'(\mathbf{U}))], \tag{2.50}$$

where $Q' \in \mathcal{Q}$ is a quantizer that shares the same partition as Q and satisfies the generalized centroid condition. Hence it is necessary for an optimal COVQ to satisfy the centroid condition. Under square error distortion, the COVQ centroid condition becomes

$$\mathbf{c}_y = E[\mathbf{U}|Y = y]. \tag{2.51}$$

This is similar to the VQ case, with the difference being that the distortion is conditioned on the received index at the channel output rather than the encoded index. Note that in cases of high channel noise and a high rate for the COVQ, empty encoding regions may be present when applying the generalized LBG-algorithm.

2.6 Channel Matched Tree Structured Vector Quantization

In this section, we introduce the Channel Matched Tree Structured Vector Quantization (CM-TSVQ) scheme, proposed in [23]. The CM-TSVQ is a multistage, residual quantization scheme, designed to reduce the encoding complexity of a COVQ with the tradeoff of decreased performance and increased memory complexity compared to a full-search COVQ. Let $\mathbf{b} = (b_1, \dots, b_n)$ denote the bit allocation for an n-stage CM-TSVQ, such that b_i represents the number of bits allocated for the i-th stage for all $i \in \{1, \dots n\}$. The overall rate of the quantizer is $\frac{1}{k}\sum_{i=1}^{n} b_i$ bits per source sample. Let $(N_1, \dots, N_n) = (2^{b_1}, \dots, 2^{b_n})$ denote the number of codewords in each stage quantizer. The first stage of the CM-TSVQ is a COVQ without any modifications. The second and subsequent stages are COVQs optimized to quantize the *quantization error* resultant from all previous stages. All quantized values from the codebook of each stage are then added together for the reconstructed vector.

2.6.1 Two-Stage Optimality Conditions

Consider a two-stage CM-TSVQ as depicted in Figure 2.2, with bit allocation $\mathbf{b} = (b_1, b_2)$, where b_1 and b_2 correspond to the bits allocated to the first stage and second stage, respectively. Set $N_1 = 2^{b_1}$ and $N_2 = 2^{b_2}$. Let $\mathcal{S}^{(1)} = \{S_0^{(1)}, \dots, S_{N_1-1}^{(1)}\}$ and $\mathcal{C}^{(1)} = \{\mathbf{c}_0^{(1)}, \dots, \mathbf{c}_{N_1-1}^{(1)}\}$ denote the encoding regions and codebook for the first stage, respectively. Also, let $\mathcal{E}^{(1)}$ and $\mathcal{D}^{(1)}$ be the encoder and decoder, respectively, for the first stage, and let $\mathcal{I}^{(1)} = \{0, \dots, N_1 - 1\}$ and $\mathcal{I}^{(2)} = \{0, \dots, N_2 - 1\}$. The encoder

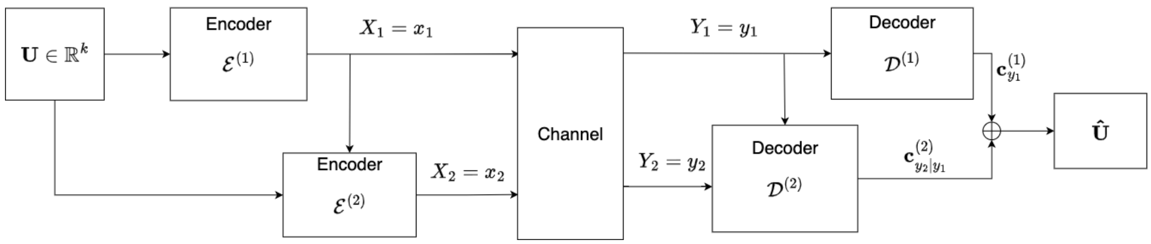

Figure 2.2: Block diagram for a two-stage CM-TSVQ.

and decoder at the second stage can be expressed as the following functions:

$$\mathcal{E}^{(2)} : \mathbb{R}^k \times \mathcal{I}^{(1)} \to \mathcal{I}^{(2)} \tag{2.52}$$

$$\mathcal{D}^{(2)} : \mathcal{I}^{(1)} \times \mathcal{I}^{(2)} \to \mathbb{R}^k. \tag{2.53}$$

Unlike COVQ, the encoder in this case takes in the source value *and the encoded index in the first stage* (x_1). Further, the decoder takes in both the first and second stage received channel indices (y_1 and y_2). In Figure 2.2, x_1 and x_2 are sent through the channel one after the other. The channel can be represented as having identical input and output alphabets given by $\mathcal{I}^{(1)} \times \mathcal{I}^{(2)}$ and having block transition probability for receiving (y_1, y_2) given that (x_1, x_2) was sent given by $P(Y_2 = y_2, Y_1 = y_1 | X_2 = x_2, X_1 = x_1)$. We then have that the expected quantization error for source value $\mathbf{u} \in \mathbb{R}^k$ given that $\mathbf{u} \in S^{(1)}_{x_1}$ for $x_1 \in \mathcal{I}$ is

$$d_2'(\mathbf{u}; x_1, x_2) = E\left[\left\| \mathbf{u} - (\mathbf{c}^{(1)}_{Y_1} + \mathbf{c}^{(2)}_{Y_2|Y_1}) \right\|^2 | X_1 = x_1, X_2 = x_2 \right] \tag{2.54}$$

$$= \sum_{y_1 \in \mathcal{I}^{(1)}} \sum_{y_2 \in \mathcal{I}^{(2)}} P(Y_2 = y_2, Y_1 = y_1 | X_2 = x_2, X_1 = x_1) \left\| \mathbf{u} - \mathbf{c}^{(1)}_{y_1} - \mathbf{c}^{(2)}_{y_2|y_1} \right\|^2. \tag{2.55}$$

Note that d_2' is a generalization of the modified distortion from the stage one case for the stage two case. Let $\mathcal{S}_{x_1}^{(2)} = \{S_{0|x_1}^{(2)}, \ldots, S_{N_2-1|x_1}^{(2)}\}$ denote the set of encoding regions used in the second stage quantizer *given that the inputted source value was encoded with $x_1 \in \mathcal{I}^{(1)}$ in the first stage* (i.e., $\mathcal{S}_{x_1}^{(2)}$ is the partition that further refines the encoding region $S_{x_1}^{(1)}$). Let $\mathcal{C}_{y_1}^{(2)} = \{\mathbf{c}_{0|y_1}^{(2)}, \ldots, \mathbf{c}_{N_2-1|y_1}^{(2)}\}$ be the codebook for the second stage when $y_1 \in \mathcal{I}^{(1)}$ is received in the first stage. The expected distortion of the second stage quantizer conditioned on $X_1 = x_1$ is

$$E[d(\mathbf{U}, \mathbf{c}_{Y_1}^{(1)} + \mathbf{c}_{Y_2|Y_1}^{(2)})|X_1 = x_1] = \sum_{y_1 \in \mathcal{I}^{(1)}} \sum_{x_2, y_2 \in \mathcal{I}^{(2)}} P(Y_2 = y_2, Y_1 = y_1 | X_2 = x_2, X_1 = x_1) \times \int_{S_{x_2|x_1}^{(2)}} \left\| \mathbf{u} - (\mathbf{c}_{y_1}^{(1)} + \mathbf{c}_{y_2|y_1}^{(2)}) \right\|^2 f_{\mathbf{U}}(\mathbf{u})\, d\mathbf{u}. \tag{2.56}$$

As a result, the overall distortion of the two stage quantizer is given by

$$E[d(\mathbf{U}, \mathbf{c}_{Y_1}^{(1)} + \mathbf{c}_{Y_2|Y_1}^{(2)})] = \sum_{y_1, x_1 \in \mathcal{I}^{(1)}} \sum_{x_2, y_2 \in \mathcal{I}^{(2)}} P(Y_2 = y_2, Y_1 = y_1 | X_2 = x_2, X_1 = x_1) \times \int_{S_{x_2|x_1}^{(2)}} \left\| \mathbf{u} - (\mathbf{c}_{y_1}^{(1)} + \mathbf{c}_{y_2|y_1}^{(2)}) \right\|^2 f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u}. \tag{2.57}$$

Note that, when deriving the necessary conditions for the optimal second stage quantizer, we will assume that $\mathcal{S}^{(1)}$ and $\mathcal{C}^{(1)}$ are fixed. To derive the nearest neighbor and centroid conditions, the expected distortion of the second stage quantizer $Q^{(2)} \in \mathcal{Q}_{N_2}$

can be expressed as [12]

$$
\begin{aligned}
E[d(\mathbf{U},&Q^{(2)}(\mathbf{U}))] \\
&= \sum_{x_1 \in \mathcal{I}^{(1)}} \sum_{x_2 \in \mathcal{I}^{(2)}} \int_{S^{(2)}_{x_2|x_1}} d_2'(\mathbf{u}; x_2, x_1) f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u} && (2.58) \\
&= \sum_{y_1 \in \mathcal{I}^{(1)}} \sum_{y_2 \in \mathcal{I}^{(2)}} E[d(\mathbf{U}, \mathbf{c}^{(1)}_{y_1} + \mathbf{c}^{(2)}_{y_2|y_1}) | Y_1 = y_1, Y_2 = y_2] \times P(Y_1 = y_1, Y_2 = y_2). && (2.59)
\end{aligned}
$$

The identities in (2.58) and (2.59) are analogous to those in (2.43) and (2.48), respectively. Following a similar proof, we can derive the following optimality conditions for the COVQ. The nearest neighbor condition for the second stage encoding regions for fixed codebook $\mathcal{C}^{(2)}_{y_1}$ for all $y_1 \in \mathcal{I}^{(1)}$ is [12]

$$
S_{x_2|x_1} = \{\mathbf{u} \in \mathbb{R}^k : d_2'(\mathbf{u}; x_1, x_2) \le d_2'(\mathbf{u}; x_1, x_2'), \quad x_2' \in \mathcal{I}^{(2)}\}, \quad x_2 \in \mathcal{I}^{(2)}. \quad (2.60)
$$

The centroid condition for the second stage codebook given fixed partition $\mathcal{S}^{(2)}_{x_1}$ for all $x_1 \in \mathcal{I}^{(1)}$ is

$$
\mathbf{c}^{(2)}_{y_2|y_1} = \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^k} E[d(\mathbf{U}, \mathbf{c}^{(1)}_{Y_1} + \boldsymbol{\omega}) | Y_1 = y_1], \quad y_1 \in \mathcal{I}^{(1)}, y_2 \in \mathcal{I}^{(2)}. \quad (2.61)
$$

Under the square error distortion, the centroid condition reduces to

$$
\begin{aligned}
\mathbf{c}^{(2)}_{y_2|y_1} &= E[\mathbf{U} - \mathbf{c}^{(1)}_{y_1} | Y_2 = y_2, Y_1 = y_1] && (2.62) \\
&= E[\mathbf{U} | Y_2 = y_2, Y_1 = y_1] - \mathbf{c}^{(1)}_{y_1}, \quad y_1 \in \mathcal{I}^{(1)}, y_2 \in \mathcal{I}^{(2)}. && (2.63)
\end{aligned}
$$

2.6.2 Generalization for Multiple Stages

We next consider optimality conditions for CM-TSVQ for multiple stages greater than 2. Let n be the number of stages in the CM-TSVQ and let $\mathbf{b} = (b_1, \ldots, b_n)$ be the bit allocation vector, such that b_j bits are allocated for the j-th stage for all $j \in \{1, \ldots, n\}$. Let $\mathcal{C}^{(i)}_{y_{i-1},\ldots,y_1}$ be the codebook for the i-th stage used when the sequence $(y_1, \ldots, y_{i-1})$ is received from the previous $i-1$ stages. Let $\mathcal{S}^{(i)}_{x_{i-1},\ldots,x_1}$ be the set of encoding regions for the i-th stage used for source values encoded with the sequence $(x_1, \ldots, x_{i-1})$ in the previous $i-1$ stages. For brevity, let $X^i = (X_i, \ldots, X_1), Y^i = (Y_i, \ldots, Y_1)$ and let $x^i = (x_i, \ldots, x_1), y^i = (y_i, \ldots, y_1)$. Further, we will denote $\mathcal{I}^i = \mathcal{I}^{(1)} \times \cdots \times \mathcal{I}^{(i)}$, where $\mathcal{I}^{(j)} = \{0, 1, \ldots, 2^{b_j} - 1\}$ for $j \in \{1, \ldots, n\}$, and $\hat{\mathbf{u}}^{(i)} = \mathbf{c}^{(1)}_{y_1} + \mathbf{c}^{(2)}_{y_2|y_1} + \cdots + \mathbf{c}^{(i)}_{y_i|y^{i-1}}$. The encoders and decoders at stage i of the CM-TSVQ can be described by the following functions,

$$\mathcal{E}^{(i)} : \mathbb{R}^k \times \mathcal{I}^{i-1} \to \mathcal{I}^{(i)} \tag{2.64}$$

$$\mathcal{D}^{(i)} : \mathcal{I}^i \to \mathbb{R}^k. \tag{2.65}$$

Specifically, at stage i, the encoder takes in the source vector and the previous $i-1$ encoding indices and outputs a channel index. The decoder takes in all received channel indices and outputs the reconstructed value $\hat{\mathbf{u}}^{(i)}$ for the source vector $\mathbf{u}$. The overall expected distortion of the CM-TSVQ at stage i is given by

$$D^{(i)} = \sum_{x^i \in \mathcal{I}^i} \sum_{y^i \in \mathcal{I}^i} P(Y^i = y^i | X^i = x^i) \int_{S^{(i)}_{x_i|x^{i-1}}} \left\| \mathbf{u} - \hat{\mathbf{u}}^{(i)} \right\|^2 f_{\mathbf{U}}(\mathbf{u}) d\mathbf{u}. \tag{2.66}$$

Let

$$d_i'(\mathbf{u};\mathbf{x}^i) = \sum_{x^i \in \mathcal{I}^i} \sum_{y^i \in \mathcal{I}^i} P(Y^i = y^i | X^i = x^i) \left\| \mathbf{u} - \mathbf{c}_{y_1} - \mathbf{c}_{y_2|y_1} - \cdots - \mathbf{c}_{y_i|y^{i-1}} \right\|^2 \quad (2.67)$$

denote the expected distortion when source vector $\mathbf{u}$ is quantized with the sequence $x_i, \ldots, x_1$. Note that when describing the necessary conditions for optimality in the ith stage, we assume that all partitions and codebooks from the previous $i-1$ stages are fixed. The nearest neighbor condition for the i-th stage encoding region given fixed codebook $\mathcal{C}_{y^i}^{(i)}$ for all $\mathbf{y}^i \in \mathcal{I}^i$ is

$$S_{x_i|x_{i-1},\ldots,x_1}^{(i)} = \{\mathbf{u} \in \mathbb{R}^k : d_i'(\mathbf{u}; x_i, x_{i-1}, \ldots, x_1) \leq d_i'(\mathbf{u}; x_i', x_{i-1}, \ldots, x_1), \quad x_i' \in \mathcal{I}^{(i)}\} \quad (2.68)$$

for $(x_i, \ldots, x_1) \in \mathcal{I}^{(i)} \times \cdots \times \mathcal{I}^{(1)}$. The centroid condition given fixed partition $\mathcal{S}_{x^{i-1}}^{(i)}$ for all $x^{i-1} \in \mathcal{I}^{i-1}$ is

$$\mathbf{c}_{y_i|y^{i-1}}^{(i)} = \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^k} E[d(\mathbf{U}, \mathbf{c}_{Y_1}^{(1)} + \mathbf{c}_{Y_2|Y_1}^{(2)} + \cdots + \boldsymbol{\omega}) | Y^i = y^i], \quad y^i \in \mathcal{I}^i \quad (2.69)$$

which under the square error distortion, reduces to

$$\mathbf{c}_{y_i|y^{i-1}}^{(i)} = E[\mathbf{U}|Y^i = y^i] - \mathbf{c}_{y_1}^{(1)} - \mathbf{c}_{y_2|y_1}^{(2)} - \cdots - \mathbf{c}_{y_{i-1}|y^{i-2}}^{(i-1)}, \quad y^i \in \mathcal{I}^i. \quad (2.70)$$

As pointed out, for source value $\mathbf{u}$ and received sequences $y_i, \ldots, y_1$, the reconstructed codeword from the decoder is $\hat{\mathbf{u}}^{(i)} = \mathbf{c}_{y_1}^{(1)} + \mathbf{c}_{y_2|y_1}^{(2)} + \cdots + \mathbf{c}_{y_i|\mathbf{y}^{i-1}}^{(i)}$.

2.6.3 Algorithm for Designing Locally Optimal CM-TSVQs

Given the optimality conditions for the CM-TSVQ, a natural question is how to design an optimal CM-TSVQ. One method for finding locally optimal quantizers follows from the LBG-algorithm, in which the generalized nearest neighbor and centroid conditions are applied successively until convergence. Below is a generalization of the LBG-algorithm for CM-TSVQs at stage i, given all codebooks and encoding regions in stages $1, \ldots, i-1$. Note that at stage i all the codebooks and partitions from the previous $i-1$ stages are fixed.

1. Given $\mathcal{S}^{(j)}_{x^{j-1}}$ and $\mathcal{C}^{(j)}_{y^{j-1}}$ for all $j = 1, \ldots, i-1$, let $D^{(m)}$ represent the distortion at the m-th iteration respectively. Set $m = 0$ and $D^{(m)} = \infty$. Choose an initial $\mathcal{S}^{(i,m)}_{x^{i-1}}$, $\mathcal{C}^{(i,m)}_{y^{i-1}}$, and T. For this algorithm let $\mathcal{S}^{(i,m)}_{x^{i-1}}$, $\mathcal{C}^{(i,m)}_{y^{i-1}}$ represent the m-th iteration of the i-th stage encoding regions and codebook respectively.

2. Using $\mathcal{C}^{(i,m)}_{y^{i-1}}$, for all $x_i \in \mathcal{I}^{(i)}$ set $\mathcal{S}^{(i,m+1)}_{x^{i-1}} = \{\mathbf{u} \in \mathbb{R}^k : d'_i(\mathbf{u}; x_i, x_{i-1}, \ldots, x_1) \leq d'_i(\mathbf{u}; x'_i, x_{i-1}, \ldots, x_1), \quad x'_i \in \mathcal{I}^{(i)}\}$.

3. Using $\mathcal{S}^{(i,m+1)}_{x^{i-1}}$, set $\mathcal{C}^{(i,m+1)}_{y^{i-1}} = E[\mathbf{U}|Y^i = y^i] - \mathbf{c}^{(1)}_{y_1} - \mathbf{c}^{(2)}_{y_2|y_1} - \cdots - \mathbf{c}^{(i-1)}_{y_{i-1}|y^{i-2}}$.

4. Set $m = m + 1$. Calculate $D^{(m+1)}$. If $\frac{D^{(m)} - D^{(m+1)}}{D^{(m)}} \geq T$, then go to step 2. Otherwise, return $\mathcal{S}^{(i,m)}_{x^{i-1}}$, $\mathcal{C}^{(i,m)}_{y^{i-1}}$.

2.7 Adaptive Channel Optimized Vector Quantization

In this section, we describe the Adaptive Channel Optimized Vector Quantization (ACOVQ) method, proposed in [5], for channels with feedback. The difference between the ACOVQ and CM-TSVQ schemes is that in the ACOVQ the encoder has access to a noiseless feedback link to the channel outputs as well as the previous encoding indices. This feedback is then used to *update* the source distribution by calculating the posterior distribution. This posterior distribution is then treated as a new source distribution and refined further by another quantizer. Additionally in this scheme, only the codebook in the *final stage* is used to reconstruct the source vector, unlike in the CM-TSVQ scheme, where the reconstructed codeword is the sum of the codeword from each stage.

2.7.1 Two-Stage Optimality Conditions

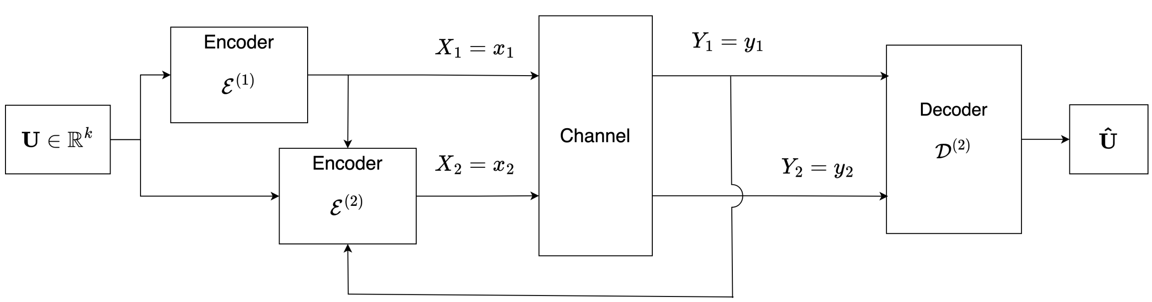

Figure 2.3: Block diagram for two-stage ACOVQ.

Similar to CM-TSVQ, the first stage of ACOVQ is a COVQ without any modifications. Consider a two stage ACOVQ with bit allocation $\mathbf{b} = (b_1, b_2)$ depicted in Figure 2.3. Let $Y_1 = y_1 \in \mathcal{I}^{(1)}$ be the index received by the decoder in the first stage. This index is then sent to the second stage encoder noiselessly via a unit-delay feedback-link. Unlike CM-TSVQ, the first stage decoder is not used in this scheme. Instead, only the codebook in the last stage is used to reconstruct the source vector. Let $f_{\mathbf{U}}(\mathbf{u})$ represent the probability density function for the source. For each $y_1 \in \mathcal{I}^{(1)}$, the source distribution is updated at the encoder by calculating the following posterior distribution:

$$f_{\mathbf{U}|Y_1}(\mathbf{u}|y_1) = \frac{f_{\mathbf{U}}(\mathbf{u})P(Y_1 = y_1|\mathbf{U} = \mathbf{u})}{P(Y_1 = y_1)} \tag{2.71}$$

$$= \frac{f_{\mathbf{U}}(\mathbf{u})\sum_{x_1 \in \mathcal{I}^{(1)}} P(Y_1 = y_1|X_1 = x_1, \mathbf{U} = \mathbf{u})P(X_1 = x_1|\mathbf{U} = \mathbf{u})}{\sum_{x_1 \in \mathcal{I}^{(1)}} P(Y_1 = y_1|X_1 = x_1)P(X_1 = x_1)} \tag{2.72}$$

$$= \frac{f_{\mathbf{U}}(\mathbf{u})\sum_{x_1 \in \mathcal{I}^{(1)}} P(Y_1 = y_1|X_1 = x_1)1\{\mathbf{u} \in S_{x_1}^{(1)}\}}{\sum_{x_1 \in \mathcal{I}^{(1)}} P(Y_1 = y_1|X_1 = x_1)\int_{S_{x_1}^{(1)}} f_{\mathbf{U}}(\mathbf{u})d\mathbf{u}} \tag{2.73}$$

$$= \frac{f_{\mathbf{U}}(\mathbf{u})P(Y_1 = y_1|X_1 = \mathcal{E}^{(1)}(\mathbf{u}))}{\sum_{x_1 \in \mathcal{I}^{(1)}} P(Y_1 = y_1|X_1 = x_1)\int_{S_{x_1}^{(1)}} f_{\mathbf{U}}(\mathbf{u})d\mathbf{u}}. \tag{2.74}$$

Note that the simplification in (2.73) is due to the causality channel condition by nature in which the channel is operated, where it is assumed that the following Markov chain holds [2, p. 160]:

$$\mathbf{U} \to (X^i, Y^{i-1}) \to Y_i, \quad i = 1, 2, 3, \ldots. \tag{2.75}$$

Note that under feedback, the property in (2.2) for memoryless channels may not necessarily hold since the feedback may affect the probability of the next encoded output [2]. In this case, the causality condition in (2.75) simplifies to [2, p. 161],

$$(\mathbf{U}, X^{i-1}, Y^{i-1}) \to X_i \to Y_i. \tag{2.76}$$

Further, for the simplification in (2.74), recall that the encoding regions form a partition over $\mathbb{R}^k$. Therefore $\mathbf{u}$ can only belong to one encoding region and $1_{\{\mathbf{u} \in S_{x_1}^{(1)}\}} = 1 \iff \mathcal{E}^{(1)}(\mathbf{u}) = x_1$. Therefore the only term in the summation $\sum_{x_1 \in \mathcal{I}^{(1)}} P(Y_1 = y_1|X_1 = x_1) 1\{\mathbf{u} \in S_{x_1}^{(1)}\}$ that is non-zero is the term where $x_1 \in \mathcal{I}^{(1)}$ corresponds to the encoding region that $\mathbf{u}$ is in (where $x_1 = \mathcal{E}^{(1)}(\mathbf{u})$). The encoder and decoder for the second stage can be represented as

$$\mathcal{E}^{(2)} : \mathbb{R}^k \times \mathcal{I}^{(1)} \times \mathcal{I}^{(1)} \to \mathcal{I}^{(2)} \tag{2.77}$$

$$\mathcal{D}^{(2)} : \mathcal{I}^{(1)} \times \mathcal{I}^{(2)} \to \mathbb{R}^k. \tag{2.78}$$

Let y_1 be the received index via the feedback link. We have encoding regions $\mathcal{S}_{y_1}^{(2)} = \{S_{0|y_1}^{(2)}, S_{1|y_1}^{(2)}, \ldots S_{N_2-1|y_1}^{(2)}\}$ and codebook $\mathcal{C}_{y_1}^{(2)} = \{\mathbf{c}_{0|y_1}^{(2)}, \mathbf{c}_{1|y_1}^{(2)}, \ldots, \mathbf{c}_{N_2-1|y_1}^{(2)}\}$ corresponding to the posterior distribution $f_{\mathbf{U}|\mathbf{y_1}}$. The expected distortion given $Y_1 = y_1$ can be expressed as

$$E[d(\mathbf{U}, Q(\mathbf{U}))|Y_1 = y_1] = \int_{\mathbb{R}^k} d(\mathbf{u}, Q(\mathbf{u})) f_{\mathbf{U}|y_1}(\mathbf{u}) d\mathbf{u} \tag{2.79}$$
$$= \sum_{x_2, y_2 \in \mathcal{I}^{(2)}} \sum_{x_1 \in \mathcal{I}^{(1)}} \int_{S_{x_1}^{(1)} \cap S_{x_2|y_1}^{(1)}} P(Y_2 = y_2 | X_2 = x_2, Y_1 = y_1, X_1 = x_1)$$

$$\times\, d(\mathbf{u}, \mathbf{c}_{y_2|y_1}) f_{\mathbf{U}|y_1}(\mathbf{u}) d\mathbf{u} \tag{2.80}$$

$$= \sum_{x_2, y_2 \in \mathcal{I}^{(2)}} \sum_{x_1 \in \mathcal{I}^{(1)}} \int_{S^{(1)}_{x_1} \cap S^{(1)}_{x_2|y_1}} P(Y_2 = y_2 | X_2 = x_2, Y_1 = y_1, X_1 = x_1)$$

$$\times \left\| \mathbf{u} - \mathbf{c}_{y_2|y_1} \right\|^2 f_{\mathbf{U}|y_1}(\mathbf{u}) d\mathbf{u}. \tag{2.81}$$

With an abuse of notation, let

$$d_2'(\mathbf{u}; y_1, \mathcal{E}^{(1)}(\mathbf{u}), x_2) = \sum_{y_2 \in \mathcal{I}^{(2)}} P(Y_2 = y_2 | X_2 = x_2, Y_1 = y_1, X_1 = \mathcal{E}^{(1)}(\mathbf{u})) \left\| \mathbf{u} - \mathbf{c}^{(2)}_{y_2|y_1} \right\|^2 \tag{2.82}$$

represent the expected distortion when source value $\mathbf{u}$ is quantized with index x_2 given $Y_1 = y_1$ and $X_1 = \mathcal{E}^{(1)}(\mathbf{u})$. We can then rewrite the expected distortion in terms of the modified distortion in (2.81) as follows

$$E[d(\mathbf{U}, Q(\mathbf{U})) | Y_1 = y_1] = \sum_{x_2 \in \mathcal{I}^{(2)}} \sum_{x_1 \in \mathcal{I}^{(1)}} \int_{S^{(1)}_{x_1} \cap S^{(1)}_{x_2|y_1}} d_2'(\mathbf{u}; y_1, x_1, x_2) f_{\mathbf{U}|y_1}(\mathbf{u}) d\mathbf{u}. \tag{2.83}$$

