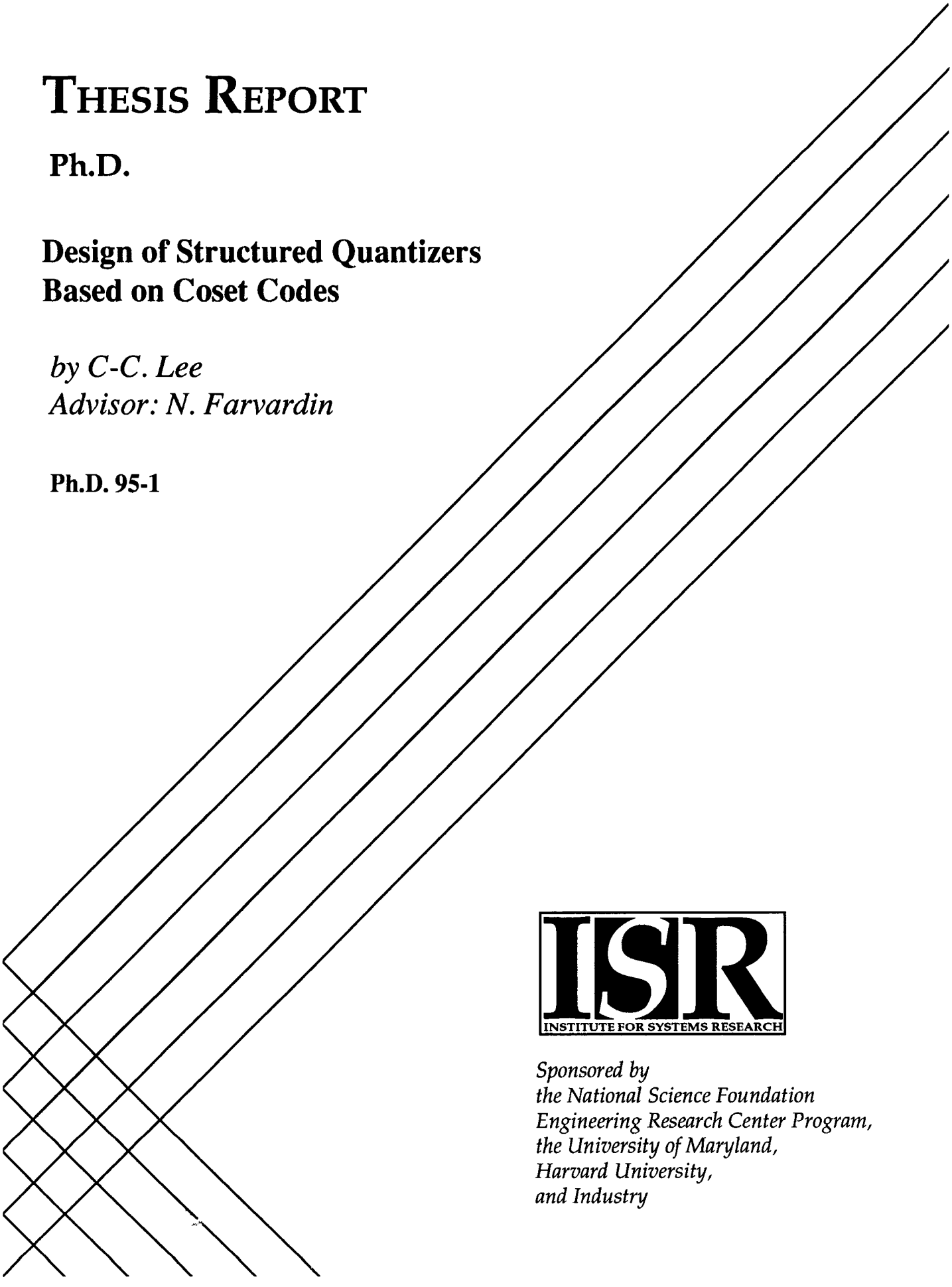
THESIS REPORT

Ph.D.

Design of Structured Quantizers Based on Coset Codes

by C-C. Lee
Advisor: N. Farvardin

Ph.D. 95-1



Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

ABSTRACT

Title of Dissertation: DESIGN OF STRUCTURED QUANTIZERS BASED ON COSET CODES

Cheng-Chieh Lee, Doctor of Philosophy, 1995

Dissertation directed by: Professor Nariman Farvardin
Department of Electrical Engineering

For memoryless sources, Entropy-Constrained Scalar Quantizers (ECSQs) can perform closely to the Gish-Pierce bound at high rates. There exist two fixed-rate variations of ECSQ – Scalar-Vector Quantizer (SVQ) and Adaptive Entropy-Coded Quantizer (AECQ) – that also perform closely to the Gish-Pierce bound. These quantization schemes have approximately cubic quantization cells while high-rate quantization theory suggests that quantization cells of the optimal quantizers should be approximately spherical. There are some coset codes whose Voronoi regions are very spherical. In this dissertation we present structured quantization schemes that combine these coset codes with the aformentioned quantizers (SVQ, ECSQ, and AECQ) so as to improve their performance beyond the Gish-Pierce bound.

By combining trellis codes (that achieve a significant granular gain) with SVQ, ECSQ, and AECQ, we obtain Trellis-Based Scalar-Vector Quantizer (TB-SVQ), Entropy-Constrained Trellis-Coded Quantizer (ECTCQ), and Pathwise-Adaptive ECTCQ (PA-ECTCQ), respectively. With an 8-state underlying trellis code, these trellis-coded quantization schemes perform about 1.0 dB better than their naive counterparts.

There are two approaches that can extend the quantizers (TB-SVQ, ECTCQ, and PA-ECTCQ) for quantizing sources with memory. The first is to combine the predictive coding operation of the Differential Pulse Code Modulation scheme with various quantizers, yielding Predictive TB-SVQ, Predictive ECTCQ, and Predictive PA-ECTCQ, respectively. There is a duality between quantizing sources with memory and transmitting data over channels with memory. Laroia, Tretter, and Farvardin have recently introduced a precoding idea that helps transmitting data efficiently over channels with memory. By exploiting this duality, the second approach combines the precoder with TB-SVQ and ECTCQ to arrive at Precoded TB-SVQ and Precoded ECTCQ, respectively. Simulation results indicate that the performance of these quantizers are also close to the rate-distortion limit.

The PA-ECTCQ performance has been shown to be robust in the presence of source scale and, to a lesser extent, shape mismatch conditions. We also considered adjusting the underlying entropy encoder based on the quantized output (which provide some approximate information on the source statistics). The performance of the resulting Shape-Adjusting PA-ECTCQ has been shown to be robust to a rather wide range of source shape mismatch conditions.

DESIGN OF STRUCTURED QUANTIZERS BASED ON COSET CODES

by

Cheng-Chieh Lee

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1995

Advisory Committee:

Professor Nariman Farvardin, Chairman/Advisor
Associate Professor Steven Tretter
Associate Professor Thomas Fuja
Assistant Professor KuoJuey Liu
Associate Professor William Gasarch

Dedication

To Jer-Hsiong Lee and Yiuh-Mei Chen

Acknowledgements

I would like to thank my advisor, Professor Nariman Farvardin, for his guidance, sustained encouragement and support. My time with him has been a rewarding learning experience. I thank Dr. Rajiv Laroia for his ideas at various stages of my study. With his guidance, I have also profited greatly while visiting AT&T Bell Laboratories under the URP summer internship program. I am grateful to the professors who have taught me in the graduate courses. I would like to also thank my friends in the Communications and Signal Processing Laboratory for their assistance and numerous interesting discussions. Finally, I thank my family and friends for their constant love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Quantization is the process of encoding to produce a digital stream (or description) for a signal from some discrete-time real-valued source and decoding from the digital stream to create an approximated replica of the signal. The compression efficiency of a quantizer is measured by its *rate*, which is the average number of bits per source sample produced by the encoder, and its average *distortion* induced by substituting the replica for the original signal. Other important parameters for assessing the performance of a quantizer include the coding delay and the complexity of implementing its encoder and decoder. In the literature, the terms *source coding with a fidelity criterion* [1]-[2], *digital coding* [3] and *lossy signal compression* [4] are all used to connote this operation of achieving a compact digital description for a signal.

An essential problem in quantizer design is to minimize the rate in the digital description of the signal while maintaining a required level of fidelity, coding delay, and implementation complexity. For stationary sources, the ultimate quantizer rate-distortion performance can be described by Shannon's theory [1]-[2],[5]-[6].

Unfortunately, this theory provides no constructive quantizer design methodology but yields a rate-distortion limit [6] that is only asymptotically (in delay and/or complexity) approachable. In the dissertation we pursue low-complexity quantizers that have an affordable coding delay and can provide a performance close to the rate-distortion limit.

There are two methods of mapping the reproduction sequence to and from its corresponding digital description. In the first method, each reproduction symbol belongs to a codebook of a specified resolution and can hence be associated with a codeword of a fixed number of digits. The corresponding quantization schemes are generally called *resolution-constrained.* Shannon's source coding theorem [1] indicates that the quantizer output entropy is the minimum average information required to faithfully represent the replica sequence (with a probability arbitrarily close to one). In the second method, assuming that there exists an ideal entropy code (such as the Huffman code [7] or the arithmetic code [8]-[9]), each reproduction symbol is indexed by a variable-length codeword from the entropy codebook. The corresponding quantizers are generally called *entropy-constrained* quantizers.

Quantizers can be broadly classified into two types — *scalar quantizers* and *delayed decision quantizers* [3]. A scalar quantizer operates on one source sample at a time. Delayed decision quantizers, on the contrary, take in more than one sample to determine the replica for each source sample. Delayed decision quantizers can be further divided into genuine *vector quantizers* [4] and multipath search coding schemes, such as tree and trellis coders [10]-[11]. A vector quantizer operates on a block of finite source samples at a time. By contrast, tree and trellis coders have a sliding-block coding nature and operate in the infinite-dimensional sequence space. Vector quantizers can generally offer more freedom than scalar quantizers in placing

the reproduction vectors in the multidimensional space. This freedom in turn leads to several specific vector quantization gains over scalar quantization [12]-[14].

The rest of this introductory chapter is organized as follows. In Section 1.1, we present a brief review of the high-rate quantization theory [12]-[14] in the context of resolution-constrained quantization. Previous work on the resolution-constrained quantizers that can achieve various vector quantization gains are also mentioned. Similar discussions for entropy-constrained quantization are provided in Section 1.2. Finally, the dissertation is outlined in Section 1.3.

1.1 Resolution-Constrained Quantizers

Resolution-constrained vector quantizers have more freedom than scalar quantizers in placing their reproduction vectors (called code-vectors) in the multidimensional space. This freedom in turn renders vector quantization three gains over uniform scalar quantization. The first is the *boundary* gain[1] [14] and is realized by selecting an appropriate codebook boundary which ensures that most of the code-vectors are placed in the high probability region $\mathcal{R}_{\text{AEP}}$ as dictated by the asymptotic equipartition property (AEP). The second type of gain is the *granular* gain[2] [14] that is achieved by having more spherical (for the squared-error distortion measure) quantization cells than the cubic cells of uniform scalar quantizers. Finally, there is the *non-uniform density* gain [15] which results from having the code-vectors closely spaced in higher probability density regions and farther apart in lower probability density regions. Several quantization schemes reported in the literature

[1]This is called the *shape* advantage in [13] for memoryless sources. The boundary gain includes the *memory* advantage in [13] for sources with memory.

[2]This is called the *space-filling* advantage in [13].

that can achieve parts or all of these gains are described next.

Optimal scalar quantizers introduced by Max [16] and Lloyd [17], known as Lloyd-Max quantizers (LMQs), minimize the average (squared-error) distortion for a given number of quantization levels. These quantizers can achieve only the non-uniform density gain. In spite of this gain, there still exists a big gap between the LMQ performance and the rate-distortion limit.

Locally optimal vector quantizers introduced by Linde, Buzo and Gray [18] (called LBG VQs) can perform arbitrarily close to the rate-distortion limit as the dimension N becomes large. However, LBG VQs are generally unstructured and their implementation complexity (computational and storage) is exponential in Nr (where r is the per sample bit rate). They are hence prohibitive even at modest rates and dimensions. Suboptimal tree-structured vector quantizers [19] reduced the computational complexity but at the cost of added storage complexity. On the other hand, multi-stage vector quantizers [19] are simple to implement for a large Nr but generally result in a significant performance degradation over optimal vector quantizers.

The *scalar-vector quantizer* (SVQ) of Laroia and Farvardin [20] is a specific kind of vector quantizer whose structure is derived from a scalar quantizer. It is shown in [20] that SVQ can (asymptotically in block-length N) achieve the optimal boundary gain for memoryless sources by placing the code-vectors on and inside the region $\mathcal{R}_{\text{AEP}}$. SVQ, however, realizes no granular gain as its quantization cells are approximately cubic. The *trellis coded quantizer* (TCQ) of Marcellin and Fischer [21] is also derived from a scalar quantizer and uses Ungerboeck's idea of coding by set partitioning [22] to realize a significant portion of the ultimate granular gain. This quantizer makes no explicit attempt to exploit the boundary

gain.

The *trellis-based scalar-vector quantizer* (TB-SVQ) [15] introduced by Laroia and Farvardin is a structured vector quantizer for stationary memoryless sources that combines the SVQ idea [20] with that of TCQ [21]. The SVQ structure allows TB-SVQ to achieve a large boundary gain while the underlying trellis code enables it to realize a significant granular gain. Assuming that the block-length N is sufficiently large so that there is no non-uniform density gain to be realized, TB-SVQ is capable of realizing both the boundary and granular gains and has been shown to outperform all (implementable) fixed-rate block-based quantization schemes reported in the literature.

1.2 Entropy-Constrained Quantizers

There are two gains that *entropy-constrained vector quantizers* (ECVQs) can realize over *entropy-constrained scalar quantizers* (ECSQs). The first, as in the resolution-constrained case, is the granular gain and is achieved by having more spherical (for the squared-error distortion measure) quantization cells than the cubic cells of scalar quantizers. The second gain is the *memory* gain [12]-[13] that is available only for sources with memory. To capitalize on the memory gain, ECVQ exploits the freedom of placing its code-vectors in the multidimensional space (depending on the source probability density function) as opposed to the Cartesian product of a scalar codebook as in the ECSQ.

For a large class of memoryless sources [23], the performance gap between the optimal ECSQ and the rate-distortion limit at high rates has been shown to be 1.53 dB (for the squared-error distortion measure). This gap is exactly the

ultimate granular gain that the scalar quantizers fail to realize. The boundary gain in the context of resolution-constrained quantization hence does not have its counterpart in the context of entropy-constrained quantization [13]. It has been shown [12, 23] that at high rates the optimal entropy-constrained quantizers are very nearly uniform. At low rates, however, there can be some non-uniform density gain over uniform quantization, as in the resolution-constrained case.

Necessary conditions for optimality of ECSQs and their design algorithms have been derived in [24]-[26]. In [27], Chou *et al.* extended these results for ECVQs. ECVQ can realize both the granular and memory gains and can, in principle, approach the rate-distortion limit. Like LBG VQ, however, ECVQ is generally unstructured and the complexity can be prohibitive even at modest rates and dimensions.

For memoryless sources, the entropy-constrained counterpart for the structured resolution-constrianed TB-SVQ is the *entropy-constrained trellis-coded quantizer* (ECTCQ) [28]. For sources with memory, ECTCQ can be incorporated into a predictive coding structure, giving rise to the so-called *predictive* ECTCQ [28]. These two entropy-constrained quantizers can achieve a significant portion of the available vector quantization (granular and memory) gains and perform very close to the rate-distortion limit.

1.3 Outline of the Dissertation

In the dissertation we treat the TB-SVQ of [15] and ECTCQ of [28] as the baseline resolution-constrained and entropy-constrained schemes, respectively, and thereby derive their possible variations and/or extensions. Generally speaking, the purpose

of the variations is to reduce the implementation complexity while their extensions capitalize on the memory gain for sources with memory.

We can look at the quantizers that will be presented in the dissertation for quantizing memoryless sources from another point of view. The optiomal ECSQ performance (within about 1.53 dB of the rate-distortion limit) is sometimes called the Gish-Pierce bound [23]. There are several variations [20, 46, 48] of ECSQ that can also perform close to the Gish-Pierce bound. These quantizers have approximately cubic quantization cells and hence fail to capitalize on the granular gain. There are some coset codes [29]-[30] whose Voronoi regions are very spherical. The quantizers that we develop for quantizing memoryless sources are obtained by combining these coset codes with ECSQ or its variations. These quantizers use the underlying coset codes to realize the granular gain and hence perform better than the Gish-Pierce bound.

The rest of the dissertation is organized as follows.

In Chapter 2 we describe the models for sources that will be considered in the dissertation, present some results from the high-rate quantization theory [12]-[14] that are related to the granular gain, and provide background materials for coset codes [29]-[30].

The TB-SVQ [15] can provide a rather competitive rate-distortion performance for memoryless sources while maintaining a fixed encoding rate. In Chapter 3 we pursue possible extensions or variations of the TB-SVQ. Specifically, these will include a simpler suboptimal codebook search algorithm for TB-SVQ and an extension of TB-SVQ for quantizing sources with memory.

The duality between quantization and transmission has been studied in [14]. In Chapter 4, we exploit this duality and present a novel quantization scheme for

quantizing sources with memory. This new quantizer is motivated by the precoding idea of Laroia *et al.* [31] that helps transmitting data efficiently over channels with memory.

In Chapter 5, we present a novel ECTCQ that has a symmetric reproduction alphabet. The added symmetry constraint is exploited to provide relatively simple implementation vis-à-vis the previous ECTCQs reported in the literature [28, 32]. Gersho's result [12] indicates that at high rates the optimal ECVQ should be very nearly uniform. Hence, we will also consider designing entropy-constrained quantizers directly based on some "good" coset codes.

Entropy-coded quantizers generally produce variable-length codewords while in some situations the communication channel operates at a fixed rate. Transmission of variable-length codewords over a fixed-rate channel necessitates the use of a buffer of finite, and preferably small, size. The buffer overflow/underflow event should be avoided as it could otherwise lead to loss of codeword synchronization (and hence a large quantization distortion). In Chapter 6, we consider buffer management strategies for the entropy-coded quantizers of Chapter 5. Adaptive modifications will also be considered in the presence of source mismatch conditions.

In Chapter 7, we provide a summary of the dissertation. During the course of this work, the quantizers developed have found extensive application in speech coding [33]-[34] and image coding [35]-[37]. While these topics are not covered in the dissertation, they are also briefly described.

Chapter 2

Preliminaries

This chapter provides preliminary materials such as a description of models for the sources we consider, a generic quantization configuration, and a description of coset codes. Notations and conventions that will be used throughout the dissertation will also be presented.

2.1 Input Sources

We will consider both memoryless sources and sources with memory. The source is a discrete-time real-valued stationary source denoted by $\{X_i : i \in \mathbf{Z}\}$ where X_i is a real random variable and $\mathbf{Z}$ is the integer set. Given a realization $\{x_i\}$ of such a source, an equivalent N-dimensional (N-D) vector sequence $\{\boldsymbol{x}_n : n \in \mathbf{Z}\}$ can be formed by blocking every contiguous N samples into a vector; that is, $\boldsymbol{x}_n \equiv (x_{nN}, x_{nN+1}, \cdots, x_{nN+N-1}) \in \mathbb{R}^N$.

For memoryless sources, we will specifically confine our attention only to a class (called *generalized Gaussian*) for which the marginal probability density function

(p.d.f.) is given by

$$p(x;\theta,\sigma) = \frac{\theta\eta(\theta,\sigma)}{2\Gamma(1/\theta)} \exp\{-[\eta(\theta,\sigma)|x|]^{\theta}\}, \tag{2.1}$$

where

$$\eta(\theta,\sigma) = \sigma^{-1}\left[\frac{\Gamma(3/\theta)}{\Gamma(1/\theta)}\right]^{1/2}, \tag{2.2}$$

the *shape parameter* $\theta > 0$ describes the exponential rate of decay, σ^2 is the source variance, and $\Gamma(\cdot)$ is the Gamma function. For $\theta = 2.0$ we have the Gaussian distribution while for $\theta = 1.0$ we obtain the Laplacian distribution.

For sources with memory, we confine our attention only to the cases in which the p.d.f. of the current source sample X_i can be completely specified based on the knowledge of a finite number, say p, of the most recent past source samples $X_{i-p}, \cdots, X_{i-2}, X_{i-1}$ — a property owned by Markov processes. For simplicity, these sources are called *Markov* sources. More specifically, we will consider sources which can be represented by the output of a stable p^{th} order autoregressive linear filter $H(z)$ driven by a memoryless generalized Gaussian innovations process $\{W_i\}$ where

$$H(z) = \frac{1}{1 - \sum_{j=1}^{p} \rho_j z^{-j}}, \tag{2.3}$$

the coefficient $\rho_j \in \mathbb{R}$, and the marginal p.d.f. of $\{W_i\}$ is as described in (2.1).

2.2 A Generic Quantization Configuration

The block diagram of a generic quantizer is shown in Figure 2.1. The *encoder* ψ maps the source N-vector $\boldsymbol{x}_n$ into a digital codeword c_n. Based on the received c_n after the communication channel (assuming no transmission error), the *decoder*

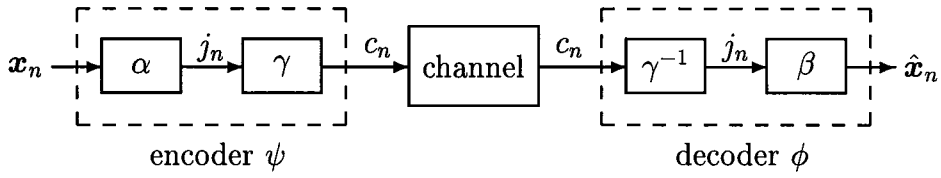


Figure 2.1: A generic quantization configuration.

ϕ then builds a replica $\hat{\boldsymbol{x}}_n$ for $\boldsymbol{x}_n$. The encoder mapping can be decomposed as $\gamma \circ \alpha$, where α maps $\boldsymbol{x}_n$ to an index $j_n \in \boldsymbol{\mathcal{J}}$ (the set of all indices), γ maps j_n into the channel codeword c_n, and $\circ$ denotes the composition operation. Similarly, the decoder mapping can be described as $\beta \circ \gamma^{-1}$ where γ^{-1} reconstructs the indices and β produces $\hat{\boldsymbol{x}}_n$ for the decoded index j_n.

The quantizer performance is evaluated by (i) the average *distortion*

$$D_N = \frac{1}{N} E[\rho(\boldsymbol{X}, \hat{\boldsymbol{X}})], \tag{2.4}$$

and (ii) the average *rate*

$$R_N = \frac{1}{N} E[l(\psi(\boldsymbol{X}))], \tag{2.5}$$

per source symbol where $\rho(\cdot, \cdot)$ and $l(\cdot)$ denote the nonnegative distortion measure and codeword length (in bits), respectively. In the sequel, we will only consider the squared-error distortion measure and assume that the quantization indices are encoded into binary codewords. For a rate r bits/sample resolution-constrained quantizer, each code-vector is represented by a codeword of Nr bits and hence $R_N = r$ bits/sample. Fixing the average rate, the design problem is to minimize D_N. For a rate r bits/sample entropy-constrained quantizer, assuming that there exists an ideal entropy encoder/decoder pair (γ, γ^{-1}), the design problem is to

determine the pair (α, β) that minimizes D_N subject to a constraint that the per sample output entropy of α is no larger than r bits/sample.

2.3 Granular Gain

Zador [38] showed that for asymptotically high-rate quantization (both for the resolution- and entropy-constrained cases) the minimum average (per dimensional) distortion D_N is proportional to the *quantization coefficient* C_N which depends only on the dimension N. Zador did not obtain C_N explicitly, but he showed that

$$\frac{1}{2+N} V_N^{-2/N} \leq C_N \leq \frac{1}{N} \Gamma(1+2/N) V_N^{-2/N}, \tag{2.6}$$

where V_N is the volume of a unit sphere in $\mathbb{R}^N$.

A geometric interpretation of the coefficients C_N was made by Gersho [12], which we briefly describe as follows. An N-D convex polytope $\mathbb{P}$ is said to generate a *tessellation* if there exists a partition of $\mathbb{R}^N$ whose regions are all congruent to $\mathbb{P}$. The *normalized second moment* of $\mathbb{P}$ is defined as

$$G(\mathbb{P}) = \frac{1}{[\mathrm{Vol}(\mathbb{P})]^{1+2/N}} \cdot \int_{\mathbb{P}} \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2 d\boldsymbol{x}, \tag{2.7}$$

where $\hat{\boldsymbol{x}}$ denotes the center of $\mathbb{P}$ and $\mathrm{Vol}(\mathbb{P})$ is the volume of $\mathbb{P}$. A class of admissible polytopes in $\mathbb{R}^N$, denoted by $\boldsymbol{P}_N$, is the set of all convex polytopes that generate tessellations of $\mathbb{R}^N$. Gersho conjectured that the optimal N-D polytope $\mathbb{P}_N^*$, defined as

$$\mathbb{P}_N^* = \arg\min_{\mathbb{P} \in \boldsymbol{P}_N} G(\mathbb{P}), \tag{2.8}$$

exists for each N and C_N is obtained as the normalized second moment of the optimal polytope $\mathbb{P}_N^*$; that is,

$$C_N = G(\mathbb{P}_N^*). \tag{2.9}$$

In Gersho's conjecture, he implicitly assumes that the quantization cells of optimal quantizers should be close (or identical) to the optimal polytope (which is the most spherical one within the class of admissible polytopes). It is believed that C_N decreases in N, implying that in higher dimensional spaces the optimal polytopes are more spherical. An optimal N-D vector quantizer can hence realize over the optimal scalar quantizer a portion of the granular gain [14] denoted by

$$\gamma_g(N) = \frac{C_1}{C_N}, \tag{2.10}$$

where $C_1 = 1/12$. It can be shown that both the upper and lower bounds on C_N given in (2.6), and hence C_N itself, converge to $1/2\pi e$ as $N \to \infty$ [39]. The ultimate granular gain $\gamma_g(\infty)$ is therefore $\pi e/6$ (1.53 dB).

The optimal convex polytope $\mathbb{P}_N^*$ is, however, unknown for most N. On the other hand, *lattices* [39] constitute an important subclass of tessellations of $\mathbb{R}^N$. There exist lattices and their extensions to the sequence space — *trellis codes* [22], collectively known as *coset codes* [29]-[30], whose constituent convex polytopes are very close to optimal. In addition, the inherent algebraic structures in coset codes often lead to fast search/encoding algorithms. We next describe some properties of coset codes that are relevant to quantizers to be presented in later chapters.

2.4 Coset Codes

Algebraically, a real *lattice* $\boldsymbol{\Lambda}$ is a discrete set of vectors in $\mathbb{R}^N$ that forms a group under ordinary vector addition operation. Geometrically, a lattice $\boldsymbol{\Lambda}$ is an infinite regular array that covers $\mathbb{R}^N$ uniformly. The *Voronoi region* of $\boldsymbol{\Lambda}$, denoted by $\mathbf{R}_V(\boldsymbol{\Lambda})$, is defined as the set of all points in $\mathbb{R}^N$ that are at least as close to the

origin as to any other point $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$; i.e.,

$$\mathbf{R}_V(\boldsymbol{\Lambda}) \triangleq \{\boldsymbol{x} : ||\boldsymbol{x}||^2 = \min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} ||\boldsymbol{x} - \boldsymbol{\lambda}||^2\}. \tag{2.11}$$

The normalized second moment of a lattice is defined as the normalized second moment of its Voronoi region; i.e.,

$$G(\boldsymbol{\Lambda}) = G(\mathbf{R}_V(\boldsymbol{\Lambda})). \tag{2.12}$$

A *sublattice* $\boldsymbol{\Lambda}'$ of $\boldsymbol{\Lambda}$ is a subset of $\boldsymbol{\Lambda}$ which is itself an N-D lattice. The sublattice induces a *partition* $\boldsymbol{\Lambda}/\boldsymbol{\Lambda}'$ of $\boldsymbol{\Lambda}$ into $|\boldsymbol{\Lambda}/\boldsymbol{\Lambda}'|$ *cosets* of $\boldsymbol{\Lambda}'$, where $|\boldsymbol{\Lambda}/\boldsymbol{\Lambda}'|$ is the *order* of the partition. If we take one element from each coset of $\boldsymbol{\Lambda}'$, we obtain a system of *coset representatives* for the partition $\boldsymbol{\Lambda}/\boldsymbol{\Lambda}'$, denoted by $[\boldsymbol{\Lambda}/\boldsymbol{\Lambda}']$. Then every element $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ can be written uniquely as a sum $\boldsymbol{\lambda} = \boldsymbol{\lambda}' + \boldsymbol{c}$ where $\boldsymbol{c} \in [\boldsymbol{\Lambda}/\boldsymbol{\Lambda}']$ is the coset representative for the coset in which $\boldsymbol{\lambda}$ lies, and $\boldsymbol{\lambda}' = \boldsymbol{\lambda} - \boldsymbol{c}$ is an element of $\boldsymbol{\Lambda}'$. This is called a *coset decomposition* of $\boldsymbol{\Lambda}$ and will be written as

$$\boldsymbol{\Lambda} = \boldsymbol{\Lambda}' \oplus [\boldsymbol{\Lambda}/\boldsymbol{\Lambda}']. \tag{2.13}$$

An integer N-D lattice $\boldsymbol{\Lambda}$ is called a *binary lattice* if $\mathbf{Z}^N/\boldsymbol{\Lambda}/2^\mu\mathbf{Z}^N$ for some integer μ. The least such μ is called the *2-depth* of the lattice. A binary lattice with 2-depth μ is usually called a *mod-2^μ* lattice. Corresponding to a mod-2^μ lattice $\boldsymbol{\Lambda}$, there exists a linear (N, k) block code $\mathcal{C}$ over the 2^μ-ary Galois field such that $\boldsymbol{\Lambda}$ consists of all integer N-tuples that are congruent modulo 2^μ to one of the codewords $\boldsymbol{c} \in \mathcal{C}$ [29]. This implies that, for example, a mod-2 lattice $\boldsymbol{\Lambda}$ can be written as

$$\boldsymbol{\Lambda} = \bigcup_{i=0}^{2^k-1} \boldsymbol{c}(i) + 2\mathbf{Z}^N, \tag{2.14}$$

where $\mathcal{C} \equiv \{\boldsymbol{c}(0), \boldsymbol{c}(1), \cdots, \boldsymbol{c}(2^k - 1)\}$ is the associated binary block code. More generally, the decomposition of $\boldsymbol{\Lambda}$ into cosets of $2^\mu \mathbf{Z}^N$ can be written as

$$\boldsymbol{\Lambda} = 2^\mu \mathbf{Z}^N \oplus [\boldsymbol{\Lambda}/2^\mu \mathbf{Z}^N] \equiv 2^\mu \mathbf{Z}^N \oplus \mathcal{C}, \tag{2.15}$$

where every N-tuples in $\mathcal{C}$ is considered as an integer vector in $\mathbb{R}^N$.

Trellis codes are generalizations of finite-dimensional lattices to the sequence space just as convolutional codes are generalizations of block codes. As for the simplest example, consider an N-D lattice $\boldsymbol{\Lambda}$ which can be decomposed into 2^{k+r} cosets of sublattice $\boldsymbol{\Lambda}'$, each indexed by a coset representative $\boldsymbol{\lambda}(i)$ for $i \in \mathcal{J}_{2^{k+r}}$[1]. Denote by Ψ the space of all admissible output sequences (over $\mathcal{J}_{2^{k+r}}$) of a rate-$k/(k+r)$ binary convolutional encoder. The associated N-D trellis code is given by

$$\boldsymbol{\Lambda}' \oplus \boldsymbol{\lambda}(\Psi) \equiv \left\{ \{\boldsymbol{\lambda}'_n + \boldsymbol{\lambda}(\boldsymbol{c}_n)\} : \boldsymbol{\lambda}'_n \in \boldsymbol{\Lambda}', \ \forall n; \ \ \{\boldsymbol{c}_n\} \in \Psi \right\}. \tag{2.16}$$

Both lattices and trellis codes can be put into the common framework of coset codes introduced by Forney [29]-[30]. For simplicity of presentation, we will confine our attention only to binary cases. The three main elements of a coset code, the general structure of whose encoder is shown in Figure 2.2, are as follows:

1) An N-D lattice $\boldsymbol{\Lambda}$.
2) A sublattice $\boldsymbol{\Lambda}'$ of $\boldsymbol{\Lambda}$ with $|\boldsymbol{\Lambda}/\boldsymbol{\Lambda}'| = 2^{(k+r)}$.
3) A rate-$k/(k+r)$ binary encoder $\mathcal{C}$, which takes in k-bit input codeword and puts out $(k+r)$-bit codeword.

The associated coset code, denoted by $\mathbb{C}(\boldsymbol{\Lambda}/\boldsymbol{\Lambda}'; \mathcal{C})$, is the set of all sequences of signal points that lie within a sequence of cosets of $\boldsymbol{\Lambda}'$ which could be specified by a $\mathcal{C}$-admissible sequence of $(k+r)$-bit codewords.

[1] $\mathcal{J}_K \triangleq \{0, 1, 2, \cdots, K-1\}$.

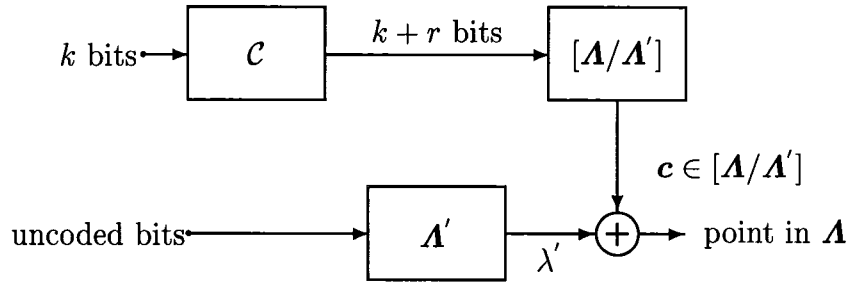


Figure 2.2: Encoder structure of the coset code.

2.4.1 Application of Coset Codes to Quantization

According to the discussion (on Gersho's conjecture [12]) provided in Section 2.3, quantizers whose codebooks are formed based on code-sequences of a coset code $\mathbb{C}(\Lambda/\Lambda';\mathcal{C})$ can realize a portion of granular gain given by

$$\gamma(\mathbb{C}(\Lambda/\Lambda';\mathcal{C})) = \frac{C_1}{G(\mathbb{C}(\Lambda/\Lambda';\mathcal{C}))}, \tag{2.17}$$

where $G(\mathbb{C}(\Lambda/\Lambda';\mathcal{C}))$ is the normalized second moment of $\mathbb{C}(\Lambda/\Lambda';\mathcal{C})$.

When $\mathcal{C}$ is a block code, $\mathbb{C}(\Lambda/\Lambda';\mathcal{C})$ reduces to the lattice Λ. In this case, $N = k + r$ and $\mathcal{C}$ is an (N, k) block code. The normalized second moments $G(\Lambda)$ for some binary lattices have been computed in the literature (see [39]); we provide in Table 2.1 the corresponding granular gains of quantizers constructed based on these lattices.

The normalized second moment $G(\mathbb{C}(\Lambda/\Lambda';\mathcal{C}))$ of Ungerboeck's famous 1-D trellis codes [22] (called Ungerboeck trellis codes in the sequel) have been obtained by simulations in [21]. In Table 2.2, we provide the corresponding granular gains of quantizers constructed based on these trellis codes.

2.4.2 Symmetry in Ungerboeck (1-D) Trellis Codes

Here, we describe some symmetry properties associated with Ungerboeck trellis codes [22]. These properties will be exploited to reduce the implementation (mainly storage) complexity when we present the structured quantization schemes in later chapters.

Let us consider an Ungerboeck trellis code [22] which is based on some rate-1/2 convolutional code and an underlying partition of $\mathbf{\Lambda} = 2\mathbf{Z} + 1$ into four cosets of $8\mathbf{Z}$, namely $\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{B}_0$, and $\mathcal{B}_1$. The convolutional code can be implemented based on some specific shift register (see [22]), the content of whose delay elements (memory) corresponds uniquely to the so-called *trellis state*. We will denote by ν and $\mathbf{\Sigma} = \{0, 1, \cdots, 2^\nu - 1\}$ the number of delay elements of the shift register and the space of all possible trellis states, respectively. Let each coset be indexed exclusively by one of the four possible two-bit output word of the shift register. Given any current trellis state, the next state is uniquely determined by the one-bit input word to the shift register and the resulting state transition is labeled by the two-bit output word of the shift register, or equivalently, one of the four cosets — $\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{B}_0$, and $\mathcal{B}_1$. Within one time frame, the set of all possible transitions describes the *trellis diagram*. As an example, the trellis diagram of a 4-state Ungerboeck trellis code along with the four partitions of $\mathbf{\Lambda}$ are shown in Figure 2.3.

