THESIS REPORT

Master's Degree

A Study of the Reflection Symmetric Residual Quantizer and Application to Speech Coding

by P. Ligdas
Advisor: N. Farvardin

M.S. 93-24

ISR
INSTITUTE FOR SYSTEMS RESEARCH

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

Abstract

Title of Thesis: A Study of the Reflection Symmetric Residual Quantizer and Application to Speech Coding

Name of degree candidate: Paschalis Ligdas

Degree and year: Master of Science, 1993

Thesis directed by: Professor Nariman Farvardin
Department of Electrical Engineering

The performance of the reflection symmetric residual quantizer (rRQ) on various types of sources is tested and certain conclusions about its capabilities and limitations are drawn. rRQ is then used to design several low-complexity speech spectrum coding schemes. Results of these methods are presented and their performance is compared and found similar to other known speech spectrum coding methods of similar complexity.

A Study of the Reflection Symmetric Residual Quantizer and Application to Speech Coding

by

Paschalis Ligdas

Thesis submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1993

Acknowledgements

I would like to thank my advisor Professor Nariman Farvardin for the long and frequent discussions I had with him. His encouragement has been most helpful.

Table of Contents

List of Tables

Number **Page**

List of Figures

Chapter 1

Introduction

One of the most practical applications of information theory is the optimal discretization of a continuous-alphabet source, known as **quantization**. The part of information theory that deals with the theoretical limitations of this procedure is called **rate-distortion theory**. Rate-distortion theory gained a lot of interest in the late 60s and almost reached its limits in the mid-70s. Unfortunately most real-world problems are too complicated for information theorists to tackle them. Even for very simple problems, rate-distortion theory can only be used to obtain performance bounds that cannot be exceeded by any quantizer. However, just like in the channel coding problem, information theory does not provide us with specific tools for designing actual source quantizers, but rather helps to gain an insight to the problem that can be useful in developing practical quantizers.

The rate-distortion function $R(D)$ of a discrete-time stationary and ergodic source $\{X_i\}_{i=1}^{\infty}$ is defined by [1]

$$R(D) = \lim_{n \to \infty} \frac{1}{n} R_n(D) \tag{1.1}$$

where $R_n(D)$ is given by

$$R_n(D) = \inf_{p(\hat{\mathbf{X}}^n|\mathbf{X}^n): E\{d_n(\mathbf{X}^n, \hat{\mathbf{X}}^n)\} \leq nD} I(\mathbf{X}^n; \hat{\mathbf{X}}^n). \tag{1.2}$$

The function $d_n : \mathbb{R}^n \times \mathbb{R}^n \mapsto [0, \infty)$, is the **distortion measure** which quantifies the "badness" of representing $\mathbf{x}$ by $\hat{\mathbf{x}}$. According to Shannon's theory, $R(D)$ is the smallest rate at which a source can be encoded so that the average distortion does not exceed D. An optimal vector quantizer achieves the rate-distortion bound as the dimensionality approaches infinity [2]. Therefore, in theory, we can obtain vector quantizers that operate arbitrarily close to the rate-distortion bound [2, 3]. In practice, however, this is not possible because of severe complexity limitations. Since the convergence of a vector quantizer's (VQ) performance to the optimal achievable performance as a function of block size is slow [4, 5], there is a great need for low-complexity quantization schemes that permit reduction in the encoding complexity without any (or small) loss of performance. This can be achieved by imposing a structure on the VQ and optimize the VQ subject to the constraints of the chosen structure. The choice of the structure is based on ad hoc techniques and the resulting optimization problem is not always straightforward. While the computational and memory complexity of an unconstrained VQ grows exponentially with the dimension, a structured VQ usually has a complexity that grows only polynomially with the dimension. Clearly, at a given encoding rate, a constrained VQ will yield a higher average distortion than an optimal unconstrained VQ of the same dimensionality. However, when we look at the distortion achieved by different quantizers of the same complexity, instead of the same dimensionality, then a structured VQ may in fact, but not necessarily, yield a smaller distortion. Of course, the notion of complexity is not as well defined as we would like. In general the number of additions and multiplications required to encode a source sample is a meaningful, and in most cases useful, measure of the computational complexity, and the amount of memory needed for encoding is a measure of the memory complexity. For example, the memory and the computational complexity of an unstructured (exhaustively searched) VQ are both proportional to the number of codevectors in the codebook. For an n-dimensional VQ of rate R, this equals 2^{Rn}.

A well-known structured VQ is the **tree-structured VQ** (TSVQ) [6] in which the codebook is organized in a tree structure that can be searched suboptimally but very efficiently by a low-cost encoder which performs a tree search of the codebook instead of a costly exhaustive search. TSVQ suffers small performance degradation and reduces significantly the computational complexity. These benefits however, come at the cost of an increase in the memory complexity, which limits its use to small values of the product Rn, where R denotes the encoding rate in bits per source sample. and n denotes the dimension of the TSVQ. Another structured VQ is the lattice VQ [7] which does not require training, and theoretically has a good performance with memoryless sources that are almost uniform, or any memoryless source when the dimension is sufficiently large. Another class of structured VQs are the **residual quantizers** (RQs) [8]. An RQ typically consists of a cascade of exhaustively searched VQs (ESVQs). At each stage, an ESVQ of relatively small size encodes the residual error of the previous stage; the residual vector of the first stage is simply the input source vector.

1.1 Motivation and Contribution

In this thesis we review a particular form of RQ known as reflection symmetric RQ (rRQ). rRQs were introduced by Barnes in [9] as a very low complexity type of RQ. We have investigated the possibility of incorporating rRQ into a system that encodes the **line spectrum pair** (LSP) vectors of speech. The LSP parameters of speech form an efficient way of speech modeling that yields good results in compression of speech. In order to obtain quantized speech of transparent quality by encoding its LSP parameters, we need to quantize the 10-dimensional LSP vectors at an encoding rate that exceeds 2.3 bits per sample. For these values of dimension and rate it is not possible to build a full search VQ because of the enormous amount of memory required to store the 2^{23} codevectors of the codebook and the huge computational power required to search exhaustively a codebook of that size. Therefore, other forms of reduced complexity VQ have been used in quantizing LSP vectors of speech. One of the most promising techniques is split-VQ in which the 10-dimensional LSP vector is split into two or three sub-vectors of smaller size, each of which is encoded by an appropriately designed vector quantizer. The motivation for using rRQ in the context of LSP coding stems from the fact that rRQ has so low complexity that we can easily design and realize such quantizers for the above values of dimension and rate. Our first system is a finite-state quantizer based on rRQ (FS-rRQ). FS-rRQ is one of the methods that can be used to utilize the inter-vector memory between successive LSP vectors without introducing additional delay to the system. The second technique we have investigated, employs rRQ as the building block in a predictive coding system (P-rRQ). Like FS-rRQ, this method utilizes the inter-vector memory without any additional delay. The drawback of this scheme is that it is not robust to channel noise since it may result in error propagation. The last method we used is a very simple and straightforward application of rRQ to LSP coding, that utilizes the inter-frame correlation very efficiently. In this technique, called matrix rRQ, two or three LSP vectors are concatenated and quantized using an rRQ of the resulting dimensionality. This permits utilization of the inter-vector memory of the source, but has the drawback that it introduces an additional delay which is an undesirable property in two-way communication systems.

The next chapter discusses residual quantizers and describes an algorithm for joint optimization of its stages. Chapter 3 introduces rRQ and describes a design algorithm. Results of rRQ on various types of sources are given. Chapter 4 describes the use of rRQ in two methods of LSP coding. Finally, Chapter 5 includes a summary and conclusions.

Chapter 2

Residual Vector Quantizers

This chapter is organized as follows. Section 2.1 reviews the basics of quantization theory. The structure of RQs is presented in Section 2.2 followed by Section 2.3 which describes the optimization procedure of RQs.

2.1 Optimal Vector Quantization

We consider the problem of quantizing a random vector $\mathbf{X}$ of dimension n with probability distribution $F_X(\cdot)$ using an n-dimensional VQ.

An n-dimensional VQ is defined by the following three entities: 1) an indexed subset $A = \{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{N-1}\}$ of $\mathbb{R}^n$ called the **codebook**, 2) a **partition** $\mathcal{P} = \{S_0, S_1, \ldots, S_{N-1}\}$ of $\mathbb{R}^n$ and 3) a **quantizer mapping** $Q : \mathbb{R}^n \mapsto A$ that maps each vector $\mathbf{x}$ in $\mathbb{R}^n$ to a unique codevector $\mathbf{y}_k$ if $\mathbf{x} \in S_k$. The triple $(A, Q, \mathcal{P})$ uniquely specifies a VQ. The necessary conditions for optimality are [10] (i) the **centroid condition**

$$\mathbf{y}_j = E_{\mathbf{X}|\mathbf{X} \in S_j}\{\mathbf{X}|\mathbf{X} \in S_j\}, \qquad (2.1)$$

and (ii) the **nearest neighbor condition**

$$\mathbf{x} \in S_j \iff d(\mathbf{x}, \mathbf{y}_j) \leq d(\mathbf{x}, \mathbf{y}_k) \quad \forall k. \tag{2.2}$$

In general, the above conditions are not sufficient for global optimality, but there is a widely held conjecture that they are sufficient for **local optimality**. However, there is no general theoretical derivation of this result [10].

2.1.1 An Algorithm for Vector Quantizer Design

The most popular design algorithm for VQs, involves the use of a training set of data generated by the probability distribution that describes the source random process. This algorithm, known as the Linde-Buzo-Gray (LBG) [11] or the Generalized Lloyd algorithm, applies iteratively the two necessary conditions of optimality (2.1) and (2.2) on the training set. This is done in two steps: First, the training data are partitioned by associating each training vector with its nearest codevector using equation (2.2); then the centroid of each cell is computed using equation (2.2).

Gray, Kieffer and Linde [12] have derived an ergodic theorem which states that if the source is a stationary ergodic discrete time process, then in the limit as $L \to \infty$, the LBG algorithm applied on a sample distribution of a training sequence of length L will produce the same codebook as if it were run on the actual underlying distribution with the same initial codebook. This is true for a broad class of distortion measures that satisfy the following rules:

1. For any fixed $\mathbf{x} \in \mathbb{R}^n$, $d(\mathbf{x}, \mathbf{y})$ is a convex function in $\mathbf{y}$.

2. For any fixed $\mathbf{x}$, if $\mathbf{y}(n) \to \infty$ as $n \to \infty$, then $d(\mathbf{x}, \mathbf{y}(n)) \to \infty$.

3. For any bounded sets $B_1, B_2 \in \mathbb{R}^n$, $\sup_{\mathbf{x} \in B_1, \mathbf{y} \in B_2} d(\mathbf{x}, \mathbf{y}) < \infty$.

Although LBG is an algorithm that always converges to a solution, there is no guarantee that the solution will be globally optimal. In [12] it is proven that under mild restrictions, a vector quantizer designed by applying the LBG algorithm on a training sequence is locally optimal. Some algorithms like simulated annealing [13] attempt to find global optima instead of just local optima. However, the additional gain is usually moderate to small, and in many cases does not justify the additional delay in the design procedure [10].

2.2 Residual Quantizers

Unstructured VQ is conceptually the simplest kind of VQ and for a given block size yields the smallest (subject to local optimality) average distortion among all types of memoryless VQs. However, both its memory and computational complexity are proportional to the number of codevectors 2^{Rk}. Thus, its complexity grows exponentially with the product Rk, where R denotes the rate and k denotes the dimension of the VQ. When the product Rk is greater than 10, the realization of such VQs becomes extremely complex. In order to be able to design and implement VQs with large dimensions and/or rates, some sort of structure should be imposed on the codebook and the encoder. A popular kind of structured VQ is the residual quantizer (RQ).

We now proceed to the definition of RQs and presentation of necessary conditions for optimality. The solution to this problem is aided by the introduction of the **equivalent single-stage quantizer** which is defined as a single-stage VQ that for any input sequence produces the same output sequence as if the

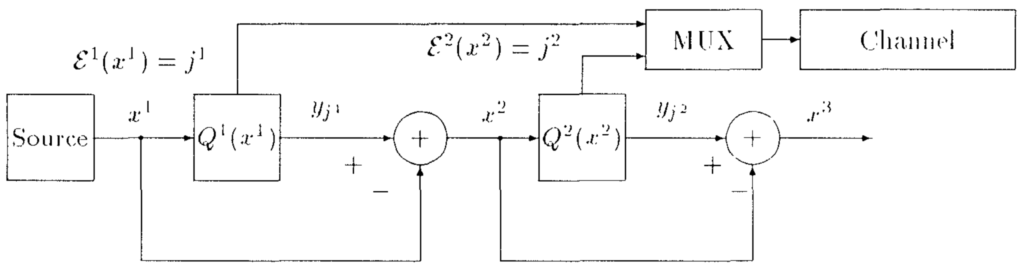

Figure 2.1: Two-stage residual quantizer encoder

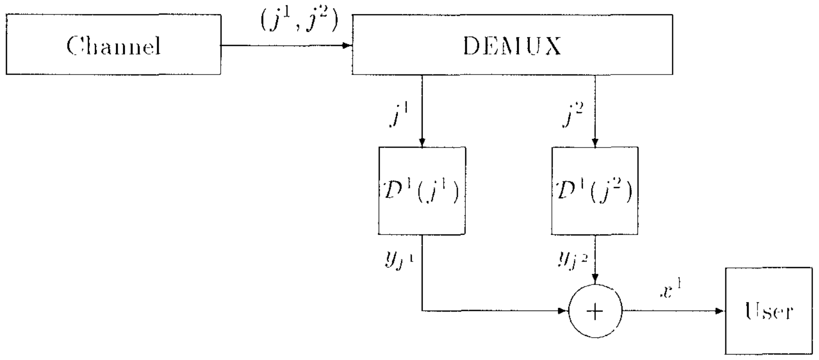

Figure 2.2: Two-stage residual quantizer decoder

input was quantized by the RQ [14].

2.2.1 Definition of RQ

The encoder and decoder of a two-stage RQ are depicted in Figures 2.1 and 2.2, respectively.

Let $\mathbf{X}^1$ represent a source vector with probability distribution $F_{X^1}(\cdot)$. A residual quantizer with P stages is defined as a sequence of P VQs $\{(\mathcal{A}^p, \mathcal{P}^p, \mathcal{Q}^p);\ 1 \leq p \leq P\}$, combined in such a way that $(\mathcal{A}^1, \mathcal{P}^1, \mathcal{Q}^1)$ quantizes the source output vector $\mathbf{x}^1$ and $(\mathcal{A}^p + 1, \mathcal{P}^p + 1, \mathcal{Q}^p + 1)$ quantizes the residual error vector $\mathbf{x}^{p+1} = \mathbf{x}^p - \mathcal{Q}^p(\mathbf{x}^p)$ of the previous stage $(\mathcal{A}^p, \mathcal{P}^p, \mathcal{Q}^p)$ for $1 \leq p < P$.

We denote by N^p the size of the stagewise codebook $A^p = \{\mathbf{y}_0^p, \mathbf{y}_1^p, \ldots, \mathbf{y}_{N^p-1}^p\}$. In the following, the superscript of a given quantity specifies its stage number, and the subscript specifies its index. The map $\mathcal{Q}^p(\cdot)$ is the composition of an encoder map $\mathcal{E}^p(\cdot) : \mathbb{R}^n \mapsto J^p$ and a decoder map $\mathcal{D}^p(\cdot) : J^p \mapsto A^p$, where $J^p = \{0, 1, \ldots, N^p - 1\}$ is the index set of stage p. The indices produced by the encoder maps are concatenated to form a P-tuple $\mathbf{j}^P = (j^1, j^2, \ldots, j^P)$. These P-tuples are called **product codewords**. The decoder $\mathcal{D}^p$ generates the code-vector $\mathbf{y}_{j^p}^p$ corresponding to the index j^p and the final quantized version $\hat{\mathbf{x}}^1$ of the input vector $\mathbf{x}^1$ is the direct sum of the stagewise codevectors, i.e.

$$\hat{\mathbf{x}}^1 = \sum_{p=1}^{P} \mathbf{y}_{j^p}^p \tag{2.3}$$

2.2.2 Definition of Equivalent Quantizers

An optimal RQ is one that minimizes the average distortion

$$E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} = \int d[\mathbf{x}^1, \sum_{p=1}^{P} \mathcal{Q}^p(\mathbf{x}^p)] dF_{X^1 \ldots X^P}. \tag{2.4}$$

The minimization of the above is simplified by introducing the equivalent quantizers [9]. Basically an equivalent quantizer is just a single-stage quantizer that produces the same representation for the source. Its advantage is that computation of the distortion in this case does not require knowledge of the unknown joint distribution $F_{X^1 \ldots X^P}(\cdot)$. We denote the equivalent quantizer by the triple $(A^e, \mathcal{P}^e, \mathcal{Q}^e)$. The **equivalent codebook** A^e is the direct sum codebook formed by the stagewise codebooks A^p, i.e. $A^e = A^1 + \cdots + A^P$. The **equivalent codevectors** $\mathbf{y}^e \in A^e$ are indexed by the P-tuple $\mathbf{j}^P = (j^1, j^2, \cdots, j^P)$ and each of them represents a path through a tree that can be associated with the RQ. The $\mathbf{j}^P$th **equivalent cell** $S^e(\mathbf{j}^P) \subset \mathbb{R}^n$ contains all the vectors in $\mathbb{R}^n$ that are mapped into the $\mathbf{j}^p$th equivalent codevector $\mathbf{y}^e(\mathbf{j}^p)$. The **equivalent partition** $\mathcal{P}^e$ is the collection of all equivalent cells. The **equivalent mapping** $\mathcal{Q}^e : \mathbb{R}^n \mapsto A^e$ is defined by $\mathcal{Q}^e(\mathbf{x}^1) = \mathbf{y}^e(\mathbf{j}^p)$ if $\mathbf{x}^1 \in S^e(\mathbf{j}^p)$.

The average distortion of the equivalent quantizer, which is the same as that of the original RQ, takes the much simpler form

$$E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} = \int d[\mathbf{x}^1, \mathcal{Q}^e(\mathbf{x}^1)] dF_{X^1}. \tag{2.5}$$

The above quantity is minimized by the optimal RQ.