It then follows that given fixed codebook $\mathcal{C}^{(2)}_{y_1}$, an optimal second stage quantizer must have encoding regions that satisfy the following generalized nearest neighbor condition:

$$S^{(2)}_{x_2|y_1} = \{\mathbf{u} \in \mathbb{R}^k : d_2'(\mathbf{u}; y_1, \mathcal{E}^{(1)}(\mathbf{u}), x_2) \leq d_2'(\mathbf{u}; y_1, \mathcal{E}^{(1)}(\mathbf{u}), x_2'), \quad x_2' \in \mathcal{I}^{(2)}\},$$

$$x_2 \in \mathcal{I}^{(2)}, y_1 \in \mathcal{I}^{(1)}. \tag{2.84}$$

The above expected distortion can also be expressed as

$$E[d(\mathbf{U}, Q(\mathbf{U}))|Y_1 = y_1] = \sum_{y_2 \in \mathcal{I}^{(2)}} P(Y_2 = y_2|Y_1 = y_1) E[d(\mathbf{U}, Q(\mathbf{U}))|Y_1 = y_1, Y_2 = y_2]. \tag{2.85}$$

Therefore, it follows that for a fixed partition $\mathcal{S}_{y_1}^{(2)}$ an optimal second stage quantizer must have a codebook that satisfies the following generalized centroid condition:

$$\mathbf{c}_{y_2|y_1}^{(2)} = \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^k} E[d(\mathbf{U}, \boldsymbol{\omega})|Y_1 = y_1, Y_2 = y_2] \tag{2.86}$$

which, under the square error distortion, reduces to

$$\mathbf{c}_{y_2|y_1}^{(2)} = E[\mathbf{U}|Y_1 = y_1, Y_2 = y_2]. \tag{2.87}$$

Figure 2.4 depicts the posterior distributions of an ACOVQ scheme in the scalar case with bit allocation $\mathbf{b} = (2, 2)$ over the Polya Contagion Channel with memory $M = 1$, $\epsilon = 0.05$, and $\delta = 5$ (details on how indices are sent over this binary channel are provided in Section 3.4.1). Each box represents a COVQ in the ACOVQ framework, displaying the source density function (blue line) and the corresponding centroids (red dots). Arrows connecting the first-stage quantizer to the second-stage quantizers indicate the channel output for the posterior distribution each second-stage quantizer is responsible for refining. For the second-stage quantizer boxes, the plot of the source density function is accompanied by the probability of the channel output and the expected distortion of the COVQ when quantizing the respective

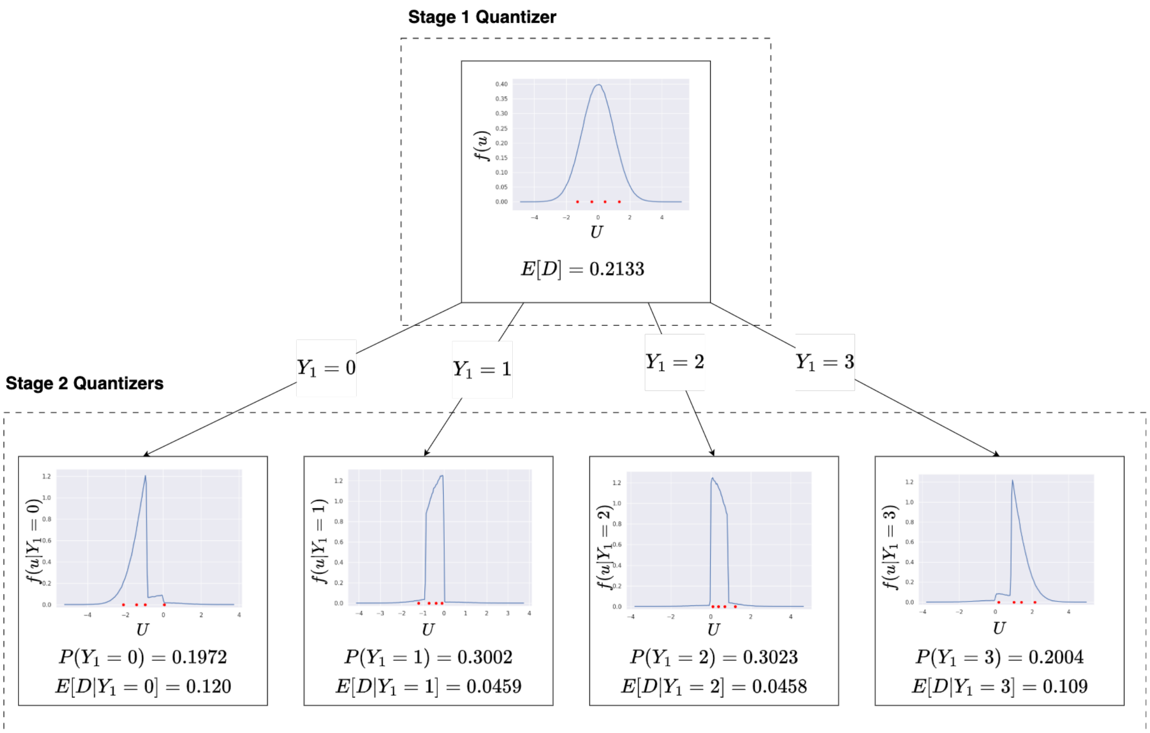

Figure 2.4: Posterior distributions of ACOVQ with bit allocation $\mathbf{b} = (2, 2)$ over the Polya contagion channel with memory $M = 1$, $\epsilon = 0.05$, and $\delta = 5$.

posterior distribution. We can see that with feedback, the variance of the posterior source distribution decreases. That is, with noiseless feedback, the decoder has more certainty about the source value that was sent. The more stages added, the more the posterior distributions will resemble the delta function. In Chapter 4, we will explore these properties and how we can use these differences to improve the performance of ACOVQ. We also observe from the figure that ACOVQ, similar to CM-TSVQ, successively refines the initial source distribution and has a tree like structure.

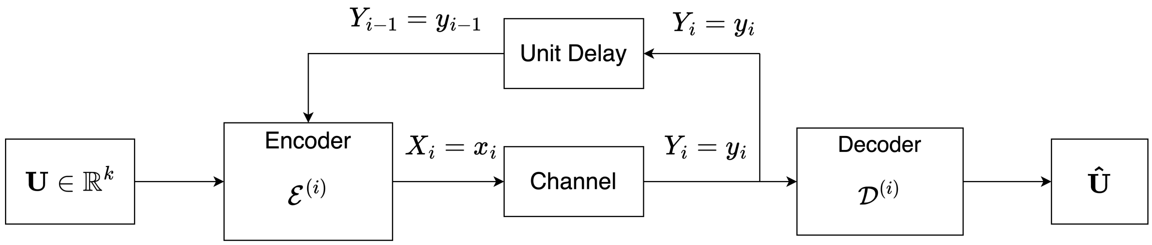

Figure 2.5: Block diagram for the ith stage of ACOVQ in a noiseless feedback communication system.

2.7.2 Generalization for Multiple Stages

The ith stage of an ACOVQ scheme is shown in Figure 2.5. Although not explicitly depicted in the figure, the encoder at stage i has access to all previous encoders (i.e., $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}$). Let $y_1, y_2, \ldots, y_{i-1}$ be the sequence of channel outputs received at the encoder via the noiseless feedback link at the i-th stage and let $\mathcal{S}^{(1)}, \mathcal{S}^{(2)}_{y_1}, \ldots, \mathcal{S}^{(i-1)}_{yi-2}$ be the fixed partitions from the previous stages. Here we will show the necessary optimality conditions for the encoder and decoder in the ith stage. The posterior distribution can be calculated recursively as follows:

$$f_{\mathbf{U}|Y^{i-1}}(\mathbf{u}|y^{i-1}) = \frac{f_{\mathbf{U}|Y^{i-2}}(\mathbf{u}|y^{i-2})P(Y_{i-1} = y_{i-1}|Y^{i-2} = y^{i-2}, \mathbf{U} = \mathbf{u})}{P(Y_{i-1} = y_{i-1}|Y^{i-2} = y^{i-2})} \tag{2.88}$$

$$= \frac{f_{\mathbf{U}|\mathbf{Y}^{i-2}}(\mathbf{u}|\mathbf{y}^{i-2})P(Y_{i-1} = y_{i-1}|Y^{i-2} = y^{i-2}, \mathbf{U} = \mathbf{u})}{\sum_{\mathbf{x}^{i-1} \in \mathcal{I}^{i-1}} P(Y_{i-1} = y_{i-1}|Y^{i-2} = y^{i-2}, X^{i-1} = x^{i-1}) \int_{\alpha(\mathbf{x}^{i-1})} f_{\mathbf{U}|Y^{i-2}}(\mathbf{u}|y^{i-2})}, \tag{2.89}$$

where $\alpha(x^{i-1}) = S^{(i-1)}_{x_{i-1}|y^{i-2}} \cap S^{(i-2)}_{x_{i-2}|y^{i-3}} \cap S^{(i-3)}_{x_{i-3}|y^{i-4}} \cap \cdots \cap S^{(1)}_{x_1}$ (i.e., the intersection of the encoding regions corresponding to received sequence y^{i-2} and encoded sequence x^{i-1}).

The encoded sequence for source vector $\mathbf{u}$ is built recursively, where the encoded index at the i-th stage depends on all the previous encoded indices $x_{i-1}, \ldots, x_1$. Correspondingly, for notational simplicity, we represent the indices for each stage as a recursive formula. For $Y^{i-1} = y^{i-1}$ let

$$\gamma^{(1)}(\mathbf{u}) = \mathcal{E}^{(1)}(\mathbf{u}) \tag{2.90}$$

$$\gamma^{(2)}_{y_1}(\mathbf{u}) = \mathcal{E}^{(2)}(\mathcal{E}^{(1)}(\mathbf{u}), y_1, \mathbf{u}) \tag{2.91}$$

$$= \mathcal{E}^{(2)}(\gamma^{(1)}(\mathbf{u}), y_1, \mathbf{u}) \tag{2.92}$$

$$\gamma^{(3)}_{y^2}(\mathbf{u}) = \mathcal{E}^{(3)}(\mathcal{E}^{(2)}(\mathcal{E}^{(1)}(\mathbf{u}), y_1, \mathbf{u}), \mathcal{E}^{(1)}(\mathbf{u}), y_2, y_1, \mathbf{u}) \tag{2.93}$$

$$= \mathcal{E}^{(3)}(\gamma^{(2)}_{y_1}(\mathbf{u}), \gamma^{(1)}(\mathbf{u}), y_2, y_1, \mathbf{u}) \tag{2.94}$$

$$\vdots \tag{2.95}$$

$$\gamma^{(i)}_{y^{i-1}}(\mathbf{u}) = \mathcal{E}^{(i)}(\gamma^{(i-1)}_{y^{i-2}}(\mathbf{u}), \ldots, \gamma^{(1)}(\mathbf{u}), y^{i-1}, \mathbf{u}). \tag{2.96}$$

That is, for received sequence y^{j-1}, $\gamma^{(j)}_{y^{j-1}}(\mathbf{u})$ represents the encoded index at stage j for all $j = 1, \ldots, i$. Let $\gamma^{i}_{y^{i-1}}(\mathbf{u}) = (\gamma^{(i)}_{y^{i-1}}(\mathbf{u}), \ldots, \gamma^{(1)}(\mathbf{u}))$ With an abuse of notation, let

$$\begin{aligned} & d'_i(\mathbf{u}; y^{i-1}, \gamma^{i-1}_{y^{i-2}}(\mathbf{u}), x_i) \\ & \quad = \sum_{y_i \in \mathcal{I}^{(i)}} P(Y_i = y_i | X_i = x_i, Y^{i-1} = y^{i-1}, X^{i-1} = \gamma^{i-1}_{y^{i-2}}(\mathbf{u})) \left\| \mathbf{u} - \mathbf{c}^{(i)}_{y_i|y^{i-1}} \right\|^2 \end{aligned} \tag{2.97}$$

be the expected distortion at stage i, when $\mathbf{u}$ is encoded with $x_i \in \mathcal{I}^{(i)}$ given received sequence $Y^{i-1} = y^{i-1}$ and encoded sequence $X^{i-1} = x^{i-1}$. Note that at stage i, y^{i-1} and $\gamma^{i-1}_{y^{i-2}}(\mathbf{u})$ are fixed values. It then follows that given fixed codebook $\mathcal{C}^{(i)}_{y^{i-1}}$, an optimal ith stage quantizer will have encoding regions that satisfy the following generalized nearest neighbor condition:

$$S^{(i)}_{x_i|y^{i-1}} = \{\mathbf{u} \in \mathbb{R}^k : d'_i(\mathbf{u}; y^{i-1}, \gamma^{i-1}_{y^{i-2}}(\mathbf{u}), x_i) \leq d'_i(\mathbf{u}; y^{i-1}, \gamma^{i-1}_{y^{i-2}}(\mathbf{u}), x'_i), \quad x'_i \in \mathcal{I}^{(i)}\} \quad x_i \in \mathcal{I}^{(i)}, y^{i-1} \in \mathcal{I}^{i-1}. \tag{2.98}$$

Conversely, it follows that for fixed partition $\mathcal{S}^{(i)}_{y^{i-1}}$ that an optimal ith stage quantizer will have encoding regions that satisfy the following generalized centroid condition:

$$\mathbf{c}^{(i)}_{y_i|y^{i-1}} = \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^k} E[d(\mathbf{U}, \boldsymbol{\omega})|Y^i = y^i] \tag{2.99}$$

which, under the square error distortion, reduces to

$$\mathbf{c}^{(i)}_{y_i|y^{i-1}} = E[\mathbf{U}|Y^i = y^i]. \tag{2.100}$$

Similar to CM-TSVQ and the LBG quantizer, locally optimal ACOVQs can be designed, given an initial set of encoding regions and codebook, by repeatedly applying the nearest neighbor and centroid conditions until convergence. However, unlike CM-TSVQ, the value of the reconstructed vector is

$$\hat{\mathbf{u}}^{(i)} = \mathbf{c}^{(i)}_{y_i|y^{i-1}} \tag{2.101}$$

for channel output sequence $Y^i = y^i$. Note that in this case, only the last set of codebooks are used for the reconstructed vector, whereas in CM-TSVQ, all codewords from all previous stage codebooks are used.

2.8 Initial Codebook Design

As noted earlier, the generalized LBG-Algorithm converges to a local minimum and requires an initial codebook to find an optimal quantizer. The choice of the initial codebook is important to avoid "bad" local minima. A naive method would be to select N random vectors for a codebook with N codewords. However, in [11] [13], the authors proposed an algorithm for index assignment that, in general, outperforms random assignment. The distortion for a VQ through a noisy channel can be rewritten as [17]:

$$
\begin{aligned}
D &= E[d(\mathbf{U}, \mathbf{c}_{\mathbf{Y}})] && (2.102)\\
&= \sum_{x\in\mathcal{I}} \int_{S_x} \sum_{y\in\mathcal{I}} \left[P(Y=y|X=x)\,\|\mathbf{u}-\mathbf{c}_y\|^2\right] f_{\mathbf{U}}(\mathbf{u})d\mathbf{u} && (2.103)\\
&= \sum_{x\in\mathcal{I}} \int_{S_x} \sum_{y\in\mathcal{I}} \left[P(Y=y|X=x)\left\|(\mathbf{u}-\mathbf{c}_x)+(\mathbf{c}_x-\mathbf{c}_y)\right\|^2\right] f_{\mathbf{U}}(\mathbf{u})d\mathbf{u} && (2.104)\\
&= \sum_{x\in\mathcal{I}} \int_{S_x} \sum_{y\in\mathcal{I}} \Big[P(Y=y|X=x)\big(\|\mathbf{u}-\mathbf{c}_x\|^2+\|\mathbf{c}_x-\mathbf{c}_y\|^2+ && (2.105)\\
&\qquad (\mathbf{u}-\mathbf{c}_x)(\mathbf{c}_x-\mathbf{c}_y)^T\big)\Big] f_{\mathbf{U}}(\mathbf{u})d\mathbf{u}\\
&= \underbrace{\sum_{x\in\mathcal{I}} \int_{S_x} \|\mathbf{u}-\mathbf{c}_x\|^2 f_{\mathbf{U}}(\mathbf{u})d\mathbf{u}}_{D_q} + \underbrace{\sum_{x\in\mathcal{I}} \sum_{y\in\mathcal{I}} P(Y=y|X=x)\|\mathbf{c}_x-\mathbf{c}_y\|^2}_{D_c} && (2.106)\\
&\qquad + \underbrace{\sum_{x\in\mathcal{I}} \int_{S_x} \sum_{y\in\mathcal{I}} \big((\mathbf{u}-\mathbf{c}_x)(\mathbf{c}_x-\mathbf{c}_y)^T\big) f_{\mathbf{U}}(\mathbf{u})d\mathbf{u}}_{D_*}.
\end{aligned}
$$

Therefore, the overall distortion can be decomposed into the following components: D_q, D_c, and D_*, the distortions from the quantizer, the channel, and the cross-term, respectively. However, for a quantizer satisfying the VQ centroid condition, $D_* = 0$ and the overall distortion can be written as

$$D = D_q + D_c. \tag{2.107}$$

Let τ represent an index assignment such that

$$\tau : \mathcal{I} \rightarrow \mathcal{I} \tag{2.108}$$

is a 1-to-1 function. The goal of the simulated annealing algorithm proposed in [11] is to find an index τ that minimizes D_c, where

$$D_c(\tau) = \sum_{x \in \mathcal{I}} \sum_{y \in \mathcal{I}} P(Y = \tau(y)|X = \tau(x)) \left\| \mathbf{c}_y - \mathbf{c}_x \right\|^2 \tag{2.109}$$

is the channel distortion given the index assignment τ. The simulated annealing algorithm is as follows.

1. Let T represent an effective temperature and set $T = T_0$ as the initial temperature. Set initial values for $\alpha, N_{success}, N_{failure}, N_{cut}$. Choose a random state τ.

2. Randomly choose another state τ' and let $\Delta D_c = D_c(\tau') - D_c(\tau)$. If $\Delta D_c < 0$, then set $b = b'$ and go to step 3. Otherwise, set $b = b'$ with probability $e^{-\Delta D_c/T}$ and go to step 3.

T_0	10
T_f	2.5×10^{-4}
α	0.97
$N_{failure}$	50000
$N_{success}$	5
N_{cut}	200

Table 2.1: Simulated annealing parameters.

3. If the number of decreases in distortion, or “energy drops,” exceeds $N_{success}$ then go to step 4. If the number of attempts exceeds N_{cut}, then go to step 4.

4. Set $T = \alpha T$. If $T < T_f$ or the number of unsuccessful “energy drop” attempts exceed $N_{failure}$, then stop the algorithm and return τ. Otherwise go to step 2.

Table 2.1 lists the parameter values used for initialization in this thesis for the simulated annealing algorithm.

Chapter 3

Tree Structured Vector Quantization With Noiseless Feedback

3.1 Introduction

In the previous chapter, different multi-stage quantization schemes were explored, including CM-TSVQ and ACOVQ. As noted earlier, CM-TSVQ is optimized for systems without noiseless feedback, while ACOVQ is optimized for systems with noiseless feedback. Given that both schemes are tree structured and have a successive refinement approach to quantizing the source, a natural question is whether CM-TSVQ can be generalized for systems with noiseless feedback and how it would compare to ACOVQ. We will call the CM-TSVQ generalized for noiseless feedback the Adaptive Tree Structured Vector Quantizer (ATSVQ). In this section, we will derive the nearest neighbor and centroid conditions for ATSVQ and show that the resulting quantizer is in fact equivalent to ACOVQ (i.e., both schemes have equivalent nearest neighbor and centroid conditions).

3.2 Preliminaries

Let $\mathbf{U}$ be a k-dimensional random vector with probability density function $f_{\mathbf{U}}(\mathbf{u})$ and support $\mathbb{R}^k$. Let $\mathbf{b} = (b_1, \ldots, b_n)$ be the bit allocation vector for a n-stage ATSVQ and let $N_i = 2^{bi}$ represent the size of the codebook at stage i. Let $\mathcal{S}^{(1,AT)} = \{S_0^{(1,AT)}, \ldots, S_{N_1-1}^{(1,AT)}\}$ and $\mathcal{C}^{(1,AT)} = \{\mathbf{c}_0^{(1,AT)}, \ldots, \mathbf{c}_{N_1-1}^{(1,AT)}\}$ denote the encoding regions and codebook for the first stage of the ATSVQ, respectively. For $i \in \{2, \ldots, n\}$, let $\mathcal{S}_{y^{i-1}}^{(i,AT)} = \{S_{0|y^{i-1}}^{(i,AT)}, \ldots, S_{N_i-1|y^{i-1}}^{(i,AT)}\}$ and $\mathcal{C}_{y^{i-1}}^{(i,AT)} = \{\mathbf{c}_{0|y^{i-1}}^{(i,AT)}, \ldots, \mathbf{c}_{N_i-1|y^{i-1}}^{(i,AT)}\}$ be the encoding regions and codebook for the i-th stage with received sequence $Y^{i-1} = y^{i-1}$, respectively. The first stage of ATSVQ consists of a COVQ without any modifications.

3.2.1 Second Stage Derivations

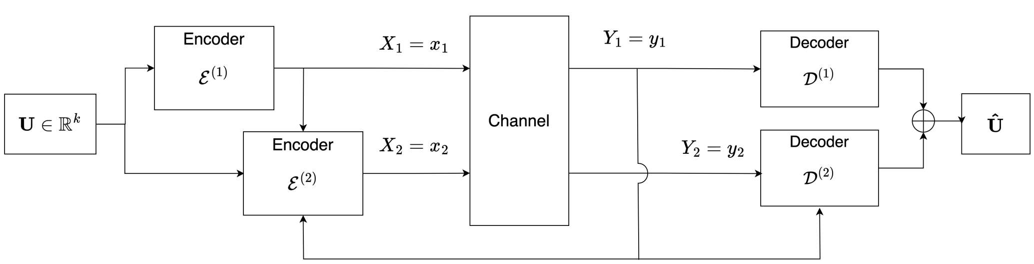

Figure 3.1: Communication block diagram for a two-stage ATSVQ.

The goal of each stage of CM-TSVQ after the first stage is to estimate the expected coding error from all the preceding quantizers and refine the quantization by adding a codeword to the existing quantization. In this section, we examine how noiseless feedback affects the optimality conditions for the second stage ATSVQ. When deriving the necessary conditions for optimality, we will assume that $\mathcal{S}^{(1)}$ and $\mathcal{C}^{(1)}$ are fixed. Consider the system's block diagram in Figure 3.1. The ATSVQ encoder and decoder can be described by the following functions:

$$\mathcal{E}^{(2,AT)} : \mathbb{R}^k \times \mathcal{I}^{(1)} \times \mathcal{I}^{(1)} \to \mathcal{I}^{(2)} \tag{3.1}$$

$$\mathcal{D}^{(2,AT)} : \mathcal{I}^{(2)} \times \mathcal{I}^{(1)} \to \mathbb{R}^k. \tag{3.2}$$

Given the channel feedback output $Y_1 = y_1$, the expected distortion of the second stage quantizer is

$$\begin{aligned}
&E[d(\mathbf{U}, \mathbf{c}_{Y_1}^{(1,AT)} + \mathbf{c}_{Y_2|Y_1}^{(2,AT)})|Y_1 = y_1] \\
&\quad = \sum_{y_2, x_2 \in \mathcal{I}^{(2)}} \sum_{x_1 \in \mathcal{I}^{(1)}} P(Y_2 = y_2 | X_2 = x_2, Y_1 = y_1, X_1 = x_1) \\
&\qquad \times \int_{S_{x_1}^{(1,AT)} \cap S_{x_2|y_1}^{(2,AT)}} d(\mathbf{u}, \mathbf{c}_{y_1}^{(1,AT)} + \mathbf{c}_{y_2|y_1}^{(2,AT)}) f_{\mathbf{U}|Y_1}(\mathbf{u}|y_1) d\mathbf{u}.
\end{aligned}$$

The expected distortion of the entire ATSVQ (without knowledge of Y_1) is

$$E[d(\mathbf{U}, \mathbf{c}_{Y_1}^{(1,AT)} + \mathbf{c}_{Y_2|Y_1}^{(2,AT)})] = \sum_{y_1 \in \mathcal{I}^{(1)}} E[d(\mathbf{U}, \mathbf{c}_{Y_1}^{(1,AT)} + \mathbf{c}_{Y_2|Y_1}^{(2,AT)})|Y_1 = y_1] P(Y_1 = y_1) \tag{3.3}$$

$$= \sum_{y_2, x_2 \in \mathcal{I}^{(2)}} \sum_{y_1, x_1 \in \mathcal{I}^{(1)}} P(Y_2 = y_2 | X_2 = x_2, Y_1 = y_1, X_1 = x_1)$$
$$\times \int_{S_{x_1}^{(1,AT)} \cap S_{x_2|y_1}^{(2,AT)}} d(\mathbf{u}, \mathbf{c}_{y_1}^{(1,AT)} + \mathbf{c}_{y_2|y_1}^{(2,AT)}) f_{\mathbf{U}|Y_1}(\mathbf{u}|y_1) d\mathbf{u}. \quad (3.4)$$

Given feedback y_1, the modified distortion function is

$$d_2'(\mathbf{u}; \mathcal{E}^{(1,AT)}(\mathbf{u}), x_2, y_1)$$
$$= \sum_{y_2 \in \mathcal{I}^{(2)}} P(Y_2 = y_2 | X_2 = x_2, Y_1 = y_1, X_1 = \mathcal{E}^{(1,AT)}(\mathbf{u})) \left\| \mathbf{u} - \mathbf{c}_{y_1}^{(1,AT)} - \mathbf{c}_{y_2|y_1}^{(2,AT)} \right\|^2. \quad (3.5)$$

Thus as in the ACOVQ case, we can rewrite the expected distortion of the second stage quantizer as

$$E[d(\mathbf{U}, \mathbf{c}_{Y_1}^{(1,AT)} + \mathbf{c}_{Y_2|Y_1}^{(1,AT)}) | Y_1 = y_1]$$
$$= \sum_{x_1 \in \mathcal{I}^{(1)}} \sum_{x_2 \in \mathcal{I}^{(2)}} \int_{S_{x_1}^{(1,AT)} \cap S_{x_2|y_1}^{(2,AT)}} d'(\mathbf{u}; x_1, x_2, y_1) f_{\mathbf{U}|Y_1}(\mathbf{u}|y_1) d\mathbf{u} \quad (3.6)$$
$$= \sum_{y_2 \in \mathcal{I}^{(\in)}} E[d(\mathbf{U}, \mathbf{c}_{Y_1}^{(1,AT)} + \mathbf{c}_{Y_2|Y_1}^{(1,AT)}) | Y_1 = y_1, Y_2 = y_2] P(Y_2 = y_2 | Y_1 = y_1). \quad (3.7)$$

It then follows from (3.7) that for fixed codebook $\mathcal{C}_{y_1}^{(2)}$, the partition of the optimal second stage quantizer satisfies the following generalized nearest neighbor condition:

$$S_{x_2|y_1}^{(2)} = \{\mathbf{u} \in \mathbb{R}^k : d_2'(\mathbf{u}; \mathcal{E}^{(1)}(\mathbf{u}), x_2, y_1) \leq d_2'(\mathbf{u}; \mathcal{E}^{(1)}(\mathbf{u}), x_2', y_1), x_2' \in \mathcal{I}^{(2)}\},$$
$$x_2 \in \mathcal{I}^{(2)}, y_1 \in \mathcal{I}^{(1)}. \quad (3.8)$$

It also follows from (3.6) that for fixed partition $\mathcal{S}_{y_1}^{(2,AT)}$, the codebook of the optimal second stage quantizer satisfies the following generalized centroid condition:

$$\mathbf{c}_{y_2|y_1}^{(2,AT)} = \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^k} E[d(\mathbf{U}, \mathbf{c}_{Y_1}^{(1,AT)} + \boldsymbol{\omega})|Y_1 = y_1, Y_2 = y_2]. \tag{3.9}$$