Let us define two supersets $\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_1$ and $\mathcal{B} = \mathcal{B}_0 \cup \mathcal{B}_1$ and denote by $\mathbf{\Lambda}^+$ the nonnegative subset of $\mathbf{\Lambda}$. There are three properties associated with Ungerboeck trellis codes (along with the underlying lattice translate $\mathbf{\Lambda}$ we assume):

1. Each point in $\mathcal{A}$ has a dual point of the same magnitude in $\mathcal{B}$.
2. For any trellis state $s \in \mathbf{\Sigma}$, the allowed superset is either $\mathcal{A}$ or $\mathcal{B}$, but not in

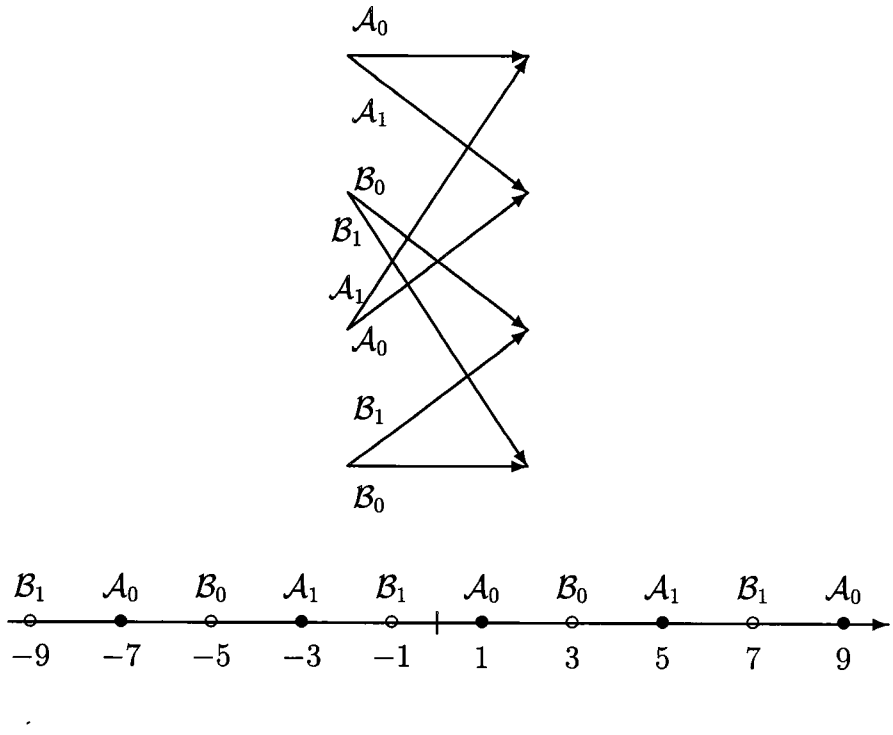


Figure 2.3: Ungerboeck's 4-state 1-D trellis code.

any other twisted form of the four cosets.

3. For any $s \in \boldsymbol{\Sigma}$, the previous allowed superset that leads to a transition to s is either $\mathcal{A}$ or $\mathcal{B}$, but not in any other twisted form of the four cosets.

Since the allowed superset for a given trellis state $s_i \in \boldsymbol{\Sigma}$ at time i is either $\mathcal{A}$ or $\mathcal{B}$ (by Property 2) and there is at most one point of a specified magnitude in that superset (by Property 1), there exists a state-dependent function $\mathcal{F} : \boldsymbol{\Sigma} \times \boldsymbol{\Lambda}^+ \rightarrow \boldsymbol{\Lambda}$ that is capable of recovering a point z_i that belongs either to $\mathcal{A}$ or $\mathcal{B}$ only by its magnitude $|z_i| \in \boldsymbol{\Lambda}^+$; that is,

$$z_i = \mathcal{F}(s_i, |z_i|). \tag{2.18}$$

The state transition that originates from s_i to the next state s_{i+1} is labeled by the coset to which z_i belongs. There hence exists another state-dependent function,

called the *next-state* function, $\mathcal{N} : \boldsymbol{\Sigma} \times \boldsymbol{\Lambda}^+ \rightarrow \boldsymbol{\Sigma}$, that can determine the next trellis state s_{i+1} based on the knowledge of s_i and $|z_i|$; that is,

$$s_{i+1} = \mathcal{N}(s_i, |z_i|). \tag{2.19}$$

Similarly, according to Properties 1 and 3, there exists the so-called *previous-state* function $\mathcal{P} : \boldsymbol{\Sigma} \times \boldsymbol{\Lambda}^+ \rightarrow \boldsymbol{\Sigma}$ that can determine s_i based on the knowledge of s_{i+1} and $|z_i|$; that is,

$$s_i = \mathcal{P}(s_{i+1}, |z_i|). \tag{2.20}$$

Now we can claim that a trellis code-sequence $\{z_i\}$ can be equivalently represented by its corresponding sequence of magnitudes $\{|z_i|\}$. This is because that, given the initial trellis state, the decoder can recursively apply (2.18) and (2.19) to reconstruct $\{z_i\}$ from the received magnitude sequence $\{|z_i|\}$.

N	Λ	Λ'	C	$\gamma(\Lambda)$ (dB)	2-depth
1	$\mathbf{Z}$	$\mathbf{Z}$	(1,0,∞)	0	
4	D_4	$2\mathbf{Z}^4$	(4,3,2)	0.37	1
8	D_8	$2\mathbf{Z}^8$	(8,7,2)	0.47	1
8	E_8	$2\mathbf{Z}^8$	(8,4,4)	0.65	1
16	Λ_{16}	$4\mathbf{Z}^{16}$	2(16,15,2)+(16,5,8)	0.86	2
24	Λ_{24}	$4\mathbf{Z}^{24}$	2(24,18,4)+(24,6,16)$'$	1.03	2

Table 2.1: Granular gains of some known binary lattices.

N	Λ	Λ'	state	$\gamma(\mathbb{C}(\Lambda/\Lambda';C))$ (dB)
			4	0.99
			8	1.10
			16	1.15
1	$\mathbf{Z}$	$4\mathbf{Z}$	32	1.23
			64	1.28
			128	1.33
			256	1.36

Table 2.2: Granular gains of Ungerboeck's 1-D rate-1/2 trellis codes.

Chapter 3

On Variations of the TB-SVQ

3.1 Introduction and Outline

In the context of resolution-constrained quantization, vector quantizers can realize over uniform scalar quantizers the granular, boundary, and non-uniform density gains (see Section 1.1). For memoryless uniform sources, there are quantizers based on lattices [39, 41] and trellis codes [21] which can realize the granular gain (to the extent indicated in Tables 2.1 and 2.2). To realize both the boundary and granular gains for memoryless non-uniform sources whose optimal codebook boundary $\mathcal{R}_{\text{AEP}}$ can be easily determined (as dictated by the AEP) and characterized, the basic idea is to optimally shape (according to $\mathcal{R}_{\text{AEP}}$) an unbounded codebook corresponding to some "good" lattice or trellis code. One generally need not be too concerned with the non-uniform density gain as it becomes diminishingly small in optimally shaped codebooks for a large vector dimension N.

The *scalar-vector quantizer* (SVQ) of Laroia and Farvardin [20] is a specific kind of fixed-rate vector quantizer whose codebook structure is derived from a

variable-length scalar quantizer. It has been shown in [20] that for memoryless sources SVQ can asymptotically (in N) achieve the ECSQ performance, which at high rates is inferior to the rate-distortion limit only by the ultimate granular gain of 1.53 dB. It should be mentioned that no granular gain is realized by SVQ as its quantization cells are approximately cubic. This implies that SVQ can asymptotically (in N) achieve the ultimate boundary gain by placing all code-vectors inside $\mathcal{R}_{\text{AEP}}$. The *trellis-based scalar-vector quantizer* (TB-SVQ) recently introduced by Laroia and Farvardin [15] is another structured fixed-rate quantizer that combines SVQ [20] with the *trellis-coded quantizer* (TCQ) of Marcellin and Fischer [21]. For memoryless sources, the SVQ structure allows TB-SVQ to achieve a large boundary gain while the underlying trellis code enables it to realize a significant granular gain. In addition, TB-SVQ could be derived from a non-uniform scalar quantizer, which results in some non-uniform density gain. TB-SVQ is hence capable of realizing all the three vector quantization gains (see Section 1.1) and can, in principle, achieve the rate-distortion bound for large block-lengths and powerful (possibly complex) trellis codes. To our knowledge, TB-SVQ has been shown to outperform all other implementable resolution-constrained quantizers reported in the literature.

Two TB-SVQ codebook design and search methods are described in [15]. In the first method, the underlying scalar quantizer is generally non-uniform and is iteratively designed based on a Lagrangian formulation similar to that for SVQ [20] or ECSQ [26]. This method results in what will be called TB-SVQ-I in the sequel. The TB-SVQ-I codebook search algorithm is based on dynamic programming and is a combination of the SVQ codebook search algorithm [20] and the Viterbi trellis search algorithm of TCQ [21]. The second approach is AEP-motivated and defines

the codebook as the collection of all uniformly spaced code-sequences that are contained inside $\mathcal{R}_{\text{AEP}}$, resulting in what will be called TB-SVQ-II in the sequel. The design problem for TB-SVQ-II is simply to select a scaling factor for the underlying trellis code that yields the desired number of code-vectors contained inside $\mathcal{R}_{\text{AEP}}$. For the codebook search, the source vector is first quantized to the nearest trellis code-sequence just as in a TCQ. If the quantized vector lies inside $\mathcal{R}_{\text{AEP}}$, one proceeds to encode the next source vector. Otherwise, the source ovector is said to be *overloaded.* In case of such an overload event, one gradually moves the source vector slightly towards $\mathcal{R}_{\text{AEP}}$ and quantizes the shifted source vector to the nearest trellis code-sequence until the quantized vector lies inside $\mathcal{R}_{\text{AEP}}$.

The TB-SVQ-I codebook search complexity is approximately that of SVQ times the number of trellis states. For high encoding rates or complex trellis codes (with a large number of states), the search complexity and storage requirement can be prohibitive. TB-SVQ-I, however, is quite implementable at low rates and when the underlying trellis code is not too complex. Also, a fixed number of operations is needed for encoding each input vector. The search complexity of TB-SVQ-II is roughly the same as that of TCQ and depends linearly on the number of trellis states, if overload events occur only negligibly. Our simulation on TB-SVQ-II at low rates, however, indicates that overload events occur quite significantly and the number of operations required to produce the quantized output varies largely depending on the input source vector.

In this chapter we describe, for memoryless sources, a hybrid of TB-SVQ-I and TB-SVQ-II. The codebook structure of such a hybrid TB-SVQ is the same as that of TB-SVQ-II, therefore rendering the design problem as simple as that of selecting a scaling factor for the underlying trellis code. This design method

is generally simpler than the Lagrangian formulation for TB-SVQ-I. On the other hand, the codebook search algorithm of the hybrid TB-SVQ is similar to that of TB-SVQ-I. At low rates, the computational complexity and storage requirement of such a dynamic programming based algorithm are quite affordable. In addition, like TB-SVQ-I, a fixed number of operations is needed for encoding each input vector.

In this chapter we also consider applying the TB-SVQ idea to encode sources with memory. A popular idea for quantizing Markov sources is one that encodes the so-called *prediction residual* — the difference between the input source sample and its predicted version based on quantized samples in the past. The class of source coders based on this idea, collectively referred to as the *predictive quantizers*, includes the well-known *differential pulse code modulation* (DPCM) scheme [3]. Specifically, we will consider a combination of both TB-SVQ and DPCM — called the *predictive* TB-SVQ — for quantizing Markov sources.

The rest of this chapter is organized as follows. The basic idea of SVQ [20] is reviewed in Section 3.2. We then describe in Section 3.3 how SVQ is combined with Ungerboeck trellis codes (for which some useful symmetry properties are discussed in Section 2.4.2), yielding the TB-SVQ scheme which can realize both the boundary and granular gains for memoryless sources. In Section 3.4, we present the dynamic programming based TB-SVQ codebook search algorithms and analyze their complexity. In Section 3.5, we present the predictive TB-SVQ scheme for quantizing Markov sources. Simulation results will be provided in Section 3.6, followed by a summary of this chapter given in Section 3.7.

3.2 Scalar-Vector Quantization

Consider an m-level scalar quantizer $\boldsymbol{\mathcal{S}}$ described in terms of a set of reproduction levels $\boldsymbol{\mathcal{Q}} = \{q_j : j \in \boldsymbol{\mathcal{J}}_m\}$ and a corresponding set of nonnegative integer lengths $\boldsymbol{\mathcal{L}} = \{\ell_j : j \in \boldsymbol{\mathcal{J}}_m\}$, where ℓ_j is the length associated with q_j. If each component of an input N-vector is separately quantized using $\boldsymbol{\mathcal{S}}$, the quantized vector will be in an N-D grid of m^N points — $\boldsymbol{\mathcal{Q}}^N$. The codebook $\boldsymbol{\mathcal{Z}}$ of a rate r bits/sample SVQ $\boldsymbol{\mathcal{V}}$ [20] derived from $\boldsymbol{\mathcal{S}} \equiv (\boldsymbol{\mathcal{Q}}, \boldsymbol{\mathcal{L}})$ is a subset of $\boldsymbol{\mathcal{Q}}^N$, given as

$$\boldsymbol{\mathcal{Z}} = \{\boldsymbol{z} \equiv (z_1, z_2, \cdots, z_N) \in \boldsymbol{\mathcal{Q}}^N : \sum_{i=1}^{N} \ell_{f(z_i)} \leq L\}, \tag{3.1}$$

where the index function $f(\cdot) : \boldsymbol{\mathcal{Q}} \rightarrow \boldsymbol{\mathcal{J}}_m$ is defined as

$$f(q_j) = j, \quad j \in \boldsymbol{\mathcal{J}}_m, \tag{3.2}$$

and the threshold L is selected as the largest integer such that $\text{card}(\boldsymbol{\mathcal{Z}}) \leq 2^{Nr}$. The actual codebook cardinality $\text{card}(\boldsymbol{\mathcal{Z}})$ is generally not equal to but less than the desired number of code-vectors — 2^{Nr}. Such a discrepancy can be quantified by the so-called SVQ coding redundancy κ (in bits/sample) where

$$2^{N\kappa} = 2^{Nr} - \text{card}(\boldsymbol{\mathcal{Z}}). \tag{3.3}$$

An N-sphere or N-pyramid bounded cubic lattice based quantizer is a special case of SVQ in which $\boldsymbol{\mathcal{S}}$ is an unbounded uniform scalar quantizer and the length ℓ_j associated with the level q_j is defined as $\ell_j = |q_j|^2$ or $\ell_j = |q_j|$, respectively. An N-D SVQ $\boldsymbol{\mathcal{V}}$ is completely defined in terms of the triplet $(\boldsymbol{\mathcal{Q}}, \boldsymbol{\mathcal{L}}, L)$. It is shown in [20] that this simple structure of the SVQ codebook results in fast and efficient algorithms for codebook search and code-vector encoding/decoding. For more details on SVQ, see [20].

3.3 Combined SVQ Shaping and Trellis Coding

In this section we describe how SVQ [20] and Ungerboeck 1-D trellis codes [22] are combined, leading to TB-SVQ [15]. The SVQ structure allows the TB-SVQ to optimally shape the codebook boundary while the underlying trellis code enables it to achieve a significant granular gain. Throughout the discussion in this section, we will only focus on the memoryless Gaussian and Laplacian sources, for which the optimal codebook boundaries are a sphere and a pyramid, respectively.

Let the TB-SVQ reproduction alphabet $\boldsymbol{\Lambda} = \{\pm(2j+1)\delta : j \in \boldsymbol{\mathcal{J}}_m\}$ (where δ is a scaling factor and $\text{card}(\boldsymbol{\Lambda}) = 2m$) be partitioned into four subsets according to Ungerboeck's partition rule [22]. The equivalence between any Ungerboeck 1-D trellis code-sequence and its corresponding magnitude sequence has earlier been established in Section 2.4.2. The basic idea of TB-SVQ is simply to achieve SVQ shaping on the magnitude sequence. We hence proceed to consider an SVQ $\boldsymbol{\mathcal{V}}$ derived from the magnitude set $\boldsymbol{\mathcal{Q}} = \{q_j = (2j+1)\delta : j \in \boldsymbol{\mathcal{J}}_m\}$ and the corresponding set of nonnegative integer lengths $\boldsymbol{\mathcal{L}} = \{\ell_j : j \in \boldsymbol{\mathcal{J}}_m\}$, where ℓ_j is the length associated with the magnitude q_j. As in the SVQ [20], a spherical or pyramidal TB-SVQ codebook boundary can now be realized by setting $\ell_j = |q_j|^2/\delta^2$ or $\ell_j = |q_j|/\delta$, respectively. A trellis code-sequence of symbols from $\boldsymbol{\Lambda}$ is said to be an N-D TB-SVQ code-sequence if every contiguous N-vector taken from the corresponding magnitude sequence is a code-vector of the N-D SVQ derived from the underlying scalar quantizer $\boldsymbol{\mathcal{S}} \equiv (\boldsymbol{\mathcal{Q}}, \boldsymbol{\mathcal{L}})$.

For most efficient implementation of the SVQ-related codebook search and code-vector encoding/decoding, it is suggested in [20] that the length set $\boldsymbol{\mathcal{L}}$ should be shifted and scaled to as small nonnegative integers as possible. The simplest case for the sphere- or pyramid-bounded TB-SVQ codebook corresponds to setting $\ell_j =$

Rate	N-sphere			N-pyramid		
	$N=16$	$N=32$	$N=64$	$N=16$	$N=32$	$N=64$
1.0	7 (0.01)	12 (0.04)	24	5 (0.11)	10 (0.05)	20 (0.01)
1.5	16 (0.03)	30	56 (0.01)	11 (0.02)	20 (0.04)	39 (0.02)
2.0	35 (0.02)	64 (0.01)	121	19 (0.01)	35 (0.01)	67 (0.01)
2.5	74	134	252	30 (0.01)	56 (0.01)	108
3.0	151	272	512	46	86	165 (0.01)

Number in the parenthesis is the corresponding SVQ coding redundancy (in bits/sample).

Table 3.1: The threshold L of the N-sphere and N-pyramid bounded TB-SVQ for various encoding rates and block lengths.

$(|q_j|^2/\delta^2 - 1)/8 = j(j+1)/2$ or $\ell_j = (|q_j|/\delta - 1)/2 = j$, respectively. Now with the given underlying scalar quantizer $\boldsymbol{\mathcal{S}} \equiv (\boldsymbol{\mathcal{Q}}, \boldsymbol{\mathcal{L}})$ where we assume $\boldsymbol{\mathcal{Q}}$ is unbounded $(m = \infty)$[1], a simple enumeration algorithm (see [20]) can be applied to determine the threshold L, which is the largest integer that yields the cardinality of the SVQ codebook $\boldsymbol{\mathcal{Z}}$ no larger than 2^{Nr} where r is the encoding rate in bits/sample. The resulting threshold L for an N-D sphere- or pyramid-bounded TB-SVQ are given in Table 3.1 for various values of N and r. For cases where the SVQ coding redundancies are noticeably large, their values are also given (in parentheses) in Table 3.1.

The granular structure and codebook boundary of the optimal TB-SVQ are already specified by the underlying trellis coding and SVQ shaping, respectively. The only unknown to be determined is the scaling factor δ for the underlying scalar

[1]After L is determined, card($\boldsymbol{\mathcal{Q}}$) can be set back to the largest integer m with $\ell_{m-1} \leq L$.

quantizer $\mathcal{Q}$. Suppose that the optimal TB-SVQ codebook search algorithm (which we will present in the next section) and a long training sequence are available, the TB-SVQ design problem is simply to select the scaling factor δ which yields the smallest empirical quantization distortion. To encode a TB-SVQ code-sequence into a binary stream, every N-vector is taken contiguously from the corresponding magnitude sequence and fed into the SVQ encoder [20]. In the decoder, the binary stream is decoded (by the SVQ decoder) into contiguous N-vectors, yielding the magnitude sequence, from which the TB-SVQ code-sequence can be sequentially reconstructed according to (2.18) and (2.19).

3.4 TB-SVQ Codebook Search Algorithm

In this section we present the TB-SVQ codebook search algorithms and analyze their complexity. We first describe in Section 3.4.1 the optimal TB-SVQ codebook search algorithm that is a combination of the SVQ codebook search algorithm [20] and the Viterbi trellis search algorithm of TCQ [21]. This dynamic programming based algorithm was derived for TB-SVQ-I in [15], but here we provide more details about its implementation. The complexity of this optimal TB-SVQ codebook search algorithm depends linearly on the number of trellis states. However, it can be shown that it is rather inefficient to employ a complex trellis code. In Section 3.4.2, we present a suboptimal TB-SVQ search algorithm that can suppress the dependence of the implementation complexity on the number of trellis states. In Section 3.4.3, the implementation complexity of these two search algorithms will be analyzed and compared.

3.4.1 Full-State Search Algorithm

Let the input sequence $\{x_i\}$ be partitioned into contiguous blocks of N-vectors, yielding $\{\boldsymbol{x}_k\}$ where $\boldsymbol{x}_k \equiv (x_{k,1}, x_{k,2}, \cdots, x_{k,N})$. In what follows, we present the optimal *full-state* TB-SVQ codebook search algorithm that can globally search over the quantizer codebook to determine the nearest code-sequence $\{\hat{\boldsymbol{x}}_k\}$ for $\{\boldsymbol{x}_k\}$.

Denote by Δ_k^s the minimum accumulated distortion that results when the first k source vectors are quantized to a code-sequence whose final trellis state is $s \in \boldsymbol{\Sigma}$. Suppose that the encoder assumes a fixed delay of d N-vectors. To handle this quantization delay, we need a buffer of $(d+1)$ N-vectors $\boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k,0}^s, \boldsymbol{q}_{k,1}^s, \cdots, \boldsymbol{q}_{k,d}^s)$ for each trellis state s that stores the unreleased portion of the trellis sequence resulting in Δ_k^s. Suppose that $k \geq d$ and denote by s^* the trellis state that minimizes Δ_k^s over $\boldsymbol{\Sigma}$, $\boldsymbol{q}_{k,0}^{s^*}$ is hence released as the TB-SVQ code-vector $\hat{\boldsymbol{x}}_{k-d}$. To make the released sequence of N-vectors consistent with the underlying trellis code, we also need another buffer $\boldsymbol{\sigma}_k^s \equiv (\sigma_{k,0}^s, \sigma_{k,1}^s, \cdots, \sigma_{k,d}^s)$ associated with each $\boldsymbol{q}_k^s$ where $\sigma_{k,j}^s \in \boldsymbol{\Sigma}$ denotes the trellis state that $\boldsymbol{q}_{k,j}^s$ reaches for all $j \in \boldsymbol{\mathcal{J}}_{d+1}$. All TB-SVQ code-vector buffers whose $\sigma_{k,0}^s$ is different from $\sigma_{k,0}^{s^*}$ will result in a released sequence inconsistent with the underlying trellis code and should therefore be removed from further consideration.

Suppose that we have just determined Δ_{k-1}^s (and hence $\boldsymbol{q}_{k-1}^s$ and $\boldsymbol{\sigma}_{k-1}^s$) for all $s \in \boldsymbol{\Sigma}$; now consider operating on the k^{th} source vector $\boldsymbol{x}_k$. Denote by $D_i^{l,s}$ the minimum accumulated distortion that results when the first i components of $\boldsymbol{x}_k$ are quantized to $\boldsymbol{z}_i^{l,s} \equiv (z_1^{l,s}, z_2^{l,s}, \cdots, z_i^{l,s})$ whose total length is l and which reaches trellis state s at time instant i. Also, let $t_i^{l,s}$ denote the initial trellis state for $\boldsymbol{z}_i^{l,s}$ (i.e., the trellis state that the i-vector $\boldsymbol{z}_i^{l,s}$ reaches initially at $i = 0$). In the beginning of the k^{th} search cycle, $D_0^{0,s}$ is initialized to the value of Δ_{k-1}^s for

all $s \in \boldsymbol{\Sigma}$. Suppose that $D_i^{l,s}$, $\boldsymbol{z}_i^{l,s}$, and $t_i^{l,s}$ are already determined. Consider for $\boldsymbol{z}_{i+1}^{l,s}$ a candidate $(i+1)$-vector which is the concatenation of the previous i-vector $\boldsymbol{z}_i^{l',s'}$ and one symbol $z \in \boldsymbol{\Lambda}$. For such a candidate $(i+1)$-vector to be consistent with the underlying trellis code, l' must be such that $(l-l') \in \boldsymbol{\mathcal{L}}$ and, according to (2.20), s' must be $\mathcal{P}(s, l-l')$[2]. According to (2.18), z is hence constrained to satisfy $z = \mathcal{F}(s', l-l')$. To determine $\boldsymbol{z}_{i+1}^{l,s}$, we need to solve for the pair (l^*, s^*) that satisfies the aforementioned constraints and minimizes the accumulated distortion $D_{i+1}^{l,s}$; that is, we need to determine

$$(l^*, s^*) = \arg\min_{(l',s')} \left[D_i^{l',s'} + [x_{k,i+1} - \mathcal{F}(s', l-l')]^2 \right]. \tag{3.4}$$

By solving (3.4), byproducts such as $D_{i+1}^{l,s}$, $\boldsymbol{z}_{i+1}^{l,s}$, and $t_{i+1}^{l,s}$ are also obtained. The above operation repeats until $i = N$ when we have $D_N^{l,s}$, $\boldsymbol{z}_N^{l,s}$, and $t_N^{l,s}$ available. We then search over $l \in \boldsymbol{\mathcal{J}}_{L+1}$ for l^* that minimizes $D_N^{l,s}$ for each $s \in \boldsymbol{\Sigma}$. $D_N^{l^*,s}$ is then copied to Δ_k^s to be used for the next search cycle. The initial trellis state for $\boldsymbol{z}_N^{l^*,s}$ is $t_N^{l^*,s}$, based on which the buffers $\boldsymbol{q}_k^s$ and $\boldsymbol{\sigma}_k^s$ are updated as $\boldsymbol{q}_k^s = (\boldsymbol{q}_{k-1}^{t_N^{l^*,s}}, \boldsymbol{z}_N^{l^*,s})$ and $\boldsymbol{\sigma}_k^s = (\boldsymbol{\sigma}_{k-1}^{t_N^{l^*,s}}, s)$, respectively. As a summary, a C-like programming language description of the full-state search algorithm is presented in Figure 3.1.

One can visualize the full-state TB-SVQ codebook search algorithm as follows. The algorithm is essentially a sliding-block Viterbi algorithm [40] that operates on each N-vector based on a 3-D search grid of $N \times (L+1) \times |\boldsymbol{\Sigma}|$ points where L is the SVQ threshold length. The special case where there is no extra quantization delay $(d = 0)$ corresponds to a genuine block-based type of search algorithm. Within

[2]It should be mentioned that there is a one-to-one correspondence between $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{Q}}$ in the specific sphere or pyramid bounded TB-SVQ we are considering. Due to this equivalence, we can replace $\boldsymbol{\mathcal{Q}}$ with $\boldsymbol{\mathcal{L}}$ in the input domain of the three functions $\mathcal{F}(\boldsymbol{\Sigma}, \cdot)$, $\mathcal{N}(\boldsymbol{\Sigma}, \cdot)$, and $\mathcal{P}(\boldsymbol{\Sigma}, \cdot)$ described in Section 2.4.2; this will be implicitly assumed in the following discussion.

for $(s \in \boldsymbol{\Sigma})$ if $(s ==$ a specified trellis state$)$ $\Delta_{-1}^{s} = 0$; else $\Delta_{-1}^{s} = \infty$;
for $(k = 0;\ k < \infty;\ k++)$ {

for $(s \in \boldsymbol{\Sigma};\ l \in \boldsymbol{\mathcal{J}}_{L+1})$ if $(l == 0)$ $D_0^{l,s} = \Delta_{k-1}^{s}$; else $D_0^{l,s} = \infty$;
for $(s \in \boldsymbol{\Sigma})$ $\boldsymbol{z}_0^{0,s} = \emptyset$;
for $(i = 0;\ i < N;\ i++)$ for $(s \in \boldsymbol{\Sigma};\ l \in \boldsymbol{\mathcal{J}}_{L+1})$ {

$(l^*, s^*) = \arg\min_{(l',s')} \left[D_i^{l',s'} + [x_{k,i+1} - \mathcal{F}(s', l - l')]^2 \right]$;
$z = \mathcal{F}(s^*, l - l^*)$;
$D_{i+1}^{l,s} = D_i^{l^*,s^*} + (x_{k,i+1} - z)^2$;
$\boldsymbol{z}_{i+1}^{l,s} \equiv \left(\boldsymbol{z}_i^{l^*,s^*}, z \right)$;
if $(i == 0)$ $t_{i+1}^{l,s} = s^*$; else $t_{i+1}^{l,s} = t_i^{l^*,s^*}$;

}
for $(s \in \boldsymbol{\Sigma})$ {

$l^* = \arg\min_{l \in \boldsymbol{\mathcal{J}}_{L+1}} D_N^{l,s}$;
$\Delta_k^s = D_N^{l^*,s}$; $\boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k-1}^{t_N^{l^*,s}}, \boldsymbol{z}_N^{l^*,s})$; $\boldsymbol{\sigma}_k^s \equiv (\boldsymbol{\sigma}_{k-1}^{t_N^{l^*,s}}, s)$;

}
if $(k \geq d)$ {

$s^* = \arg\min_{s \in \boldsymbol{\Sigma}} \Delta_k^s$;
release $\hat{\boldsymbol{x}}_{k-d} = \boldsymbol{q}_{k,0}^{s^*}$ for TB-SVQ code-vector encoding;
for $(s \in \boldsymbol{\Sigma})$ {

if $(\sigma_{k,0}^s \neq \sigma_{k,0}^{s^*})$ $\Delta_k^s = \infty$;
$\boldsymbol{\sigma}_k^s \equiv (\sigma_{k,1}^s, \sigma_{k,2}^s, \cdots, \sigma_{k,d}^s)$;
$\boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k,1}^s, \boldsymbol{q}_{k,2}^s, \cdots, \boldsymbol{q}_{k,d}^s)$;

}

}

}

Figure 3.1: Full-state TB-SVQ codebook search algorithm.

each search cycle, associated with the grid point (i, l, s) at time instant i, there is an optimal i-vector $\boldsymbol{z}_i^{l,s}$ that has an accumulated length l and reaches trellis state s. The resulting accumulated distortion is $D_i^{l,s}$. Before operating on $\boldsymbol{x}_k$, the value of Δ_{k-1}^s is copied to $D_0^{0,s}$ for all $s \in \boldsymbol{\Sigma}$. We then sequentially (in i) determine $\boldsymbol{z}_i^{l,s}$ according to (3.4) and thereby update $D_i^{l,s}$ for all $l \in \boldsymbol{\mathcal{J}}_{L+1}$ and $s \in \boldsymbol{\Sigma}$. After $D_N^{l,s}$ and $\boldsymbol{z}_N^{l,s}$ are determined, for each s, the minimum $D_N^{l,s}$ (over $l \in \boldsymbol{\mathcal{J}}_{L+1}$) is copied to Δ_k^s and the buffers $\boldsymbol{q}_k^s$ can be accordingly updated.

3.4.2 State-Suppressed Search Algorithm

The implementation complexity of the full-state search algorithm described in the previous subsection is proportional to the number of trellis states. As can be observed from Table 2.2, after the first 1 dB or so granular gain has been realized by the 4-state trellis code, of the remaining granular gain only about 0.05-0.10 dB is realized for every factor of two increase in the number of trellis states. This indicates that it is rather inefficient (in terms of the tradeoff between complexity and performance) to employ a complex trellis code. In what follows, we present a suboptimal TB-SVQ search algorithm that can suppress the dependence of the implementation complexity on the number of trellis states.

The basic idea of what will be called the *state-suppressed* search algorithm is as follows. As opposed to the full-state search algorithm in which each search grid point is a 3-D composite of the time instant i, the accumulated length l, and the trellis state s, the state-suppressed search algorithm suppresses the trellis state dimension and simply assigns a trellis state s_i^l that satisfies the state transition constraint for the $(i, l)^{th}$ point in the now 2-D search grid. We hence do not need state-dependent buffers (like $\boldsymbol{q}_k^s$ and $\boldsymbol{\sigma}_k^s$ as in the full-state search algorithm)

$s^* =$ a pre-specified trellis state;
for $(k = 0;\ k < \infty;\ k++)$ {

for $(l \in \mathcal{J}_{L+1})$ if (l==0) $D_0^l = 0$; else $D_0^l = \infty$;
$s_0^0 = s^*$;
for $(i = 0;\ i < N;\ i++)$ for $(l \in \mathcal{J}_{L+1})$ {

$l^* = \arg\min_{l'} \left[D_i^{l'} + [x_{k,i+1} - \mathcal{F}(s_i^{l'}, l - l')]^2 \right]$;
$z = \mathcal{F}(s_i^{l^*}, l - l^*)$;
$D_{i+1}^l = D_i^{l^*} + (x_{k,i+1} - z)^2$;
$\boldsymbol{z}_{i+1}^l \equiv \left(\boldsymbol{z}_i^{l^*}, z\right)$;
$s_{i+1}^l = \mathcal{N}(s_i^{l^*}, l - l^*)$;

}
$l^* = \arg\min_{l \in \mathcal{J}_{L+1}} D_N^l$;
$s^* = s_N^{l^*}$;
release $\hat{\boldsymbol{x}}_k = \boldsymbol{z}_N^{l^*}$ for TB-SVQ code-vector encoding;

}

Figure 3.2: State-suppressed TB-SVQ codebook search algorithm.

because the freedom of having a code-sequence reaching a specified trellis state at any time instant is removed. Unlike the sliding-block coding nature of the full-state search algorithm, the state-suppressed search algorithm is a genuine block-based scheme and allows no extra quantization delay.

Denote by D_i^l the distortion that results when the first i source samples of $\boldsymbol{x}_k$ are quantized to the optimal i-vector whose accumulated length is l. This i-vector is denoted by $\boldsymbol{z}_i^l \equiv (z_1^l, z_2^l, \cdots, z_i^l)$. Also, let s_i^l denote the trellis state that $\boldsymbol{z}_i^l$ reaches. In the beginning of operating on each source N-vector, D_0^0 and s_0^0 are set to 0 and a pre-specified trellis state, respectively. Suppose that D_i^l, $\boldsymbol{z}_i^l$, and s_i^l are already determined for all $l \in \mathcal{J}_{L+1}$. Consider for $\boldsymbol{z}_{i+1}^l$ a candidate $(i+1)$-vector which is a concatenation of the previous i-vector $\boldsymbol{z}_i^{l'}$ and some symbol $z \in \boldsymbol{\Lambda}$. For

such a candidate $(i+1)$-vector to be consistent with the underlying trellis code, l' must be such that $(l-l') \in \mathcal{L}$ and the corresponding trellis state s_{i+1}^{l} and symbol z must satisfy $s_{i+1}^{l} = \mathcal{N}(s_i^{l'}, l-l')$ and $z = \mathcal{F}(s_i^{l'}, l-l')$, respectively. To determine $\boldsymbol{z}_{i+1}^{l}$, we need to solve for l^* that satisfies the aforementioned constraints and minimizes the accumulated distortion D_{i+1}^{l}; that is, we need to determine

$$l^* = \arg\min_{l'} \left[D_i^{l'} + [x_{i+1} - \mathcal{F}(s_i^{l'}, l-l')]^2 \right]. \tag{3.5}$$

By solving (3.5), byproducts such as D_{i+1}^{l}, $\boldsymbol{z}_{i+1}^{l}$, and s_{i+1}^{l} are also obtained. The above operation repeats until $i = N$ when we have D_N^l and $\boldsymbol{z}_N^l$ available. Finally, we search for l^* that minimizes D_N^l over $l \in \mathcal{J}_{L+1}$ and release $\boldsymbol{z}_N^{l^*}$ as $\hat{\boldsymbol{x}}_k$. The state-suppressed TB-SVQ codebook search algorithm is summarized in Figure 3.2.

3.4.3 Complexity Issues

Recall that a TB-SVQ code-sequence is converted into its corresponding magnitude sequence before a magnitude-based SVQ encoding algorithm [20] is employed to further encode it into a binary stream. The code-vector encoding complexity (in terms of operations per sample and overall memory requirement) can therefore be determined as in [20] and, under our new notations, is presented in Table 3.2. We should mention that for code-vector encoding this result will be common to all variations of the TB-SVQ to be presented in the sequel and hence will not be reiterated.

