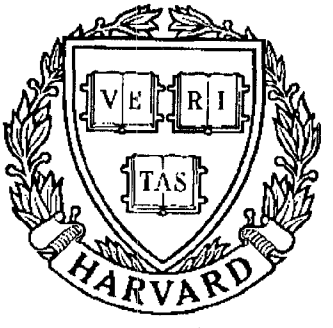
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

A Unified Approach to Tree-Structured and Multi-Stage Vector Quantization for Noisy Channels

by N. Phamdo, N. Farvardin and T. Moriya

TR 91-105

A Unified Approach to Tree-Structured and Multi-Stage Vector Quantization for Noisy Channels †

Nam Phamdo‡, *Nariman Farvardin*‡ *and Takehiro Moriya*§

‡Electrical Engineering Department
and Systems Research Center
University of Maryland
College Park, Maryland 20742

§NTT Human Interface Laboratories
Nippon Telegraph and Telephone
Corporation
3-9-11 Midori-Cho, Musashino-Shi
Tokyo, 180 Japan

Abstract

Vector quantization (VQ) is a powerful and effective scheme which is widely used in speech and image coding applications. Two basic problems can be associated with VQ: (i) its large encoding complexity, and (ii) its sensitivity to channel errors. These two problems have been independently studied in the past. In this paper, we examine these two problems jointly. Specifically, the performances of two low-complexity VQs – the tree-structured VQ (TSVQ) and the multi-stage VQ (MSVQ) – when used over noisy channels are analyzed. An algorithm is developed for the design of channel-matched TSVQ (CM-TSVQ) and channel-matched MSVQ (CM-MSVQ) under the squared-error criterion. Extensive numerical results are given for the memoryless Gaussian source and the Gauss-Markov source with correlation coefficient 0.9. Comparisons with the ordinary TSVQ and MSVQ designed for the noiseless channel show substantial improvements when the channel is very noisy. The CM-MSVQ, which can be regarded as a block-structured combined source-channel coding scheme, is then compared with a block-structured tandem source-channel coding scheme (with the same block length as the CM-MSVQ). For the Gauss-Markov source, the CM-MSVQ outperforms the tandem scheme in all cases which we have considered. Furthermore, it is demonstrated that the CM-MSVQ is fairly robust to channel mismatch.

Index Terms: Tree-structured vector quantization; multi-stage vector quantization; combined source-channel coding.

---

†This work was supported in part by National Science Foundation grants NSFD MIP-86-57311 and NSFD CDR-85-00108, and in part by NTT Corporation and General Electric Co.

I. Introduction

One of the most important results of Shannon's pioneering work was that source coding and channel coding can be treated separately without sacrificing optimality [1, 2]. This "separation principle" has led to the development of what is known as tandem source-channel coding scheme, in which the source and channel codes are separately designed but are then used in a cascade combination with one another. Shannon's separation principle holds provided that the source and channel codes are allowed to operate on blocks of arbitrarily large lengths [3]. In practice, however, due to certain constraints on the delay or complexity, the block lengths of the source and channel codes must be finite. In such situations, an alternative approach is to combine the source and channel codes into a single code in which the objective is to minimize the average distortion between the source and its reproduction. Such an approach is referred to as combined source-channel coding.

The study of combined source-channel coding for continuous-amplitude sources leads naturally to the study of quantizer design for noisy channels. Design algorithms for an optimum scalar quantizer operating over a noisy channel were developed in [4, 5]. These ideas were extended to vector quantizers (VQs) in [6, 7]. The results in [7] indicate that the VQ designed for a noisy channel has better performance than its scalar counterpart – especially when the source has memory. The performance improvement increases as the block length increases. Unfortunately, increasing the block length is not always possible since the VQ is constrained by its complexity, which grows exponentially with the product of the rate and block length.

In the context of source coding alone, i.e., when the channel is assumed to be noiseless, methods of reducing VQ complexity have been studied extensively [9, 10, 11]. Some examples of low-complexity VQs are tree-structured VQ (TSVQ), multi-stage VQ (MSVQ), product VQ and lattice VQ. For a complete description of these schemes, the interested reader is referred to [9, 12]. In this paper, we will focus

attention to the TSVQ and MSVQ. A TSVQ is a VQ scheme in which the encoder computational complexity is reduced by imposing a tree structure on the codebook [10]. Instead of quantizing the source vector using a large-size VQ, the source vector is first quantized by a small-size "primary" VQ. Depending on which of the codevectors (in the primary VQ) the source vector is closest to, a different (small-size) "secondary" VQ is then used for quantization. This process is repeated until the desired rate is obtained. Even though the encoder computational complexity is reduced using the TSVQ, the encoder memory is increased [10]. The MSVQ, on the other hand, is a VQ scheme in which both the encoder computational complexity and memory are reduced [11]. It is similar to the TSVQ except that it has the additional constraint that all secondary VQs are equivalent – up to a linear translation.

In this paper, the performances of the TSVQ and MSVQ when used over a noisy channel are analyzed. Necessary conditions for optimality of the secondary VQs given the primary VQ are developed and an algorithm for designing TSVQ and MSVQ for a given noisy channel is presented. The resulting quantizers are thus referred to as channel-matched TSVQ (CM-TSVQ) and channel-matched MSVQ (CM-MSVQ). Numerical results are presented for the memoryless Gaussian source and the Gauss-Markov source with correlation coefficient 0.9 and comparisons are made with the ordinary TSVQ and MSVQ designed for the noiseless channel. The performance of the CM-MSVQ is then compared with a tandem source-channel code which has the same block length (hence, same encoding delay) as the CM-MSVQ and consists of an optimum (LBG) VQ followed by an optimum linear block code. It is demonstrated that, for the Gauss-Markov source, the combined source-channel coding scheme using CM-MSVQ is superior to the tandem source-channel coding scheme. Study of the CM-MSVQ under channel mismatch shows that it has a favorable robustness property.

The rest of this paper is organized as follows. In Section II, preliminary issues concerning the structure of the TSVQ and MSVQ are presented. Section III includes

a description of the CM-TSVQ and CM-MSVQ problems, the necessary conditions for optimality as well as the design algorithm. Numerical results are presented in Section IV. In Section V, a description of the tandem source-channel coding scheme is given and a model of the channel and the channel code is established for comparison purposes. Channel mismatch issues are addressed in Section VI. Section VII contains a brief discussion of the complexity of the CM-TSVQ and CM-MSVQ. Finally, a summary and conclusions are presented in Section VIII.

II. Preliminaries

In the sequel we assume that the source to be encoded is a real-valued, stationary and ergodic process $\mathcal{X} = \{X_n\}_{n=0}^{\infty}$. A k-dimensional N-level vector quantizer (VQ) for $\mathcal{X}$ is in general described by an encoder mapping α and a decoder mapping β. The encoder mapping $\alpha : \mathbb{R}^k \mapsto \mathcal{J}_N \triangleq \{0, 1, \ldots, N-1\}$ is given by

$$\alpha(\mathbf{x}) = i \quad \text{if} \quad \mathbf{x} \in S_i, \quad i \in \mathcal{J}_N, \tag{1}$$

where $\mathbf{x} = (x_{nk}, x_{nk+1}, \ldots, x_{nk+k-1})$ is a block of k successive samples from $\mathcal{X}$ and $\mathcal{P} \triangleq \{S_0, S_1, \ldots, S_{N-1}\}$ is a partition of $\mathbb{R}^k$. The decoder mapping $\beta : \mathcal{J}_N \mapsto \mathbb{R}^k$ is described in terms of a reproduction codebook $\mathcal{C} = \{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{N-1}\}$ according to

$$\beta(i) = \mathbf{c}_i, \quad i \in \mathcal{J}_N. \tag{2}$$

For a given input vector $\mathbf{x}$, its reproduction is given by the quantization rule $q = \beta \circ \alpha$, where

$$q(\mathbf{x}) = \beta(\alpha(\mathbf{x})). \tag{3}$$

The rate of the VQ is

$$R = \frac{1}{k} \log_2 N, \quad \text{bits/sample.} \tag{4}$$

With respect to a distortion measure $d(\cdot, \cdot)$, a VQ is said to be optimal if the average distortion [1] $D = \frac{1}{k} E[d(\mathbf{X}, q(\mathbf{X}))]$ is minimized over all partitions $\mathcal{P}$ and reproduction

[1] Here $\mathbf{X}$ is the k-dimensional random vector associated with $\mathbf{x}$ and $E[\cdot]$ denotes the expectation.

codebooks $\mathcal{C}$. Two necessary conditions for optimality are [8],

$$S_i = \{\mathbf{x} : \ d(\mathbf{x}, \mathbf{c}_i) \leq d(\mathbf{x}, \mathbf{c}_j) \ \forall j \in \mathcal{J}_N\} \quad \forall i \in \mathcal{J}_N, \tag{5}$$

and

$$\mathbf{c}_i = \arg \min_{\mathbf{y} \in \mathbb{R}^k} E[d(\mathbf{X}, \mathbf{y}) | \mathbf{X} \in S_i] \quad \forall i \in \mathcal{J}_N. \tag{6}$$

Based on (5) and (6), Linde, Buzo and Gray [8] developed an algorithm, known as the LBG algorithm, for designing (locally) optimal VQs using a sequence of training data. The resulting VQ is often referred to as an LBG-VQ.

For a fixed k, the average distortion of the LBG-VQ decreases as R increases. Similarly, for a fixed R, the average distortion decreases as k increases. Unfortunately, the complexity required to implement the LBG-VQ increases exponentially with the product kR. Hence, in practice, complexity constraints prohibit the use of the LBG-VQ for large rate and/or dimension. The complexity can be reduced by using sub-optimal VQs, such as tree-structured VQ (TSVQ), multi-stage VQ (MSVQ), product VQ or lattice VQ [9]. In this paper, we will restrict our attention to TSVQ and MSVQ.

A. Structure of TSVQ and MSVQ

The TSVQ and MSVQ are similar in that they share a "successive approximation" characteristic. Instead of using a "full-rate" (N-level) VQ, the source vector, $\mathbf{x}$, is first approximated by a lower-rate (N_1-level) VQ with partition $\mathcal{P}_1$, codebook $\mathcal{C}_1$ and quantization rule $q_1 = \beta_1 \circ \alpha_1$ ($N_1 < N$). The quantization error of this stage,

$$\mathbf{e}_1 = \mathbf{x} - q_1(\mathbf{x}), \tag{7}$$

is then quantized by a second-stage (N_2-level) VQ with corresponding $\mathcal{P}_2, \mathcal{C}_2$ and $q_2 = \beta_2 \circ \alpha_2$. The main difference between the TSVQ and the MSVQ is that in the MSVQ, a single VQ is used in the second stage, whereas in the TSVQ a different VQ is used for each different value of α_1 [9, 10, 11]. Actually, as defined in [9, 10], the

second stage of the TSVQ quantizes $\mathbf{x} = \mathbf{e}_1 + q_1(\mathbf{x})$ directly. However, in order to bring both TSVQ and MSVQ under a unified framework, it would be easier to view the second-stage quantizer operating on $\mathbf{e}_1$ rather than $\mathbf{x}$. The quantization error of the second stage,

$$\mathbf{e}_2 = \mathbf{e}_1 - q_2(\mathbf{e}_1) = \mathbf{x} - q_1(\mathbf{x}) - q_2(\mathbf{e}_1), \tag{8}$$

can be further quantized by a third-stage VQ, and so on. Thus, on the average, a better approximation of the source vector can be obtained at each stage. In general, a TSVQ or MSVQ can have S stages, with each stage having an intermediate rate $R_s = \frac{1}{k}\log_2 N_s$, a partition $\mathcal{P}_s$, a codebook $\mathcal{C}_s$ and a quantization rule $q_s = \beta_s \circ \alpha_s$ for $s = 1, 2, \ldots, S$. Hereafter, we will assume that for each s, $\log_2 N_s$ is an integer. The overall rate of the quantizer is $R = R_1 + R_2 + \cdots + R_S$ and the overall encoder mapping is

$$\alpha(\mathbf{x}) = \sum_{s=1}^{S} M_s \alpha_s(\mathbf{e}_{s-1}), \tag{9}$$

where $\mathbf{e}_0 = \mathbf{x}$, $M_s = \prod_{r=s+1}^{S} N_r$ for $s = 1, 2, \ldots, S-1$, and $M_S = 1$. The overall decoder mapping is

$$\beta(i) = \sum_{s=1}^{S} \beta_s \left(\left\lfloor \frac{i}{M_s} \right\rfloor \bmod N_s \right), \tag{10}$$

and the overall quantization rule is given by

$$q(\mathbf{x}) = \beta(\alpha(\mathbf{x})) = \sum_{s=1}^{S} q_s(\mathbf{e}_{s-1}). \tag{11}$$

Equation (9) states that, in binary representation, the overall encoder output α is obtained by multiplexing the outputs of the intermediate encoders α_s for $s = 1, 2, \ldots, S$, with α_1 taking the $\log_2 N_1$ most significant bit (MSB) positions, α_2 taking the next $\log_2 N_2$ MSB positions, and so forth. That is, in binary representation, $\alpha = [\alpha_1 \alpha_2 \ldots \alpha_S]$. Likewise, equation (10) states that the overall decoder is the sum of the outputs of the intermediate decoders, where the inputs of the intermediate decoders are obtained by demultiplexing $\alpha = i$.