Under the square error distortion, this reduces to

$$\mathbf{c}_{y_2|y_1}^{(2,AT)} = E[\mathbf{U}|Y_2 = y_2, Y_1 = y_1] - \mathbf{c}_{y_1}^{(1,AT)}. \tag{3.10}$$

3.2.2 Generalization for Multiple Stages

Here, we will derive the ATSVQ nearest neighbor and centroid conditions for stage $i \geq 2$. Assume that we have the received sequence $Y^{i-1} = y^{i-1}$ and fixed partitions and fixed codebooks $\mathcal{S}^{(1,AT)}, \mathcal{S}_{y_1}^{(2,AT)}, \ldots, \mathcal{S}_{y^{i-2}}^{(i-1,AT)}$ and $\mathcal{C}^{(1,AT)}, \mathcal{C}_{y_1}^{(2,AT)}, \ldots, \mathcal{C}_{y^{i-2}}^{(i-1,AT)}$, respectively. Let $\mathcal{S}_{y^{i-1}}^{(i,AT)}$ and $\mathcal{C}_{y^{i-1}}^{(i,AT)}$ be the partitions and codebook for the i-th stage quantizer with feedback $Y^{i-1} = y^{i-1}$. The encoder and decoder for the ATSVQ at the i-th stage can be characterized by the following functions:

$$\mathcal{E}^{(i,AT)} : \mathbb{R}^k \times \mathcal{I}^{i-1} \times \mathcal{I}^{i-1} \to \mathcal{I}^{(i)} \tag{3.11}$$

$$\mathcal{D}^{(i,AT)} : \mathcal{I}^i \to \mathbb{R}^k, \tag{3.12}$$

where $\mathcal{I}^i = \mathcal{I}^{(1)} \times \mathcal{I}^{(2)} \times \cdots \times \mathcal{I}^{(i)}$. Similar to the ACOVQ recursive encoding functions in (2.90) - (2.96), we recursively use $\zeta_{y^{i-1}}^{(i)}$ to denote the encoded sequence for the

i stages of ATSVQ. For $Y^{i-1} = y^{i-1}$, let

$$\zeta^{(1)}(\mathbf{u}) = \mathcal{E}^{(1,AT)}(\mathbf{u}) \tag{3.13}$$

$$\zeta^{(2)}_{y^1}(\mathbf{u}) = \mathcal{E}^{(2,AT)}(\mathcal{E}^{(1,AT)}(\mathbf{u}), y_1, \mathbf{u}) \tag{3.14}$$

$$= \mathcal{E}^{(2,AT)}(\zeta^{(1)}(\mathbf{u}), y_1, \mathbf{u}) \tag{3.15}$$

$$\vdots \tag{3.16}$$

$$\zeta^{(i)}_{y^{i-1}}(\mathbf{u}) = \mathcal{E}^{(i,AT)}(\zeta^{(i-1)}_{y^{i-2}}(\mathbf{u}), \ldots, \zeta^{(1)}(\mathbf{u}), y^{i-1}, \mathbf{u}). \tag{3.17}$$

That is, for the received sequence y^{i-1}, $\zeta^{(j)}_{y^{j-1}}(\mathbf{u})$ represents the encoded index at stage j for all $j = 1, \ldots, i$. Let $\zeta^{i}_{y^{i-1}}(\mathbf{u}) = (\zeta^{(i)}_{y^{i-1}}(\mathbf{u}), \zeta^{(i-1)}_{y^{i-2}}(\mathbf{u}), \ldots, \zeta^{(1)}(\mathbf{u}))$. Given feedback $Y^{i-1} = y^{i-1}$, we have that the expected distortion at the ith stage is given by

$$E[d(\mathbf{U}, Q(\mathbf{U}))|Y^{i-1} = y^{i-1}]$$

$$= \sum_{y_i \in \mathcal{I}^{(1)}} \sum_{x^i \in \mathcal{I}^i} \int_{\alpha(x^{i-1})} P(Y_i = y_i | Y^{i-1} = y^{i-1}, X^i = x^i) \tag{3.18}$$

$$\times\, d(\mathbf{u}, \mathbf{c}^{(1,AT)}_{y_1} + \mathbf{c}^{(2,AT)}_{y_2|y_1} + \cdots + \mathbf{c}^{(i,AT)}_{y_i|y^{i-1}}) f_{\mathbf{U}|\mathbf{Y^{i-1}}}(\mathbf{u}|y^i) d\mathbf{u} \tag{3.19}$$

$$= \sum_{y_i \in \mathcal{I}^{(1)}} \sum_{x^i \in \mathcal{I}^i} \int_{\alpha(x^{i-1})} P(Y_i = y_i | Y^{i-1} = y^{i-1}, X^i = x^i)$$

$$\times \left\| \mathbf{u} - (\mathbf{c}^{(1,AT)}_{y_1} + \mathbf{c}^{(2,AT)}_{y_2|y_1} + \cdots + \mathbf{c}^{(i,AT)}_{y_i|y^{i-1}}) \right\|^2 f_{\mathbf{U}|\mathbf{Y^{i-1}}}(\mathbf{u}|y^i) d\mathbf{u} \tag{3.20}$$

where $\alpha(x^{i-1}) = S^{(i-1,AT)}_{x_{i-1}|y^{i-2}} \cap S^{(i-2,AT)}_{x_{i-2}|y^{i-3}} \cap S^{(i-3,AT)}_{x_{i-3}|y^{i-4}} \cap \cdots \cap S^{(1,AT)}_{x_1}$, which are the encoding regions corresponding to the sequence $X^{i-1} = x^{i-1}$ given feedback $Y^{i-2} = y^{i-2}$. Let $d'_i(\mathbf{u}; x_i, \zeta^{i-1}_{y^{i-2}}(\mathbf{u}), y^{i-1})$ represent the modified distortion at the ith stage, which can be expressed as

$$\begin{aligned}
&d_i'(\mathbf{u}; x_i, \zeta_{y^{i-2}}^{i-1}(\mathbf{u}), y^{i-1}) \\
&\quad = \sum_{y_i \in \mathcal{I}^{(i)}} P(Y_i = y_i | X_i = x_i, X^{i-1} = \zeta_{y^{i-2}}^{i-1}(\mathbf{u}), Y^{i-1} = y^{i-1}) \\
&\qquad\qquad \times d(\mathbf{u}, \mathbf{c}_{y_1}^{(1,AT)} + \mathbf{c}_{y_2|y_1}^{(2,AT)} + \cdots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)}) && (3.21) \\
&\quad = \sum_{y_i \in \mathcal{I}^{(i)}} P(Y_i = y_i | X_i = x_i, X^{i-1} = \zeta_{y^{i-2}}^{i-1}(\mathbf{u}), Y^{i-1} = y^{i-1}) \\
&\qquad\qquad \times \left\| \mathbf{u} - (\mathbf{c}_{y_1}^{(1,AT)} + \mathbf{c}_{y_2|y_1}^{(2,AT)} + \cdots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)}) \right\|^2 . && (3.22)
\end{aligned}$$

The expected distortion can then be rewritten as

$$\begin{aligned}
&E[d(\mathbf{U}, Q(\mathbf{U})) | Y^{i-1} = y^{i-1}] \\
&\quad = \sum_{x^i \in \mathcal{I}^i} \int_{\alpha(x^{i-1})} d'(\mathbf{u}; x_i, x^{i-1}, y^{i-1}) f_{\mathbf{U}|Y^{i-1}}(\mathbf{u}|y^{i-1}) d\mathbf{u} && (3.23) \\
&\quad = \sum_{y^i \in \mathcal{I}^{(i)}} P(Y_i = y_i | Y^{i-1} = y^{i-1}) E[d(\mathbf{U}, \mathbf{c}_{y_1}^{(1,AT)} + \cdots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)}) | Y^i = y^i]. && (3.24)
\end{aligned}$$

It follows from (3.23) that for fixed codebook $\mathcal{C}_{y^{i-1}}^{(i,AT)}$, the optimal i-th stage quantizer will have encoding regions satisfying

$$S_{x_i|y^{i-1}}^{(i,AT)} = \{\mathbf{u} \in \mathbb{R}^k : d_i'(\mathbf{u}; x_i, \zeta_{y^{i-2}}^{i-1}(\mathbf{u}), y^{i-1}) \le d_i'(\mathbf{u}; x_i', \zeta_{y^{i-2}}^{i-1}(\mathbf{u}), y^{i-1}), \quad x_i' \in \mathcal{I}^{(i)}\},$$
$$x_i \in \mathcal{I}^{(i)}, y^{i-1} \in \mathcal{I}^{i-1}. \qquad (3.25)$$

It also follows from (3.24) that for fixed partition $\mathcal{S}^{(i,AT)}_{y^{i-1}}$, the optimal i-th stage quantizer will have a codebook with codewords given by

$$\mathbf{c}^{(i,AT)}_{y_i|y^{i-1}} = \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^k} E[d(\mathbf{U}, \mathbf{c}^{(1,AT)}_{y_1} + \cdots + \mathbf{c}^{(i,AT)}_{y_i|y^{i-1}} + \boldsymbol{\omega})|Y^i = y^i], \tag{3.26}$$

which under the square error distortion, becomes (as shown in Appendix A)

$$\mathbf{c}^{(i,AT)}_{y_i|y^{i-1}} = E[\mathbf{U}|Y^i = y^i] - (\mathbf{c}^{(1,AT)}_{y_1} + \cdots + \mathbf{c}^{(i,AT)}_{y_i|y^{i-1}}). \tag{3.27}$$

In addition to the tree like structures present, we can already see that ACOVQ and ATSVQ share similarities in their generalized nearest neighbor and centroid conditions. In the next section, we show that these conditions are indeed equivalent.

3.3 Equivalence of ATSVQ and ACOVQ

The *overall* encoder and decoder at stage i of ATSVQ is given by

$$\mathcal{E}^i_{AT} : \mathbb{R}^k \times \mathcal{I}^{i-1} \to \mathcal{I}^i, \text{ such that } \mathcal{E}^i_{AT}(\mathbf{u}, y^{i-1}) = \zeta^i_{y^{i-1}}(\mathbf{u}) \tag{3.28}$$

$$\mathcal{D}^i_{AT} : \mathcal{I}^i \to \mathbb{R}^k, \text{ such that}$$

$$\mathcal{D}^i_{AT}(y^i) = \mathcal{D}^{(1,AT)}(y_1) + \mathcal{D}^{(2,AT)}(y_2, y_1) + \cdots + \mathcal{D}^{(i,AT)}(y^i), \tag{3.29}$$

and let the *overall* encoder and decoder at stage i of the ACOVQ be defined as

$$\mathcal{E}^i_{AC} : \mathbb{R}^k \times \mathcal{I}^{i-1} \to \mathcal{I}^i, \text{ such that } \mathcal{E}^i_{AC}(\mathbf{u}, y^{i-1}) = \gamma^i_{y^{i-1}}(\mathbf{u}) \tag{3.30}$$

$$\mathcal{D}^i_{AC} : \mathcal{I}^i \to \mathbb{R}^k, \text{ such that } \mathcal{D}^i_{AC}(y^i) = \mathcal{D}^{(i,AC)}(y^i). \tag{3.31}$$

In this section, we show that the generalized nearest neighbor and centroid conditions of the ATSVQ and ACOVQ will lead to equivalent overall encoders under certain conditions.

3.3.1 Conditions for Equivalence

Let $\mathcal{C}^{(1,AC)}, \mathcal{C}^{(2,AC)}_{y_1}$ and $\mathcal{S}^{(1,AC)}, \mathcal{S}^{(2,AC)}_{y_1}$ be the codebook and partitions for a 2 stage ACOVQ, respectively, and let $\mathcal{C}^{(1,AT)}, \mathcal{C}^{(2,AT)}_{y_1}$ and $\mathcal{S}^{(1,AT)}, \mathcal{S}^{(2,AT)}_{y_1}$ be the codebook and partitions for a 2 stage ATSVQ, respectively, with channel output $Y_1 = y_1$ received at the encoders via a noiseless feedback link. We assume that the bit allocations for both quantizers and the first stage quantizers for ATSVQ and ACOVQ are equivalent (i.e., $\mathcal{C}^{(1,AC)} = \mathcal{C}^{(1,AT)}$ and $\mathcal{S}^{(1,AC)} = \mathcal{S}^{(1,AT)}$). Consider the ATSVQ generalized centroid condition given in (3.9), which states that for a fixed first stage quantizer and fixed partitions $\mathcal{S}^{(2,AT)}_{y_1}$ an optimal second stage quantizer must have a codebook that satisfies

$$\mathbf{c}^{(2,AT)}_{y_2|y_1} = \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^k} E[d(\mathbf{U}, \mathbf{c}^{(1,AT)}_{Y_1} + \boldsymbol{\omega})|Y_1 = y_1, Y_2 = y_2] \tag{3.32}$$

$$= \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^k} E[d(\mathbf{U}, \mathbf{c}^{(1,AT)}_{y_1} + \boldsymbol{\omega})|Y_1 = y_1, Y_2 = y_2]. \tag{3.33}$$

Let $z = \mathbf{c}_{y_1}^{(1,AT)} + \boldsymbol{\omega} \implies \boldsymbol{\omega} = z - \mathbf{c}_{y_1}^{(1,AT)}$. Since $\mathbf{c}_{y_1}^{(2,AT)}$ is constant given $Y_1 = y_1$, we have (after a change of variables) that

$$\mathbf{c}_{y_2|y_1}^{(2,AT)} = \arg\min_{z-\mathbf{c}_{y_1}^{(1,AT)}} E[d(\mathbf{U}, z)|Y_1 = y_1, Y_2 = y_2] \tag{3.34}$$

$$\implies \mathbf{c}_{y_2|y_1}^{(2,AT)} + \mathbf{c}_{y_1}^{(1,AT)} = \arg\min_{z\in\mathbb{R}^k} E[d(\mathbf{U}, z)|Y_1 = y_1, Y_2 = y_2], \tag{3.35}$$

which under square error distortion becomes

$$\mathbf{c}_{y_2|y_1}^{(2,AT)} + \mathbf{c}_{y_1}^{(1,AT)} = E[\mathbf{U}|Y_1 = y_1, Y_2 = y_2], \tag{3.36}$$

Note that the right-hand side of (3.36) is exactly the centroid condition of ACOVQ in (2.87). The value of the expectation is solely defined by the partitions in the first and second stage and the channel. Thus, assuming that the partitions of ATSVQ and ACOVQ are equivalent and satisfy the nearest neighbor condition, under the square error distortion we have that $\mathbf{c}_{y_2|y_1}^{(2,AT)} + \mathbf{c}_{y_1}^{(2,AT)} = \mathbf{c}_{y_2|y_1}^{(2,AC)}$ for all $y_2 \in \mathcal{I}^{(2)}$. Note that this implies that the overall second stage decoders are equivalent (i.e., $\mathcal{D}_{AT}^2(y_1, y_2) = \mathcal{D}_{AC}^2(y_1, y_2)$).

Now assume that $\mathbf{c}_{y_2|y_1}^{(2,AT)} + \mathbf{c}_{y_1}^{(2,AT)} = \mathbf{c}_{y_2|y_1}^{(2,AC)}$ for all $y_2 \in \mathcal{I}^{(2)}$ (i.e., equivalent overall decoders). The generalized nearest neighbor condition states that for a fixed first stage quantizer and fixed codebook $\mathcal{C}_{y_2}^{(2,AT)}$, the optimal second stage quantizer for

ATSVQ will have partitions that satisfy

$$S^{(2,AT)}_{x_2|y_1} = \{\mathbf{u} \in \mathbb{R}^k : d'_2(\mathbf{u}; \mathcal{E}^{(1,AT)}(\mathbf{u}), x_2, y_1) \leq d'_2(\mathbf{u}; \mathcal{E}^{(1,AT)}(\mathbf{u}), x'_2, y_1), \quad x'_2 \in \mathcal{I}^{(2)}\},$$
$$x_2 \in \mathcal{I}^{(2)}, y_1 \in \mathcal{I}^{(1)}, \tag{3.37}$$

where the modified distortion is given by

$$\begin{aligned} &d'_2(\mathbf{u}; \mathcal{E}^{(1,AT)}(\mathbf{u}), x_2, y_1) \\ &= \sum_{y_2 \in \mathcal{I}^{(2)}} P(Y_2 = y_2 | X_2 = x_2, Y_1 = y_1, X_1 = \mathcal{E}^{(1,AT)}(\mathbf{u})) \left\| \mathbf{u} - \mathbf{c}^{(1,AT)}_{y_1} - \mathbf{c}^{(2,AT)}_{y_2|y_1} \right\|^2 . \end{aligned} \tag{3.38}$$

With the assumption that $\mathbf{c}^{(2,AT)}_{y_2|y_1} + \mathbf{c}^{(2,AT)}_{y_1} = \mathbf{c}^{(2,AC)}_{y_2|y_1}$ for all $y_2 \in \mathcal{I}^{(2)}$ and equivalent first stage encoders, the modified distortion can be rewritten as

$$\begin{aligned} &d'_2(\mathbf{u}; \mathcal{E}^{(1,AT)}(\mathbf{u}), x_2, y_1) \\ &= \sum_{y_2 \in \mathcal{I}^{(2)}} P(Y_2 = y_2 | X_2 = x_2, Y_1 = y_1, X_1 = \mathcal{E}^{(1,AC)}(\mathbf{u})) \left\| \mathbf{u} - \mathbf{c}^{(2,AC)}_{y_2|y_1} \right\|^2 , \end{aligned} \tag{3.39}$$

which is the modified distortion for ACOVQ in (2.82), implying that the second stage generalized nearest neighbor conditions are equivalent.

Note that the codebooks and partitions satisfying the generalized nearest neighbor and centroid conditions may not be unique, but may be chosen arbitrarily without affecting the expected distortion. The consequences of these properties is that when initializing the second stage with either the same partitions or codebooks such that $\mathbf{c}_{y_2|y_1}^{(2,AT)} + \mathbf{c}_{y_1}^{(2,AT)} = \mathbf{c}_{y_2|y_1}^{(2,AC)}$ for all $y_2 \in \mathcal{I}^{(2)}$ and applying the generalized LBG-algorithm, we can see that ATSVQ and ACOVQ will converge to equivalent encoders and decoders. Consequently, this shows that in the communication model with feedback, there is no difference between successively refining by quantizing the coding error and refining by quantizing a distribution conditioned on feedback. A similar proof can be done in the general case for a quantizer at stage i with received sequence $Y^{i-1} = y^{i-1}$.

3.4 Simulations Results

In this section, we present the results of experimental simulations evaluating the performance of ACOVQ and demonstrate the empirical equivalence between the ATSVQ and ACOVQ. We begin by describing the channel model and the computation of block transition probabilities, followed by a presentation of the simulation results. Up to this point, the transition probabilities for the channel indices have been kept general. However, only the binary Polya contagion channel with Markov memory 1 ($M = 1$) is considered for simulations in this thesis. In the following section, we detail how these transition probabilities are assigned.

3.4.1 Channel Properties

The way that the channel indices are transmitted in the simulations is that an input index is converted into a binary vector. Afterwards the vector is transmitted through the Polya contagion channel bit-by-bit. Also, if the channel has memory, the channel memory is preserved between stages (i.e., the channel noise in the current stage is correlated with the channel noise in the previous stages). Let $\beta_l : \mathbb{N} \to \{0,1\}^l$ be a function that outputs the l-bit binary representation of an integer (e.g., $\beta_4(3) = (0,0,1,1)$), and let $\oplus$ represent modulo-2 bit-wise addition. Recall from Chapter 2 that the Polya channel with memory $M = 1$ is a discrete channel where the channel noise $\{Z_i\}_{i=1}^{\infty}$ is characterized by $P(Z_i = 1) = \epsilon$, for $i = 1$, and

$$P(Z_i = 1 | Z_{i-1} = e_{i-1}, \ldots, Z_1 = e_1) = P(Z_i = 1 | Z_{i-1} = e_{i-1}) \tag{3.40}$$

$$= \frac{\epsilon + e_{i-1}\delta}{1 + \delta}, \tag{3.41}$$

for $i \geq 2$, where $e_j \in \{0,1\}, j = 1, 2, \ldots, i-1$. Let $\mathbf{b} = (b_1, \ldots, b_i)$ be the bit allocation vector and let $x^i \in \mathcal{I}^i$ and $y^{i-1} \in \mathcal{I}^{i-1}$ represent the encoded and received indices respectively. Let $B_j = \sum_{l=1}^{j} b_l$ and let $Z_l^p = (Z_{l+1}, Z_{l+2}, \ldots, Z_p)$, for $l < p$, and $Z^l = (Z_1, \ldots, Z_l)$ for $l \geq 2$. Then the binary noise at stage-j can be represented as

$$Z_{B_j}^{B_{j+1}} = \beta_{b_j}(x_j) \oplus \beta_{b_j}(y_j). \tag{3.42}$$

Note that we can also "concatenate" the binary representation of the channel indices as follows

$$Z^{B_j} = \beta_{B_j}\left(\sum_{l=1}^{j} x_l \cdot 2^{B_l}\right) \oplus \beta_{B_j}\left(\sum_{l=1}^{j} y_l \cdot 2^{B_l}\right), \tag{3.43}$$

for all $j = 1, \ldots, i$. Note that $\left(\sum_{l=1}^{j} x_l \cdot 2^{B_l}\right)$ and $\left(\sum_{l=1}^{j} y_l \cdot 2^{B_l}\right)$ are integers in $\{0, 1, \ldots, 2^{B_j} - 1\}$. The results in (3.43) are equivalent to calculating the entire noise sequence using (3.42), but (3.43) can serve as a practical alternative to calculating the binary sequence. We then have that the transition probability of $Y_i = y_i$ given $X^i = x^i$ and $Y^{i-1} = y^{i-1}$ is

$$\begin{aligned}
&P(Y_i = y_i | X^i = x^i, Y^{i-1} = y^{i-1}) \\
&\quad = P(Z_{B_{i-1}}^{B_i} = \beta_{b_i}(x_i) \oplus \beta_{b_i}(y_i) | Z_{B_{i-2}}^{B_{i-1}} = \beta_{b_{i-1}}(x_{i-1}) \oplus \beta_{b_i - 1}(y_{i-1}), \ldots, \\
&\qquad\qquad Z^{b_1} = \beta_{b_1}(x_1) \oplus \beta_{b_1}(y_1)) \qquad (3.44) \\
&\quad = P\left(Z_{B_{i-1}}^{B_i} = \beta_{b_i}(x_i) \oplus \beta_{b_i}(y_i) \middle| Z^{B_{i-1}} = \beta_{B_{i-1}} \left(\sum_{l=1}^{i-1} x_l \cdot 2^{B_l} \right) \right. \\
&\qquad\qquad \left. \oplus\, \beta_{B_{i-1}} \left(\sum_{l=1}^{i-1} y_l \cdot 2^{B_l} \right) \right). \qquad (3.45)
\end{aligned}$$

Let $z_{B_{i-1}}^{B_i} = \beta_{b_i}(x_i) \oplus \beta_{b_i}(y_i)$ and $z^{B_{i-1}} = \beta_{B_{i-1}} \left(\sum_{l=1}^{i-1} x_l \cdot 2^{B_l}\right) \oplus \beta_{B_{i-1}} \left(\sum_{l=1}^{i-1} y_l \cdot 2^{B_l}\right)$. For a Polya channel with memory ($M = 1$), we then have that (3.45) evaluates to

$$\begin{aligned}
P(Z_{B_{i-1}}^{B_i} = z_{B_{i-1}}^{B_i} | Z^{B_{i-1}} = z^{B_{i-1}}) &= P(Z_{B_{i-1}}^{B_i} = z_{B_{i-1}}^{B_i} | Z_{B_{i-1}} = z_{B_{i-1}}) \qquad (3.46) \\
&= \prod_{j=B_{i-1}+1}^{B_i} P(Z_j = z_j | Z_{j-1} = z_{j-1}). \qquad (3.47)
\end{aligned}$$

Expression (3.45) will then be used to calculate the transition probabilities used in the modified distortion calculations in the generalized nearest neighbor conditions and when generating noise samples in the simulations.

3.4.2 ACOVQ and ATSVQ Performance Results

ACOVQ Results

The following tables show the results of ACOVQ for various bit allocations. In these simulations, the initial codebook values were determined by the VQ generalized Lloyd's algorithm with indexing determined by the simulated annealing algorithm. Each quantizer in the ACOVQ is then trained using the generalized LBG-algorithm along a sequence of increasing and decreasing channel error rate ϵ values ranging from 0 and 0.1, using the increase-decrease method in [12], while keeping the noise correlation parameter δ fixed. The highest performing quantizer for each set of channel parameters is reported in Tables 3.1 - 3.6. The performance of each quantizer here and throughout this thesis will be measured by its signal-to-noise ratio (SNR) in decibels (dB), which is defined as

$$SNR = 10 \log_{10} \left(\frac{\sigma^2}{\frac{1}{k} E[(\mathbf{U} - Q(\mathbf{U}))^2]} \right), \tag{3.48}$$

where σ^2 is the variance of the source distribution. Note that the highest SNR value in each row is put in bold. The source used in each simulation is a memoryless, independently and identically distributed Gaussian source.

ϵ	(4)	(1,3)	(2,2)	(3,1)	(1,1,2)	(1,2,1)	(2,1,1)	(1,1,1,1)
0.0000	**20.221**	20.217	20.134	20.160	20.136	20.205	20.081	20.116
0.0005	18.658	19.439	19.373	19.042	19.649	19.608	19.372	**19.664**
0.0010	17.601	18.765	18.713	18.318	19.200	19.119	18.776	**19.274**
0.0050	14.373	15.793	15.940	14.809	16.738	16.452	15.782	**16.955**
0.0100	12.652	14.072	14.224	13.308	15.060	14.617	14.423	**15.199**
0.0500	8.292	9.205	9.132	9.375	9.778	9.895	9.914	**10.404**
0.1000	5.888	6.296	6.410	6.579	6.813	6.909	7.092	**7.397**

Table 3.1: ACOVQ SNR performance (in dB) on a 1-dimensional memoryless Gaussian source ($k = 1$) for $\delta = 0$ and various bit allocations and ϵ values.

ϵ	(4)	(1,3)	(2,2)	(3,1)	(1,1,2)	(1,2,1)	(2,1,1)	(1,1,1,1)
0.0000	20.223	**20.227**	20.127	20.154	20.153	20.204	20.096	20.099
0.0005	19.599	19.713	**19.736**	19.726	19.673	19.704	19.723	19.684
0.0010	19.129	19.259	19.333	**19.393**	19.290	19.195	19.379	19.253
0.0050	16.695	16.868	17.198	17.284	16.960	16.888	**17.359**	17.013
0.0100	15.229	15.209	15.769	15.605	15.574	15.149	**15.772**	15.333
0.0500	11.069	11.366	11.504	11.047	11.668	11.227	11.760	**11.837**
0.1000	8.942	8.901	9.366	9.077	9.594	9.345	9.932	**9.995**

Table 3.2: ACOVQ SNR performance (in dB) on a 1-dimensional memoryless Gaussian source ($k = 1$) for $\delta = 5$ and various bit allocations and ϵ values.

ϵ	(4)	(1,3)	(2,2)	(3,1)	(1,1,2)	(1,2,1)	(2,1,1)	(1,1,1,1)
0.0000	20.218	**20.228**	20.126	20.199	20.134	20.191	20.099	20.107
0.0005	19.865	19.767	19.845	**19.897**	19.786	19.772	19.836	19.741
0.0010	19.575	19.349	19.536	**19.671**	19.414	19.446	19.645	19.429
0.0050	17.786	17.295	17.970	**18.126**	17.554	17.421	18.060	17.442
0.0100	16.466	16.312	16.690	**16.833**	16.592	15.950	16.732	16.307
0.0500	12.666	12.144	13.188	12.474	12.912	12.689	**13.303**	13.000
0.1000	10.342	10.260	11.255	10.539	11.224	10.739	**11.657**	11.511

Table 3.3: ACOVQ SNR performance (in dB) on a 1-dimensional memoryless Gaussian source ($k = 1$) for $\delta = 10$ and various bit allocations and ϵ values.