We next analyze the implementation complexity associated with the full-state codebook search algorithm. Determining each $D_{i+1}^{l,s}$, through solving (3.4), requires $4m-1$ operations ($2m$ additions, m multiplications, and $m-1$ comparisons) where $m = \text{card}(\mathcal{Q})$. This costs $2^{\nu}N(L+1)(4m-1)$ operations per search cycle. At

Computational (32-bit Adds/Sample)	Storage (32-bit Words)
$Nmr/32$	$N^2(L+1)r/32$

Table 3.2: TB-SVQ code-vector encoding complexity.

the end of each search cycle, it takes $2^\nu L$ comparisons to determine the least accumulated distortion for each trellis state. Also, at the end of each search cycle, it requires extra $2^\nu - 1$ comparisons to determine the optimal candidate trellis code-sequence. Overall, the computational requirement is approximately $2^\nu(L+1)(4m-1)$ operations per source sample. The major storage requirement comes from variables such as $\Delta^s, D^{l,s}, \boldsymbol{z}^{l,s}, t^{l,s}, \boldsymbol{q}^s$, and $\boldsymbol{\sigma}^s$. All except Δ^s need to be alternatively updated and hence cost twice as much in the required storage. Let us assume that each accumulated distortion variable (such as Δ^s and $D^{l,s}$) is stored in a 32-bit word. The overall cost for the accumulated distortion variables is $2^\nu(2L+3)$ 32-bit words. Each component of $\boldsymbol{z}^{l,s}$ and $\boldsymbol{q}^s$ can be stored in the form of an index of $\lceil \log_2 m \rceil$ bits to the magnitude set $\boldsymbol{\mathcal{Q}}$. The total storage cost for $\boldsymbol{z}^{l,s}$ and $\boldsymbol{q}^s$ is hence $2^{\nu+1}N(d+L+2)\lceil \log_2 m \rceil/32$ 32-bit words. Each component of $t^{l,s}$ and $\boldsymbol{\sigma}^s$ is a ν-bit index to $\boldsymbol{\Sigma}$; the total storage cost in this regard is $2^{\nu+1}N\nu(d+L+2)/32$ 32-bit words. Overall, the storage requirement of the TB-SVQ full search algorithm is approximately the sum of all we have just mentioned and is given in Table 3.3.

The implementation complexity associated with the state-suppressed search algorithm can be similarly quantified. We omit the complexity analysis here but present the computational and storage requirements in Table 3.3. Clearly, the

	Computational (Operations/Sample)	Storage (32-bit Words)
Full-State	$2^{\nu}(L+1)(4m-1)$	$2^{\nu}\left[(2L+3)+\frac{N(d+L+2)(\nu+\lceil\log_2 m\rceil)}{16}\right]$
State-Suppressed	$(L+1)(4m-1)$	$2(L+1)\left[1+\frac{\nu+N\lceil\log_2 m\rceil}{32}\right]$

Table 3.3: TB-SVQ codebook search complexity.

state-suppressed search algorithm is less complicated than the full-state search algorithm by about a factor equal to the number of trellis states.

To get a clearer picture of the implementation complexity of these two TB-SVQ codebook search algorithms, let us consider a more concrete example in which we quantize the memoryless Gaussian source at 1 bit/sample using a 64-D 8-state TB-SVQ; we have $N = 64$, $\nu = 3$. From Table 3.1, we can determine $L = 24$ and subsequently $m = 7$. Also, suppose that the full-state search algorithm assumes an encoding delay of four N-vectors (i.e., $d = 4$). Now, using Table 3.3, we can determine that the computational and storage requirement for the full-state search algorithm are 5,400 operations per source sample and 6,168 32-bit words, respectively. For the state-suppressed search algorithm, the computational and storage requirement are 675 operations per source sample and about 355 32-bit words, respectively.

3.5 Predictive TB-SVQ

In this section, we consider a combination of both the TB-SVQ and DPCM schemes — called *predictive* TB-SVQ — for quantizing Markov sources. The dynamic programming based TB-SVQ codebook search algorithm presented in Section 3.4 is a multipath search method in which each search path stores a candidate quantized sequence. The basic idea behind predictive TB-SVQ is simply to perform the DPCM predictive coding operation in each search path of the TB-SVQ codebook search algorithm. A similar idea has been applied to the TCQ codebook search algorithm, leading to the so-called *predictive* TCQ (PTCQ) [21]. Before describing the algorithm we look at a naive coding scheme so as to motivate the predictive TB-SVQ scheme.

First, we should mention that there is an invertible linear function $1/H(z)$ (see Section 2.1) that maps $\{x_i\}$ — a realization of some p^{th}-order Markov source to its corresponding innovations sequence $\{w_i\}$. Each innovations sample w_i can be obtained as a linear combination of several source samples as follows:

$$w_i = x_i - \sum_{j=1}^{p} \rho_j x_{i-j}. \tag{3.6}$$

A naive quantization scheme for such a Markov source is one that quantizes the innovations sequence $\{w_i\}$ and reconstructs the replica $\hat{x}_i$ for x_i as the output of source filter $H(z)$ driven by the quantized innovations sequence $\{\hat{w}_i\}$; that is,

$$\hat{x}_i = \hat{w}_i + \sum_{j=1}^{p} \rho_j \hat{x}_{i-j}. \tag{3.7}$$

For example, the D*PCM scheme (see [3]) belongs to this class of source coders that employs an underlying scalar quantizer to encode the innovations sequence. One may now replace the scalar quantizer in D*PCM with a more powerful quantizer

(like TB-SVQ) in an attempt to realize both the boundary and granular gains in the innovations domain. For this class of coders, however, optimal quantization in the innovations domain does not necessarily lead to optimal quantization in the source domain. This is due to the so-called *quantization error accumulation* phenomenon [3] which we briefly explain as follows. Define $e_i \stackrel{\Delta}{=} x_i - \hat{x}_i$ as the quantization error in the source domain. By substracting (3.7) off (3.6), one obtains

$$e_i = w_i - \hat{w}_i + \sum_{j=1}^{p} \rho_j e_{i-j}, \tag{3.8}$$

indicating that e_i is equal to the innovations quantization error $w_i - \hat{w}_i$ plus some accumulation of the most recent p source quantization errors.

The basic idea of DPCM — a closed-loop variation of D*PCM — is that of quantizing the prediction residual $\epsilon_i = x_i - \tilde{x}_i$ where

$$\tilde{x}_i = \sum_{j=1}^{p} \rho_j \hat{x}_{i-j}, \tag{3.9}$$

is a prediction of x_i that is also known to the decoder. Denote by $\hat{\epsilon}_i$ the quantized version of ϵ_i. In the decoder, the quantized replica for x_i is obtained as $\hat{x}_i = \hat{\epsilon}_i + \tilde{x}_i$. Obviously, the source quantization error $x_i - \hat{x}_i$ is identical to the quantization error $\epsilon_i - \hat{\epsilon}_i$ in the prediction residual domain and there is hence no quantization error accumulation in DPCM.

Now we consider the predictive TB-SVQ scheme for quantizing Markov sources. Essentially, we wish to generalize the underlying scalar quantizer in DPCM to the more powerful TB-SVQ (so as to realize both the granular and boundary gains in the prediction residual domain). In practice, however, we perform the DPCM predictive coding operation in each search path of the TB-SVQ codebook search algorithm. For simplicity, we next describe this generalization only on the state-

suppressed search algorithm presented in Section 3.4.2; the extension for the full-state search algorithm should be straightforward and will be omitted.

Let us take the state-suppressed TB-SVQ codebook search algorithm described in Section 3.4.2 as a baseline algorithm and consider the extra effort required to perform the DPCM operation in each search path. First, we need an extra variable $\hat{\boldsymbol{x}}_i^l \equiv (\hat{x}_{i-p+1}^l, \hat{x}_{i-p+2}^l, \cdots, \hat{x}_i^l)$ for each l that stores the most recent p quantized samples associated with $\boldsymbol{z}_i^l$ — the quantized i-vector for the "pathwise" prediction residuals. Suppose that D_i^l, $\boldsymbol{z}_i^l$, $\hat{\boldsymbol{x}}_i^l$, and s_i^l are just determined, we next show how these variables are updated. It should be mentioned that a prediction of $x_{k,i+1}$ associated with $\boldsymbol{z}_i^{l'}$ can be made as

$$\tilde{x}^{l'} = \sum_{j=1}^{p} \rho_j \hat{x}_{i+1-j}^{l'}, \tag{3.10}$$

and the corresponding predicton residual is $x_{k,i+1} - \tilde{x}^{l'}$. Treating the predicton residual as an input to the quantizer, the minimization problem in (3.5) becomes

$$l^* = \arg\min_{l'} \left[D_i^{l'} + [x_{k,i+1} - \tilde{x}^{l'} - \mathcal{F}(s_i^{l'}, l - l')]^2 \right]. \tag{3.11}$$

By solving (3.11), byproducts such as D_{i+1}^l, $\boldsymbol{z}_{i+1}^l$, and $\hat{\boldsymbol{x}}_{i+1}^l$ are also obtained. The above operation repeats until $i = N$ and the rest of the algorithm is the same as in Figure 3.2. In the decoder, the decoded TB-SVQ code-sequence is delivered as input to the source filter $H(z)$ to produce the reproduction sequence $\{\hat{x}_i\}$. A C-like program summarizing the state-suppressed predictive TB-SVQ codebook search algorithm is provided in Figure 3.3.

To optimally design this type of predictive source coders, a key problem is to design the optimal quantizer matched to the probability density function (p.d.f.) of the prediction residuals. However, there is an interdependence between the quantizer structure and the prediction residual process. This interdependence has

$s^* =$ a pre-specified trellis state;
for $(k = 0;\ k < \infty;\ k++)$ {

for $(l \in \mathcal{J}_{L+1})$ if (l==0) $D_0^l = 0$; else $D_0^l = \infty$;
$s_0^0 = s^*$;
for $(i = 0; i < N;\ i++)$ {

for $(l \in \mathcal{J}_{L+1})\ \tilde{x}^l = \sum_{j=1}^{p} \rho_j \hat{x}_{i+1-j}^l$;
for $(l \in \mathcal{J}_{L+1})$ {

$l^* = \arg\min_{l'} \left[D_i^{l'} + [x_{k,i+1} - \tilde{x}^{l'} - \mathcal{F}(s_i^{l'}, l - l')]^2 \right]$;
$z = \mathcal{F}(s_i^{l^*}, l - l^*)$;
$D_{i+1}^l = D_i^{l^*} + (x_{k,i+1} - \tilde{x}^{l^*} - z)^2$;
$\hat{\boldsymbol{x}}_{i+1}^l \equiv (\hat{\boldsymbol{x}}_i^{l^*}, \tilde{x}^{l^*} + z)$;
$\boldsymbol{z}_{i+1}^l \equiv \left(\boldsymbol{z}_i^{l^*}, z\right)$;
$s_{i+1}^l = \mathcal{N}(s_i^{l^*}, l - l^*)$;

}

}
$l^* = \arg\min_{l \in \mathcal{J}_{L+1}} D_N^l$; $\quad s^* = s_N^{l^*}$; $\quad \hat{\boldsymbol{x}}_0^0 = \hat{\boldsymbol{x}}_N^{l^*}$;
release $\boldsymbol{z}_N^{l^*}$ for TB-SVQ code-vector encoding;

}

Figure 3.3: State-suppressed predictive TB-SVQ codebook search algorithm.

been shown to be very complicated even in the simplest case — the DPCM [42]. Partially because of this interdependence that leads to difficulties in determining the actual residual p.d.f., and partially due to the fact that at high rates the residual process is very close to the innovations process, it is common to design the quantizer for the p.d.f. of the innovations process; we adopt this simple practice in the course of developing the predictive TB-SVQ.

The implementation complexity associated with the predictive TB-SVQ (using either the state-suppressed or full-state search algorithm) can be easily analyzed

	Computational (Operations/Sample)	Storage (32-bit Words)
Full-State	$2^{\nu}(L+1)(m+2p)$	$2^{\nu}(2p+1)(L+1)$
State-Suppressed	$(L+1)(m+2p)$	$(2p+1)(L+1)$

Table 3.4: Extra complexity of predictive TB-SVQ codebook search.

and is presented, in terms of extra complexity as compared to that required by the baseline algorithm for memoryless sources (see Table 3.3), in Table 3.4. We should also mention that the predictive TB-SVQ codebook search algorithm is merely suboptimal. The suboptimality is due to the "greedy" nature in truncating the search paths which currently have larger accumulated distortions than the survivor one. As the future prediction residuals depend on the current quantization, chances are that paths which eventually lead to smaller quantization distortions are already abandoned.

3.6 Simulation Results

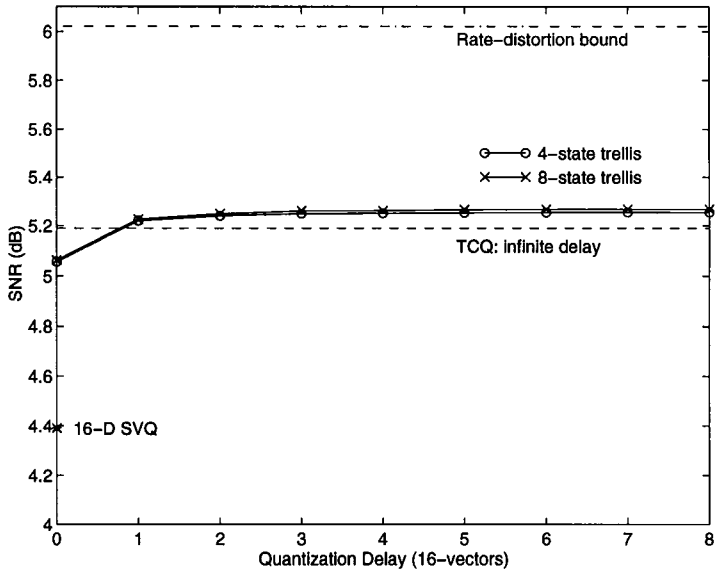
The TB-SVQ and predictive TB-SVQ performance are presented and discussed in this section. The results reported are given as signal-to-noise ratio (SNR) in dB and are obtained based on simulations performed on at least 160,000 source samples. The memoryless sources we consider here include Gaussian and Laplacian sources, while for Markov sources we consider two specific types of Gauss-Markov sources. The PTCQ performance for Markov sources [21] will be included for comparison

against the predictive TB-SVQ, while for memoryless sources the performance of the SVQ [20], TB-SVQ-I and TB-SVQ-II [15], and TCQ [21] will be included to compare against the TB-SVQ. We will also include the rate-distortion limit for the specific source in all cases.

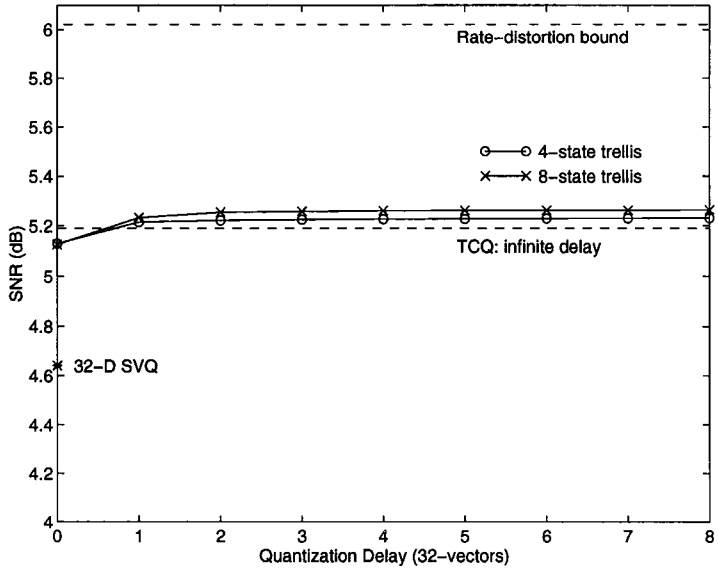
We first consider the quantizer performance as a function of the quantization delay[3]. The dependence of the TB-SVQ performance on the delay d (in number of N-vectors) for encoding a memoryless Gaussian source at 1 bit/sample is presented in Figure 3.4; the block-length N is 16 or 32 for subfigure (a) or (b), respectively. Similar plots for the predictive TB-SVQ for two types of Gauss-Markov sources are given in Figure 3.5. These plots indicate that the delay required to saturate the performance is smaller when the underlying trellis code has fewer states or the block-length is larger. In generally, both the TB-SVQ and predictive TB-SVQ require only a delay of one (16-D or 32-D) source vector to perform within 0.05 dB of the saturated result (which can be obtained with an extremely large quantization delay).

Now we shall assume a quantization delay of five source N-vectors which nearly saturates the TB-SVQ performance using the full-state search algorithm. Table 3.5 illustrates the TB-SVQ performace with various block-lengths and encoding rates; all quantizers considered here are based on Ungerboeck 4-state 1-D trellis code. Simulation results in Table 3.5 indicate that TB-SVQ is no inferior to TB-SVQ-I or TB-SVQ-II for the Gaussian source. TB-SVQ-I, however, outperforms TB-SVQ for the Laplacian source. This is probably because TB-SVQ-I (whose underlying scalar quantizer is non-uniform) can realize the non-uniform density gain which is significant for the Laplacian source. TB-SVQ consistenly outperforms TCQ in

[3]The full-state codebook search algorithm is implicitly assumed.

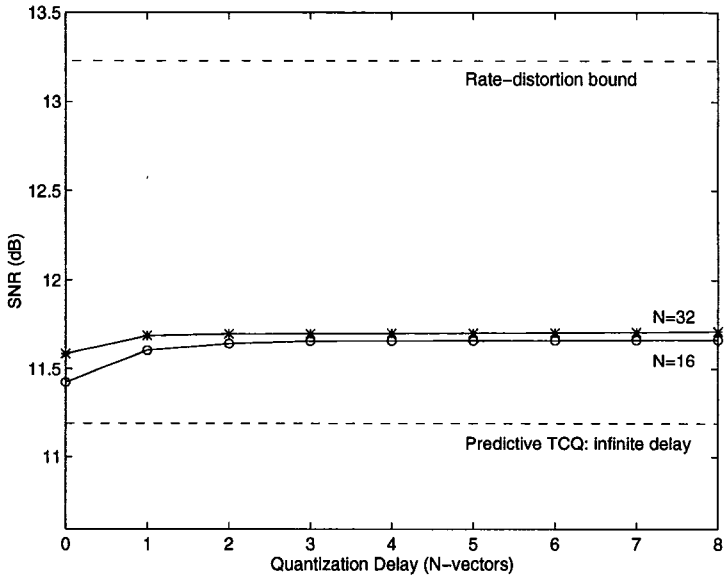


(a) $N = 16$.

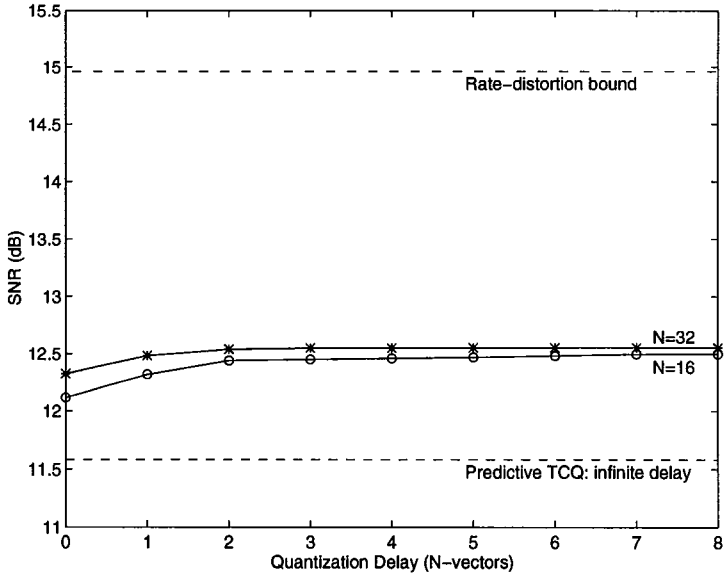


(b) $N = 32$.

Figure 3.4: Performance (SNR in dB) of TB-SVQ on encoding a memoryless Gaussian source at 1 bit/sample versus the quantization delay.



(a) $\rho_1 = 0.9$.



(b) $\rho_1 = 1.515$, $\rho_2 = -0.752$.

Figure 3.5: Performance (SNR in dB) of predictive TB-SVQ (with 4-state trellis) on encoding a (a) 1^{th}-order (b) 2^{nd}-order Gauss-Markov source at 1 bit/sample versus the quantization delay.

Source	Rate	Full-State TB-SVQ			TB-SVQ-I	TB-SVQ-II	TCQ	$R(D)$
		$N = 16$	$N = 32$	$N = 64$	$N = 32$	$N = 64$		
Gaussian	1.0	5.26	5.24	5.49	5.14	5.47	5.00	6.02
	1.5	8.01	8.30	8.38	—	8.35	—	9.03
	2.0	10.84	11.10	11.26	11.11	11.51	10.56	12.04
Laplacian	1.0	4.79	5.37	5.72	5.85	5.68	4.34	6.62
	1.5	8.09	8.36	8.70	—	8.64	—	9.64
	2.0	10.90	11.31	11.58	11.52	11.45	9.41	12.66

Table 3.5: Performance (SNR in dB) of the 4-state TB-SVQ using the full-state search algorithm with a delay of five N-vectors on encoding memoryless Gaussian and Laplacian sources.

all cases we considered. This should be credited to the boundary gain realized by the SVQ shaping of the codebook boundary. We should also mention that the TB-SVQ-I quantization delay, as reported in [15], is small (large) when quantizing the Gaussian (Laplacian) distribution, while the TCQ quantization delay is equal to the source sequence length.

Presented in Table 3.6 is the performance of all the quantizers considered in Table 3.5 but now based on Ungerboeck 8-state 1-D trellis code. A comparison between Tables 3.5 and 3.6 suggests that the 8-state TB-SVQ can achieve over the 4-state counterpart an improvement of less than 0.1 dB at the cost of approximately twice as much implementation complexity. It is foreseeable that the improvement with increasing the number of trellis states shall become less significant while the implementation complexity continues to increase, suggesting that it will be inefficient to employ a TB-SVQ with a trellis code that has too many states.

The state-suppressed search algorithm is known to be simpler than the full-state search algorithm by a factor approximately equal to the number of trellis states; see Tables 3.3 and 3.4. We next study the relative merits of the two

Source	Rate	Full-State TB-SVQ			TB-SVQ-I	TB-SVQ-II	TCQ	$R(D)$
		$N = 16$	$N = 32$	$N = 64$	$N = 32$	$N = 64$		
Gaussian	1.0	5.28	5.26	5.55	5.27	5.52	5.19	6.02
	1.5	8.04	8.35	8.45	—	8.40	—	9.03
	2.0	10.89	11.15	11.33	11.19	11.26	10.70	12.04
Laplacian	1.0	4.80	5.39	5.76	5.91	5.70	4.47	6.62
	1.5	8.12	8.42	8.75	—	8.64	—	9.64
	2.0	10.92	11.37	11.63	11.62	11.50	9.56	12.66

Table 3.6: Performance (SNR in dB) of the 8-state TB-SVQ using the full-state search algorithm with a delay of five N-vectors on encoding memoryless Gaussian and Laplacian sources.

search algorithms by comparing the performance of the corresponding TB-SVQ and predictive TB-SVQ schemes on encoding memoryless (Gaussian and Laplacian) and Gauss-Markov sources, which are presented in Table 3.7 and Table 3.8, respectively. Note that the state-suppressed search algorithm operates on each source vector separately and the resulting quantizer is hence a genuine block coding scheme. For a fair comparison, we assume no extra quantization delay in the full-state search algorithm; hence the resulting quantizer is also a genuine block coding scheme. The simulation results in Tables 3.7 and 3.8 suggest that the state-supressed search algorithm results in an SNR performance loss of about 0.1 to 0.4 dB; the performance loss is generally smaller at a higher encoding rate or with a smaller block-length. It should be mentioned that this performance loss is paid off by a reduction in the implementation complexity by a factor equal to the number of trellis states. In most cases, the TB-SVQ using the state-supressed search algorithm still outperforms the SVQ which has about the same implementation complexity but cannot achieve the granular gain for the memoryless sources. The predictive TB-SVQ using the state-supressed search algorithm also significantly

outperforms PTCQ[4] for the Gauss-Markov sources. This is credited to a better codebook shaping by the SVQ in the prediction residual domain.

Source	Rate	Full-State		State-Suppressed		SVQ		$R(D)$
		$N = 16$	$N = 32$	$N = 16$	$N = 32$	$N = 16$	$N = 32$	
Gaussian	1.0	5.09	5.13	4.71	4.67	4.39	4.64	6.02
	1.5	7.82	8.21	7.50	7.79	7.41	7.55	9.03
	2.0	10.66	10.99	10.37	10.65	10.29	10.38	12.04
	2.5	13.44	13.78	13.27	13.57	12.94	13.15	15.05
	3.0	16.15	16.52	16.10	16.41	15.62	15.96	18.06
Laplacian	1.0	4.59	5.25	4.30	4.89	5.32	5.54	6.62
	1.5	7.90	8.24	7.63	7.93	8.02	8.23	9.64
	2.0	10.69	11.14	10.50	10.86	10.50	10.82	12.66
	2.5	13.32	13.87	13.14	13.66	12.96	13.45	15.67
	3.0	15.97	16.54	15.80	16.42	15.23	15.91	18.68

Table 3.7: Performance (SNR in dB) of the 4-state TB-SVQ using the full-state search algorithm (wth no extra quantization delay) and the state-suppressed search algorithm on encoding memoryless Gaussian and Laplacian sources.

Source Type	Rate	Full-State			State-Suppressed			PTCQ	$R(D)$
		$N = 16$	$N = 32$	$N = 64$	$N = 16$	$N = 32$	$N = 64$		
	1.0	11.42	11.59	12.14	11.29	11.42	11.83	11.19	13.23
AR(1):	2.0	17.89	18.35	18.61	17.71	18.10	18.32	17.21	19.25
$\rho_1 = 0.9$	3.0	23.60	24.09	24.38	23.50	23.91	24.19	22.92	25.27
AR(2):	1.0	12.12	12.33	12.95	11.62	11.70	12.39	11.58	14.96
$\rho_1 = 1.515$	2.0	19.68	20.22	20.47	19.62	20.11	20.41	18.92	21.64
$\rho_2 = -0.752$	3.0	25.88	26.27	26.62	25.69	26.20	26.52	25.00	27.66

Table 3.8: Performance (SNR in dB) of the 4-state predictive TB-SVQ using the full-state search algorithm (wth no extra quantization delay) and the state-suppressed search algorithm on encoding two specified Gauss-Markov sources.

[4]The PTCQ quantization delay is equal to the entire source sequence length which is 1,000 [21].

3.7 Summary and Conclusions

In this chapter several variations on the TB-SVQ for quantizing stationary sources were presented. For memoryless sources the TB-SVQ we considered is a hybrid variation of the TB-SVQs proposed by Laroia and Farvardin [15] — the codebook structure is the same as that of TB-SVQ-II while the dynamic programming based algorithm of TB-SVQ-I is used for optimal codebook search. For Markov sources TB-SVQ was generalized to the predictive TB-SVQ that performs the DPCM operation of encoding the prediction residual in each search path of the dynamic programming based TB-SVQ codebook search algorithm.

Given a sufficiently large quantization delay, the full-state search algorithm globally (hence optimally) searches over the TB-SVQ codebook and is better than the AEP-motivated search algorithm of TB-SVQ-II. The full-state TB-SVQ codebook search algorithm is optimal for memoryless sources while for Markov sources the full-state predictive TB-SVQ codebook search algorithm is suboptimal due to the greedy nature of possibly selecting the suboptimal survivor search path. We demonstrated that it is too costly to employ a highly complex trellis code attempting to realize all the possible granular gain. We have thereby proposed a suboptimal state-suppressed search algorithm that can still realize some granular gain but also removes the dependence of the implementation complexity on the number of trellis states.

All these codebook search algorithms have been derived and are presented in a C-like programming language format. Also, their implementation complexity has been carefully analyzed.

Simulation results presented indicate that TB-SVQ using the full-state search algorithm performs slightly better than TB-SVQ-II and outperforms most known

resolution-constrained quantization schemes reported in the literature. For Markov sources the predictive TB-SVQ is also very competitive and performs very close to the rate-distortion limit. We have observed that the state-supressed search algorithm sacrifices only an SNR performance loss of about 0.1 to 0.4 dB for a reduction in the implementation complexity at a factor equal to the number of trellis states. The quantizers presented in this chapter should be specifically useful when the block-length is large (the codebook structure is optimal according to the AEP) and the encoding rate is low (the implementaion cost is low). Work is underway to apply a 64-D predictive TB-SVQ in the context of low-rate speech waveform coding [33].

Chapter 4

Combined Precoding and TB-SVQ

4.1 Introduction and Outline

While TB-SVQ can realize both the boundary and granular gains for memoryless sources, it is not as straightforward to realize these gains for Markov sources. The predictive TB-SVQ described in Section 3.5 does realize some of both gains in the prediction residual domain and performs very closely to the rate-distortion limit for Markov sources. There are, however, two unsolved issues that keep it from further approaching the rate-distortion bound. The first issue comes from the difficulty in determining the probability density function of the prediction residual process, which is due to the complex interdependence between TB-SVQ and the corresponding prediction residual. The second issue is that the available full-state codebook search algorithm is merely suboptimal; while an optimal codebook search algorithm, if exists, will be extremely complicated and hence prohibitive. Another

drawback of predictive TB-SVQ is that at high rates the search algorithm (full-state or state-suppressed) has a large implementation complexity. In this chapter, we describe another extension of TB-SVQ to Markov sources that is capable of realizing both boundary and granular gains and can asymptotically (in both rate and block-length) achieve the rate-distortion limit.

We first explain why it is not a simple matter to realize both the boundary and granular gains for Markov sources by considering the example of a Gauss-Markov source. It is well known that the optimal N-D codebook boundary for such a source is not a sphere (as in the memoryless Gaussian case) but some N-D "ellipsoid" [3]. The codebook that realizes both gains must therefore consist of lattice points or trellis sequences that are contained inside this ellipsoid. Since there is no known direct algorithmic method to index lattice points or trellis sequences inside an arbitrary ellipsoid, such a codebook is not implementable in large dimensions where table look-up would be prohibitive.

The duality between quantization and transmission problems has been studied in [14]. Like codebooks, signal constellations for transmitting data also consist of lattice points or trellis sequences that are enclosed inside a bounding region. It was shown that quantization of memoryless Gaussian sources is the dual problem for transmitting data over memoryless additive white Gaussian noise (AWGN) channels under the average power constraint. The quantization boundary gain is analogous to the transmission *coding* gain while the quantization granular gain corresponds to the transmission *shaping* gain. Because of this duality, efficient techniques developed for one problem always have their useful counterparts in the other problem. For example, the AWGN transmission dual of TB-SVQ is the SVQ-shaped trellis coded constellation [43] which can realize both the coding and

shaping gains.

In practice, however, many important channels are not memoryless but suffer from intersymbol interference (ISI). Transmission over ISI channels, while realizing both the coding and shaping gains, was traditionally considered a hard task. Until recently the only implementable solution to this problem was the trellis precoding scheme of Eyuboğlu and Forney [44]. This is a clever scheme but it is complex to implement and is not general enough as it restricts the constellation boundary to be the Voronoi region of a trellis code. This clearly results in suboptimal shaping gain for a given delay. To solve this problem, Laroia, Tretter, and Farvardin have recently proposed another precoding scheme [31] that makes it possible to realize both coding and shaping gains over ISI channels. This scheme uses a nonlinear precoder to slightly disturb the signal constellation (designed for a memoryless channel) before transmission. This precoding scheme is simpler to implement than trellis precoding and does not place any constraint on the method of shaping. When used with the SVQ-shaped trellis coded constellation, it can realize both coding and shaping gains for transmission over ISI channels.

As will be discussed in Section 4.2, quantization of Markov sources is the dual problem for data transmission over ISI channels. This duality paves the way for employing the precoding idea [31] to the problem of quantizing Markov sources. The new quantizer described in this chapter — dubbed *precoded* TB-SVQ — is the quantization dual of the precoded SVQ-shaped trellis coded constellation for ISI channels and it can realize both granular and boundary gains for Markov sources. The granular gain is realized by the underlying trellis code while the combination of the precoder and the SVQ structure provides the boundary gain. Hence the precoder effectively extends the scope of TB-SVQ to quantizing Markov sources.

The rest of this chapter is organized as follows. The next section discusses the duality between quantizing Markov sources and transmission over ISI channels. In Section 4.3, we describe the precoding idea of Laroia *et al.* [31] and present the precoded TB-SVQ for quantization of Markov sources. In Section 4.4, we provide codebook search algorithms for the precoded TB-SVQ and analyze their implementation complexity. Two improved precoders more recently proposed by Laroia [45] are presented in Section 4.5. Simulation results will be provided in Section 4.6. Finally, a summary and conclusions are presented in Section 4.7.

4.2 Duality — Markov Sources/ISI Channels

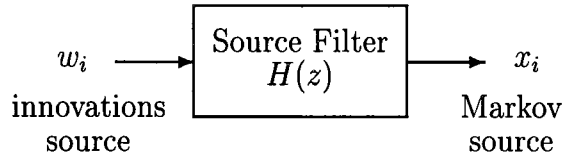
In this section, we describe the duality between quantization of Markov sources and data transmission over ISI channels. This duality can be extended to other sources and channels, but for simplicity of discussion we shall be mainly concerned with Gaussian sources and channels.

With a high probability, a vector of a large block-length from a memoryless Gaussian source lies inside a sphere — called the *source sphere* whose normalized per dimension squared-radius is equal to the source variance. When designing a quantizer for this source, the boundary gain is maximized (overload probability is minimized) if the codebook has the smallest volume and contains the source sphere. Likewise, with a high probability, the received point after an AWGN channel for a transmitted point from a large dimensional constellation lies inside a sphere — called the *noise sphere* whose normalized per dimension squared-radius is equal to the noise variance and which is centered at the transmitted point. When designing a constellation for this channel, the coding gain is maximized (error probability is

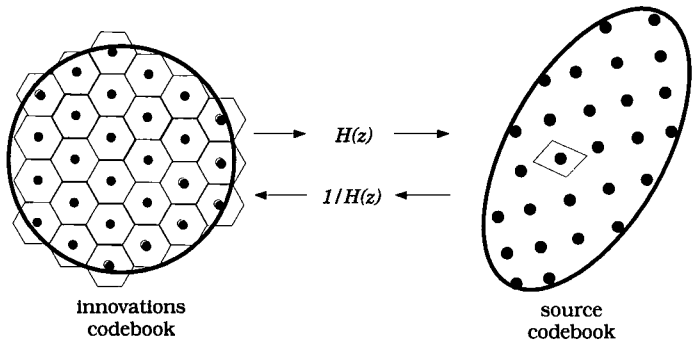
minimized) if the Voronoi cells of the constellation points are the smallest volume cells each containing its corresponding noise sphere. We hence see that the Voronoi cells of a transmission constellation play a role similar in nature to the boundary of the quantization codebook.

The average power constraint in transmission dictates that for a given power (normalized per dimension second moment) the signal constellation should have the maximum volume. The shape that has the maximum volume (highest rate) for a given second moment in N dimensions is an N-sphere. Similarly, the Voronoi cells of the quantizer codebook for the squared-error distortion measure should have the maximum volume (smallest rate) for a given second mement (distortion). Hence the shaping gain in transmission plays a role similar in nature to the granular gain in quantization. We will next study how the above analogies extend to Markov sources and ISI channels.