In the rate-distortion sense, the performances of the TSVQ and MSVQ are inferior to that of the LBG-VQ. This is because the overall partition of (9) and the overall reproduction codebook of (10), in general, do not satisfy the necessary conditions of (5) and (6), respectively. For fixed k, S and $\{R_s\}_{s=1}^{S}$, since the TSVQ is more general and less restrictive, it is expected to have better rate-distortion performance than the MSVQ. The computational complexities (associated with the codebook search) of the TSVQ and MSVQ are equivalent, but depending on the number of stages S their computational complexities can be significantly less than the LBG-VQ. The TSVQ requires more memory (to store the codebook) than the LBG-VQ, while the MSVQ requires the least amount of memory among the three. More will be said about the complexities of these three schemes later on.

B. Channel Noise

Channel noise can have a damaging effect on the performance of VQs. In the following, we assume that the encoder α and the decoder β are separated by a discrete memoryless channel (DMC) characterized by the random mapping $\gamma : \mathcal{J}_N \mapsto \mathcal{J}_N$. The reconstructed vector is no longer $q(\mathbf{x}) = \beta(\alpha(\mathbf{x}))$ but is a composition of the three mappings:

$$q(\mathbf{x}) = \beta(\gamma(\alpha(\mathbf{x}))). \tag{12}$$

The effect of channel errors on the LBG-VQ has been studied in [6]-[7]. In the next section, we study the channel-error effect on the TSVQ and MSVQ and develop a scheme to combat this undesirable effect.

III. Channel-Matched TSVQ and MSVQ

In this section, the performances of the TSVQ and MSVQ over noisy channels are analyzed and an algorithm is developed for designing TSVQ and MSVQ for a given noisy channel. The resulting schemes are thus called channel-matched TSVQ (CM-TSVQ) and channel-matched MSVQ (CM-MSVQ), respectively. We have avoided

using the term "channel-optimized" as in [7] since, by their definitions in [10, 11], the TSVQ and MSVQ are sub-optimal. The proposed algorithms do, however, result in local optimality at each stage of the design process.

Since TSVQ is the more general of the two, we will discuss it first. For now, let us consider only the two-stage case ($S = 2$). Extension to the general case ($S > 2$) will be made later.

A. Problem Statement

Consider the block diagram given in Figure 1. The first-stage (or primary) encoder, α_1, is described in terms of a partition $\mathcal{P}_1 = \{S_{1,0}, S_{1,1}, \ldots, S_{1,N_1-1}\}$ according to

$$\alpha_1(\mathbf{x}) = i \quad \text{if} \quad \mathbf{x} \in S_{1,i}, \quad i \in \mathcal{J}_{N_1}. \tag{13}$$

The output of this encoder is transmitted over a DMC described by the random mapping $\gamma_1 : \mathcal{J}_{N_1} \mapsto \mathcal{J}_{N_1}$ and the transition matrix:

$$Q_1(j|i) = \Pr\{\gamma_1(i) = j\}, \quad i, j \in \mathcal{J}_{N_1}. \tag{14}$$

The primary decoder, β_1, is given in terms of the reproduction codebook, $\mathcal{C}_1 = \{\mathbf{c}_{1,0}, \mathbf{c}_{1,1}, \ldots, \mathbf{c}_{1,N_1-1}\}$ according to

$$\beta_1(j) = \mathbf{c}_{1,j}, \quad j \in \mathcal{J}_{N_1}. \tag{15}$$

The output of this decoder is denoted by $\mathbf{z} = \mathbf{c}_{1,j}$.

Both the source vector, $\mathbf{x}$, and the output of the primary encoder, i, are inputs to the second-stage (or secondary) encoder. Note that this is quite different from the noiseless-channel case, where the input to the second-stage encoder is the coding error, $\mathbf{x} - \mathbf{z}$, of the first stage. Since the channel here is noisy, the value of j and hence $\mathbf{z} = \mathbf{c}_{1,j}$ is not known at the transmitter. Later on, we will see that, for the squared-error distortion measure, what the secondary encoder actually does is

encode the *expected* coding error between $\mathbf{x}$ and $\mathbf{Z}$. For now, let us just assume that the secondary encoder is the mapping $\alpha_2 : \mathbb{R}^k \times \mathcal{J}_{N_1} \mapsto \mathcal{J}_{N_2}$, described by

$$\alpha_2(\mathbf{x}, i) = u \quad \text{if } \mathbf{x} \in S_i^u, \quad i \in \mathcal{J}_{N_1}, u \in \mathcal{J}_{N_2}, \tag{16}$$

where $S_i^u \triangleq S_{1,i} \cap S_{2,u}$ and $\mathcal{P}_2 = \{S_{2,0}, S_{2,1}, \ldots, S_{2,N_2-1}\}$ is another partition of $\mathbb{R}^k$. The output of this encoder is transmitted over a second DMC described by $\gamma_2 : \mathcal{J}_{N_2} \mapsto \mathcal{J}_{N_2}$ and

$$Q_2(v|u) = \Pr\{\gamma_2(u) = v\}, \quad u, v \in \mathcal{J}_{N_2}. \tag{17}$$

Observe that, beside having two inputs to the secondary encoder, another difference between the formulation in Figure 1 and that of Section II is that the output of the primary and secondary encoders are transmitted over two different channels as opposed to being multiplexed together (as in (9)) and transmitted over a single channel. In this paper, we assume that this single channel is a model of a binary symmetric channel (BSC) which is used $\log_2(N_1 N_2)$ times for each block of k source samples. Each channel-output bit depends only on the corresponding input bit. Thus, the combination of the multiplexer, the BSC and the demultiplexer can be modeled by two *independent* channels – the first of which accepts $\log_2 N_1$ bits and the second $\log_2 N_2$ bits. Hence, for the BSC case, the single-channel formulation of Section II and the two-channel formulation of Figure 1 are exactly the same. If the channel is not a BSC, the two formulations are still equivalent – except that the two channels may be dependent.

The secondary decoder, β_2, depends on the output of both channels, j and v, and it is described by N_1 *different* codebooks, $\mathcal{C}_{2,j} = \{\mathbf{c}_{2,j,0}, \mathbf{c}_{2,j,1}, \ldots, \mathbf{c}_{2,j,N_2-1}\}$, $j \in \mathcal{J}_{N_1}$, according to the formula

$$\beta_2(j, v) = \mathbf{c}_{2,j,v}, \quad j \in \mathcal{J}_{N_1}, \ v \in \mathcal{J}_{N_2}. \tag{18}$$

Here, we shall use $\mathcal{C}_2$ to refer to the collection $\{\mathcal{C}_{2,0}, \mathcal{C}_{2,1}, \ldots, \mathcal{C}_{2,N_1-1}\}$. The output of β_2 is denoted by $\mathbf{w}$ and the reconstructed vector is $\tilde{\mathbf{x}} = \mathbf{z} + \mathbf{w}$.

The primary encoder and decoder are assumed to be given and are fixed. Our problem then is to minimize the average distortion $D = \frac{1}{k}E[d(\mathbf{X}, \tilde{\mathbf{X}})]$ by appropriate choices of $\mathcal{P}_2$ and $\mathcal{C}_2$.

B. Necessary Conditions

Assuming that the two channels are independent, the average distortion may be written as

$$D = \frac{1}{k}\sum_{i,u}\sum_{j,v} Q_1(j|i)Q_2(v|u)\int_{S_i^u} d(\mathbf{x}, \mathbf{c}_{1,j} + \mathbf{c}_{2,j,v})f(\mathbf{x})d\mathbf{x}, \tag{19}$$

where $f(\mathbf{x})$ is the k-fold probability density function (p.d.f) of the source.

Following the developments in [7] and [14], the second summation and the integral in (19) can be interchanged, yielding

$$D = \frac{1}{k}\sum_{i,u}\int_{S_i^u}\left\{\sum_{j,v} Q_1(j|i)Q_2(v|u)d(\mathbf{x}, \mathbf{c}_{1,j} + \mathbf{c}_{2,j,v})\right\} f(\mathbf{x})d\mathbf{x}. \tag{20}$$

The term in braces is defined as the *modified* distortion measure [7, 14],

$$d'(\mathbf{x}; i, u) \triangleq \sum_{j,v} Q_1(j|i)Q_2(v|u)d(\mathbf{x}, \mathbf{c}_{1,j} + \mathbf{c}_{2,j,v}), \tag{21}$$

which can be interpreted as the expected distortion between $\mathbf{x}$ and $\tilde{\mathbf{X}}$ given that i and u are transmitted. With this modified distortion measure, it is clear that for a fixed codebook, the optimum partition, $\mathcal{P}_2^* = \{S_{2,0}^*, S_{2,1}^*, \ldots, S_{2,N_2-1}^*\}$, must satisfy

$$S_{2,u}^* = \{\mathbf{x} : d'(\mathbf{x}; i, u) \le d'(\mathbf{x}; i, u') \text{ if } \mathbf{x} \in S_{1,i}\ \forall u' \in \mathcal{J}_{N_2}\}, \quad u \in \mathcal{J}_{N_2}. \tag{22}$$

Similarly, for a fixed partition, the optimum codebook, $\mathcal{C}_2^* = \{\mathcal{C}_{2,0}^*, \mathcal{C}_{2,1}^*, \ldots, \mathcal{C}_{2,N_1-1}^*\}$, where $\mathcal{C}_{2,j}^* = \{\mathbf{c}_{2,j,0}^*, \mathbf{c}_{2,j,1}^*, \ldots, \mathbf{c}_{2,j,N_2-1}^*\}$ for each j in $\mathcal{J}_{N_1}$, can be derived using estimation theoretic arguments:

$$\begin{aligned}\mathbf{c}_{2,j,v}^* &= \arg\min_{\mathbf{w}\in\mathbb{R}^k} E[d(\mathbf{X}, \check{\mathbf{X}})|j, v] \\ &= \arg\min_{\mathbf{w}\in\mathbb{R}^k} E[d(\mathbf{X}, \mathbf{Z} + \mathbf{w})|j, v], \quad j \in \mathcal{J}_{N_1}, v \in \mathcal{J}_{N_2}.\end{aligned} \tag{23}$$

In the case where $d(\cdot,\cdot)$ is the squared-error distortion, i.e.,

$$d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|^2, \tag{24}$$

the modified distortion measure of (21) can be simplified as follows. Let

$$\mathbf{y}_{1,i} \triangleq \sum_j Q_1(j|i)\mathbf{c}_{1,j}, \quad i \in \mathcal{J}_{N_1}, \tag{25}$$

$$\mathbf{y}_{2,i,u} \triangleq \sum_{j,v} Q_1(j|i)Q_2(v|u)\mathbf{c}_{2,j,v}, \quad i \in \mathcal{J}_{N_1},\ u \in \mathcal{J}_{N_2}, \tag{26}$$

and

$$\delta_{1,i} \triangleq \sum_j Q_1(j|i)\|\mathbf{c}_{1,j}\|^2, \quad i \in \mathcal{J}_{N_1}, \tag{27}$$

$$\delta_{2,i,u} \triangleq \sum_{j,v} Q_1(j|i)Q_2(v|u)\|\mathbf{c}_{2,j,v}\|^2, \quad i \in \mathcal{J}_{N_1},\ u \in \mathcal{J}_{N_2}. \tag{28}$$

Then the modified distortion measure can be expressed as

$$d'(\mathbf{x};i,u) = \|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \mathbf{y}_{1,i}+\mathbf{y}_{2,i,u}\rangle + \delta_{1,i} + 2\langle \mathbf{y}_{1,i}, \mathbf{y}_{2,i,u}\rangle + \delta_{2,i,u}, \tag{29}$$

or equivalently,

$$d'(\mathbf{x};i,u) = \|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \mathbf{y}_{1,i}\rangle + \delta_{1,i} + \delta_{2,i,u} - 2\langle \mathbf{x}-\mathbf{y}_{1,i}, \mathbf{y}_{2,i,u}\rangle. \tag{30}$$

The sum of the first three terms in (30) is defined as $d_1(\mathbf{x};i)$ which can be viewed as the expected distortion of the first stage in encoding $\mathbf{x}$ given that $\mathbf{x}$ is in $S_{1,i}$, where the expectation is taken over all possible channel transitions. The sum of the last two terms is defined as $d_2(\mathbf{x};i,u)$, where $-d_2(\mathbf{x};i,u)$ can be regarded as the amount of distortion reduction associated with the second stage given that $\mathbf{x}$ is in S_i^u.

The codebook search of the secondary encoder (equation (22)) involves the determination of the value of u which minimizes $d'(\mathbf{x};i,u)$ while keeping i fixed. Accordingly, it suffices to compute $d_2(\mathbf{x};i,u)$ for each u in $\mathcal{J}_{N_2}$. The distortion d_2 corresponds exactly to the distortion measure used in the channel-optimized VQ (COVQ) codebook search in [7], with $(\mathbf{x}-\mathbf{y}_{1,i})$ replacing $\mathbf{x}$. Hence, the secondary encoder acts just

like the COVQ encoder with $(\mathbf{x} - \mathbf{y}_{1,i})$ as its input vector. Note that $\mathbf{y}_{1,i}$, as defined by (25), is the expected value of $\mathbf{Z}$ (the output of the primary decoder) given that i is transmitted. Thus, $(\mathbf{x} - \mathbf{y}_{1,i})$, the vector which is encoded by the secondary encoder, is nothing but the *expected* coding error of the first stage.