ϵ	(4)	(1,3)	(2,2)	(3,1)	(1,1,2)	(1,2,1)	(2,1,1)	(1,1,1,1)
0	**9.674**	9.588	9.393	9.095	9.387	9.062	9.281	9.303
0.0005	**9.527**	9.471	9.335	9.032	9.339	9.010	9.247	9.250
0.001	**9.454**	9.403	9.282	8.968	9.280	8.959	9.196	9.197
0.005	8.791	**8.904**	8.869	8.511	8.871	8.573	8.791	8.795
0.01	8.152	8.345	8.410	8.022	**8.414**	8.146	8.349	8.350
0.05	5.530	5.806	6.037	5.709	**6.037**	5.860	6.020	6.008
0.1	3.875	4.103	4.308	4.066	4.309	4.231	4.306	**4.314**

Table 3.4: ACOVQ SNR performance (in dB) on a 2-dimensional memoryless Gaussian source ($k = 2$) for $\delta = 0$ and various bit allocations and ϵ values.

ϵ	(4)	(1,3)	(2,2)	(3,1)	(1,1,2)	(1,2,1)	(2,1,1)	(1,1,1,1)
0	**9.685**	9.588	9.394	9.093	9.390	9.063	9.301	9.303
0.0005	**9.635**	9.543	9.358	9.065	9.360	9.028	9.270	9.272
0.001	**9.587**	9.507	9.324	9.187	9.329	8.998	9.237	9.236
0.005	**9.258**	9.178	9.078	8.804	9.086	8.775	8.989	8.991
0.01	**8.972**	8.902	8.786	8.544	8.808	8.502	8.692	8.708
0.05	7.184	7.066	7.181	7.095	**7.199**	6.933	7.030	7.026
0.1	5.920	5.975	**6.112**	6.069	6.109	5.928	5.902	5.918

Table 3.5: ACOVQ SNR performance (in dB) on a 2-dimensional memoryless Gaussian source ($k = 2$) for $\delta = 5$ and various bit allocations and ϵ values.

ϵ	(4)	(1,3)	(2,2)	(3,1)	(1,1,2)	(1,2,1)	(2,1,1)	(1,1,1,1)
0	**9.680**	9.589	9.393	9.093	9.391	9.064	9.301	9.301
0.0005	**9.641**	9.551	9.369	9.221	9.363	9.038	9.269	9.280
0.001	**9.612**	9.492	9.336	9.204	9.343	9.011	9.254	9.252
0.005	**9.369**	9.267	9.127	8.876	9.147	8.826	9.054	9.057
0.01	**9.167**	9.038	8.916	8.837	8.904	8.610	8.824	8.826
0.05	7.566	7.607	7.601	**7.638**	7.588	7.314	7.421	7.424
0.1	6.684	6.686	6.824	6.605	**6.824**	6.562	6.354	6.268

Table 3.6: ACOVQ SNR performance (in dB) on a 2-dimensional memoryless Gaussian source ($k = 2$) for $\delta = 10$ and various bit allocations and ϵ values.

The results indicate that as the channel gets noisier, ACOVQ benefits from additional stages of feedback. Furthermore, as δ increases, the performance of ACOVQ improves regardless of the bit allocation, suggesting that the quantizer is able to exploit the memory in the channel.

ACOVQ and ATSVQ results

We next train ACOVQ and ATSVQ on a sample of 4 million vectors drawn from a memoryless Gaussian source, using the generalized LBG-algorithm. Each stage is trained under equivalent channel parameters and equivalent initializations— i.e., settings in which the nearest-neighbor and centroid conditions are equivalent for ACOVQ and ATSVQ, as discussed in Section 3.3.1. After training the final codebooks of each quantizer are compared. The comparison is based on the maximum codeword distance: the largest Euclidean distance between any pair of corresponding codewords for the ACOVQ and ATSVQ codebooks, evaluated across all stages and channel output sequences. This is defined as:

$$\arg\max_{j\in\{1,\dots,n\},y^j\in\mathcal{I}^j} \left\| \mathbf{c}^{(j,AC)}_{y_j|y^{j-1}} - \left(\mathbf{c}^{(j,AT)}_{y_j|y^{j-1}} + \mathbf{c}^{(j-1,AT)}_{y_{j-1}|y^{j-2}} + \cdots + \mathbf{c}^{(1,AT)}_{y_1} \right) \right\|. \tag{3.49}$$

Simulation results for various channel parameters and bit allocations are presented in Tables 3.7 - 3.10.

Dim. k	δ	ϵ	SNR (TSVQ)	SNR (ACOVQ)	Greatest Codeword Distance
1	0	0.00	20.087198	20.087198	2.47×10^{-14}
1	0	0.05	10.375423	10.367799	8.20×10^{-3}
1	0	0.10	7.376438	7.382376	3.87×10^{-2}
1	5	0.00	20.082411	20.082411	2.93×10^{-14}
1	5	0.05	11.615976	11.586737	3.07×10^{-2}
1	5	0.10	9.977197	9.984788	9.77×10^{-3}
4	0	0.00	4.395605	4.395605	3.75×10^{-14}
4	0	0.05	3.147361	3.149369	1.28×10^{-2}
4	0	0.10	2.274173	2.274005	2.08×10^{-2}
4	5	0.00	4.399283	4.399283	2.80×10^{-14}
4	5	0.05	3.728805	3.727707	8.46×10^{-3}
4	5	0.10	3.221204	3.219431	1.31×10^{-2}

Table 3.7: Comparison of ACOVQ and ATSVQ SNRs and codebooks for bit allocation $(1, 1, 1, 1)$ and memoryless Gaussian source.

Dim. k	δ	ϵ	SNR (TSVQ)	SNR (ACOVQ)	Greatest Codeword Distance
1	0	0.00	20.169320	20.169320	2.73×10^{-14}
1	0	0.05	8.802735	8.800211	5.54×10^{-3}
1	0	0.10	6.040480	6.030800	5.14×10^{-3}
1	5	0.00	20.172459	20.169811	4.31×10^{-5}
1	5	0.05	10.566610	10.582667	1.43×10^{-2}
1	5	0.10	8.737369	8.725517	8.91×10^{-3}
4	0	0.00	4.460133	4.460133	4.78×10^{-14}
4	0	0.05	3.052052	3.051580	5.00×10^{-2}
4	0	0.10	2.158082	2.158576	3.81×10^{-2}
4	5	0.00	4.450385	4.450385	3.14×10^{-14}
4	5	0.05	3.582166	3.581317	1.30×10^{-2}
4	5	0.10	3.017881	3.022005	8.69×10^{-2}

Table 3.8: Comparison of ACOVQ and ATSVQ SNRs and codebooks for bit allocation $(1, 3)$ and memoryless Gaussian source.

Dim. k	δ	ϵ	SNR (TSVQ)	SNR (ACOVQ)	Greatest Codeword Distance
1	0	0.00	20.103408	20.103408	2.39×10^{-14}
1	0	0.05	8.535109	8.534380	4.55×10^{-2}
1	0	0.10	5.988914	5.993359	5.98×10^{-3}
1	5	0.00	20.127089	20.127089	3.05×10^{-14}
1	5	0.05	9.829185	9.849087	1.76×10^{-2}
1	5	0.10	8.829142	8.823401	8.92×10^{-3}
4	0	0.00	4.360361	4.360361	3.55×10^{-14}
4	0	0.05	2.927418	2.924842	2.15×10^{-2}
4	0	0.10	2.134878	2.135311	5.56×10^{-2}
4	5	0.00	4.359788	4.359788	3.59×10^{-14}
4	5	0.05	3.595184	3.589755	1.05×10^{-2}
4	5	0.10	3.057900	3.055866	1.43×10^{-2}

Table 3.9: Comparison of ACOVQ and ATSVQ SNRs and codebooks for bit allocation $(2, 2)$ and memoryless Gaussian source.

Dim. k	δ	ϵ	SNR (TSVQ)	SNR (ACOVQ)	Greatest Codeword Distance
1	0	0.00	20.125013	20.125013	1.86×10^{-14}
1	0	0.05	8.442011	8.468965	1.11×10^{-2}
1	0	0.10	6.445608	6.459866	6.74×10^{-3}
1	5	0.00	20.110080	20.110080	1.03×10^{-14}
1	5	0.05	10.539527	10.525425	1.26×10^{-2}
1	5	0.10	9.098137	9.089128	1.24×10^{-2}
4	0	0.00	4.190830	4.190830	2.29×10^{-14}
4	0	0.05	2.810843	2.817062	5.39×10^{-2}
4	0	0.10	2.210464	2.212517	1.27×10^{-2}
4	5	0.00	4.195522	4.195522	2.39×10^{-14}
4	5	0.05	3.466058	3.468024	9.11×10^{-3}
4	5	0.10	2.903744	2.905105	2.44×10^{-2}

Table 3.10: Comparison of ACOVQ and ATSVQ SNRs and codebooks for bit allocation $(3, 1)$ and memoryless Gaussian source.

As shown in Tables 3.7 - 3.10, regardless of the channel parameters, bit allocation or dimension, the maximum codeword distance is minimal relative to the source variance. This distance is notably smaller in the noiseless channel case than in the presence of channel noise. The discrepancy arises because the noise sequences for ATSVQ and ACOVQ are generated independently (i.e., separate noise sequence realizations are used for each quantizer). Nevertheless, even under noisy conditions, the distance remains modest, indicating that the equivalence between the codebooks is robust to different noise sequences from the same distribution. Additionally, the SNR performances for ACOVQ and TSVQ match within hundredths of a decibel in all cases. These results demonstrate that ACOVQ and ATSVQ, with the same initializations, produce equivalent encoders, decoders, and overall performances.

Chapter 4

Variable-Rate Adaptive Tree Structure Vector Quantization

4.1 Introduction

In the previous chapter, we showed that the necessary conditions for optimality for ACOVQ is equivalent to those of ATSVQ. Up to this point, ACOVQ has only been studied for fixed rates (i.e., all quantizers in a given stage have the same number of bits). However, the posterior distributions at each stage of the ACOVQ, in general, exhibit different variances and shapes, especially when the channel is noisy. This suggests that a better performance can be obtained if we allocate bits non-uniformly across the quantizers for that stage. In this chapter, we explore the performance of a variable-rate ACOVQ (VR-ACOVQ) and algorithms to find an optimal bit allocation for each stage given a constraint. We then compare the performance of

VR-ACOVQ to that of the fixed-rate ACOVQ (FR-ACOVQ) under the same average rate constraints.

4.2 Variable-Rate Quantization

In variable-rate quantization, the number of bits used to quantize a source can vary depending on the input source value. There are multiple well-studied methods for variable-rate quantization such as quadtree-based quantization, pruned tree structured quantization, and greedy tree growing quantization [15]. The central principle underlying these methods is that not all input vectors require a high rate quantization; consequently, different rate quantizers can be used based on the characteristics of the input vector. A common application of variable-rate vector quantization is image compression: fairly homogeneous regions of an image, such as a solid colored background, can be compressed at a low rate without incurring excessive distortion, while high rate quantization can be reserved for highly detailed regions, such as edges of objects. This allows the variable-rate quantizer to perform similarly to a high fixed-rate quantizer while using a lower average rate. The same principle can be applied in ACOVQ. As shown in Figure 2.4 not all posterior distributions have the same variances or shapes. This suggests that a nonuniform bit allocation for each posterior distribution of a given stage in the ACOVQ can lead to performance gains on average compared to its fixed-rate counterpart. However, a question that arises is how we allocate bits optimally. In the next section, we discuss existing literature on optimal bit allocation in transform coding and TSVQ and extend those methods to the VR-ACOVQ bit allocation problem.

4.3 Optimal Bit Allocation Problem

The VR-ACOVQ bit allocation problem is as follows. Consider a VR-ACOVQ at the $i \geq 2$ stage with a set of m_{i-1} channel output index sequences $\mathcal{H}^{i-1} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{m_{i-1}}\}$, where $\mathbf{h}_j \in \mathbb{N}^{i-1}$ for $j = 1, \ldots, m_{i-1}$, from stage $i-1$, with corresponding posterior probability density functions $f_{\mathbf{U}|\mathbf{Y^{i-1}}}(\mathbf{u}|\mathbf{h}_j)$ for $j = 1, \ldots, m_{i-1}$. Here m_{i-1} is an arbitrary number denoting the size of $\mathcal{H}^{i-1}$. Specific details on how the set $\mathcal{H}^{i-1}$ is constructed are included in Section 4.4.1. Let $\boldsymbol{\phi}^{(i)} = (\phi_1^{(i)}, \ldots, \phi_{m_{i-1}}^{(i)})$ be the bit allocation of the ith stage of the VR-ACOVQ, such that $\phi_j^{(i)}$ bits are allocated to the quantizer, at stage i, corresponding to the posterior source distribution with density function $f_{\mathbf{U}|\mathbf{Y^{i-1}}}(\mathbf{u}|\mathbf{h}_j)$. Let $D(\phi_j^{(i)}, \mathbf{h}_j)$ be the expected distortion when quantizing a source with density function $f_{\mathbf{U}|\mathbf{Y^{i-1}}}(\mathbf{u}|\mathbf{h}_j)$ with a $\phi_j^{(i)}$ bit COVQ. The VR-ACOVQ bit allocation problem is to find an optimal $\boldsymbol{\phi}^{(i)}$ that minimizes

$$E[D^{(i)}] = \sum_{j=1}^{m_{i-1}} P(Y^i = \mathbf{h}_j) D(\phi_j^{(i)}, \mathbf{h}_j), \tag{4.1}$$

where $E[D^{(i)}]$ is the expected distortion of the ACOVQ at stage i, such that

$$\bar{\phi}^{(i)} = \sum_{j=1}^{m_{i-1}} P(Y^i = \mathbf{h}_j) \phi_j^{(i)} \leq \Phi^{(i)}, \tag{4.2}$$

where $\bar{\phi}^{(i)}$ is the average bits at stage i and $\Phi^{(i)}$ is a given average bit allocation constraint.

A similar bit allocation problem has been explored in [15, p. 226-231] for quantizing a block of Gaussian random variables under a bit allocation constraint. Consider a set of m scalar random variables $U_1, \ldots, U_m$. Let Q_j denote the quantizer optimized to quantize random variable U_j, and with an abuse of notation, let ϕ_j be the bits allocated to Q_j such that Q_j is a 2^{ϕ_j}-level quantizer. Let $D_j(\phi_j)$ denote the expected distortion of Q_j with ϕ_j bits allocated when quantizing U_j. The optimal bit allocation problem is to find optimal $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_m)$ such that the total distortion

$$D = \sum_{j=1}^{m} D_j(\phi_j), \tag{4.3}$$

is minimized given the constraint

$$\sum_{j=1}^{m} \phi_j \leq \Phi, \tag{4.4}$$

where Φ is the given fixed quota of total bits used. Using a high rate approximation for VQs, the optimal bit allocation is given by [15, p. 229]

$$\phi_j = \frac{\Phi}{m} + \frac{1}{2} \log_2 \frac{\sigma_j^2}{\rho^2}, \tag{4.5}$$

where σ_j^2 is the variance of U_j and

$$\rho^2 = (\prod_{j=1}^{m} \sigma_j^2)^{\frac{1}{m}}, \tag{4.6}$$

which is the geometric mean of the random variable variances. However, the derived optimal bit allocation result permits non-integer and even negative bit values, which are impractical in real-world applications. Further, the solution in (4.5) assumes a high-rate, closed form approximation for the distortion-rate function D_j, which may not hold for low-rate COVQ with high channel noise, let alone low-rate VQs. A greedy algorithm, introduced in [33], addresses these issues in the context of optimal bit allocation when quantizing a block of discrete cosine transform (DCT) coefficients. This algorithm was then extended to a system where the DCT coefficients were quantized by a COVQ and transmitted over a noisy channel in [7] [8].

Given the structural similarities between TSVQ and ACOVQ, it is natural to examine existing variable-rate TSVQ algorithms to determine whether they can be adapted or extended to the ACOVQ framework. Several pruning algorithms have been proposed in [20] [9] [6] [27] [16] to find an optimal bit allocation for variable-rate TSVQs. In these algorithms, a large fixed-rate tree is created, then the nodes that produced the lowest decrease in distortion per bits used (i.e., the "least efficient" nodes) from the tree is pruned repeatedly until only the root node remains, creating a sequence of sub-trees. The sub-tree in the sequence, that provides the best performance for a given average rate, is then selected. For best results, a large initial TSVQ would need to be trained to allow for a larger sub-tree sequence, which can be computationally expensive. However, rather than pruning a large TSVQ, additional bits and nodes can be allocated to an existing tree greedily, until a constraint can no longer be held. A greedy growing tree algorithm is discussed in [28] to split nodes; however, the steps in the pruning methods in [27] can be done in reverse to also create a greedy growing tree algorithm. In the next section, we will detail the growing version of the pruning algorithm in [27] and extend it to the ACOVQ bit allocation problem.

4.3.1 The Generalized BFOS Algorithm

The generalized Breiman, Friedman, Olshen, and Stone (BFOS) algorithm, proposed in [27], provides a greedy bit allocation method for TSVQ. Although the algorithm is detailed as a pruning method, the author notes that the algorithm steps can be done in reverse to greedily "grow" the tree structured quantizer. Here we describe the "growing" version of the algorithm detailed in [27]. Assume we have a "root" VQ with m encoding regions $S_1, S_2, \ldots, S_m$. We then have m additional VQs $Q_1, Q_2, \ldots, Q_m$, such that Q_j further refines source values in S_j, for $j = 1, \ldots, m$. Let ϕ_j be the number of bits allocated for Q_j for $j = 1, \ldots, m$. Let $P(\mathbf{u} \in S_j)$ be the probability a source vector is in S_j. Let $D_j(\phi_j)$ represent the distortion of a ϕ_j-bit quantizer on S_j for $j = 1, \ldots, m$. Let $D, \bar{\phi}$ represent the average distortion and rate of the m quantizers, respectively, as shown:

$$D = \sum_{j=1}^{m} P(\mathbf{u} \in S_j) D_j(\phi_j) \tag{4.7}$$

$$\bar{\phi} = \sum_{j=1}^{m} P(\mathbf{u} \in S_j) \phi_j. \tag{4.8}$$

The objective of this algorithm is to find $\phi_1, \ldots, \phi_m$ that minimizes D and satisfies $\bar{\phi} \leq \Phi$, where Φ is a given average bit allocation constraint. The generalized BFOS algorithm is detailed in the following steps:

1. For $j = 1, \ldots, m$, set $\phi_j = 0$ for the initial bit allocation.

2. Determine $I = \arg\max_{j \in \{1,\ldots,m\}} \{D_j(\phi_i) - D_j(\phi_j + 1)\}$.

3. Calculate D and $\bar{\phi}$. Check if $\bar{\phi} \geq \Phi$. If so, stop. Else set $\phi_I = \phi_I + 1$ and repeat Step 2.

In each iteration, the algorithm uses a simple resource allocation strategy: find the quantizer with the highest performance increase per additional bit allocated, then increment the bits for that quantizer until we exceed our constraint. In the next section we extend this algorithm to the ACOVQ bit allocation problem.

4.4 VR-ACOVQ Bit Allocation Algorithm

4.4.1 Algorithm Overview

In this section, we present a high-level overview of the bit allocation algorithm, focusing on its inputs, outputs, and how it is applied recursively to construct each stage of a VR-ACOVQ. The algorithm is treated as a "black box"; the specific steps will be detailed in the next section. Consider a VR-ACOVQ with a maximum of n stages and a sequence of non-zero constraints $\Phi^{(1)}, \Phi^{(2)}, \ldots, \Phi^{(n)}$. Let $\mathcal{H}^{i-1}$ be the set of channel output sequences up to the ith stage of the VR-ACOVQ for $i = 1, \ldots, n$, and let $\mathcal{G}_{\mathbf{h}}^{(i)} = \{0, 1, \ldots, 2^{\phi_{\mathbf{h}}^{(i)}} - 1\}$ for $\mathbf{h} \in \mathcal{H}^{i-1}$, where, with abuse of notation, $\phi_{\mathbf{h}}^{(i)}$ is the number of bits allocated given $Y^{i-1} = \mathbf{h}$.

First Stage Overview

The first stage of the VR-ACOVQ is a $\lfloor \Phi^{(1)} \rfloor$-bit COVQ with channel indices $\mathcal{I}^{(1)} = \{0, 1, \ldots, 2^{\lfloor \Phi^{(1)} \rfloor} - 1\}$. The algorithm, given $\Phi^{(2)}$, $\mathcal{H}^1$, and associated probabilities for each sequence in $\mathcal{H}^1$ (i.e., $P(Y_1 = h)$ for $h \in \mathcal{H}^1$), will output a bit allocation $\phi_h^{(2)}$ for each $h \in \mathcal{I}^{(1)}$.

Second and Subsequent Stages Overview

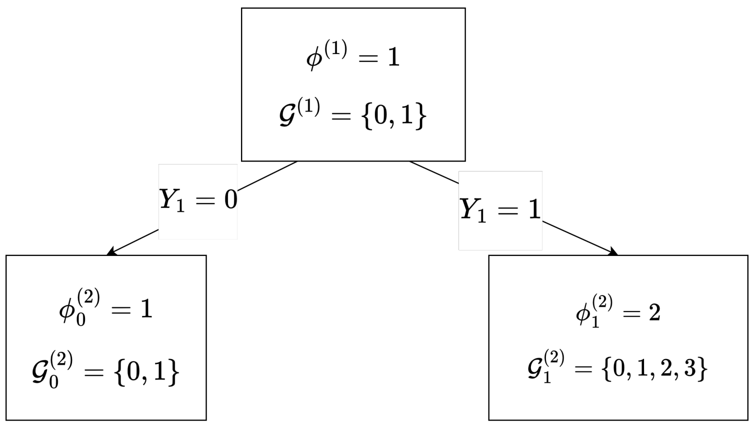

Figure 4.1: Tree diagram for a two-stage VR-ACOVQ.

After finding $\boldsymbol{\phi}^{(2)}$, we construct $\mathcal{H}^2$ using $\boldsymbol{\phi}^{(2)}$ and $\mathcal{H}^1$. The set $\mathcal{H}^1$ provides the channel output received by the encoder via noiseless feedback and $\boldsymbol{\phi}^{(2)}$ provides the bits allocated to each COVQ for each corresponding feedback value. COVQs in the second stage are then added to the VR-ACOVQ with bit allocations provided by $\boldsymbol{\phi}^{(2)}$. We can then construct $\mathcal{H}^2$ as follows:

$$\mathcal{H}^2 = \bigcup_{h_1 \in \mathcal{H}^1} \{h_1\} \times \mathcal{G}_{h_1}^{(2)} \tag{4.9}$$

$$= \bigcup_{h_1 \in \mathcal{H}^1} \{h_1\} \times \left\{0, 1, \ldots, 2^{\phi^{(2)}_{h_1}} - 1\right\}. \tag{4.10}$$

A tree-structured visual depiction of (4.10) can be seen in Figure 4.1. Each rectangle represents a COVQ in the VR-ACOVQ. The rectangle on top is the stage 1 COVQ and the rectangles below are the stage 2 COVQs. Source samples whose stage 1 channel outputs match the labels on the arrows will be further quantized by the corresponding COVQs. Visually, the construction of $\mathcal{H}^2$ is taking all "paths" of the tree and concatenating the "paths" with the corresponding channel indices for the stage 2 COVQs. In this example, the possible channel outputs on the left side of the tree up to stage 2 (i.e., the outputs corresponding with $Y_1 = 0$) is $\{0\} \times \{\mathcal{G}^{(2)}_0\}$ and the possible channel outputs on the right side of the tree at stage 2 is $\{1\} \times \{\mathcal{G}^{(2)}_1\}$. We then have that $\mathcal{H}^2 = \{0\} \times \{\mathcal{G}^{(2)}_0\} \cup \{1\} \times \{\mathcal{G}^{(2)}_1\}$.

The algorithm will then be applied recursively for subsequent stages. Consider stage $i \geq 2$. The algorithm, given $\mathcal{H}^{i-1}$, associated probabilities for each sequence in $\mathcal{H}^{i-1}$ (i.e., $P(Y^{i-1} = \mathbf{h})$ for $\mathbf{h} \in \mathcal{H}^{i-1}$), and $\Phi^{(i)}$ will output $\boldsymbol{\phi}^{(i)}$. We then construct $\mathcal{H}^i$ as follows:

$$\mathcal{H}^i = \bigcup_{\mathbf{h} \in \mathcal{H}^{i-1}} \{h_1\} \times \{h_2\} \times \cdots \times \{h_{i-1}\} \times \mathcal{G}^{(i)}_{\mathbf{h}}. \tag{4.11}$$

Specifically, $\{h_1\} \times \{h_2\} \times \cdots \times \{h_{i-1}\}$ in (4.11) denotes a sequence from stages 1 up to $i - 1$. We then append the new channel indices at stage i, given by the bit allocation $\phi^{(i)}_{\mathbf{h}}$.

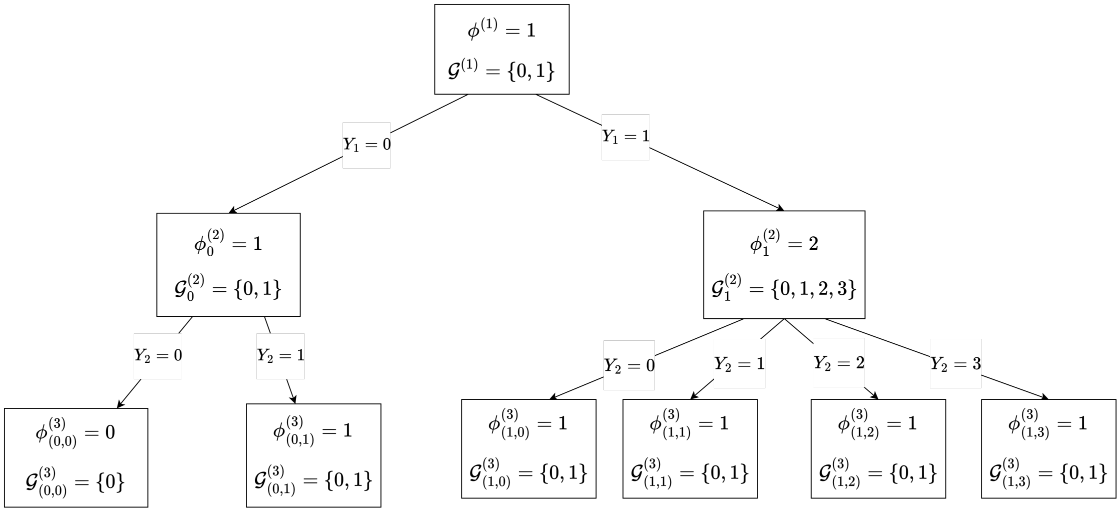

Figure 4.2: Tree diagram for a three-stage VR-ACOVQ.

We can see in Figure 4.2 a three-stage VR-ACOVQ tree diagram, which is a continuation of the 2-stage VR-ACOVQ depicted in Figure 4.1. In this case, we have that $\mathcal{H}^2 = \{(0,0), (0,1), (1,0), (1,1), (1,2), (1,3)\}$, which represents the set of all channel indices up to stage 2. Due to the size of $\mathcal{H}^3$, we will only exhaustively list elements in the set corresponding to $h_1 = 0$. The set of channel indices up to stage 3 corresponding to $h_1 = 0$ (i.e., the "left side" of Figure 4.2) is $\{(0,0,0), (0,1,0), (0,1,1)\}$.