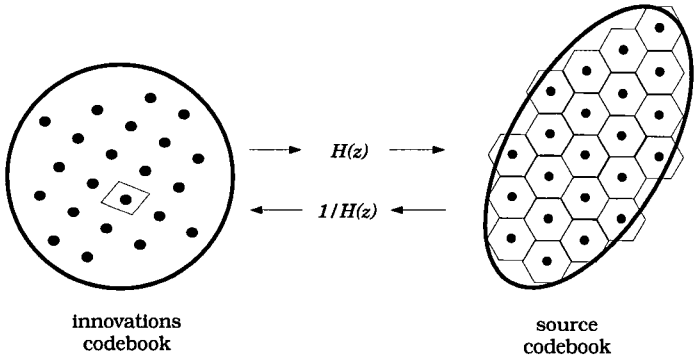
First, we consider quantization of a Markov source whose source model is shown in Figure 4.1:(a). The quantization codebooks used in the source and innovations domains will be called the *source codebook* and *innovations codebook*, respectively. The basic idea of the naive source coding scheme which extends the underlying scalar quantizer in the D*PCM to the TB-SVQ (as was described in Section 3.5) is to define the source codebook as the image under $H(z)$ of the optimal TB-SVQ innovations codebook; see Figure 4.1:(b) for a generic picture of the corresponding source and innovations codebooks. This quantizer will have the right boundary properties because it minimizes the overload probability of the source codebook. This naive quantizer, however, does not have good granular gain properties as the quantization cells of the source codebook have an arbitrary shape obtained by filtering under $H(z)$ the corresponding (approximately) spherical quantization cells



(a)



(b)



(c)

Figure 4.1: Quantization of Markov sources: (a) source model; (b) naive source and innovations codebooks; (c) ideal source and innovations codebooks.

of the innovations codebook. Since the objective of quantization is to minimize the squared-error distortion in the source domain, a good quantizer will require the quantization cells of the source codebook to be approximately spherical in order to capitalize on the granular gain. Also, to maximize the boundary gain, the corresponding innovations codebook should have a spherical boundary. Hence for Markov sources, the granular gain is determined from the source codebook while th boundary gain is determined from the innovations codebook. A generic picture of the structure of ideal source and innovations codebooks for a Markov source is presented in Figure 4.1:(c).

Next consider an ISI channel shown in Figure 4.2:(a). The transmitted sequence u_i goes through a channel filter $G(z)$ and at the output of the filter white Gaussian noise is added to the signal. To send information over the channel, binary data is used to address a point in a signal constellation which we call the *transmit constellation.* The image of the transmit constellation under the channel map $G(z)$ is called the *receive constellation.* The channel output is mapped to the nearest point in the receive constellation. The binary data is recovered by determining the address of this point. Due to the average power constraint the transmitte constellation should have a spherical boundary. The shaping gain is therefore determined by the boundary of the transmit constellation. Also, because the noise is white Gaussian and introduced in the receive constellation, the corresponding receive constellation must have Voronoi cells that cover most of the noise sphere and hence minimizes the probability of error. This is equivalent to a large minimum distance between its points (sequences). The coding gain is therefore determined by the Voronoi cells of the receive constellation. A generic picture of the ideal transmit and receive constellations structure for transmission over ISI channels is

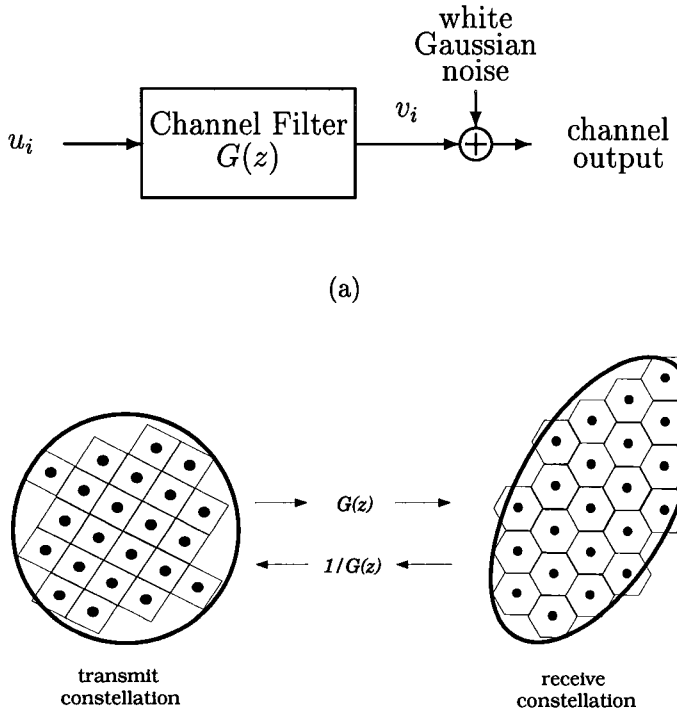


Figure 4.2: Transmission over ISI channels: (a) channel model; (b) ideal transmit and receive constellations.

presented in Figure 4.2:(b).

Now, the duality between quantizing Markov sources and transmitting over ISI channels is quite apparent. The quantization granular gain — determined from the quantization cells of the source codebook, is the dual of the transmission shaping gain — determined from the boundary of the transmit constellation. Similarly, the quantization boundary gain — determined from the boundary of the innovations codebook, is the dual of the transmission coding gain — determined from the Voronoi cells of the receive constellation.

4.3 Precoded TB-SVQ

By applying the duality between quantization and transmission problems, one can always find useful counterparts in one problem from efficient techniques developed for the other problem. For example, TB-SVQ can realize both the boundary and granular gains when quantizing memoryless sources and is the motivation for its AWGN transmission dual — the SVQ-shaped trellis coded constellation [43] which can realize the dual coding and shaping gains. Transmission over ISI channels while realizing both coding and shaping gains was traditionally considered a difficult problem, until recently when it was solved by a precoding scheme proposed by Laroia, Tretter, and Farvardin [31]. In what follows, we address the problem of combining the precoding idea with TB-SVQ for quantizing Markov sources while realizing both boundary and granular gains. For simplicity of presentation we shall be mainly concerned with Gauss-Markov sources.

Realizing a significant granular gain requires that the quantization cells of the source codebook be as spherical as possible. This can be achieved if the source codebook is a subset of sequences of some "good" trellis code which will be called the *source* trellis code. The innovations codebook thus consists of the *filtered* trellis sequences which are the images under the inverse source filter $1/H(z)$ of the source trellis sequences. Further, to realize the optimal boundary gain, the innovations codebook should have a spherical boundary. Consider an ideal quantizer whose codebook has the above desired properties. Such a codebook, however, is not easy to implement as there is no known simple algorithm to index either the filtered trellis sequences in the sphere-bounded innovations codebook or the source trellis sequences in the ellipsoid-bounded source codebook.

Suppose that we can find a transformation — called the *precoder* — that maps

the filtered trellis sequences to some source trellis sequences and possesses the following two properties:

I. The precoder disturbs the input only slightly — just enough to ensure that the output is a legitimate source trellis sequence.

II. The precoding transformation is invertible.

Redefine the innovations codebook as consisting of those filtered trellis sequences for which the corresponding precoder output sequences are contained inside a (Cartesean product of) sphere(s). Because of Property I, the boundary of this innovations codebook would still be very close to spherical, resulting in a close to optimal boundary gain. The image under the precoding transformation of the innovations codebook is called the *precoded* codebook which resembles a TB-SVQ codebook. Due to Property II, there is a one-to-one correspondence between the innovations codebook and the precoded codebook. The indexing problem is hence solved by indexing the precoded codebook based on the available TB-SVQ indexing algorithm.

Now with such a precoder available to solve the codebook indexing problem, TB-SVQ can be extended to a novel quantization scheme which will be called *precoded* TB-SVQ. We next describe the operation of this new quantization scheme whose block diagram is provided in Figure 4.3. The input source sequence $\{x_i\}$ is first quantized by the source codebook quantizer (which we will elaborate in the next section) to a preferably nearest source trellis sequence $\{\hat{x}_i\}$, which then passes through the inverse source filter $1/H(z)$ to produce the corresponding filtered trellis sequence $\{y_i\}$. The precoder then transforms $\{y_i\}$ into a TB-SVQ code-sequence $\{\hat{y}_i\}$ which is encoded into binary data (using the TB-SVQ encoding algorithm) and transmitted. The receiver decodes the received binary stream into $\{\hat{y}_i\}$. Since the

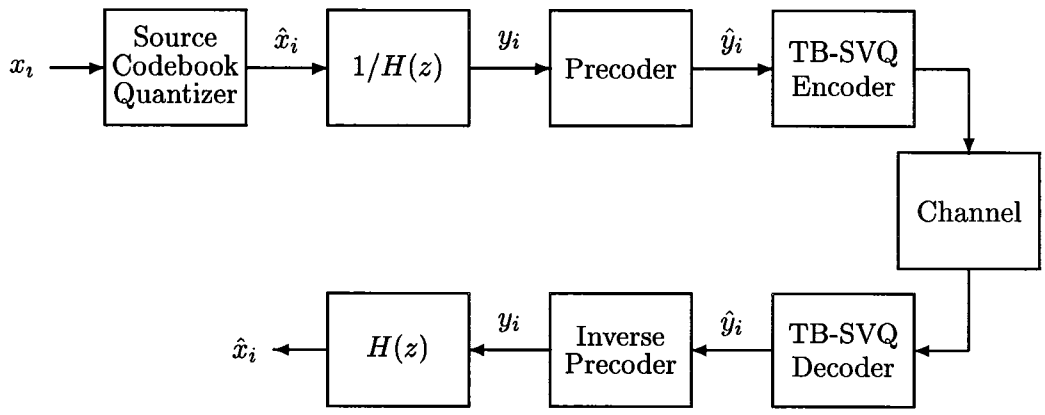


Figure 4.3: Block diagram of precoded TB-SVQ.

precoder transformation is invertible, the inverse precoder exists and can recover the filtered trellis sequence $\{y_i\}$ from $\{\hat{y}_i\}$. The quantized output $\{\hat{x}_i\}$ is hence obtained by filtering $\{y_i\}$ with the source filter $H(z)$. Since the corresponding innovations codebook has a nearly spherical boundary, the source codebook has a nearly optimal codebook boundary. Also, the source codebook has approximately spherical quantization cells. Precoded TB-SVQ can therefore realize both granular and boundary gains for Markov sources provided a suitable precoder exists.

We next show that a naive precoder indeed exists and its corresponding inverse precoder is the same as the precoder used for transmissions over ISI channels [31]. Recall that in Ungerboeck's 1-D trellis codes, the 1-D lattice translate $\boldsymbol{\Lambda} = 2\mathbf{Z} + 1$ is partitioned into four cosets of the sublattice $\boldsymbol{\Lambda}' = 8\mathbf{Z}$ — called the coset lattice. This precoder is simply a "memoryless quantizer" that maps y_i to the nearest point $\hat{y}_i$ equivalent modulo $\boldsymbol{\Lambda}'$ to $\hat{x}_i$. The corresponding quantization error — called the *precoding error* — $e_i \stackrel{\Delta}{=} \hat{y}_i - y_i$ is always in the Voronoi region $\mathcal{V} = [-4, 4)$ (neglecting the scaling factor δ) of the coset lattice. It is obvious that $\{\hat{y}_i\}$ is also

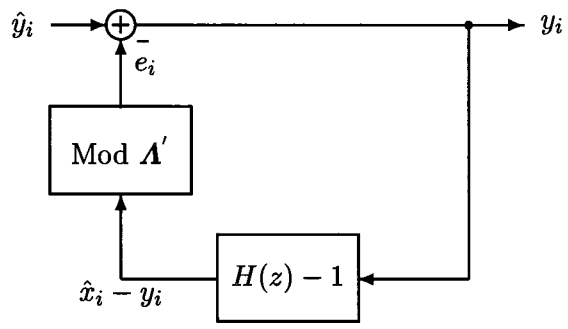


Figure 4.4: Inverse transformation of the naive precoder.

a source trellis sequence. Now let us examine if this naive precoder possesses the two properties described above. The first requirement is that the y_i and $\hat{y}_i$ be close in Euclidean distance, i.e., $E[e_i^2]$ should be much smaller than the source variance. Since e_i is always constrained to lie inside the Voronoi region $\mathcal{V}$ (multiplied by δ), this requirement will automatically be satisfied for high rate quantization when the average energy of $\mathcal{V}$ is much smaller than the source variance. Now consider the inverse precoder shown in Figure 4.4 which is the quantization dual of the ISI transmission precoder of [31]. At time instant i, the filter $H(z) - 1$ first produces $\hat{x}_i - y_i$ based on its input and output samples in the past. Since $\hat{x}_i$ and $\hat{y}_i$ are equivalent modulo Λ', a modulo operation can be performed on $\hat{x}_i - y_i$ to obtain the precoding error e_i. Since $\hat{y}_i = y_i + e_i$, y_i can hence be reconstructed by removing e_i from $\hat{y}_y$. We have therefore verified the second requirement that the precoder be an invertible transformation.

The granular and boundary structures of the precoded TB-SVQ codebook are already exactly specified by the underlying trellis code and the combination of the precoder and SVQ, respectively. The only unknown to be determined is the scaling factor δ for the underlying lattice translate Λ. The precoded TB-SVQ

design problem is hence simply to select the scaling factor δ that yields the smallest empirical quantization distortion, provided that a long training sequence and the codebook search algorithm are available.

4.4 Source Codebook Search Algorithms

We will present in this section two different approaches that can quantize the input sequence $\{x_i\}$ to the trellis sequence $\{\hat{x}_i\}$ in the source codebook. The first approach is AEP-motivated and is an extension of the TB-SVQ-II codebook search algorithm [15] to Markov sources. The second approach is dynamic programming based and assimilates the predictive TB-SVQ codebook search algorithms given in Section 3.5.

4.4.1 AEP-Motivated Approach

Suppose that the vector block-length N is sufficiently large. The AEP suggests that, with a high probability, a vector from a Gauss-Markov source lies inside an ellipsoid — called the *source ellipsoid* — which is the image under the source filter $H(z)$ of a sphere containing the corresponding innovations vector with the same probability. At high rates, the precoded TB-SVQ source codebook consists of a subset of source trellis sequences that are enclosed inside the source ellipsoid. If the input sequence $\{x_i\}$ is quantized to the nearest trellis sequence $\{\hat{x}_i\}$ by a simple trellis search algorithm (just as a TCQ [21] but in the unbounded space), the AEP tells us that almost all contiguous N-vectors from $\{\hat{x}_i\}$ should lie inside the source ellipsoid. In case of an overload event, we can repeatedly shift the corresponding source N-vector toward the source ellipsoid and apply the TCQ search again until

the quantized N-vector is inside the source ellipsoid. At high rates, overload event rarely occurs and the codebook search is hence as simple as that of a TCQ.

The source sequence $\{x_i\}$ is segmented into contiguous blocks of N-vectors, yielding $\{\boldsymbol{x}_k\}$ where $\boldsymbol{x}_k \equiv (x_{k,1}, x_{k,2}, \cdots, x_{k,N})$. Our objective is to quantize $\{\boldsymbol{x}_k\}$ to the nearest source trellis sequence whose corresponding precoder output $\{\hat{\boldsymbol{y}}_k\}$ is a legitimate TB-SVQ sequence. Denote by Δ_k^s the minimum accumulated distortion that results when the first k source vectors are mapped to a TB-SVQ sequence whose final trellis state is $s \in \boldsymbol{\Sigma}$. Suppose that the quantizer assumes a fixed delay of d N-vectors. To handle this quantization delay, we need a buffer of $(d+1)$ N-vectors $\boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k,0}^s, \boldsymbol{q}_{k,1}^s, \cdots, \boldsymbol{q}_{k,d}^s)$ for each trellis state s that stores the unreleased portion of the trellis sequence resulting in Δ_k^s. Suppose that $k \geq d$ and denote by s^* the trellis state that minimizes Δ_k^s over $\boldsymbol{\Sigma}$, $\boldsymbol{q}_{k,0}^{s^*}$ is hence released as the TB-SVQ code-vector $\hat{\boldsymbol{y}}_{k-d}$. To make the released sequence of N-vectors consistent with the underlying trellis code, we can employ another buffer $\boldsymbol{\sigma}_k^s \equiv (\sigma_{k,0}^s, \sigma_{k,1}^s, \cdots, \sigma_{k,d}^s)$ associated with each $\boldsymbol{q}_k^s$ where $\sigma_{k,j}^s \in \boldsymbol{\Sigma}$ denotes the trellis state that $\boldsymbol{q}_{k,j}^s$ reaches, $\forall j \in \boldsymbol{\mathcal{J}}_{d+1}$. All TB-SVQ code-vector buffers whose $\sigma_{k,0}^s$ is different from $\sigma_{k,0}^{s^*}$ will result in released sequence inconsistent with the underlying trellis code and should therefore be removed from further consideration.

Suppose that we have just determined Δ_{k-1}^s (and hence $\boldsymbol{q}_{k-1}^s$ and $\boldsymbol{\sigma}_{k-1}^s$) for all $s \in \boldsymbol{\Sigma}$, now consider operating on the k^{th} source vector $\boldsymbol{x}_k$. Denote by D_i^s the minimum accumulated distortion that results when the first i components of $\boldsymbol{x}_k$ are quantized to a trellis sequence whose final trellis state is s; the most recent $p+1$ quantized samples associated with this trellis sequence are stored in $\hat{\boldsymbol{x}}_i^s \equiv (\hat{x}_{i-p}^s, \hat{x}_{i-1}^s, \cdots, \hat{x}_i^s)$. Also, associated with the trellis sequence resulting in D_i^s, let the corresponding precoder output and initial trellis state (at $i = 0$) be

denoted by $\hat{\boldsymbol{y}}_i^s \equiv (\hat{y}_1^s, \hat{y}_2^s, \cdots, \hat{y}_i^s)$[1] and t_i^s, respectively. We repeat the following procedure until there is at least one $\hat{\boldsymbol{y}}_N^s$ that is a TB-SVQ code-vector. First, D_0^s is initialized to the value of Δ_{k-1}^s, $\forall s \in \boldsymbol{\Sigma}$. Suppose that D_i^s, $\hat{\boldsymbol{x}}_i^s$, $\hat{\boldsymbol{y}}_i^s$, and t_i^s are already determined. Given a trellis state $s \in \boldsymbol{\Sigma}$ and one, say s', of its two possible previous states, let Q^* denote a function that determines for the input sample x the reproduction symbol $\hat{x}$ in the allowed coset as dictated by s and s'; that is,

$$\hat{x} = Q^*(x, s', s). \tag{4.1}$$

To determine $\hat{x}_{i+1}^s$, we need to solve the minimization problem:

$$s^* = \arg\min_{s'} \left[D_i^{s'} + [x_{k,i+1} - Q^*(x_{k,i+1}, s', s)]^2 \right], \tag{4.2}$$

by which byproducts such as $\hat{\boldsymbol{x}}_{i+1}^s$, D_{i+1}^s and t_{i+1}^s can also be obtained. The inverse source filter output, denoted by y, is then computed according to

$$y = \hat{x}_{i+1}^s - \sum_{j=1}^{p} \rho_j \hat{x}_{i+1-j}^s. \tag{4.3}$$

The corresponding precoder output $\hat{y}_{i+1}^s$ is obtained simply by quantizing y to the closest point that is in the same coset lattice as is $\hat{x}_{i+1}^s$; that is,

$$\hat{y}_{i+1}^s = Q^*(y, s^*, s). \tag{4.4}$$

The above procedure repeats until we get to $i = N$. Should there be no $\hat{y}_N^s$ that is a legitimate TB-SVQ code-vector, we say that the source vector $\boldsymbol{x}_k$ lies upon the surface of the so-called *overload ellipsoid.* We then slightly disturb $\boldsymbol{x}_k$ along the gradient vector (at $\boldsymbol{x}_k$) of the overload ellipsoid surface (in an attempt to move it toward or into the source ellipsoid) and start the search procedure again. Once

[1]It should be mentioned that $\hat{\boldsymbol{y}}_i^s$ is a local variable within the k^{th} search cycle and should not be confused with the global variable $\hat{\boldsymbol{y}}_k$.

there is at least one $\hat{y}_N^s$ that is a legitimate TB-SVQ code-vector, the value of D_N^s is copied to Δ_k^s for the next search cycle. The begining trellis state for $\hat{\boldsymbol{y}}_N^s$ is t^s, based on which the buffers $\boldsymbol{q}_k^s$ and $\boldsymbol{\sigma}_k^s$ are updated as $\boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k-1}^{t^s}, \hat{\boldsymbol{y}}_N^s)$ and $\boldsymbol{\sigma}_k^s \equiv (\boldsymbol{\sigma}_{k-1}^{t^s}, s)$, respectively. This AEP-motivated precoded TB-SVQ codebook search algorithm is summarized in Figure 4.5.

For a sufficiently large vector block-length and rate, the AEP suggests that the overload events occur only insignificantly. In this case, it takes approximately the computational complexity of an unbounded TCQ trellis search algorithm to search for the reproduction sequence, which is $16+3\cdot 2^{\nu}$ operations/sample. Some other extra computational requirements arise for the inverse source filtering and precoding. For each search path, it takes $2p$ operations to compute the inverse source filter output. The precoding operation then takes two more operations (one for shifting and the other for scaling) to quantize the the inverse source filter output to the nearest point in a specified translate of the coset lattice. The overall computational complexity is hence $16+(5+2p)\cdot 2^{\nu}$ operations/sample. We omit the storage requirement analysis for the AEP-motivated precoded TB-SVQ codebook search algorithm here but give the result in Table 4.1.

Computational (Operations/Sample)	Storage (32-bit Words)
$16+(5+2p)2^{\nu}$	$3\cdot 2^{\nu}+2(p+1)+\frac{N(d+2)\lceil \log_2 m\rceil+\nu[d+1+2^{\nu}]}{16}$

Table 4.1: AEP-motivated precoded TB-SVQ codebook search complexity.

for $(s \in \boldsymbol{\Sigma})$ if $(s ==$ a specified trellis state$)$ $\Delta_{-1}^{s} = 0$; else $\Delta_{-1}^{s} = \infty$;
for $(k = 0;\ k < \infty;\ k++)$ {

Searched_Flag $= 0$;
while (Searched_Flag==0) {

for $(s \in \boldsymbol{\Sigma})$ $D_0^s = \Delta_{k-1}^s$;
for $(i = 0;\ i < N;\ i++)$ for $(s \in \boldsymbol{\Sigma})$ {

$s^* = \arg\min_{s'} \left[D_i^{s'} + [x_{k,i+1} - Q^*(x_{k,i+1}, s', s)]^2 \right]$;
$\hat{x} = Q^*(x_{k,i+1}, s^*, s)$;
$D_{i+1}^s = D_i^{s^*} + (x_{k,i+1} - \hat{x})^2$;
$\hat{\boldsymbol{x}}_{i+1}^s \equiv \left(\hat{\boldsymbol{x}}_i^{s^*}, \hat{x} \right)$;
$y = \hat{x} - \sum_{j=1}^{p} \rho_j \hat{x}_{i+1-j}^s$;
$\hat{\boldsymbol{y}}_{i+1}^s \equiv \left(\hat{\boldsymbol{y}}_i^{s^*}, Q^*(y, s^*, s) \right)$;
if $(i == 0)$ $t_{i+1}^s = s^*$; else $t_{i+1}^s = t_i^{s^*}$;

}
for $(s \in \boldsymbol{\Sigma})$ if $(\hat{\boldsymbol{y}}_N^s$ is a TB-SVQ code-vector) Searched_Flag $= 1$;
if (Searched_Flag==0) shift $\boldsymbol{x}_k$ toward the codebook boundary;

}
for $(s \in \boldsymbol{\Sigma})$ { $\Delta_k^s = D_N^s$; $\quad \boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k-1}^{t_N^s}, \hat{\boldsymbol{y}}_N^s)$; $\quad \boldsymbol{\sigma}_k^s \equiv (\boldsymbol{\sigma}_{k-1}^{t_N^s}, s)$; }
if $(k \geq d)$ {

$s^* = \arg\min_{s \in \boldsymbol{\Sigma}} \Delta_k^s$;
release $\hat{\boldsymbol{y}}_{k-d} = \boldsymbol{q}_{k,0}^{s^*}$ for TB-SVQ code-vector encoding;
for $(s \in \boldsymbol{\Sigma})$ {

if $(\sigma_{k,0}^s \neq \sigma_{k,0}^{s^*})$ $\Delta_k^s = \infty$;
$\boldsymbol{\sigma}_k^s \equiv (\sigma_{k,1}^s, \sigma_{k,2}^s, \cdots, \sigma_{k,d}^s)$;
$\boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k,1}^s, \boldsymbol{q}_{k,2}^s, \cdots, \boldsymbol{q}_{k,d}^s)$;

}

}

}

Figure 4.5: AEP-motivated precoded TB-SVQ codebook search algorithm.

4.4.2 Dynamic Programming Based Approach

The AEP-motivated precoded TB-SVQ codebook search algorithm generally works very well when both vector block-length and encoding rate are large. Like the search algorithm for TB-SVQ-II [15], however, when the encoding rate is low overload events occur much more often and the actual computational complexity accordingly increases. Besides, the number of operations required to produce the quantized vector varies largely depending on the input source vector. In this case for memoryless sources the dynamic programming based TB-SVQ codebook search algorithm presented in Section 3.4 should be more useful than the AEP-motivated counterpart (TB-SVQ-II) because (i) fixed number of operations can be expected to determine the quantized vector; and (ii) its implementation complexity is quite affordable. Here we consider this type of dynamic programming based codebook search algorithm for precoded TB-SVQ.

In what follows, we describe the full-state search algorithm and omit the simpler state-suppressed search algorithm. The variables $\boldsymbol{x}_k$, $\hat{\boldsymbol{y}}_k$, Δ_k^s, d, $\boldsymbol{q}_k^s$, and $\boldsymbol{\sigma}_k^s$ used in the AEP-motivated search algorithm will stand for the same variables here; the reader is referred to Section 4.4.1 for their specific meanings. Also, how $\boldsymbol{q}_k^s$ and $\boldsymbol{\sigma}_k^s$ operate according to the delay d is exactly the same as the AEP-motivated algorithm and will be omitted, too.

Suppose that we have just determined Δ_{k-1}^s, $\boldsymbol{q}_{k-1}^s$, and $\boldsymbol{\sigma}_{k-1}^s$ for all $s \in \boldsymbol{\Sigma}$, now consider operating on $\boldsymbol{x}_k$. Denote by $D_i^{l,s}$ the minimum accumulated distortion that results when the first i components of $\boldsymbol{x}_k$ are quantized to $\hat{\boldsymbol{x}}_i^{l,s}$ whose corresponding precoder output $\hat{\boldsymbol{y}}_i^{l,s}$ reaches trellis state $s \in \boldsymbol{\Sigma}$ and has a total length $l \in \boldsymbol{\mathcal{J}}_{L+1}$. Denote by $t_i^{l,s}$ the initial ($i = 0$) trellis state for $\hat{\boldsymbol{y}}_i^{l,s}$. Suppose that $D_i^{l,s}$, $\hat{\boldsymbol{x}}_i^{l,s}$, $\hat{\boldsymbol{y}}_i^{l,s}$, and $t_i^{l,s}$ are already determined. We next describe how to determine $\hat{\boldsymbol{y}}_{i+1}^{l,s}$ for

all $l \in \mathcal{J}_{L+1}$ and $s \in \mathbf{\Sigma}$. Consider for $\hat{\boldsymbol{y}}_{i+1}^{l,s}$ a candidate $(i+1)$-vector which is the concatenation of the previous i-vectors $\hat{\boldsymbol{y}}_i^{l',s'}$ and one TB-SVQ symbol $\hat{y}_{i+1}^{l,s}$ satisfying the state transition and length accumulation constraints; that is,

$$\hat{y}_{i+1}^{l,s} = \mathcal{F}(s', l - l'). \tag{4.5}$$

Since the precoder output $\hat{y}_{i+1}^{l,s}$ is equivalent modulo the coset lattice $\mathbf{\Lambda}'$ to $\hat{x}_{i+1}^{l,s}$ — the actual quantization replica for $x_{k,i+1}$, we have

$$\hat{y}_{i+1}^{l,s} = \hat{x}_{i+1}^{l,s} - z \tag{4.6}$$

for some $z \in \mathbf{\Lambda}'$. On the other hand, the corresponding input to the precoder is $\hat{x}_{i+1}^{l,s} - \tilde{x}_{i+1}^{l',s'}$ where

$$\tilde{x}_{i+1}^{l',s'} = \sum_{j=1}^{p} \rho_j \hat{x}_{i+1-j}^{l',s'} \tag{4.7}$$

is a prediction of $x_{k,i+1}$ based on $\hat{\boldsymbol{x}}_i^{l',s'}$. The coset lattice point z hence depends only on $\hat{\boldsymbol{x}}_i^{l',s'}$ — now written as $z^{l',s'}$ — and can be obtained by quantizing $\tilde{x}_{i+1}^{l',s'}$ to the nearest point in $\mathbf{\Lambda}'$. The corresponding reproduction symbol for $x_{k,i+1}$ is $\hat{y}_{i+1}^{l,s} + z^{l',s'}$ and the resulting accumulated distortion is hence

$$D_{i+1}^{l,s} = D_i^{l',s'} + [x_{k,i+1} - \hat{y}_{i+1}^{l,s} - z^{l',s'}]^2. \tag{4.8}$$

To determine $\hat{\boldsymbol{y}}_{i+1}^{l,s}$, we need to solve for the optimal length l^* and state s^* (who can be determined by $\mathcal{P}(s, l - l^*)$) that minimizes the accumulated distortion $D_{i+1}^{l,s}$; that is,

$$(l^*, s^*) = \arg\min_{(l',s')} \left[D_i^{l',s'} + [x_{k,i+1} - z^{l',s'} - \mathcal{F}(s', l - l')]^2 \right]. \tag{4.9}$$

Now suppose that $D_N^{l,s}$, $\hat{\boldsymbol{x}}_N^{l,s}$, $\hat{\boldsymbol{y}}_N^{l,s}$, and $t_N^{l,s}$ are determined. For each $s \in \mathbf{\Sigma}$, we search for l^* that minimizes $D_N^{l,s}$. Δ_k^s is thereby set to the value of $D_N^{l^*,s}$.

Buffers $\boldsymbol{q}_k^s$ and $\boldsymbol{\sigma}_k^s$ are updated according to $\boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k-1}^{t_N^{l^*,s}}, \hat{\boldsymbol{y}}_N^{l^*,s})$ and $\boldsymbol{\sigma}_k^s \equiv (\boldsymbol{\sigma}_{k-1}^{t_N^{l^*,s}}, s)$, respectively. Also, the most recent p samples of $\hat{\boldsymbol{x}}_N^{l^*,s}$ are copied to $\hat{\boldsymbol{x}}_0^{0,s}$ that will be used by the inverse source filter in the next search cycle.

Summarized in Figure 4.6 is the full-state precoded TB-SVQ codebook search algorithm, which is very similar to that for predictive TB-SVQ presented in Section 3.5. In predictive TB-SVQ, the DPCM operation is performed in each search path. While in precoded TB-SVQ, we pathwise perform the inverse source filtering and precoding transformation. The implementation complexity of precoded TB-SVQ under the dynamic programming based search algorithm is about the same as that for predictive TB-SVQ. We will omit the complexity analysis here but refer the reader to the result given in Section 3.5.

4.5 Further Reduction in Precoding Error

The precoding error is the quantization dual of the so-called *precoding loss* [31] in data transmission. Just like the precoding loss will disturb the boundary of the transmit constellation [31], the precoding error will disturb the innovations codebook boundary. The precoding error (loss), if significantly large, could reduce the boundary (shaping) gain and hence should be kept as small as possible. In this section two precoding schemes more recently developed by Laroia [45] that can further reduce the transmission precoding loss are described. These new precoding ideas can also be used to reduce the quantization precoding error. These modified precoders were described based on Ungerboeck's 2-D trellis codes in [45]. For simplicity and continuity of presentation, we shall use Ungerboeck's 1-D trellis codes.

for $(s \in \boldsymbol{\Sigma})$ if $(s ==$ a specified trellis state$)$ $\Delta_{-1}^{s} = 0$; else $\Delta_{-1}^{s} = \infty$;
for $(k = 0;\ k < \infty;\ k++)$ {

for $(s \in \boldsymbol{\Sigma};\ l \in \boldsymbol{\mathcal{J}}_{L+1})$ if $(l == 0)$ $D_0^{l,s} = \Delta_{k-1}^{s}$; else $D_0^{l,s} = \infty$;
for $(i = 0;\ i < N;\ i++)$ {

for $(s \in \boldsymbol{\Sigma};\ l \in \boldsymbol{\mathcal{J}}_{L+1})$ $z^{l,s}$ = nearest point in $\boldsymbol{\Lambda}'$ to $\sum_{j=1}^{p} \rho_j \hat{x}_{i-j-1}^{l,s}$;
for $(s \in \boldsymbol{\Sigma};\ l \in \boldsymbol{\mathcal{J}}_{L+1})$ {

$(l^*, s^*) = \arg\min_{(l',s')} \left[D_i^{l',s'} + [x_{k,i+1} - z^{l',s'} - \mathcal{F}(s', l - l')]^2 \right]$;
$\hat{y}^{l,s} = \mathcal{F}(s^*, l - l^*)$;
$D_{i+1}^{l,s} = D_i^{l^*,s^*} + (x_{k,i+1} - z^{l^*,s^*} - \hat{y}^{l,s})^2$;
$\hat{\boldsymbol{y}}_{i+1}^{l,s} \equiv \left(\hat{\boldsymbol{y}}_i^{l^*,s^*}, \hat{y}^{l,s} \right)$;
$\hat{\boldsymbol{x}}_{i+1}^{l,s} \equiv \left(\hat{\boldsymbol{x}}_i^{l^*,s^*}, z^{l^*,s^*} + \hat{y}^{l,s} \right)$;
if $(i == 0)$ $t_{i+1}^{l,s} = s^*$; else $t_{i+1}^{l,s} = t_i^{l^*,s^*}$;

}

}
for $(s \in \boldsymbol{\Sigma})$ {

$l^* = \arg\min_{l \in \boldsymbol{\mathcal{J}}_{L+1}} D_N^{l,s}$;

$\Delta_k^s = D_N^{l^*,s}$; $\quad \boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k-1}^{t_N^{l^*,s}}, \hat{\boldsymbol{y}}_N^{l^*,s})$; $\quad \boldsymbol{\sigma}_k^s \equiv (\boldsymbol{\sigma}_{k-1}^{t_N^{l^*,s}}, s)$;
$\hat{\boldsymbol{x}}_0^{0,s} = \hat{\boldsymbol{x}}_N^{l^*,s}$;

}
if $(k \geq d)$ {

$s^* = \arg\min_{s \in \boldsymbol{\Sigma}} \Delta_k^s$;
release $\hat{\boldsymbol{y}}_{k-d} = \boldsymbol{q}_{k,0}^{s^*}$ for TB-SVQ code-vector encoding;
for $(s \in \boldsymbol{\Sigma})$ {

if $(\sigma_{k,0}^s \neq \sigma_{k,0}^{s^*})$ $\Delta_k^s = \infty$;
$\boldsymbol{\sigma}_k^s \equiv (\sigma_{k,1}^s, \sigma_{k,2}^s, \cdots, \sigma_{k,d}^s)$;
$\boldsymbol{q}_k^s \equiv (\boldsymbol{q}_{k,1}^s, \boldsymbol{q}_{k,2}^s, \cdots, \boldsymbol{q}_{k,d}^s)$;

}

}

}

Figure 4.6: Full-state precoded TB-SVQ codebook search algorithm.

4.5.1 Modified-I Precoder

The inverse precoder of this new precoding scheme, called the Modified-I precoder, is the quantization dual of the so-called ISI coder of Laroia [45]. Compared to the naive precoder described in Section 4.3, it can further reduce the precoding error by a factor equal to two.

Let us now denote by $\Lambda' = 4\mathbf{Z}$ — the sublattice of Λ with a partition order of two. In Figure 2.3, the supersets $\mathcal{A}$ and $\mathcal{B}$ are both translates of Λ'. This new precoder is simply another "memoryless quantizer" that maps y_i to the nearest point $\hat{y}_i$ in the pre-specified superset $\mathcal{A}$ (or $\mathcal{B}$). The corresponding precoding error $e_i = \hat{y}_i - y_i$ is always in the Voronoi region $\mathcal{V} = [-2, 2)$ of Λ'. It should be mentioned that the sequence $\{\hat{y}_i\}$ is not congruent to $\{\hat{x}_i\}$. Generally speaking, $\{\hat{y}_i\}$ may even not be a source trellis sequence. But the TB-SVQ coder/decoder pair can still be applied here to transmit $\{\hat{y}_i\}$, except that the decoded sequence are equivalent to $\{\hat{y}_i\}$ only in magnitude. Because we know which superset ($\mathcal{A}$ or $\mathcal{B}$) was used to quantize y_i, the magnitude information of the TB-SVQ decoded sequence suffices to exactly recover $\{\hat{y}_i\}$. It can be shown that the inverse precoding operation is identical to that of the naive precoder (see Figure 4.4), except that now the lattice Λ' is twice as much denser and the Voronoi region $\mathcal{V}$ of Λ' (which confines the precoding error) is reduced by a factor equal to two.

4.5.2 Modified-II Precoder

This new precoder, called the Modified-II precoder, is motivated by the modified ISI coder of Laroia [45]. Compared to the Modified-I precoder, it can further reduce the precoding error by another factor equal to two. We next describe this more sophisticated precoding scheme.