2.3 Optimization of Residual Quantizers

First we consider the scalar case under the squared-error distortion measure. Our objective is to choose all stagewise codebooks A^p and partitions $\mathcal{P}^p$ so that the resulting RQ minimizes the average distortion. In the following we assume that the size of each stagewise codebook is predetermined and fixed. The choice of the size of each stagewise codebook is usually made in an ad-hoc fashion and will not be discussed in this thesis.

2.3.1 Optimal Scalar Residual Quantizers

<u>Optimal Stagewise RQ Levels</u>

First we calculate the optimal stagewise codebooks A^p. The expression (2.5) for the average distortion takes the form

$$E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} = \int_{-\infty}^{\infty} [x^1 - y^e(\mathbf{j}^p)]^2 f_{X^1}(x^1) dx^1. \tag{2.6}$$

Assuming a fixed partition $\mathcal{P}^e$ and fixed stagewise codebooks except for codebook A^p, $E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\}$ is minimized by setting its partial derivative with respect to

$y^p_{k^p}$ equal to zero. After some algebra this yields the result [14]

$$y^p_{k^p} = \frac{\sum_{\mathbf{j}^P \in H^p_{k^p}} \int_{S^e(\mathbf{j}^P)} (x^1 - \sum_{\rho \neq p} y^\rho_{j^\rho}) f_{X^1}(x^1) dx^1}{\sum_{\mathbf{j}^P \in H^p_{k^p}} \int_{S^e(\mathbf{j}^P)} f_{X^1}(x^1) dx^1}. \tag{2.7}$$

where $H^p_{k^p}$ is the set of all $\mathbf{j}^P$ such that the pth element of $\mathbf{j}^P = (j^1, \ldots, j^p, \ldots, j^P)$ is equal to k^p, i.e., $H^p_{k^p} = \{\mathbf{j}^P : j^p = k^p\}$. If we define the pth **grafted branch** as

$$g^p(\mathbf{j}^P) = y^e(\mathbf{j}^P) - y^p_{j^p} \tag{2.8}$$

and the pth **graft residual** as

$$\xi^p = x^1 - g^p(\mathbf{j}^P), \tag{2.9}$$

then (2.7) becomes

$$y^p_{k^p} = \frac{\int \xi^p \sum_{\mathbf{j}^P \in H^p_{k^p}} I_{G^p(\mathbf{j}^P)} f_{X^1}[g^p(\mathbf{j}^P) + \xi^p] d\xi^p}{\int \sum_{\mathbf{j}^P \in H^p_{k^p}} I_{G^p(\mathbf{j}^P)} f_{X^1}[g^p(\mathbf{j}^P) + \xi^p] d\xi^p}. \tag{2.10}$$

The sum in the denominator can be viewed as the conditional pdf of Ξ^p given $y^p_{k^p}$. After some manipulation we can write this conditional pdf as

$$f_{\Xi^p | x^p \in S^p_{k^p}}(\xi^p | x^p \in S^p_{k^p}) = \frac{\sum_{\mathbf{j}^P \in H^p_{k^p}} I_{G^p(\mathbf{j}^P)} f_{X^1}(x^1)[g^p(\mathbf{j}^P) + \xi^p]}{\text{Prob}(x^1 \in \mathcal{H}^p_{k^p})}. \tag{2.11}$$

where we have defined $\mathcal{H}^p_{k^p} = \cup_{\mathbf{j}^P \in H^p_{k^p}} S^e(\mathbf{j}^P)$. The expression for the optimal quantization levels becomes

$$y^p_{k^p} = \int \xi^p f_{\Xi^p | x^p \in S^p_{k^p}}(\xi^p | x^p \in S^p_{k^p}) d\xi^p = E\{\Xi^p | x^p \in S^p_{k^p}\} \tag{2.12}$$

for $1 \leq p \leq P$ and $0 \leq k^p < N^p$.

Equation (2.12) describes the stagewise quantizer levels as a conditional expectation and adds some useful insight to the problem of joint optimization. In general, the equivalent codevectors of an RQ are constrained by the underlying tree structure. Changing a single stagewise codevector affects all the equivalent codevectors which contain that modified stagewise codevector. This constraint makes it in general impossible to satisfy the centroid condition which is a necessary condition for optimality. Instead, what we can do is choose the stagewise codevectors $y^{p}_{k^p}$ such that the expected (conditional) distortion given that $x^1 \in \mathcal{H}^{p}_{k^p}$ is minimized.

At this point we can make an important comparison between the traditional (sequential) design method for RQs and the above condition of optimality. In the sequential design of RQs, the stagewise quanta $y^{p}_{k^p}$ are chosen to be the centroids of the residuals that result from the previous stages only. The optimal values of $y^{p}_{k^p}$, however, described by equation (2.12) suggest that they are the centroids of the residuals that result from **all** other stages, not only the previous ones. For this reason this technique is called **joint optimization** of the stages of an RQ.

Optimal Stagewise Partitions

This optimization problem is simplified by first solving the corresponding problem for the equivalent quantizer. Clearly, the optimal equivalent partition $\mathcal{P}^e$ is the one that maps each input sample x^1 into that equivalent codevector $y^e(\mathbf{j}^P)$ in the equivalent codebook A^e that minimizes the distortion $d(x^1, \cdot)$. Thus,

$$x^1 \in S^e(\mathbf{j}^P) \qquad \text{if} \qquad d(x^1, y^e(\mathbf{j}^P)) \leq d(x^1, y^e(\mathbf{k}^P)) \quad \forall \quad \mathbf{k}^P. \tag{2.13}$$

The above equation describes the well-known **nearest-neighbor** condition for optimality. Next, we specify a sequence of stagewise partitions $\{\mathcal{P}^1, \mathcal{P}^2, \ldots, \mathcal{P}^P\}$ that yield the above equivalent partition. Equation (2.13) can be written as

$$x^1 \in S^e(\mathbf{j}^P) \qquad \text{if} \qquad d(x^1, y^1_{j^1} + y^2_{j^2} + \cdots + y^P_{j^P}) \leq d(x^1, A^1 + A^2 + \cdots + A^P). \tag{2.14}$$

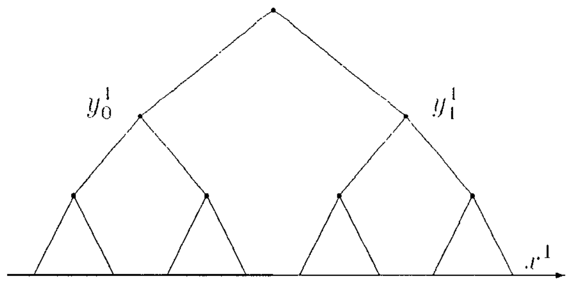

Figure 2.3: The first stage partition $\mathcal{P}^1$ for an unentangled tree (thick line for S_1^1 and thin line for S_1^1).

From this it is clear that the optimal first stage partition cells satisfy

$$x^1 \in S_{j^1}^1 \qquad \text{if} \qquad d(x^1, y_{j^1}^1 + A^2 + \cdots + A^P) \leq d(x^1, A^1 + A^2 + \cdots + A^P). \quad (2.15)$$

If the distortion measure is translation invariant, i.e. $d(x, y) = d(x-z, y-z)$, then it can be shown recursively that the optimal stagewise partitions $\mathcal{P}^p$, $1 \leq p \leq P$ are defined by

$$x^p \in S_{j^p}^p \qquad \text{if} \qquad d(x^p, y_{j^p}^p + A^{p+1} + \cdots + A^P) \leq d(x^p, A^p + A^{p+1} + \cdots + A^P), \tag{2.16}$$

where $x^p = x^1 - \sum_{i=1}^{p-1} Q^i(x^i)$.

To illustrate the various forms that stagewise partitions may take, depending on the specific structure of the tree, we give two examples in Figures 2.3 and 2.4.

It is useful to determine whether the tree structure of RQ results in a reduction in the encoding complexity when **optimal** stagewise partitions are used. In Figure 2.3 the stagewise partition cells are connected intervals and therefore a single floating point comparison is required to implement it. Since this is true

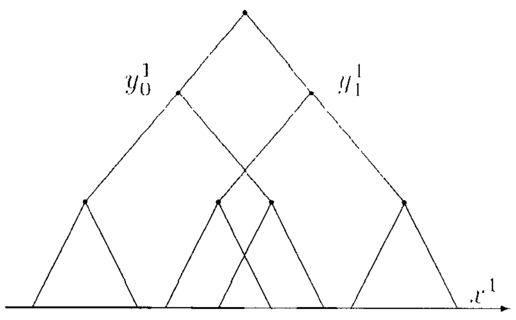

Figure 2.4: The first stage partition $\mathcal{P}^1$ for an entangled tree (thick line for S_0^1 and thin line for S_1^1).

for subsequent stages as well it is readily seen that the encoding complexity is reduced. If N^c is the total number of codevectors then the complexity is reduced from N^c to $\log N^c$ by using a binary **unentangled** tree that implements the given equivalent quantizer. The situation is quite different in the case depicted in Figure 2.4. Here, we see that the first stage partition cells are not connected intervals any more. Thus, more than one comparison may be needed in order to perform the encoding step of the first stage. In fact, for a completely entangled tree, the encoding complexity may be the same as that of the corresponding one stage quantizer. The above discussion shows that RQs with optimal partitions may or may not reduce the encoding complexity, depending on whether the corresponding tree is entangled or not [9]. One may choose, however, to search a given RQ codebook sequentially even when this does not realize the optimal partition. In this case the encoding complexity is reduced at the cost of an increase in the distortion. Another suboptimal encoding algorithm is the M-algorithm which at each stage keeps track of the M best candidate paths through the tree [15]. In many cases M has to be considerably large in order

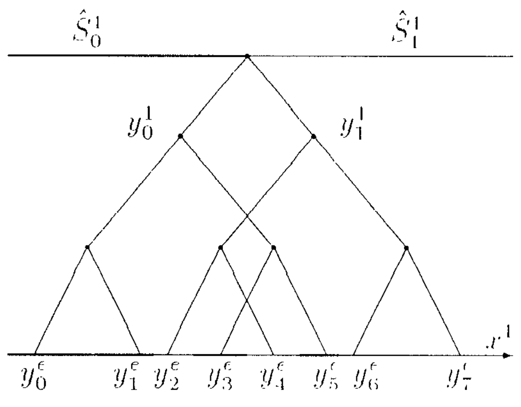
$\hat{S}_0^1$
$\hat{S}_1^1$
y_0^1
y_1^1
x^1
y_0^e y_1^e y_2^e y_3^e y_4^e y_5^e y_6^e y_7^e

Figure 2.5: Optimal first stage partition (bottomline) and the partition resulting from a tree-structured encoder for an entangled tree.

to obtain a performance close to that of the exhaustively searched system, and therefore, the computational savings are not very significant [9].

In general, if an RQ is to be used in a system of low encoding complexity, the encoder should have a tree structure. From Figure 2.3 we see that the optimal stagewise partitions can be realized by such an encoder if the tree that corresponds to the RQ is not entangled. However, if the tree is entangled, then an encoder with a tree structure is suboptimal. This is illustrated in Figure 2.5 where we see the optimal first stage partition and the first stage partition that results from a tree-structured encoder. Clearly, the low-complexity tree-structured encoder is suboptimal. One way to see this is the following: The equivalent codevectors y_2^e and y_5^e will never be chosen by this encoder. This, of course, is a very inefficient use of the codebook and is due to the entanglement of that codebook.

2.3.2 Optimum Vector Residual Quantizers

After discussing the necessary conditions for optimality of scalar RQs, we now turn to the more general problem of vector RQs. One of the differences between these two cases is that for vector RQs, the encoding algorithm can be inefficient even when the (vector) tree is not entangled.

<u>Optimum Stagewise Codevectors</u>

We need to specify the stagewise codebooks $A^1, A^2, \ldots, A^P$ that for a given set of stagewise partitions $\mathcal{P}^1, \mathcal{P}^2, \ldots, \mathcal{P}^P$, minimize the average distortion

$$\begin{aligned} E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} &= E\{d(\mathbf{X}^1, \mathcal{Q}(\mathbf{X}^1))\} & (2.17)\\ &= \sum_{\mathbf{j}^P} E\{d(\mathbf{X}^1, \mathbf{y}^e(\mathbf{j}^P))|\mathbf{X}^1 \in S^e(\mathbf{j}^P)\}\Pr(\mathbf{X}^1 \in S^e(\mathbf{j}^P)) & (2.18)\end{aligned}$$

Again, under the assumption that the distortion measure is translation invariant, we can write the above as

$$\begin{aligned} E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} &= \sum_{\mathbf{j}^P} E\{d(\mathbf{X}^1 - \mathbf{g}^p(\mathbf{j}^P), \mathbf{y}^p_{j^p})|\mathbf{X}^1 \in S^e(\mathbf{j}^P)\}\Pr(\mathbf{X}^1 \in S^e(\mathbf{j}^P)) \\ &= \sum_{k^p} \sum_{\mathbf{j}^P \in H^p_{k^p}} E\{d(\mathbf{X}^1 - \mathbf{g}^p(\mathbf{j}^P), \mathbf{y}^p_{j^p})|\mathbf{X}^1 \in S^e(\mathbf{j}^P)\} \cdot \\ &\qquad \cdot \Pr(\mathbf{X}^1 \in S^e(\mathbf{j}^P)). & (2.19)\end{aligned}$$

Introducing again the pth graft residual $\xi^p = \mathbf{x}^1 - \mathbf{g}^p(\mathbf{j}^P)$, we can write

$$E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} = \sum_{k^p} E\{d(\Xi^p, \mathbf{y}^p_{k^p})|\mathbf{X}^1 \in \mathcal{H}^p_{k^p}\}\Pr(\mathbf{X}^1 \in \mathcal{H}^p_{k^p}). \qquad (2.20)$$

Using the definition of $\mathcal{H}^p_{k^p}$ the above is written

$$\begin{aligned} E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} &= \sum_{k^p} E\{d(\Xi^p, \mathbf{y}^p_{k^p})|\mathbf{X}^p \in S^p_{k^p}\}\Pr(\mathbf{X}^p \in S^p_{k^p}) & (2.21)\\ &\geq \sum_{k^p} \inf_{\mathbf{u} \in \mathbf{R}^n} E\{d(\Xi^p, \mathbf{u})|\mathbf{X}^p \in S^p_{k^p}\}\Pr(\mathbf{X}^p \in S^p_{k^p}). & (2.22)\end{aligned}$$

In [14] it is shown that the infimum in (2.22) is actually achievable if for all $\mathbf{j}^P \in H_{k^\rho}^\rho$ the sets $S^e(\mathbf{j}^P)$ have non-zero measure.

Finally, the necessary condition for optimality, becomes

$$E\{d(\Xi^\rho, \mathbf{y}_{k^\rho}^\rho)|\mathbf{X}^\rho \in S_{k^\rho}^\rho\} = \min_{\mathbf{u}\in \mathbb{R}^n} E\{d(\Xi^\rho, \mathbf{u})|\mathbf{X}^\rho \in S_{k^\rho}^\rho\}. \tag{2.23}$$

The expectation in the above equation uses the conditional pdf of the graft residual $f_{\Xi^\rho|\mathbf{X}^\rho \in S_{k^\rho}^\rho}(\cdot)$ given by

$$f_{\Xi^\rho|\mathbf{X}^\rho \in S_{k^\rho}^\rho}(\xi^\rho|\mathbf{X}^\rho \in S_{k^\rho}^\rho) = \frac{\sum_{\mathbf{j}^P \in H_{k^\rho}^\rho} I_{G^\rho()\mathbf{j}^P} f_{\mathbf{X}^1}(\mathbf{g}^\rho(\mathbf{j}^P) + \xi^\rho)}{\Pr(\mathbf{X}^1 \in \mathcal{H}_{k^\rho}^\rho)}. \tag{2.24}$$

<u>Optimum Stagewise Partitions</u>

The optimum partitions for this case follow from the same analysis as in the scalar case. Therefore the optimal stagewise partitions are given by

$$\mathbf{x}^p \in S_{j^p}^p \iff d(\mathbf{x}^p, \mathbf{y}_{j^p}^p + A^{p+1} + \cdots + A^P) \leq d(\mathbf{x}^p, A^p + A^{p+1} + \cdots + A^P). \tag{2.25}$$

Equations (2.23) and (2.25) describe the necessary conditions for optimality of the stagewise codevectors and the stagewise partitions in a residual vector quantizer. It turns out that in most cases the optimal partition cannot be realized by a low-complexity tree-searched encoder. Thus, although we have succeeded in improving the preformance of an RQ, this was done at the cost of an increase of the encoder complexity. In the following chapter we introduce an additional constraint to the already constrained RQ codebook in order to be able to design jointly optimized RQs while keeping the encoder complexity low.

Chapter 3

The Reflection Symmetric Residual Quantizer

RQs are introduced for reducing the complexity of codebook search. However, the reduction in complexity requires that the codebook be searched using a tree-search algorithm. But a joint optimization of all the stages of an RQ results in non-convex stagewise partition cells [9]. Thus a computationally efficient tree-search algorithm will result in poor performance. Several techniques have been proposed in order to overcome this difficulty. The M-algorithm [15] that keeps track of the M best paths at each stage is one of them. The algorithm is a direct tradeoff between computational complexity and average distortion, but in many cases the tradeoff is not very efficient. Other methods impose additional structure in the codebook so that it can be efficiently searched by a low-complexity encoder [16].

In this chapter a structured residual quantizer introduced in [9] called reflection symmetric RQ (rRQ) is described. An algorithm for the optimal design of rRQ based on the joint optimization algorithm for RQ presented in the previous chapter is also described. The conditions for optimality however, cannot always be satisfied for all stages and certain ad hoc techniques may be necessary during the design process. Finally simulation results are given for Gaussian and Laplacian sources.