4.4.2 Steepest Descent Bit Allocation Algorithm

In this section, we detail the bit allocation algorithm that will be applied in between stages of VR-ACOVQ. At a given stage $i \geq 2$ in a VR-ACOVQ, assume we have a set of m_{i-1} channel output sequences, $\mathcal{H}^{i-1} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{m_{i-1}}\}$, received by the encoder via noiseless feedback link at stage i. Let $\boldsymbol{\phi}^{(i)} = (\phi_1^{(i)}, \ldots, \phi_{m_{i-1}}^{(i)})$ be the bit allocation vector such that $\phi_j^{(i)}$ corresponds to the bits allocated to the COVQ quantizing the posterior distribution corresponding to channel output $\mathbf{h}_j$ for $j = 1, \ldots, m_{i-1}$. Let the maximum bits allocated to any single quantizer be Φ_{max}, such that $\phi_j^{(i)} \leq \Phi_{max}$, for $j = 1, \ldots, m_{i-1}$. Let $\Phi^{(i)}$ be the maximum average rate such that $\overline{\phi}^{(i)} := \sum_{j=1}^{m_{i-1}} P(Y^{i-1} = \mathbf{h}_j)\phi_j^{(i)} \leq \Phi^{(i)}$. For all $j = 1, \ldots, m_{i-1}$, let $D(\phi_j, \mathbf{h}_j)$ be the expected distortion of a ϕ_j bit quantizer with source distribution $f_{\mathbf{U}|\mathbf{h}_j} := f_{\mathbf{U}|Y^{i-1}}(\mathbf{u}|\mathbf{h}_j)$, and let

$$D^{(i)} = \sum_{j=1}^{m_{i-1}} P(Y^{i-1} = \mathbf{h}_j) D(\phi_j^{(i)}, \mathbf{h}_j) \tag{4.12}$$

represent the average distortion at stage i for a given $\boldsymbol{\phi}^{(i)}$. Further, if $\phi_j^{(i)} = 0$, let $D(\phi_j^{(i)}, \mathbf{h}_j) = \sigma^2_{f_{\mathbf{U}|\mathbf{h}_j}}$, where $\sigma^2_{f_{\mathbf{U}|\mathbf{h}_j}} = \sum_{j=1}^{k} \mathrm{Var}(f_{u_j|\mathbf{h}_j})$ is the sum of the marginal variances of the conditional source distribution.

1. Set $\phi_j^{(i)} = 0$ for all $j = 1, \ldots, m_{i-1}$. This will be the initial state of the bit allocation algorithm. Set $\mathcal{J} = \{1, \ldots, m_{i-1}\}$. The set $\mathcal{J}$ will represent the indices of all quantizers whose bit allocation can be incremented without violating any constraints. Elements of $\mathcal{J}$ will be removed if a bit increase for the corresponding index violate the constraint.

2. Set

$$\lambda_j = \frac{\Delta D^{(i)}}{\Delta \overline{\phi}^{(i)}} \tag{4.13}$$

$$= \frac{P(Y^{i-1} = \mathbf{h}_j)\left(D(\phi_j^{(i)}, \mathbf{h}_j) - D(\phi_j^{(i)} + 1, \mathbf{h}_j)\right)}{P(Y^{i-1} = \mathbf{h}_j)\left((\phi_j^{(i)} + 1) - (\phi_j^{(i)})\right)} \tag{4.14}$$

$$= D(\phi_j^{(i)}, \mathbf{h}_j) - D(\phi_j^{(i)} + 1, \mathbf{h}_j), \quad \forall j \in \mathcal{J}. \tag{4.15}$$

Each element λ_j represents the ratio of the decrease in average distortion per increase in average rate by allocating an extra bit to the quantizer corresponding to sequence $\mathbf{h}_j$ for all $j = 1, \ldots, m_{i-1}$.

3. Find $j_{max} = \arg\max_{j \in \mathcal{J}} \lambda_j$. Determine if

$$\sum_{l \in \{1,\ldots,m_{i-1}\} \setminus \{j_{max}\}} P(Y^{i-1} = \mathbf{h}_l)\phi_l^{(i)} + P(Y^{i-1} = \mathbf{h}_{j_{max}}) \left(\phi_{j_{max}}^{(i)} + 1\right) > \Phi^{(i)}$$

   or if $\phi_{j_{max}}^{(i)} = \Phi_{max}$. The inequality determines whether this increase in allocation would violate the average bit allocation constraint for the given stage. If either statement is true, set $\mathcal{J} = \mathcal{J} \setminus j_{max}$ and set $\lambda_{j_{max}} = 0$. Else set $\phi_{j_{max}}^{(i)} = \phi_{j_{max}}^{(i)} + 1$.

4. If $\mathcal{J} = \emptyset$ or $\lambda_j \leq 0$ for all $j = 1, \ldots, m_{i-1}$ stop and return $\boldsymbol{\phi}^{(i)}$. Else repeat steps 2 and 3.

In the absence of an analytical distortion-rate function for COVQ, we evaluate $D(\phi_j^{(i)}, \mathbf{h}_j)$ by training and evaluating the expected distortion of a COVQ for each value of $\phi_j^{(i)}$ and distribution $f_{\mathbf{U}|\mathbf{h}_j}$ for $j = 1, \ldots, m_{i-1}$. The training for these quantizers and distortion calculations will be done offline, and only the final quantizers (i.e., quantizers whose allocated bits correspond to the final values in $\boldsymbol{\phi}$) will be stored and used in the VR-ACOVQ.

4.5 Complexity Analysis

4.5.1 Computational and Storage Complexity of FR-ACOVQ Encoder

Methods for reducing the complexity of CM-TSVQ and COVQ are detailed in [21] and [13], respectively. In this section, we extend this method to ACOVQ. Consider an ACOVQ at stage $i \geq 2$ that satisfies the generalized nearest neighbor and centroid conditions with partitions $\mathcal{S}^{(1)}, \mathcal{S}^{(2)}_{y_1}, \ldots, \mathcal{S}^{(i)}_{y^{i-1}}$. Let $\mathbf{u} \in \mathbb{R}^k$ be the source vector, $Y^{i-1} = y^{i-1}$ be the channel output sequence received by the encoder via noiseless feedback link, and $X^{i-1} = x^{i-1} = \gamma^{i-1}_{y^{i-2}}(\mathbf{u})$ be the encoded sequence for the previous $i-1$ stages. Recall from (2.99) that an ith stage ACOVQ satisfying the generalized nearest neighbor condition has encoding regions given by

$$S^{(i)}_{x_i|y^{i-1}} = \{\mathbf{u} \in \mathbb{R}^k : d'_i(\mathbf{u}; y^{i-1}, \gamma^{i-1}_{y^{i-2}}(\mathbf{u}), x_i) \leq d'_i(\mathbf{u}; y^{i-1}, \gamma^{i-1}_{y^{i-2}}(\mathbf{u}), x'_i), \quad x'_i \in \mathcal{I}^{(i)}\} \quad x_i \in \mathcal{I}^{(i)}, y^{i-1} \in \mathcal{I}^{i-1}. \tag{4.16}$$

We then have that

$$\mathcal{E}^{(i)}(\gamma^{i-1}_{y^{i-2}}(\mathbf{u}), y^{i-1}, \mathbf{u}) = x_i \iff x_i \in S^{(i)}_{x_i|y^{i-1}} \tag{4.17}$$

$$\implies \mathcal{E}^{(i)}(\gamma^{i-1}_{y^{i-2}}(\mathbf{u}), y^{i-1}, \mathbf{u}) = \arg\min_{x_i \in \mathcal{I}^{(i)}} \{d'(\mathbf{u}; \gamma^{i-1}_{y^{i-2}}(\mathbf{u}), x_i, y^{i-1})\}, \tag{4.18}$$

where

$$
\begin{aligned}
d'(\mathbf{u};& x^{i-1}, x_i, y^{i-1}) \\
&= \sum_{y_i \in \mathcal{I}^{(i)}} P(Y_i = y_i | X^i = x^i, Y^{i-1} = y^{i-1}) \left\| \mathbf{u} - \mathbf{c}^{(i,AC)}_{y_i|y^{i-1}} \right\|^2 \qquad (4.19) \\
&= \sum_{y_i \in \mathcal{I}^{(i)}} P(Y_i = y_i | X^i = x^i, Y^{i-1} = y^{i-1}) \\
&\qquad \times \left(\|\mathbf{u}\|^2 - 2 < \mathbf{u}, \mathbf{c}^{(i,AC)}_{y_i|y^{i-1}} > + \left\| \mathbf{c}^{(i,AC)}_{y_i|y^{i-1}} \right\|^2 \right). \qquad (4.20)
\end{aligned}
$$

Let

$$
\kappa_1^{(i)}(y^{i-1}, x^i) = \sum_{y_i \in \mathcal{I}^{(i)}} P(Y_i = y_i | X^i = x^i, Y^{i-1} = y^{i-1}) \mathbf{c}^{(i,AC)}_{y_i|y^{i-1}} \qquad (4.21)
$$

$$
\kappa_2^{(i)}(y^{i-1}, x^i) = \sum_{y_i \in \mathcal{I}^{(i)}} P(Y_i = y_i | X^i = x^i, Y^{i-1} = y^{i-1}) \left\| \mathbf{c}^{(i,AC)}_{y_i|y^{i-1}} \right\|^2. \qquad (4.22)
$$

The encoding function can then be reduced as follows:

$$
\begin{aligned}
&\mathcal{E}^{(i)}(\gamma^{i-1}_{y^{i-2}}(\mathbf{u}), y^{i-1}, \mathbf{u}) \qquad (4.23) \\
&\quad = \arg \min_{x_i \in \mathcal{I}^{(i)}} \{ d'(\mathbf{u}; \gamma^{i-1}_{y^{i-2}}(\mathbf{u}), x_i, y^{i-1}) \} \qquad (4.24) \\
&\quad = \arg \min_{x_i \in \mathcal{I}^{(i)}} \Bigg\{ \sum_{y_i \in \mathcal{I}^{(i)}} P(Y_i = y_i | X^i = x^i, Y^{i-1} = y^{i-1}) \\
&\qquad\qquad \times \left(\|\mathbf{u}\|^2 - 2 < \mathbf{u}, \mathbf{c}^{(i,AC)}_{y_i|y^{i-1}} > + \left\| \mathbf{c}^{(i,AC)}_{y_i|y^{i-1}} \right\|^2 \right) \Bigg\} \qquad (4.25)
\end{aligned}
$$

$$= \arg\min_{x_i \in \mathcal{I}^{(i)}} \left\{ \left(\sum_{y_i \in \mathcal{I}^{(i)}} P(Y_i = y_i | X^i = x^i, Y^{i-1} = y^{i-1}) \right) \|\mathbf{u}\|^2 \right.$$

$$\left. - 2 < \mathbf{u}, \kappa_1^{(i)}(y^{i-1}, x^i) > + \kappa_2^{(i)}(y^{i-1}, x^i) \right\} \quad (4.26)$$

$$= \arg\min_{x_i \in \mathcal{I}^{(i)}} \{ \mathbf{u} - 2 < \mathbf{u}, \kappa_1^{(i)}(y^i, x^i) > + \kappa_2^{(i)}(y^i, x^i) \} \quad (4.27)$$

$$= \arg\min_{x_i \in \mathcal{I}^{(i)}} \{ -2 < \mathbf{u}, \kappa_1^{(i)}(y^i, x^i) > + \kappa_2^{(i)}(y^i, x^i) \}. \quad (4.28)$$

Assuming $b_i > 0$, the encoding computational complexity of (4.28) is $k \cdot 2^{b_i}$ floating point operations (FLOPs), or 2^{b_i} FLOPs per sample, which comes from the inner product. If $b_i = 0$, then $|\mathcal{I}^{(i)}| = 1$ and no computations are needed to encode any values. Note the encoder does not need to consider indices that correspond to empty encoding regions when finding the encoding sequence corresponding to the minimum value in (4.28). If a quantizer has empty encoding regions, we can replace $x_i \in \mathcal{I}^{(i)}$, in (4.28), with $x_i \in \mathcal{I}^{(i)}_{nonempty}$, where $\mathcal{I}^{(i)}_{nonempty} \subset \mathcal{I}^{(i)}$ is the set of indices corresponding to nonempty encoding regions. As a result, the complexity can be further reduced to $k \cdot |\mathcal{I}^{(i)}_{nonempty}|$ FLOPs or $|\mathcal{I}^{(i)}_{nonempty}|$ FLOPs per sample.

To reduce the computational complexity, the encoder stores all outputs from $\kappa_1^{(i)}$ and $\kappa_2^{(i)}$. The total number of inputs for $\kappa_1^{(i)}$ and $\kappa_2^{(i)}$ is $|\mathcal{I}^{i-1}| \cdot |\mathcal{I}^i|$ for each function. Consequently, we have that the number of scalars stored for all outputs of $\kappa_1^{(i)}$ is $k \cdot (|\mathcal{I}^{i-1}| \cdot |\mathcal{I}^i|)$, and for $\kappa_2^{(i)}$ is $|\mathcal{I}^{i-1}| \cdot |\mathcal{I}^i|$. Note that when the channel has finite memory, (4.21) and (4.22) may depend only on a subsequence of X^i, allowing for further reductions in storage complexity. For example, consider a $\mathbf{b} = (2, 1)$ ACOVQ, with codebooks $\mathcal{C}^{(1)}, \mathcal{C}^{(2)}_{y_1}$ for $y_1 \in \mathcal{I}^{(1)}$, and a Polya contagion channel with memory $M = 1$, detailed in Section 3.4.1, satisfying the following transition probability property

$$P(Z_i = z_i | Z^{i-1} = z^{i-1}) = P(Z_i = z_i | Z_{i-1} = z_{i-1}). \tag{4.29}$$

For stage $i = 2$, we have

$$\kappa_1^{(2)}(y_1, x^2) = \sum_{y_2 \in \mathcal{I}^{(2)}} P(Y_2 = y_2 | X^2 = x^2, Y_1 = y_1) \mathbf{c}_{y_2|y_1}^{(2,AC)} \tag{4.30}$$

$$= \sum_{y_2 \in \mathcal{I}^{(2)}} P(Z_3 = \beta_1(x_2) \oplus \beta_1(y_2) | Z^2 = \beta_2(x_1) \oplus \beta_2(y_1)) \mathbf{c}_{y_2|y_1}^{(2,AC)} \tag{4.31}$$

$$= \sum_{y_2 \in \mathcal{I}^{(2)}} P(Z_3 = \beta_1(x_2) \oplus \beta_1(y_2) | Z_2 = (\beta_2(x_1) \oplus \beta_2(y_1))_2) \mathbf{c}_{y_2|y_1}^{(2,AC)}, \tag{4.32}$$

where $\beta_\eta(x)$, is the η-bit binary representation of the integer x (see Section 3.4.1) and $(\beta_2(x_1) \oplus \beta_2(y_1))_2$ refers to the 2nd component of the binary vector. We can see that $\kappa_1^{(2)}$ does not vary with the term $(\beta_2(x_1))_1$. Hence, we have that $\kappa_1^{(2)}$ only depends on the 2 of the 3 binary vector values of (x_1, x_2). Thus instead of accounting for all 8 values for $(x_1, x_2) \in \mathcal{I}^2$, we only need to account for the first 2 bits. Therefore the storage complexity can be reduced from $|\mathcal{I}^2| \cdot |\mathcal{I}^1| = 32$ scalar values to $4 \cdot |\mathcal{I}^1| = 16$ scalar values. A similar argument can be used to show that the storage complexity of $\kappa_2^{(2)}$ can be reduced from $k \cdot 32$ scalar values to $k \cdot 16$ scalar values in this example.

4.5.2 Encoding Complexity of VR-ACOVQ

In this section, we compare the average encoding computational complexity and average rate of the VR-ACOVQ and FR-ACOVQ. Assume that we have an n stage, k-dimensional, FR-ACOVQ with bit allocation $\mathbf{b}$ and an n stage, k-dimensional, VR-ACOVQ with bit allocation $\boldsymbol{\phi}^{(i)}$ such that $\bar{\phi}^{(i)} = b_i$ for $i = 1, \ldots, n$. Further, we assume that $\phi_j^{(i)} \geq 1$ for $j = 1, \ldots, m_{i-1}$ and that there are no empty encoding regions for all quantizers at this stage. The average rate of the VR-ACOVQ for a given stage $2 \leq i \leq n$ is given by

$$\frac{1}{k}\sum_{j=1}^{m_{i-1}} P(Y^{i-1} = \mathbf{h}_j)\phi_j^{(i)} = \frac{1}{k}\bar{\phi}^{(i)} \tag{4.33}$$

$$= \frac{1}{k}b_i \tag{4.34}$$

bits per source sample, which is equivalent to the average rate of its FR-ACOVQ counterpart. From (4.28), the encoding complexity of a ϕ-bit quantizer is

$$g(\phi) = \begin{cases} 2^{\phi} & \phi \geq 1 \\ 0 & \phi = 0 \end{cases} \tag{4.35}$$

FLOPs per source sample. However, if we restrict the domain such that $\phi \geq 1$, the piecewise function reduces to $g(\phi) = 2^{\phi}$, which is a convex function. The encoding complexity of a VR-ACOVQ for a given stage $2 \leq i \leq n$ is then

$$\sum_{j=1}^{m_{i-1}} P(Y^{i} = \mathbf{h}_j)2^{\phi_j^{(i)}} \geq 2^{\sum_{j=1}^{m_{i-1}} P(Y^{i-1}=\mathbf{h}_j)\phi_j^{(i)}} \tag{4.36}$$

$$= 2^{\bar{\phi}^{(i)}} \tag{4.37}$$

$$= 2^{b_i} \tag{4.38}$$

FLOPs per source sample. Note that the inequality from (4.36) comes from Jensen's inequality and from the convexity of the function $g(\phi) = 2^{\phi}$. We can see from (4.38) that the encoding complexity of the the VR-ACOVQ will be lower bounded by the complexity of the FR-ACOVQ. Thus, in addition to the multiple quantizers that need to be trained for the VR-ACOVQ, the encoding complexity on average will be higher than its FR-ACOVQ counterpart.

However, the encoding complexity can be upper bounded based on the selected value of Φ_{max}, the maximum bits that can be allocated to any quantizer in a VR-ACOVQ. Consider a VR-ACOVQ at stage i with an average bit allocation constraint $\Phi^{(i)}$, maximum bit allocation for any quantizer of Φ_{max}, such that $\Phi_{max} \geq \Phi^{(i)}$, and the set of channel output sequences $\mathcal{H}^{i-1} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{m_{i-1}}\}$. Let $p_j = P(Y^{i-1} = \mathbf{h}_j)$ for $j = 1, \ldots, m_{i-1}$. We want to find $p_1, \ldots, p_{m_{i-1}}$ and $\phi_1^{(i)}, \ldots, \phi_{m_{i-1}}^{(i)}$ that satisfy

$$\sum_{l=1}^{m_{i-1}} p_l \phi_l^{(i)} \leq \Phi^{(i)} \tag{4.39}$$

$$\sum_{l=1}^{m_{i-1}} p_l = 1 \tag{4.40}$$

$$0 \leq \phi_j^{(i)} \leq \Phi_{max} \tag{4.41}$$

$$p_j \geq 0, \tag{4.42}$$

for $j = 1, \ldots, m_{i-1}$, and maximizes

$$V(p_1, \ldots, p_{m_{i-1}}, \boldsymbol{\phi}^{(i)}) := \sum_{l=1}^{m_{i-1}} p_l g(\phi_l^{(i)}), \tag{4.43}$$

the average encoding complexity of stage i. Because $g(\cdot)$ is a strictly increasing function, the maximum of $V(\cdot)$ will be found when $\phi_l^{(i)} = 0$ or Φ_{max} for $l = 1, \ldots, m_{i-1}$ (i.e., only values on the boundaries of the intervals in (4.41) are used). Equivalently, we can let $\phi_1^{(i)} = 0$ and $\phi_2^{(i)} = \Phi_{max}$ and let $p_3 = p_4 = \cdots p_{m_{i-1}} = 0$. Furthermore, the maximum complexity is achieved when the maximum average rate is achieved in (4.40) (i.e., when $\sum_{l=1}^{m_{i-1}} p_l \phi_l^{(i)} = \Phi^{(i)}$). We then have that the maximum complexity is achieved when

$$p_1 + p_2 = 1 \tag{4.44}$$

$$p_1 * 0 + p_2 \Phi_{max} = \Phi^{(i)}, \tag{4.45}$$

which implies that $p_2 = \frac{\Phi^{(i)}}{\Phi_{max}}, p_1 = 1 - \frac{\Phi^{(i)}}{\Phi_{max}}$. Plugging these values into (4.43) we then have that the upper bound for complexity is

$$V(p_1, \ldots, p_{m_{i-1}}, \boldsymbol{\phi}^{(i)}) = p_1 g(0) + p_2 g(\Phi_{max}) \tag{4.46}$$

$$= \frac{\Phi^{(i)}}{\Phi_{max}} 2^{\Phi_{max}} \tag{4.47}$$

FLOPs per source sample. Hence, the encoding complexity of the VR-ACOVQ is dominated by Φ_{max}.

4.6 Simulation Results

The following tables compare the performances of a VR-ACOVQ to a FR-ACOVQ. Note that the first stage for both quantizers is a COVQ. Let n denote the total number of stages for both quantizers and let $\mathbf{\Phi} = (\Phi^{(1)}, \Phi^{(2)}, \ldots, \Phi^{(n)})$ denote the average bit allocation constraints for the n stages. The VR-ACOVQ will be trained by recursively adding stages with a bit allocation given by the steepest decent algorithm in Section 4.4.2. The FR-ACOVQ will have bit allocation $\mathbf{b} = (\Phi^{(1)}, \Phi^{(2)}, \ldots, \Phi^{(n)})$. Note that although $\Phi^{(j)}$ for $j = 1, \ldots, n$ do not need to be integers when used as a constraint for the VR-ACOVQ, only integer values will be considered for the simulations to allow $\mathbf{b} = \mathbf{\Phi}$ to be a valid FR-ACOVQ bit allocation, allowing the two quantizers to have similar, if not equal, average rates. For the cases where "Balanced Tree" is true, the FR-ACOVQ and VR-ACOVQ are expected to have similar performances. The quantizers were trained on a source of 4 million vectors over a sequence of increasing and decreasing ϵ values for various δ and average bit allocations constraints with the best performing quantizers stored. Various 4-bit and 6-bit VR-ACOVQ and FR-ACOVQ are trained and compared with each other over memoryless Laplacian and Gaussian sources. For the 4-bit quantizers we have $\Phi_{max} = 5$, and for the 6-bit quantizers we have $\Phi_{max} = 8$.

Tables 4.1 - 4.6 and Tables 4.13 - 4.16 display the performance of FR-ACOVQ and VR-ACOVQ for various 4-bit and 6-bit bit allocations respectively. Each have a column called "Balanced Tree", which contains booleans: when true, the optimal bit allocation from the steepest descent algorithm is equivalent to a fixed-rate ACOVQ. Further for the columns "VR-ACOVQ SNR" and "FR-ACOVQ SNR," the greater value in each row is bolded. Tables 4.13 - 4.13 and Tables 4.21 - 4.30 display the performance differences between FR-ACOVQ and VR-ACOVQ. Note that in these tables, unbolded values correspond to when the VR-ACOVQ and FR-ACOVQ bit allocations are equal (i.e., when the corresponding “Balanced Tree” value is true), while bolded values indicate a difference between the bit allocations. In this thesis, only specific bit allocations will be selected for brevity. Additional simulation results can be seen in Appendix B.

4.6.1 4-Bit VR-ACOVQ Simulation Results

Tables 4.1 - 4.6 show the performances for the VR-ACOVQ and FR-ACOVQ for various 4-bit bit allocations. Note that the Bit Allocation Average column depicts a vector of bit allocation averages for each stage of the VR-ACOVQ (i.e., $(\bar{\phi}^{(1)}, \bar{\phi}^{(2)}, \ldots, \bar{\phi}^{(n)})$). We can see that in general, the VR-ACOVQ outperforms the FR-ACOVQ regardless of the ϵ and δ communication channel parameters despite having a lower average rate. Further, from Tables 4.7 - 4.12 we can see that performance gap between the VR-ACOVQ and FR-ACOVQ is greater for a memoryless Laplacian source compared to a memoryless Gaussian source. This indicates that the gap between a VR-ACOVQ and FR-ACOVQ widens with a more biased source distribution. An explanation for this behavior is that a bit allocation determined by the steepest descent bit allocation algorithm outperforms a fixed-rate bit when the posterior distributions at the given stage vary greatly. With a biased source distribution, such as the Laplacian distribution, the posterior distributions exhibit greater variation, leading to poorer performance of fixed-rate allocation and improved performance of the algorithm-driven allocation. Table 4.29 shows that for 2-dimensional memoryless Gaussian sources, the bit allocation significantly affects the performance difference between FR-ACOVQ and VR-ACOVQ. Overall, the bit allocations $(3, 1)$ and $(1, 2, 1)$ resulted in the most consistent performance improvements. In contrast, bit allocations $(1, 1, 2), (1, 3),$ and $(2, 1, 1)$ consistently resulted in VR-ACOVQs with bit allocations equivalent to FR-ACOVQ, leading to negligible performance differences. This suggests that, at low rates, VR-ACOVQ offers little to no benefit when the average bit allocation constraint for the first and intermediate stages are 1 bit with some exceptions.