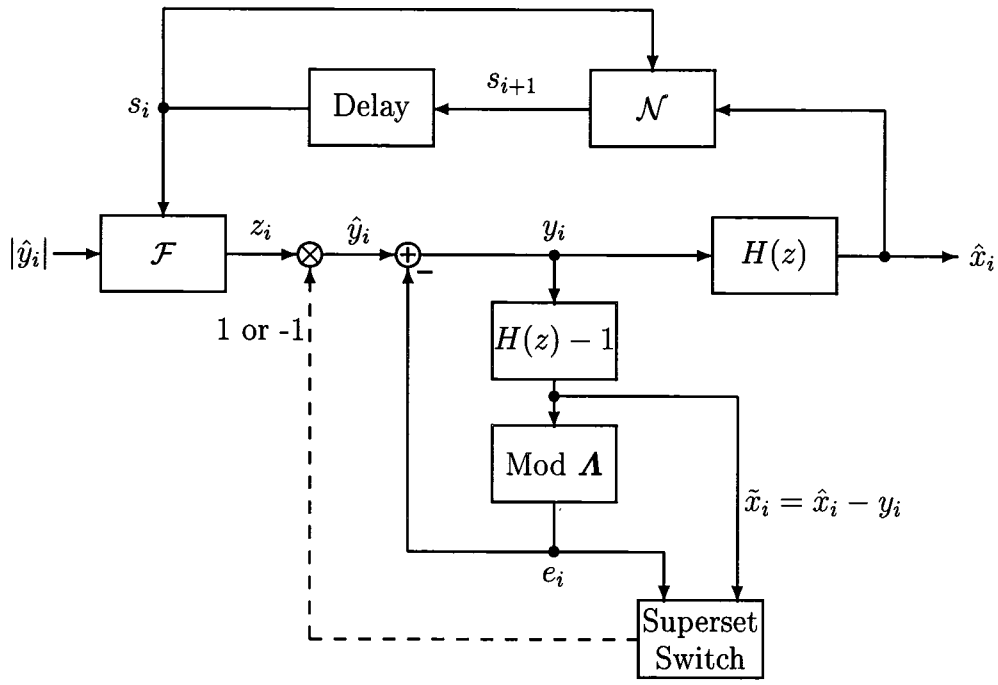


Figure 4.7: Inverse transformation of the modified precoder.

The Modified-II precoder is also a memoryless quantizer that maps y_i to the nearest point $\hat{y}_i$ in $\boldsymbol{\Lambda}$. The precoding error $e_i = \hat{y}_i - y_i$ hence lies inside the Voronoi region $\mathcal{V} = [-1, 1)$ of $\boldsymbol{\Lambda}$. Like the Modified-I precoder, the precoder output sequence $\{\hat{y}_i\}$ may not be a source trellis sequence, but we still apply the TB-SVQ encoder to encode the corresponding magnitude sequence into a binary stream. The TB-SVQ decoded sequence is generally different from $\{\hat{y}_i\}$ but has an identical magnitude sequence as that corresponding to $\{\hat{y}_i\}$. We hence need to show that there exists an inverse precoding transformation that is able to reconstruct the precoder input $\{y_i\}$ from the decoded magnitude sequence $\{|\hat{y}_i|\}$.

The block diagram of such an inverse precoder is presented in Figure 4.7, in which the source filter $H(z)$ is also shown following the inverse precoder to produce the quantized output $\{\hat{x}_i\}$. It should be mentioned that $y_i = \hat{x}_i - \tilde{x}_i$ where $\tilde{x}_i$ is

a prediction of x_i based on past samples of $\{\hat{x}_i\}$. We therefore have the following useful identity

$$\hat{y}_i = \hat{x}_i - \tilde{x}_i + e_i. \tag{4.10}$$

At time instant i, the filter $H(z) - 1$ first operates on past samples of $\{y_i\}$ and yields $\tilde{x}_i = \hat{x}_i - y_i$. Since $\hat{x}_i \in \boldsymbol{\Lambda}$, we can apply a modulo $\boldsymbol{\Lambda}$ filter to operate on $\tilde{x}_i$ and produce the precoding error e_i. For simplicity of discussion, let us neglect the lattice scaling factor δ. Since both $\hat{y}_i$ and $\hat{x}_i$ are points in $\boldsymbol{\Lambda}$, the identity in (4.10) tells us that $\tilde{x}_i - e_i$ should be $2k$ for some $k \in \mathbf{Z}$. Besides, if k is even, $\hat{y}_i$ and $\hat{x}_i$ must belong to the same superset ($\mathcal{A}$ or $\mathcal{B}$) which is dictated by the current trellis state s_i. Otherwise, $\hat{y}_i$ and $\hat{x}_i$ belong to different supersets. In either case, the state-dependent function $\mathcal{F}$ given in (2.18) can be used to obtain a candidate point $z_i \in \boldsymbol{\Lambda}$ that is in the same superset as is $\hat{x}_i$ and has the same magnitude as $\hat{y}_i$. If k is even, z_i is equal to $\hat{y}_i$, or otherwise it is equal to $-\hat{y}_i$. The box labeled "Superset Switch" determines this condition and thereby controls if z_i should be switched to the dual point of the identical magnitude in the other superset to yield $\hat{y}_i$. Now we have $\hat{y}_i = y_i + e_i$ and e_i available, y_i can be obtained. Accordingly, the output of the source filter yields $\hat{x}_i$ and we can use the next-state function $\mathcal{N}$ given in (2.19) to update the trellis state at next time instant.

4.6 Simulation Results

The precoded TB-SVQ performance are presented and discussed in this section. The results reported here are given as signal-to-noise ratio (SNR) in dB and are obtained based on simulations performed on at least 160,000 source samples. We will specifically consider two types of Gauss-Markov sources: one is a 1^{st}-order

Trellis States (2^ν)	Block-Length (N)	Rate (r)	Precoded TB-SVQ Precoder Version			Predictive TB-SVQ	$R(D)$
			Naive	Mod.-I	Mod.-II		
4	32	2.0	16.92	18.15	18.36	18.40	19.25
		3.0	23.75	24.00	24.02	24.13	25.27
		4.0	29.54	29.59	29.59	—	31.29
	64	2.0	16.77	18.17	18.42	18.63	
		3.0	23.93	24.21	24.26	24.41	
		4.0	29.85	29.91	29.92	—	
8	32	2.0	17.24	18.42	18.57	18.46	
		3.0	23.88	24.11	24.12	24.17	
		4.0	29.60	29.64	29.64	—	
	64	2.0	17.12	18.44	18.60	18.68	
		3.0	24.10	24.33	24.36	24.46	
		4.0	29.94	29.98	29.99	—	

Table 4.2: Performance (SNR in dB) of precoded TB-SVQ using the AEP-motivated codebook search algorithm with a delay of five N-vectors on encoding a 1^{st}-order Gauss-Markov source ($\rho_1 = 0.9$).

source with correlation coefficient $\rho_1 = 0.9$ while the other is a 2^{nd}-order source with correlation coefficients $\rho_1 = 1.515$ and $\rho_2 = -0.752$.

Presented in Table 4.2 is the precoded TB-SVQ performance on encoding the 1^{st}-order Gauss-Markov source using the AEP-motivated search algorithm with a quantization delay of five N-vectors; for comparison, the rate-distortion limit and the predictive TB-SVQ performance using the full-state search algorithm with the same quantization delay are also included. The results presented indicate that the precoder with a smaller precoding error does provide a better overall system performance. Generally speaking, the precode TB-SVQ performance can be improved by increasing the block-length N or the number of trellis states 2^ν. There exists, however, the example in which with the naive precoder and a 4-state trellis

Trellis States (2^ν)	Block-Length (N)	Rate (r)	Precoded TB-SVQ			Predictive TB-SVQ	$R(D)$
			Precoder Version				
			Naive	Mod.-I	Mod.-II		
4	32	2.0	18.92	20.18	20.47	20.36	21.64
		3.0	26.26	26.47	26.52	26.41	27.66
		4.0	32.06	32.10	32.11	—	33.68
	64	2.0	18.65	20.20	20.52	20.56	
		3.0	26.39	26.68	26.75	26.72	
		4.0	32.42	32.49	32.50	—	
8	32	2.0	19.43	20.53	20.72	20.49	
		3.0	26.48	26.62	26.65	26.50	
		4.0	32.16	32.17	32.17	—	
	64	2.0	19.15	20.54	20.79	20.71	
		3.0	26.63	26.84	26.89	26.79	
		4.0	32.54	32.58	32.59	—	

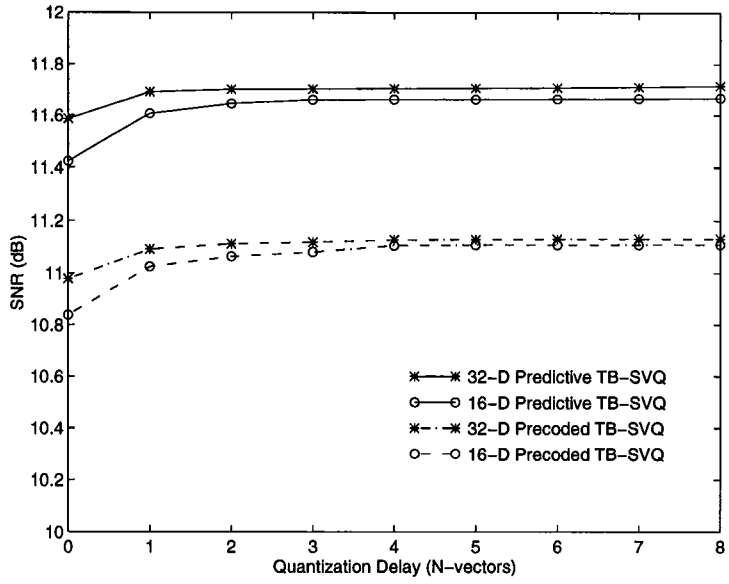
Table 4.3: Performance (SNR in dB) of precoded TB-SVQ using the AEP-motivated codebook search algorithm with a delay of five N-vectors on encoding a 2^{nd}-order Gauss-Markov source ($\rho_1 = 1.515$, $\rho_2 = -0.752$).

the 32-D quantizer outperforms the 64-D counterpart at 2 bits/sample. This could happen because the precoding error disturbs the optimal codebook boundary and effects the boundary gain. In all cases we considered, predictive TB-SVQ slightly outperforms precoded TB-SVQ based on the Modified-II precoding scheme. But we should mention that precoded TB-SVQ is much simpler to implement than predictive TB-SVQ in all these cases. Similar simulation results for the 2^{nd}-order Gauss-Markov source are provided in Table 4.3. All the previous discussions on Table 4.2 seems still valid for Table 4.3 except that the simpler precoded TB-SVQ (based on the Modified-II precoder) now slightly outperforms the more complex predictive TB-SVQ.

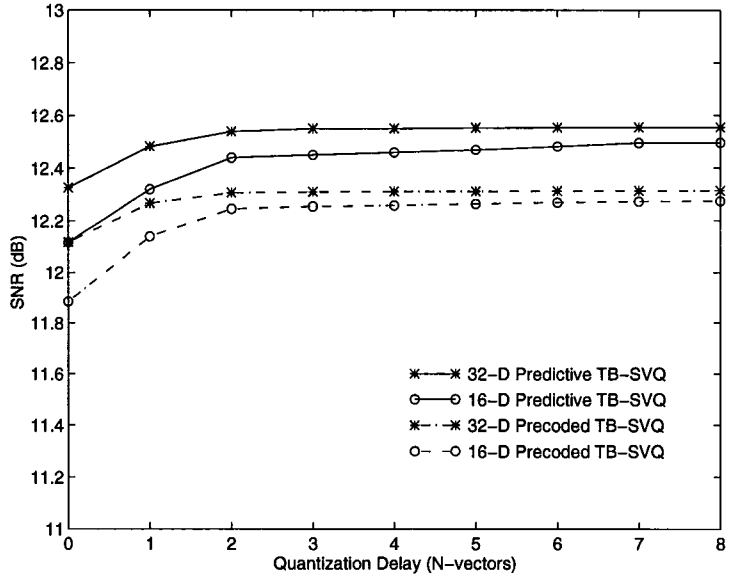
In what follows, we will implicitly assume that precoded TB-SVQ is based on

the Modified-II precoding scheme as it provides the smallest precoding error. The variation with the quantization delay of the precoded TB-SVQ performance at 1 bit/sample for the two specified Gauss-Markov sources is presented in Figure 4.8; for comparison, we have also included the predictive TB-SVQ performance. These plots indicate that precoded TB-SVQ, like predictive TB-SVQ, requires only a delay of one (16-D or 32-D) source vector to perform within 0.05 dB of the saturated (arbitrarily long delay) system performance. Also, at such an encoding rate as low as 1 bit/sample, predictive TB-SVQ consistently outperforms precoded TB-SVQ by about 0.6 dB for the 1^{st}-order Gauss-Markov source and, to a lesser extent, by about 0.2 dB for the 2^{nd}-order Gauss-Markov source.

The precoded TB-SVQ performance obtained using the dynamic programming based (full-state or state-suppressed) search algorithm with various block-lengths and rates are provided in Table 4.4. At rates 2 and 3 bits/sample, the results obtained by the full-state search algorithm are somewhat inferior to those given in Tables 4.2 and 4.3 (obtained by the AEP-motivated search algorithm). This probably has to do with the suboptimality of the dynamic programming based search algorithm, which is due to the greedy nature in selecting the survivor search path. The simulation results in Table 4.4 also suggest that the state-suppressed search algorithm results in an SNR performance loss of about 0.1 to 0.4 dB; this performance loss is generally smaller at a higher encoding rate or with a smaller block-length. Like predictive TB-SVQ, this performance loss is paid off by a reduction in the implementation complexity at a factor equal to the number of trellis states.



(a) $\rho_1 = 0.9$.



(b) $\rho_1 = 1.515$, $\rho_2 = -0.752$.

Figure 4.8: Performance (SNR in dB) versus quantization delay of precoded TB-SVQ using the full-state search algorithm on encoding a (a) 1^{th}- (b) 2^{nd}-order Gauss-Markov source; the encoding rate is 1 bits/sample.

Source Type	Rate	Full-State			State-Suppressed			PTCQ	$R(D)$
		$N = 16$	$N = 32$	$N = 64$	$N = 16$	$N = 32$	$N = 64$		
	1.0	10.84	10.98	11.48	10.40	10.40	10.93	11.19	13.23
AR(1):	2.0	17.65	18.07	18.31	17.42	17.82	17.99	17.21	19.25
$\rho_1 = 0.9$	3.0	23.40	23.83	24.10	23.27	23.71	23.96	22.92	25.27
AR(2):	1.0	11.89	12.12	12.73	11.23	11.34	12.07	11.58	14.96
$\rho_1 = 1.515$	2.0	19.75	20.24	20.48	19.55	20.03	20.31	18.92	21.64
$\rho_2 = -0.752$	3.0	25.86	26.35	26.61	25.78	26.27	26.54	25.00	27.66

Table 4.4: Precoded TB-SVQ performance (SNR in dB) using dynamic programming based algorithm with no extra quantization delay on encoding two specified Gauss-Markov sources.

4.7 Summary and Conclusions

In this chapter we discussed the duality between quantizing Markov sources and transmitting data over ISI channels. We then described the precoding idea of Laroia *et al.* [31] that solves the problem of realizing both shaping and coding gains for data transmission over ISI channels. Due to the duality between quantization and transmission, we have utilized this precoding idea in developing precoded TB-SVQ that can realize both granular and boundary gains for Markov sources. The granular gain is realized by the underlying trellis code while the combination of the precoder and the SVQ structure provides the boundary gain. This new quantization scheme is asymptotically optimal and can, in principle, approach the rate-distortion bound for Markov sources.

We have presented two different suboptimal precoded TB-SVQ codebook search algorithms: the first is AEP-motivated while the second is dynamic programming based. Both algorithms have been carefully derived and were summarized in a C-like programming language format. Also, their implementation complexity are

briefly discussed. At high rates, the AEP-motivated algorithm is as simple as the Viterbi trellis search algorithm of the TCQ [21]. At low rates, however, overload events occur too often, hence significantly raising the actually search complexity. In this case, the dynamic programming based algorithm should be used instead because its implementation complexity is quite affordable and fixed number of operations can be expected to produce the quantized output.

The precoding error will disturb the optimal codebook boundary and reduce the boundary gain if it is significantly large. Two modified precoding schemes recently preposed by Laroia [45] were presented in their simplest form based on Ungerboeck's 1-D trellis code. These modified precoders can effectively reduce the precoding error and have been employed to further improve the precoded TB-SVQ performance.

Simulation results presented indicate that at high rates precoded TB-SVQ outperforms predictive TB-SVQ for higher order Gauss-Markov sources. Predictive TB-SVQ slightly or significantly outperforms precoded TB-SVQ for lower order Gauss-Markov sources or at lower rates, respectively. This suggests that while predictive TB-SVQ could be useful for low-rate coding of Markov sources precoded TB-SVQ could be potentially useful for high-rate coding of higher order Gauss-Markov sources.

Chapter 5

Entropy-Constrained Coset-Coded Quantization

5.1 Introduction and Outline

Due to its simplicity, entropy-constrained scalar quantizer (ECSQ) has been and continues to be a popular method of digitizing analog signals. Optimal ECSQs are known to perform within about 1.53 dB of the rate-distortion bound for a large class of memoryless sources [23]. ECSQ, however, cannot capitalize on the granular gain as its quantization cells are approximately cubic. While entropy-constrained vector quantizers (ECVQs) [27] can achieve the granular gain and possibly memory gain for Markov sources, the lack of their codebook structures makes them difficult to implement for a large block-length or encoding rate. In this chapter, we address the problem of designing "structured" entropy-constrained quantizers that can achieve both the granular and memory gains.

The widespread use of ECSQs has naturally spurred a significant activity in

optimizing their performance [23]-[26]. We next provide some results within this activity that are related to the course of this chapter. Gish and Pierce's results [23] suggest that optimal ECSQ tends toward uniform scalar quantizer (USQ) in the limit of large entropy. Optimal quantizers in the low bit-rate region, however, are generally non-uniform and are designed based on solving a Lagrangian functional minimization problem [24]-[26]. Farvardin and Modestino [26] later show that the optimal ECSQ performance can be obtained using *uniform-threshold quantizers* (UTQs) which have the uniformly distributed quantization thresholds like the USQ but have their reproduction values at the centroids of each quantization interval. Rate-distortion performance of the DPCM schemes for Markov sources have also been studied in [42].

For high-rate quantization of memoryless sources, Fischer and Wang [28] show that the performance improvement over ECSQ of a coding scheme placing an entropy coder in tandem with TCQ is roughly source independent [28]. This fact suggests that a significant granular gain that can be realized by TCQ [21] for memoryless uniform sources should also be achievable in their entropy-constrained TCQ (ECTCQ) [28] for memoryless non-uniform sources. For Markov sources, Fischer and Wang [28] also extend ECTCQ to *predictive* ECTCQ — a variable-rate predictive coding scheme that performs the DPCM predictive coding operation in each search path of the ECTCQ codebook search algorithm.

ECTCQ can also be considered as a trellis-coded extension of ECSQ. Likewise, predictive ECTCQ is a trellis-coded extension of the DPCM scheme. Like ECSQ, optimal ECTCQ is typically designed based on solving a Lagrangian functional minimization problem [28]; consequently, the ECTCQ codebook search algorithm is similar to that for TCQ but subject to a new distortion measure (to be described

in the sequel).

The focus of this chapter is threefold: (i) development of ECTCQ that takes into account symmetry properties of the underlying Ungerboeck trellis code; (ii) study of possible quantization schemes for Markov sources other than predictive ECTCQ; and (iii) development of the trellis-coded extensions of USQ and UTQ to the sequence space. So far, the ECTCQs reported in the literature [28, 32] do not exploit the symmetry properties of Ungerboeck trellis codes in any sense. We will investigate if these symmetry properties, when exploited appropriately, can effectively reduce the system implementation complexity. Besides describing predictive ECTCQ, we will also consider combining the precoding idea described in Section 4.3 and ECTCQ for quantizing Markov sources. There is no question that both USQ and UTQ are simpler to design and implement than optimal ECSQ. We expect that this fact should carry over to the multidimensional space, so that entropy-constrained quantizers based on coset codes (see Section 2.4) can also be simpler to design and implement than optimal ECTCQ.

The rest of this chapter is organized as follows. In Section 5.2, we address issues related to implementation of ECTCQ. Extensions of ECTCQ to Markov sources are considered in Section 5.3. In Section 5.4, we describe the possibly simpler coset-based entropy-constrained quantizers. Finally, a summary and conclusions are provided in Section 5.5.

5.2 ECTCQ

In this section, we consider issues related to implementation of ECTCQ. We first present the codebook structure of ECTCQ and its search algorithm in Section 5.2.1.

In Section 5.2.2, we describe three different approaches of entropy encoding the ECTCQ reproduction sequence, resulting in three respective versions of ECTCQ. What is novel here is the third version of ECTCQ whose reproduction alphabet is constrained to be symmetric about the origin. The motivation for placing this symmetry constraint is that the probability density functions (p.d.f.'s) of most of the useful sources are symmetric about the origin. In Section 5.2.3, we show that this symmetry property, along with the symmetry properties of Ungerboeck trellis codes, can be exploited to provide relatively simple implementation vis-à-vis the ECTCQ reported in the literature [28, 32]. Numerical results of ECTCQ for quantizing memoryless sources are provided in Section 5.2.4.

5.2.1 Generic Codebook Structure and Search Algorithm

In what follows we describe the generic codebook structure of ECTCQ and its codebook search algorithm. We will assume that the underlying trellis code is an Ungerboeck trellis code (see Section 2.4.2). Generalization to other trellis codes is possible but will be omitted here.

An ECTCQ has a scalar reproduction alphabet $\mathcal{Y} \equiv \{y_1, y_2, \cdots, y_{2m}\}$, which is partitioned into four subsets, namely $\mathcal{D}_0$, $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$. For each Ungerboeck trellis code, there is an underlying rate-1/2 convolutional code $\mathcal{C}$ and each of the four codebook subsets is indexed exclusively by a 2-bit output codeword of the convolutional encoder. A sequence of symbols from $\mathcal{Y}$ is said to be an ECTCQ reproduction sequence if its subset indices correspond to a code-sequence of $\mathcal{C}$.

The ECTCQ codebook design problem was solved by Fischer and Wang in [28] based on a Lagrangian formulation typically used for solving entropy-constrained minimization problems [26]-[27]. This design algorithm will not be reiterated here;

but we provide in Appendix A details of the design algorithm for a special type of ECTCQ that will be described shortly. For continuity of presentation, however, we do need to mention that under such a Lagrangian formulation the optimal ECTCQ reproduction sequence is the one that is closet to the input source sequence subject to the so-called *biased squared-error distortion measure* $\xi : \mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ evaluated as

$$\xi(x, y_j) = (x - y_j)^2 + \lambda \ell_j, \tag{5.1}$$

where $\lambda > 0$ is the Lagrangian multiplier and ℓ_j (in bits) denotes the minimum information required to be sent for an event that an input random variable is quantized to y_j. The Lagrangian multiplier λ has a geometrical interpretation as the negative slope of the tangent line supporting the optimal quantizer operational rate-distortion performance for a given source. At high rates, λ tends to zero and $\xi(\cdot, \cdot)$ hence gradually degenerates to the normal squared-error distortion measure.

We next describe the ECTCQ codebook search algorithm which determines for the input source sequence $\{x_i\}$ the optimal ECTCQ reproduction sequence $\{\hat{x}_i\}$. This is simply the Viterbi trellis search algorithm of TCQ [21] but subject to the biased squared-error distortion measure $\xi(\cdot, \cdot)$. Denote by Δ_i^s the minimum accumulated biased distortion (or called Lagrangian functional) that results when the first i source samples are quantized to a reproduction sequence reaching trellis state $s \in \boldsymbol{\Sigma}$. Suppose that the quantizer assumes a quantization delay of d source samples. To handle this quantization delay, we need buffers $\boldsymbol{q}_i^s \equiv (q_{i,0}^s, q_{i,1}^s, \cdots, q_{i,d}^s)$ and $\boldsymbol{\sigma}_i^s \equiv (\sigma_{i,0}^s, \sigma_{i,1}^s, \cdots, \sigma_{i,d}^s)$ for each trellis state s; the specific meanings for these buffers and how they operate can be found in the codebook search algorithm for any delayed decision quantizer presented in the previous chapters. Recursively, we update Δ_i^s (and hence determine $\boldsymbol{q}_i^s$ and $\boldsymbol{\sigma}_i^s$) by solving the following minimization

for $(s \in \boldsymbol{\Sigma})$ if $(s ==$ a specified trellis state$)$ $\Delta_{-1}^{s} = 0$; else $\Delta_{-1}^{s} = \infty$;
for $(i = 0;\ i < \infty;\ i++)$ {

for $(k \in \boldsymbol{\mathcal{J}}_4)$ $y^k = \arg\min_{y_j \in \mathcal{D}_k} \xi(x_i, y_j)$;
for $(s \in \boldsymbol{\Sigma})$ {

$s^* = \arg\min_{s'} \Delta_{i-1}^{s'} + \xi(x_i, y^{k(s',s)})$;
$z = y^{k(s^*,s)}$;
$\Delta_i^s = \Delta_{i-1}^{s^*} + \xi(x_i, z)$;
$\boldsymbol{q}_i^s \equiv (\boldsymbol{q}_{i-1}^{s^*}, z)$;
$\boldsymbol{\sigma}_i^s \equiv (\boldsymbol{\sigma}_{i-1}^{s^*}, s)$;

}

if $(i \geq d)$ {

$s^* = \arg\min_{s \in \boldsymbol{\Sigma}} \Delta_i^s$;
release $\hat{x}_{i-d} = q_{i,0}^{s^*}$ for entropy encoding;
for $(s \in \boldsymbol{\Sigma})$ {

if $(\sigma_{i,0}^s \neq \sigma_{i,0}^{s^*})$ $\Delta_i^s = \infty$;
$\boldsymbol{\sigma}_i^s \equiv (\sigma_{i,1}^s, \sigma_{i,2}^s, \cdots, \sigma_{i,d}^s)$;
$\boldsymbol{q}_i^s \equiv (q_{i,1}^s, q_{i,2}^s, \cdots, q_{i,d}^s)$;

}

}

}

Figure 5.1: ECTCQ codebook search algorithm.

problem:

$$s^* = \arg\min_{s'} \Delta_{i-1}^{s'} + \xi(x_i, y^{k(s',s)}), \tag{5.2}$$

where s' ranges between two possible previous states for s, $k(s', s)$ is the index to the codebook subset allowed for the specific trellis transition from s' to s, and $y^{k(s',s)}$ is the closest point within $D_{k(s',s)}$ to x_i under the distortion measure $\xi(\cdot, \cdot)$. The ECTCQ codebook search algorithm is summarized in Figure 5.1.

5.2.2 Three Versions of ECTCQ

Here we present three versions of ECTCQ corresponding to various approaches of entropy encoding the reproduction sequence. The first is the most general one and works even for other than Ungerboeck trellis codes. The resulting ECTCQ, however, can operate only at rates no less than 1 bit/sample. The other approaches work only for Ungerboeck trellis codes and the resulting quantizers can operate at any possible rate. The third approach further requires that the reproduction alphabet satisfies a symmetry property to be discussed shortly. In all cases, we assume that an ideal entropy coder always exists and encodes y_j into a binary codeword of ℓ_j bits — the minimum information required for representing an event that an input random variable is quantized to y_j.

Subset-Based ECTCQ [28]

This is the naive ECTCQ proposed by Fischer and Wang [28]. Recall that each state transition in the trellis diagram is labeled by one of the four possible codebook subsets and can be specified by a corresponding 1-bit input codeword to the underlying convolutional encoder. Given a reproduction sequence, this ECTCQ explicitly specifies the sequence of subset indices by the corresponding sequence of 1-bit input codewords to the convolutional encoder. Besides, each symbol in $\mathcal{D}_k$ is translated into a binary codeword that is a concatenation of a 1-bit codeword specifying the trellis transition (and hence $\mathcal{D}_k$) and a codeword from a subset-dependent[1] entropy codebook specifying the symbol within $\mathcal{D}_k$; the resulting ECTCQ is hence called the *subset-based* ECTCQ. The average encoding

[1]For simplicity, one usually assumes that the entropy coder is state-independent.

rate is $1 + H(Y|D)$ bits/sample where

$$H(Y|D) = -\sum_{k=0}^{3} P(\mathcal{D}_k) \sum_{y \in \mathcal{D}_k} P(y|\mathcal{D}_k) \log_2 P(y|\mathcal{D}_k), \tag{5.3}$$

$P(\mathcal{D}_k)$ is the probability of selecting a reproduction symbol from $\mathcal{D}_k$, and $P(y|\mathcal{D}_k)$ is the conditional probability of selecting y within $\mathcal{D}_k$.

It is obvious that the subset-based ECTCQ can operate only at rates no less than 1 bit/sample[2]. In what follows we describe two other approaches of entropy encoding the reproduction sequence such that the resulting ECTCQs can also operate at lower rates.

Superset-Based ECTCQ [32]

This is an ECTCQ based on a clever idea proposed by Marcellin [32] that absorbs the subset index information into the entropy encoding operation.

Define two *supersets* $\mathcal{B}_0 = \mathcal{D}_0 \bigcup \mathcal{D}_2$ and $\mathcal{B}_1 = \mathcal{D}_1 \bigcup \mathcal{D}_3$. Let the underlying trellis code be an Ungerboeck trellis code, we have that for any trellis state all the allowed reproduction symbols constitute either $\mathcal{B}_0$ or $\mathcal{B}_1$. We can therefore convert each reproduction symbol into a binary codeword from one of the two superset-dependent entropy codebooks; the resulting ECTCQ is hence called the *superset-based* ECTCQ. Given a current trellis state the reproduction symbol determines uniquely the trellis transition (and hence the next state); there is no need to send separate information regarding the subset indices. The average encoding rate is $H(Y|B)$ bits/sample where

$$H(Y|B) = -\sum_{k=0}^{1} P(\mathcal{B}_k) \sum_{y \in \mathcal{B}_k} P(y|\mathcal{B}_k) \log_2 P(y|\mathcal{B}_k), \tag{5.4}$$

$P(\mathcal{B}_k)$ is the probability of selecting a reproduction symbol from $\mathcal{B}_k$, and $P(y|\mathcal{B}_k)$ is the conditional probability of selecting y within $\mathcal{B}_k$.

[2]Reasonable performance was achieved only for rates higher than 1.5 bits/sample [28].

Magnitude-Based ECTCQ

The p.d.f.'s of most of the useful sources are symmetric about the origin. For this type of sources we consider a specific ECTCQ whose reproduction alphabet $\mathcal{Y} \equiv \{\pm y_1, \pm y_2, \cdots, \pm y_m\}$ (where $0 < y_1 < y_2 < \cdots < y_m$) is also symmetric about the origin. The ECTCQ design algorithm this added symmetry constraint is provided in Appendix A.

According to Ungerboeck's rule [22], one can partition $\mathcal{Y}$ into two supersets $\mathcal{B}_0$ and $\mathcal{B}_1$ such that (i) for each trellis state all the allowed reproduction symbols constitute either $\mathcal{B}_0$ or $\mathcal{B}_1$; and (ii) two reproduction symbols of the identical magnitude are assigned exclusively to $\mathcal{B}_0$ or $\mathcal{B}_1$. Denote by $\mathcal{Q} \equiv \{y_1, y_2, \cdots, y_m\}$ the set of all possible magnitudes. Following the discussion in Section 2.4.2, one can reconstruct the actual reproduction sequence from its corresponding magnitude sequence using a state-dependent function $\mathcal{F} : \Sigma \times \mathcal{Q} \rightarrow \mathcal{Y}$ similar to (2.18).

Like TB-SVQ which encodes the magnitude sequence using an SVQ encoder, the basic idea of what we will call the *magnitude-based* ECTCQ is simply to entropy encode the magnitude sequence. The average encoding rate is $H(Y)$ bits/sample where

$$H(Y) = -\sum_{y \in \mathcal{Q}} P(y) \log_2 P(y), \tag{5.5}$$

and $P(y)$ is the probability of selecting reproduction symbols of magnitude y.

5.2.3 Complexity Issues

Here we describe the complexity issues for implementing the ECTCQ. We first analyze both computational operations per sample and overall memory required for the ECTCQ codebook search algorithm. This is followed by a brief discussion

on the relative complexity required for the entropy coder. We should mention that the three versions of ECTCQ described in Section 5.2.2 differ only in the way the reproduction sequence is entropy encoded but not for the codebook search algorithm presented in Section 5.2.1.

Codebook Search

Let us keep Figure 5.1 as a reference and derive the complexity required for the ECTCQ codebook search. Computing all the branch metrics over $\mathcal{Y}$ for an input sample requires $6m$ operations ($4m$ additions and $2m$ multiplications) where $\text{card}(\mathcal{Y}) = 2m$; we assume that each $\lambda\ell_j$ is pre-computed and stored. Besides, it takes $2m - 4$ comparisons to determine the smallest branch metrics (each for one of the four codebook subsets). Determining Δ_i^s, through solving (5.2), requires 3 operations (2 additions and 1 comparison). Since $\text{card}(\boldsymbol{\Sigma}) = 2^\nu$, it totally costs $3 \cdot 2^\nu$ operations to determine Δ_i^s for all $s \in \boldsymbol{\Sigma}$. Finally, $2^\nu - 1$ comparisons are required to choose the optimal reproduction sequence. Overall, the computational cost is $[8m + 4 \cdot 2^\nu - 5]$ operations/sample. The major memory requirement comes from variables such as Δ_i^s, $\boldsymbol{q}_i^s$, and $\boldsymbol{\sigma}_i^s$. After some computations, the total storage cost for these variables is found to be $\left[2^{\nu+1} + \frac{d+1}{16}(\nu + \lceil \log_2 m \rceil)\right]$ 32-bit words.

Entropy Encoding

Now we consider the complexity required to entropy encode the reproduction sequence. Specifically, the well-known Huffman codes [7] and arithmetic codes [8, 9] are considered. We should mention that the arithmetic coder has now been known to be superior to the better-known Huffman coder in many aspects (see [8, 9]).

Suppose that every N-vector taken from the reproduction sequence is encoded by an N^{th} order Huffman coder. As the Huffman coding operation is essentially a table lookup (computational complexity is trivial), the overall size of lookup table is

used as an indication to the storage complexity. In the subset-based ECTCQ [28], each subset has $m/2$ reproduction symbols. A block of N subsets have therefore $(m/2)^N$ entries. There are 4^N different types of a block of N subsets. The overall lookup table size is hence $(2m)^N$. Similar discussion applies and we obtain an identical amount of lookup table size for the superset-based ECTCQ [32]. In the magnitude-based ECTCQ, the magnitude set $\mathcal{Q}$ has m symbols. It is obvious that an N^{th} order Huffman coder based on $\mathcal{Q}$ has a lookup table of m^N entries. The magnitude-based ECTCQ is hence 2^N times less complex (in terms of memory requirement) than the other counterparts in encoding the reproduction sequence using an N^{th} order Huffman coder.

Now consider encoding the reproduction sequence by an arithmetic coder. In the subset-based (superset-based) ECTCQ where the entropy coder operates in somewhat a way similar to a finite-state machine, one needs to store the sets of conditional probabilities of reproduction symbols within all subsets (supersets) and recursively adopts one set as specified by the trellis transition. It is simpler to arithmetic encode the magnitudes in the magnitude-based ECTCQ because the magnitude set $\mathcal{Q}$ has one half less the reproduction alphabet and the probabilities of symbols in $\mathcal{Q}$ are always fixed inherently in the arithmetic coder.