3.1 Definition of rRQ

Since there is no concise mathematical definition of rRQ, we will resort to an illustrative example. For the sake of simplicity, we assume that the tree that is associated with the rRQ is a binary tree, i.e., each stage p contains only two codevectors denoted as $\mathbf{y}_0^p$ and $\mathbf{y}_1^p$. rRQ has the following constraint imposed on its codebook. If we call L^p the hyperplane that contains all points in $\mathbb{R}^n$ that are equi-distant from $\mathbf{y}_0^p$ and $\mathbf{y}_1^p$ then we require that each equivalent codevector on the one side of L^p has a mirror image on the other side of L^p. If the final codebook satisfies the reflection symmetry condition at all stages, and the optimal stagewise partition cells are convex, then they are described by the hyperplanes L^p. The reflection symmetry property is depicted in Figure 3.1. The equations that describe this condition are as follows [9]. In the following, superscript p indicates that the pth stage is under consideration. The point midway between codevectors $\mathbf{y}_0^p$ and $\mathbf{y}_1^p$ is given by

$$\mathbf{m}^p = \frac{\mathbf{y}_0^p + \mathbf{y}_1^p}{2}. \tag{3.1}$$

This point lies on the hyperplane L^p which is the nearest neighbor (in a Euclidean sense) boundary for the two stagewise codevectors $\mathbf{y}_0^p$ and $\mathbf{y}_1^p$. The vector $\hat{\mathbf{n}}^p$ that is normal to this hyperplane and has unitary length is given by

$$\hat{\mathbf{n}}^p = \frac{\mathbf{n}^p}{||\mathbf{n}^p||}, \tag{3.2}$$

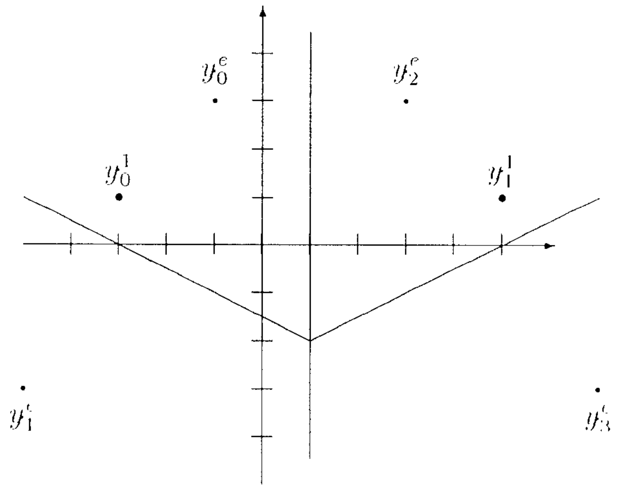

Figure 3.4: The reflection symmetry in a two-stage rRQ codebook

where $\mathbf{n}^p = \mathbf{y}_1^p - \mathbf{y}_0^p$ and $||\cdot||$ denotes the Euclidean norm. With these definitions, the equation for the hyperplane L^p becomes

$$\hat{\mathbf{n}}^p \cdot (\mathbf{u}^p - \mathbf{m}^p), \tag{3.3}$$

where "$\cdot$" denotes the operation of inner product and $\mathbf{u}^p$ is any point on the hyperplane. Then the distance δ of any point $\mathbf{x}^p \in \mathbb{R}^n$ from this hyperplane is given by

$$\delta = ||\hat{\mathbf{n}}^p \cdot (\mathbf{x}^p - \mathbf{m}^p)||. \tag{3.4}$$

We now define the reflection operator $\mathcal{R}^p_{j^p}(\cdot)$ for stage p by

$$\mathcal{R}^p_{j^p}(\mathbf{x}^p) = \begin{cases} \mathbf{x}^p & \text{if } \mathbf{x}^p \in S_0^p, \\ \mathbf{x}^p - 2\delta\hat{\mathbf{n}}^p & \text{if } \mathbf{x}^p \in S_1^p. \end{cases} \tag{3.5}$$

We denote the reflected vector $\mathcal{R}^p_{j^p}(\mathbf{x}^p)$ by $\check{\mathbf{x}}^p$. Here we use the convention that the reflection operator $\mathcal{R}^p_{j^p}(\cdot)$ maps S_1^p into S_0^p and coincides with the identity mapping on S_0^p. We denote by $\mathcal{R}_{j^p}^{-p}(\cdot)$ the inverse reflection operator, so that $\mathcal{R}_{j^p}^{-p}(\mathcal{R}_{j^p}^p(\mathbf{x}^p)) = \mathbf{x}^p$. The probability density function (pdf) of the reflected vector $\check{\mathbf{X}}^p$ is given by $f_{\check{\mathbf{X}}^p}(\check{\mathbf{x}}^p) = f_{\mathbf{X}^p}(\mathbf{x}^p) + f_{\mathbf{X}^p}(\mathcal{R}_1^{-p}(\check{\mathbf{x}}^p))$.

Now, we define the $(p+1)$st **reflected residual** $\mathbf{x}^{p+1} = \check{\mathbf{x}}^p - \hat{\mathbf{y}}^p$, and the cell $S^{p+1} = \check{S}^p - \hat{\mathbf{y}}^p$. The pdf of the reflected residual is $f_{\mathbf{X}^{p+1}}(\mathbf{x}^{p+1}) = f_{\check{\mathbf{X}}^p}(\mathbf{x}^{p+1} + \hat{\mathbf{y}}^p)$. In the next stage, S^{p+1} is subdivided into S_0^{p+1} and S_1^{p+1}.

A recursive definition of the reflected residual $\mathbf{x}^{p+1}$ is

$$\mathbf{x}^{p+1} = \mathcal{Q}_{j^p}^p(\mathbf{x}^p) \overset{\text{def}}{=} \mathcal{R}_{j^p}^p(\mathbf{x}^p) - \hat{\mathbf{y}}^p. \tag{3.6}$$

An expression for the final reflected residual x^{P+1} is

$$\begin{aligned} \mathbf{x}^{P+1} &= \mathcal{Q}_{j^P}^P(\ldots \mathcal{Q}_{j^2}^2(\mathcal{Q}_{j^1}^1(\mathbf{x}^1))), & (3.7)\\ &= \bigsqcup_{p+1}^{P} \mathcal{Q}_{j^p}^p(\mathbf{x}^p). & (3.8) \end{aligned}$$

The reconstructed vector of the encoded input vector is derived by performing the inverse operations, i.e., reflections and translations. We can write an expression for the **partially reconstructed** codevector $\hat{\mathbf{y}}^p(j^p, j^{p+1}, \ldots, j^P)$ as

$$\hat{\mathbf{y}}^p(j^p, j^{p+1}, \ldots, j^P) = \mathcal{Q}_{j^p}^{-p}(\hat{\mathbf{y}}^{p+1}). \tag{3.9}$$

where the $\mathcal{Q}_{j^p}^{-p}(\cdot)$ is the inverse reflected residual operator defined by

$$\mathcal{Q}_{j^p}^{-p}(\hat{\mathbf{y}}^{p+1}) = \mathcal{R}_{j^p}^{-p}(\hat{\mathbf{y}}^{p+1} + \hat{\mathbf{y}}^p). \tag{3.10}$$

Then the reconstructed vector $\hat{\mathbf{x}}^1$ of the input vector $\mathbf{x}^1$ is given by

$$\hat{\mathbf{x}}^1 = \hat{\mathbf{y}}^1(j^1, j^2, \ldots, j^P) = \bigsqcup_{P}^{1} \mathcal{Q}_{j^p}^{-p}(\hat{\mathbf{y}}^{p+1}). \tag{3.11}$$

In the above we define $\hat{\mathbf{y}}^{P+1} = \mathbf{0}$.

A simple way of visualizing the structure of a P-stage rRQ is the following. Fold a sheet of paper over onto itself P times and open a hole in it. This will create 2^P holes in the paper. After unfolding the paper each hole will represent a codevector in the codebook. The resulting codebook satisfies the reflection symmetry condition at each stage. The creases of the folded sheet represent the stagewise boundaries. Note that when the stagewise boundaries have been fixed (after folding the paper P times) the codebook is uniquely determined by the choice of $\mathbf{y}_0^P$. This fact, that resuts from the reflection symmetry constraint, suggests a strong structure in the codebook. From a practical point of view. this introduces some difficulties in the joint optimization of all the stages. The reflection symmetry constraint also increases slightly the encoding and decoding complexity. However, since rRQ has only two codevectors at each stage, the overall complexity is still very low and allows us to design codebooks with a huge number of equivalent codevectors.

3.2 Optimum Design of rRQ Codebook

We now derive necessary conditions for the optimality of rRQ. The average distortion is given by

$$E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} = \sum_{\mathbf{j}^P} E\{d(\mathbf{x}^1, \hat{\mathbf{y}}^1(\mathbf{j}^P))|\mathbf{x}^1 \in S^e(\mathbf{j}^P)\}\Pr(\mathbf{x}^1 \in S^e(\mathbf{j}^P)). \qquad (3.12)$$

Now, making use of the fact that $d(\cdot, \cdot)$ is translation invariant and the reflection operation preserves the Euclidean distance, we can write the expression for the distortion as

$$E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} \quad = \textstyle\sum_{\mathbf{j}^P} E\{d(\bigcup_{p=1}^{\rho-1} Q_{j^p}^p(\mathbf{x}^1), \bigcup_{p=P}^{\rho} Q_{j^p}^{-P}(\hat{\mathbf{y}}^{\rho+1}))|\mathbf{x}^1 \in S^e(\mathbf{j}^P)\}$$

$$\cdot \Pr(\mathbf{x}^1 \in S^e(\mathbf{j}^P)) \tag{3.13}$$

$$= \textstyle\sum_{\mathbf{j}^P} E\{d(\mathbf{x}^p, \hat{\mathbf{y}}^p(\grave{\mathbf{j}}^p))|\mathbf{x}^1 \in S^e(\mathbf{j}^P)\}\Pr(\mathbf{x}^1 \in S^e(\mathbf{j}^P)) \tag{3.14}$$

where $\hat{\mathbf{y}}^p(\grave{\mathbf{j}}^p)$ is the partially reconstructed codevector. Applying the reflection operation to both vectors yields

$$\begin{aligned} d(\mathbf{x}^p, \hat{\mathbf{y}}^p) &= d(\mathcal{R}^p_{j^p}(\mathbf{x}^p), \mathcal{R}^p_{j^p}(\hat{\mathbf{y}}^p)). & (3.15)\\ &= d(\grave{\mathbf{x}}^p, \hat{\mathbf{y}}^{p+1} + \hat{\mathbf{y}}^p), & (3.16)\\ &= d(\grave{\mathbf{x}}^p - \hat{\mathbf{y}}^{p+1}, \hat{\mathbf{y}}^p). & (3.17)\end{aligned}$$

By defining the pth **reflected** graft residual $\grave{\xi}^p$ as

$$\grave{\xi}^p = \grave{\mathbf{x}}^p - \hat{\mathbf{y}}^{p+1}(\grave{\mathbf{j}}^{p+1}) \text{ if } \mathbf{x}^1 \in S^e(\mathbf{j}^P) \tag{3.18}$$

the above equations yield

$$E\{d(\mathbf{X}^1, \hat{\mathbf{X}}^1)\} = \sum_{\mathbf{j}^P} E\{d(\grave{\xi}^p, \hat{\mathbf{y}}^p(\grave{\mathbf{j}}^p))|\mathbf{x}^1 \in S^e(\mathbf{j}^P)\}\Pr(\mathbf{x}^1 \in S^e(\mathbf{j}^P)). \tag{3.19}$$

Finally, the condition to be satisfied by the stagewise codevectors $\hat{\mathbf{y}}^p$ is

$$E\{d(\grave{\xi}^p, \hat{\mathbf{y}}^p)|\grave{\mathbf{x}}^p \in \grave{S}^p\} = \min_{\mathbf{u} \in \mathbb{R}^n} E\{d(\grave{\xi}^p, \mathbf{u})|\grave{\mathbf{x}}^p \in \grave{S}^p\} \tag{3.20}$$

for $(1 \leq p \leq P)$. Although this seems similar to equation (2.23), the result in equation (3.20) differs in that if the stagewise boundaries are held fixed then we can only optimize one stagewise codevector $\hat{\mathbf{y}}^p$ since the rest will be uniquely determined by $\hat{\mathbf{y}}^p$ and the stagewise boundaries. In practice, it is preferable to optimize both stagewise codevectors at each stage, instead of just the reflected codevector. The stagewise boundary is determined by the two codevectors and is updated whenever the stagewise codevectors change.

This method however, does not guarantee that the resulting codebook will not be entangled. Moreover, it is still not always possible to simultaneously satisfy the above condition for all P stagewise codevectors without violating the reflection symmetry condition. This introduces some difficulties during the design process that can be attributed to the highly structured nature of rRQ codebooks.

3.2.1 An algorithm for rRQ design

In the design algorithm of rRQ we make use of certain ad hoc techniques in order to overcome the difficulties in the design process. One problem in the design algorithm is the fact that unlike the well known LBG algorithm, the rRQ design algorithm does not guarantee that the distortion will decrease after each iteration. Thus the distortion after each iteration is not monotonically decreasing. This, of course, poses the question of whether this algorithm is convergent to at least a locally optimal solution or not. Unfortunately, there is no proof of convergence of the design algorithm. In fact, in many cases the algorithm does not converge, and the distortion tends to oscillate between two different values. Therefore certain correction steps are necessary in order to obtain a reasonable solution by using the rRQ codebook design algorithm.

Following is a description of the algorithm that was used in our simulations.

1. Set the number of stages $P = Rd$, where R is the rate and d is the dimension of the rRQ. Choose the stopping criteria ϵ_1, ϵ_2 and ϵ_3. Let $\{\mathbf{x}_k^1\}_{k=1}^{L}$ be the sequence of the training vectors.

2. Set $P_{cur} = 0$.

3. Increment P_{cur} and use the splitting algorithm to select two initial codevectors for stage P_{cur}.

4. Compute the midpoints and normal vectors for stage P_{cur}:

$$\mathbf{m}^{P_{cur}} = \tfrac{1}{2}(\mathbf{y}_0^{P_{cur}} + \mathbf{y}_1^{P_{cur}}) \tag{3.21}$$

$$\hat{\mathbf{n}}^{P_{cur}} = \frac{\mathbf{y}_1^{P_{cur}} - \mathbf{y}_0^{P_{cur}}}{||\mathbf{y}_1^{P_{cur}} - \mathbf{y}_0^{P_{cur}}||} \tag{3.22}$$

5. Improve the codebook of stage P_{cur}. Set $d_{prev} = \infty$.

   (a) Set $\mathbf{c}_1 = \mathbf{c}_0 = \mathbf{0}$, $n_1 = n_0 = 0$ and $d_{cur} = 0$.

   (b) For all training vectors do:

      i. For $j = 1$ to P_{cur} do:

         A. Compute the distance $l_j = \hat{\mathbf{n}}^j \cdot (\mathbf{x}^j - \mathbf{m}^j)$

         B. If $l_j > 0$ set $\dot{\mathbf{x}}^j = \mathbf{x}^j - 2l^j\hat{\mathbf{n}}^j$ /* If necessary reflect the input */

         C. Compute the reflected residual $\mathbf{x}^{j+1} = \dot{\mathbf{x}}^j - \mathbf{y}_0^j$.

      ii. If $l_{P_{cur}} > 0$ then let $\mathbf{c}_1 = \mathbf{c}_1 + \mathbf{x}^{P_{cur}}$, and increment n_1.

      iii. Else let $\mathbf{c}_0 = \mathbf{c}_0 + \mathbf{x}^{P_{cur}}$ and increment n_0.

      iv. Let $d_{cur} = d_{cur} + \mathbf{x}^{P_{cur}+1} \cdot \mathbf{x}^{P_{cur}+1}$.

   (c) Let $d_{cur} = d_{cur}/L$, $\mathbf{y}_0^{P_{cur}} = \mathbf{c}_0/n_0$, $\mathbf{y}_1^{P_{cur}} = \mathbf{c}_1/n_1$.

   (d) Update the midpoint $\mathbf{m}^{P_{cur}}$ and normal vectors $\hat{\mathbf{n}}^{P_{cur}}$ using the equations of step 4.

   (e) Let $d_{rel} = (d_{prev} - d_{cur})/d_{cur}$ and $d_{prev} = d_{cur}$.

   (f) If $d_{rel} > \epsilon_1$ go to step 5a.

6. Jointly optimize all stages from 1 to P_{cur}.

   (a) Let $D_{prev} = \infty$.

```
(b) For $j = 1$ to $P_{cur}$ do:
    i. Let $d_{prev} = \infty$.
    ii. Set $\mathbf{c}_0 = \mathbf{c}_1 = \mathbf{0}$, $n_0 = n_1 = 0$, $d_{cur} = 0$.
    iii. For all training vectors do:
        • For $k = 1$ to $P_{cur}$ do:
            – Set $t_k = \hat{\mathbf{n}}^k \cdot (\mathbf{x}^k - \mathbf{m}^k)$
            – If $t_k > 0$ set $\grave{\mathbf{x}}^k = \mathbf{x}^k - 2t_k\hat{\mathbf{n}}^k$.
            – $\mathbf{x}^{k+1} = \grave{\mathbf{x}}^k - \mathbf{y}_0^k$
        • Partially reconstruct the codevector:
            A. If $j < P_{cur}$ set $\grave{\mathbf{y}}^j = 0$.
            B. If $j < P_{cur}$ for $k = P_{cur}$ to $j + 1$ do:
                – Set $\grave{\mathbf{y}}^j = \grave{\mathbf{y}}^j + \mathbf{y}_0^k$.
                – If $t_k > 0$ then let $\grave{\mathbf{y}}^j = \grave{\mathbf{y}}^j + 2t_k\hat{\mathbf{n}}^k$    /* Reflect it if
                  necessary */
            C. Else let $\grave{\mathbf{y}}^j = \mathbf{x}^{P_{cur}}$ . /* i.e. if $j = P_{cur}$ */
            D. If $t_j > 0$ then let $\mathbf{c}_1 = \mathbf{c}_1 + \grave{\mathbf{y}}^j$, and increment $n_1$.
            E. Else let $\mathbf{c}_0 = \mathbf{c}_0 + \grave{\mathbf{y}}^j$, and increment $n_0$.
        • Set $d_{cur} = d_{cur} + \mathbf{x}^{P_{cur}+1} \cdot \mathbf{x}^{P_{cur}+1}$.
    iv. Set $d_{cur} = d_{cur}/L$, $\mathbf{y}_0^j = \mathbf{c}_0/n_0$, $\mathbf{y}_1^j = \mathbf{c}_1/n_1$.
    v. Update $\mathbf{m}^j$ and $\hat{\mathbf{n}}^j$ using the equations of step 1.
    vi. Let $d_{rel} = (d_{prev} - d_{cur})/d_{prev}$ and $d_{prev} = d_{cur}$.
    vii. If $d_{rel} > \epsilon_2$ go to step 6(b)ii.
(c) Let $D_{rel} = (D_{prev} - d_{cur})/D_{prev}$ and let $D_{prev} = d_{cur}$.
```

(d) If $D_{prev} > \epsilon_3$ go to step 6b.