4.6.2 6-Bit VR-ACOVQ Simulation Results

Tables 4.13 - 4.20 present the performances of VR-ACOVQ and FR-ACOVQ for various 6-bit bit allocations on 1 and 2 dimensional memoryless Gaussian and Laplacian sources. We can see that in Tables 4.21 and 4.23 the SNR gain of VR-ACOVQ over FR-ACOVQ ranges from 0.6 dB - 2.1 dB for bit allocations (4,1,1) and (3,1,1,1) on a 1-dimensional Gaussian source. On a 1-dimensional Laplacian source, we can see in Tables 4.25 and 4.27 the SNR gain of VR-ACOVQ over FR-ACOVQ ranges from 1.2 dB - 4 dB for bit allocations (4,1,1) and (3,1,1,1). In the 4-bit bit allocations, the SNR gain between the two schemes goes up to 1.7 dB and 3 dB on Gaussian and Laplacian sources, respectively. This indicates that the SNR gain between VR-ACOVQ and FR-ACOVQ is increased with a higher rate and a more concentrated source distribution. Furthermore, in Table 4.30, negligible gain is observed with the bit allocation $(1, 1, 4)$, while consistent gain is observed with allocations such as $(1, 4, 1)$ and $(4, 1, 1)$. This suggests that the bit allocation algorithm generally defaults to a fixed-rate allocation when all stages—except the last—are constrained to 1 bit for a 2-dimensional Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**20.674**	20.116	False	(1.0, 1.0, 1.0, 1.0)
0.0005	0	**20.004**	19.664	False	(1.0, 1.0, 1.0, 1.0)
0.0010	0	**19.562**	19.274	False	(1.0, 1.0, 1.0, 0.961)
0.0050	0	**17.158**	16.955	False	(1.0, 1.0, 1.0, 0.97)
0.0100	0	**15.377**	15.199	False	(1.0, 1.0, 1.0, 0.956)
0.0500	0	**10.624**	10.404	False	(1.0, 1.0, 1.0, 0.995)
0.1000	0	**7.523**	7.397	False	(1.0, 1.0, 1.0, 0.99)
0.0000	5	**20.674**	20.099	False	(1.0, 1.0, 1.0, 1.0)
0.0005	5	**20.011**	19.684	False	(1.0, 1.0, 1.0, 0.955)
0.0010	5	**19.573**	19.253	False	(1.0, 1.0, 1.0, 0.959)
0.0050	5	**17.469**	17.013	False	(1.0, 1.0, 1.0, 0.986)
0.0100	5	**15.711**	15.333	False	(1.0, 1.0, 1.0, 0.955)
0.0500	5	11.644	**11.837**	False	(1.0, 1.0, 1.0, 1.0)
0.1000	5	**10.407**	9.995	False	(1.0, 1.0, 1.0, 0.999)
0.0000	10	**20.667**	20.107	False	(1.0, 1.0, 1.0, 1.0)
0.0005	10	**20.087**	19.741	False	(1.0, 1.0, 1.0, 0.956)
0.0010	10	**19.749**	19.429	False	(1.0, 1.0, 1.0, 0.957)
0.0050	10	**17.669**	17.442	False	(1.0, 1.0, 1.0, 0.965)
0.0100	10	**16.433**	16.307	False	(1.0, 1.0, 1.0, 0.983)
0.0500	10	**13.233**	13.000	False	(1.0, 1.0, 1.0, 0.983)
0.1000	10	**11.977**	11.511	False	(1.0, 1.0, 0.999, 0.989)

Table 4.1: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**20.588**	20.160	False	(3.0, 0.993)
0.0005	0	**19.758**	19.042	False	(3.0, 0.985)
0.0010	0	**19.173**	18.318	False	(3.0, 0.986)
0.0050	0	**16.503**	14.809	False	(3.0, 0.995)
0.0100	0	**14.709**	13.308	False	(3.0, 0.995)
0.0500	0	**10.245**	9.375	False	(3.0, 0.993)
0.1000	0	**7.164**	6.579	False	(3.0, 0.998)
0.0000	5	**20.556**	20.154	False	(3.0, 0.99)
0.0005	5	**20.206**	19.726	False	(3.0, 0.998)
0.0010	5	**19.798**	19.393	False	(3.0, 0.994)
0.0050	5	**18.100**	17.284	False	(3.0, 0.981)
0.0100	5	**16.901**	15.605	False	(3.0, 0.988)
0.0500	5	**12.424**	11.047	False	(3.0, 0.952)
0.1000	5	**10.156**	9.077	False	(3.0, 0.963)
0.0000	10	**20.528**	20.199	False	(3.0, 0.987)
0.0005	10	**20.288**	19.897	False	(3.0, 0.989)
0.0010	10	**20.184**	19.671	False	(3.0, 0.997)
0.0050	10	**18.775**	18.126	False	(3.0, 0.981)
0.0100	10	**17.836**	16.833	False	(3.0, 0.993)
0.0500	10	**13.897**	12.474	False	(3.0, 0.948)
0.1000	10	**11.918**	10.539	False	(3.0, 0.988)

Table 4.2: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(3, 1)$ and 1-dimensional $(k = 1)$ memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**20.666**	20.081	False	(2.0, 1.0, 0.999)
0.0005	0	**19.766**	19.372	False	(2.0, 1.0, 1.0)
0.0010	0	**19.119**	18.776	False	(2.0, 1.0, 0.962)
0.0050	0	**17.405**	15.782	False	(2.0, 0.993, 0.999)
0.0100	0	**16.032**	14.423	False	(2.0, 0.837, 0.999)
0.0500	0	**11.073**	9.914	False	(2.0, 0.956, 0.995)
0.1000	0	**7.283**	7.092	False	(2.0, 0.813, 0.995)
0.0000	5	**20.674**	20.096	False	(2.0, 1.0, 1.0)
0.0005	5	**20.150**	19.723	False	(2.0, 1.0, 0.999)
0.0010	5	**19.803**	19.379	False	(2.0, 1.0, 0.964)
0.0050	5	**17.868**	17.359	False	(2.0, 1.0, 0.965)
0.0100	5	**16.716**	15.772	False	(2.0, 1.0, 0.959)
0.0500	5	**13.079**	11.760	False	(2.0, 0.989, 1.0)
0.1000	5	**10.382**	9.932	False	(2.0, 0.88, 0.995)
0.0000	10	**20.527**	20.099	False	(2.0, 1.0, 1.0)
0.0005	10	**20.402**	19.836	False	(2.0, 1.0, 0.999)
0.0010	10	**19.980**	19.645	False	(2.0, 1.0, 0.957)
0.0050	10	**18.425**	18.060	False	(2.0, 1.0, 0.962)
0.0100	10	**17.428**	16.732	False	(2.0, 1.0, 0.949)
0.0500	10	**14.136**	13.303	False	(2.0, 0.967, 0.994)
0.1000	10	**11.750**	11.657	False	(2.0, 0.85, 0.996)

Table 4.3: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(2, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0	0	**16.857**	14.945	False	(1.0, 1.0, 0.986, 1.0)
0.001	0	**15.982**	14.590	False	(1.0, 1.0, 0.94, 0.999)
0.001	0	**15.610**	14.249	False	(1.0, 1.0, 0.937, 0.999)
0.005	0	**13.109**	12.267	False	(1.0, 1.0, 0.904, 0.996)
0.010	0	**11.630**	10.672	False	(1.0, 1.0, 0.966, 0.999)
0.050	0	**6.371**	5.940	False	(1.0, 1.0, 0.923, 0.999)
0.100	0	**3.389**	3.244	False	(1.0, 1.0, 0.925, 1.0)
0	5	**16.931**	14.950	False	(1.0, 1.0, 0.995, 1.0)
0.001	5	**15.991**	14.628	False	(1.0, 1.0, 0.94, 0.999)
0.001	5	**15.215**	14.329	False	(1.0, 1.0, 0.913, 0.998)
0.005	5	**13.518**	12.572	False	(1.0, 1.0, 0.925, 1.0)
0.010	5	**12.467**	11.092	False	(1.0, 1.0, 0.941, 0.997)
0.050	5	**7.997**	7.516	False	(1.0, 1.0, 0.941, 0.999)
0.100	5	**6.873**	5.878	False	(1.0, 1.0, 0.982, 0.987)
0	10	**16.893**	14.935	False	(1.0, 1.0, 0.993, 1.0)
0.001	10	**15.544**	14.699	False	(1.0, 1.0, 0.91, 0.999)
0.001	10	**15.402**	14.470	False	(1.0, 1.0, 0.937, 0.998)
0.005	10	**14.246**	13.019	False	(1.0, 1.0, 0.943, 1.0)
0.010	10	**12.556**	11.718	False	(1.0, 1.0, 0.932, 0.998)
0.050	10	**8.750**	8.506	False	(1.0, 1.0, 0.977, 1.0)
0.100	10	**7.724**	7.221	False	(1.0, 1.0, 0.935, 0.996)

Table 4.4: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Laplacian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.000	0	**16.670**	15.094	False	(3.0, 1.0)
0.001	0	**15.840**	14.082	False	(3.0, 0.987)
0.001	0	**15.324**	13.293	False	(3.0, 0.984)
0.005	0	**13.062**	10.776	False	(3.0, 0.999)
0.010	0	**11.140**	8.235	False	(3.0, 0.987)
0.050	0	**6.537**	4.989	False	(3.0, 0.988)
0.100	0	**3.445**	2.390	False	(3.0, 0.986)
0.000	5	**16.624**	15.086	False	(3.0, 0.999)
0.001	5	**16.268**	14.652	False	(3.0, 0.991)
0.001	5	**15.950**	14.107	False	(3.0, 0.987)
0.005	5	**14.713**	12.324	False	(3.0, 0.971)
0.010	5	**13.683**	10.581	False	(3.0, 0.958)
0.050	5	**9.394**	6.610	False	(3.0, 0.981)
0.100	5	**7.026**	4.861	False	(3.0, 0.983)
0.000	10	**16.602**	15.089	False	(3.0, 0.999)
0.001	10	**16.377**	14.785	False	(3.0, 0.989)
0.001	10	**16.204**	14.542	False	(3.0, 0.999)
0.005	10	**15.062**	12.831	False	(3.0, 0.999)
0.010	10	**14.524**	11.988	False	(3.0, 0.997)
0.050	10	**10.797**	7.943	False	(3.0, 0.992)
0.100	10	**8.349**	6.159	False	(3.0, 0.953)

Table 4.5: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(3, 1)$ and 1-dimensional ($k = 1$) memoryless Laplacian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0	0	**16.901**	14.931	False	(2.0, 0.997, 1.0)
0.001	0	**15.307**	14.396	False	(2.0, 0.923, 1.0)
0.001	0	**15.175**	13.877	False	(2.0, 0.971, 1.0)
0.005	0	**13.780**	11.140	False	(2.0, 0.943, 0.996)
0.010	0	**12.531**	10.022	False	(2.0, 0.921, 0.999)
0.050	0	**7.238**	5.331	False	(2.0, 0.861, 0.999)
0.100	0	**3.901**	2.862	False	(2.0, 0.841, 0.995)
0.000	5	**16.922**	14.937	False	(2.0, 0.991, 1.0)
0.001	5	**15.522**	14.546	False	(2.0, 0.915, 1.0)
0.001	5	**15.363**	14.242	False	(2.0, 0.92, 0.997)
0.005	5	**14.441**	12.333	False	(2.0, 0.995, 1.0)
0.010	5	**13.500**	10.867	False	(2.0, 0.929, 1.0)
0.050	5	**9.681**	7.368	False	(2.0, 0.931, 0.999)
0.100	5	**6.914**	5.575	False	(2.0, 0.912, 0.999)
0.000	10	**16.946**	14.951	False	(2.0, 0.996, 1.0)
0.001	10	**16.625**	14.665	False	(2.0, 0.998, 1.0)
0.001	10	**15.524**	14.426	False	(2.0, 0.93, 0.999)
0.005	10	**14.834**	12.939	False	(2.0, 0.99, 0.999)
0.010	10	**14.086**	11.745	False	(2.0, 0.978, 0.995)
0.050	10	**10.448**	8.649	False	(2.0, 0.905, 1.0)
0.100	10	**7.987**	7.167	False	(2.0, 0.875, 0.998)

Table 4.6: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(2, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Laplacian source.

ϵ	δ		
	0	5	10
0.0	**0.559**	**0.576**	**0.559**
0.0005	**0.34**	**0.327**	**0.346**
0.001	**0.288**	**0.32**	**0.32**
0.005	**0.203**	**0.456**	**0.227**
0.01	**0.178**	**0.378**	**0.126**
0.05	**0.22**	**-0.193**	**0.233**
0.1	**0.125**	**0.411**	**0.466**

Table 4.7: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(1, 1, 1, 1)$ and memoryless 1-dimensional ($k = 1$) Gaussian source.

ϵ	δ		
	0	5	10
0.0	**1.915**	**1.977**	**1.964**
0.0005	**1.39**	**1.364**	**0.845**
0.001	**1.364**	**0.882**	**0.919**
0.005	**0.833**	**0.944**	**1.221**
0.01	**0.97**	**1.375**	**0.838**
0.05	**0.44**	**0.484**	**0.255**
0.1	**0.151**	**1.0**	**0.503**

Table 4.8: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(1, 1, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Laplacian source.

ϵ	δ		
	0	5	10
0.000000	**0.428**	**0.402**	**0.33**
0.000500	**0.717**	**0.48**	**0.392**
0.001000	**0.855**	**0.405**	**0.512**
0.005000	**1.695**	**0.816**	**0.649**
0.010000	**1.401**	**1.296**	**1.003**
0.050000	**0.87**	**1.377**	**1.423**
0.100000	**0.585**	**1.079**	**1.379**

Table 4.9: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(3, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

ϵ	δ		
	0	5	10
0.000000	**1.576**	**1.55**	**1.522**
0.000500	**1.749**	**1.616**	**1.597**
0.001000	**2.031**	**1.848**	**1.662**
0.005000	**2.277**	**2.389**	**2.229**
0.010000	**2.912**	**3.094**	**2.535**
0.050000	**1.556**	**2.773**	**2.848**
0.100000	**1.058**	**2.158**	**2.186**

Table 4.10: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(3, 1)$ and 1-dimensional ($k = 1$) memoryless Laplacian source.

	δ		
ϵ	0	5	10
0.000000	**0.584**	**0.578**	**0.428**
0.000500	**0.393**	**0.427**	**0.566**
0.001000	**0.343**	**0.424**	**0.335**
0.005000	**1.623**	**0.508**	**0.364**
0.010000	**1.608**	**0.943**	**0.696**
0.050000	**1.159**	**1.319**	**0.833**
0.100000	**0.191**	**0.45**	**0.093**

Table 4.11: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(2, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.000000	**1.976**	**1.985**	**1.99**
0.000500	**0.912**	**0.975**	**1.96**
0.001000	**1.297**	**1.116**	**1.098**
0.005000	**2.641**	**2.101**	**1.906**
0.010000	**2.516**	**2.635**	**2.34**
0.050000	**1.91**	**2.314**	**1.8**
0.100000	**1.033**	**1.339**	**0.818**

Table 4.12: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(2, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Laplacian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**32.709**	31.670	False	(4.0, 0.994, 1.0)
0.0005	0	**27.194**	25.679	False	(4.0, 0.999, 1.0)
0.0010	0	**25.607**	24.225	False	(4.0, 0.999, 1.0)
0.0050	0	**22.581**	21.333	False	(4.0, 0.988, 1.0)
0.0100	0	**20.994**	19.451	False	(4.0, 1.0, 1.0)
0.0500	0	**14.221**	13.331	False	(4.0, 0.997, 1.0)
0.1000	0	**10.310**	9.472	False	(4.0, 0.997, 1.0)
0.0000	5	**32.684**	31.651	False	(4.0, 1.0, 1.0)
0.0005	5	**29.257**	27.544	False	(4.0, 1.0, 1.0)
0.0010	5	**28.062**	26.045	False	(4.0, 0.995, 1.0)
0.0050	5	**23.556**	22.292	False	(4.0, 0.995, 1.0)
0.0100	5	**21.578**	20.815	False	(4.0, 0.984, 1.0)
0.0500	5	**17.598**	15.623	False	(4.0, 1.0, 1.0)
0.1000	5	**15.237**	13.404	False	(4.0, 0.998, 1.0)
0.0000	10	**32.693**	31.680	False	(4.0, 0.999, 1.0)
0.0005	10	**30.508**	28.889	False	(4.0, 0.996, 1.0)
0.0010	10	**29.515**	27.395	False	(4.0, 0.998, 1.0)
0.0050	10	**25.444**	23.960	False	(4.0, 0.999, 1.0)
0.0100	10	**23.729**	22.271	False	(4.0, 1.0, 1.0)
0.0500	10	**19.605**	18.197	False	(4.0, 0.999, 1.0)
0.1000	10	**16.994**	16.054	False	(4.0, 0.997, 1.0)

Table 4.13: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(4, 1, 1)$ and 1-dimensional $(k = 1)$ memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**32.858**	31.630	False	(3.0, 0.992, 1.0, 1.0)
0.0005	0	**28.664**	26.867	False	(3.0, 0.987, 0.999, 1.0)
0.0010	0	**27.770**	25.976	False	(3.0, 0.991, 1.0, 1.0)
0.0050	0	**24.722**	23.199	False	(3.0, 0.951, 1.0, 1.0)
0.0100	0	**22.438**	21.541	False	(3.0, 0.951, 1.0, 1.0)
0.0500	0	**15.476**	14.094	False	(3.0, 0.991, 1.0, 1.0)
0.1000	0	**10.449**	9.754	False	(3.0, 0.994, 1.0, 1.0)
0.0000	5	**32.814**	31.653	False	(3.0, 0.989, 1.0, 1.0)
0.0005	5	**29.990**	28.232	False	(3.0, 0.978, 1.0, 1.0)
0.0010	5	**28.287**	26.886	False	(3.0, 0.973, 0.999, 1.0)
0.0050	5	**24.534**	23.102	False	(3.0, 0.991, 0.999, 1.0)
0.0100	5	**22.746**	21.089	False	(3.0, 0.97, 1.0, 1.0)
0.0500	5	**17.657**	16.006	False	(3.0, 0.954, 1.0, 1.0)
0.1000	5	**15.522**	13.611	False	(3.0, 0.974, 1.0, 1.0)
0.0000	10	**32.758**	31.616	False	(3.0, 0.983, 1.0, 1.0)
0.0005	10	**30.732**	29.299	False	(3.0, 0.991, 1.0, 1.0)
0.0010	10	**29.646**	27.905	False	(3.0, 0.978, 1.0, 1.0)
0.0050	10	**26.245**	24.650	False	(3.0, 0.988, 1.0, 1.0)
0.0100	10	**24.752**	23.050	False	(3.0, 0.989, 1.0, 1.0)
0.0500	10	**20.535**	18.469	False	(3.0, 0.97, 1.0, 1.0)
0.1000	10	**17.956**	16.515	False	(3.0, 0.968, 1.0, 1.0)

Table 4.14: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(3, 1, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**15.221**	14.935	False	(4.0, 0.994, 0.998)
0.0005	0	**14.878**	14.651	False	(4.0, 0.972, 0.999)
0.0010	0	**14.572**	14.400	False	(4.0, 0.997, 0.999)
0.0050	0	**13.284**	12.987	False	(4.0, 0.995, 1.0)
0.0100	0	**12.230**	11.835	False	(4.0, 0.985, 1.0)
0.0500	0	**8.459**	8.372	False	(4.0, 0.999, 1.0)
0.1000	0	**6.208**	5.974	False	(4.0, 0.982, 1.0)
0.0000	5	**15.209**	14.944	False	(4.0, 0.995, 0.997)
0.0005	5	**15.047**	14.797	False	(4.0, 0.994, 0.998)
0.0010	5	**14.861**	14.697	False	(4.0, 0.999, 0.996)
0.0050	5	**14.059**	13.880	False	(4.0, 0.978, 1.0)
0.0100	5	**13.342**	13.189	False	(4.0, 0.982, 0.999)
0.0500	5	**10.739**	10.137	False	(4.0, 0.97, 0.999)
0.1000	5	**9.098**	8.687	False	(4.0, 0.998, 0.998)
0.0000	10	**15.183**	14.900	False	(4.0, 1.0, 0.997)
0.0005	10	**15.072**	14.842	False	(4.0, 0.981, 1.0)
0.0010	10	**15.042**	14.760	False	(4.0, 0.994, 0.998)
0.0050	10	**14.436**	14.250	False	(4.0, 0.967, 0.997)
0.0100	10	**13.860**	13.647	False	(4.0, 0.991, 0.997)
0.0500	10	**11.311**	11.000	False	(4.0, 0.991, 1.0)
0.1000	10	**10.244**	9.792	False	(4.0, 0.986, 0.999)

Table 4.15: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(4, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**14.942**	14.533	False	(3.0, 1.0, 0.993, 0.995)
0.0005	0	**14.641**	14.366	False	(3.0, 0.946, 0.99, 1.0)
0.0010	0	**14.497**	14.449	False	(3.0, 1.0, 0.994, 1.0)
0.0050	0	**13.552**	13.111	False	(3.0, 0.965, 0.999, 1.0)
0.0100	0	**12.610**	12.292	False	(3.0, 1.0, 0.994, 1.0)
0.0500	0	8.531	**8.671**	False	(3.0, 0.925, 1.0, 0.998)
0.1000	0	**6.196**	6.187	False	(3.0, 1.0, 1.0, 1.0)
0.0000	5	**15.022**	14.851	False	(3.0, 1.0, 1.0, 0.999)
0.0005	5	**14.837**	14.750	False	(3.0, 1.0, 0.998, 1.0)
0.0010	5	**14.865**	14.395	False	(3.0, 0.961, 0.993, 0.997)
0.0050	5	**14.101**	13.743	False	(3.0, 1.0, 1.0, 0.988)
0.0100	5	**13.370**	13.119	False	(3.0, 1.0, 1.0, 1.0)
0.0500	5	**10.680**	10.515	False	(3.0, 0.914, 0.995, 1.0)
0.1000	5	**9.278**	9.010	False	(3.0, 0.965, 0.989, 1.0)
0.0000	10	**14.888**	14.847	False	(3.0, 1.0, 0.997, 0.996)
0.0005	10	14.780	**14.787**	False	(3.0, 1.0, 0.985, 1.0)
0.0010	10	**14.899**	14.428	False	(3.0, 1.0, 0.996, 1.0)
0.0050	10	**14.560**	13.975	False	(3.0, 1.0, 0.984, 0.999)
0.0100	10	**13.842**	13.477	False	(3.0, 1.0, 0.994, 0.996)
0.0500	10	**11.576**	11.388	False	(3.0, 1.0, 1.0, 0.992)
0.1000	10	**10.452**	10.212	False	(3.0, 1.0, 1.0, 0.993)

Table 4.16: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(3, 1, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**31.699**	29.546	False	(1.0, 3.0, 1.0, 0.999)
0.0005	0	**27.390**	25.177	False	(1.0, 3.0, 0.998, 1.0)
0.0010	0	**26.483**	23.728	False	(1.0, 3.0, 0.998, 1.0)
0.0050	0	**23.026**	20.163	False	(1.0, 3.0, 0.999, 1.0)
0.0100	0	**21.450**	18.568	False	(1.0, 3.0, 0.994, 1.0)
0.0500	0	**14.426**	12.551	False	(1.0, 3.0, 1.0, 1.0)
0.1000	0	**10.339**	8.568	False	(1.0, 3.0, 0.99, 1.0)
0.0000	5	**31.655**	29.669	False	(1.0, 3.0, 0.999, 1.0)
0.0005	5	**28.322**	26.021	False	(1.0, 3.0, 0.999, 1.0)
0.0010	5	**26.853**	24.611	False	(1.0, 3.0, 0.999, 1.0)
0.0050	5	**23.925**	21.306	False	(1.0, 3.0, 0.999, 1.0)
0.0100	5	**22.221**	19.342	False	(1.0, 3.0, 0.995, 1.0)
0.0500	5	**17.903**	14.616	False	(1.0, 3.0, 0.996, 1.0)
0.1000	5	**14.650**	12.278	False	(1.0, 3.0, 0.999, 1.0)
0.0000	10	**31.658**	29.546	False	(1.0, 3.0, 0.999, 0.999)
0.0005	10	**28.806**	26.943	False	(1.0, 3.0, 1.0, 1.0)
0.0010	10	**27.837**	25.843	False	(1.0, 3.0, 0.999, 1.0)
0.0050	10	**25.327**	22.395	False	(1.0, 3.0, 0.997, 1.0)
0.0100	10	**24.192**	20.557	False	(1.0, 3.0, 1.0, 1.0)
0.0500	10	**20.083**	16.441	False	(1.0, 3.0, 0.999, 1.0)
0.1000	10	**17.406**	14.299	False	(1.0, 3.0, 0.996, 1.0)

Table 4.17: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 3, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Laplacian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**31.797**	29.671	False	(4.0, 0.997, 1.0)
0.0005	0	**27.423**	23.974	False	(4.0, 0.997, 1.0)
0.0010	0	**25.727**	22.978	False	(4.0, 0.996, 1.0)
0.0050	0	**22.455**	19.560	False	(4.0, 0.999, 1.0)
0.0100	0	**21.021**	18.271	False	(4.0, 0.998, 1.0)
0.0500	0	**13.419**	11.906	False	(4.0, 0.998, 1.0)
0.1000	0	**9.957**	8.131	False	(4.0, 0.998, 1.0)
0.0000	5	**31.775**	29.700	False	(4.0, 1.0, 0.999)
0.0005	5	**29.221**	25.232	False	(4.0, 0.997, 1.0)
0.0010	5	**27.898**	23.917	False	(4.0, 0.996, 1.0)
0.0050	5	**23.431**	20.640	False	(4.0, 0.997, 1.0)
0.0100	5	**21.975**	19.091	False	(4.0, 1.0, 1.0)
0.0500	5	**17.319**	14.280	False	(4.0, 0.997, 1.0)
0.1000	5	**14.750**	11.893	False	(4.0, 0.999, 1.0)
0.0000	10	**31.796**	29.564	False	(4.0, 1.0, 0.999)
0.0005	10	**30.031**	26.928	False	(4.0, 0.992, 1.0)
0.0010	10	**29.301**	25.442	False	(4.0, 0.997, 1.0)
0.0050	10	**25.152**	21.890	False	(4.0, 0.997, 1.0)
0.0100	10	**23.686**	20.244	False	(4.0, 0.996, 1.0)
0.0500	10	**19.339**	16.301	False	(4.0, 0.999, 1.0)
0.1000	10	**16.651**	14.132	False	(4.0, 0.997, 1.0)

Table 4.18: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(4, 1, 1)$ and 1-dimensional $(k = 1)$ memoryless Laplacian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**14.698**	13.749	False	(1.0, 3.0, 1.0, 1.0)
0.0005	0	**14.435**	13.567	False	(1.0, 3.0, 0.996, 1.0)
0.0010	0	**14.140**	13.394	False	(1.0, 3.0, 0.997, 1.0)
0.0050	0	**13.593**	12.217	False	(1.0, 3.0, 0.994, 1.0)
0.0100	0	**12.566**	11.198	False	(1.0, 3.0, 0.997, 1.0)
0.0500	0	**8.581**	7.741	False	(1.0, 3.0, 0.99, 1.0)
0.1000	0	**5.925**	5.446	False	(1.0, 3.0, 0.982, 0.999)
0.0000	5	**15.007**	13.798	False	(1.0, 3.0, 0.994, 1.0)
0.0005	5	**14.808**	13.670	False	(1.0, 3.0, 0.996, 1.0)
0.0010	5	**14.590**	13.598	False	(1.0, 3.0, 0.988, 1.0)
0.0050	5	**13.997**	12.928	False	(1.0, 3.0, 0.989, 1.0)
0.0100	5	**13.300**	12.267	False	(1.0, 3.0, 0.998, 1.0)
0.0500	5	**10.562**	9.580	False	(1.0, 3.0, 0.99, 1.0)
0.1000	5	**9.323**	7.922	False	(1.0, 3.0, 0.987, 1.0)
0.0000	10	**14.805**	13.800	False	(1.0, 3.0, 1.0, 1.0)
0.0005	10	**14.879**	13.749	False	(1.0, 3.0, 0.995, 1.0)
0.0010	10	**14.732**	13.667	False	(1.0, 3.0, 0.992, 1.0)
0.0050	10	**14.152**	13.120	False	(1.0, 3.0, 0.998, 1.0)
0.0100	10	**13.444**	12.585	False	(1.0, 3.0, 1.0, 1.0)
0.0500	10	**11.653**	10.332	False	(1.0, 3.0, 0.992, 1.0)
0.1000	10	**10.620**	9.108	False	(1.0, 3.0, 0.996, 1.0)

Table 4.19: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 3, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Laplacian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**15.165**	13.772	False	(4.0, 0.992, 1.0)
0.0005	0	**14.907**	13.525	False	(4.0, 0.99, 1.0)
0.0010	0	**14.642**	13.257	False	(4.0, 0.982, 1.0)
0.0050	0	**13.461**	11.959	False	(4.0, 0.997, 1.0)
0.0100	0	**12.493**	11.063	False	(4.0, 0.998, 1.0)
0.0500	0	**8.616**	7.567	False	(4.0, 0.998, 1.0)
0.1000	0	**5.846**	5.327	False	(4.0, 0.99, 1.0)
0.0000	5	**15.412**	13.816	False	(4.0, 0.997, 1.0)
0.0005	5	**15.033**	13.662	False	(4.0, 0.989, 1.0)
0.0010	5	**15.075**	13.561	False	(4.0, 1.0, 1.0)
0.0050	5	**14.529**	12.843	False	(4.0, 0.983, 1.0)
0.0100	5	**13.768**	12.219	False	(4.0, 0.981, 1.0)
0.0500	5	**10.889**	9.318	False	(4.0, 0.978, 0.999)
0.1000	5	**9.079**	7.909	False	(4.0, 0.995, 1.0)
0.0000	10	**15.289**	13.829	False	(4.0, 0.997, 1.0)
0.0005	10	**15.039**	13.697	False	(4.0, 0.984, 1.0)
0.0010	10	**15.096**	13.674	False	(4.0, 0.984, 1.0)
0.0050	10	**14.456**	13.065	False	(4.0, 0.983, 1.0)
0.0100	10	**14.047**	12.587	False	(4.0, 0.997, 1.0)
0.0500	10	**11.913**	10.184	False	(4.0, 0.992, 1.0)
0.1000	10	**10.611**	9.025	False	(4.0, 0.99, 1.0)

Table 4.20: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(4, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Laplacian source.

	δ		
ϵ	0	5	10
0.0000	**1.04**	**1.033**	**1.013**
0.0005	**1.516**	**1.712**	**1.619**
0.0010	**1.382**	**2.017**	**2.12**
0.0050	**1.248**	**1.264**	**1.483**
0.0100	**1.544**	**0.763**	**1.458**
0.0500	**0.891**	**1.975**	**1.408**
0.1000	**0.838**	**1.833**	**0.941**

Table 4.21: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(4, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.0000	**0.285**	**0.265**	**0.282**
0.0005	**0.227**	**0.25**	**0.23**
0.0010	**0.172**	**0.164**	**0.283**
0.0050	**0.297**	**0.179**	**0.186**
0.0100	**0.394**	**0.153**	**0.214**
0.0500	**0.087**	**0.602**	**0.312**
0.1000	**0.233**	**0.411**	**0.453**

Table 4.22: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(4, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.000000	**1.228**	**1.16**	**1.142**
0.000500	**1.798**	**1.758**	**1.433**
0.001000	**1.794**	**1.4**	**1.741**
0.005000	**1.523**	**1.432**	**1.595**
0.010000	**0.897**	**1.657**	**1.703**
0.050000	**1.382**	**1.651**	**2.066**
0.100000	**0.696**	**1.911**	**1.441**

Table 4.23: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(3, 1, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.000000	**0.409**	**0.171**	**0.042**
0.000500	**0.275**	**0.088**	**-0.006**
0.001000	**0.049**	**0.47**	**0.472**
0.005000	**0.441**	**0.358**	**0.585**
0.010000	**0.317**	**0.251**	**0.364**
0.050000	**-0.14**	**0.165**	**0.188**
0.100000	**0.008**	**0.268**	**0.239**

Table 4.24: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(3, 1, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.0000	**2.246**	**2.3**	**2.303**
0.0005	**2.858**	**2.692**	**2.565**
0.0010	**2.412**	**2.624**	**2.678**
0.0050	**2.142**	**3.258**	**2.975**
0.0100	**1.739**	**3.189**	**3.251**
0.0500	**2.149**	**2.793**	**2.862**
0.1000	**1.204**	**2.461**	**2.474**

Table 4.25: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(3, 1, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Laplacian source.