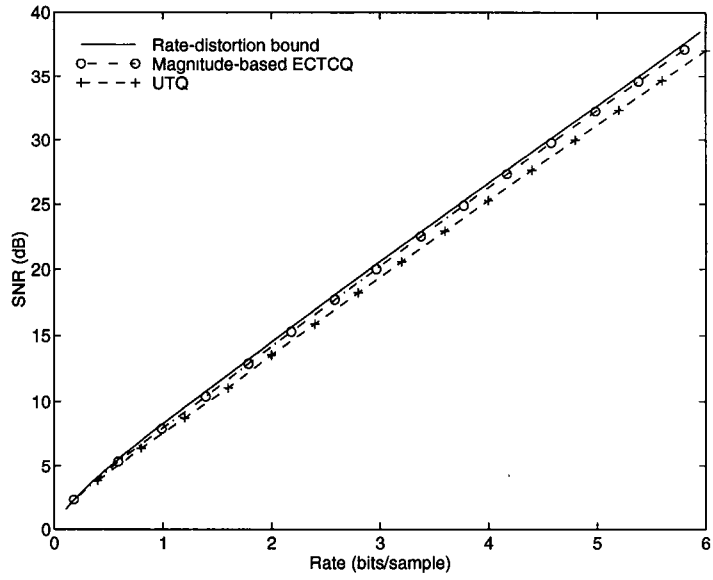
5.2.4 Simulation Results

We have designed the magnitude-based ECTCQ (using the algorithm provided in Appendix A based on a training sequence of at least 10^6 samples) for some typically used generalized Gaussian sources. In all cases, Ungerboeck's 1-D 8-state trellis code [22] is assumed and the initial trellis state is set to 0; the convergence threshold ϵ is set to 0.0005; and the Lagrange multiplier λ is tuned by experiments to yield an

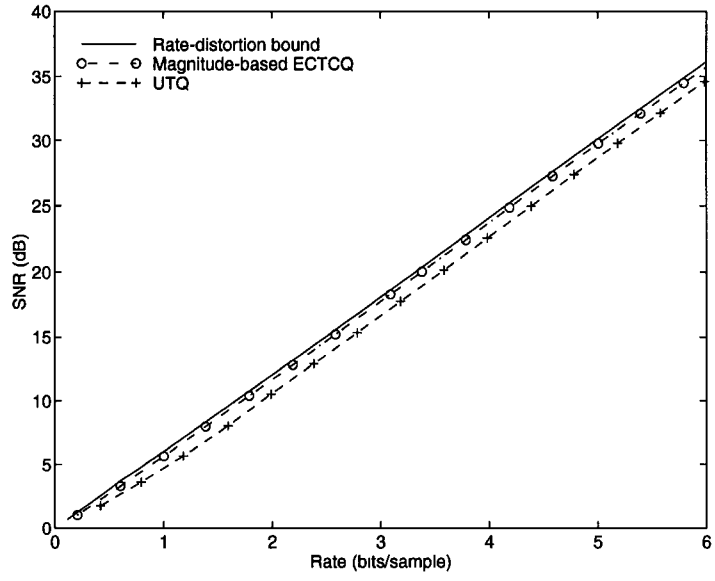
output entropy close to design target. In what follows we provide the performance of the optimized quantizers on encoding separately generated sequence of $1,000$ blocks, each of $1,000$ samples.

Figure 5.2 illustrates the magnitude-based ECTCQ SNR-entropy performance for two different generalized Gaussian sources. For comparison, the rate-distortion bound and entropy-constrained UTQ performance for each source are also included. The UTQ performance is known to be close to that of the optimal ECSQ [26]. The performance gap between the magnitude-based ECTCQ and UTQ hence gives an indication to the granular gain achieved by the trellis coding. Numerical results indicate that the magnitude-based ECTCQ performs about 1 dB better than the UTQ and within 0.5 dB of the rate-distortion bound at high rates. At lower rates, ECTCQ can achieve relatively fewer granular gain over UTQ, especially for sources with broad-tailed p.d.f. (e.g., the one with shape parameter $\theta = 0.6$). We should also mention that there is no noticeable performance difference between the magnitude-based ECTCQ and the subset-based or superset-based ECTCQ reported in the literature [28, 32], indicating that the imposition of the symmetry constraint on the reproduction alphabet essentially costs no performance loss. We will in the sequel assume implicitly that ECTCQ is magnitude-based.

We have also conducted experiments on entropy encoding the ECTCQ output. Figure 5.3 illustrates the coding redundancy (in bits/sample) — the difference between the actual coding rate and the average output entropy — yielded when using Huffman or arithmetic coders to encode the ECTCQ reproduction seqeunce for the memoryless Gaussian source. In the Huffman coding approach, to reduce the coding redundancy while simultaneously taking the implementation issue into account, an N-th order Huffman code is used where N is chosen such that the



(a)



(b)

Figure 5.2: Magnitude-based ECTCQ performance (SNR in dB) on encoding two generalized Gaussian sources with shape parameters (a) $\theta = 0.6$ and (b) $\theta = 2.0$.

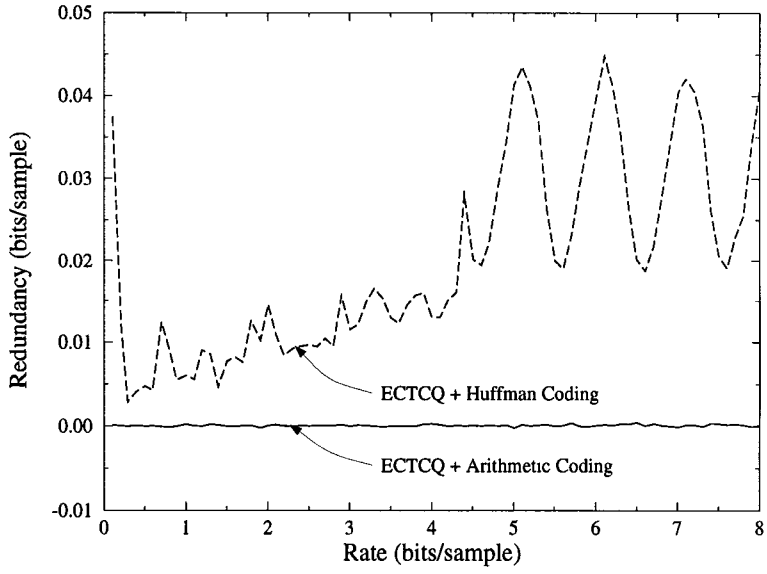


Figure 5.3: ECTCQ coding redundancy using Huffman and arithmetic coders; the source is memoryless Gaussian.

overall codebook size m^N is no larger than 2048. Among other feasible advantages mentioned in [8, 9], the arithmetic coding approach apparently has a smaller coding redundancy than the Huffman coding approach.

All the previous simulation results were obtained by assuming a sufficiently large quantization delay (of $1,000$ source samples). The ECTCQ performance on encoding the memoryless Gaussian source at 3.0 bits/sample as a function of the quantization delay d (in source samples) is illustrated in Figure 5.4. We observe that while larger quantization delay does yield better quantization performance, in practice, a delay of 100-200 samples is sufficient as the ECTCQ performance saturates quickly at such a quantization delay.

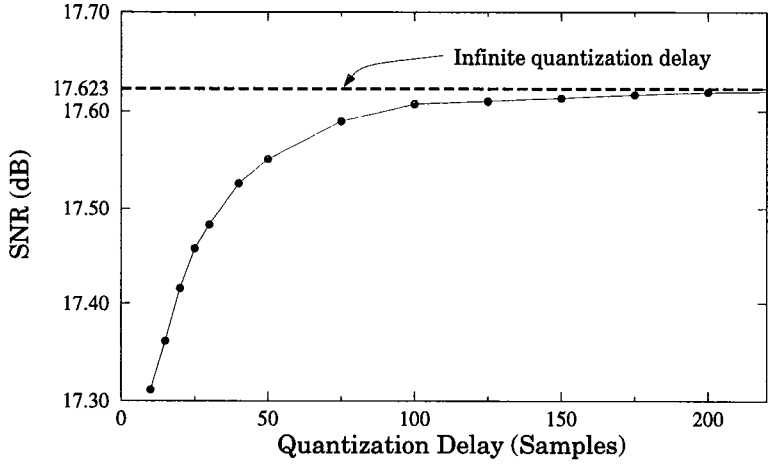


Figure 5.4: ECTCQ performance (SNR in dB) as a function of the quantization delay on encoding a memoryless Gaussian source at 3 bits/sample.

5.3 Extensions of ECTCQ to Markov Sources

In this section, we present two quantization schemes — dubbed predictive and precoded ECTCQs — that are extensions of ECTCQ to Markov sources. The way we extend ECTCQ to predictive or precoded ECTCQ is exactly the same as we extended TB-SVQ to predictive or precoded TB-SVQ in Chapter 3 or 4, respectively. The basic idea is to perform the DPCM predictive coding operation or the precoding transformation separately in each search path of the ECTCQ codebook search algorithm.

5.3.1 Predictive ECTCQ

This quantization scheme generalizes the ECTCQ codebook search algorithm such that each search path performs the predictive coding operation of the DPCM scheme. The survivor path reaching trellis state s at time $i-1$ has stored itself the most recent p samples in the corresponding candidate quantized output sequence

up to time $i-1$, denoted by $\hat{\boldsymbol{x}}_{i-1}^{s} \equiv (\hat{x}_{i-p}^{s}, \cdots, \hat{x}_{i-2}^{s}, \hat{x}_{i-1}^{s})$. Based on $\hat{\boldsymbol{x}}_{i-1}^{s}$, a *pathwise* prediction $\tilde{x}_i^s$ of the input sample x_i can be made according to

$$\tilde{x}_i^s = \sum_{j=1}^{p} \rho_j \hat{x}_{i-j}^s; \tag{5.6}$$

the corresponding prediction residual is $x_i - \tilde{x}_i^s$. Denote by Δ_i^s the minimum accumulated Lagrangian functional (biased suqared-error distortion) that results when the first i prediction residuals (along the search path) are quantized to an ECTCQ reproduction sequence reaching trellis state $s \in \boldsymbol{\Sigma}$. The minimization problem in (5.2) now becomes

$$(s^*, y^*) = \arg\min_{(s',y')} \Delta_{i-1}^{s'} + \xi(x_i - \tilde{x}_i^{s'}, y'), \tag{5.7}$$

where s' ranges between two possible previous states for s and y' is the closest point to $x_i - \tilde{x}_i^{s'}$ within the codebook subset allowed for the trellis transition from s' to s under the biased squared-error distortion measure $\xi(\cdot, \cdot)$. The predictive ECTCQ codebook search algorithm is summarized in Figure 5.5.

Predictive ECTCQs belong to the class of closed-loop predictive quantizers (see Section 3.5). For this class of predictive quantizers, there is an unsolved problem as to detremine the p.d.f. of the prediction residual. We will bypass this problem in the course of developing predictive ECTCQ by designing the underlying ECTCQ matched directly to the p.d.f. of the innovations process. We should mention that the abovementioned predictive ECTCQ codebook search algorithm is not optimal. The suboptimality is due to the "greedy" nature in discarding the search paths which currently have larger accumulated Lagrangian functionals than the survivor one. As the future prediction residuals depend on the current quantization, chances are that paths which eventually lead to smaller accumulated Lagrangian functionals are already abondoned. Optimal codebook search algorithm for predictive ECTCQ

```
for (s ∈ Σ) if (s == a specified trellis state) Δ^s_{-1} = 0; else Δ^s_{-1} = ∞;
for (i = 0; i < ∞; i++) {

    for (s ∈ Σ) x̃^s_i = Σ_{j=1}^{p} ρ_j x̂^s_{i-j};
    for (s ∈ Σ) {

        (s*, y*) = arg min_{(s',y')} Δ^{s'}_{i-1} + ξ(x_i − x̃^{s'}_i, y');
        Δ^s_i = Δ^{s*}_{i-1} + ξ(x_i − x̃^{s*}_i, y*);
        x̂^s_i ≡ (x̂^{s*}_{i-1}, x̃^{s*}_i + y*);
        q^s_i ≡ (q^{s*}_{i-1}, y*);
        σ^s_i ≡ (σ^{s*}_{i-1}, s);

    }

    if (i ≥ d) {

        s* = arg min_{s∈Σ} Δ^s_i;
        release ŷ_{i-d} = q^{s*}_{i,0} for entropy encoding;
        for (s ∈ Σ) {

            if (σ^s_{i,0} ≠ σ^{s*}_{i,0}) Δ^s_i = ∞;
            σ^s_i ≡ (σ^s_{i,1}, σ^s_{i,2}, ···, σ^s_{i,d});
            q^s_i ≡ (q^s_{i,1}, q^s_{i,2}, ···, q^s_{i,d});

        }

    }

}
```

Figure 5.5: Predictive ECTCQ codebook search algorithm.

is not available yet if it is not an exhaustive approach which is prohibitive.

5.3.2 Precoded ECTCQ

This is a quantization scheme that combines ECTCQ and the precoding idea of Laroia *et al.* [31]. Note that a synchronous operation between the precoder and inverse precoder requires that the reproduction alphabet $\mathcal{Y}$ be unformly spaced.

We will hence consider only the uniform ECTCQ[3] whose reproduction alphabet $\mathcal{Y}$ can be written as $\{\pm\delta, \pm 3\delta, \cdots, \pm(2m-1)\delta\}$ for some scaling factor δ. The codebook search is essentially the same as that for predictive ECTCQ, except that in each search path we perform the precoding transformation.

In what follows we will simplify the derivation of the codebook search algorithm by neglecting some (possibly critical) technical background; the reader is referred to the discussion on the dynamic programming based precoded TB-SVQ codebook search algorithm provided in Section 4.4.2 for what she/he feels missed but crucial. Also, we will denote by $\boldsymbol{\Lambda}'$ a coset lattice of the coarsest lattice still containing $\mathcal{Y}$ as a subset. What $\boldsymbol{\Lambda}'$ really stands for will depend on the version of the precoder (see Sections 4.3 and 4.5).

Like the codebook search algorithm for predictive ECTCQ, the survivor path reaching trellis state $s \in \boldsymbol{\Sigma}$ at time $i-1$ has stored the most recent p samples in the corresponding candidate quantized output sequence up to time $i-1$ — $\hat{\boldsymbol{x}}_{i-1}^s \equiv (\hat{x}_{i-p}^s, \cdots, \hat{x}_{i-2}^s, \hat{x}_{i-1}^s)$. A prediction $\tilde{x}_i^s = \sum_{j=1}^p \rho_j \hat{x}_{i-j}^s$ of x_i based on $\hat{\boldsymbol{x}}_{i-1}^s$ is then quantized to the nearest point z_i^s in $\boldsymbol{\Lambda}'$. Denote by Δ_i^s the minimum accumulated (biased) distortion that results when the first i source samples are quantized such that the corresponding precoder output is a trellis-coded sequence of symbols from $\mathcal{Y}$ reaching state s. The minimization problem in (5.2) now becomes

$$(s^*, y^*) = \arg\min_{(s', y')} \Delta_{i-1}^{s'} + \xi(x_i - z_i^{s'}, y'), \tag{5.8}$$

where s' ranges between two possible previous states for s and y' is the closest point to $x_i - z_i^{s'}$ under the biased squared-error distortion measure within the

[3]Design algorithm for this type of uniform ECTCQ is provided in Appendix A.

```
for (s ∈ Σ) if (s == a specified trellis state) Δ^s_{-1} = 0; else Δ^s_{-1} = ∞;
for (i = 0; i < ∞; i++) {

    for (s ∈ Σ) z_i^s = nearest point in Λ' to Σ_{j=1}^{p} ρ_j x̂_{i-j}^s;
    for (s ∈ Σ) {

        (s*, y*) = arg min_{(s',y')} Δ_{i-1}^{s'} + ξ(x_i - z_i^{s'}, y');
        Δ_i^s = Δ_{i-1}^{s*} + ξ(x_i - z_i^{s*}, y*);
        x̂_i^s ≡ (x̂_{i-1}^{s*}, z_i^{s*} + y*);
        q_i^s ≡ (q_{i-1}^{s*}, y*);
        σ_i^s ≡ (σ_{i-1}^{s*}, s);

    }

    if (i ≥ d) {

        s* = arg min_{s∈Σ} Δ_i^s;
        release ŷ_{i-d} = q_{i,0}^{s*} for entropy encoding;
        for (s ∈ Σ) {
            if (σ_{i,0}^s ≠ σ_{i,0}^{s*}) Δ_i^s = ∞;
            σ_i^s ≡ (σ_{i,1}^s, σ_{i,2}^s, ..., σ_{i,d}^s);
            q_i^s ≡ (q_{i,1}^s, q_{i,2}^s, ..., q_{i,d}^s);
        }

    }

}
```

Figure 5.6: Precoded ECTCQ codebook search algorithm.

codebook subset allowed for the trellis transition from s' to s. This precoded ECTCQ codebook search algorithm is summarized in Figure 5.6.

Recall that at high rates the AEP-motivated codebook search algorithm is simpler than the dynamic programming based algorithm for precoded TB-SVQ (see Chapter 4). We consider the following precoded ECTCQ codebook search approach. Here we do not pathwise perform the precoding transformation but simply place the precoder in tandem with an unbounded TCQ (whose reproduction

alphabet is the coarsest lattice still contatining $\mathcal{Y}$). We have mentioned earlier in Section 5.2.1 that at high rates the biased squared-error distortion mensure $\xi(\cdot,\cdot)$ tends toward the normal squared-error distortion measure; TCQ is hence simply a Viterbi trellis search algorithm subject to the squared-error distortion measure. If the ECTCQ reproduction alphabet $\mathcal{Y}$ has sufficiently many symbols, almost all the precoder output samples will be in $\mathcal{Y}$. In case there is a precoder output sample that is not a symbol in $\mathcal{Y}$ (this could occur only infrequently), it can be shown that the reproduction sequence can be locally modified so as to regulate the corresponding precoder output sample into $\mathcal{Y}$.

5.3.3 Complexity and Simulation Results

In this subsection, we analyze the implementation complexity of predictive and precoded ECTCQs and provide their simulation results.

Let us first derive the complexity required for the predictive ECTCQ codebook search algorithm (presented in Figure 5.5). Computing each pathwise prediction $\tilde{x}_i^s$ requires $2p-1$ operations ($p-1$ additions and p multiplications). It hence requires $2^\nu(2p-1)$ operations per source sample to compute all the pathwise predictions. For each one of the two previous trellis states of $s \in \boldsymbol{\Sigma}$, denoted by s', computing the corresponding accumulated Lagrangian functional requires $(2m+1)$ operations (1 addition for computing pathwise prediction residual $x_i - \tilde{x}_i^{s'}$, $(2m-1)$ for selecting the symbol $y \in \mathcal{Y}$ that yields the smallest branch metric, and 1 addition for accumulating the Lagrangian functional). Determining Δ_i^s for all $s \in \boldsymbol{\Sigma}$, through solving (5.7), hence requires $2^\nu(4m+3)$ operations. Finally, $2^\nu - 1$ comparisons are required to choose the optimal search path. The overall computational cost is $[2^\nu(2p+4m+3)-1]$ operations/sample. Compared to the ECTCQ codebook

search algorithm (provided in Figure 5.1), extra memory requirement comes from variable such as $\hat{\boldsymbol{x}}_i^s$, which amounts to $[2^{\nu+1}p]$ 32-bit words.

The implementation complexity required for the precoded ECTCQ codebook search algorithm (presented in Figure 5.6) can be similarly analyzed and will be omitted. We next briefly disuss the implementation complexity required for the high-rate degenerated search algorithm mentioned near the end of Section 5.3.2. We will assume that the reproduction alphabet $\mathcal{Y}$ is large enough to encompass all the precoder output samples. The computational complexity is hence that of an uniform TCQ plus about $2p$ operations/sample for the precoding transformation. The memory requirement for the degenerated algorithm should be somewhat less than the algorithm presented in Figure 5.6 as there is no need to perform pathwise precoding operation (the pathwise variable $\hat{x}_i^s$ can be saved).

Based on simulations performed on test sequence of $1,000$ blocks, each of $1,000$ samples, the predictive and precoded ECTCQ performance (SNR in dB) with a sufficiently large quantization delay on encoding a 1^{th}-order Gauss-Markov source ($\rho_1 = 0.9$) are given in Figure 5.7; the underlying trellis code is Ungerboeck's 8-state trellis code. The simulation results indicate that at high rates (i) both predictive and precoded ECTCQs perform very close to the rate-distortion bound; and (ii) precoded ECTCQ slightly outperforms predictive ECTCQ. At rate as low as 1.0 bit/sample, predictive ECTCQ still performs reasonably well. Precoded ECTCQ starts to fall apart at rate near 3.0 bits/sample. This could be partially due to that fact that the precoding error gradually becomes significant and due to the discrepancy between the biased squared-error and squared-error distortion measures at low rates. Simulation results for precoded ECTCQ at lower rates are not available because the design algorithm given in Appendix A provides uniform

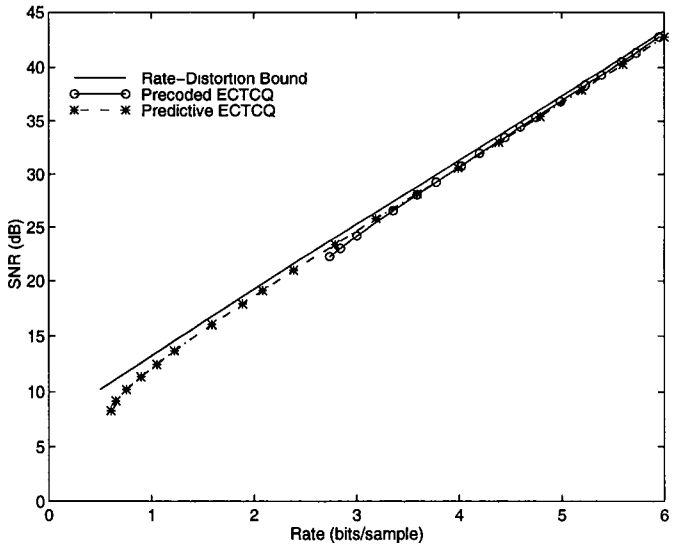


Figure 5.7: Performance (SNR in dB) of predictive and precoded ECTCQs on encoding a 1^{st}-order Gauss-Markov source ($\rho_1 = 0.9$).

ECTCQs that perform well only at rates no less than about 3.0 bits/sample.

5.4 Extensions of USQ and UTQ

In the limit of large rate, the (generally) non-uniform reproduction alphabet $\mathcal{Y}$ of ECTCQ tends toward uniform and the biased squared-error distortion measure degenerates to the normal squared-error distortion measure. One can hence expect that a high-rate ECTCQ will be very close in codebook structure to a naive coding scheme that is a concatenation of an unbounded uniform TCQ and an entropy encoder — an extension of the entropy-coded USQ to the sequence space. In Section 5.4.1, we generalize this idea and introduce a class of high-rate efficient entropy-coded quantizers that simply places an entropy encoder in tandem with a coset code quantizer (see Section 2.4). Since the underlying coset codes will be more

explicitly utilized in specifying the quantizer structure, we expect that this class of quantizers — called *coset-based quantizers* (CBQs) — should be simpler to design and implement than ECTCQ. In Section 5.4.2, we consider a modified version of CBQ used for low-rate quantization of memoryless sources — dubbed *coset-threshold quantizer* (CTQ) — that is an extension of UTQ to the multidimensional space.

5.4.1 Coset-Based Quantizers

Each one in this class of entropy-coded quantizers has its codebook structures explicitly specified by some coset codes $\mathbb{C}(\boldsymbol{\Lambda}/\boldsymbol{\Lambda}';\mathcal{C})$; the generic block diagram of this type of quantizers is illustrated in Figure 5.8. Depending on whether $\mathcal{C}$ is a block or convolutional code, $\mathbb{C}(\boldsymbol{\Lambda}/\boldsymbol{\Lambda}';\mathcal{C})$ reduces to the lattice $\boldsymbol{\Lambda}$ or a trellis code based on the lattice partition $\boldsymbol{\Lambda}/\boldsymbol{\Lambda}'$ and the binary code $\mathcal{C}$. Likewise, we shall say that CBQ reduces to *lattice-based quantizer* (LBQ) or *trellis-based quantizer* (TBQ). We should also mention that $\mathbb{C}(\boldsymbol{\Lambda}/\boldsymbol{\Lambda}';\mathcal{C})$ is scaled by a factor δ. How this scaling factor should be selected will be described shortly. For simplicity of discussion, we shall assume that $\delta = 1$.

The CBQ operates as follows. The input source N-vectors $\{\boldsymbol{x}_n\}$ is mapped by the coset quantizer to some code-sequence $\{\hat{\boldsymbol{x}}_n\}$ in $\mathbb{C}(\boldsymbol{\Lambda}/\boldsymbol{\Lambda}';\mathcal{C})$. Each N-vector $\hat{\boldsymbol{x}}_n \in \boldsymbol{\Lambda}$ is the sum of some coset representative $\boldsymbol{\lambda}(\boldsymbol{c}_n)$ and a point $\boldsymbol{\lambda}'_n$ in $\boldsymbol{\Lambda}'$ where $\boldsymbol{c}_n$ is the $(k+r)$-bit output codeword of the underlying rate-$k/(k+r)$ binary code $\mathcal{C}$. We can encode $\hat{\boldsymbol{x}}_n$ into a binary string that is a concatenation of the corresponding k-bit input codeword $\boldsymbol{w}_n$ to $\mathcal{C}$ and some variable-length codeword associated with $\boldsymbol{\lambda}'_n$. In the receiver, the reproduction vector $\hat{\boldsymbol{x}}_n$ can be reconstructed as the sum of the (entropy) decoded $\boldsymbol{\lambda}'_n$ and $\boldsymbol{\lambda}(\boldsymbol{c}_n)$ where $\boldsymbol{c}_n$ is the output codeword of $\mathcal{C}$

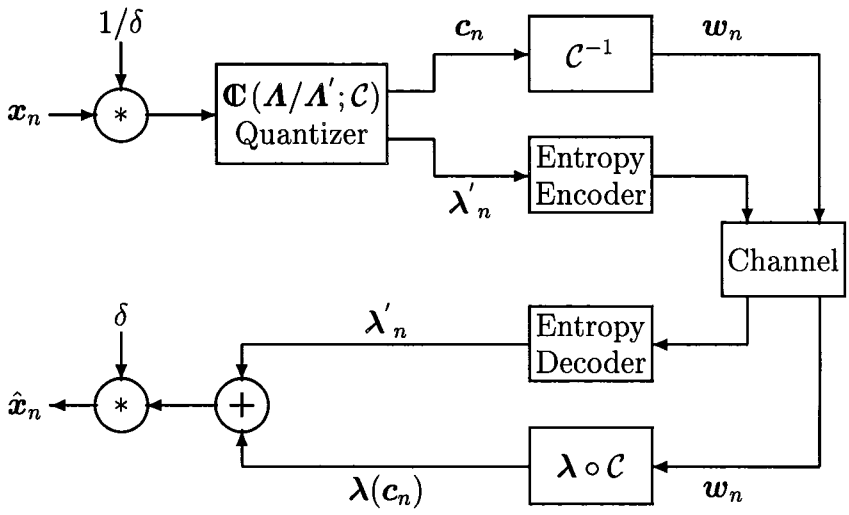


Figure 5.8: Block diagram of the coset-based quantizer.

corresponding to the input codeword $\boldsymbol{w}_n$.

We should mention that if $\boldsymbol{\Lambda}'$ is cubic the entropy encoding of $\boldsymbol{\lambda}'_n$ at high rates could be made as simple as that of N repetitions of a scalar entropy encoding. This is an essential property to coset-based quantizers such that the simplest effort of entropy encoding can be achieved; we shall assume that all coset-based quantizers satisfy this property. We next provide more respective details for some lattice-based or trellis-based quantizers that will be considered in the sequel.

Lattice-Based Quantizers

In this case, $\mathcal{C}$ is an (N, k) binary block code (where $N = k+r$) and $\mathbb{C}(\boldsymbol{\Lambda}/\boldsymbol{\Lambda}';\mathcal{C})$ reduces to the lattice $\boldsymbol{\Lambda}$ itself. Specific lattices that we will use in developing LBQ include $\mathbf{Z}$, D_4, and E_8; their corresponding block codes are described in Table 2.1 and the quantization algorithms for these lattices can be found in [39]. Note that the sublattices of these lattices are all cubic.

Trellis-Based Quantizers

The underlying trellis codes of trellis-based quantizers we will consider are Ungerboeck's rate-1/2 1-D trellis codes ($N = k = r = 1$). In this case, the box labeled "$\mathbb{C}(\boldsymbol{\Lambda}/\boldsymbol{\Lambda}';\mathcal{C})$ quantizer" in Figure 5.8 represents the Viterbi search algorithm [40] employed for $\mathbb{C}(\boldsymbol{\Lambda}/\boldsymbol{\Lambda}';\mathcal{C})$. We will assume that $\boldsymbol{\Lambda}$ is shifted such that it is symmetric about the origin. This yields the equivalence between any trellis code-sequence and its corresponding magnitude sequence. As we have shown in Section 5.2.3 for the magnitude-based ECTCQ, this can effectively reduce the entropy encoding complexity.

It is obvious that the only design issues for coset-based quantizer are the scaling factor δ for the coset code and the scalar entropy encoder. We next briefly describe a design methodology for this purpose. For a given scaling factor δ, a long training sequence $\{\boldsymbol{x}_n\}$ is quantized by the coset quantizer to some code-sequence $\{\hat{\boldsymbol{x}}_n\}$ in $\mathbb{C}(\boldsymbol{\Lambda}/\boldsymbol{\Lambda}';\mathcal{C})$. Taking the coset representative off each quantized N-vector $\hat{\boldsymbol{x}}_n$, we obtain $\{\boldsymbol{\lambda}'_n\}$ — a sequence of points in $\boldsymbol{\Lambda}'$. Since $\boldsymbol{\Lambda}'$ is cubic, an empirical scalar entropy codebook can be designed based on $\{\boldsymbol{\lambda}'_n\}$. The average quantizer output rate is the sum of r/N bits/sample (for the coset representives) and the average entropy rate of $\{\boldsymbol{\lambda}'_n\}$. This procedure is repeated for several values of δ until the average quantizer output rate is close to a desired entropy constraint.

The above CBQ design approach was performed for the memoryless Gaussian and Laplacian sources; the resulting design performance (SNR in dB) are presented in Tables 5.1 and 5.2. Note that the special case of LBQ based on $\mathbf{Z}$ actually corresponds to USQ. The simulation results indicate that at high rates the other quantizers outperform USQ by an amount equal to the granular gain achievable by the underlying coset codes. Also, the 8-state TBQ peformance is found to be close to that of the 8-state ECTCQ (presented in Section 5.2) at high rates. We should

	LBQ			TBQ: (2^ν)				
H_0	Z	D_4	E_8	4	8	16	32	$R(D)$
2.0	10.10	9.60	9.45	11.19	11.27	11.34	11.41	12.04
3.0	16.43	16.59	16.72	17.45	17.53	17.60	17.66	18.06
4.0	22.53	22.84	23.09	23.53	23.61	23.67	23.73	24.08
5.0	28.57	28.92	29.20	29.56	29.65	29.71	29.77	30.10
6.0	34.59	34.96	35.24	35.59	35.67	35.73	35.79	36.12
7.0	40.61	40.98	41.27	41.61	41.69	41.76	41.82	42.14
8.0	46.64	47.01	47.29	47.63	47.72	47.78	47.84	48.16

Table 5.1: Design performance (SNR in dB) of the entropy-constrained (by H_0) CBQ for the memoryless Gaussian source.

mention that these numerical results in Tables 5.1 and 5.2 are the ultimate "design" performance. When operating the designed quantizers on separately generated test sequence, chances are that source samples are quantized to some *null* symbols that have an empirically design probability of zero. To reduce the possibility of this co-called *overload* event, we have used an extraordinarily long training sequence (of 10^7 samples) in the CBQ design procedure. To completely avoid the overload event, one can either pad in the null symbols some non-zero probabilities according to the source p.d.f. (and redesign the entropy codebook) or locally disturb the coset code-sequence into the codebook. Either way, however, should yield the overall quantization performance slightly inferior to those given in Tables 5.1 and 5.2.

5.4.2 Coset-Threshold Quantizers

A necessary condition for quantizer optimality is that each reproduction vector should be located at the centroid of its corresponding quantization cell. In coset-based quantizers, each reproduction vector is actually located at the center of its corresponding quantization cell. This should not be a problem at high rates as the

	LBQ			TBQ: (2^ν)				
H_0	Z	D_4	E_8	4	8	16	32	$R(D)$
2.0	10.96	10.33	10.20	11.69	11.77	11.84	11.91	12.67
3.0	17.09	17.12	17.28	18.04	18.12	18.19	18.25	18.69
4.0	23.16	23.43	23.68	24.15	24.23	24.29	24.35	24.71
5.0	29.20	29.56	29.81	30.19	30.28	30.34	30.40	30.73
6.0	35.22	35.58	35.86	36.22	36.30	36.36	36.42	36.75
7.0	41.24	41.60	41.89	42.24	42.32	42.38	42.44	42.77
8.0	47.28	47.63	47.92	48.27	48.36	48.42	48.48	48.79

Table 5.2: Design performance (SNR in dB) of the entropy-constrained (by H_0) CBQ for the memoryless Laplacian source.

source p.d.f. is approximately uniform within each quantization cell (the center is effectively the same as the centroid). At low rates, however, this necessary condition generally is not satisfied.

Following the above discussion, we next provide a possible modification on the quantizer structure such that the necessary condition for optimality is better satisfied. We can employ another reproduction alphabet $\mathcal{Y} \equiv \{\boldsymbol{y}_j^k\}$ (other than the original one which is a subset of $\boldsymbol{\Lambda}$) where $\boldsymbol{y}_j^k$ — the j^{th} reproduction vector within the k^{th} coset — is derived as the empirical centroid of all those training data mapped to this specific quantization bin. This modification results in what we will call *coset-threshold quantizer* (CTQ) in the sequel. Again, depending on whether $\mathcal{C}$ is a block or convolutional code, we shall say that CTQ reduces to *lattice-threshold quantizer* (LTQ) or *trellis-threshold quantizer* (TTQ). We will, however, no longer discuss LTQ in the sequel as it entails an unstructured reproduction alphabet $\mathcal{Y}$ in the multidimensional space.

The least possible encoding rate of TBQ (or TTQ) will be 1 bits/sample if we explicitly specify the sequence of coset representatives as in the subset-based

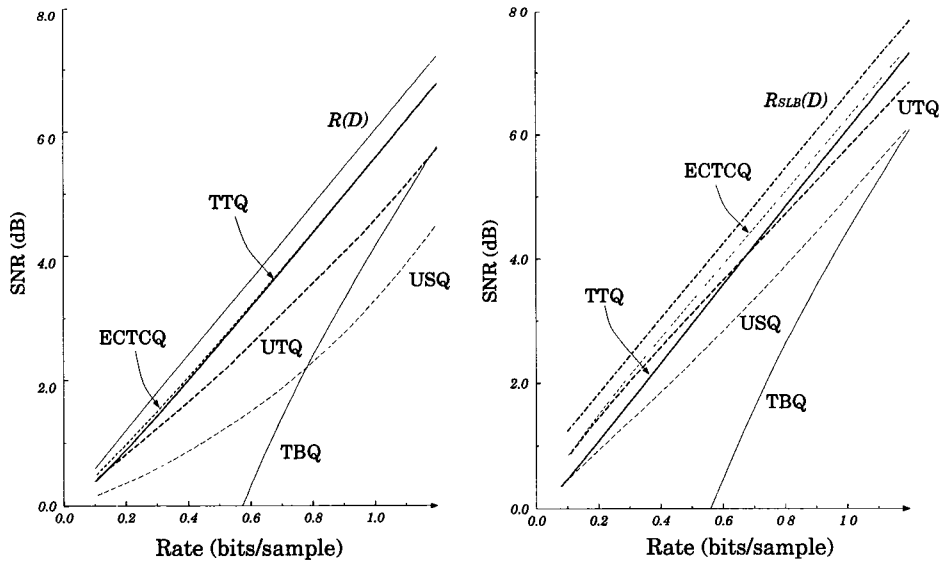


Figure 5.9: TTQ design performance (SNR in dB) for the memoryless (a) Gaussian, (b) Laplacian source; Ungerboeck's 8-state trellis code is assumed in TTQ, TBQ, and ECTCQ.

ECTCQ. At high rates, it is justifiable to do so since a source sample is quantized to each one of the two admissible cosets of Λ' with about the same probability. At low rates when the scale of the coset code becomes large, however, this probability distribution may be highly biased to some specific coset. There hence exists some redundancy in fixed-rate encoding the coset representatives. One can exploit the equivalence between any trellis sequence and its corresponding magnitude sequence and thereby reduce such a coding redundancy by entropy encoding the magnitude sequence as in the magnitude-based ECTCQ.

Figure 5.9 displays the 8-state TTQ design performance in the low bit-rate region for the memoryless Gaussian and Laplacian sources. The rate-distortion limit and performance of the 8-state ECTCQ, 8-state TBQ, USQ, and UTQ for these two specific sources are also included for comparision. The TTQ performance is

very close to that of ECTCQ at almost all rates for the Guassian source while TTQ is inferior to ECTCQ by about 0.2 - 0.4 dB for the Laplacian source. Besides, TTQ consistently outperforms UTQ for the Gaussian case while the TTQ performance is inferior to UTQ for the Laplacian source at rates less than 0.6 bits/sample. Again, these simulation results in Figure 5.9 are the ultimate design performance. The actual quantization performance for separately generated test sequence should be slightly inferior to those given in Figure 5.9.