7. If $P_{cur} < P$ go to step 3. Else exit with $\{\mathbf{y}_0^j, \mathbf{y}_1^j\}_{j=1}^{P_{cur}}$ as the codebook.

Although the algorithm presented here is detailed there are still some programming difficulties associated with it. One of the major problems of rRQ is the fact that it has "too much structure" in it. Being more precise, the problem is related to the fact that a small modification of the codebook of a given stage will violate the reflection symmetry in all subsequent stages, thus requiring an appropriate correction of the stagewise codebooks of those stages too. This makes it increasingly difficult to satisfy both the optimality conditions and the reflection symmetry property in all stages, as the number of stages increases. This problem makes it necessary to limit the joint optimization procedure to a small number of stages. Large rRQ codebooks are obtained by direct concatenation of the appropriate number of rRQs with smaller rate and therefore smaller number of stages. In our simulations, the joint optimization was done for 8 stages at most.

Secondly, as it was mentioned in the beginning of this subsection, the main iteration loop of this algorithm does not guarantee a monotonically decreasing distortion. Two options are available to the programmer in order to obtain a solution. The first approach is to abandon the optimization process of a given stage as soon as an increase in the average distortion on the training sequence is observed. The second approach, is a delayed decision approach, in which the optimization is continued even when the distortion increases. If after a certain number of iterations the distortion is still higher than the minimum distortion obtained so far, then the optimization is abandoned and the best stagewise codebook is restored. Simulations show that the first method is too conservative and results in higher final distortions than the second one. The delayed method results in much better codebooks at the cost of additional time required for the design process.

3.2.2 Discussion of rRQ

In order to obtain a qualitative picture of the potential of rRQ we present in the following pages a few two dimensional rRQ codebooks designed for Gaussian and generalized Gaussian sources, both with and without memory. Figures 3.2 and 3.3 show the rRQ codebooks for a memoryless Gaussian source at rates 1.5 and 3 bps, respectively. Similarly, Figures 3.4 and 3.5 depict similar codebooks for the case of a Gauss-markov source with correlation coefficient $\rho = 0.9$. Finally, Figures 3.6 and 3.7 show codebooks for the generalized Gaussian source [17] with parameter 0.5 for rates 1.5 and 2.5 bps respectively. In these figures, asterisks denote the equivalent codevectors that correspond to the elements of H_0^1, whereas crosses denote those codevectors that correspond to the elements of H_1^1.

The topology of the codebooks for the Gaussian source reveals some similarity between these codebooks and the codebooks obtained by scalar quantizers. In particular, Figures 3.2 and 3.3 suggest that in this case the rRQ codebook is very similar to the codebook of a uniform scalar quantizer except that it is slightly rotated. This rotation is more or less random and depends on the way that the a new stagewise pair of codevectors is introduced by using a splitting technique. Since the source pdf has a spherical symmetry this rotation does not affect the performance of the codebook. In the Gauss-Markov case we get a different picture. In this case the source pdf does not have the spherical symmetry property anymore and this has an effect on the support region of the rRQ codebook. The shape of the support region is still rectangular (in the 2-dimensional space) but now it has a definite orientation that matches that of the principal axis of the elliptical contours of constant pdf. This is a desirable property, since in effect, it minimizes the overload probability over all rectangular support regions of the same size, a condition which, asymptotically in rate, is satisfied by optimal codebooks [7]. An intuitive interpretation of this behavior is the following: the input training sequence is first rotated around the origin by $\pi/4$. Then an rRQ codebook is designed using the resulting training sequence. Finally the codebook is rotated around the origin by $-\pi/4$. Note that this rotation is equivalent to the Karhunen-Loeve transform (KLT) for the given input vector [18]. The rRQ quantization process for Gauss-Markov sources is very similar to a transform coding system that is based on KLT and uniform scalar quantization. This example shows that rRQ is capable of utilizing the source memory efficiently and with very low complexity.

Figures 3.6 and 3.7 reveal a situation different from that in the previous figures. The generalized Gaussian source for $\alpha = 0.5$ is a source with very peaked pdf. In this case, resolution-constrained scalar quantization yields very poor performance since its equivalent 2-dimensional support region is far from the optimal support region. In this case, rRQ actually outperforms a scalar quantizer since its support region is closer to the optimal support region that minimizes the overload probability for a given volume.

The above observations show some of the main characteristics of rRQ, and qualitatively sketch its capabilities and its weaknesses.

3.2.3 Complexity of rRQ

rRQ is a highly structured residual quantizer that can be used for the realization of huge equivalent codebooks without imposing excessive complexity requirements. Here, we determine the memory and computational complexity of the encoding and decoding systems of rRQ. In the following we will be referring to an rRQ with dimensionality n and rate R bits per sample. These parameters completely determine the complexity of the rRQ. Since this is a binary residual quantizer, i.e., at each stage the stagewise codebook has only two codevectors, each stage will correspond to a binary decision and thus contribute one bit in the resulting codeword. Since the total number of bits per vector is Rn this will also equal the number of stages P. Finally, the size of the equivalent codebook is of course $N = 2^{Rn}$.

At each stage we compute the Euclidean distance of the input from the stagewise boundary and if necessary reflect it. Then we subtract from it the stagewise codevector. The distance computation requires 2 additions and 1 multiplication per sample. The reflection corresponds to 1 addition and one multiplication. Finally the last step is merely 1 addition. This gives us a total of 4 additions and 2 multiplications per sample per stage, i.e, the encoder's computational complexity is $6Rn$ operations in total per sample. Roughly, the decoder's computational complexity is the same, except that the distance computation is needed only when reflection is necessary.

The memory requirements of rRQ are as follows. For each stage, two codevectors, one midpoint and one normal vector, need to be stored. Therefore the memory complexity of rRQ is $4Rn^2$.

3.2.4 Experimental Results

We designed and simulated rRQs for several synthetic sources and the results are presented in the following pages. The sources that were tested are the Gaussian and Laplacian memoryless sources and the 1st-order Markov sources with Gaussian and Laplacian innovations and correlation coefficient $\rho = 0.9$. Figures 3.8 and 3.9 show the signal-to-noise ratio (SNR) for the memoryless Gaussian source as a function of the dimension. The SNR is mostly independent of the dimensionality and at high rates, when the number of stages is large, the SNR decreases slightly as the dimension increases. This should not be surprising and can be explained as follows: In subsection 3.2.2 we mentioned that for a memoryless Gaussian source, rRQ behaves similarly to a uniform scalar quantizer. Therefore its performance should not depend on the dimensionality. In practice however, as the dimension increases, so does the number of stages and therefore it becomes increasingly difficult to jointly optimize the total number of stages. This results in an increased average distortion due to the suboptimal design.

In Figures 3.10 and 3.11 the SNR for a Gauss-Markov source is shown. In this case, we see a clear improvement of the performance as a function of the dimension. The additional gain achieved by increasing the dimension is due to the utilization of the memory of the source. In order to provide more insight on how efficiently rRQ utilizes the intra-vector memory, we compare rRQ to a transform coding system based on KLT and Lloyd-Max scalar quantization [18] (KLT-SQ). The bit allocation was performed in an optimal fashion using the algorithm in [19, 20]. The results suggest that rRQ behaves similarly to KLT-SQ. As we allow the dimension of the vector to increase, both systems achieve approximately the same memory gain and the difference in their performance is approximately constant. Note that the complexity of KLT is proportional to n^2, and therefore the rRQ system achieves the memory gain with a lower complexity. The complexity of other suboptimal transforms, like the DCT, is proportional to $n \log n$ which is still higher than the complexity of rRQ.

Figure 3.12 shows the SNR for a memoryless Laplacian source. We see that in this case, unlike the memoryless Gaussian source, a small improvement in the SNR occurs by increasing the dimension. Figure 3.13 shows the SNR for a 1st-order Markov source with correlation coefficient $\rho = 0.9$ and Laplacian innovations. Here, as expected, the SNR improves significantly as the dimension increases and saturates approximately at dimension 32.

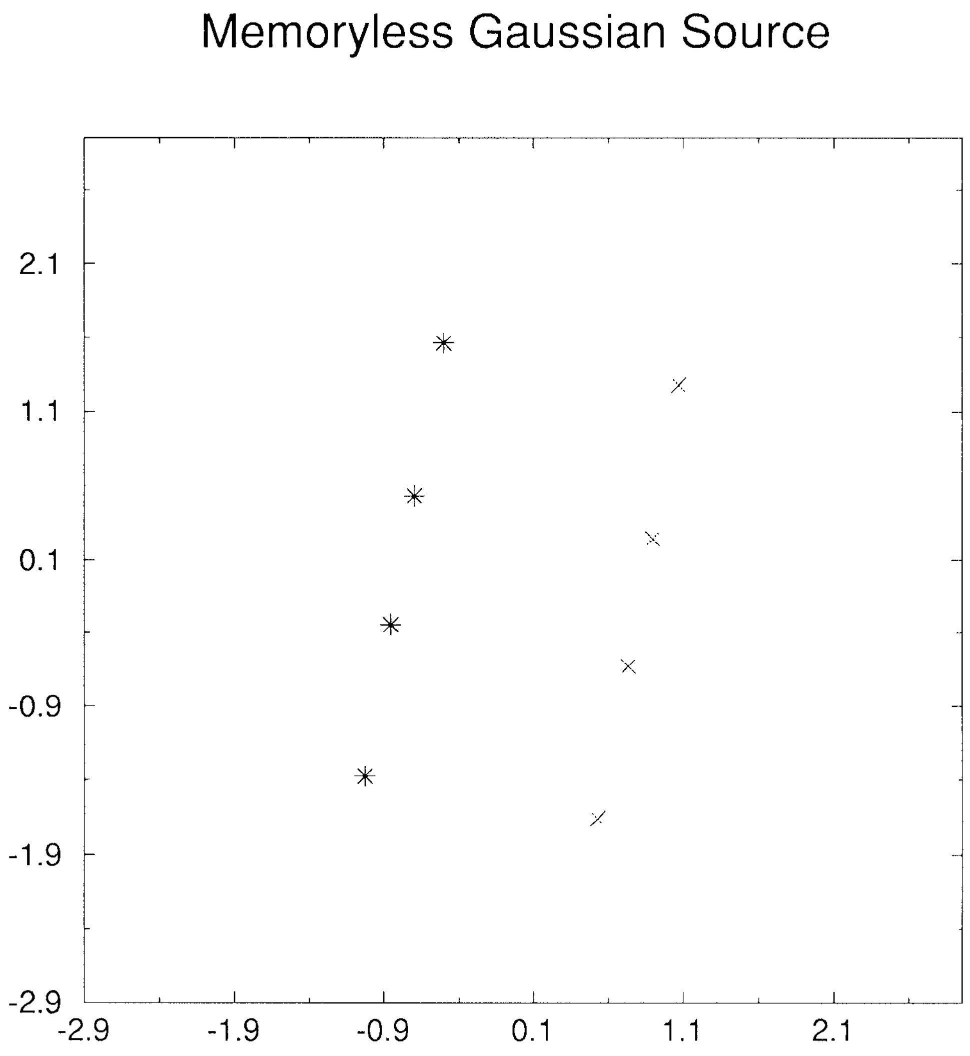

Figure 3.2: rRQ codebook with $n = 2$ and $R = 1.5$ bps.

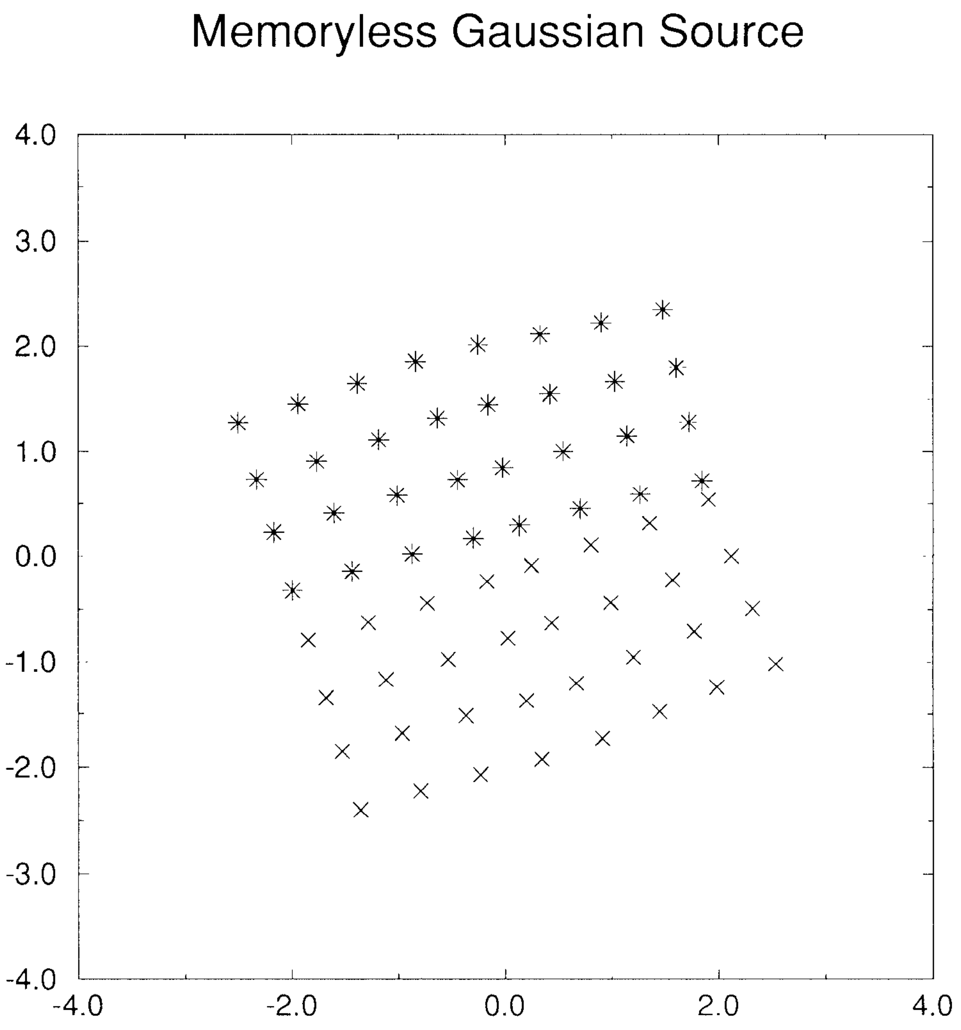

Figure 3.3: rRQ codebook with $n = 2$ and $R = 3$ *bps*.

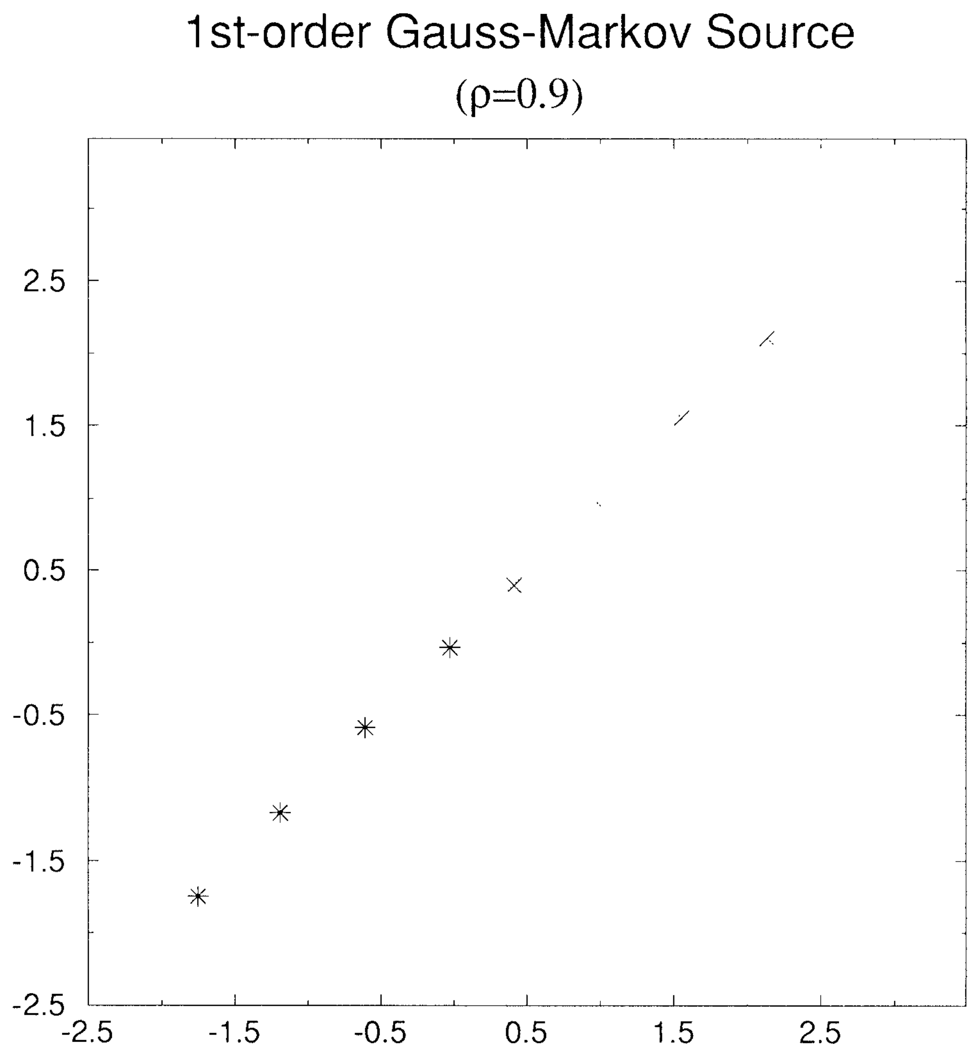

Figure 3.4: rRQ codebook with $n = 2$ and $R = 1.5$ bps.