Finally, for the squared-error distortion measure, it can be readily shown that the optimum codebook of (23) reduces to

$$\mathbf{c}^*_{2,j,v} = \frac{\sum_{i,u} Q_1(j|i)Q_2(v|u) \int_{S_i^u} (\mathbf{x} - \mathbf{c}_{1,j}) f(\mathbf{x}) d\mathbf{x}}{\sum_{i,u} Q_1(j|i)Q_2(v|u) \int_{S_i^u} f(\mathbf{x}) d\mathbf{x}}, \quad j \in \mathcal{J}_{N_1},\ v \in \mathcal{J}_{N_2}. \tag{31}$$

C. Design Algorithm

An algorithm for designing the channel-matched TSVQ (CM-TSVQ) for the squared-error distortion measure is presented next. This algorithm is a generalization of the iterative algorithms of [8] and [10]. A training sequence can be used for the design if each integral is replaced by the appropriate sum.

CM-TSVQ Design Algorithm:

(1) Given $\mathcal{P}_1$, $\mathcal{C}_1$ and $\epsilon_0 > 0$, set $m = 0$ and $D^{(m)} = \infty$ (m =iteration index). Choose initial $\mathcal{P}_2^{(m)}$ and $\mathcal{C}_2^{(m)}$.

(2) Compute $\mathbf{y}_{1,i}$ and $\delta_{1,i}$ $\forall i \in \mathcal{J}_{N_1}$ using the codebook $\mathcal{C}_1$ and equations (25) and (27).

(3) Set $m = m + 1$.

(4) Compute $\mathbf{y}_{2,i,u}$ and $\delta_{2,i,u}$ $\forall i \in \mathcal{J}_{N_1}, \forall u \in \mathcal{J}_{N_2}$ using the codebook $\mathcal{C}_2^{(m-1)}$ and equations (26) and (28).

(5) $\forall i \in \mathcal{J}_{N_1}$:

$\{\forall \mathbf{x} \in S_{1,i}$:

$\{$ Set $u^* = \arg \min_{u \in \mathcal{J}_{N_2}} d'(\mathbf{x}; i, u) = \arg \min_{u \in \mathcal{J}_{N_2}} d_2(\mathbf{x}; i, u)$.

Set $\mathbf{x} \in S^*_{2,u^*}$.$\}\}$

(6) Compute $D^{(m)}$ using $\mathcal{C}_2^{(m-1)}$, the optimum partition $\mathcal{P}_2^*$ obtained in step **(5)** and equation (20). Compute the optimum codebook $\mathcal{C}_2^*$ using $\mathcal{P}_2^*$ and equation (31). Set $\mathcal{C}_2^{(m)} = \mathcal{C}_2^*$ and $\mathcal{P}_2^{(m)} = \mathcal{P}_2^*$.

(7) If $(\frac{D^{(m-1)}-D^{(m)}}{D^{(m)}} > \epsilon_0)$ go to **(3)**.

(8) Stop with $\mathcal{P}_2^{(m)}$ and $\mathcal{C}_2^{(m)}$ as the final partition and codebook.

The choice of $\mathcal{P}_1$ and $\mathcal{C}_1$ is particularly important in the CM-TSVQ design. A natural candidate for these is the optimum partition and codebook associated with the COVQ [7] designed for γ_1 and rate R_1. We will see later that this choice is not necessarily the "best" one.

Designing CM-TSVQ for more than two stages is simple. Suppose that we are interested in designing a CM-TSVQ with three stages. This can be done by first designing one with only two stages and then viewing these two stages as a single VQ, i.e., as the first-stage VQ and then proceed as before. This procedure can be repeated until the desired number of stages is reached. We should note that this procedure does not guarantee that the overall encoder-decoder pair will be optimal - neither globally nor locally.

D. Channel-Matched MSVQ

As stated earlier, the above development is for the more general case of TSVQ. For the MSVQ, it suffices to add the constraint that all codebooks in the second stage are the same. This implies that the secondary decoder, β_2, depends only on the output of the second channel, v, and is described by a *single* codebook, $\mathcal{C}_2 = \{\mathbf{c}_{2,0}, \mathbf{c}_{2,1}, \ldots, \mathbf{c}_{2,N_2-1}\}$, according to the formula

$$\beta_2(v) = \mathbf{c}_{2,v}, \quad v \in \mathcal{J}_{N_2}. \tag{32}$$

In this case, the average distortion in equations (19) and (20) and the modified distortion measure in equation (21) are the same except that $\mathbf{c}_{2,j,v}$ is replaced by

$\mathbf{c}_{2,v}$. The optimum partition is the same as in (22), while the optimum codebook of (23) is replaced by

$$\mathbf{c}^*_{2,v} = \arg \min_{\mathbf{w}\in\mathbb{R}^k} E[d(\mathbf{X}, \mathbf{Z}+\mathbf{w})|v], \quad v \in \mathcal{J}_{N_2}. \tag{33}$$

For the squared-error distortion measure, $\mathbf{y}_{2,i,u}$ and $\delta_{2,i,u}$ in equations (26), (28), (29) and (30) are replaced by

$$\mathbf{y}_{2,u} \triangleq \sum_v Q_2(v|u)\mathbf{c}_{2,v}, \quad u \in \mathcal{J}_{N_2}, \tag{34}$$

and

$$\delta_{2,u} \triangleq \sum_v Q_2(v|u)\|\mathbf{c}_{2,v}\|^2, \quad u \in \mathcal{J}_{N_2}. \tag{35}$$

Finally, the simplified optimum codebook of equation (31) can be replaced by

$$\mathbf{c}^*_{2,v} = \frac{\sum_{i,u} Q_2(v|u) \int_{S^u_i} (\mathbf{x} - \mathbf{y}_{1,i}) f(\mathbf{x}) d\mathbf{x}}{\sum_{i,u} Q_2(v|u) \int_{S^u_i} f(\mathbf{x}) d\mathbf{x}}, \quad v \in \mathcal{J}_{N_2}. \tag{36}$$

The design algorithm for the CM-MSVQ is identical to that of the CM-TSVQ, so we will not repeat it here.

IV. Numerical Results

Performances of the CM-TSVQ and CM-MSVQ are presented next for the memoryless Gaussian source and the Gauss-Markov source with correlation coefficient $\rho = 0.9$. We consider the binary symmetric channel (BSC) with bit error rates (BERs) $\epsilon = 0.0, 0.005, 0.01, 0.05$ and 0.1. In all cases, we assume that $N_s = N_1$ for all s. Numerical results for the CM-TSVQ and CM-MSVQ at the overall rate of 1 bit/sample are presented in Tables 1-4 for dimensions $k = 2, 4, 6, 8$, and 10. These results are given in

$$\text{SNR} = 10 \log_{10} \frac{\sigma^2}{D}, \qquad \text{dB}, \tag{37}$$

where σ^2 is the source variance and D is the average per letter squared-error distortion. The results in these tables were obtained using 50,000 vectors outside of the training

sequence which itself consists of 50,000 vectors[2]. In the design, the first-stage VQs are chosen to be the COVQ [7] designed for the given BER and the method of extension (Section III.C) is used whenever there are more than two stages.

The procedure for choosing the initial codebook, $\mathcal{C}_2^{(0)}$, and partition, $\mathcal{P}_2^{(0)}$, in step **(1)** of the design algorithm (borrowed from [7]) is as follows. At each stage s, a codebook, designed for a noiseless channel, is obtained for the expected coding error of the first $(s-1)$ stages. For the noiseless channel codebook we have used a simulated annealing algorithm, described in [13], to assign binary indices to the codevectors. This codebook is used as the initial codebook in the design for the channel with BER $\epsilon = 0.005$. The final codebook for $\epsilon = 0.005$ is used as the initial choice for the $\epsilon = 0.01$ design, and so on until the desired BER is obtained.

Also included in these tables are the results of the ordinary TSVQ [10] and MSVQ [11], i.e., TSVQ and MSVQ designed for the noiseless channel. The cases where $S = 1$ correspond to the full-searched COVQ or LBG-VQ.

Several interesting observations and conclusions can be made from these tables: (i) When the channel is very noisy, i.e., $\epsilon = 0.05$ or 0.1, the channel-matched schemes always outperform the ordinary schemes. The largest performance gain in these cases is 4.90 dB (Table 3, $k = 10, S = 10, \epsilon = 0.1$). When the channel is less noisy ($\epsilon = 0.005$ or 0.01), there are a few exceptions in which the ordinary schemes outperform the channel-matched schemes. For example, in Table 4, for $k = 4, S = 4$ and $\epsilon = 0.005$, the SNR of the CM-MSVQ is 8.50 dB while that of the ordinary MSVQ is 8.53 dB. In these cases the differences in performance are always less than or equal to 0.03 dB. We feel that these exceptions are due to the sub-optimality of the design algorithm. (ii) Even though in a noiseless channel the ordinary MSVQ is inferior to the ordinary

[2]We have found that for the TSVQ (Tables 1 and 3), the training data size were too small for the $k = 8$ and 10 cases. Thus, in these two cases, we have re-designed the codebook using larger training data consisting of 500,000 vectors for $k = 8$ and 1,000,000 vectors for $k = 10$. In both cases, the results are given for 50,000 test vectors. The same method was used to obtain the full-searched ($N_1 = 256$ and 1024) results given in Tables 2 and 4.

TSVQ (for fixed R, k and S), it is much more robust to channel errors than the TSVQ. This can be explained by observing that a single bit error during transmission can cause a decoding error at only a single stage of the MSVQ, while all other stages are not affected [15]; in the case of TSVQ, a single error in stage s will cause decoding errors in all stages $r \geq s$. For fixed R and k, the robustness of the ordinary MSVQ increases as S increases. (iii) For the memoryless Gaussian source with fixed R and N_1, the performance of the channel-matched schemes does not always increase as the dimensionality increases. In fact, in many instances it decreases. On the contrary, for the Gauss-Markov source (with $\rho = 0.9$), the performance of the channel-matched schemes tends to increase monotonically with k (for fixed R and N_1). (iv) Surprisingly, for fixed R and k, the performance of the channel-matched schemes does not increase monotonically with N_1. In many cases, the results for $N_1 = 2$ are better than for $N_1 = 4$. This again can be attributed to the structure of the CM-TSVQ and CM-MSVQ and the sub-optimality of the design algorithm. Similar observations for the MSVQ with $\epsilon = 0.0$ have been reported in [16] for the memoryless Gaussian source. (v) Finally, an interesting result in Table 4 for $k = 2$ and $\epsilon = 0.1$ indicates that the two-stage CM-MSVQ performs better than the COVQ. This implies that the COVQ design for this case[3] resulted in a locally optimum solution rather than a global one.

A. Choice of BER for the First Stage

As mentioned earlier in Section III.C, the choice of the first-stage VQ is important in the overall performance of the TSVQ or MSVQ. We have hinted that using a COVQ designed for the given BER in the first stage does not necessarily lead to the best results. In Table 5, we present several results on two-stage CM-TSVQ and CM-MSVQ in which the first-stage VQ is designed for a BER (denoted by ϵ_1) which is different from the actual BER. The second stage is always designed for the actual BER since

[3]Essentially the same result was reported in [7] for the COVQ with $R = 1, k = 2$ and $\epsilon = 0.1$.

this leads to a locally optimum system (given the first stage). It can be seen from the results in Table 5 that choosing ϵ_1 slightly larger than the actual BER can lead to some performance improvements. These improvements are more noticeable for very noisy channels.

In a two-stage TSVQ or MSVQ, the codevectors in the first stage are related to the source vector $\mathbf{x}$ while the second-stage codevectors are related to the error vector of the first stage. Thus, one may argue that the first stage contains more useful information than the second stage – even if the same number of bits are allocated to each stage. That is, given the first-stage codevector by itself, we have more information than if we are given only the second-stage codevector. Hence, intuitively, one would like to have more protection (against channel noise) in the first stage than in the second stage. A way of achieving this goal is to design the first stage for a slightly larger BER than the actual channel, as evidenced by the results in Table 5.

In the design of a CM-TSVQ or CM-MSVQ, it is left up to the designer to make a judgement as to the choice of ϵ_1.

B. Larger Block Sizes for the CM-MSVQ

As noted earlier, one of the main advantages of the MSVQ is its low complexity – both in terms of computation and memory. Thus the MSVQ can operate at rates and/or block lengths larger than what is possible for VQ or TSVQ. Even though the TSVQ has low computational complexity, it still requires large memory. Furthermore, to design a TSVQ at large rates or block lengths requires a training sequence which can be prohibitively large. For the MSVQ, it suffices to have a training sequence which is large enough for the codebook of each stage.

In this subsection, we will examine the performance of the CM-MSVQ as the *block length* becomes large. Table 6 shows the performances at $k = 12, 18$ and 24 for the Gauss-Markov source. We consider only the case where $N_1 = 64$ and we have made

use of the findings of the previous subsection to choose the design BER of each stage. Our choices of the design BERs are provided in Table 7. We make no claim that these choices are optimum in any sense. However, we have tried several others and have found these to be among the best.

It is important to note that, for this source, the 12-dimensional CM-MSVQ is comparable to the 8-dimensional COVQ (Table 4; $N_1 = 256$) in SNR performance even though the CM-MSVQ has approximately half the computational complexity (128-vector codebook search compared to 256). We will have more discussion on the complexity issues in a later section.