5.5 Summary and Conclusions

In this chapter we have developed structured and efficient entropy-constrained quantization schemes for digitizing stationary sources. These quantization schemes are simple to implement as their codebook structures are specified based on the coset codes. Besides, these quantization schemes can realize over the optimal ECSQ the granular gain to provide performance much closer to the rate-distortion bound for memoryless sources. Possible extensions of these quantizers to Markov sources have also been discussed.

We have presented the generic ECTCQ codebook structure, its optimal search algorithm, and three versions of ECTCQ. What is novel is the third version — the magnitude-based ECTCQ — whose reproduction alphabet is constrained to be symmetric about the origin. Imposing such a symmetry constraint on the ECTCQ reproduction alphabet is heuristically reasonable as the p.d.f.'s of most of the useful sources are symmetric about the origin. Simulation results indicate that placing such a symmetry constraint costs no rate-distortion performance loss and can provide relatively simple implementation vis-à-vis the ECTCQ reported

in the literature [28, 32].

We have presented predictive and precoded ECTCQs that are extensions of ECTCQ for quantizing Markov sources. The way ECTCQ is extended to these two quantization schemes is equivalent to that of extending TB-SVQ to predictive or precoded TB-SVQ described in Chapter 3 or 4, respectively. We have also demonstrated that at high rates precoded ECTCQ degenerates to a simple scheme that is essentially a concatenation of an unbounded TCQ (subject to the normal squared-error distortion measure), a precoder, and an entropy encoder. Simulation results indicate that at high rates both quantizers (with an 8-state trellis code) perform within 0.5 dB of the rate-distortion limit for Markov sources. Generally speaking, at high rates, precoded ECTCQ has a better performance than predictive ECTCQ. However, precoded ECTCQ falls apart at rate near 3.0 bits/sample while predictive ECTCQ still performs reasonably well at lower rates.

We have also introduced a class of entropy-coded quantizers, called CBQ, that each places an entropy encoder in tandem with some coset code quantizer. These quantization schemes are extensions of the asymptotically (in rate) optimal USQ to the multidimensional space. A special case of CBQ — TBQ — can be considered as the high-rate degenerated version of ECTCQ. Since the underlying coset codes are more explicitly utilized in specifying the quantizer structure, CBQs are simpler to design and implement than ECTCQ. We also considered TTQ which is a modified version of TBQ used for digitizing memoryless sources at low rates and is an extension of UTQ to the sequence space.

One important feature of the quantizers presented in this chapter is that the quantizers operate effectively in the multidimensional space (the granular gain can be achieved) while the effort of entropy encoding the multidimensional symbols

is reduced to a simple repetition of some scalar encoding operations. This is in some sense equivalent to that TB-SVQ [15] (or see Chapter 3) is derived from an underlying scalar quantizer. We should mention that the fixed-rate TB-SVQ has to afford relatively higher complexity to provide a rate-distortion performance equal to that of the variable-rate quantizers presented in this chapter. Although the variable-rate coding nature renders the entropy-coded quantizers the drawback of possibly catastrophic error propagation in the case of noisy communication channel. There are, however, situations where the problem of error propagation can be effectively protected and hence the simpler variable-rate quantizers developed in this chapter should be favored.

Chapter 6

Adaptive Buffer-Instrumented Entropy-Coded Quantization

6.1 Introduction and Outline

While entropy-coded quantizers generally produce variable-length codewords, in most two-way communication situations the communication channel operates at a fixed rate. Transmission of the variable-length codewords over a fixed-rate channel necessitates the use of a buffer of finite, and preferably small, size. Clearly, one should prevent the buffer overflow/underflow event from occurring for otherwise it could lead to loss of codeword synchronization and hence a large quantization distortion. In this chapter, we consider buffer management strategies for ECTCQ and predictive ECTCQ.

The *adaptive entropy-coded quantizer* (AECQ) [46] is a buffer-instrumented variation of entropy-coded scalar quantizer in which a simple feedback control mechanism is used to maintain the buffer occupancy at the desired "half-full"

state. This scheme has been shown to be very effective in avoiding the buffer overflow/underflows. It has been shown in [46] that given a sufficiently large buffer size and a suitable feedback control mechanism, the cost for such a buffer-instrumentation adaptation is only some negligibly small additional quantization error. Also, the AECQ performance has been shown to be robust in the presence of some specific source mismatch conditions [47].

AECQ is essentially a scalar quantizer. To capitalize on the granular gain, we may generalize the underlying ECSQ to its vector extension — entropy-coded vector quantizer (ECVQ). In spite that it could outperform AECQ, the resulting quantization scheme is generally non-structured and will be prohibitive for a large block-length or encoding rate. It is due to this concern that we shall not pursue this issue but instead consider buffer-instrumented variations of the more structured ECTCQ.

In AECQ, the buffer occupancy at each time instant is always referenced to adapt the encoder characteristics right before the next source sample arrives. The feedback control of the AECQ hence operates in somewhat an "instantaneous" manner. A naive buffer-instrumented ECTCQ is one that replaces the underlying scalar quantizer of AECQ with ECTCQ. Due to the encoding delay associated with the ECTCQ codebook search procedure, however, the feedback control of this naive quantization scheme can only operate in a "delayed" manner.

There are three contributions that will be presented in this chapter: (i) the idea of *pathwise adaptation* which allows the feedback control to adjust the ECTCQ encoder characteristics instantaneously; (ii) extension of the buffer-instrumented quantization scheme to Markov sources; and (iii) study of the buffer-instrumented ECTCQ performance in the presence of source probability density function (p.d.f.)

mismatch conditions. The main theme of pathwise adaptation is simply that of performing the buffer-instrumented feedback control operation separately in each search path of the ECTCQ codebook search algorithm. This simple idea gives rise to what we will call *pathwise-adaptive* ECTCQ (PA-ECTCQ) in the sequel. To realize the memory gain for Markov sources, PA-ECTCQ can be extended to *predictive* PA-ECTCQ by combining the DPCM predictive coding operation into the PA-ECTCQ codebook search algorithm. Study of the PA-ECTCQ performance to some typical source mismatch conditions is encouraged by the fact that the AECQ performance is to some extent robust in the presence of these mismatch conditions [47].

The rest of this chapter is organized as follows. In Section 6.2, we briefly review AECQ, describe a naive buffer-instrumented variation of ECTCQ, and present PA-ECTCQ. Generalization (of PA-ECTCQ) to predictive PA-ECTCQ for Markov sources is presented in Section 6.3. Along presenting PA-ECTCQ and predictive PA-ECTCQ, their codebook search algorithms and extensive simulation results are also described. In Section 6.4, we study the PA-ECTCQ performance in the presence of source mismatch conditions. We will also consider approaches of adapting the underlying ECTCQ codebook structure in accordance with the knowledge gradually acquired by learning the statistical properties of the input source sequence. Finally, a summary and conclusions are presented in Section 6.5.

6.2 Buffer-Instrumented Quantizers

The scenario of this section is as follows. We describe AECQ and a relatively simple approach of implementing this buffer-instrumented quantizer in Section 6.2.1. In

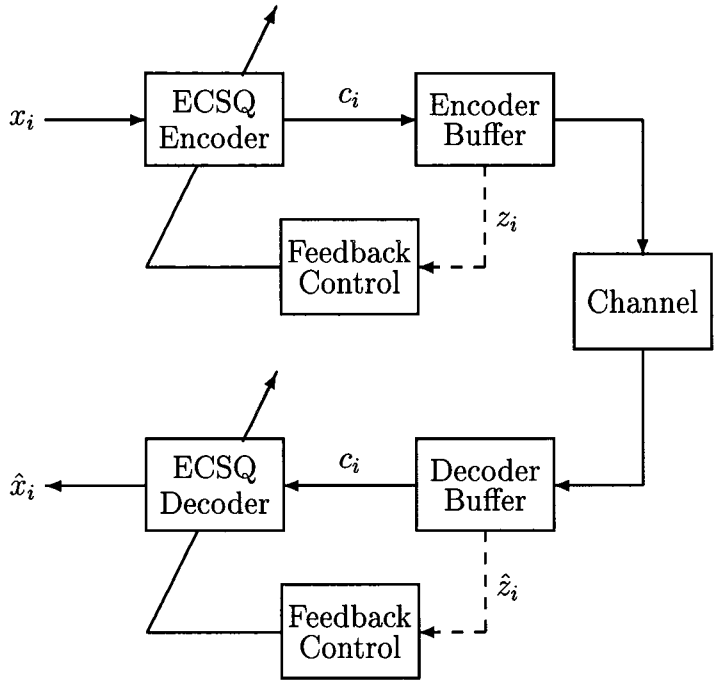


Figure 6.1: Block diagram of adaptive entropy-coded quantizer (AECQ).

Section 6.2.2, we first describe a naive buffer-instrumented variation of ECTCQ and point out its drawback of operating the feedback control in a delayed manner. We then describe PA-ECTCQ and present its codebook search algorithm. Simulation results for these quantization schemes are provided and discussed in Section 6.2.3.

6.2.1 AECQ

The block diagram of AECQ is illustrated in Figure 6.1. We will assume that the channel operates at a rate of r bits/sample, the buffer size is B bits, and the underlying entropy encoder is a 1^{st}-order encoder in the sense that it operates on one quantization index at a time. Generalization to a block-based entropy encoder is straightforward (see [46]) but, for simplicity, will be omitted.

We now describe AECQ. At the i^{th} encoding cycle, the channel codeword c_i of m_i bits produced from encoding x_i is delivered as input to the buffer while

simultaneously r bits are released for transmission over the channel. The encoder buffer state after this encoding cycle, denoted by z_i, is hence recursively updated according to $z_i = z_{i-1} + m_i - r$. The basic idea of AECQ is to adapt the encoder characteristics (base on feedback information of z_{i-1}) before encoding x_i such that the buffer state z_i at the end of the i^{th} encoding cycle is regulated as close to the desired "half-full" state ($B/2$) as possible. In the decoder side, suppose that the decoder buffer state $\hat{z}_{i-1}$ can address to a synchronous mode as is assumed in the encoder. The entropy decoder hence is able to correctly decode c_i (of m_i bits). At the end of the i^{th} decoding cycle, the decoder buffer is updated according to $\hat{z}_i = \hat{z}_{i-1} - m_i + r$. Clearly, if the initial states z_{-1} and $\hat{z}_{-1}$ are both set to $B/2$, z_i and $\hat{z}_i$ should be symmetric about $B/2$[1]. According to this property, synchronous operation between the AECQ encoder and decoder can thus be achieved without any side information.

While a number of strategies for adapting the encoder characteristics (see [46]) are possible, one that is relatively simple to implement and has been shown to provide quite effective rate-distortion performance is as follows. For most useful sources, the p.d.f.'s are decreasing in magnitude of the argument. Besides, the entropy encoder is such that input samples of smaller (larger) magnitudes are generally encoded into codewords of shorter (longer) lengths. Based on these observations, instead of actually modifying the encoder characteristics, the source sample x_i is pre-divided by a feedback signal $f(z_{i-1} - B/2)$ before being encoded where $f(\cdot)$ is a monotone increasing function with $f(0) = 1$. An example of such

[1] z_{-1} and $\hat{z}_{-1}$ can be set to any number b with $0 < b < B$. In this case, z_i and $\hat{z}_i$ will be symmetric about b if $0 \leq \hat{z}_i \leq B$. However, if $b \neq B/2$, buffer overflow/underflow event may occur in the decoder while it is avoided in the encoder.

a function, which was used in [47] and will be used throughout this chapter, is

$$f(x) = \begin{cases} \frac{1}{1-\gamma/2}; & x > \min(K, B/2), \\ \frac{1}{1-\gamma x/2K}; & 0 \le x \le \min(K, B/2), \\ 1 + \gamma x/2K; & -\min(K, B/2) \le x \le 0, \\ 1 - \gamma/2; & x < -\min(K, B/2), \end{cases} \tag{6.1}$$

where $0 \le \gamma \le 2$ and K is a specified positive integer. The roles that γ and K play in determining the feedback characteristics are described in [46] and are omitted here. The decoder can track the buffer occupancy history and hence yield the feedback signal $f(z_{i-1} - B/2)$. The replica $\hat{x}_i$ for x_i is then constructed by multiplying $f(z_{i-1} - B/2)$ to the entropy decoded symbol.

6.2.2 Buffer-Instrumented ECTCQ

In a strict sense, AECQ is a scalar quantizer and hence fails to capitalize on the granular gain. Based on the AECQ idea of buffer-instrumented adaptation, it is straightforward to generalize the underlying entropy-coded scalar quantizer to the entropy-coded vector quantizer so as to realize the granular gain. The resulting quantization scheme, however, is generally non-structured and could be prohibitive for a large block-length or encoding rate. As the more structured ECTCQ is a simpler alternative than ECVQ to realize the granular gain, we next consider adaptive buffer-instrumented entropy-coded ECTCQ.

Let us first consider a naive adaptive ECTCQ (A-ECTCQ) which replaces the underlying ECSQ with ECTCQ. There is no direct interaction between ECTCQ and the feedback control mechanism in this naive coding scheme. We should also mention that there is an inherent encoding delay d (in samples) associated with the ECTCQ codebook search algorithm (see Section 5.2.1). Based on the buffer state

z_{i-1}, the feedback control hence adjusts the magnitude of the source sample x_{i+d} also in a delayed manner. By contrast, the AECQ feedback control can adjust the magnitude of x_i "instantaneously". ECTCQ (where the buffer adaptaton is absent) generally requires a sufficiently large quantization delay to realize a significant granular gain (see, for example, Figure 5.4). In A-ECTCQ, however, a large delay could possibly deteriorate the effectiveness of the feedback control as the delayed feedback control cannot adjust the magnitudes of the intermediate samples which have already been encoded in the search path (and whose associated codewords are already determined to be delivered to the channel buffer).

It should be mentioned that the released ECTCQ reproduction sequence is a delayed version of one of the candidate sequences stored in the search paths of the ECTCQ codebook search algorithm. We have earlier used this fact to derive the predictice ECTCQ codebook search algorithm (see Section 5.3.1) which pathwise performs the DPCM predictive coding operation. Similar idea gives rise to a modified buffer-instrumented ECTCQ — dubbed *pathwise adaptive* ECTCQ (PA-ECTCQ) — that "pathwise" performs the buffer-instrumented adaptation.

We now describe the PA-ECTCQ codebook search algorithm which, like that for predictive ECTCQ, is a variation of the ECTCQ codebook search algorithm presented in Section 5.2.1. This new quantization scheme is based on the fact that the actual buffer occupancy z_i after having delivered into buffer the binary stream associated with all the intermediate symbols up to time instant i in the released sequence can be predicted. Denote by Δ_i^s the minimum accumulated nonstationarily scaled distortion[2] that results when the first i source samples are quantized to a reproduction sequence reaching trellis state $s \in \boldsymbol{\Sigma}$. Also, let z_i^s

[2]This is the biased squared-error distortion scaled nonstationarily by the feedback control.

denote the actual buffer state if the reproduction sequence associated with Δ_i^s is the released sequence. The problem of selecting one of the two possible survivor paths at time $i-1$ extending to trellis state s at time i (i.e., determining Δ_i^s) can be written as

$$(s^*, y^*) = \arg\min_{(s', y')} \Delta_{i-1}^{s'} + \xi(\frac{x_i}{f(z_{i-1}^{s'} - B/2)}, y'), \tag{6.2}$$

where s' ranges between two possible previous states for s, $f(\cdot)$ is the feedback control mapping, and y' is the closest point to the pathwise scaled input sample within the codebook subset allowed for the trellis transition from s' to s. The buffer occupancy z_i^s is then updated according to

$$z_i^s = z_{i-1}^{s^*} - r + \ell(y^*), \tag{6.3}$$

where r is the channel operating rate in bits/sample and $\ell(y^*)$ denotes the codeword length (in bits) generated by the entropy encoder for the reproduction symbol y^*. To prevent the search path from yielding the buffer overflow/underflow event, we should place an extra constraint on y' in solving (6.3) as $0 \leq z_{i-1}^{s'} - r + \ell(y') \leq B$. The decoder can track the buffer occupancy history and hence yield the feedback signal $f(z_{i-1}^{s^*} - B/2)$. The reproduction symbol $\hat{x}_i$ for x_i is then obtained by multiplying $f(z_{i-1}^{s^*} - B/2)$ to the decoded symbol y^*. This PA-ECTCQ codebook search algorithm is summarized in Figure 6.2.

6.2.3 Simulation Results

The A-ECTCQ and PA-ECTCQ performance for some memoryless generalized Gaussian sources are presented and discussed in this subsection. All numerical results to be presented are obtained based on simulation performed on test sequence

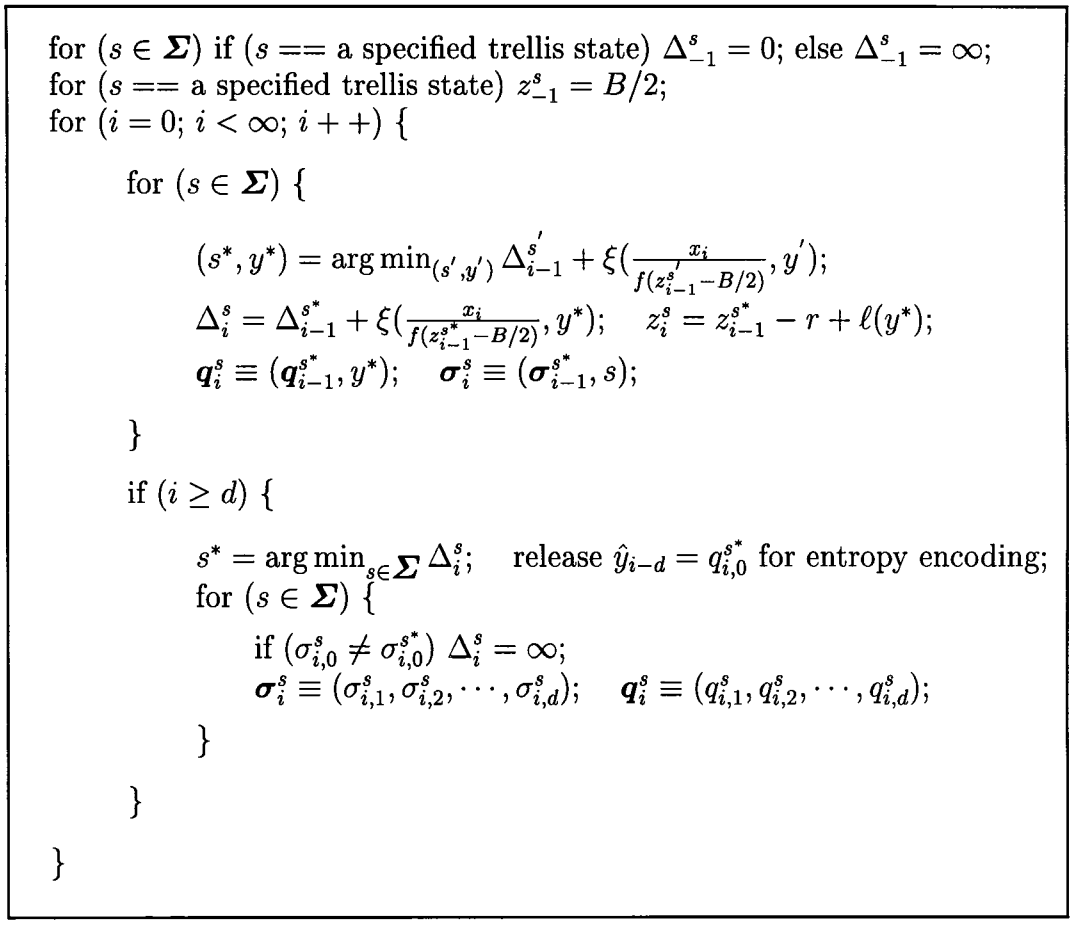


Figure 6.2: PA-ECTCQ codebook search algorithm.

of at least 10^6 samples. We should mention that both quantzition schemes can be considered as fixed-rate quantizers and their encoding rate is equavalent to the average channel operating rate. The underlying ECTCQ is chosen as the one whose output entropy is closest to the encoding rate. Besides, it is possible to achieve a nonintegral encoding rate by time sharing of two integral channel operating rates.

The buffer-state histograms of A-ECTCQ and PA-ECTCQ for some specific encoding delays on encoding the Gaussian source at 3 bits/sample are illustrated in Figure 6.3. In simulations leading to these results, the underlying trellis code is Ungerboeck's 8-state trellis code, the buffer size $B = 192$ (in bits), the feedback

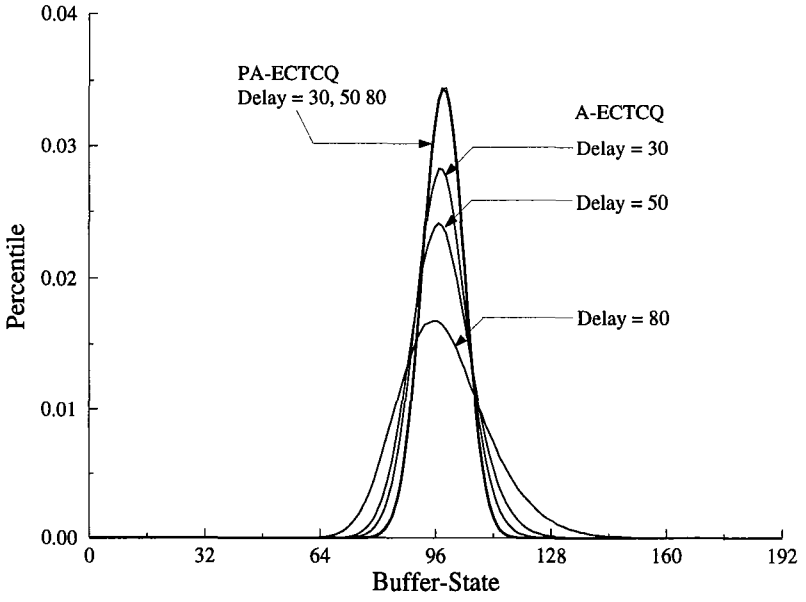


Figure 6.3: Buffer occupancy histogram of A-ECTCQ and PA-ECTCQ on encoding a memoryless Gaussian source at 3 bits/sample.

control is such that $K = 96$ and $\gamma = 2.0$, and the entropy encoder is a 2^{nd}-order Huffman encoder. In A-ECTCQ, a larger delay results in a wider dispersion in the buffer-state distribution — which suggests a higher probability of buffer overflow/underflow[3]. By contrast, the PA-ECTCQ buffer-state histograms are somewhat indistinguishable among various choices of the encoding delay and have a relatively narrow dispersion in distribution. The A-ECTCQ and PA-ECTCQ normalized mean squared-error (MSE) performance as a function of the delay are given in Figure 6.4. We can observe that in PA-ECTCQ a larger encoding delay does yield a better performance while this is not true for A-ECTCQ. These results indicate the given the same buffer size and feedback control mechanism

[3]We have observed from our experiments that for A-ECTCQ the buffer overflow/underflow event begins to occur at an encoding delay of 90 samples.

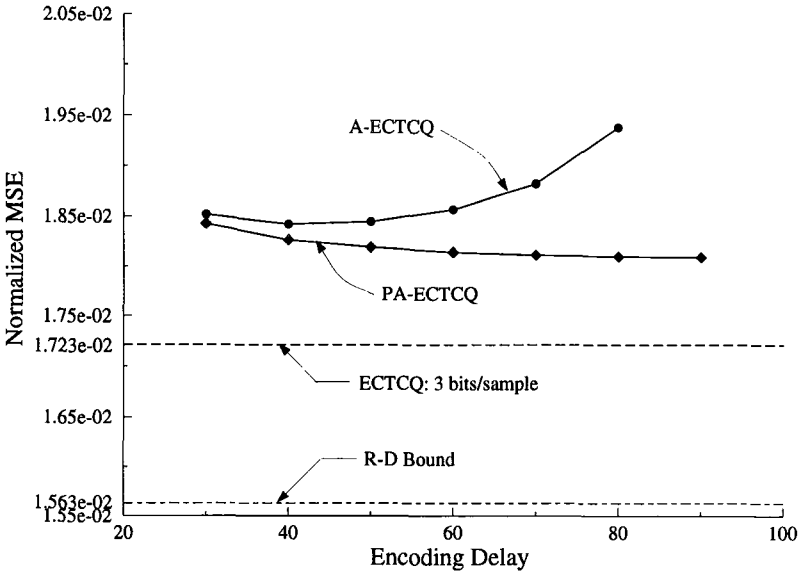
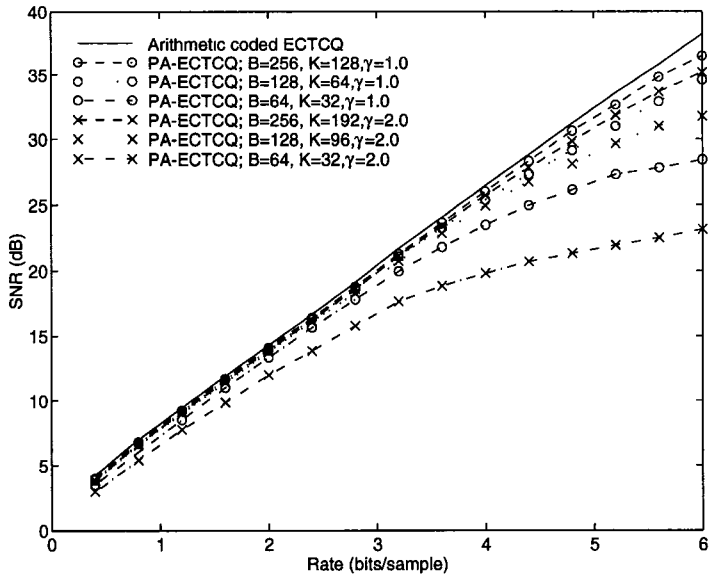


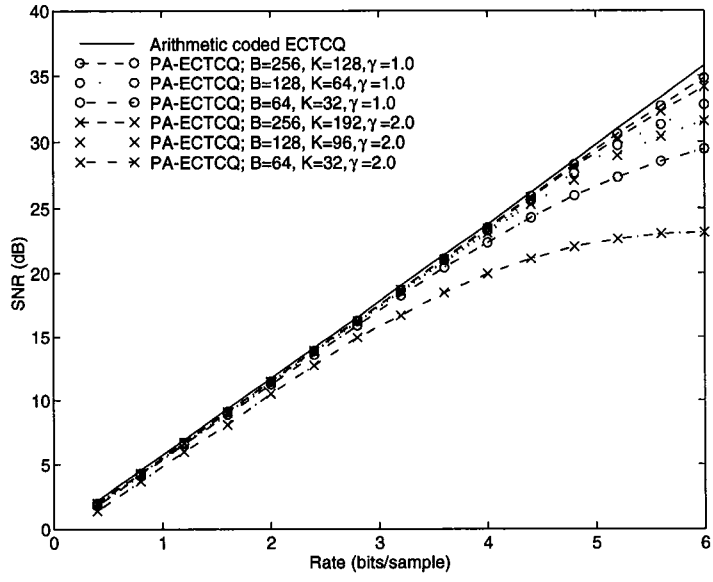
Figure 6.4: A-ECTCQ and PA-ECTCQ performance (in normalized MSE) as a function of the delay d (in source samples) on encoding a memoryless Gaussian source at 3 bits/sample.

PA-ECTCQ is more effective in avoiding the buffer underflow/overflow event than A-ECTCQ. Besides, like ECTCQ, PA-ECTCQ performance can be improved by increasing the encoding delay.

Since PA-ECTCQ outperforms A-ECTCQ in almost all respects, we will in the sequel skip A-ECTCQ and concentrate only on PA-ECTCQ (with a sufficiently large encoding delay). The performance of PA-ECTCQ on encoding generalized Gaussian sources of shape parameters 0.6 and 2.0 are presented in Figure 6.5. Here we have adopted the more efficient arithmetic encoder (than the Huffman encoder) as the underlying entropy encoder. For reference, we have also included in Figure 6.5 the non-feedback arithmetic coded ECTCQ performance. The results indicate that the buffer-instrumented adaptation, while regulating the variable-



(a)



(b)

Figure 6.5: PA-ECTCQ performance (SNR in dB) for memoryless generalized Gaussian sources with shape parameter (a) $\theta = 0.6$ and (b) $\theta = 2.0$.

Source	Rate	Trellis States 1	4	8	16	32	$R(D)$
Gaussian	1.0	4.63	5.50	5.58	5.65	5.70	6.02
	2.0	10.56	11.48	11.59	11.63	11.71	12.04
	3.0	16.57	17.51	17.59	17.65	17.71	18.06
Laplacian	1.0	5.80	6.20	6.29	6.37	6.41	6.62
	2.0	11.40	12.12	12.19	12.24	12.36	12.66
	3.0	17.27	18.12	18.17	18.19	18.36	18.68

Table 6.1: PA-ECTCQ performance (SNR in dB) on encoding memoryless Gaussian and Laplacian sources.

rate codewords to the fixed-rate channel[4], induces some certain performance loss — represented by the performance gap between PA-ECTCQ and non-feedback arithmetic coded ECTCQ. Generally speaking, a larger buffer size or smaller value of γ can yield better PA-ECTCQ performance. For a fixed buffer size and feedback function, the performance loss is somewhat more significant for the broader tailed sources.

Presented in Table 6.1 are the performance (SNR in dB) of PA-ECTCQ using a relatively large buffer size on encoding memoryless Gaussian and Laplacian sources at various encoding rates. In simulations leading to these results, the buffer size is $B = 1024$ bits, the feedback control is specified by $K = 768$ and $\gamma = 2.0$, and the underlying entropy encoder is an arithmetic encoder. We should mention that the scheme with 1-state trellis is actually AECQ (where the underlying quantizer is the UTQ). We have noticed that PA-ECTCQ performs very close to ECTCQ (or UTQ). This could be due to the fact that the buffer size B is sufficiently large for these specific cases.

[4]In all cases we considered, the buffer overflow/underflow event has been avoided.

6.3 Predictive Buffer-Instrumented Quantizers

In this section, we present a novel quantizer — called predictive PA-ECTCQ — that is a generalization of PA-ECTCQ for quantizing Markov sources. The way we generalize PA-ECTCQ to predictive PA-ECTCQ is exactly the same as we generalize ECTCQ to predictive ECTCQ (see Chapter 5). The basic idea is to further incorporate the DPCM predictive coding operation in each search path of the PA-ECTCQ codebook search algorithm.

We next present the predictive PA-ECTCQ codebook search algorithm, which is an amalgamation of the DPCM and PA-ECTCQ codebook search algorithms. For simplicity and continuity of presentation, we will use the PA-ECTCQ codebook search algorithm as a baseline and describe only extra efforts required to exploit the source correlation. Like the codebook search algorithm for predictive ECTCQ, we assign for the survivor search path reaching trellis state $s \in \boldsymbol{\Sigma}$ at time $i-1$ a buffer that stores the corresponding p most recent quantized output samples, denoted by $\hat{\boldsymbol{x}}_{i-1}^{s} \equiv (\hat{x}_{i-p}^{s}, \cdots, \hat{x}_{i-2}^{s}, \hat{x}_{i-1}^{s})$. Based on $\hat{\boldsymbol{x}}_{i-1}^{s}$, a pathwise prediction $\tilde{x}_i^s$ of x_i can be made according to (5.6) and the corresponding prediction residual is $x_i - \tilde{x}_i^s$. Replacing the source sample x_i with the pathwise prediction residual $x_i - \tilde{x}_i^{s'}$ for some previous state s' of s, the minimization in (6.2) becomes

$$(s^*, y^*) = \arg\min_{(s', y')} \Delta_{i-1}^{s'} + \xi\left(\frac{x_i - \tilde{x}_i^{s'}}{f(z_{i-1}^{s'} - B/2)}, y'\right), \tag{6.4}$$

where s' ranges between two possible previous states for s, $f(\cdot)$ is the feedback control mapping, and y' is the closest point to the pathwise scaled prediction residual within the codebook subset allowed for the trellis transition from s' to s. The decoder can track both $\tilde{x}_i^{s^*}$ and $f(z_{i-1}^{s^*} - B/2)$. The replica $\hat{x}_i$ for x_i is therefore obtained as the sum of $\tilde{x}_i^{s^*}$ and the entropy decoded symbol y^* scaled by

for $(s \in \boldsymbol{\Sigma})$ if ($s ==$ a specified trellis state) $\Delta_{-1}^{s} = 0$; else $\Delta_{-1}^{s} = \infty$;
for ($s ==$ a specified trellis state) $z_{-1}^{s} = B/2$;
for $(i = 0;\ i < \infty;\ i++)$ {

for $(s \in \boldsymbol{\Sigma})$ $\tilde{x}_i^s = \sum_{j=1}^{p} \rho_j \hat{x}_{i-j}^s$;
for $(s \in \boldsymbol{\Sigma})$ {

$(s^*, y^*) = \arg\min_{(s',y')} \Delta_{i-1}^{s'} + \xi(\frac{x_i - \tilde{x}_i^{s'}}{f(z_{i-1}^{s'} - B/2)}, y')$;

$\Delta_i^s = \Delta_{i-1}^{s^*} + \xi(\frac{x_i - \tilde{x}_i^{s^*}}{f(z_{i-1}^{s^*} - B/2)}, y^*)$; $\quad z_i^s = z_{i-1}^{s^*} - r + \ell(y^*)$;
$\hat{\boldsymbol{x}}_i^s \equiv (\hat{\boldsymbol{x}}_{i-1}^{s^*}, \tilde{x}_i^{s^*} + f(z_{i-1}^{s^*} - B/2) \cdot y^*)$;
$\boldsymbol{q}_i^s \equiv (\boldsymbol{q}_{i-1}^{s^*}, y^*)$; $\quad \boldsymbol{\sigma}_i^s \equiv (\boldsymbol{\sigma}_{i-1}^{s^*}, s)$;

}

if $(i \geq d)$ {

$s^* = \arg\min_{s \in \boldsymbol{\Sigma}} \Delta_i^s$; $\quad$ release $\hat{y}_{i-d} = q_{i,0}^{s^*}$ for entropy encoding;
for $(s \in \boldsymbol{\Sigma})$ {

if $(\sigma_{i,0}^s \neq \sigma_{i,0}^{s^*})$ $\Delta_i^s = \infty$;
$\boldsymbol{\sigma}_i^s \equiv (\sigma_{i,1}^s, \sigma_{i,2}^s, \cdots, \sigma_{i,d}^s)$; $\quad \boldsymbol{q}_i^s \equiv (q_{i,1}^s, q_{i,2}^s, \cdots, q_{i,d}^s)$;

}

}

}

Figure 6.6: Predictive PA-ECTCQ codebook search algorithm.

$f(z_{i-1}^{s^*} - B/2)$; that is,

$$\hat{x}_i = \tilde{x}_i^{s^*} + f(z_{i-1}^{s^*} - B/2) \cdot y^*. \tag{6.5}$$

To continue the pathwise predictive operation, the buffer $\hat{\boldsymbol{x}}_i^s$ should be updated according to $\hat{\boldsymbol{x}}_i^s \equiv \left(\hat{\boldsymbol{x}}_{i-1}^{s^*}, \tilde{x}_i^{s^*} + f(z_{i-1}^{s^*} - B/2) \cdot y^*\right)$. For completeness, we have summarized the predictive PA-ECTCQ codebook search algorithm in Figure 6.6.

The predictive PA-ECTCQ performance (SNR in dB) on encoding two specific Gauss-Markov sources are presented in Table 6.2. In simulatons (performed on

Source	Rate	Trellis States 1	4	8	16	32	$R(D)$
	1.0	10.39	11.91	12.16	12.47	12.54	13.23
AR(1):	2.0	17.35	18.21	18.38	18.50	18.66	19.25
$\rho_1 = 0.9$,	3.0	23.64	24.20	24.30	24.55	24.74	25.27
AR(2):	1.0	9.18	13.37	13.67	13.92	14.12	14.96
$\rho_1 = 1.515$	2.0	19.39	20.38	20.64	20.84	20.99	21.64
$\rho_2 = -0.752$	3.0	25.85	26.37	26.65	26.86	26.98	27.66

Table 6.2: Predictive PA-ECTCQ performance (SNR in dB) on encoding two specific Gauss-Markov sources.

test sequence of at least 10^6 samples) leading to these results, the buffer size is $B = 1024$ bits, the feedback control is specified by $K = 768$ and $\gamma = 2.0$, and the underlying entropy encoder is an arithmetic encoder. In terms of approaching the rate-distortion bound, the predictive PA-ECTCQ performance for Markov sources is relatively about 0.4 dB inferior to that of PA-ECTCQ for memoryless sources. This could be due the suboptimality of the predictive ECTCQ codebook search algorithm. Nevertheless, predictive PA-ECTCQ still performs very close to the rate-distortion limit and significantly outperforms the other fixed-rate quantizers such as predictive and precoded TB-SVQs considered in Chapters 3 and 4.