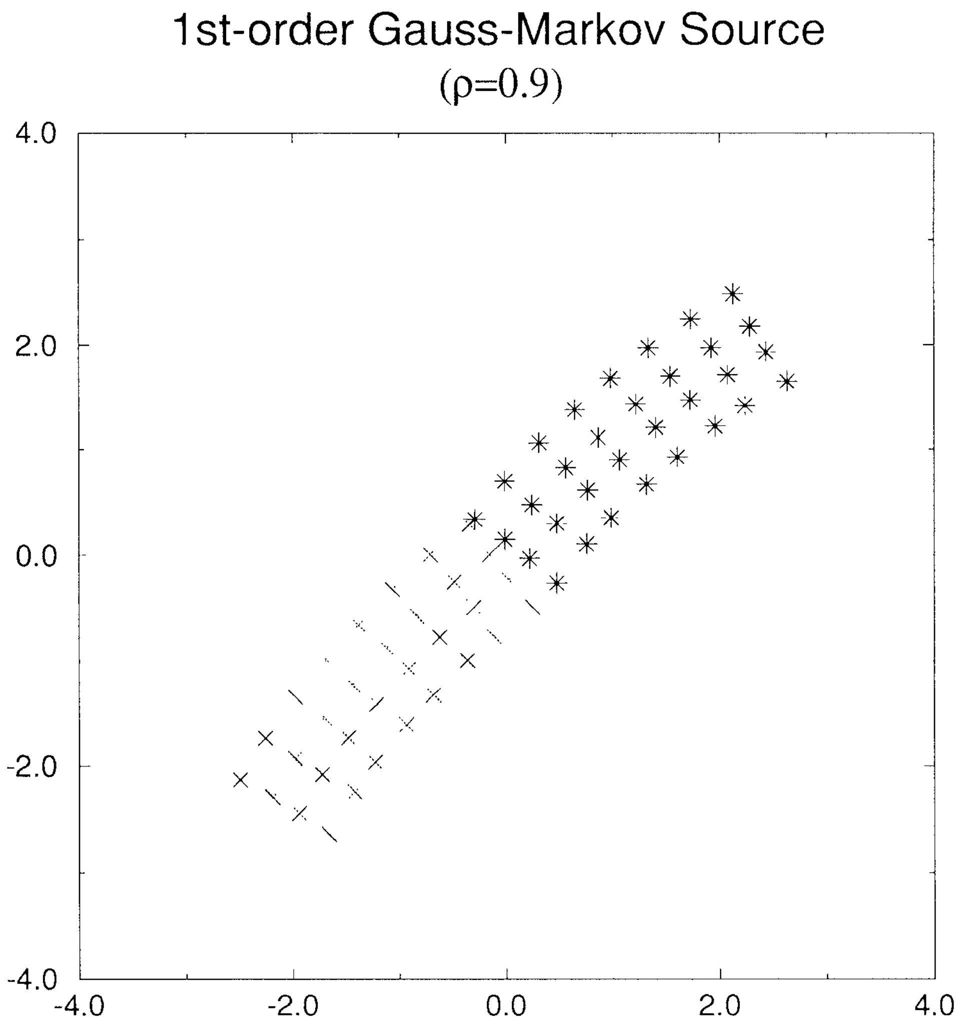

Figure 3.5: rRQ codebook with $n = 2$ and $R = 3$ *bps*.

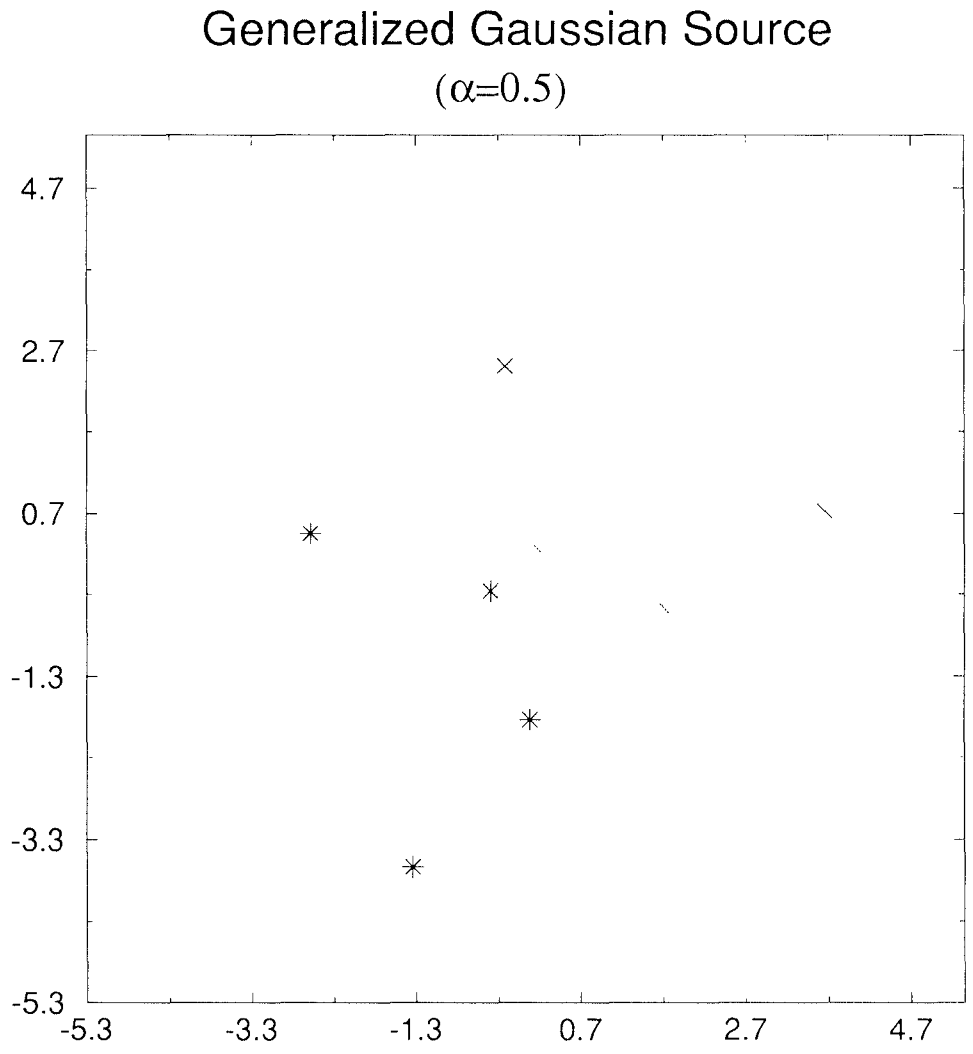

Figure 3.6: rRQ codebook with $n = 2$ and $R = 1.5$ *bps*.

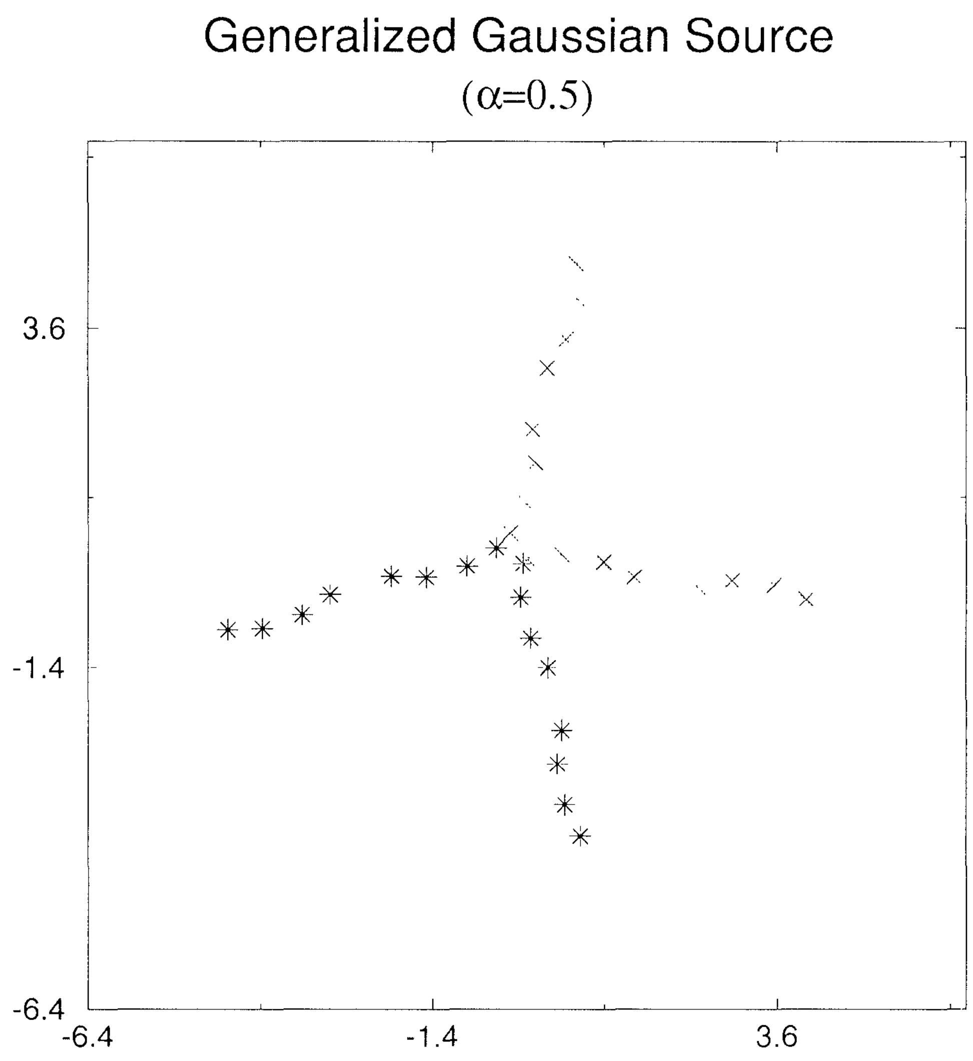

Figure 3.7: rRQ codebook with $n = 2$ and $R = 2.5$ bps.

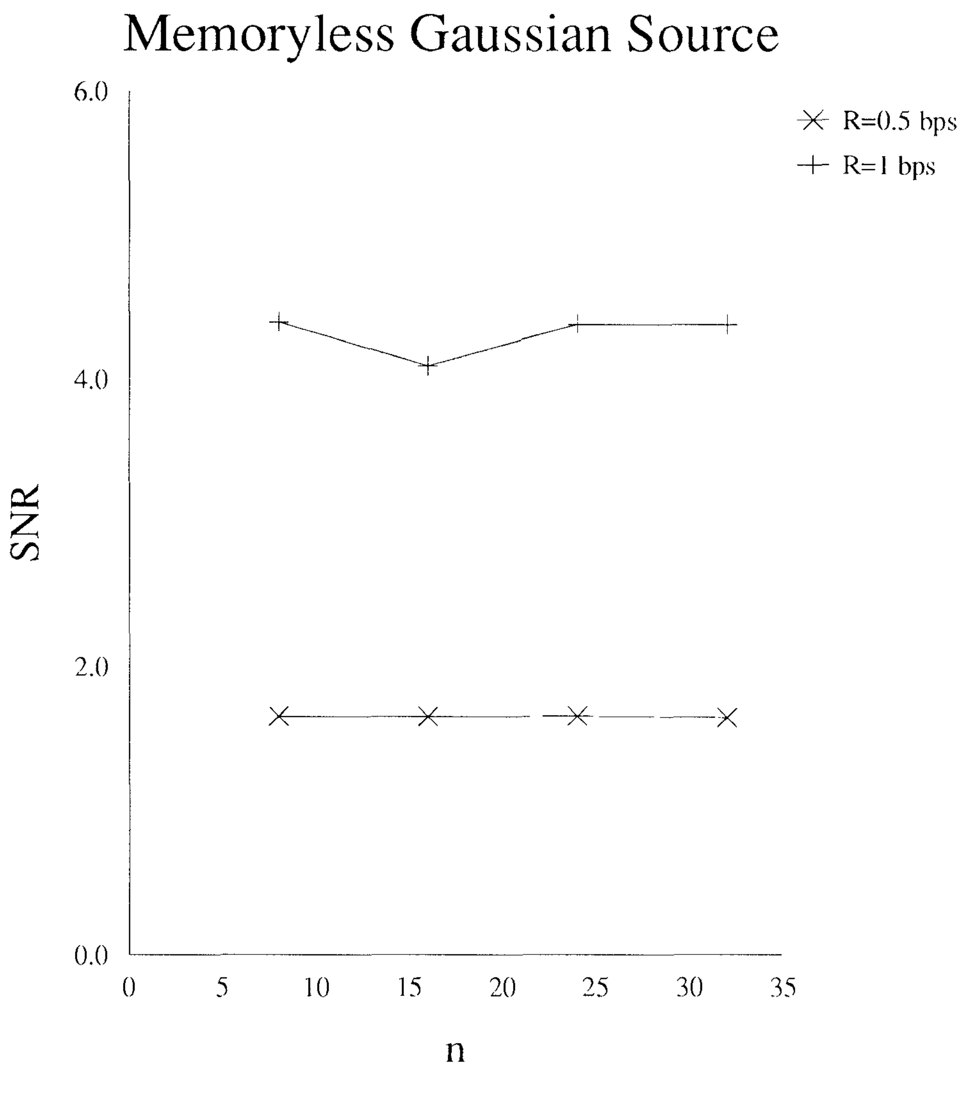

Figure 3.8: Performance of rRQ on memoryless Gaussian source for rates 0.5 and 1 bps.

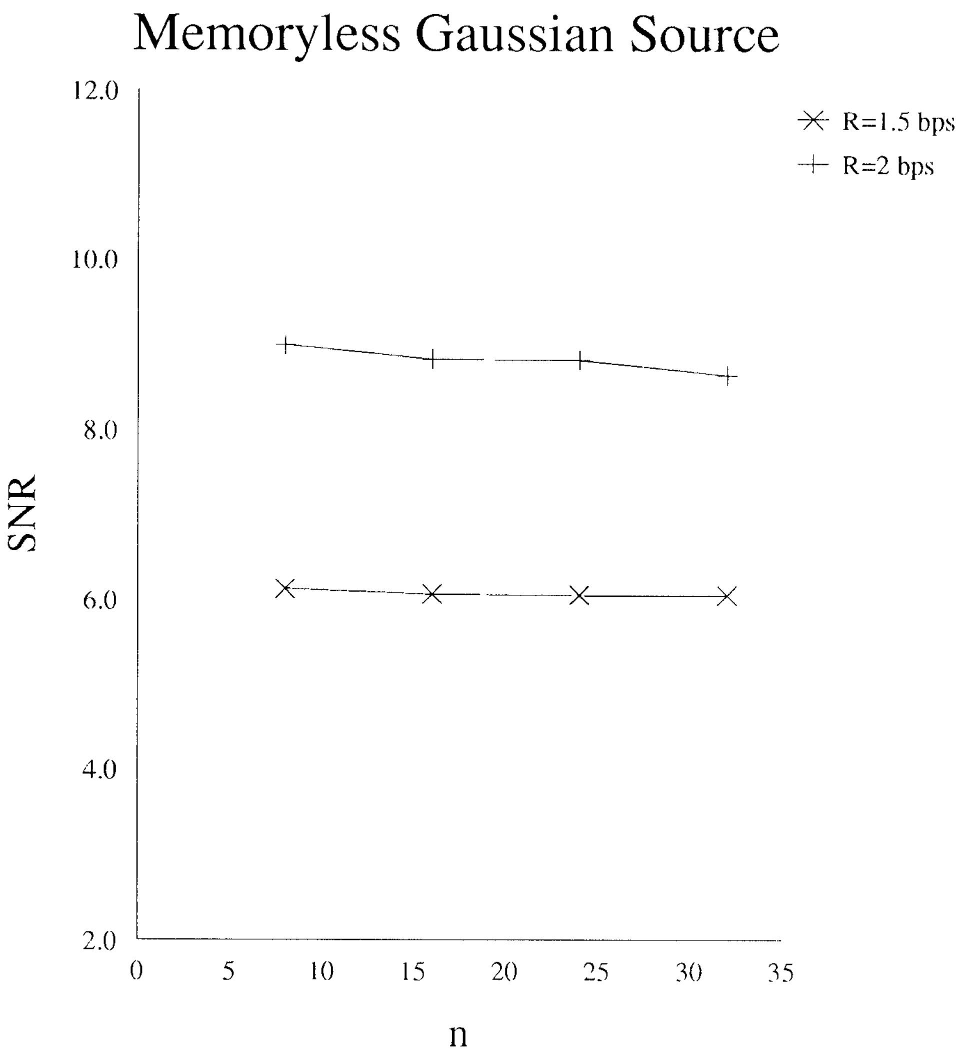

Figure 3.9: Performance of rRQ on memoryless Gaussian source for rates 1.5 and 2 bps.