	δ		
ϵ	0	5	10
0.0000	**-0.035**	**0.124**	**0.175**
0.0005	**-0.503**	**0.663**	**0.089**
0.0010	**0.366**	**0.503**	**0.184**
0.0050	**0.703**	**0.371**	**-0.728**
0.0100	**1.062**	**0.856**	**-0.176**
0.0500	**1.297**	**1.339**	**1.135**
0.1000	**0.793**	**0.772**	**0.66**

Table 4.26: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(3, 1, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Laplacian source.

	δ		
ϵ	0	5	10
0.0000	**2.126**	**2.074**	**2.231**
0.0005	**3.449**	**3.988**	**3.103**
0.0010	**2.75**	**3.981**	**3.859**
0.0050	**2.895**	**2.791**	**3.262**
0.0100	**2.75**	**2.884**	**3.441**
0.0500	**1.513**	**3.038**	**3.038**
0.1000	**1.826**	**2.857**	**2.518**

Table 4.27: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(4, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Laplacian source.

	δ		
ϵ	0	5	10
0.0000	**1.392**	**1.595**	**1.46**
0.0005	**1.382**	**1.371**	**1.342**
0.0010	**1.384**	**1.514**	**1.422**
0.0050	**1.502**	**1.686**	**1.392**
0.0100	**1.429**	**1.55**	**1.46**
0.0500	**1.049**	**1.571**	**1.729**
0.1000	**0.519**	**1.17**	**1.587**

Table 4.28: SNR gain of VR-ACOVQ over FR-ACOVQ for bit allocation $(4, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Laplacian source.

δ	Bit Allocation ϵ	(1, 1, 2)	(1, 2, 1)	(1, 3)	(2, 1, 1)	(3, 1)
0	0.0000	0.005	**0.152**	-0.070	0.017	**0.105**
	0.0005	-0.002	**0.152**	0.032	0.001	-0.001
	0.0010	0.005	**0.120**	0.035	-0.001	**0.275**
	0.0050	-0.000	**0.067**	-0.090	-0.028	**0.262**
	0.0100	-0.002	**0.225**	-0.006	0.003	**0.360**
	0.0500	-0.005	**0.010**	-0.001	-0.014	-0.004
	0.1000	-0.004	0.014	-0.001	0.002	**0.021**
5	0.0000	-0.004	**0.286**	-0.000	-0.000	**0.253**
	0.0005	0.002	**0.211**	0.003	-0.005	**0.195**
	0.0010	-0.011	**0.280**	-0.040	0.003	**0.037**
	0.0050	-0.009	**0.118**	0.045	0.002	**0.250**
	0.0100	0.003	**0.207**	-0.011	-0.004	**0.047**
	0.0500	-0.005	**0.066**	0.032	0.011	**0.060**
	0.1000	0.001	**-0.005**	-0.007	-0.016	**0.108**
10	0.0000	-0.000	**0.292**	0.000	0.000	**0.242**
	0.0005	0.002	-0.002	0.002	0.005	**0.048**
	0.0010	-0.002	**0.283**	0.033	-0.002	**0.056**
	0.0050	0.005	0.002	0.014	0.000	**0.134**
	0.0100	0.027	0.004	-0.000	-0.001	**0.115**
	0.0500	-0.014	**0.114**	-0.019	0.011	**0.002**
	0.1000	0.001	0.000	0.007	0.002	**0.135**

Table 4.29: SNR gain of VR-ACOVQ over FR-ACOVQ for various 4-bit bit allocations on a 2-dimensional ($k = 2$) memoryless Gaussian source.

δ	Bit Allocation ϵ	(1, 1, 4)	(1, 4, 1)	(4, 1, 1)
0	0.0000	-0.000	**0.404**	**0.285**
	0.0005	0.000	**0.438**	**0.227**
	0.0010	-0.004	**0.312**	**0.172**
	0.0050	-0.011	**0.362**	**0.297**
	0.0100	0.009	**0.253**	**0.394**
	0.0500	0.009	**0.310**	**0.087**
	0.1000	-0.016	**0.184**	**0.233**
5	0.0000	-0.000	**0.525**	**0.265**
	0.0005	-0.004	**0.413**	**0.250**
	0.0010	-0.010	**0.509**	**0.164**
	0.0050	0.008	**0.442**	**0.179**
	0.0100	-0.031	**0.386**	**0.153**
	0.0500	0.055	**0.325**	**0.602**
	0.1000	0.050	**0.307**	**0.411**
10	0.0000	-0.006	**0.436**	**0.282**
	0.0005	0.001	**0.481**	**0.230**
	0.0010	-0.006	**0.562**	**0.283**
	0.0050	0.024	**0.516**	**0.186**
	0.0100	-0.028	**0.369**	**0.214**
	0.0500	-0.053	**0.281**	**0.312**
	0.1000	-0.002	**0.256**	**0.453**

Table 4.30: SNR gain of VR-ACOVQ over FR-ACOVQ for various 6-bit bit allocations on a 2-dimensional ($k = 2$) memoryless Gaussian source.

Chapter 5

Conclusion

5.1 Summary of Work

This thesis explored the design and performance of JSCC schemes in communication systems with a noisy discrete channel with memory and noiseless feedback. The work focused on extending existing JSCC vector quantization schemes to such communication systems.

In Chapter 2, we introduced various discrete channel models, such as the Polya contagion and binary symmetric channel. We then introduced COVQ, ACOVQ, and CM-TSVQ along with their respective necessary conditions for optimality. Afterwards, we described the generalized LBG-algorithm to design locally optimal COVQs.

In Chapter 3, we extended the necessary conditions of optimality for CM-TSVQ in communication systems with noiseless feedback. We then showed that the nearest neighbor and centroid conditions are equivalent to those in ACOVQ. The simulations supported this claim as ACOVQ and ATSVQ with equivalent initializations converged to the same codebooks and encoding regions under the generalized LGB-algorithm and had near equivalent performances.

In Chapter 4, we leveraged the tree-structured nature of ACOVQ and explored various variable rate schemes for tree-strictured quantization that could be generalized to ACOVQ. We then extended the generalized BFOS algorithm, introduced in [27], to ACOVQ. Simulations showed that in general, the VR-ACOVQ outperformed FR-ACOVQ under the same average rate constraints at the cost of a higher encoding complexity. Further, simulations indicate that the performance gap between the quantizers increases with a higher rate and with a more biased source distribution.

5.2 Future Work

For ACOVQ, this thesis only considered discrete one way channels with a noiseless feedback link. Future work may include generalizing the ACOVQ scheme to account for communication systems with noisy feedback. Additionally, the VR-ACOVQ bit allocation algorithm in Chapter 4 determines a bit allocation given an average rate constraint for each stage and a predetermined number of stages. Additional work may include developing an algorithm to output a bit allocation that is constrained by an overall rate.

Appendix A

Proof for ATSVQ Generalized Centroid Condition Reduction

In this section we will prove that (3.26) under the square error distortion reduces to (3.27). Let $i \geq 2$ denote the quantizer stage. From (3.26) we have that

$$\mathbf{c}_{y_i|y^{i-1}}^{(i,AT)} = \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^k} E[d(\mathbf{U}, \mathbf{c}_{y_1}^{(1,AT)} + \ldots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)} + \boldsymbol{\omega})|Y^i = y^i], \tag{A.1}$$

given feedback $Y^i = y^i$ and fixed partitions $\mathcal{S}^{(1,AT)}, \mathcal{S}_{y_1}^{(2,AT)}, \ldots, \mathcal{S}_{y^{i-1}}^{(i,AT)}$ fixed codebooks $\mathcal{C}^{(1,AT)}, \mathcal{C}_{y_1}^{(2,AT)}, \ldots, \mathcal{C}_{y^{i-2}}^{(i-1,AT)}$, which under square error distortion becomes

$$\mathbf{c}_{y_i|y^{i-1}}^{(i,AT)} = \arg\min_{\boldsymbol{\omega}\in\mathbb{R}^k} E\left[\left\|\mathbf{U} - \left(\mathbf{c}_{y_1}^{(1,AT)} + \ldots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)} + \boldsymbol{\omega}\right)\right\|^2 \middle| Y^i = y^i\right]. \tag{A.2}$$

Because $h(\mathbf{x}) = \|\mathbf{x}\|^2$ is a convex function, we can find the global minimum of $\boldsymbol{\omega} \in \mathbb{R}^k$ by holding all codewords constant and finding the critical points of the function

$$g(\boldsymbol{\omega}) = E\left[\left\|\mathbf{U} - \left(\mathbf{c}_{y_1}^{(1,AT)} + \ldots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)} + \boldsymbol{\omega}\right)\right\|^2 \middle| Y^i = y^i\right]. \tag{A.3}$$

That is, the global minimum is given by

$$0 = \frac{\partial}{\partial \omega_i} g(\omega) \implies \frac{\partial}{\partial \omega_i} E\left[\left\|\mathbf{U} - \left(\mathbf{c}_{y_1}^{(1,AT)} + \ldots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)} + \boldsymbol{\omega}\right)\right\|^2 \middle| Y^i = y^i\right] = 0 \tag{A.4}$$

$$\implies E\left[\frac{\partial}{\partial \omega_i}\left\|\mathbf{U} - \left(\mathbf{c}_{y_1}^{(1,AT)} + \ldots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)} + \boldsymbol{\omega}\right)\right\|^2 \middle| Y^i = y^i\right] = 0 \tag{A.5}$$

$$\implies E\left[2\left(U_i - \left(\mathbf{c}_{y_1}^{(1,AT)} + \ldots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)} + \boldsymbol{\omega}\right)_i\right) \middle| Y^i = y^i\right] = 0 \tag{A.6}$$

$$\implies \omega_i = E[U_i|Y^i = y^i] - \left(\mathbf{c}_{y_1}^{(1,AT)} + \ldots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)}\right)_i, \tag{A.7}$$

for all $i = 1, \ldots, k$, where $(\mathbf{v})_i$ represents the ith component for any $\mathbf{v} \in \mathbb{R}^k$. Note that assuming the source is of finite variance, we can interchange the expectation and partial derivative in (A.5) by the dominated convergence theorem. Hence we have that

$$\boldsymbol{\omega} = E[\mathbf{U}|Y^i = y^i] - \left(\mathbf{c}_{y_1}^{(1,AT)} + \ldots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)}\right), \tag{A.8}$$

implying that under square error distortion, the centroid condition reduces to

$$\mathbf{c}_{y_i|y^{i-1}}^{(i,AT)} = E[\mathbf{U}|Y^i = y^i] - \left(\mathbf{c}_{y_1}^{(1,AT)} + \ldots + \mathbf{c}_{y_i|y^{i-1}}^{(i,AT)}\right). \tag{A.9}$$

Appendix B

Additional Simulation Results for VR-ACOVQ and FR-ACOVQ

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**32.683**	31.819	False	(1.0, 3.0, 0.999, 1.0)
0.0005	0	**28.832**	26.907	False	(1.0, 3.0, 0.996, 1.0)
0.0010	0	**27.871**	26.089	False	(1.0, 3.0, 0.996, 1.0)
0.0050	0	**24.017**	22.245	False	(1.0, 3.0, 1.0, 1.0)
0.0100	0	**22.446**	20.598	False	(1.0, 3.0, 0.995, 1.0)
0.0500	0	**15.658**	13.811	False	(1.0, 3.0, 0.994, 1.0)
0.1000	0	**10.815**	9.826	False	(1.0, 3.0, 0.999, 1.0)
0.0000	5	**32.695**	31.793	False	(1.0, 3.0, 1.0, 1.0)
0.0005	5	**28.766**	27.937	False	(1.0, 3.0, 0.992, 1.0)
0.0010	5	**27.650**	26.497	False	(1.0, 3.0, 0.994, 1.0)
0.0050	5	**25.560**	22.978	False	(1.0, 3.0, 0.992, 1.0)
0.0100	5	**23.675**	21.051	False	(1.0, 3.0, 0.983, 1.0)
0.0500	5	**18.839**	16.247	False	(1.0, 3.0, 0.998, 1.0)
0.1000	5	**15.658**	13.758	False	(1.0, 3.0, 0.993, 1.0)
0.0000	10	**32.718**	31.781	False	(1.0, 3.0, 1.0, 1.0)
0.0005	10	**29.897**	28.666	False	(1.0, 3.0, 0.997, 1.0)
0.0010	10	**28.544**	27.485	False	(1.0, 3.0, 0.996, 1.0)
0.0050	10	**26.007**	24.429	False	(1.0, 3.0, 0.996, 1.0)
0.0100	10	**25.249**	22.563	False	(1.0, 3.0, 0.988, 1.0)
0.0500	10	**20.793**	17.985	False	(1.0, 3.0, 1.0, 1.0)
0.1000	10	**17.952**	15.808	False	(1.0, 2.999, 1.0, 1.0)

Table B.1: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 3, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**32.810**	31.662	False	(1.0, 1.0, 2.99, 1.0)
0.0005	0	**29.380**	27.730	False	(1.0, 1.0, 2.991, 1.0)
0.0010	0	**28.723**	26.814	False	(1.0, 1.0, 2.993, 1.0)
0.0050	0	**25.245**	22.637	False	(1.0, 1.0, 3.0, 0.999)
0.0100	0	**22.871**	20.774	False	(1.0, 1.0, 2.844, 1.0)
0.0500	0	**16.193**	13.791	False	(1.0, 1.0, 3.0, 0.999)
0.1000	0	**10.977**	9.554	False	(1.0, 1.0, 3.0, 0.992)
0.0000	5	**32.768**	31.669	False	(1.0, 1.0, 2.99, 1.0)
0.0005	5	**29.162**	28.047	False	(1.0, 1.0, 2.992, 1.0)
0.0010	5	**27.893**	26.986	False	(1.0, 1.0, 2.992, 1.0)
0.0050	5	**25.326**	23.625	False	(1.0, 1.0, 3.0, 0.998)
0.0100	5	**23.757**	21.454	False	(1.0, 1.0, 3.0, 0.999)
0.0500	5	**19.061**	15.720	False	(1.0, 1.0, 3.0, 0.999)
0.1000	5	**15.699**	13.338	False	(1.0, 1.0, 3.0, 1.0)
0.0000	10	**32.799**	31.666	False	(1.0, 1.0, 2.989, 1.0)
0.0005	10	**29.955**	29.065	False	(1.0, 1.0, 2.99, 1.0)
0.0010	10	**28.984**	27.640	False	(1.0, 1.0, 2.992, 1.0)
0.0050	10	**26.968**	24.929	False	(1.0, 1.0, 3.0, 1.0)
0.0100	10	**25.149**	22.697	False	(1.0, 1.0, 3.0, 1.0)
0.0500	10	**20.985**	17.711	False	(1.0, 1.0, 3.0, 0.999)
0.1000	10	**18.524**	15.752	False	(1.0, 1.0, 3.0, 0.999)

Table B.2: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 3, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**32.652**	31.583	False	(1.0, 1.0, 1.0, 2.999)
0.0005	0	**29.207**	28.683	False	(1.0, 1.0, 1.0, 2.981)
0.0010	0	**27.772**	27.483	False	(1.0, 1.0, 1.0, 2.99)
0.0050	0	**23.663**	23.368	False	(1.0, 1.0, 1.0, 2.989)
0.0100	0	**21.102**	20.941	False	(1.0, 1.0, 1.0, 2.998)
0.0500	0	**13.542**	13.483	False	(1.0, 1.0, 1.0, 2.994)
0.1000	0	**9.441**	9.356	False	(1.0, 1.0, 1.0, 2.944)
0.0000	5	**32.675**	31.585	False	(1.0, 1.0, 1.0, 3.0)
0.0005	5	**29.030**	28.864	False	(1.0, 1.0, 1.0, 2.955)
0.0010	5	**27.631**	27.296	False	(1.0, 1.0, 1.0, 2.98)
0.0050	5	23.270	**23.442**	False	(1.0, 1.0, 1.0, 2.987)
0.0100	5	**21.365**	21.359	False	(1.0, 1.0, 1.0, 2.999)
0.0500	5	**16.108**	15.919	False	(1.0, 1.0, 0.807, 2.999)
0.1000	5	**14.312**	13.392	False	(1.0, 1.0, 1.0, 2.991)
0.0000	10	**32.650**	31.586	False	(1.0, 1.0, 1.0, 3.0)
0.0005	10	**29.876**	29.237	False	(1.0, 1.0, 1.0, 2.98)
0.0010	10	**28.715**	28.426	False	(1.0, 1.0, 1.0, 2.98)
0.0050	10	24.754	**24.920**	False	(1.0, 1.0, 1.0, 2.981)
0.0100	10	**22.875**	22.819	False	(1.0, 1.0, 1.0, 2.992)
0.0500	10	**18.991**	17.986	False	(1.0, 1.0, 1.0, 2.999)
0.1000	10	**16.581**	15.578	False	(1.0, 1.0, 1.0, 2.989)

Table B.3: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 1, 3)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**32.767**	31.597	False	(2.0, 2.0, 0.992, 1.0)
0.0005	0	**28.930**	28.178	False	(2.0, 2.0, 0.999, 1.0)
0.0010	0	**29.130**	26.708	False	(2.0, 1.995, 1.0, 1.0)
0.0050	0	**24.733**	23.667	False	(2.0, 1.891, 1.0, 1.0)
0.0100	0	**22.688**	21.430	False	(2.0, 1.84, 1.0, 1.0)
0.0500	0	**15.632**	13.870	False	(2.0, 1.957, 1.0, 1.0)
0.1000	0	**11.195**	9.742	False	(2.0, 2.0, 0.998, 1.0)
0.0000	5	**32.727**	31.609	False	(2.0, 2.0, 0.991, 1.0)
0.0005	5	**29.550**	28.185	False	(2.0, 2.0, 0.993, 1.0)
0.0010	5	**28.364**	27.056	False	(2.0, 1.999, 0.997, 1.0)
0.0050	5	**25.390**	23.421	False	(2.0, 1.999, 0.997, 1.0)
0.0100	5	**23.339**	21.586	False	(2.0, 1.849, 0.999, 1.0)
0.0500	5	**18.855**	16.225	False	(2.0, 1.989, 0.999, 1.0)
0.1000	5	**15.795**	13.889	False	(2.0, 1.88, 0.999, 1.0)
0.0000	10	**32.728**	31.594	False	(2.0, 2.0, 0.997, 1.0)
0.0005	10	**30.556**	29.349	False	(2.0, 2.0, 0.992, 1.0)
0.0010	10	**29.343**	27.932	False	(2.0, 1.998, 1.0, 1.0)
0.0050	10	**26.969**	24.915	False	(2.0, 1.994, 1.0, 1.0)
0.0100	10	**24.514**	23.361	False	(2.0, 1.866, 1.0, 1.0)
0.0500	10	**20.635**	18.699	False	(2.0, 1.964, 1.0, 1.0)
0.1000	10	**17.477**	16.295	False	(2.0, 1.844, 0.997, 1.0)

Table B.4: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(2, 2, 1, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**32.608**	31.700	False	(1.0, 2.0, 1.978, 1.0)
0.0005	0	**29.339**	27.841	False	(1.0, 2.0, 1.967, 1.0)
0.0010	0	**28.411**	27.112	False	(1.0, 2.0, 1.974, 0.999)
0.0050	0	**25.746**	23.229	False	(1.0, 2.0, 2.0, 1.0)
0.0100	0	**22.732**	21.099	False	(1.0, 2.0, 1.948, 1.0)
0.0500	0	**15.152**	13.579	False	(1.0, 2.0, 1.925, 1.0)
0.1000	0	**11.070**	9.515	False	(1.0, 2.0, 1.981, 1.0)
0.0000	5	**32.598**	31.706	False	(1.0, 2.0, 1.976, 1.0)
0.0005	5	**29.041**	27.734	False	(1.0, 2.0, 1.998, 1.0)
0.0010	5	**27.990**	26.959	False	(1.0, 2.0, 1.976, 1.0)
0.0050	5	**24.817**	23.219	False	(1.0, 2.0, 1.988, 1.0)
0.0100	5	**23.110**	21.458	False	(1.0, 2.0, 1.962, 1.0)
0.0500	5	**19.030**	16.294	False	(1.0, 2.0, 1.999, 1.0)
0.1000	5	**16.047**	13.552	False	(1.0, 2.0, 1.996, 1.0)
0.0000	10	**32.586**	31.680	False	(1.0, 2.0, 1.977, 1.0)
0.0005	10	**29.822**	28.913	False	(1.0, 2.0, 1.979, 1.0)
0.0010	10	**28.720**	27.739	False	(1.0, 2.0, 1.986, 1.0)
0.0050	10	**26.183**	24.752	False	(1.0, 2.0, 1.978, 1.0)
0.0100	10	**25.483**	22.995	False	(1.0, 2.0, 1.984, 1.0)
0.0500	10	**21.386**	18.367	False	(1.0, 2.0, 1.991, 1.0)
0.1000	10	**18.814**	15.936	False	(1.0, 2.0, 1.998, 1.0)

Table B.5: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 2, 2, 1)$ and 1-dimensional $(k = 1)$ memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**32.708**	31.628	False	(1.0, 1.0, 2.0, 1.993)
0.0005	0	**29.683**	28.750	False	(1.0, 1.0, 2.0, 1.993)
0.0010	0	**28.333**	27.282	False	(1.0, 1.0, 2.0, 1.998)
0.0050	0	**24.026**	23.133	False	(1.0, 1.0, 1.835, 1.999)
0.0100	0	**22.446**	21.439	False	(1.0, 1.0, 2.0, 1.986)
0.0500	0	**14.995**	13.472	False	(1.0, 1.0, 2.0, 1.994)
0.1000	0	**10.355**	9.335	False	(1.0, 1.0, 2.0, 1.99)
0.0000	5	**32.686**	31.603	False	(1.0, 1.0, 2.0, 1.991)
0.0005	5	**29.408**	28.572	False	(1.0, 1.0, 2.0, 2.0)
0.0010	5	**27.992**	27.453	False	(1.0, 1.0, 2.0, 1.999)
0.0050	5	**23.564**	23.432	False	(1.0, 1.0, 1.835, 2.0)
0.0100	5	**21.843**	21.499	False	(1.0, 1.0, 1.862, 2.0)
0.0500	5	**18.185**	16.072	False	(1.0, 1.0, 2.0, 1.997)
0.1000	5	**15.275**	13.518	False	(1.0, 1.0, 1.965, 1.989)
0.0000	10	**32.672**	31.648	False	(1.0, 1.0, 2.0, 1.99)
0.0005	10	**29.980**	29.239	False	(1.0, 1.0, 2.0, 1.994)
0.0010	10	**28.821**	28.259	False	(1.0, 1.0, 2.0, 1.993)
0.0050	10	**26.262**	25.129	False	(1.0, 1.0, 2.0, 1.994)
0.0100	10	**24.651**	23.168	False	(1.0, 1.0, 2.0, 1.991)
0.0500	10	**20.658**	18.135	False	(1.0, 1.0, 2.0, 1.998)
0.1000	10	**18.070**	15.896	False	(1.0, 1.0, 2.0, 1.998)

Table B.6: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 2, 2)$ and 1-dimensional $(k = 1)$ memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**32.711**	31.824	False	(1.0, 4.0, 1.0)
0.0005	0	**28.791**	26.405	False	(1.0, 4.0, 0.999)
0.0010	0	**27.629**	24.389	False	(1.0, 4.0, 1.0)
0.0050	0	**23.509**	21.131	False	(1.0, 4.0, 1.0)
0.0100	0	**21.323**	19.407	False	(1.0, 4.0, 0.999)
0.0500	0	**15.268**	13.175	False	(1.0, 4.0, 0.998)
0.1000	0	**10.469**	9.329	False	(1.0, 4.0, 0.999)
0.0000	5	**32.704**	31.823	False	(1.0, 4.0, 0.999)
0.0005	5	**29.032**	27.702	False	(1.0, 4.0, 0.998)
0.0010	5	**27.712**	26.365	False	(1.0, 4.0, 0.993)
0.0050	5	**25.034**	22.573	False	(1.0, 4.0, 0.998)
0.0100	5	**23.906**	20.624	False	(1.0, 4.0, 0.999)
0.0500	5	**18.889**	14.665	False	(1.0, 4.0, 1.0)
0.1000	5	**16.055**	12.802	False	(1.0, 4.0, 0.999)
0.0000	10	**32.669**	31.840	False	(1.0, 4.0, 1.0)
0.0005	10	**29.684**	28.423	False	(1.0, 4.0, 1.0)
0.0010	10	**28.622**	27.572	False	(1.0, 4.0, 0.998)
0.0050	10	**26.144**	24.222	False	(1.0, 4.0, 1.0)
0.0100	10	**24.676**	22.137	False	(1.0, 4.0, 0.995)
0.0500	10	**20.072**	17.205	False	(1.0, 4.0, 1.0)
0.1000	10	**17.824**	15.249	False	(1.0, 4.0, 1.0)

Table B.7: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 4, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**31.887**	31.679	False	(1.0, 1.0, 3.991)
0.0005	0	27.901	**27.948**	False	(1.0, 1.0, 3.99)
0.0010	0	**26.491**	26.383	False	(1.0, 1.0, 3.992)
0.0050	0	**22.337**	22.261	True	(1.0, 1.0, 4.0)
0.0100	0	**19.938**	19.835	True	(1.0, 1.0, 4.0)
0.0500	0	12.765	**12.783**	True	(1.0, 1.0, 4.0)
0.1000	0	**8.852**	8.777	True	(1.0, 1.0, 4.0)
0.0000	5	**31.945**	31.689	False	(1.0, 1.0, 3.993)
0.0005	5	**28.631**	28.209	False	(1.0, 1.0, 3.992)
0.0010	5	**27.090**	26.802	False	(1.0, 1.0, 3.994)
0.0050	5	22.853	**23.029**	True	(1.0, 1.0, 4.0)
0.0100	5	20.840	**20.893**	True	(1.0, 1.0, 4.0)
0.0500	5	15.253	**15.320**	True	(1.0, 1.0, 4.0)
0.1000	5	**12.739**	12.712	True	(1.0, 1.0, 4.0)
0.0000	10	**31.906**	31.698	False	(1.0, 1.0, 3.989)
0.0005	10	29.118	**29.128**	False	(1.0, 1.0, 3.989)
0.0010	10	27.746	**27.827**	True	(1.0, 1.0, 4.0)
0.0050	10	24.300	**24.550**	True	(1.0, 1.0, 4.0)
0.0100	10	22.309	**22.376**	True	(1.0, 1.0, 4.0)
0.0500	10	17.123	**17.306**	True	(1.0, 1.0, 4.0)
0.1000	10	**14.852**	14.736	True	(1.0, 1.0, 4.0)