6.4 Source Mismatch Issues

An important issue in assessing the practical utility of adaptive entropy-coded quantization schemes is the inherent *robustness* of these techniques. That is, the sensitivity of the resulting performance to various deviations from nominal design conditions. In this section, we assess the robustness of PA-ECTCQ. We will present

in Section 6.4.1 simulation results of the PA-ECTCQ performance in the presence of source mismatch conditions. We must mention that PA-ECTCQ can only adapt its reproduction levels. In Section 6.4.2, we describe an approach of adapting the underlying entropy encoder in accordance with the knowledge gradually acquired by learning the statistical properties of the input source sequence.

6.4.1 PA-ECTCQ Performance

We will confine our attention to the class of generalized Gaussian sources whose p.d.f.'s can be described as in (2.1). Within this class, the p.d.f. $p_0(x)$ of the nominal source is such that the shape parameter $\theta = \theta_0$ and the variance $\sigma^2 = \sigma_0^2$.

Mismatch to Scale

Mismatch with respect to scale occurs when the p.d.f. $p(x)$ of the input source differs from the nominal p.d.f. $p_0(x)$ only in the source variance. That is, the actual source p.d.f. is described by $p(x) = (\sigma_0/\sigma)p_0((\sigma_0/\sigma)x)$. We define $\zeta = (\frac{\sigma}{\sigma_0})^2$ to represent the scale mismatch parameter.

The PA-ECTCQ performance (in normalized MSE) on encoding a memoryless Gaussian source of a mismatched scale (parameterized by ζ) at 3 bits/sample is illustrated in Figure 6.7. In simulations leading to these results, the underlying ECTCQ (with an 8-state trellis) is designed for the nominal Gaussian source at 3 bits/sample, the encoding delay d is 100 samples, $K = B/2$, and the entropy encoder is a 2^{nd}-order Huffman encoder. For comparison, performance for Lloyd-Max quantizer (LMQ) and ECTCQ, both matched to the source scale, and the rate-distortion bound are also included. For a large buffer size and a large value of γ, the PA-ECTCQ performance is insensitive to a rather wide range of source scale mismatch. For a fixed buffer size, the buffer overflow/underflow immunity

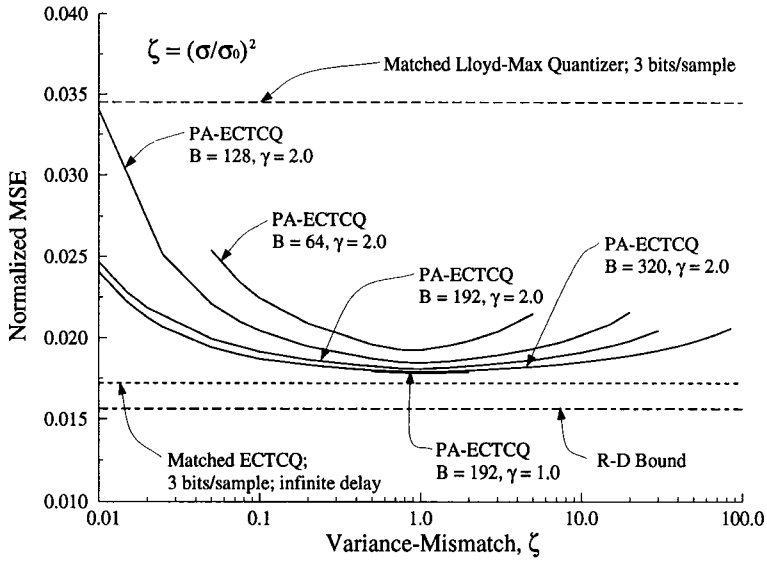


Figure 6.7: PA-ECTCQ performance (in normalized MSE) in the presence of source scale mismatch.

can be improved with an increased value of γ, but at the cost of some increased distortion.

We have also observed in our experiments that the source scale mismatch results in an increase (decrease) in the average occupancy level from the desired point $B/2$ for $\zeta > 1$ ($\zeta < 1$). This phenomenon is very similar to that for AECQ subject to source scale mismatch [47] and can be explained likewise.

Mismatch to Shape

In this case, the source has the same variance σ_0^2 as in the nominal design. But the shape (exponential decay) parameter θ is different from the nominal shape parameter θ_0.

In Figure 6.8, we illustrate the PA-ECTCQ performance (in normalized MSE) for test sources of various shape parameters θ; the underlying ECTCQ (with an 8-state trellis) is designed for the nominal Gaussian source ($\theta_0 = 2.0$) at 3

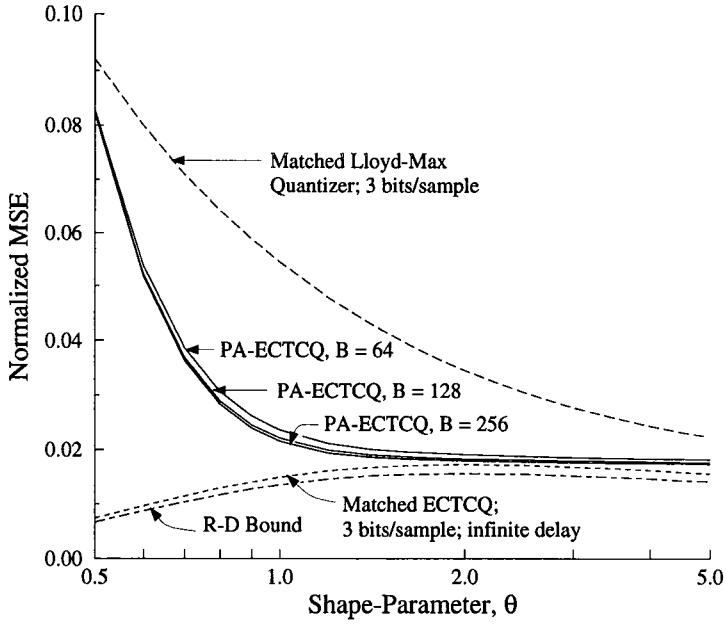


Figure 6.8: PA-ECTCQ performance (in normalized MSE) in the presence of source shape mismatch.

bits/sample, the encoding delay d is 100 samples, the feedback control is such that $K = B/2$ and $\gamma = 2.0$, and the entropy encoder is a 2^{nd}-order Huffman encoder. Also included are the rate-distortion limit and the performance for LMQ and ECTCQ, both matched to each value of θ. We can observe from Figure 6.8 that the PA-ECTCQ performance can be improved by increasing the buffer size. In all cases we considered, PA-ECTCQ consistently outperforms LMQ. Also, the PA-ECTCQ performance is rather insensitive to source shape mismatch for sources with shape parameters close to the nominal shape parameter ($\theta_0 = 2.0$). For sources with broader-tailed distributions ($\theta < 2.0$), there is still a significant gap between the PA-ECTCQ performance and rate-distortion bound.

The average buffer occupancy state under shape mismatch is always shifted somewhat to the left. As was explained in [47], this is because the p.d.f. of the

memoryless Gaussian source ($\theta_0 = 2$) has the highest differential entropy for a given variance. The histograms for broad-tailed distributions ($\theta < 2.0$) generally have wider buffer-state dispersions. For more narrow-tailed distributions ($\theta > 2.0$) the dispersion decreases and the buffer occupancy histogram is highly peaked. Explanations for such phenomena can also be found in [47].

6.4.2 Shape-Adjusting PA-ECTCQ

Simulation results in Section 6.4.1 indicate that the PA-ECTCQ performance is robust in the presence of source scale and, to a lesser extent, shape mismatch conditions. In this subsection, we attempt to improve its robustness to source shape mismatch conditions. We will specifically provide several heuristic approaches that could help shortening the significant performance gap between PA-ECTCQ and the rate-distortion bound for sources with broad-tailed distributions, which is demonstrated shown in Figure 6.8.

First, we can employ the arithmetic encoder as the underlying entropy encoder in PA-ECTCQ. Simulation results for such a modified PA-ECTCQ (with $B = 256$, $K = 192$, and other parameters equal to those in Figure 6.8) are presented in Figure 6.9. The resulting PA-ECTCQ performance is slightly superior to those (with similar buffer size and feedback control function) given in Figure 6.8. This should be credited to that the arithmetic encoder has a smaller coding redundancy than the Huffman encoder. However, the performance gap between this modified PA-ECTCQ and rate-distortion bound for sources with broad-tailed distributions is still significant. We will take such a PA-ECTCQ as a baseline; all other schemes to be discussed in the sequel will assume the same buffer size, number of trellis states, and feedback control function.

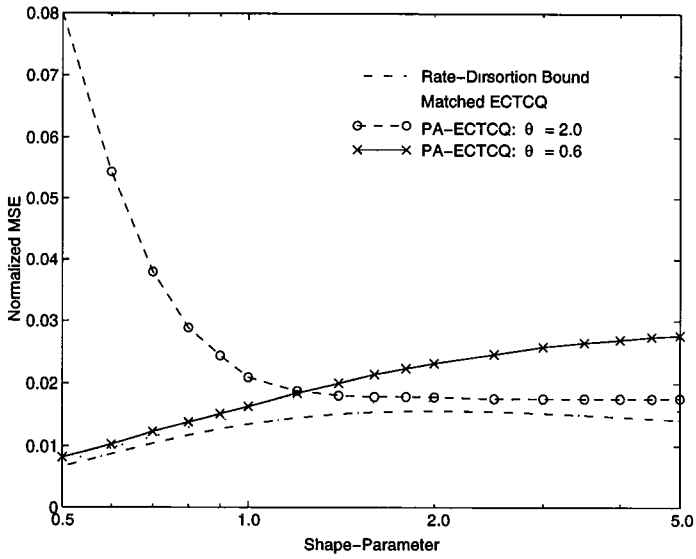


Figure 6.9: PA-ECTCQ performance (in normalized MSE) in the presence of source shape mismatch; the underlying ECTCQ is designed for nominal source of shape parameter 0.6 or 2.0.

It can be argued that for sources with broad-tailed distributions, samples with large magnitudes occur with a significant probability. The PA-ECTCQ designed for a nominal Gaussian ($\theta = 2.0$) source generally does not have a reproduction alphabet large enough to deal with the source samples with large magnitudes. Simulation results in [47] indicate that the AECQ performance in the presence of source shape mismatch condition can indeed be improved by using a larger set of reproduction symbols. To execute a similar experimentation for PA-ECTCQ, we need to design the ECTCQ with a somewhat expanded reproduction alphabet. In practice, we have observed that it is inefficient to serve this purpose[5] by applying the training sequence oriented ECTCQ design algorithm (given in Appendix A). On the other hand, the ECTCQ we have designed for the generalized Gaussian source with a shape parameter of 0.6 has a relatively large reproduction alphabet.

[5]We have later realized that it could be easier to design TBQ described in Chapter 5.

Simulation results for PA-ECTCQ based on such an ECTCQ codebook is given in Figure 6.9, from which we can assert that PA-ECTCQ generally performs well for sources whose shape parameters range around the shape parameter θ_0 assumed in the nominal design.

Since the ECTCQ designed for $\theta_0 = 0.6$ seems to have a reproduction alphabet large enough to cope with all the generalized Gaussian sources we considered (with $0.5 \leq \theta \leq 5.0$), we will implicitly assume its codebook structure in PA-ECTCQ. According to the simulation results in Figure 6.9, our new problem becomes that PA-ECTCQ does not perform well for sources with narrow-tailed distributions ($\theta > 2.0$). Actually, the PA-ECTCQ performance is even inferior to the LMQ performance for $\theta > 2$. We should mention that the adaptation we have been considering so far only modifies the ECTCQ reproduction levels; the underlying arithmetic encoder is not adaptive. We attempt to solve this new problem by making the arithmetic encoder adaptive based on an algorithm derived by Witten *et al.* [9]. In such an adaptive arithmetic encoding algorithm, the source p.d.f. is described by an empirical set of symbol occuring frequencies and is updated once immediately after each symbol is encoded. We now assign for each search path of the PA-ECCTCQ codebook search algorithm an adaptive arithmetic coder. The arithmetic coders are pathwise adapted based on increasing (by one) the occuring frequency of the symbol closest to the reproduction symbol. This new quantization scheme can adapt its underlying entropy encoder In some sense, this new adaptive quantizer attempts to adjust its underlying entropy encoder so as to match (its nominal shape parameter) to the input source based on the quantized output (which is also available to the decoder). We will call this quantization scheme *shape-adjusting* PA-ECTCQ.

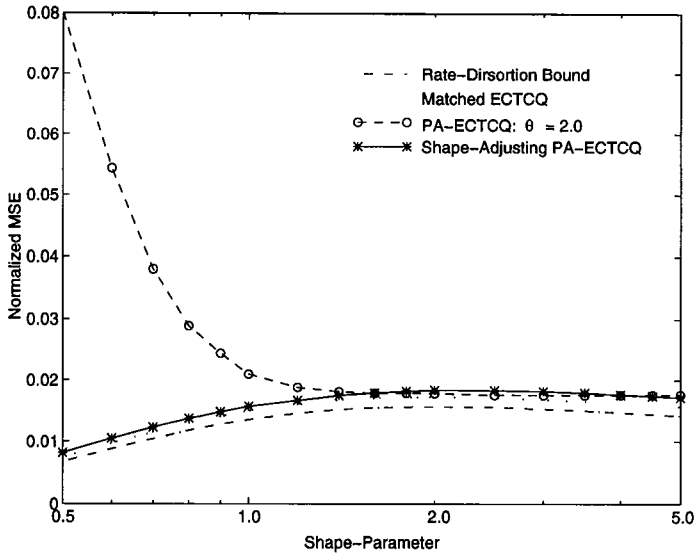


Figure 6.10: Shape-adjusting PA-ECTCQ performance (in normalized MSE) in the presence of source shape mismatch.

The performance (in normalized MSE) for shape-adjusting PA-ECTCQ in the presence of source shape mismatch conditions is illustrated in Figure 6.10. These results are obtained based on simulations performed on a test sequence of at least 10^6 samples. By comparing Figures 6.9 and 6.10, we can observe that for sources whose shapes are close to the nominal shape ($\theta_0 = 0.6$), PA-ECTCQ ($\theta_0 = 0.6$) and shape-adjusting PA-ECTCQ have about the identical performance. This fact suggests that entropy encoder adaptation is not so crucial in these cases. For narrow-tailed sources ($\theta > 2.0$), however, there is a significant performance improvement of shape-adjusting PA-ECTCQ over PA-ECTCQ ($\theta_0 = 0.6$). In all cases we considered, shape-adjusting PA-ECTCQ performs very close to ECTCQ — matched to all various source shapes. We can hence assert that shape-adjusting PA-ECTCQ performance is robust subject to a rather wide range of source shape mismatch conditions.

6.5 Summary and Conclusions

In this chapter we developed various extensions of AECQ [46]. To capitalize on the granular gain, we have combined the buffer-instrumented adaptation idea of AECQ [46] with ECTCQ of Chapter 5 and proposed a new quantizer called pathwise-adaptive ECTCQ (PA-ECTCQ). To exploit the memory gain for Markov sources, we combined the DPCM predictive coding operation with PA-ECTCQ and proposed predictive PA-ECTCQ. In both PA-ECTCQ and predictive PA-ECTCQ, the buffer-instrumentation adaptation has successfully regulated the variable-rate codewords through a fixed-rate communication channel and both quantizers are free of any buffer overflow/underflows. Given a large buffer size and an appropriate feedback control mechanism, the tradeoff is only some negligibly small additional quantization distortion.

Our simulation results indicate that the PA-ECTCQ performance is robust in the presence of source scale and, to a lesser extend, shape mismatch conditions. We have also considered adjusting the underlying entropy encoder in accordance with the quantized output and proposed shape-adjusting PA-ECTCQ. The shape-adjusting PA-ECTCQ performance has been shown to be robust to a rather wide range of source shape mismatch conditions.

It is likely impossible to find a unique statistic model well matched universally to all types of real-world sourecs. In the quantization process of any analog source digitization system, there hence exist more or less source mismatch conditions (see [36]). We have demonstrated the potential of PA-ECTCQ in providing efficient and robust performance in the context of image transform coding [36]-[37].

There are two open problems related to the quantizers described in this chapter. The first is that of determining in some sense the optimal feedback function for

a given unknown source mismatch condition and buffer size. We should mention that quantizers described in this chapter are slideing-block coders and operate in the sequence space. Although we consider these quantization schemes as fixed-rate quantizers, their inherent entropy coding nature leaves them still vulnerable and sensitive to channel noise. The second problem is to prevent a single channel error from causing catastrophic reconstruction errors (also called *error propagation*). One possibility is to convert these sequence-based quantizers into block-constrained schemes like those considered by Balamesh and Neuhoff [48]. We are currently in search of methods implementing quantizers described in this chapter on a block basis. The objective is to derive a quantizaton scheme that is block-constrained (hence free of error propagation problem) and can capitalize on all the possible vector quantization gains.

Chapter 7

Conclusions and Related Work

7.1 Summary and Conclusions

The implementation complexity of the optimal vector quantizer is unaffordable even for quantization at low rates and moderate block-lengths. To overcome the complexity problem, in this thesis we developed structured resolution-constrained or entropy-constrained quantization schemes for quantizing stationary memoryless or Markov sources.

According to high-rate quantization theory [14]-[13], vector quantizers whose quantization cells are approximately spherical can realize the granular gain over scalar quantizers whose quantization cells are rectangular. There exist the so-called coset codes [29]-[30] that have been shown to be very useful in both quantization and transmission problems. Our motivation for developing structured quantizers based on coset codes are twofold: (i) the Voronoi regions (quantization cells) are approximately spherical; and (ii) their inherent algebraic sructures help in reducing the implementation complexity. In Chapter 2, we provided preliminary materials

which include a brief review of coset codes. We should mention that coset codes include trellis codes [22] and lattices [39] as the special cases. During the course of this thesis, we have more often used trellis codes than lattices. This is because that for a given complexity trellis codes are more efficient in capitalizeing on the granular gain than lattices. We have also described in Chapter 2 some symmetry properties of Ungerboeck's 1-D trellis codes [22] that were used in quantization schemes presented in Chapters 3-6.

The trellis-based scalar-vector quantizer (TB-SVQ) is a state-of-the-art efficient resolution-constrained quantizer recently proposed by Laroia and Farvardin [15]. In Chapter 3 we presented several variations on TB-SVQ. First, for memoryless sources we considered a hybrid of two TB-SVQs described in [15]. This TB-SVQ is simpler to design and has about the same performance as those in [15]. Second, for Markov sources TB-SVQ was generalized to predictive TB-SVQ that combines the DPCM predictive coding opration with TB-SVQ. We have also derived a simpler suboptimal codebook search algorithm — called the state-suppressed codebook search algorithm — for these two new quantization schemes.

In Chapter 4 we discussed the duality between quantizing Markov sources and transmitting data over ISI channels. The precoding idea of Laroia *et al.* [31] that solves the problem of realizing both shaping and coding gains for transmission over ISI channels was also described. Due to the quantization/transmission duality, we then combined this precoding idea with TB-SVQ. The resulting quantization scheme — dubbed precoded TB-SVQ — is asymptotically optimal and can, in principle, approach the rate-distortion bound for Markov sources. The granular gain is realized by the underlying trellis code while the combination of the SVQ structure and precoder provides the boundary gain. At low rates, the precoding

error is significantly large and will disturb the optimal codebook boundary (hence reduce the boundary gain). Two modified precoding schemes recently preposed by Laroia [45] were presented in their simplest form based on Ungerboeck's 1-D trellis code. We have employed these two modified precoders to improve the precoded TB-SVQ performance.

In Chapter 5 we developed structured entropy-coded quantization schemes. We first described a novel entropy-constrained trellis-coded quantization (ECTCQ) scheme that has a symmetric reproduction alphabet. Simulation results indicate that placing such a symmetry constraint costs no rate-distortion performance loss and can provide relatively simple implementation vis-à-vis the ECTCQ reported in the literature [28, 32]. We have also presented predictive and precoded ECTCQs that are extensions of ECTCQ for quantizing Markov sources. The way ECTCQ is extended to predictive or precoded ECTCQ is similar to that of extending TB-SVQ to predictive or precoded TB-SVQ described in Chapter 3 or 4, respectively. We also introduced coset-based quantizer (CBQ) that is in essence an extension of uniform scalar quantizer (USQ) to the multidimensional space. Since the underlying coset codes are more explicitly utilized in specifying the quantizer structure, CBQ is simpler to design and implement than ECTCQ.

In Chapter 6 we have combined the buffer-instrumented adaptation idea of AECQ [46] with ECTCQ of Chapter 5 and proposed a new quantizer called pathwise-adaptive ECTCQ (PA-ECTCQ). To exploit the memory gain for Markov sources, we combined the DPCM predictive coding operation with PA-ECTCQ, leading to predictive PA-ECTCQ. Both quantizers have successfully regulated the variable-rate ECTCQ output codewords through a fixed-rate communication channel and are free of the buffer overflow/underflow problem. Given a large buffer size

and an appropriate feedback control mechanism, the feedback control costs only negligibly small additional quantization distortion. The PA-ECTCQ performance has been shown to be robust in the presence of source scale and, to a lesser extend, shape mismatch conditions. We have also considered adjusting the underlying entropy encoder in accordance with the quantized output (a replica for the input source sequence) and proposed shape-adjusting PA-ECTCQ. The shape-adjusting PA-ECTCQ performance has been shown to be robust to a rather wide range of source shape mismatch conditions.

7.2 Related Work

To establish the usefulness of the quantization schemes described in the thesis on real-world sources we have applied them to problems in quantization of speech and image data. In [33], predictive TB-SVQ is used to efficiently quantize the speech waveform at low bit rates. The performance is significantly better than that obtained by predictive TCQ in [49]. For high-rate quantization of speech data, predictive TB-SVQ becomes too complicated while the simpler precoded TB-SVQ can provide a relatively equivalent performance [34]. Due to its efficient rate-distortion performance, ECTCQ has been applied in the context of transform image coding [35, 37]. There always exist more or less source mismatch conditions in a transform image coding system (see [36]). We demonstrated the potential of PA-ECTCQ in providing relatively efficient and robust performance in problems of quantizing image transform coefficients [36]-[37].

Appendix A

Magnitude-Based ECTCQ Design Algorithm

An entropy-constrained quantizer can be expressed in terms of the configuration provided in Figure 2.1. At each time instant, α maps the input random vector $\boldsymbol{X}$ to a quantization index $\alpha(\boldsymbol{X})$. An ideal entropy encoder γ then converts $\alpha(\boldsymbol{X})$ into a binary codeword $\gamma(\alpha(\boldsymbol{X}))$ whose length is denoted by $|\gamma(\alpha(\boldsymbol{X}))|$. One is generally interested in finding the triplet (α, γ, β) that minimizes the expected distortion between $\boldsymbol{X}$ and its reproduction $\beta(\alpha(\boldsymbol{X}))$ subject to a constraint on the quantizer output entropy $E[|\gamma(\alpha(\boldsymbol{X}))|]$. This constrained minimization problem can be solved by minimizing the Lagrangian functional

$$J_\lambda(\alpha, \gamma, \beta) = E\left[[\boldsymbol{X} - \beta(\alpha(\boldsymbol{X}))]^2 + \lambda|\gamma(\alpha(\boldsymbol{X}))|\right], \tag{A.1}$$

where $\lambda \geq 0$ is the Lagrangian multiplier and represents the slope of the tangent line supporting the quantizer operational rate-distortion performance. This design approach is usually employed for various values of λ, leading to a variety of output entropies.

Suppose that the input source is ergodic and we have a long training sequence $\mathcal{X} \equiv \{x_i\}$ avaiable. Based on this training sequence, we next provide the design algorithm for the magnitude-based ECTCQ by deriving a solution to minimizing an empirical form of the Lagrangian functional $J_\lambda(\alpha, \gamma, \beta)$ given in (A.1). What we will present is in essence similar to the design algorithm for the ECVQ [27] or the subset-based ECTCQ [28].

The codebook structure of a magnitude-based ECTCQ is completely specified by its magnitude set $\mathcal{Q} \equiv \{y_1, y_2, \cdots, y_m\}$, the corresponding entropy codebook $\mathcal{C} \equiv \{c_1, c_2, \cdots, c_m\}$, and the Lagrangian multiplier λ. We assume that an ideal entropy codebook $\mathcal{C}$ always exists and the codeword length of c_j, denoted by ℓ_j, is eqal to the self-information [5] of the event $\beta(\alpha(X)) = \pm y_j$ where X is a source random variable.

For a given triplet (α, γ, β), define $\mathcal{X}_j \equiv \{x \in \mathcal{X} : |\beta(\alpha(x))| = y_j\}$. $\mathcal{X}_j$ can be divided into $\mathcal{X}_{j,+} = \{x \in \mathcal{X} : \beta(\alpha(x)) = y_j\}$ and $\mathcal{X}_{j,-} = \{x \in \mathcal{X} : \beta(\alpha(x)) = -y_j\}$. An empirical form for the Lagrangian functional in (A.1) becomes

$$J_\lambda(\alpha, \gamma, \beta) = \frac{1}{||\mathcal{X}||} \sum_{j=1}^{m} \left[\sum_{x \in \mathcal{X}_j^+} \xi(x, y_j) + \sum_{x \in \mathcal{X}_j^-} \xi(x, -y_j) \right], \tag{A.2}$$

where

$$\xi(x, \pm y_j) = (x \mp y_j)^2 + \lambda \ell_j, \tag{A.3}$$

is called the *biased squared-error distortion measure.*

To minimize the Lagrangian functional given in (A.2) starting from an arbitrary coder $(\alpha^{(0)}, \gamma^{(0)}, \beta^{(0)})$, the basic idea is to repeatedly apply a transformation

$$(\alpha^{(t+1)}, \gamma^{(t+1)}, \beta^{(t+1)}) = \mathcal{T}(\alpha^{(t)}, \gamma^{(t)}, \beta^{(t)}) \tag{A.4}$$

such that $J^{(t)} \triangleq J_\lambda(\alpha^{(t)}, \gamma^{(t)}, \beta^{(t)})$ decreases in t. Since $J^{(t)}$ is bounded below by zero, $\{J^{(t)}\}$ is guaranteed to converge. As a convergence criterion for the algorithm, we may use the simple stopping rule

$$(J^{(t)} - J^{(t+1)})/J^{(t+1)} < \epsilon, \tag{A.5}$$

where $\epsilon > 0$ is a convergence threshold. We next describe a possible transformation $\mathcal{T}$ that operates as follows:

$$\alpha^{(t+1)} = \arg\min_{\alpha} J_\lambda(\alpha, \gamma^{(t)}, \beta^{(t)}), \tag{A.6.a}$$

$$\gamma^{(t+1)} = \arg\min_{\gamma} J_\lambda(\alpha^{(t+1)}, \gamma, \beta^{(t)}), \tag{A.6.b}$$

$$\beta^{(t+1)} = \arg\min_{\beta} J_\lambda(\alpha^{(t+1)}, \gamma^{(t+1)}, \beta). \tag{A.6.c}$$

Fixing β and γ, (A.6.a) means applying the Viterbi trellis search algorithm to determine the optimal reproduction sequence $\{\hat{x}_i^{(t+1)}\}$ for the training sequence $\{x_i\}$. One can then determine $\mathcal{X}_j^{(t+1)}$ based on $\{\hat{x}_i^{(t+1)}\}$. The (empirical) probability that a source sample is mapped (by α) to y_j or $-y_j$ can hence be estimated as $P_j^{(t+1)} \approx ||\mathcal{X}_j^{(t+1)}||/||\mathcal{X}||$. Note that γ has nothing to do with the quantization error, the second step of $\mathcal{T}$ (A.6.b) decreases the Lagrangian functional by modifying the codeword length according to

$$\ell_j = -\log_2 P_j. \tag{A.7}$$

Now fixing α and γ, the mapping β that minimizes (A.2) is one that, for each j, minimizes

$$\sum_{x \in \mathcal{X}_{j,+}^{(t+1)}} (x - y_j)^2 + \sum_{x \in \mathcal{X}_{j,-}^{(t+1)}} (x + y_j)^2. \tag{A.8}$$

0. given $\mathcal{X}$, λ, ϵ, and $(\alpha^{(0)}, \gamma^{(0)}, \beta^{(0)})$;
   set $t = 0$ and $J^{(0)} = \infty$;

1. encode $\mathcal{X}$ with $\gamma^{(t)}$ and $\beta^{(t)}$, yielding $\{\hat{x}_i^{(t+1)}\}$ and $J^{(t+1)}$;
   if $(J^{(t)} - J^{(t+1)})/J^{(t+1)} < \epsilon$, quit;

2. based on $\{\hat{x}_i^{(t+1)}\}$, determine $\{\mathcal{X}_j^{(t+1)}\}$ and $\{P_j^{(t+1)}\}$;
   update $\gamma^{(t+1)}$ according to (A.7).

3. encode $\mathcal{X}$ with $\gamma^{(t+1)}$ and $\beta^{(t)}$, yielding new $\{\mathcal{X}_j^{(t+1)}\}$;
   update $\beta^{(t+1)}$ according to (A.9);
   replace t by $t + 1$; go to step 1;

Figure A.1: Magnitude-based ECTCQ design algorithm.

The j^{th} magnitude $y_j^{(t+1)}$ should therefore be updated as the weighted centroid

$$y_j^{(t+1)} = \left(\sum_{x \in \mathcal{X}_{j,+}^{(t+1)}} x - \sum_{x \in \mathcal{X}_{j,-}^{(t+1)}} x \right) \Big/ ||\mathcal{X}_j^{(t+1)}||. \tag{A.9}$$

Figure A.1 summarizes this magnitude-based ECTCQ design algorithm.

The classical result of Gish and Pierce [23] indicates that the optimal entropy-constrained quantizers at high rates are very nearly uniform. Besides, the uniform quantizers are easier to implement. These have been the motivation for designing ECTCQ whose reproduction symbols are taken from an appropriately scaled and shifted version of the integer set [28]. Here we also consider the uniform ECTCQ in which the magnitude $y_j = (2j-1)\delta$ where δ is a scaling factor. Replacing this new constraint into (A.2) and setting $\partial J_\lambda / \partial \delta = 0$, we obtain a new updating formula for β in terms of δ as follows:

$$\delta^{(t+1)} = \left(\sum_{j=1}^{m} (2j-1) \left[\sum_{x \in \mathcal{X}_{j,+}^{(t+1)}} x - \sum_{x \in \mathcal{X}_{j,-}^{(t+1)}} x \right] \right) \Big/ \left(\sum_{j=1}^{m} (2j-1)^2 ||\mathcal{X}_j|| \right). \tag{A.10}$$

Bibliography

[1] C. E. Shannon, "A mathematical theory for communications," *Bell Syst. Tech. J.*, vol. 27, pp. 397–423, 623–656, 1948.

[2] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE Nat'l. Conv. Rec. part 4*, pp. 142–1632, 1959.

[3] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video Coding.* Englewood Cliffs, NJ: Prentice-Hall, 1984.

[4] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Boston: Kluwer Academic Publishers, 1992.

[5] R. G. Gallager, *Information Theory and Reliable Communications.* New York: Wiley, 1968.

[6] T. Berger, *Rate-Distortion Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

[7] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, pp. 1098–1101, Sept. 1952.

[8] J. Rissanen and G. Langdon, Jr., "Arithmetic coding," *IBM J. Res. Develop.*, pp. 149–162, March 1979.

[9] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520–540, June 1987.

[10] F. Jenilek, "Tree encoding of memoryless time-discrete sources with a fidelity criterion," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 584–590, Sept. 1969.

[11] A. J. Viterbi and J. K. Omura, "Trellis encoding of a memoryless time-discrete source with a fidelity criterion," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 325–332, May 1974.

[12] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 373–380, July 1979.

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

[14] V. Eyuboğlu and G. D. Forney, Jr., "Lattice and trellis quantization with lattice- and trellis-bounded codebooks — High-rate theory for memoryless sources," *IEEE Trans. Inform. Theory*, vol. IT-39, pp. 46–59, Jan. 1993.

[15] R. Laroia and N. Farvardin, "Trellis-based scalar vector quantizer for memoryless sources," *IEEE Trans. Inform. Theory*, vol. IT-40, pp. 860–870, May 1994.

[16] J. Max, "Quantization for minimum distortion," *IEEE Trans. Inform. Theory*, vol. IT-6, pp. 7–12, Mar. 1960.

[17] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 129–137, March 1982.

[18] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.

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

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

[21] M. Marcellin and T. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources," *IEEE Trans. Commun.*, vol. COM-38, pp. 82–93, Jan. 1990.

[22] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 55–67, Jan. 1982.

[23] H. Gish and J. N. Pierce, "Asymptotically efficient quantizing," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 676–683, Sept. 1968.

[24] T. Berger, "Optimum quantizers and permutation codes," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 759–764, Nov. 1972.

[25] T. Berger, "Minimum entropy quantizers and permutation codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 149–157, Mar. 1982.

[26] N. Farvardin and J. Modestino, "Optimum quantizer performance for a class of non-Gaussian memoryless sources," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 485–497, May 1984.

[27] P. Chou, T. Lookabaugh, and R. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-37, pp. 31–42, Jan. 1989.

[28] T. Fischer and M. Wang, "Entropy-constrained trellis-coded quantization," *IEEE Trans. Inform. Theory*, vol. IT-38, pp. 415–426, March 1992.

[29] D. Forney Jr., "Coset codes — Part I: Introduction and geometrical classification," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1123–1151, Sept. 1988.

[30] D. Forney Jr., "Coset codes — Part II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1152–1187, Sept. 1988.

[31] R. Laroia, S. Tretter, and N. Farvardin, "A simple and effective precoding scheme for noise whitening on intersymbol interference channels," *IEEE Trans. Commun.*, vol. 41, pp. 460–463, October 1993.

[32] M. W. Marcellin, "On entropy-constrained trellis coded quantization," *IEEE Trans. Commun.*, vol. 42, pp. 14–16, Jan. 1994.

[33] C.-C. Lee, N. Phamdo, R. Laroia, and N. Farvardin, "On predictive TB-SVQ of speech at low bit rates," submitted to *CISS'95*, (Baltomore, MD), March 1995.

[34] N. Phamdo, C.-C. Lee, and R. Laroia, "Speech coding using ISI coded quantization," to be presented at *ICASSP'95*, (Detroit, MI), Aug. 1994.

[35] N. Farvardin, X. Ran, and C.-C. Lee, "Adaptive DCT coding of images using entropy-constrained trellis coded quantization," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. V, pp. 397–401, April 1993.

[36] C.-C. Lee, N. Farvardin, and X. Ran, "Robust quantization of DCT coefficients in adaptive transform coding of images," *IEEE Workshop on Intelligent Signal Process. and Commun. Systems*, (Sendai, Japan), Nov. 1993.

[37] H. Jafarkhani, N. Farvardin, and C.-C. Lee, "Adaptive discrete wavelet transform coding of images," to be presented at *ICIP'94*, (Austin, TX), Nov. 1994.

[38] P. L. Zador, "Asymptotic quantization error of continuous signals and the quantization dimension," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 139–149, March 1982.

[39] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups.* New York: Springer-Verlag, 1988.

[40] D. Forney Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, 1973.

[41] K. Sayood, J. D. Gibson, and M. C. Rost, "An algorithm for uniform vector quantizer design," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 805–814, Nov. 1984.

[42] N. Farvardin and J. Modestino, "Rate-distortion performance of DPCM schemes for autoregressive sources," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 402–418, May 1985.

[43] R. Laroia, N. Farvardin, and S. Tretter, "On optimal shaping of multidimensional constellations," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1044–1056, July 1994.

[44] V. Eyuboğlu and G. D. Forney, Jr., "Trellis precoding: Combined coding, precoding and shaping for intersymbol interference channels," *IEEE Trans. Inform. Theory*, vol. IT-38, pp. 301–314, Jan. 1992.

[45] R. Laroia, "Coding for intersymbol interference channels — combined coding and precoding," submitted to *IEEE Trans. Inform. Theory*, July 1994.

[46] N. Farvardin and J. Modestino, "Adaptive buffer-instrumented entropy-coded quantizer performance for memoryless sources," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 9–22, Jan. 1986.

[47] J. Modestino, D. Harrison, and N. Farvardin, "Robust adaptive buffer-instrumented entropy-coded quantization of stationary sources," *IEEE Trans. Commun.*, vol. COM-38, pp. 859–867, June 1990.

[48] A. Balamesh and D. Neuhoff, "Block-constrained methods of fixed-rate entropy-coded scalar quantization," submitted to *IEEE Trans. Inform. Theory*, Sep. 1992.

[49] M. W. Marcellin, T. R. Fischer, and J. D. Gibson, "Predictive trellis coded quantization of speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, pp. 46–55, Jan. 1990.