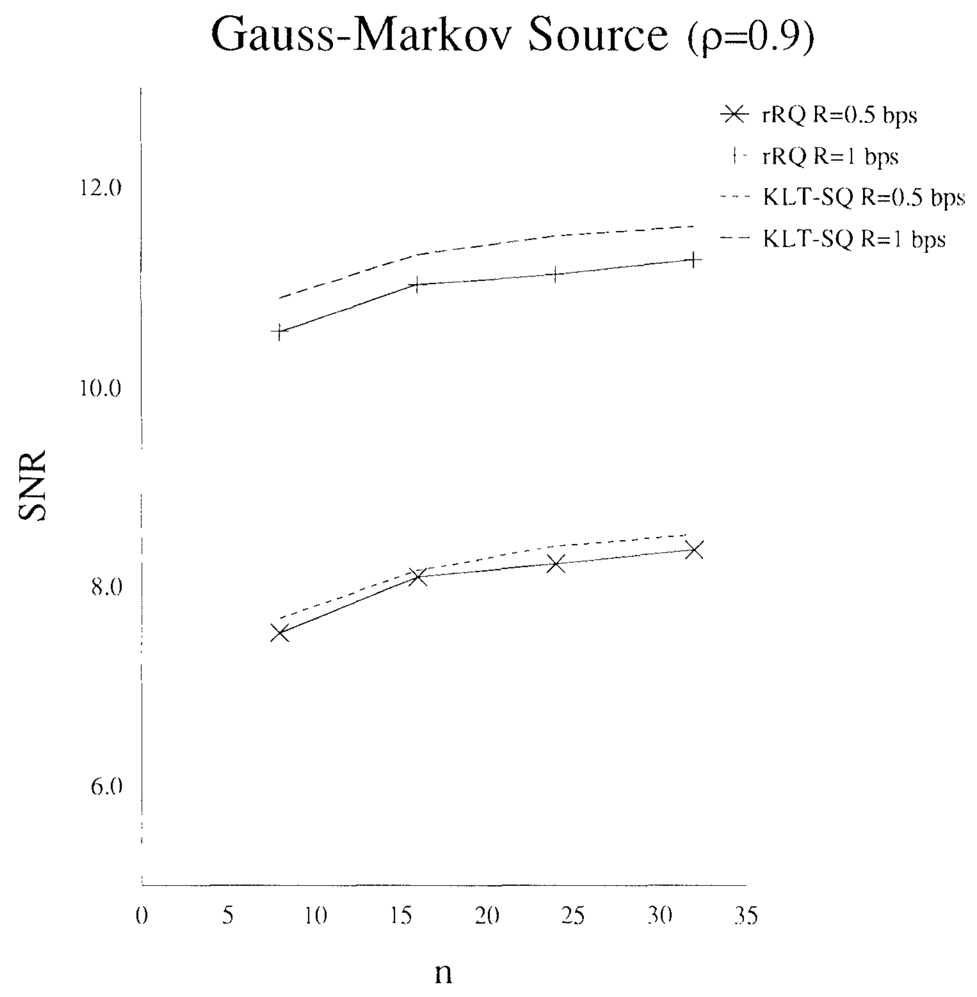

Figure 3.10: Performance of rRQ and KLT-SQ on Gauss-Markov source for rates 0.5 and 1 bps.

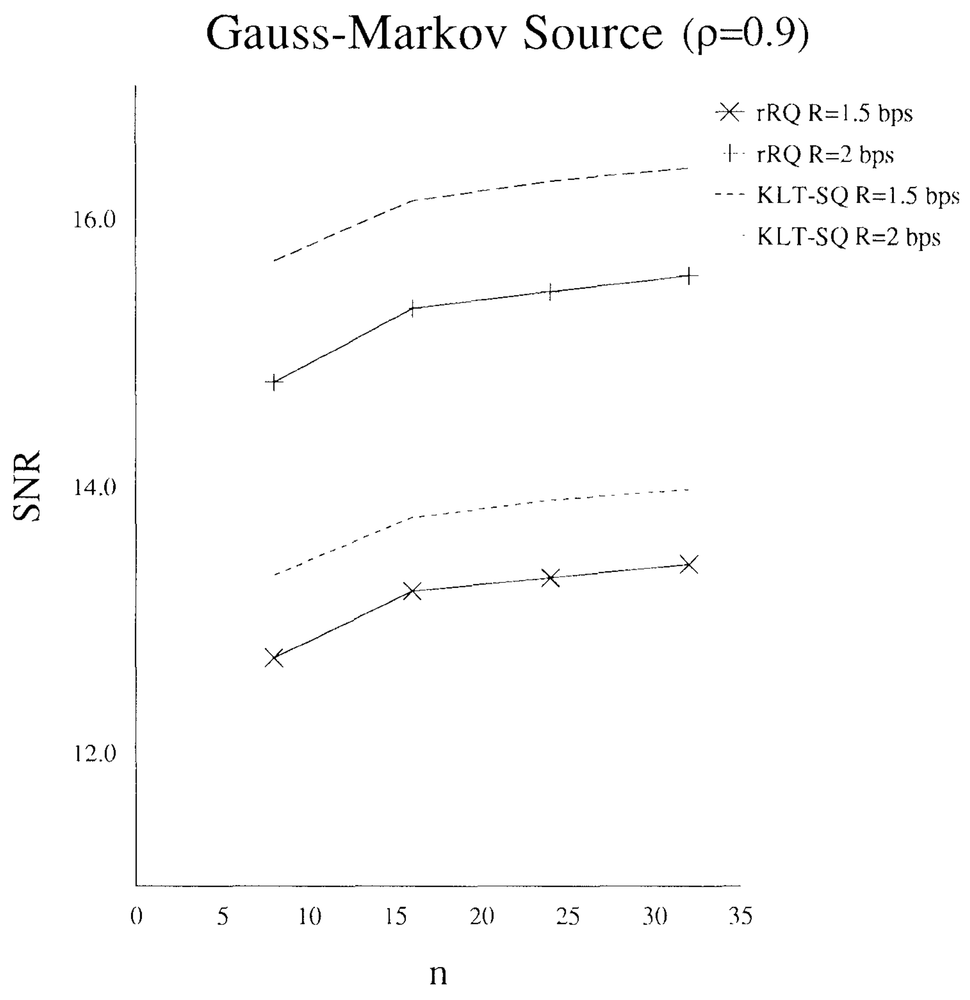

Figure 3.11: Performance of rRQ and KLT-SQ on Gauss-Markov source for rates 1.5 and 2 bps.

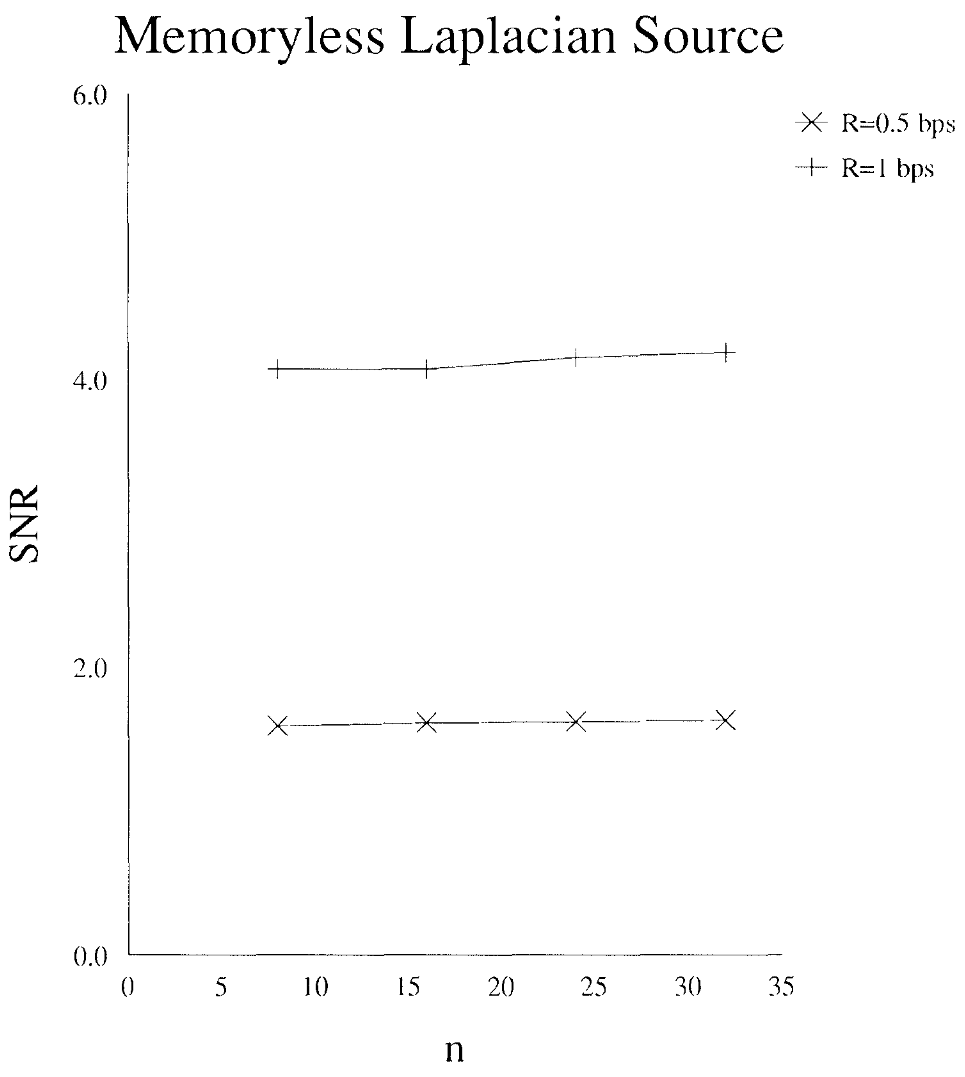

Figure 3.12: Performance of rRQ on memoryless Laplacian source for rates 0.5 and 1 bps.

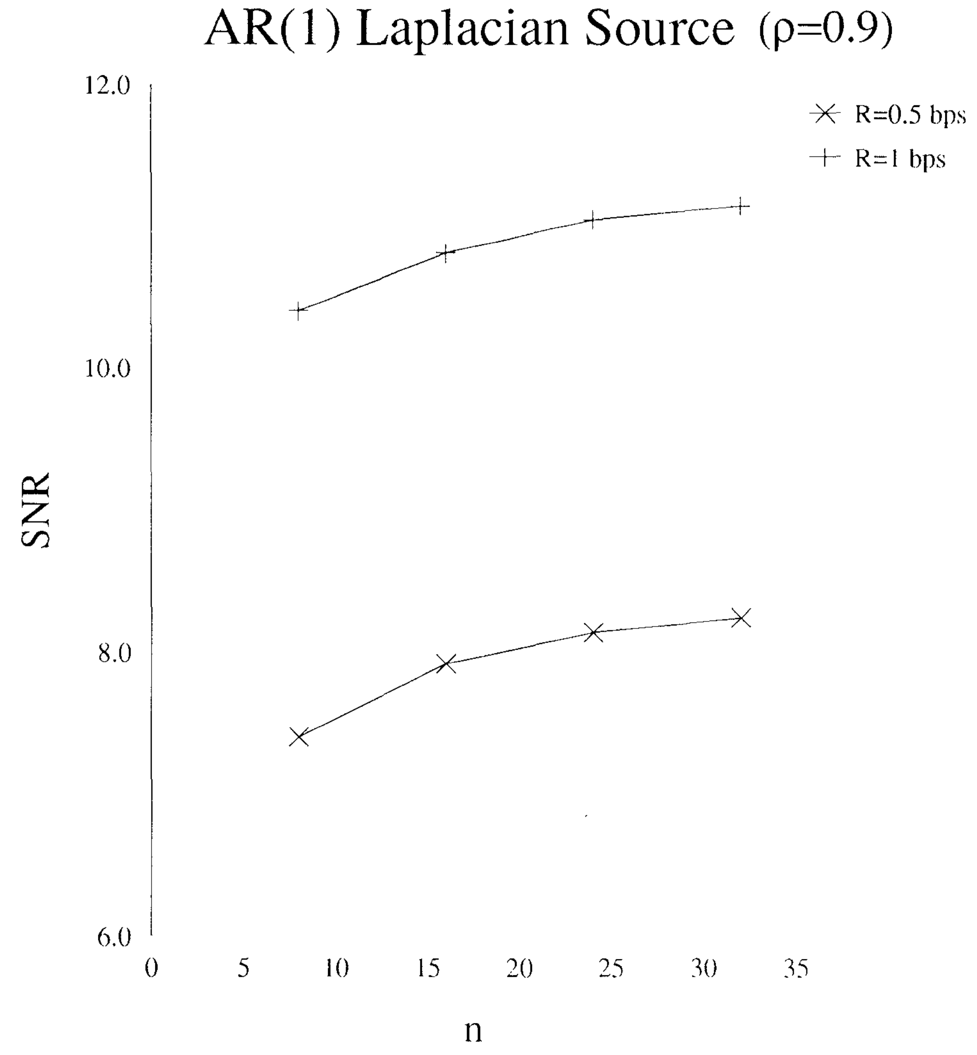

Figure 3.13: Performance of rRQ on Markov source with Laplacian innovations process source for rates 0.5 and 1 bps.

Chapter 4

Application to Speech Coding

In this chapter rRQ is applied to coding of speech spectral parameters. In order to quantize speech, we need to convert the speech waveform into a form that is well suited to quantizing. Linear predictive coding (LPC) parameters have become the standard method for representing the **short-term** spectral information of speech, because they are easily derived from the speech waveform and can be used for its reproduction with a great fidelity. The two major types of speech coding are the **vocoders** [21] and the Code Excited Linear Prediction (CELP) coders [22]. The first type generates speech by exciting a linear filter with pitch pulses or white noise. In CELP coders the excitation is selected from a codebook of signals. Several different representations of the LPC parameters have been used in speech coding. Log-area ratios (LARs) [23], arcsine reflection coefficients [24] and the line spectrum pair (LSP) [25] are some of the representations that have been used for coding the LPC parameters. Although all of these representations are equivalent and LPC parameters can be recovered from them, their coding properties differ from one another. In particular, the LSP parameters can be encoded more efficiently than other representations of the

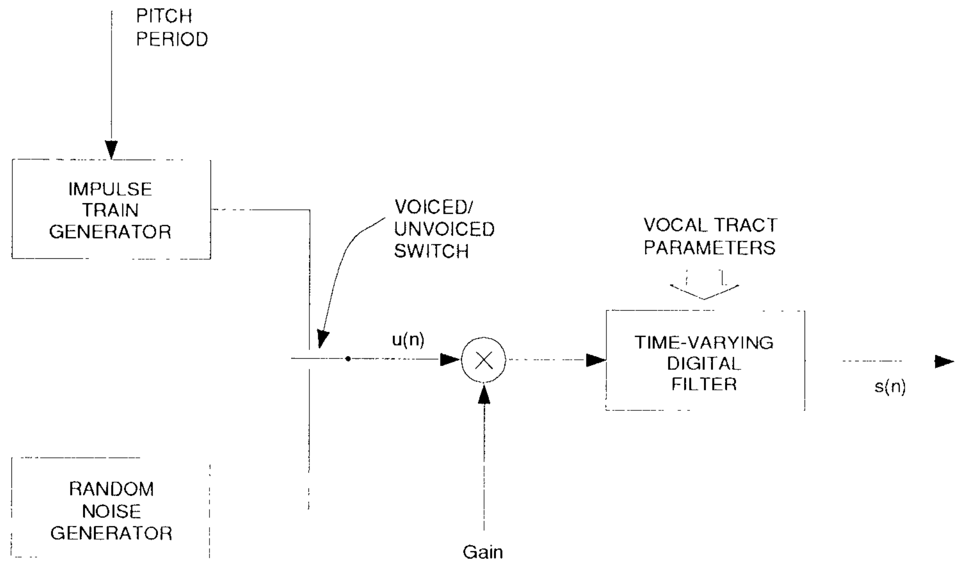

Figure 4.1: Block diagram of a simplified model for speech reproduction.

LPC parameters, and have received more attention during the last few years.

Section 4.1 reviews the basics of LPC analysis and introduces the LSP parameters. The rest of the chapter describes several methods of LSP coding based on rRQ. Section 4.2 describes the use of a finite-state VQ based on rRQ and its application to coding the LSP parameters. In Section 4.3 predictive rRQ for LSP coding is introduced. Finally, Section 4.4 describes a simple coding scheme that utilizes the inter-vector memory in a straightforward manner.

4.1 Linear Predictive Analysis

Figure 4.1 depicts a simple speech reproduction system based on linear predictive analysis. The speech waveform is divided into short frames of 20-25 ms. For each frame an LPC analysis is performed in order to obtain the parameters of the linear time-invariant model that best fits the samples of the given frame. For a given frame the transfer function of this system has the form

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}}. \tag{4.1}$$

where $A(z)$ is the inverse filter, G is a gain parameter and $\{a_k\}$ are the LPC coefficients. The input to this system is either an impulse train (voiced speech), or a random noise sequence (unvoiced speech). The parameters of this model are: a flag determining whether the given frame is voiced or unvoiced, the **pitch period** for voiced speech, the **gain** parameter G, and the filter coefficients $\{a_k\}$. The model assumes that these coefficients vary slowly with time, so that it is sufficient to update them every 20-25 ms.

According to acoustic theory, nasal and fricative sounds are represented by a transfer function with both zeros and poles. However, if the order p of the LPC analysis is high enough, the simple all-pole model of Figure 4.1 provides a very good representation for almost all speech sounds [21]. A 10th-order LPC analysis is enough to yield a good representation of speech and so it has become a standard.

The LSP parameters [26] are defined by constructing two polynomials from the inverse filter polynomial:

$$P(z) = A(z) + z^{-(p+1)}A(z^{-1}) \tag{4.2}$$

and

$$Q(z) = A(z) - z^{-(p+1)}A(z^{-1}). \tag{4.3}$$

The LSP parameters are the roots of the polynomials $P(z)$ and $Q(z)$. These two polynomials have some very interesting properties:

1. All zeroes of $P(z)$ and $Q(z)$ lie on the unit circle.

2. The zeroes of $P(z)$ and $Q(z)$ are interlaced with one another.

These properties allow for efficient numerical computation of the LSPs from $P(z)$ and $Q(z)$. Moreover, it can be shown [27] that $A(z)$ has the minimum phase property if its LSPs satisfy the above two properties. This is a useful result because the stability of the LPC synthesis filter after quantization can be checked and guaranteed in the LSP domain.