C. Multiple Candidate Codebook Search (M-Algorithm)

In the previous section, we stated that the modified distortion measure of equations (21) and (30) can be expressed as

$$d'(\mathbf{x}; i, u) = d_1(\mathbf{x}; i) + d_2(\mathbf{x}; i, u), \tag{38}$$

where

$$d_1(\mathbf{x}; i) = \|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \mathbf{y}_{1,i} \rangle + \delta_{1,i}, \tag{39}$$

and

$$d_2(\mathbf{x}; i, u) = \begin{cases} \delta_{2,i,u} - 2\langle \mathbf{x} - \mathbf{y}_{1,i}, \mathbf{y}_{2,i,u} \rangle & \text{for the TSVQ,} \\ \delta_{2,u} - 2\langle \mathbf{x} - \mathbf{y}_{1,i}, \mathbf{y}_{2,u} \rangle & \text{for the MSVQ.} \end{cases} \tag{40}$$

The purpose of the primary encoder is to provide an index i to the secondary encoder (when the primary quantizer is a COVQ designed for the given BER, this value of i is that which minimizes $d_1(\mathbf{x}; i)$). For this i, the secondary encoder then chooses the u which minimizes $d'(\mathbf{x}; i, u)$, or equivalently, $d_2(\mathbf{x}; i, u)$. Note that this procedure does not necessarily lead to an optimum index pair (i, u), i.e., that which minimizes $d'(\mathbf{x}; i, u)$. This is a disadvantage associated with the successive approximation method.

For the two-stage MSVQ, if the encoder structure is removed while still maintaining the decoder structure, the resulting quantizer is equivalent to the *two-channel*

conjugate vector quantizer (2C-CVQ) proposed by Moriya [15]. The 2C-CVQ encoder chooses the pair (i, u) which minimizes $d'(\mathbf{x}; i, u)$ by exhaustively searching the two codebooks. A generalization of the 2C-CVQ to more than two stages, called *exhaustive search residual quantizer* (ESRQ), was proposed by Barnes and Frost [16] for the noiseless channel. Locally optimum design algorithms for the 2C-CVQ and ESRQ are known [15, 16].

Even though the 2C-CVQ and ESRQ are nice, they are not always desirable because of the large amount of computation required on the encoder[4]. In this subsection we introduce a scheme whose performance and complexity is in between that of the two-stage CM-MSVQ and 2C-CVQ. This scheme, referred to as the *multiple candidate* CM-MSVQ (or, the M-algorithm), operates as follows. Instead of passing the single index i, the primary encoder passes a set consisting of the top M candidates, $\mathcal{I} = (i_1, i_2, \ldots, i_M) \subset \mathcal{J}_{N_1}$, to the secondary encoder. The secondary encoder then chooses the pair (i_m, u) which minimizes $d'(\mathbf{x}; i_m, u)$, where $i_m \in \mathcal{I}$ and $u \in \mathcal{J}_{N_2}$. The multiple candidate CM-MSVQ is equivalent to the 2C-CVQ when $M = N_1$. Using the M-algorithm, the computational complexity of the secondary encoder increases by a factor approximately equal to M. The increase in memory is negligible.

The performances of the multiple candidate CM-MSVQ are tabulated in Table 8 for the cases $k = 6, S = 2$ and $k = 10, S = 2$ for the Gauss-Markov source with $M = 1, 2, 4$ and 8. The choices of the design BER are given in Table 7. We have obtained two sets of results. In the first set, the codebooks are designed using the same algorithm described earlier. In the second set (indicated by the * in Table 8), the primary codebook is kept the same, but the secondary codebook was re-optimized[5]

[4]For the 2C-CVQ, Moriya has proposed a method for reducing the computational complexity while increasing the memory requirement of the encoder. He does this by storing a matrix of inner products of every pair of codevectors in the two codebooks [15].

[5]The algorithm for re-optimization is the same as before except that in step **(5)** u^* is replaced by $u^* = \arg \min_{u \in \mathcal{J}_{N_2}} \{\min_{i_m \in \mathcal{I}} d'(\mathbf{x}; i_m, u)\}$.

accordingly.

First, note that in both sets the performance increases monotonically with M. Second, unlike the findings in Section IV.A, the most noticeable gains are found for the relatively clean channels. In this case, the biggest gain was 0.31 dB. Third, we can see that re-optimizing the secondary codebook does provide some additional gain. Finally, looking at the results for the $M = 4^*$ case, we can see that by judiciously choosing the design BER of the first stage and by using the M-algorithm we have been able to narrow the gap between the two-stage CM-MSVQ and the COVQ (Table 4) by approximately 40-60%. For the $k = 10, S = 2$ and $M = 4^*$ case, with the above modifications, the encoder computational complexity increases by 250% and the encoder memory increases by 50%. However, with these modifications, the encoder computational complexity and memory are still six and ten times less than that of the COVQ, respectively.

We have also obtained the results of the multiple candidate CM-MSVQ when the block length is large ($k = 12, 18$ and 24). These results are tabulated in Table 9. Here we have not re-optimized the codebooks. The choices of design BERs are the same as in Table 7. In the case where there are more than two stages, at each stage $s > 1$, the set of the top M candidates, $\mathcal{I} = \{\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_M\}$, are retained from the previous $(s-1)$ stages, where $\mathbf{i}_m = \{i_{m,1}, i_{m,2}, \ldots, i_{m,s-1}\}$ is the path map of the m-th best candidate.

V. Tandem Source-Channel Coding

In this section, we make comparisons between the CM-MSVQ scheme (in which, in a sense, the source and channel coding operations are implicitly combined), and a traditional tandem source-channel coding scheme with the same effective block length (hence, encoding delay).

The block diagram of the tandem source-channel code is depicted in Figure 2.

Here, we assume that the source code is an LBG-VQ with block length k_s and rate $R_s = m/k_s$ bits/sample. The source encoder thus produces m bits for every k_s source samples. The channel code is assumed to be a (q, lm) *linear block* code. Hence, the channel encoder accepts lm information bits and produces q bits for transmission ($q \geq lm$). That is, the channel encoder takes l successive m-bit codewords generated by the source encoder and produces a q-bit codeword. The overall rate of the tandem scheme is $R = q/(lk_s)$ transmitted bits per source sample and the effective block length (or delay) is $k = lk_s$ source samples. The reason we have considered the LBG-VQ and the linear block code is because they are both among the most well-known and conceptually simple block-structured source and channel codes, respectively.

To be in harmony with the more conventional notation used in the channel coding literature, from now on we will use (n', k') to denote (q, lm). The channel is assumed to be a BSC. For an (n', k') linear block code operating over a BSC with BER ϵ, the probability of correct decoding is upper bounded by

$$P_c = \sum_{i=0}^{t} \binom{n'}{i} \epsilon^i (1-\epsilon)^{n'-i} + A_t \epsilon^{t+1} (1-\epsilon)^{n'-t-1}, \tag{41}$$

where t is the greatest integer such that

$$A_t \triangleq 2^{n'-k'} - \sum_{i=0}^{t} \binom{n'}{i} \geq 0. \tag{42}$$

The above is known as the sphere-packing bound and is achievable if and only if the (n', k') code is quasi-perfect [17].

To analyze the performance of the tandem scheme, we first assume that the channel code is quasi-perfect. Secondly, we model the channel encoder, the BSC and the channel decoder as an *equivalent BSC* (represented by the dashed box in Figure 2). The assumption that the equivalent channel is a BSC is obviously incorrect. However, this assumption leads to a very simple analysis of the tandem scheme. The only problem is to specify the BER of the equivalent BSC. Here, we assume that, in the

equivalent channel, the probability that a block of k' consecutive bits are received without error is equal to P_c. Thus the BER of this channel can be computed as

$$\epsilon' = 1 - P_c^{1/k'}. \tag{43}$$

With this equivalent model, the analysis for the tandem source-channel coding scheme reduces to that of the LBG-VQ operating over a BSC with BER ϵ' [6, 7].

Using the equivalent model, we have evaluated the performance of the tandem scheme for $k_s = 6$ and $l = 2, 3$ and 4. These results (for the Gauss-Markov source with $\rho = 0.9$) are plotted in Figures 3,4 and 5, respectively. For the LBG-VQ of the tandem scheme, a simulated annealing algorithm [13] was used for index assignment. In each plot, we have chosen $m = 3, 4, 5$ and 6 (corresponding to $R_s = 0.5, 0.67, 0.83$ and 1.0). In all cases, we have chosen $q = lk_s$ so that the overall rate is always 1 bit/sample. We have also plotted on these graphs the optimum performance theoretically attainable (OPTA) obtained by equating the rate-distortion function to the channel capacity [18] and the performances of the CM-MSVQ ($M = 1$) and the multiple candidate CM-MSVQ ($M = 8$) with the same k as that of the tandem scheme (thus, similar encoding delays). The CM-MSVQ results are taken directly from Table 9. In these graphs, the performance of the multiple candidate CM-MSVQ with $M = 8$ is always better than the tandem scheme. Looking at the results of the tandem scheme, it is clear that when the channel is relatively noise-free, all of the available bits should be allocated to source coding. As the channel becomes noisier, more and more bits should be allocated to channel coding (in the form of redundancy). In any case, even with an optimum allocation of bits between the source and channel codes, the tandem scheme is always inferior to the CM-MSVQ for the cases considered.

Note that the CM-MSVQ outperforms the tandem scheme even when $\epsilon = 0.00$. The reason for this is that the source code of the tandem scheme has a block length of $k_s = 6$ only, while the CM-MSVQ operates on blocks of lengths $k = lk_s = 12, 18$ or

24. The argument here is that the CM-MSVQ is a low-complexity scheme which can operate at larger block sizes than the LBG-VQ. Since the source has memory, the CM-MSVQ with large block lengths beats the LBG-VQ with a small block length ($k_s = 6$). An important point to make is that the CM-MSVQ is a suboptimal combined source-channel coding scheme; whereas the tandem scheme consists of a source code which, by itself, is optimum for its block length and a channel code which, by itself, is optimum (according to the equivalent model) for its block length. For this reason, we believe that, for the source and block sizes considered, the combined source-channel coding approach may be more desirable than the traditional tandem approach.

Before closing this section, we should make a note that our analysis of the tandem scheme was somewhat optimistic. First of all, a quasi-perfect code does not exist for all values of (n', k') [17]. In fact, for all the cases which we have considered, only when (n', k')=(24,12), which corresponds to $R_s = 0.5$ in Figure 5, is there a known quasi-perfect code – the extended Golay code [17]. In all other cases, the probability of correct decoding is smaller than P_c [17]. Secondly, even if a quasi-perfect code does exist for (n', k'), there is still the question about the validity of the equivalent BSC model. It is clear that the equivalent model is incorrect; however, it is unclear how the inaccuracy in the model affects the performance analysis. We conjecture that our analysis of the tandem scheme over-estimates its actual performance.

To support this conjecture, we have simulated the tandem scheme with $R_s = 0.5$ using the (24,12) extended Golay code for 300,000 test samples of the Gauss-Markov source. The simulation was performed twenty times using twenty different realizations of a noise process. In Figure 6, the simulation results are compared with the analytical results using the equivalent BSC model. For the simulated results, the average SNRs are shown; the error bars indicate the minimum and maximum SNRs in the twenty trials. From this figure, we can see that the two curves coincide when the channel is relatively clean, but they differ dramatically when the channel is noisy.

At $\log_{10}\epsilon = -1$, they differ by more than 3 dB.

Since we do not know the optimum channel code for other values of R_s, we have not simulated those cases. However, to make fair comparisons, we have simulated the multiple candidate CM-MSVQ and the results are also plotted in Figure 6. For the CM-MSVQ, the simulated results coincide almost exactly with the analytical results, as expected.

In summary, the equivalent model for the tandem source-channel coding scheme is overly optimistic in two senses. First, it assumes that an (n', k') quasi-perfect code exists and secondly it assumes that the actual equivalent channel is a BSC. In light of this discussion, we have every reason to believe that the actual gap between the CM-MSVQ and the tandem scheme is even larger than what is suggested by Figures 3-5.

VI. Channel Mismatch

Up to this point, we have been working under the assumption that the channel and hence the channel BER is given. In real-life situations, the channel BER is not known *a priori* to the designer or, in some situations, it is time-varying. In this section, we will examine the performance of CM-TSVQ and CM-MSVQ under mismatch, i.e., when they are designed for a BER ϵ_d, but applied to a channel whose BER is actually ϵ_a.

The mismatch results are plotted in Figures 7 and 8 for $k = 10$ and $S = 2$. In these figures, the design BER is the same for both the first and second stages. It can be seen that the CM-TSVQ and CM-MSVQ are fairly robust to channel mismatch, though the CM-MSVQ exhibits more robustness than the CM-TSVQ. An interesting phenomenon can be observed in these figures. For the CM-TSVQ results in Figure 7, at $\epsilon_a = 0.005$ ($\log_{10}\epsilon_a = -2.3$), the performance with $\epsilon_d = 0.010$ is better than that with $\epsilon_d = 0.005$. The same thing occurs for the CM-MSVQ in Figure 8 at

$\epsilon_a = 0.005, 0.010$ and 0.050. This can be explained by the fact that we have not chosen the design BER for the first stage correctly. In fact, after carefully examining the numbers, we found that if we had chosen it correctly (according to Table 7) then the best result would have occurred when $\epsilon_d = \epsilon_a$.