Table B.8: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 4)$ and 1-dimensional $(k = 1)$ memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**15.168**	14.904	False	(1.0, 3.0, 0.978, 0.999)
0.0005	0	**14.878**	14.666	False	(1.0, 3.0, 1.0, 1.0)
0.0010	0	**14.633**	14.483	False	(1.0, 3.0, 1.0, 0.997)
0.0050	0	**13.599**	13.340	False	(1.0, 3.0, 0.996, 0.996)
0.0100	0	**12.449**	12.284	False	(1.0, 3.0, 0.999, 0.999)
0.0500	0	**8.905**	8.533	False	(1.0, 3.0, 0.992, 0.999)
0.1000	0	**6.152**	6.087	False	(1.0, 3.0, 0.981, 0.999)
0.0000	5	**15.051**	14.918	False	(1.0, 3.0, 0.971, 0.999)
0.0005	5	**14.973**	14.781	False	(1.0, 3.0, 0.997, 0.999)
0.0010	5	**14.874**	14.650	False	(1.0, 3.0, 1.0, 0.999)
0.0050	5	**14.033**	13.882	False	(1.0, 3.0, 0.991, 0.997)
0.0100	5	**13.337**	13.148	False	(1.0, 3.0, 0.996, 0.999)
0.0500	5	**10.644**	10.456	False	(1.0, 3.0, 0.997, 1.0)
0.1000	5	**9.425**	8.825	False	(1.0, 3.0, 0.991, 1.0)
0.0000	10	**15.223**	14.900	False	(1.0, 3.0, 1.0, 0.998)
0.0005	10	**14.931**	14.727	False	(1.0, 3.0, 0.98, 0.999)
0.0010	10	**14.837**	14.693	False	(1.0, 3.0, 1.0, 0.996)
0.0050	10	**14.322**	14.122	False	(1.0, 3.0, 0.977, 0.997)
0.0100	10	**13.665**	13.478	False	(1.0, 3.0, 0.967, 0.998)
0.0500	10	**11.762**	11.462	False	(1.0, 3.0, 0.998, 1.0)
0.1000	10	**10.585**	10.082	False	(1.0, 3.0, 0.995, 1.0)

Table B.9: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 3, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**14.992**	14.592	False	(1.0, 1.0, 3.0, 0.991)
0.0005	0	**14.771**	14.436	False	(1.0, 1.0, 3.0, 0.992)
0.0010	0	**14.586**	14.288	False	(1.0, 1.0, 3.0, 0.996)
0.0050	0	**13.559**	13.232	False	(1.0, 1.0, 3.0, 0.994)
0.0100	0	**12.463**	12.180	False	(1.0, 1.0, 3.0, 0.99)
0.0500	0	**8.728**	8.611	False	(1.0, 1.0, 3.0, 0.999)
0.1000	0	**6.178**	6.084	False	(1.0, 1.0, 3.0, 0.997)
0.0000	5	**15.116**	14.598	False	(1.0, 1.0, 3.0, 0.994)
0.0005	5	**14.861**	14.492	False	(1.0, 1.0, 3.0, 0.996)
0.0010	5	**14.769**	14.390	False	(1.0, 1.0, 3.0, 0.997)
0.0050	5	**14.094**	13.708	False	(1.0, 1.0, 3.0, 0.998)
0.0100	5	**13.389**	13.036	False	(1.0, 1.0, 3.0, 0.991)
0.0500	5	**10.647**	10.314	False	(1.0, 1.0, 3.0, 0.997)
0.1000	5	**9.116**	8.745	False	(1.0, 1.0, 3.0, 0.997)
0.0000	10	**14.994**	14.597	False	(1.0, 1.0, 3.0, 0.996)
0.0005	10	**14.960**	14.492	False	(1.0, 1.0, 3.0, 0.998)
0.0010	10	**14.838**	14.424	False	(1.0, 1.0, 3.0, 0.993)
0.0050	10	**14.302**	13.890	False	(1.0, 1.0, 3.0, 0.99)
0.0100	10	**13.672**	13.352	False	(1.0, 1.0, 3.0, 0.999)
0.0500	10	**11.758**	11.203	False	(1.0, 1.0, 3.0, 1.0)
0.1000	10	**10.318**	9.977	False	(1.0, 1.0, 3.0, 1.0)

Table B.10: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 3, 1)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**15.208**	15.123	False	(1.0, 1.0, 0.999, 2.996)
0.0005	0	**14.923**	14.921	True	(1.0, 1.0, 1.0, 3.0)
0.0010	0	**14.809**	14.731	False	(1.0, 1.0, 1.0, 2.997)
0.0050	0	**13.468**	13.456	True	(1.0, 1.0, 1.0, 3.0)
0.0100	0	12.311	**12.355**	False	(1.0, 1.0, 1.0, 2.952)
0.0500	0	8.455	**8.515**	True	(1.0, 1.0, 1.0, 3.0)
0.1000	0	**6.030**	6.021	True	(1.0, 1.0, 1.0, 3.0)
0.0000	5	**15.122**	15.122	True	(1.0, 1.0, 1.0, 3.0)
0.0005	5	14.989	**14.999**	True	(1.0, 1.0, 1.0, 3.0)
0.0010	5	**15.053**	14.878	False	(1.0, 1.0, 1.0, 2.992)
0.0050	5	**14.102**	14.102	True	(1.0, 1.0, 1.0, 3.0)
0.0100	5	**13.346**	13.322	True	(1.0, 1.0, 1.0, 3.0)
0.0500	5	10.557	**10.558**	True	(1.0, 1.0, 1.0, 3.0)
0.1000	5	**8.987**	8.966	True	(1.0, 1.0, 1.0, 3.0)
0.0000	10	**15.281**	15.108	False	(1.0, 1.0, 0.994, 2.993)
0.0005	10	**15.010**	15.007	True	(1.0, 1.0, 1.0, 3.0)
0.0010	10	**14.935**	14.930	True	(1.0, 1.0, 1.0, 3.0)
0.0050	10	**14.325**	14.301	True	(1.0, 1.0, 1.0, 3.0)
0.0100	10	13.692	**13.733**	True	(1.0, 1.0, 1.0, 3.0)
0.0500	10	11.503	**11.543**	True	(1.0, 1.0, 1.0, 3.0)
0.1000	10	**10.178**	10.167	True	(1.0, 1.0, 1.0, 3.0)

Table B.11: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 1, 3)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**14.996**	14.825	False	(2.0, 2.0, 0.992, 1.0)
0.0005	0	**14.812**	14.657	False	(2.0, 2.0, 0.999, 1.0)
0.0010	0	**14.727**	14.506	False	(2.0, 2.0, 0.995, 1.0)
0.0050	0	**13.561**	13.424	False	(2.0, 2.0, 0.971, 0.997)
0.0100	0	**12.500**	12.395	False	(2.0, 2.0, 0.99, 0.998)
0.0500	0	**9.027**	8.883	False	(2.0, 2.0, 1.0, 0.989)
0.1000	0	**6.390**	6.343	False	(2.0, 2.0, 1.0, 0.988)
0.0000	5	**15.120**	14.826	False	(2.0, 2.0, 0.993, 0.998)
0.0005	5	**15.092**	14.717	False	(2.0, 2.0, 0.992, 1.0)
0.0010	5	**14.656**	14.634	False	(2.0, 2.0, 1.0, 0.997)
0.0050	5	**14.239**	13.932	False	(2.0, 2.0, 0.976, 0.999)
0.0100	5	**13.477**	13.254	False	(2.0, 2.0, 0.996, 1.0)
0.0500	5	**10.746**	10.671	False	(2.0, 2.0, 1.0, 0.997)
0.1000	5	**9.319**	8.966	False	(2.0, 2.0, 0.981, 1.0)
0.0000	10	**15.113**	14.821	False	(2.0, 2.0, 0.992, 0.999)
0.0005	10	**14.976**	14.750	False	(2.0, 2.0, 0.985, 1.0)
0.0010	10	**14.887**	14.679	False	(2.0, 2.0, 0.99, 0.999)
0.0050	10	**14.294**	14.128	False	(2.0, 2.0, 0.982, 0.998)
0.0100	10	**13.691**	13.553	False	(2.0, 2.0, 0.998, 0.998)
0.0500	10	**11.734**	11.640	False	(2.0, 2.0, 0.99, 1.0)
0.1000	10	**10.451**	10.210	False	(2.0, 2.0, 0.991, 1.0)

Table B.12: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(2, 2, 1, 1)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**15.036**	14.528	False	(1.0, 2.0, 1.942, 1.0)
0.0005	0	**14.813**	14.373	False	(1.0, 2.0, 1.944, 0.999)
0.0010	0	**14.635**	14.227	False	(1.0, 2.0, 1.947, 0.997)
0.0050	0	**13.572**	13.205	False	(1.0, 2.0, 1.962, 1.0)
0.0100	0	**12.456**	12.320	False	(1.0, 2.0, 1.973, 0.999)
0.0500	0	**8.976**	8.815	False	(1.0, 2.0, 2.0, 0.992)
0.1000	0	**6.312**	6.227	False	(1.0, 2.0, 2.0, 0.998)
0.0000	5	**15.029**	14.536	False	(1.0, 2.0, 1.941, 0.999)
0.0005	5	**14.775**	14.401	False	(1.0, 2.0, 1.943, 0.996)
0.0010	5	**14.747**	14.350	False	(1.0, 2.0, 1.944, 0.996)
0.0050	5	**14.044**	13.679	False	(1.0, 2.0, 1.95, 1.0)
0.0100	5	**13.411**	13.044	False	(1.0, 2.0, 1.986, 0.999)
0.0500	5	**10.788**	10.531	False	(1.0, 2.0, 2.0, 0.999)
0.1000	5	**9.288**	9.091	False	(1.0, 2.0, 1.998, 0.996)
0.0000	10	**14.961**	14.546	False	(1.0, 2.0, 1.942, 1.0)
0.0005	10	**14.940**	14.459	False	(1.0, 2.0, 1.971, 0.998)
0.0010	10	**14.873**	14.383	False	(1.0, 2.0, 1.972, 0.996)
0.0050	10	**14.280**	13.845	False	(1.0, 2.0, 1.95, 0.998)
0.0100	10	**13.716**	13.374	False	(1.0, 2.0, 1.957, 0.996)
0.0500	10	**11.576**	11.320	False	(1.0, 2.0, 2.0, 1.0)
0.1000	10	**10.492**	10.231	False	(1.0, 2.0, 2.0, 0.995)

Table B.13: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 2, 2, 1)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**15.428**	14.942	False	(1.0, 1.0, 2.0, 1.983)
0.0005	0	**15.146**	14.681	False	(1.0, 1.0, 2.0, 2.0)
0.0010	0	**15.079**	14.585	False	(1.0, 1.0, 2.0, 1.99)
0.0050	0	**13.851**	13.470	False	(1.0, 1.0, 2.0, 1.997)
0.0100	0	**12.758**	12.475	False	(1.0, 1.0, 2.0, 1.975)
0.0500	0	**8.837**	8.837	True	(1.0, 1.0, 2.0, 2.0)
0.1000	0	6.240	**6.246**	True	(1.0, 1.0, 2.0, 2.0)
0.0000	5	**15.428**	14.917	False	(1.0, 1.0, 2.0, 1.982)
0.0005	5	**15.339**	14.849	False	(1.0, 1.0, 2.0, 1.985)
0.0010	5	**15.229**	14.667	False	(1.0, 1.0, 2.0, 1.99)
0.0050	5	**14.382**	13.974	False	(1.0, 1.0, 2.0, 1.968)
0.0100	5	**13.628**	13.294	False	(1.0, 1.0, 2.0, 1.973)
0.0500	5	**10.849**	10.722	False	(1.0, 1.0, 2.0, 1.998)
0.1000	5	**9.403**	9.002	False	(1.0, 1.0, 2.0, 1.995)
0.0000	10	**15.438**	14.953	False	(1.0, 1.0, 2.0, 1.982)
0.0005	10	**15.211**	14.825	False	(1.0, 1.0, 2.0, 1.979)
0.0010	10	**15.290**	14.788	False	(1.0, 1.0, 2.0, 1.988)
0.0050	10	**14.686**	14.197	False	(1.0, 1.0, 2.0, 1.998)
0.0100	10	**14.047**	13.657	False	(1.0, 1.0, 2.0, 1.999)
0.0500	10	**11.820**	11.671	False	(1.0, 1.0, 2.0, 1.975)
0.1000	10	**10.537**	10.311	False	(1.0, 1.0, 2.0, 1.987)

Table B.14: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 2, 2)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	**15.016**	14.611	False	(1.0, 4.0, 0.992)
0.0005	0	**14.825**	14.387	False	(1.0, 4.0, 0.993)
0.0010	0	**14.465**	14.153	False	(1.0, 4.0, 0.996)
0.0050	0	**13.173**	12.811	False	(1.0, 4.0, 0.999)
0.0100	0	**11.976**	11.722	False	(1.0, 4.0, 0.996)
0.0500	0	**8.516**	8.206	False	(1.0, 4.0, 0.997)
0.1000	0	**6.039**	5.856	False	(1.0, 4.0, 0.999)
0.0000	5	**15.135**	14.610	False	(1.0, 4.0, 0.999)
0.0005	5	**14.895**	14.481	False	(1.0, 4.0, 1.0)
0.0010	5	**14.876**	14.367	False	(1.0, 4.0, 0.989)
0.0050	5	**13.877**	13.436	False	(1.0, 4.0, 0.991)
0.0100	5	**13.108**	12.722	False	(1.0, 4.0, 0.988)
0.0500	5	**10.428**	10.103	False	(1.0, 4.0, 0.997)
0.1000	5	**8.747**	8.440	False	(1.0, 4.0, 0.999)
0.0000	10	**15.017**	14.582	False	(1.0, 4.0, 0.999)
0.0005	10	**14.919**	14.438	False	(1.0, 4.0, 0.998)
0.0010	10	**14.917**	14.355	False	(1.0, 4.0, 0.993)
0.0050	10	**14.241**	13.726	False	(1.0, 4.0, 0.999)
0.0100	10	**13.531**	13.162	False	(1.0, 4.0, 0.997)
0.0500	10	**11.218**	10.937	False	(1.0, 4.0, 0.998)
0.1000	10	**9.891**	9.634	False	(1.0, 4.0, 0.999)

Table B.15: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 4, 1)$ and 2-dimensional $(k = 2)$ memoryless Gaussian source.

ϵ	δ	VR-ACOVQ SNR (dB)	FR-ACOVQ SNR (dB)	Balanced Tree	Bit Allocation Average
0.0000	0	15.173	**15.173**	True	(1.0, 1.0, 4.0)
0.0005	0	**14.946**	14.946	True	(1.0, 1.0, 4.0)
0.0010	0	14.726	**14.730**	True	(1.0, 1.0, 4.0)
0.0050	0	13.364	**13.374**	True	(1.0, 1.0, 4.0)
0.0100	0	**12.196**	12.188	True	(1.0, 1.0, 4.0)
0.0500	0	**8.231**	8.222	True	(1.0, 1.0, 4.0)
0.1000	0	5.717	**5.733**	True	(1.0, 1.0, 4.0)
0.0000	5	15.164	**15.164**	True	(1.0, 1.0, 4.0)
0.0005	5	15.038	**15.042**	True	(1.0, 1.0, 4.0)
0.0010	5	14.910	**14.920**	True	(1.0, 1.0, 4.0)
0.0050	5	**14.118**	14.111	True	(1.0, 1.0, 4.0)
0.0100	5	13.303	**13.334**	True	(1.0, 1.0, 4.0)
0.0500	5	**10.386**	10.332	True	(1.0, 1.0, 4.0)
0.1000	5	**8.600**	8.551	True	(1.0, 1.0, 4.0)
0.0000	10	15.169	**15.175**	True	(1.0, 1.0, 4.0)
0.0005	10	**15.063**	15.062	True	(1.0, 1.0, 4.0)
0.0010	10	14.962	**14.968**	True	(1.0, 1.0, 4.0)
0.0050	10	**14.321**	14.298	True	(1.0, 1.0, 4.0)
0.0100	10	13.718	**13.746**	True	(1.0, 1.0, 4.0)
0.0500	10	11.257	**11.310**	True	(1.0, 1.0, 4.0)
0.1000	10	9.822	**9.824**	True	(1.0, 1.0, 4.0)

Table B.16: VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 4)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**0.864**	**0.903**	**0.938**
0.00050	**1.924**	**0.829**	**1.231**
0.00100	**1.781**	**1.153**	**1.059**
0.00500	**1.772**	**2.581**	**1.578**
0.01000	**1.848**	**2.624**	**2.686**
0.05000	**1.847**	**2.593**	**2.808**
0.10000	**0.99**	**1.9**	**2.143**

Table B.17: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1,3,1,1)$ and 1-dimensional $(k=1)$ memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**0.264**	**0.134**	**0.322**
0.00050	**0.212**	**0.193**	**0.205**
0.00100	**0.151**	**0.224**	**0.144**
0.00500	**0.259**	**0.151**	**0.2**
0.01000	**0.165**	**0.189**	**0.187**
0.05000	**0.372**	**0.188**	**0.3**
0.10000	**0.065**	**0.6**	**0.503**

Table B.18: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1,3,1,1)$ and 2-dimensional $(k=2)$ memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**1.148**	**1.099**	**1.133**
0.00050	**1.65**	**1.115**	**0.89**
0.00100	**1.909**	**0.906**	**1.345**
0.00500	**2.608**	**1.7**	**2.039**
0.01000	**2.097**	**2.303**	**2.453**
0.05000	**2.402**	**3.342**	**3.274**
0.10000	**1.423**	**2.361**	**2.772**

Table B.19: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1,1,3,1)$ and 1-dimensional $(k=1)$ memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**0.399**	**0.519**	**0.397**
0.00050	**0.335**	**0.37**	**0.468**
0.00100	**0.297**	**0.379**	**0.414**
0.00500	**0.327**	**0.386**	**0.411**
0.01000	**0.283**	**0.353**	**0.32**
0.05000	**0.117**	**0.333**	**0.555**
0.10000	**0.094**	**0.371**	**0.341**

Table B.20: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1,1,3,1)$ and 2-dimensional $(k=2)$ memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**1.069**	**1.09**	**1.063**
0.00050	**0.524**	**0.166**	**0.639**
0.00100	**0.289**	**0.335**	**0.289**
0.00500	**0.295**	**-0.172**	**-0.166**
0.01000	**0.162**	**0.006**	**0.055**
0.05000	**0.059**	**0.189**	**1.005**
0.10000	**0.084**	**0.92**	**1.003**

Table B.21: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 1, 3)$ and 1-dimensional $(k = 1)$ memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**0.086**	0.000	**0.173**
0.00050	0.002	-0.009	0.003
0.00100	**0.077**	**0.175**	0.004
0.00500	0.012	0.001	0.024
0.01000	**-0.044**	0.023	-0.042
0.05000	-0.061	-0.000	-0.040
0.10000	0.009	0.020	0.011

Table B.22: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 1, 3)$ and 2-dimensional $(k = 2)$ memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**1.169**	**1.118**	**1.134**
0.00050	**0.752**	**1.365**	**1.207**
0.00100	**2.422**	**1.309**	**1.411**
0.00500	**1.065**	**1.969**	**2.054**
0.01000	**1.258**	**1.753**	**1.153**
0.05000	**1.762**	**2.63**	**1.936**
0.10000	**1.453**	**1.906**	**1.183**

Table B.23: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(2, 2, 1, 1)$ and 1-dimensional $(k = 1)$ memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**0.171**	**0.295**	**0.292**
0.00050	**0.154**	**0.375**	**0.226**
0.00100	**0.221**	**0.021**	**0.208**
0.00500	**0.137**	**0.307**	**0.167**
0.01000	**0.105**	**0.222**	**0.138**
0.05000	**0.144**	**0.075**	**0.095**
0.10000	**0.047**	**0.353**	**0.241**

Table B.24: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(2, 2, 1, 1)$ and 2-dimensional $(k = 2)$ memoryless Gaussian source.

ϵ	δ 0	5	10
0.00000	**0.908**	**0.892**	**0.906**
0.00050	**1.498**	**1.306**	**0.909**
0.00100	**1.3**	**1.031**	**0.982**
0.00500	**2.517**	**1.598**	**1.431**
0.01000	**1.633**	**1.651**	**2.488**
0.05000	**1.573**	**2.736**	**3.019**
0.10000	**1.556**	**2.495**	**2.879**

Table B.25: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 2, 2, 1)$ and 1-dimensional $(k = 1)$ memoryless Gaussian source.

ϵ	δ 0	5	10
0.00000	**0.508**	**0.493**	**0.415**
0.00050	**0.44**	**0.374**	**0.481**
0.00100	**0.408**	**0.397**	**0.49**
0.00500	**0.368**	**0.365**	**0.434**
0.01000	**0.136**	**0.367**	**0.342**
0.05000	**0.161**	**0.257**	**0.256**
0.10000	**0.085**	**0.197**	**0.261**

Table B.26: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 2, 2, 1)$ and 2-dimensional $(k = 2)$ memoryless Gaussian source.

ϵ	δ 0	5	10
0.00000	**1.08**	**1.083**	**1.024**
0.00050	**0.933**	**0.836**	**0.741**
0.00100	**1.051**	**0.539**	**0.561**
0.00500	**0.893**	**0.132**	**1.134**
0.01000	**1.007**	**0.343**	**1.483**
0.05000	**1.523**	**2.113**	**2.523**
0.10000	**1.02**	**1.757**	**2.173**

Table B.27: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 2, 2)$ and 1-dimensional $(k = 1)$ memoryless Gaussian source.

ϵ	δ 0	5	10
0.00000	**0.486**	**0.511**	**0.486**
0.00050	**0.465**	**0.49**	**0.386**
0.00100	**0.494**	**0.562**	**0.501**
0.00500	**0.382**	**0.408**	**0.488**
0.01000	**0.283**	**0.334**	**0.39**
0.05000	0.000000	**0.126**	**0.15**
0.10000	-0.006000	**0.401**	**0.226**

Table B.28: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 2, 2)$ and 2-dimensional $(k = 2)$ memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**0.887**	**0.881**	**0.829**
0.00050	**2.386**	**1.33**	**1.261**
0.00100	**3.24**	**1.348**	**1.051**
0.00500	**2.378**	**2.46**	**1.922**
0.01000	**1.915**	**3.282**	**2.539**
0.05000	**2.094**	**4.224**	**2.867**
0.10000	**1.141**	**3.253**	**2.575**

Table B.29: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 4, 1)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**0.404**	**0.525**	**0.436**
0.00050	**0.438**	**0.413**	**0.481**
0.00100	**0.312**	**0.509**	**0.562**
0.00500	**0.362**	**0.442**	**0.516**
0.01000	**0.253**	**0.386**	**0.369**
0.05000	**0.31**	**0.325**	**0.281**
0.10000	**0.184**	**0.307**	**0.256**

Table B.30: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 4, 1)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	**0.208**	**0.256**	**0.208**
0.00050	**-0.047**	**0.422**	**-0.01**
0.00100	**0.107**	**0.288**	-0.080
0.00500	0.075	-0.175	-0.250
0.01000	0.103	-0.053	-0.067
0.05000	-0.018	-0.067	-0.183
0.10000	0.0750	0.027	0.116

Table B.31: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 4)$ and 1-dimensional ($k = 1$) memoryless Gaussian source.

	δ		
ϵ	0	5	10
0.00000	-0.000	-0.000	-0.006
0.00050	0.000	-0.004	0.001
0.00100	-0.004	-0.010	-0.006
0.00500	-0.011	0.008	0.024
0.01000	0.009	-0.031	-0.028
0.05000	0.009	0.055	-0.053
0.10000	-0.016	0.050	-0.002

Table B.32: Difference between VR-ACOVQ and FR-ACOVQ SNRs for bit allocation $(1, 1, 4)$ and 2-dimensional ($k = 2$) memoryless Gaussian source.

Bibliography

[1] F. Alajaji. Feedback does not increase the capacity of discrete channels with additive noise. *IEEE Transactions on Information Theory*, 41(2):546–549, 1995.

[2] F. Alajaji and P.-N. Chen. *An Introduction to Single-User Information Theory.* Springer Undergraduate Texts in Mathematics and Technology. Springer, Singapore, 1st edition, 2018.

[3] F. Alajaji and T. Fuja. A communication channel modeled on contagion. *IEEE Transactions on Information Theory*, 40(6):2035–2041, 1994.

[4] F. Alajaji, N. Phamdo, N. Farvardin, and T. E. Fuja. Detection of binary Markov sources over channels with additive Markov noise. *IEEE Transactions on Information Theory*, 42(1):230–239, 1996.

[5] A. S. Amanullah and M. Salehi. Joint source-channel coding in the presence of feedback. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pages 930–934 vol.2, 1993.

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[7] J. Cheng. *Channel Optimized Quantization of Images Over Binary Channels with Memory*. PhD thesis, Queen's University, 1997.

[8] J. Cheng and F. Alajaji. Channel optimized quantization of images over bursty channels. In *Proceedings of the Canadian Workshop on Information Theory (CWIT)*, 1997.

[9] P. A. Chou, T. Lookabaugh, and R. M. Gray. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Transactions on Information Theory*, 35(2):299–315, 1989.

[10] T. M. Cover and S. Pombra. Gaussian feedback capacity. *IEEE Transactions on Information Theory*, 35(1):37–43, 1989.

[11] N. Farvardin. A study of vector quantization for noisy channels. *IEEE Transactions on Information Theory*, 36(4):799–809, 1990.

[12] N. Farvardin and V. Vaishampayan. Optimal quantizer design for noisy channels: An approach to combined source - channel coding. *IEEE Transactions on Information Theory*, 33(6):827–838, 1987.

[13] N. Farvardin and V. Vaishampayan. On the performance and complexity of channel-optimized vector quantizers. *IEEE Transactions on Information Theory*, 37(1):155–160, 1991.

[14] P. E. Fleischer. Sufficient conditions for achieving minimum distortion in a quantizer. In *IEEE International Convention Record, Part I*, pages 104–111, 1964.

[15] A. Gersho and R. M. Gray. *Vector quantization and signal compression.* Kluwer Academic Publishers, 1992.

[16] S.-Z. Kiang, G.J. Sullivan, C.-Y. Chiu, and R. L. Baker. Recursive optimal pruning of tree-structured vector quantizers. In *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, pages 2285–2288 vol.4, 1991.

[17] A. Kurtenbach and P. Wintz. Quantizing for noisy channels. *IEEE Transactions on Communication Technology*, 17(2):291–302, 1969.

[18] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[19] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[20] S. M. Perlmutter and R. M. Gray. A low complexity multiresolution approach to image compression using pruned nested tree-structured vector quantization. In *Proceedings of 1st International Conference on Image Processing*, volume 1, pages 588–592 vol.1, 1994.

[21] N. Phamdo. *Quantization Over Discrete Noisy Channels Under Complexity Constraints.* PhD thesis, University of Maryland, 1993.

[22] N. Phamdo, F. Alajaji, and N. Farvardin. Quantization of memoryless and Gauss-Markov sources over binary Markov channels. *IEEE Transactions on Communications*, 45(6):668–675, 1997.

[23] N. Phamdo, N. Farvardin, and T. Moriya. A unified approach to tree-structured and multistage vector quantization for noisy channels. *IEEE Transactions on Information Theory*, 39(3):835–850, 1993.

[24] Z. Raza. Sample adaptive product quantization for memoryless noisy channels. Master's thesis, Queen's University, 2002.

[25] Z. Raza, F. Alajaji, and T. Linder. Design of sample adaptive product quantizers for noisy channels. *IEEE Transactions on Communications*, 53(4):576–580, 2005.

[26] S. Rezazadeh. Scalar quantization algorithms for the robust transmission of correlated sources over one- and two-way channels. Master's thesis, Queen's University, 2019.

[27] E. A. Riskin. Optimal bit allocation via the generalized BFOS algorithm. *IEEE Transactions on Information Theory*, 37(2):400–402, 1991.

[28] E. A. Riskin and R. M. Gray. A greedy tree growing algorithm for the design of variable rate vector quantizers (image compression). *IEEE Transactions on Signal Processing*, 39(11):2500–2507, 1991.

[29] S. Shahidi, F. Alajaji, and T. Linder. MAP decoding of quantized sources over soft-decision fading channels with memory. In *2012 IEEE International Conference on Communications (ICC)*, pages 2277–2282, 2012.

[30] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[31] C. E. Shannon. The zero error capacity of a noisy channel. *IRE Transactions on Information Theory*, 2(3):8–19, 1956.

[32] L. Song, F. Alajaji, and T. Linder. Capacity of burst noise-erasure channels with and without feedback and input cost. *IEEE Transactions on Information Theory*, 65(1):276–291, 2019.

[33] V. A. Vaishampayan and N. Farvardin. Optimal block cosine transform image coding for noisy channels. *IEEE Transactions on Communications*, 38(3):327–336, 1990.

[34] X. Yu, H. Wang, and E. Yang. Optimal quantization for noisy channels with random index assignment. In *2008 IEEE International Symposium on Information Theory*, pages 2727–2731, 2008.