The performance of an LPC coding system is measured in terms of the **spectral distortion** measure. The spectral distortion SD for a given frame in decibels is given by

$$\mathrm{SD} = \frac{1}{F_s}\int_0^{F_s} [10\log(P(f)) - 10\log(\hat{P}(f))]^2 df, \tag{4.4}$$

where F_s is the sampling frequency in Hz, $P(f)$ and $\hat{P}(f)$ are the LPC power spectra of the original and the quantized LPC polynomials respectively, i.e.

$$P(f) = 1/|A(\exp(j2\pi f/F_s))|^2 \tag{4.5}$$

and

$$\hat{P}(f) = 1/|\hat{A}(\exp(j2\pi f/F_s))|^2. \tag{4.6}$$

and $\hat{A}(z)$ is the quantized LPC polynomial. SD is computed for each of the frames of the testing data and then its average is computed. Although the speech quality is subjective, the SD measure manages to capture many of the features that determine the quality of reproduced speech. An average SD of 1 dB has been used as a threshold for **transparent** coding of the speech spectrum, i.e. replacing the coded LSP vectors by the uncoded LSP vectors will have no audible effect on the human auditory system. However, experimental results suggest that when there are too many frames of speech with an SD much higher than the average SD, then the distortion can be audible even if the average SD is less than 1 dB. Such frames with an SD greater than a given threshold are called **outliers**. In order to have speech of transparent quality we must limit the number of outliers. Recent studies [28] indicate that the following conditions are sufficient in order to have transparent quantization of speech spectrum: 1) The average SD is less than 1 dB, 2) The number of outliers with SD in the range 2-4 dB is less than 2%, and 3) There are no outliers with SD greater than 4 dB.

We close this section by noting that the SD is usually computed over the total bandwidth of 4 kHz. However, some researchers [29] prefer to compute the SD only on a smaller bandwidth on the basis that most of the speech spectral content is concentrated in a smaller portion of the entire 4kHz bandwidth that corresponds to the standard sampling frequency of 8 kHz. Computing the SD on the 300-3,700 Hz. or the 0-3 kHz bands may yield different results that sometimes are misleading or cause confusion.

4.2 Finite-State rRQ for LSP Vectors

LSP vectors have been used in the past in speech coding with very good results. As we mentioned in the previous section, the standard order of the LPC analysis filter is $p = 10$. Coding of the LSP parameters requires a rate of 2.4-3.3 bits/component of the LSP vector in order to obtain a quantized speech spectrum of transparent quality. For this range of bits per vector it is not possible to design and use an LBG VQ due to the prohibitively large memory and computational complexity. Therefore other forms of quantizers have been applied to coding the LSP vectors. Scalar quantization [30], hybrid vector-scalar quantizer [31], multi-stage VQ and split VQ [29].

In this section we describe a finite-state VQ based on rRQ which we call FS-rRQ. The rationale behind the idea of using a finite-state quantizer for the LSP vectors resides in the fact that there exists significant correlation between successive LSP vectors. The strong correlation between successive LSP vectors suggests that the spectral envelope of speech varies slowly with time. This inter-vector (inter-frame) correlation can be utilized by the quantizer in order to obtain improved performance without increasing the complexity of the system significantly. One way of achieving this gain is through the use of a finite-state encoder.

4.2.1 Definition of FS-rRQ and an algorithm for FS-rRQ design

A finite-state rRQ is defined similarly to a finite-state VQ (FSVQ) [32]. An n-dimensional K-state FS-rRQ is specified by a state space $\mathcal{S} = \{1, 2, \ldots, K\}$, an initial state s_0 and three mappings:

1. $\alpha : \mathbb{R}^n \times \mathcal{S} \rightarrow \mathcal{N}$: finite-state equivalent encoder.

2. $\beta : \mathcal{N} \times \mathcal{S} \rightarrow \mathcal{A}$: finite-state equivalent decoder.

3. $f : \mathcal{N} \times \mathcal{S} \rightarrow \mathcal{S}$: next state function,

where α and β are explained below. $\mathcal{N}$ and $\mathcal{A}$ are the channel alphabet and the reproduction space, respectively.

Given an input sequence $\{\mathbf{x}_i\}$, let $\{\mathbf{j}_i\}$, $\{s_i\}$ and $\{\hat{\mathbf{x}}_i\}$ denote the codeword sequence, state sequence, and reproduction sequence, respectively. These are determined as follows:

$$\mathbf{j}_i = \alpha(\mathbf{x}_i, s_i) \tag{4.7}$$

$$\hat{\mathbf{x}}_i = \beta(\mathbf{j}_i, s_i) \tag{4.8}$$

$$s_{i+1} = f(\mathbf{j}_i, s_i) \tag{4.9}$$

for $i = 0, 1, \ldots$. For each $s \in \mathcal{S}$, $\alpha(\cdot, s)$ and $\beta(\cdot, s)$ are the equivalent encoder and decoder of an rRQ respectively. The reproduction space $\mathcal{A}$ can be written as $\mathcal{A} = \cup_{k=1}^{K} \mathcal{A}_k^e$ where $\mathcal{A}_k^e$ is the equivalent rRQ codebook associated with state k.

A FS-rRQ design algorithm is given below:

1. Design an LBG VQ with K codevectors for the given training sequence. This is called the state-label VQ. Denote its codebook by $\mathcal{C} = \{\mathbf{c}(k), k \in \mathcal{S}\}$.

2. For each state k design an initial rRQ using the training subsequence that is obtained by keeping only those vectors in the training sequence for which the previous vector is represented by k when encoded by the state-label VQ.

3. Define the next state function f by

$$f(\mathbf{j}, k) = \arg\min_{s \in \mathcal{S}} d(\beta(\mathbf{j}, k), \mathbf{c}(s)), \quad k \in \mathcal{S}, \mathbf{j} \in \mathcal{N}. \tag{4.10}$$

4. Iteratively improve the codebooks $\{\mathcal{A}_k, k \in \mathcal{S}\}$ in the following manner: Using the current codebooks and the next state function defined in Step 3 obtain a new state sequence $\{s_i\}$ using equation (4.9). Then partition the training sequence into K training subsequences based on the state

sequence $\{s_i\}$ and update each of the K rRQs using the updated training subsequence. Repeat this step as long as the average distortion decreases significantly.

The above algorithm is not guaranteed to converge but in most practical cases it actually does. Typically, 2 or 3 iterations of the last step are sufficient to get close to what seems to be a good solution. Note, that the finite-state structure of the quantizer effectively increases the size of the codebook. This is done at the expense of a K-times increase of the memory complexity. The computational complexity is practically unaffected. Finally the design procedure is more complex and requires a longer training sequence since each state rRQ uses its own training subsequence.

In our experiments we have kept the size of all the state rRQ codebooks fixed. Performance improvements have been reported when the above constraint is dropped [33].

Fs-rRQ was designed and used to encode a sequence of LSP vectors. The number of states used was $K = 4$ or 8. Typically, introduction of more states increases the complexity without substantially improving the performance of FS-rRQ. The results are given in Table 4.3 and discussed in Section 4.5.

4.3 Predictive rRQ Coding of LSP Parameters

In this section we describe a predictive VQ system based on rRQ (P-rRQ) and its application to speech spectral coding. Predictive VQ was introduced in [34] as a way of incorporating memory in an otherwise memoryless VQ system. Introducing memory to a VQ system enables it to deal more efficiently with vector

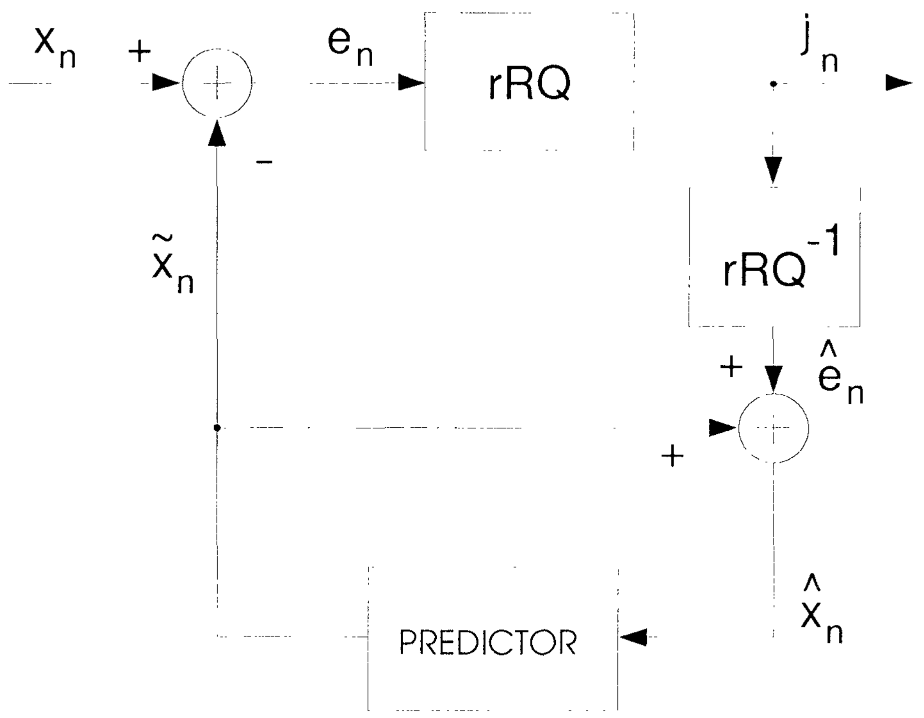

Figure 4.2: Predictive rRQ encoder

sources with statistical dependence between successive vectors. One such example of a vector source with memory is the sequence of LSP vectors. Each vector can be thought of as a realization of a given random vector with the same pdf, but there exists a strong correlation between successive vectors. This correlation reflects the fact that the short-term spectral envelope of speech changes slowly from frame to frame.

P-rRQ is a system that consists of a vector predictor and an rRQ in a closed-loop form. The encoder, shown in Figure 4.2, operates in the following manner: The difference $\mathbf{e}_n$ between the input vector $\mathbf{x}_n$ and its prediction $\tilde{\mathbf{x}}_n$ is vector quantized using the rRQ. The output symbol $\mathbf{j}_n$ is output to the channel, while its reconstruction $\hat{\mathbf{e}}_n$ is added to $\tilde{\mathbf{x}}_n$ and the resulting vector $\hat{\mathbf{x}}_n$ is fed to the

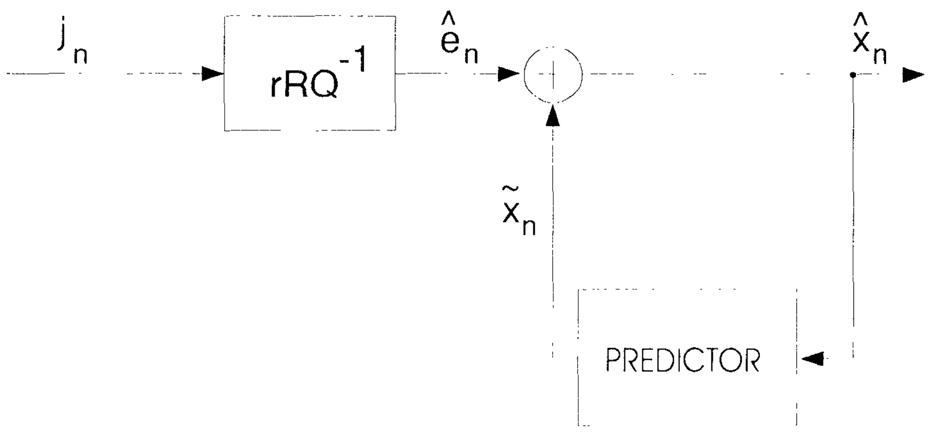

Figure 4.3: Predictive rRQ decoder

predictor which in turn will produce a prediction $\tilde{\mathbf{x}}_{n+1}$ of the next input vector. Note that $\hat{\mathbf{x}}_n$ is just the reconstructed input vector. The decoder, depicted in Figure 4.3, works in the following way: The channel symbol $\mathbf{j}_n$ is used to obtain a reconstruction $\hat{\mathbf{e}}_n$ of the error vector, which is then added to the prediction $\tilde{\mathbf{x}}_n$. This yields the reconstructed source vector $\hat{\mathbf{x}}_n$ which is then used to obtain the next prediction. Note that the encoder includes a copy of the decoder.

Before discussing the structure of the vector predictor that appears in both the encoder and the decoder, we will describe an algorithm used in the design of P-rRQ.

4.3.1 Predictive rRQ design

Designing a predictive VQ system involves the design of a predictor and the design of a VQ. There are two main approaches to this problem [10]. The simplest approach to designing such a system is the **open-loop** methodology. In this method, the predictor is designed for the input source $\{\mathbf{x}_n\}$ to be quantized.

Then using this predictor, the sequence of prediction residuals $\{\mathbf{e}_n\}$ is obtained, where

$$\mathbf{e}_n = \mathbf{x}_n - P(\mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \ldots, \mathbf{x}_{n-m}). \tag{4.11}$$

In the above equation, $P(\cdot)$ denotes the predictor and m is the predictor order. The next step is to design a VQ using the sequence $\{\mathbf{e}_n\}$ as the training sequence. Finally, the predictor and the VQ are combined to obtain the system of Figures 4.2 and 4.3. Note that both the predictor and the VQ are now suboptimal. The predictor in the closed-loop system operates on the sequence $\{\hat{\mathbf{x}}_n\}$ of the reconstructed vectors instead of the source sequence, whereas the VQ operates on an error sequence which differs from the one that was used during the design process, since only the reconstructed vectors $\hat{\mathbf{x}}_n$ are now available to the predictor. However, if the VQ operates at high rates, the suboptimality of the design may not be too severe.

Another design approach that results in a better performance is the **closed-loop** method [10]. According to this method, first we obtain an initial VQ and predictor using the open-loop design. Then the system is used to encode the training sequence and the resulting sequence of residuals is used to run one iteration of the VQ design algorithm. This process is repeated until a stopping criterion is satisfied, or a maximum number of iterations is exceeded. A description of the algorithm that we have used in our experiments is given below:

1. Given a training sequence $\{\mathbf{x}_n\}_{n=-m+1}^{L}$, a predictor $P(\cdot)$ and stopping threshold $\epsilon > 0$, obtain the sequence $\{\mathbf{e}_n\}_{n=1}^{L}$ as in 4.11.

2. Design an rRQ for the training sequence $\{\mathbf{e}_n\}_{n=1}^{L}$. Set $k = 0$ and $D_0 = \infty$.

3. Let $k = k + 1$. Encode the training sequence $\{\mathbf{x}_n\}$ using the current P-rRQ

and update the residual sequence $\{\mathbf{e}_n\}$ in the following way:

$$\begin{aligned} \mathbf{e}_n &= \mathbf{x}_n - \grave{\mathbf{x}}_n & (4.12) \\ \hat{\mathbf{x}}_n &= \grave{\mathbf{x}}_n + \hat{\mathbf{e}}_n & (4.13) \\ \grave{\mathbf{x}}_{n+1} &= P(\hat{\mathbf{x}}_n, \hat{\mathbf{x}}_{n-1}, \ldots, \hat{\mathbf{x}}_{n-m+1}). & (4.14) \end{aligned}$$

where $\hat{\mathbf{e}}_n$ is the reconstruction of $\mathbf{e}_n$. Compute the average distortion D_k between the input and the output of the system. If $(D_{k-1} - D_k)/D_k < \epsilon$, then stop.

4. Using the updated residual sequence, run one iteration of the rRQ design algorithm. Go to step 3.

Although there is no proof of convergence or optimality of the above algorithm, our results indicate that the improvement over the open loop design method justifies the additional computational effort.

In the above algorithm no effort is made to optimize the predictor for the actual sequence $\{\hat{\mathbf{x}}_n\}$. A truly closed-loop design method will attempt to jointly optimize the VQ and the predictor. This case is investigated by Chang and Gray in [35]. The resulting improvement in the performance of the overall system is insignificant and does not justify the effort.

Next, we address the issue of what form of predictor to use with the P-rRQ system. Several different predictor structures and their design are described briefly.

4.3.2 Predictor forms used with P-rRQ

There are two main categories of predictors, the linear, and the nonlinear predictors. The optimal finite-memory predictor under the squared-error distortion measure is in general nonlinear. However, for a stationary Gaussian random process the optimal predictor and the optimal linear predictor coincide [36].

Consider the stationary random vector sequence $\{\mathbf{X}_n\}$ and let n be the dimension of the vectors. An mth-**order vector linear predictor** is defined by a set of m, $n \times n$ linear predictor coefficient matrices, $\mathbf{A}_j$, for $j = 1, 2, \ldots, m$. The estimate $\check{\mathbf{X}}_n$ of $\mathbf{X}_n$ is formed according to

$$\check{\mathbf{X}}_n = \sum_{j=1}^{m} \mathbf{A}_j \mathbf{X}_{n-j}. \tag{4.15}$$

Note that the predictor described above utilizes the correlation between the current vector and **every** component of the m previous vectors. However, in the case of a sequence of LSP vectors, the correlation between different components of successive vectors is very low and most of the prediction gain can be achieved by exploiting only the correlation between the same components of successive vectors. This, essentially, amounts to looking at 10 different scalar sources instead of one 10-dimensional vector source. This reduces the complexity of the predictor significantly without sacrificing much of its gain.

If the source does not have zero mean, as is the case for the LSP vector source, the predictor performance can be improved by subtracting the mean from the source before the predictor is applied to it and adding the mean to the predictor output. Notice that this procedure is equivalent to using an affine predictor of the form

$$\check{X}_n = \sum_{j=1}^{m} a_j X_{n-j} + b. \tag{4.16}$$

The above equation describes the linear (scalar) predictor that was used in our experiments. For the cases $m = 1$ and $m = 2$ that we tested the optimal values of the predictor coefficients, determined using the orthogonality principle [10], are as follows:

$$a_1 = C[1]/C[0], \quad b = (1 - C[1]/C[0])E\{X\}, \tag{4.17}$$

for $m = 1$, and

$$a_1 = \frac{C[1](R[2] - R[0])}{(R[1] - R[0])(C[1] + C[0])} \tag{4.18}$$

$$a_2 = \frac{C[1] - a_1 C[0]}{C[1]} \tag{4.19}$$

$$b = E\{X\}(1 - a_1 - a_2), \tag{4.20}$$

for $m = 2$, where

$$R[i] = E\{X_n X_{n+i}\}, \quad C[i] = R[i] - E\{X\}^2, \tag{4.21}$$

are the autocorrelation and autocovariance functions of the stationary process $\{X_n\}$, respectively. The parameters concerning the scalar 1st and 2nd-order linear predictors are shown in Table 4.1.