Next, we evaluate the performance of the CM-MSVQ (with two stages) when the first stage is designed for $\epsilon_{d,1}$ and the second stage for $\epsilon_d = \epsilon_{d,2}$. These results are given in Figure 9 for $k = 10$ and $S = 2$. As expected, when $\epsilon_{d,1}$ is slightly larger than $\epsilon_{d,2}$ the performance improves at $\epsilon_a = \epsilon_d$ but degrades at $\epsilon_a = 0.0$. We have also studied the performance of the M-algorithm under channel mismatch conditions and these results are plotted in Figure 10 again for $k = 10$ and $S = 2$. From this figure, we may conclude that the M-algorithm does not degrade the performance of the CM-MSVQ under mismatch conditions. In fact, in most cases the performance improves.

Finally, we have examined the performance of the CM-MSVQ scheme at a large block size ($k = 24$) and compared this with the tandem scheme. Figure 11 shows the results for the CM-MSVQ matched to $\epsilon_d = 0.050$ but tested for various values of ϵ_a. Here, the first stage of the CM-MSVQ was designed for a BER of 0.200, the second stage for 0.100 and the third and fourth stages for 0.050 (see Table 7). It is clear from this figure that the CM-MSVQ is a robust scheme. For a wide range of BERs ($\epsilon_a \gtrsim 0.0016$), this single CM-MSVQ scheme outperforms all tandem schemes considered. For the other BERs ($\epsilon_a \lesssim 0.0016$), its performance is only slightly inferior to the tandem scheme with $R_s = 1.0$. Furthermore, the reader should keep in mind the earlier discussion that the analysis of the tandem scheme over-estimates its actual performance. Considering this fact, the significance of the CM-MSVQ scheme becomes more evident.

VII. Complexity Issues

Let us now analyze the complexity of the CM-MSVQ and CM-TSVQ and compare these with the full-searched COVQ. We will consider the two-stage CM-MSVQ and CM-TSVQ. Let us assume that the first stage is a COVQ whose design BER is different from that of the second stage. Hence, the first-stage encoder requires kN_1 words for storing $\{\mathbf{y}_{1,i}^{(1)} : i \in \mathcal{J}_{N_1}\}$ and N_1 words for storing $\{\delta_{1,i}^{(1)} : i \in \mathcal{J}_{N_1}\}$, where $\{\delta_{1,i}^{(1)}, \mathbf{y}_{1,i}^{(1)} : i \in \mathcal{J}_{N_1}\}$ are computed according to equations (25) and (27) with ϵ_1 used in the channel transition matrix. It also requires N_1 floating-point operations (FLOPs) per source sample for encoding. The secondary codebook requires $(k+1)N_1$ words to store $\{\delta_{1,i}^{(2)}, \mathbf{y}_{1,i}^{(2)} : i \in \mathcal{J}_{N_1}\}$ (these are different from those in the first stage since they are computed using ϵ_2). In addition, it requires $(k+1)N_1N_2$ words to store $\{\delta_{2,i,u}^{(2)}, \mathbf{y}_{2,i,u}^{(2)} : i \in \mathcal{J}_{N_1}, u \in \mathcal{J}_{N_2}\}$ for the CM-TSVQ or $(k+1)N_2$ words to store $\{\delta_{2,u}^{(2)}, \mathbf{y}_{2,u}^{(2)} : u \in \mathcal{J}_{N_2}\}$ for the CM-MSVQ. The computational complexity of the second stage is N_2 FLOPs per sample. For the decoder, the CM-MSVQ requires $k(N_1+N_2)$ words for storing $\{\mathbf{c}_{1,i} : i \in \mathcal{J}_{N_1}\}$ and $\{\mathbf{c}_{2,u} : u \in \mathcal{J}_{N_2}\}$ while the CM-TSVQ requires kN_1N_2 words for storing $\{\hat{\mathbf{c}}_{i,u} = \mathbf{c}_{1,i} + \mathbf{c}_{2,i,u} : i \in \mathcal{J}_{N_1}, u \in \mathcal{J}_{N_2}\}$. Table 10 summarizes the overall complexity of the two-stage CM-MSVQ and CM-TSVQ. We have also included the complexity of the COVQ with $N = N_1N_2$ codevectors. The extension of this analysis to more than two stages is obvious. It should be noted that the encoder memory and computational complexity in Table 10 are upper bounds on the actual complexity. It has been stated in [7] that for the COVQ when ϵ_d is large the number of non-empty encoding regions can be significantly less than the number of codevectors. The same phenomenon occurs for CM-MSVQ and CM-TSVQ. In such cases, the encoder memory and computational requirement can be reduced accordingly. Furthermore, if the design BERs are the same for the first and second stages, then the encoder memory can be reduced as well. Note that the decoder memory and the encoder computational requirement of the CM-TSVQ and

CM-MSVQ are the same as those of the ordinary TSVQ and MSVQ. The encoder memory of the channel matched schemes is slightly larger than the ordinary schemes.

Finally, examining the complexity of the tandem scheme, we see that the comparisons in Figures 3-6 are not entirely fair. In some cases, the CM-MSVQ has significantly higher complexity than the tandem scheme. To make a fair comparison, we plot in Figure 12 the performances of a CM-MSVQ and a tandem scheme whose complexities are approximately equal (see Table 11). For the CM-MSVQ, both stages were designed for the same BER, $\epsilon_d = 0.050$. Also, the complexity of the channel encoder/decoder of the tandem scheme is assumed to be negligibly small and is not included in Table 11. Note that the CM-MSVQ beats the tandem scheme in most cases even though its delay is less than half that of the tandem scheme.

VIII. Summary and Conclusions

We have presented a unified approach for the design of TSVQ and MSVQ in the presence of channel noise. Numerical results indicate that this combined source-channel coding scheme yields substantial improvements over the ordinary TSVQ and MSVQ when the channel is very noisy. For the two-stage TSVQ and MSVQ, we have shown that designing the first stage for a channel noisier than the actual channel, in general, leads to performance gains. We have also demonstrated that additional gains can be obtained using the M-algorithm. Our comparisons of the CM-MSVQ scheme with a more traditional tandem source-channel coding scheme (utilizing the LBG-VQ as the source code and a quasi-perfect block code as the channel code) have indicated superior performance of the CM-MSVQ in all cases considered. It is also shown that CM-MSVQ is robust against channel mismatch. These findings, while important from a theoretical point of view, may pave the way for the development of practical coding schemes for highly noisy channels such as those encountered in mobile radio situations.

References

[1] C. E. Shannon, "A Mathematical Theory of Communication," *Bell Syst. Tech. J.,* Vol. 27, pp. 379-423 and 623-656, 1948.

[2] C. E. Shannon, "Coding Theorems for a Discrete Source with a Fidelity Criterion," *IRE Nat. Conv. Rec.,* pp. 142-163, Mar. 1959.

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

[4] A. Kurtenbach and P. Wintz, "Quantizing for Noisy Channels," *IEEE Trans. Commun. Technology,* Vol. 17, pp. 291-302, Apr. 1969.

[5] N. Farvardin and V. Vaishampayan, "Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding," *IEEE Trans. Inform. Theory,* Vol. 33, No. 6, pp. 827-838, Nov. 1987.

[6] H. Kumazawa, M. Kasahara and T. Namekawa, "A Construction of Vector Quantizers for Noisy Channels," *Electronics and Engineering in Japan,* Vol. 67-B, No. 4, pp. 39-47, 1984.

[7] N. Farvardin and V. Vaishampayan, "On the Performance and Complexity of Channel-Optimized Vector Quantizers," *IEEE Trans. Inform. Theory,* Vol. 37, No. 1, pp. 155-160, Jan. 1991.

[8] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantization Design," *IEEE Trans. Commun.*, Vol. 28, pp. 84-95, Dec. 1980.

[9] R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, Vol. 1, pp. 4-29, Apr. 1984.

[10] R. M. Gray and Y. Linde, "Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources," *IEEE Trans. Commun.*, Vol. COM-30, pp. 381-389, Feb. 1982.

[11] B. H. Juang and A. H. Gray, Jr., "Multiple Stage Vector Quantization for Speech Coding," *Proc. ICASSP-82*, pp. 597-600, April 1982.

[12] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Massachusetts: Kluwer Academic Press, 1991.

[13] N. Farvardin, "A Study of Vector Quantization for Noisy Channels," *IEEE Trans. Inform. Theory,* Vol. 36, No. 4, pp. 799-809, July 1990.

[14] E. Ayanoglu and R. M. Gray, "The Design of Joint Source and Channel Trellis Waveform Coders," *IEEE Trans. Inform. Theory,* Vol. 33, pp. 855-865, Nov. 1987.

[15] T. Moriya, "Two-Channel Conjugate Vector Quantizer for Noisy Channel Speech Coding," *IEEE Trans. Commun.*, accepted for publication, Sept. 1990.

[16] C. F. Barnes and R. L. Frost, "On the Design and Performance of Residual Vector Quantization," submitted to *Advances in Electronics and Electron Physics,* 1991.

[17] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes.* Cambridge, MA: The MIT Press, 1972.

[18] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

			CM-TSVQ					Ordinary TSVQ				
			BER					BER				
k	N_1	S	0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
2	2	2	4.35	4.21	4.07	3.11	2.23	4.35	4.21	4.06	3.06	2.06
	4	1	4.36	4.21	4.07	3.11	2.23	4.36	4.19	4.05	3.04	2.04
4	2	4	4.37	4.11	3.89	2.80	1.89	4.37	4.11	3.87	2.35	1.12
	4	2	4.33	4.09	3.87	2.86	2.19	4.33	4.06	3.81	2.32	1.09
	16	1	4.63	4.41	4.21	3.14	2.26	4.63	4.40	4.19	2.81	1.61
6	2	6	4.38	4.06	3.85	2.80	2.27	4.38	4.06	3.77	2.06	0.78
	4	3	4.29	3.98	3.75	2.88	2.18	4.29	4.00	3.74	2.14	0.91
	8	2	4.49	4.15	3.89	2.92	2.18	4.49	4.15	3.84	2.06	0.75
	64	1	4.77	4.50	4.27	3.15	2.27	4.77	4.47	4.20	2.55	1.25
8	2	8	4.39	3.99	3.81	2.84	2.27	4.39	3.99	3.63	1.68	0.36
	4	4	4.36	3.98	3.71	2.92	2.13	4.36	3.97	3.61	1.70	0.39
	16	2	4.56	4.14	3.85	2.87	2.14	4.56	4.11	3.72	1.64	0.25
	256	1	4.88	4.52	4.27	3.15	2.27	4.88	4.47	4.10	2.10	0.70
10	2	10	4.16	3.75	3.67	2.78	2.26	4.16	3.71	3.32	1.27	-0.03
	4	5	4.12	3.89	3.56	2.83	2.13	4.12	3.70	3.32	1.34	0.03
	32	2	4.64	4.14	3.75	2.80	2.06	4.64	4.11	3.65	1.37	-0.04
	1024	1	4.97	4.55	4.27	3.14	2.26	4.97	4.30	3.74	1.12	-0.35

Table 1: SNR (in dB) Performance Results of the CM-TSVQ and the Ordinary TSVQ for the Memoryless Gaussian Source; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

			CM-MSVQ					Ordinary MSVQ				
			BER					BER				
k	N_1	S	0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
2	2	2	4.34	4.20	4.05	3.10	2.23	4.34	4.21	4.06	3.06	2.06
	4	1	4.36	4.21	4.07	3.11	2.23	4.36	4.19	4.05	3.04	2.04
4	2	4	4.38	4.23	4.08	3.13	2.26	4.38	4.23	4.09	3.08	2.08
	4	2	4.09	3.92	3.78	2.90	2.18	4.09	3.92	3.77	2.69	1.67
	16	1	4.63	4.41	4.21	3.14	2.26	4.63	4.40	4.19	2.81	1.61
6	2	6	4.39	4.24	4.08	3.14	2.27	4.39	4.24	4.09	3.09	2.08
	4	3	4.02	3.86	3.69	2.81	2.16	4.02	3.85	3.69	2.58	1.55
	8	2	4.33	4.13	3.94	2.84	2.12	4.33	4.11	3.90	2.57	1.40
	64	1	4.77	4.50	4.27	3.15	2.27	4.77	4.47	4.20	2.55	1.25
8	2	8	4.39	4.24	4.10	3.14	2.27	4.39	4.24	4.09	3.08	2.08
	4	4	4.10	3.93	3.77	2.85	2.16	4.10	3.94	3.78	2.69	1.66
	16	2	4.41	4.12	3.88	2.80	2.10	4.41	4.13	3.86	2.24	0.96
	256	1	4.88	4.52	4.27	3.15	2.27	4.88	4.47	4.10	2.10	0.70
10	2	10	4.38	4.24	4.08	3.14	2.26	4.38	4.23	4.09	3.08	2.08
	4	5	4.01	3.87	3.71	2.80	2.12	4.01	3.85	3.69	2.60	1.57
	32	2	4.45	4.16	3.91	2.79	2.10	4.45	4.14	3.86	2.17	0.87
	1024	1	4.97	4.55	4.27	3.14	2.26	4.97	4.30	3.74	1.12	-0.35

Table 2: SNR (in dB) Performance Results of the CM-MSVQ and the Ordinary MSVQ for the Memoryless Gaussian Source; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

			CM-TSVQ					Ordinary TSVQ				
			BER					BER				
k	N_1	S	0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
2	2	2	7.84	7.29	6.81	4.36	2.77	7.84	7.29	6.70	4.14	2.21
	4	1	7.83	7.29	6.81	4.36	2.77	7.83	7.28	6.80	4.14	2.20
4	2	4	9.98	9.03	8.24	5.63	4.49	9.98	8.89	8.02	4.15	1.78
	4	2	10.16	9.10	8.29	6.08	4.61	10.16	9.06	8.19	4.30	1.93
	16	1	10.18	9.18	8.45	6.24	4.64	10.18	9.11	8.25	4.43	2.08
6	2	6	10.67	9.43	8.57	6.71	4.90	10.67	9.32	8.30	4.06	1.61
	4	3	10.80	9.53	8.50	6.85	5.06	10.80	9.43	8.39	4.14	1.70
	8	2	10.89	9.57	8.94	6.72	5.17	10.89	9.48	8.42	4.13	1.72
	64	1	11.00	9.97	9.30	6.98	5.32	11.00	9.74	8.77	4.65	2.24
8	2	8	10.96	9.63	9.14	7.11	5.66	10.96	9.37	8.22	3.72	1.25
	4	4	11.14	9.65	9.06	7.12	5.61	11.14	9.54	8.38	3.90	1.45
	16	2	11.29	9.90	9.37	7.30	5.74	11.29	9.70	8.55	4.08	1.66
	256	1	11.39	10.31	9.74	7.51	5.83	11.39	9.95	8.87	4.55	2.12
10	2	10	11.14	9.54	9.32	7.27	5.89	11.14	9.34	8.08	3.43	0.99
	4	5	11.31	9.69	9.22	7.32	5.89	11.31	9.52	8.28	3.65	1.22
	32	2	11.51	9.86	9.56	7.53	5.98	11.51	9.75	8.51	3.91	1.48
	1024	1	11.64	10.35	9.94	7.79	6.10	11.64	9.80	8.53	3.87	1.46

Table 3: SNR (in dB) Performance Results of the CM-TSVQ and the Ordinary TSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

			CM-MSVQ					Ordinary MSVQ				
			BER					BER				
k	N_1	S	0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
2	2	2	7.47	7.05	6.67	4.64	3.24	7.47	7.07	6.70	4.51	2.78
	4	1	7.83	7.29	6.81	4.36	2.77	7.83	7.28	6.80	4.14	2.20
4	2	4	9.14	8.50	7.96	5.37	3.82	9.14	8.53	7.99	5.14	3.11
	4	2	9.71	8.84	8.14	5.11	3.52	9.71	8.85	8.14	4.70	2.48
	16	1	10.18	9.18	8.45	6.24	4.64	10.18	9.11	8.25	4.43	2.08
6	2	6	9.50	8.83	8.27	5.67	4.12	9.50	8.83	8.25	5.25	3.16
	4	3	10.14	9.19	8.44	5.47	3.89	10.14	9.20	8.42	4.82	2.54
	8	2	10.48	9.40	8.61	6.18	4.51	10.48	9.40	8.54	4.71	2.37
	64	1	11.00	9.97	9.30	6.98	5.32	11.00	9.74	8.77	4.65	2.24
8	2	8	10.00	9.23	8.61	5.86	4.26	10.00	9.24	8.60	5.40	3.23
	4	4	10.37	9.38	8.64	5.68	4.09	10.37	9.39	8.59	4.92	2.61
	16	2	11.00	9.88	9.08	6.31	5.11	11.00	9.78	8.83	4.77	2.36
	256	1	11.39	10.31	9.74	7.51	5.83	11.39	9.95	8.87	4.55	2.12
10	2	10	10.10	9.31	8.72	5.95	4.35	10.10	9.33	8.68	5.43	3.24
	4	5	10.55	9.56	8.75	5.76	4.21	10.55	9.52	8.69	4.93	2.61
	32	2	11.22	10.05	9.31	6.96	5.34	11.22	9.88	8.87	4.67	2.25
	1024	1	11.64	10.35	9.94	7.79	6.10	11.64	9.80	8.53	3.87	1.46

Table 4: SNR (in dB) Performance Results of the CM-MSVQ and the Ordinary MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

	CM-TSVQ					CM-MSVQ				
	Actual BER					Actual BER				
ϵ_1	0.00	0.005	0.01	0.05	0.10	0.00	0.005	0.01	0.05	0.10
$k = 6; N_1 = 8; S = 2$										
0.000	10.89	9.55	8.92	6.68	5.14	10.48	9.41	8.56	4.84	2.61
0.005	10.91	9.57	8.89	6.69	5.17	10.43	9.40	8.58	4.96	2.76
0.010	10.90	9.59	8.94	6.74	5.13	10.39	9.41	8.61	5.07	2.88
0.050	10.63	9.55	9.01	6.72	5.04	9.95	9.35	8.84	6.18	4.27
0.100	10.34	9.59	9.05	6.90	5.17	9.79	9.23	8.75	6.32	4.51
0.200	9.67	9.11	8.77	6.83	5.21	8.88	8.50	8.17	6.39	4.91

Table 5: SNR (in dB) Performance Results of the CM-TSVQ and CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Different Values of ϵ_1; ϵ_1 = Design BER of the First Stage; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

			CM-MSVQ				
			Actual BER				
k	N_1	S	0.00	0.005	0.01	0.05	0.10
12	64	2	11.39	10.21	9.71	7.41	5.85
18	64	3	11.46	10.37	9.69	7.35	5.78
24	64	4	11.47	10.40	9.80	7.36	5.81

Table 6: SNR (in dB) Performance Results of the CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Large Values of k; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

			Actual BER				
k	S	ϵ_s	0.00	0.005	0.01	0.05	0.10
6	2	ϵ_1	0.000	0.010	0.050	0.200	0.200
		ϵ_2	0.000	0.005	0.010	0.050	0.100
10	2	ϵ_1	0.000	0.010	0.050	0.200	0.200
		ϵ_2	0.000	0.005	0.010	0.050	0.100
12	2	ϵ_1	0.000	0.010	0.050	0.200	0.200
		ϵ_2	0.000	0.005	0.010	0.050	0.100
18	3	ϵ_1	0.000	0.050	0.100	0.200	0.300
		ϵ_2	0.000	0.010	0.050	0.100	0.200
		ϵ_3	0.000	0.005	0.010	0.050	0.100
24	4	ϵ_1	0.000	0.050	0.100	0.200	0.300
		ϵ_2	0.000	0.010	0.050	0.100	0.200
		ϵ_3	0.000	0.005	0.010	0.050	0.100
		ϵ_4	0.000	0.005	0.010	0.050	0.100

Table 7: Choices of Design BER for the Results in Tables 6 and 8; k = Dimension; S = Number of stages; ϵ_s = Design BER for Stage s.

	Multiple Candidate CM-MSVQ				
	Actual BER				
M	0.00	0.005	0.01	0.05	0.10
$k = 6; N_1 = 8; S = 2$					
1	10.48	9.41	8.84	6.39	4.91
2	10.58	9.48	8.94	6.48	4.93
4	10.59	9.49	8.97	6.49	4.94
8	10.79	9.49	8.98	6.50	4.94
2*	10.69	9.58	8.93	6.49	4.94
4*	10.82	9.66	9.07	6.51	4.94
8*	10.82	9.66	9.07	6.52	4.95
$k = 10; N_1 = 32; S = 2$					
1	11.22	10.11	9.56	7.27	5.64
2	11.35	10.21	9.67	7.35	5.67
4	11.38	10.25	9.70	7.37	5.67
8	11.38	10.26	9.71	7.38	5.67
2*	11.38	10.22	9.69	7.35	5.67
4*	11.44	10.28	9.73	7.39	5.69
8*	11.46	10.30	9.75	7.39	5.69

Table 8: SNR (in dB) Performance Results of the Multiple Candidate CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Different Values of M; M = Number of Candidates; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages; * = Re-optimized Secondary Codebook.

	Multiple Candidate CM-MSVQ				
	Actual BER				
M	0.00	0.005	0.01	0.05	0.10
$k = 12; N_1 = 64; S = 2$					
1	11.39	10.21	9.71	7.41	5.85
2	11.53	10.32	9.83	7.51	5.88
4	11.58	10.36	9.87	7.54	5.89
8	11.59	10.37	9.88	7.55	5.89
$k = 18; N_1 = 64; S = 3$					
1	11.46	10.37	9.69	7.35	5.78
2	11.68	10.56	9.90	7.49	5.87
4	11.80	10.66	10.01	7.55	5.91
8	11.84	10.70	10.06	7.57	5.92
$k = 24; N_1 = 64; S = 4$					
1	11.47	10.40	9.80	7.36	5.81
2	11.74	10.61	10.01	7.49	5.91
4	11.90	10.73	10.13	7.55	5.96
8	11.97	10.79	10.18	7.58	5.98

Table 9: SNR (in dB) Performance Results of the Multiple Candidate CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 for Large Values of k and Different Values of M; M = Number of Candidates; Overall Rate = 1 Bit/Sample; k = Dimension; N_1 = Number of Codevectors at Each Stage; S = Number of Stages.

	CM-MSVQ	CM-TSVQ	COVQ
Encoder Memory (32-bit words)	$(k+1)(2N_1 + N_2)$	$(k+1)(2N_1 + N_1N_2)$	$(k+1)(N_1N_2)$
Decoder Memory (32-bit words)	$k(N_1 + N_2)$	kN_1N_2	kN_1N_2
Encoder Computational Requirement (FLOPs/sample)	$N_1 + N_2$	$N_1 + N_2$	N_1N_2

Table 10: Complexity of Two-Stage CM-MSVQ and CM-TSVQ and COVQ; k = Dimension; N_i = Number of Codevectors at Stage i, $i = 1, 2$.

	CM-MSVQ	Tandem Scheme
Encoder Memory (32-bit words)	704	832
Decoder Memory (32-bit words)	640	768
Encoder Computational Requirement (FLOPs/sample)	64	64
Delay (Number of samples)	10	24

Table 11: Complexity and Delay of the CM-MSVQ and the Tandem Scheme in Figure 12.

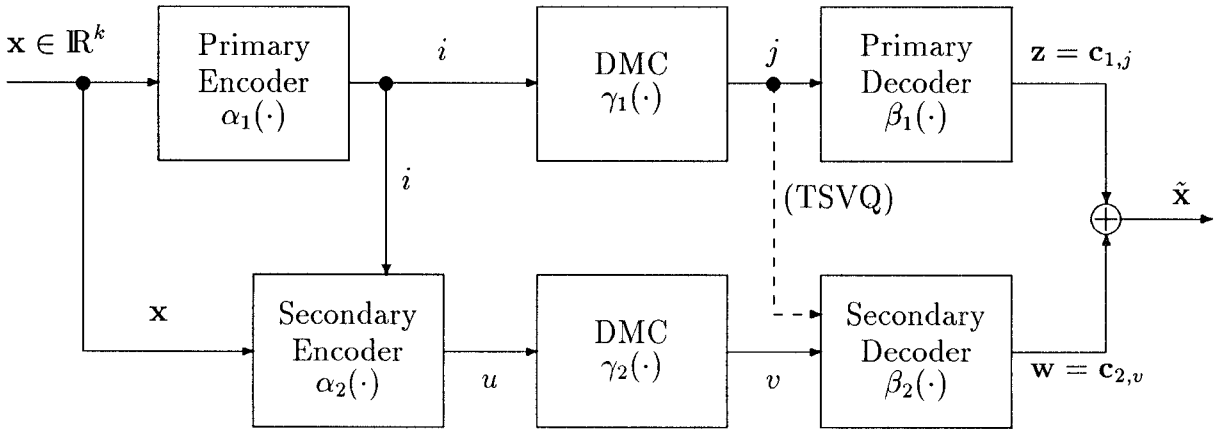


Figure 1: Block Diagram of the TSVQ/MSVQ Scheme Over Noisy Channels.

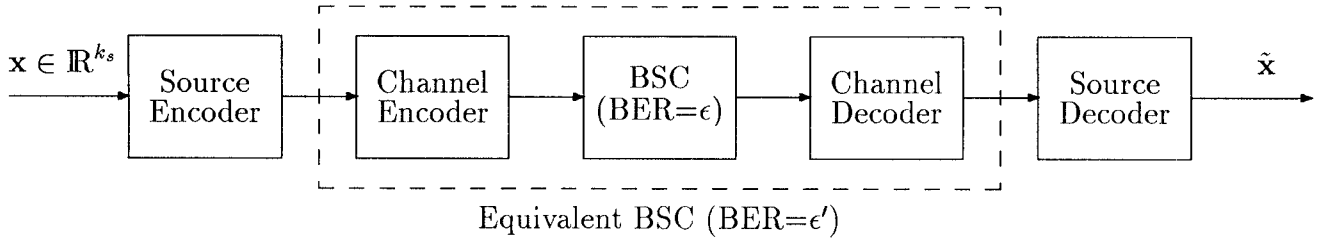


Figure 2: Block Diagram of a Tandem Source-Channel Coding Scheme Over a BSC and the Equivalent Channel Model.