The optimal (non-linear) mth-order scalar predictor with respect to the squared-error distortion measure takes the form [10]

$$\check{X}_n = E\{X_n | X_{n-1}, X_{n-2}, \ldots, X_{n-m}\}. \tag{4.22}$$

This predictor can be realized approximately by using a quantizer $\mathcal{Q}$ of sufficient resolution and estimating the above expectation by

$$\check{X}_n = E\{X_n | \mathcal{Q}(X_{n-1}), \mathcal{Q}(X_{n-2}), \ldots, \mathcal{Q}(X_{n-m})\}. \tag{4.23}$$

In order to quantify the benefits of a given predictor, we define the **closed-loop prediction gain ratio** [10] by

$$G_{\text{clp}} = \frac{E\{||\mathbf{X}_n||^2\}}{E\{||\mathbf{e}_n||^2\}}. \tag{4.24}$$

LSP #	1st-order		2nd-order		
	a_1	b	a_1	a_2	b
0	0.75	82.56	0.72	0.05	78.41
1	0.73	142.79	0.77	-0.05	150.50
2	0.74	211.04	0.82	-0.11	233.25
3	0.81	231.03	0.89	-0.10	254.32
4	0.85	243.93	1.03	-0.22	296.51
5	0.82	359.90	0.95	-0.17	421.21
6	0.81	441.75	0.96	-0.18	519.58
7	0.79	580.64	0.89	-0.13	656.93
8	0.74	815.45	0.77	-0.03	842.13
9	0.75	885.79	0.77	-0.03	914.55

Table 4.1: The 1st and 2nd-order scalar linear predictors for each LSP parameter.

The coding gain ratio of the rRQ is defined by

$$G_{\mathrm{Q}} = \frac{E\{||\mathbf{e}_n||^2\}}{E\{||\mathbf{e}_n - \hat{\mathbf{e}}_n||^2\}}. \tag{4.25}$$

Then the overall system's signal-to-noise power ratio is given by

$$\frac{E\{||\mathbf{X}_n||^2\}}{E\{||\mathbf{X}_n - \hat{\mathbf{X}}_n||^2\}}. \tag{4.26}$$

or in dBs,

$$\mathrm{SNR}_{\mathrm{sys}} = 10\log_{10} G_{\mathrm{clp}} + \mathrm{SNR}_{\mathrm{Q}}. \tag{4.27}$$

where $\mathrm{SNR}_{\mathrm{Q}} = 10\log_{10} G_{\mathrm{Q}}$ is the signal-to-noise ratio of the rRQ applied on $\mathbf{e}_n$. Assuming that $\mathrm{SNR}_{\mathrm{Q}}$ is approximately equal to the SNR obtained when rRQ operates on the signal $\mathbf{X}_n$. Equation (4.27) shows that the predictor gain

LSP #	$Var\{\mathbf{X}_n\}$	Linear 1		Linear 2		Non-linear	
		Var$\{\mathbf{e}_n\}$	G_{clp}	Var$\{\mathbf{e}_n\}$	G_{clp}	Var$\{\mathbf{e}_n\}$	G_{clp}
0	15430.0	7780.99		7815.91		7772.80	
1	21291.4	10426.18		10348.58		10317.12	
2	38038.2	17600.21		17435.98		17616.22	
3	58769.5	19844.04		19602.58		19766.39	
4	76034.8	23480.57	2.59	22433.07	2.64	23342.63	2.60
5	71831.6	25566.35		25001.87		25162.88	
6	56321.0	19678.57		18919.86		19591.32	
7	50949.5	21277.52		21013.24		21238.41	
8	34494.0	16380.67		16356.87		16390.43	
9	22076.2	9457.75		9418.21		9440.13	

Table 4.2: Variances of the residual error sequences and the predictor gains for the 1st and 2nd-order linear predictors and the 1st-order nonlinear predictor.

is a meaningful additive gain. Table 4.2 shows the variances of the LSP components, the variances of each component of the residual error vectors $\mathbf{e}_n$ and the total prediction gain for the 1st-order and 2nd-order linear predictors defined by Equations (4.17)-(4.20).

However, it turns out that in the case of the LSP vector source, the nonlinear predictor offers insignificant improvement over the linear predictor and therefore its additional complexity is not justified. Furthermore, the 1st-order linear predictor achieves most of the gain that can be achieved by linear prediction. The results on speech coding obtained by P-rRQ, given in Table 4.4, are based on a simple scalar 1st-order linear predictor. The results are discussed in Section 4.5.

4.4 Matrix rRQ Coding of Speech

In this section we describe and evaluate another scheme for LSP coding based on rRQ. We use the term **matrix rRQ** to refer to the following simple quantization method. Consider an arbitrary vector source $\{\mathbf{X}_i\}$ of dimension n. A matrix rRQ is an rRQ of dimension ln for some integer l, that operates on l successive source vectors which are regarded as a single ln-dimensional vector. Clearly, the main advantage of such a scheme is that it utilizes the correlation between successive source vectors. This property makes it suitable for coding the LSP vectors, since they are highly correlated. Note, that although this is a very simple idea, it cannot be realized by using an arbitrary VQ due to complexity limitations. The extremely low complexity of rRQ, enables us to design and operate rRQs with very large rates and/or dimensions.

There are two important observations to be made about matrix rRQ. The first concerns the possible limitations to its dimensionality. The complexity of rRQ is sufficiently low, so that rRQ with equivalent codebooks of enormous size can be designed and used. In theory, rRQ is designed by jointly optimizing the total number of stages. In practice, however, this joint optimization procedure is not possible when the number of stages is too high. Thus, only a limited number of stages can be jointly optimized at a time. Therefore, unlike the LBG VQ whose performance improves with the dimensionality, the performance of rRQ may in fact become worse by increasing the dimensionality too much. This imposes a limitation on the dimension of matrix rRQ.

The second observation is about the encoding delay of the system. The encoder outputs one codeword for every l input LSP vectors. Since each LSP frame corresponds to approximately 20 ms of speech, the total encoding delay is approximately $l \cdot 20$ ms. Typically, in a real-time two-way communication system the encoding delay should not exceed 40 ms. This limits the value of l to 2. However, it should be noted that for certain applications longer encoding delays may be acceptable. The matrix rRQ results that we've obtained are for the cases of $l = 2$ and 3, and are given in Table 4.5. In the following section the results for all the systems presented so far are discussed and compared.

4.5 Experimental Results and Discussion

The speech database used for training the various quantizers discussed in this chapter consisted of 120 min of speech from the TIMIT database. The speech was sampled at 8 KHz and contains utterances from several male and female speakers. A $10th$-order LPC analysis based on the standard autocorrelation method was performed every 22.5 msec using a 30 ms analysis window. The size of the training sequence that was actually used depended on the quantization scheme. P-rRQ required a smaller training sequence of 40,000 vectors, whereas matrix rRQ and FS-rRQ with 4 states required 100,000 training vectors. Finally, the 8-state FS-rRQ was designed using 200,000 training vectors. The testing sequence in all cases consisted of 12,000 vectors outside the training sequence.

The performance of each quantization scheme was measured in terms of the spectral distortion and the number of outliers. The spectral distortion was computed over the 0-4 KHz frequency band using Equation (4.1). In the reconstruction of the LSP vectors we found that the following technique decreased both the spectral distortion and the number of outliers: After quantization by a residual quantizer, the LSP coefficients are not necessarily ordered any more. This may lead to an unstable filter in the speech reproduction system. Therefore the decoder should check whether this condition is violated and fix the problem by slightly modifying the coefficients that are not in order. Furthermore, our experiments showed that even better results are obtained if we require that the LSP coefficients be at least 100 Hz apart. This condition was imposed by our decoder simulator.

Tables 4.3-4.5 show the results of the various LSP coding schemes that were described in this chapter. In Table 4.3 we see that an 8-state FS-rRQ saves about 1 bit/frame over the 4-state FS-rRQ. The less complex P-rRQ (based on a scalar 1st-order linear predictor) achieves better performance than FS-rRQ, requiring approximately 2 bits/frame less than the 8-state FS-rRQ. This suggests that the simple 1st-order linear predictor utilizes the inter-frame correlation more efficiently than FS-rRQ. Finally, the matrix-rRQ for $l = 2$ performs approximately 1 bit/frame worse than P-rRQ, whereas for $l = 3$ its performance is close to that of P-rRQ. It is worth mentioning that (for the same SD) matrix rRQ produces fewer outliers than P-rRQ. However, as it was pointed out earlier in this chapter, the long delay associated with matrix rRQ is often an undesirable property.

For comparison, in Table 4.6 we present the results of several scalar quantization coding schemes whose complexities are similar to that of P-rRQ. It can be seen that P-rRQ performs similarly to scalar quantization of the LSP differences producing slightly fewer outliers. It should be mentioned that in all our systems, rRQ was designed using the squared-error distortion measure. It is

	$K = 4$			$K = 8$		
Bits/frame	SD	2-4 dB	> 4 dB	SD	2-4 dB	> 4 dB
24	1.68	17.7	0.6	1.59	13.4	0.6
25	1.61	14.7	0.6	1.55	12.7	0.3
26	1.54	12.5	0.4	1.48	10.9	0.3
27	1.48	10.7	0.3	1.42	8.7	0.3
28	1.41	8.2	0.3	1.35	6.7	0.3
29	1.34	6.5	0.3	1.29	5.9	0.2
30	1.29	5.8	0.3	1.21	5.1	0.2
31	1.22	4.3	0.2	1.18	4.3	0.2
32	1.15	3.7	0.2	1.14	3.8	0.1
33	1.13	3.8	0.2	1.10	3.1	0.1

Table 4.3: Spectral distortion and percentage of outliers for FS-rRQ with 4 and 8 states

known [29] that certain weighted distortion measures, like the inverse harmonic mean (IHM), perform better than the simple squared-error distortion measure. Unfortunately, it is not possible to design an rRQ for such measures and this limits the performance of the systems based on rRQ.

Bits/frame	SD	2-4 dB (in%)	> 4 dB (in%)
24	1.49	10.7	0.3
25	1.41	8.5	0.3
26	1.37	7.3	0.3
27	1.33	6.7	0.3
28	1.27	5.0	0.3
29	1.21	4.7	0.2
30	1.15	3.8	0.1
31	1.10	3.1	0.1
32	1.07	3.0	0.2
33	1.01	2.3	0.1

Table 4.4: Spectral distortion and percentage of outliers for P-rRQ using a 1st-order scalar linear predictor

	l=3			l=2		
Bits/frame	SD	2-4 dB	> 4 dB	SD	2-4 dB	> 4 dB
24	1.50	10.8	0.1	1.56	13.5	0.1
25	1.43	8.5	0.1	1.50	11.3	0.1
26	1.37	6.9	0.1	1.44	9.0	0.1
27	1.32	5.4	0.0	1.39	7.1	0.1
28	1.26	4.2	0.1	1.32	6.1	0.1
29	1.21	3.5	0.0	1.27	1.5	0.0
30	1.16	2.7	0.0	1.23	4.2	0.0
31	1.12	2.3	0.0	1.18	3.3	0.0
32	1.07	1.6	0.0	1.13	2.6	0.1
33	1.03	1.5	0.0	1.07	2.0	0.0

Table 4.5: Spectral distortion and percentage of outliers for matrix rRQ with $l = 3$ and $l = 2$

Bits/frame	Parameter	SD	2-4 dB (in%)	> 4 dB (in%)
28	LSP	1.40	9.21	0.05
28	LSPD	1.25	7.36	0.05
28	ASRC	1.32	9.29	0.23
28	LAR	1.34	9.51	0.16
32	LSP	1.10	2.21	0.03
32	LSPD	1.05	3.13	0.01
32	ASRC	1.04	3.30	0.09
32	LAR	1.04	3.20	0.04
34	LSP	0.92	1.00	0.01
34	LSPD	0.86	1.10	0.01
34	ASRC	0.92	2.05	0.08
34	LAR	0.92	1.65	0.04

Table 4.6: Performance of several scalar quantizers using the LSP, LSP difference (LSPD), arcsine reflection coefficient (ASRC), and log-area ratio (LAR) representations.

Chapter 5

Conclusions and Future Work

We now present an overview of the quantization systems examined in this thesis and comment on their performance, quantization properties and limitations. These conclusions are followed by a proposal for possible future work in this area.

In Chapter 3 we investigated the performance of plain rRQ on various types of sources. The results obtained there suggest the following behavior of rRQ. For the memoryless Gaussian source rRQ behaved similarly to a scalar quantizer but an improvement over scalar quantizers was noted on more peaked memoryless sources like the generalized Gaussian source with parameter $\alpha = 0.5$. Our comparison of rRQ with KLT-SQ for the Markov sources demonstrated the ability of rRQ to utilize the memory of the source very efficiently. Therefore rRQ can be viewed as a very low complexity method of utilizing the source memory.

rRQ was incorporated in several LSP coding systems. Our results indicate that P-rRQ and matrix rRQ compared to several scalar speech coding schemes perform similarly or better. Matrix rRQ has the undesirable property of introducing a long delay. P-rRQ can be useful in speech coding systems in which complexity rather than bit rate is the major concern.

Future work may be concentrated on modifying rRQ in order to construct a system of higher complexity and better performance. One possible way of achieving this is to cascade rRQ by a lattice or LBG VQ. It may be possible that such a system achieves the memory gain through the rRQ block, and part of the boundary and granular gains through the second-stage VQ. To perform well, such a system should also be jointly optimized. It would also be desirable to enable rRQ to use more complicated distortion measures, like the IHM. This would improve the performance of speech coding systems based on rRQ.

Bibliography

[1] T. Berger, *Rate Distortion Theory*, Prentice-Hall Inc. Englewood Cliffs, New Jersey, 1971.

[2] R. M. Gray, *Source Coding Theory*, Kluwer Academic Publishers, Boston, 1990.

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

[4] A. J. Viterbi, and J. K. Omura, *Principles of Digital Communications and Coding*, McGraw-Hill, New York, 1979.

[5] R. Pile, "The transmission distortion of a source as a function of the encoding block length," *Bell Syst. Tech. J.*, vol. 47, pp. 827-885, 1968.

[6] A. Buzo, A. H. Gray, R. M. Gray, and J .D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-28, pp. 562-574, Oct. 1980.

[7] M. V. Eyboğlu, and G. D. Forney, "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.

[8] B. H. Juang, and A. H. Gray, "Multiple stage vector quantization for speech coding," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 1, pp. 597-600, Apr. 1982.

[9] C. F. Barnes, *Residual Quantizers*. PhD thesis, Brigham Young University, Provo, Utah, Dec. 1989.

[10] A. Gersho, and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1991.

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

[12] R. M. Gray, J. C. Kieffer, and Y. Linde, "Locally optimal block quantizer design," *Inform. and Control*, vol. 45(2), pp. 178-198, May 1980.

[13] K. Zeger, J. Vaisey, and A. Gersho, "Globally optimal vector quantizer design by stochastic relaxation," *IEEE Trans. Signal Process.*, vol. SP-40, pp. 310-322, Feb. 1992.

[14] C. F. Barnes and R. L. Frost, "Vector quantizers with direct sum codebooks," *IEEE Trans. Inform. Theory*, vol. IT-39, pp. 565-580, 1990.

[15] F. Jelinek, and J. B. Anderson, "Instrumentable tree encoding of information sources," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 118-119, Jan. 1971.

[16] D. Chen, and A. C. Bovik, "Visual pattern image coding," *IEEE Trans. Comm.*, vol. COM-38, pp. 2137-2146, Dec. 1990.

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

[18] N. S. Jayant,and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall Inc. Englewood Cliffs, New Jersey, 1984.

[19] B. Fox, "Discrete optimization via marginal analysis," *Management Science*, vol. 13, no. 3, pp. 210-216, Nov. 1966.

[20] A. V. Trushkin, "Optimal bit allocation algorithm for quantizing a random vector," *Problems of Inform. Transmission*, vol. 17, pp. 156-161, 1981 (Translated from Russian).

[21] L. R. Rabiner, and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall Inc. Englewood Cliffs, New Jersey, 1978.

[22] M. R. Schroeder, and B. S. Atal, "Code-excited linear prediction (CELP): High-quality speech at very low bit rates," in *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, vol. 1, Mar. 1985.

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

[24] A. H. Gray, Jr. and J. D. Markel, "Quantization and bit allocation in speech processing," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-24, pp. 459-473, Dec. 1976.

[25] F. Itakura, "Line spectrum representation of linear predictive coefficients of speech signals," *J. Acoust. Soc. Amer.*, vol. 57, p. S35, Apr. 1975.

[26] N. Sugamura, and F. Itakura, "Speech analysis and synthesis methods developed at ECL in NTT—from LPC to LSP," *Speech Commun.*, vol. 5, pp. 199-215, June 1986.

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

[28] B. S. Atal, R. V. Cox. and P. Kroon, "Spectral quantization and interpolation for CELP coders," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Glasgow. Scotland, pp. 69-72, May 1989.

[29] K. K. Paliwal, and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame," *IEEE Trans. Speech and Audio Process.*, vol. SA-1, pp. 3-14. Jan. 1993.

[30] F. K. Soong, and B. H. Juang, "Optimal quantization of LSP parameters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, pp. 394-397, Apr. 1988.

[31] T. Moriya, and M. Honda, "Speech coder using phase equalization and vector quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Tokyo, Japan, pp. 1701-1704, Apr. 1986.

[32] J. Foster, R. M. Gray, and M.O.Dunham. "Finite-state vector quantization for waveform coding," *IEEE Trans. Inform. Theory*. vol. IT-31, pp. 348-359. May 1985.

[33] Y. Hussain, and N. Farvardin, "Variable-rate finite-state vector quantization and applications to speech and image coding," *IEEE Trans. Speech and Audio Process.*, vol. SA-1, pp. 25-38, Apr. 1993.

[34] V. Cuperman, and A. Gersho, "Adaptive differential vector coding of speech," *Conference Record GlobeCom 82*, pp. 1092-1096, Dec. 1982.

[35] P. C. Chang, and R. M. Gray, "Gradient algorithms for designing predictive vector quantizers," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-34, pp. 679-690, Aug. 1986.

[36] H. V. Poor, *Signal Detection and Estimation*, Springer-Verlag, New York, 1988.