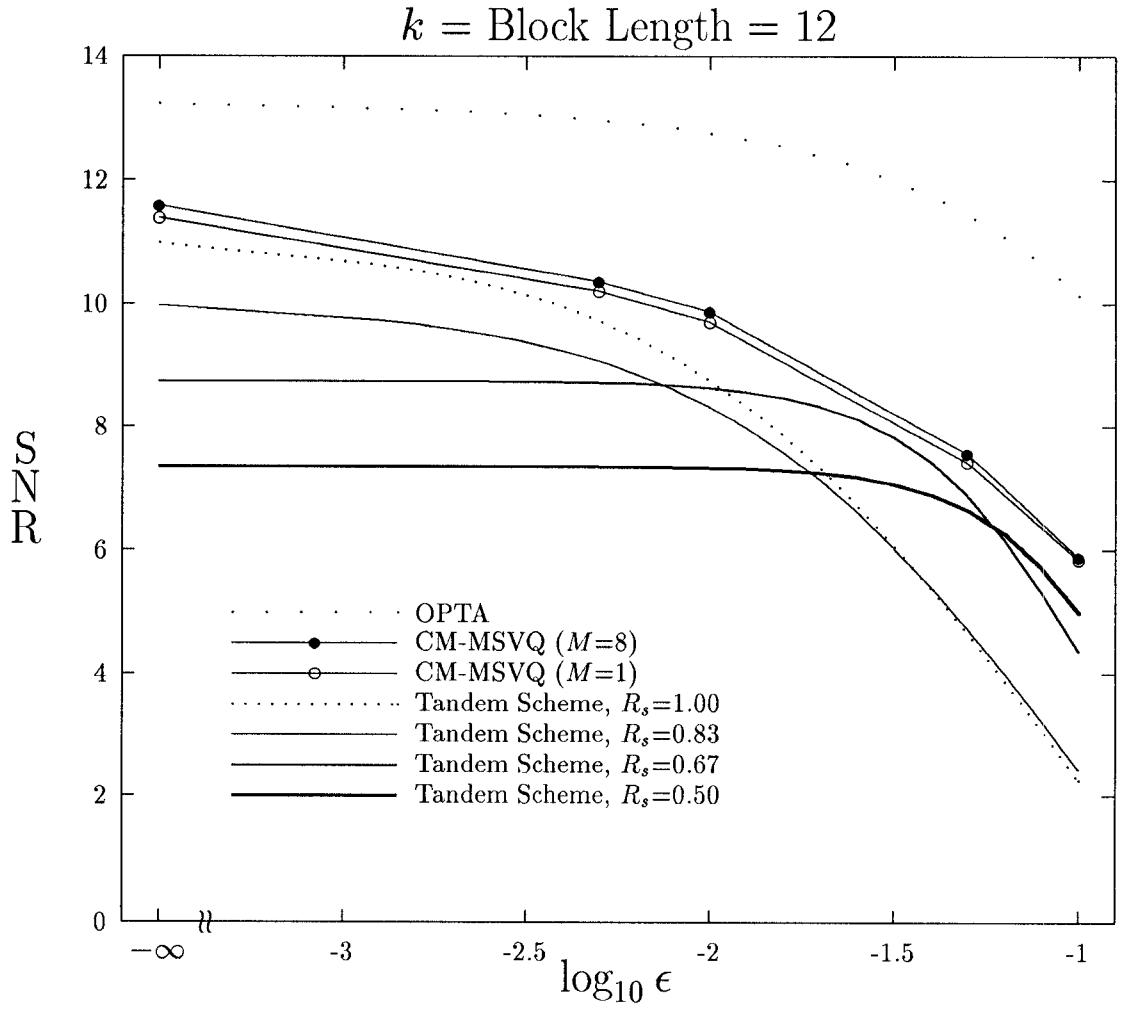


Figure 3: SNR (in dB) Performances of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme = 6; l = Number of Source Blocks per Channel Block = 2; k = Effective Block Length of Both Schemes = 12.

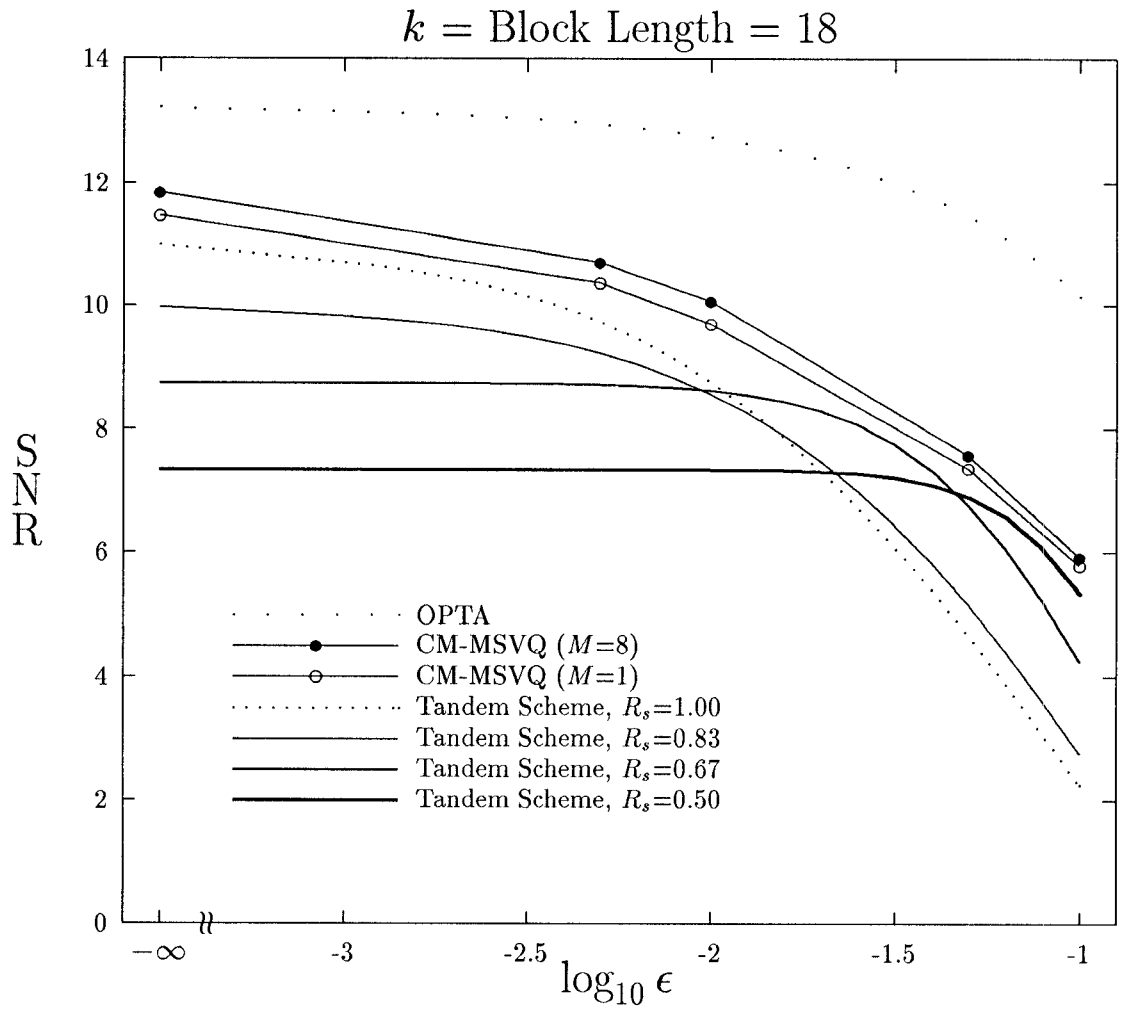


Figure 4: SNR (in dB) Performances of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme = 6; l = Number of Source Blocks per Channel Block = 3; k = Effective Block Length of Both Schemes = 18.

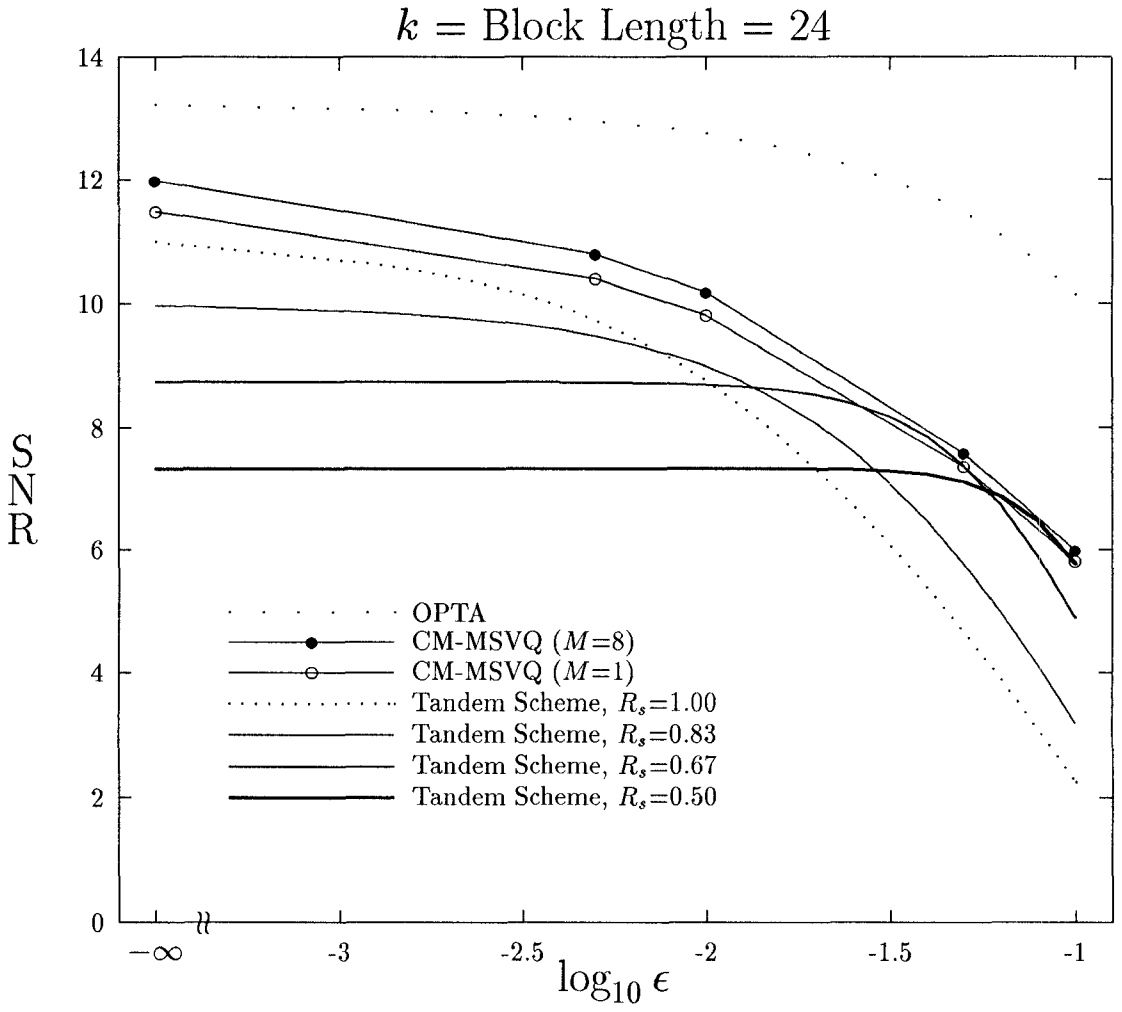


Figure 5: SNR (in dB) Performances of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme = 6; l = Number of Source Blocks per Channel Block = 4; k = Effective Block Length of Both Schemes = 24.

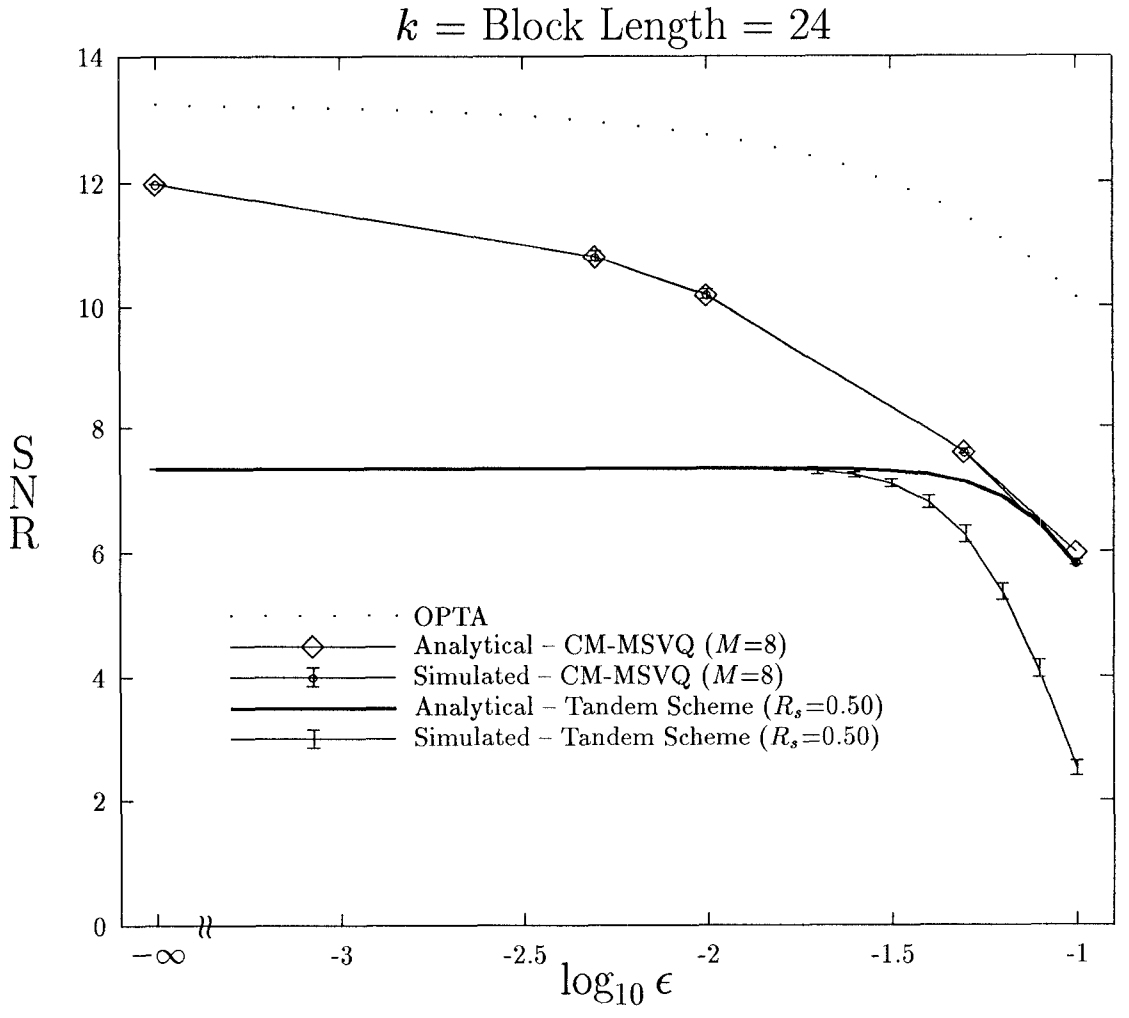


Figure 6: Comparisons Between Analytical Results and Simulated Results of the Tandem Source-Channel Coding Scheme and the CM-MSVQ Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme = 6; l = Number of Source Blocks per Channel Block = 4; k = Effective Block Length of Both Schemes = 24; Error Bars Indicate the Minimum and Maximum SNRs of the Simulations.

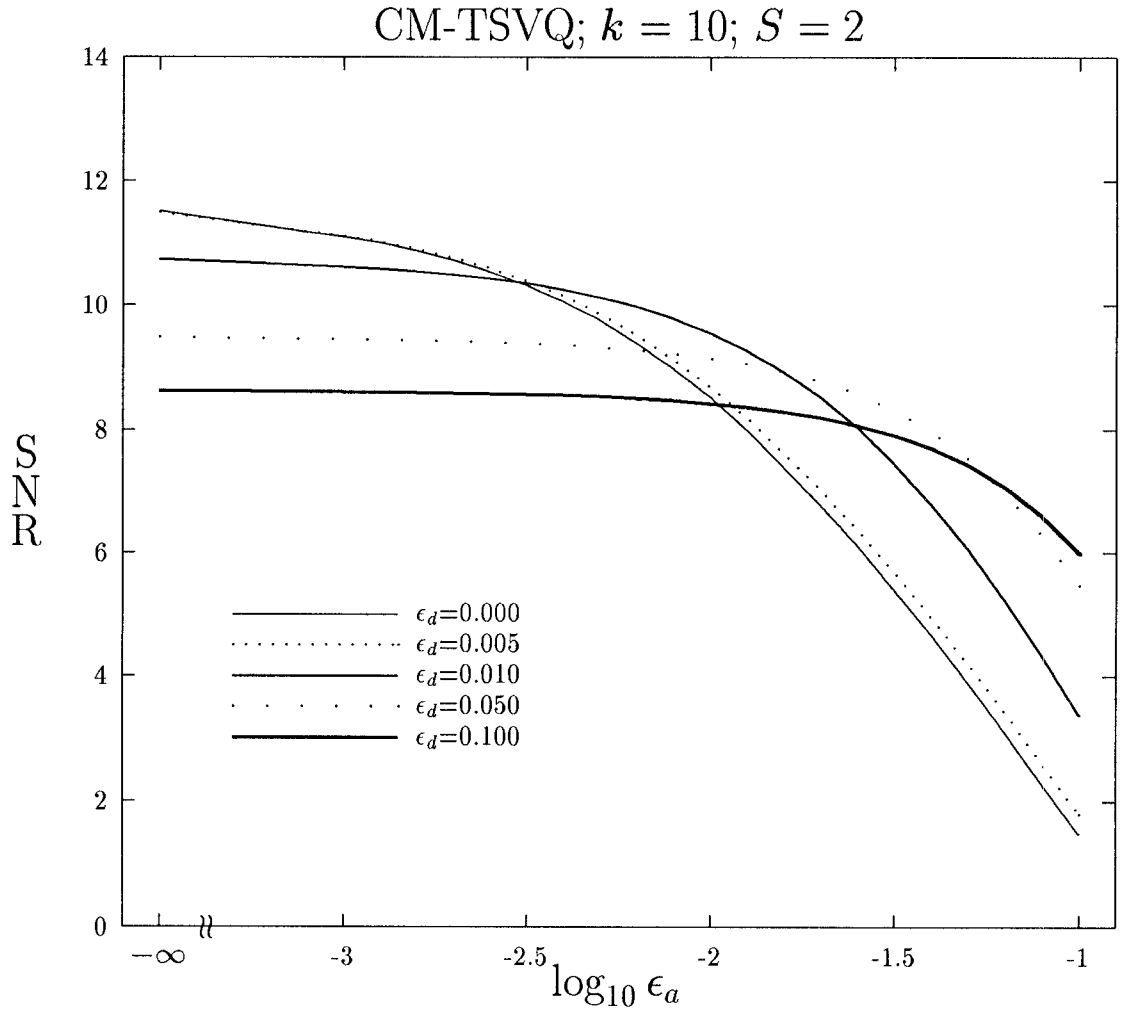


Figure 7: SNR (in dB) Performances of the CM-TSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; ϵ_a = Actual Channel BER; ϵ_d = Design BER for the First and Second Stages; k = Dimension; S = Number of Stages.

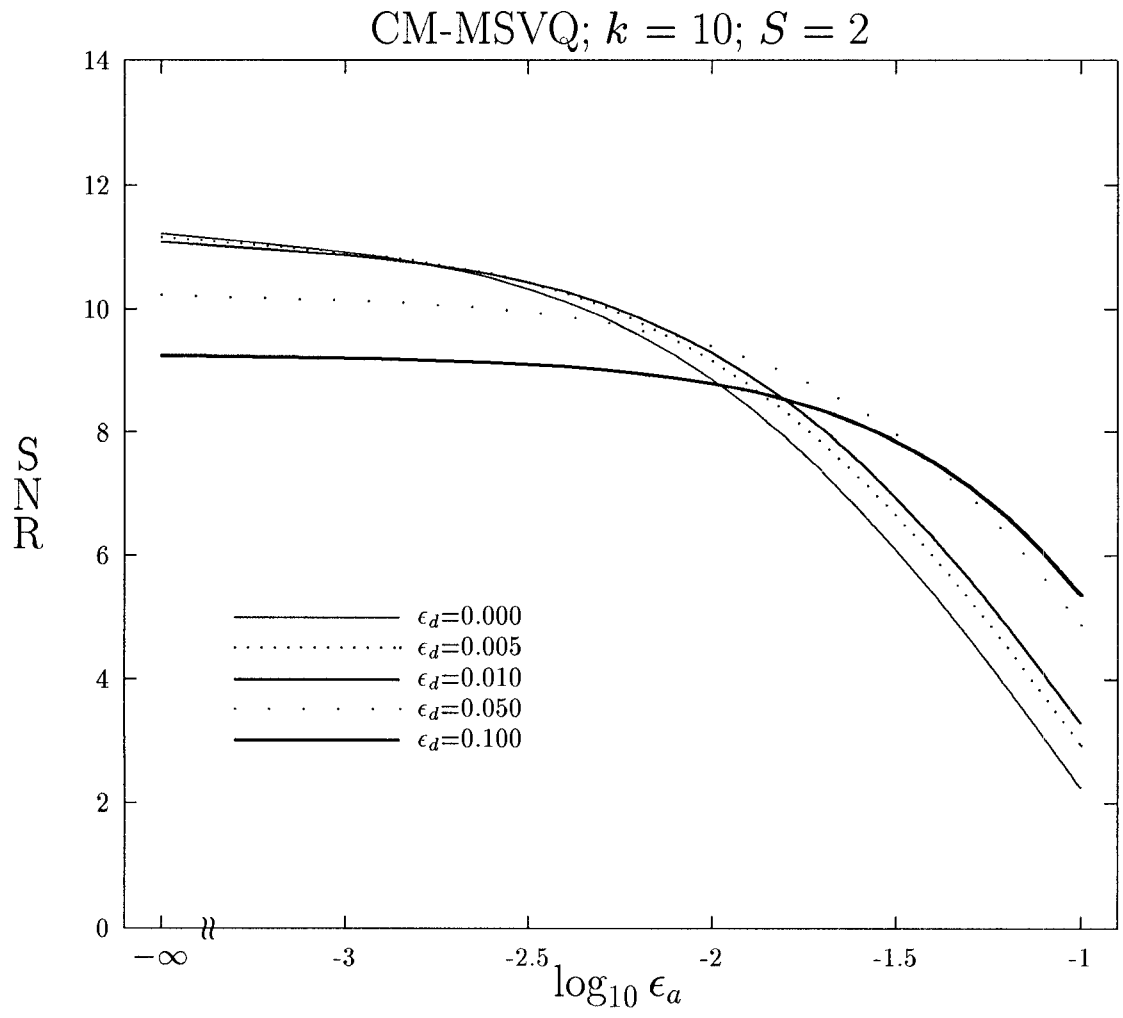


Figure 8: SNR (in dB) Performances of the CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; ϵ_a = Actual Channel BER; ϵ_d = Design BER for the First and Second Stages; k = Dimension; S = Number of Stages.

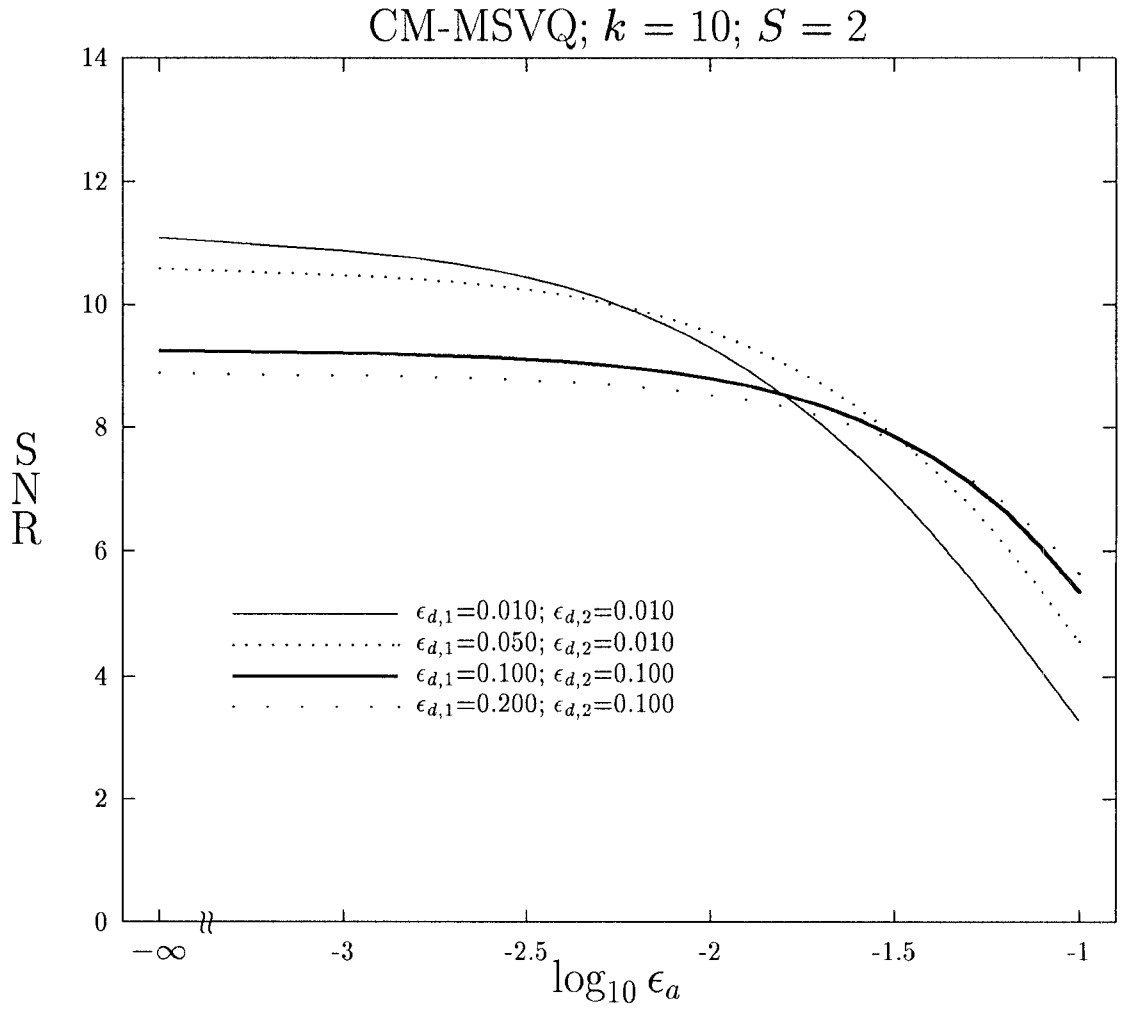


Figure 9: SNR (in dB) Performances of the CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; ϵ_a = Actual Channel BER; $\epsilon_{d,1}$ = Design BER for the First Stage; $\epsilon_{d,2}$ = Design BER for the Second Stage; k = Dimension; S = Number of Stages.

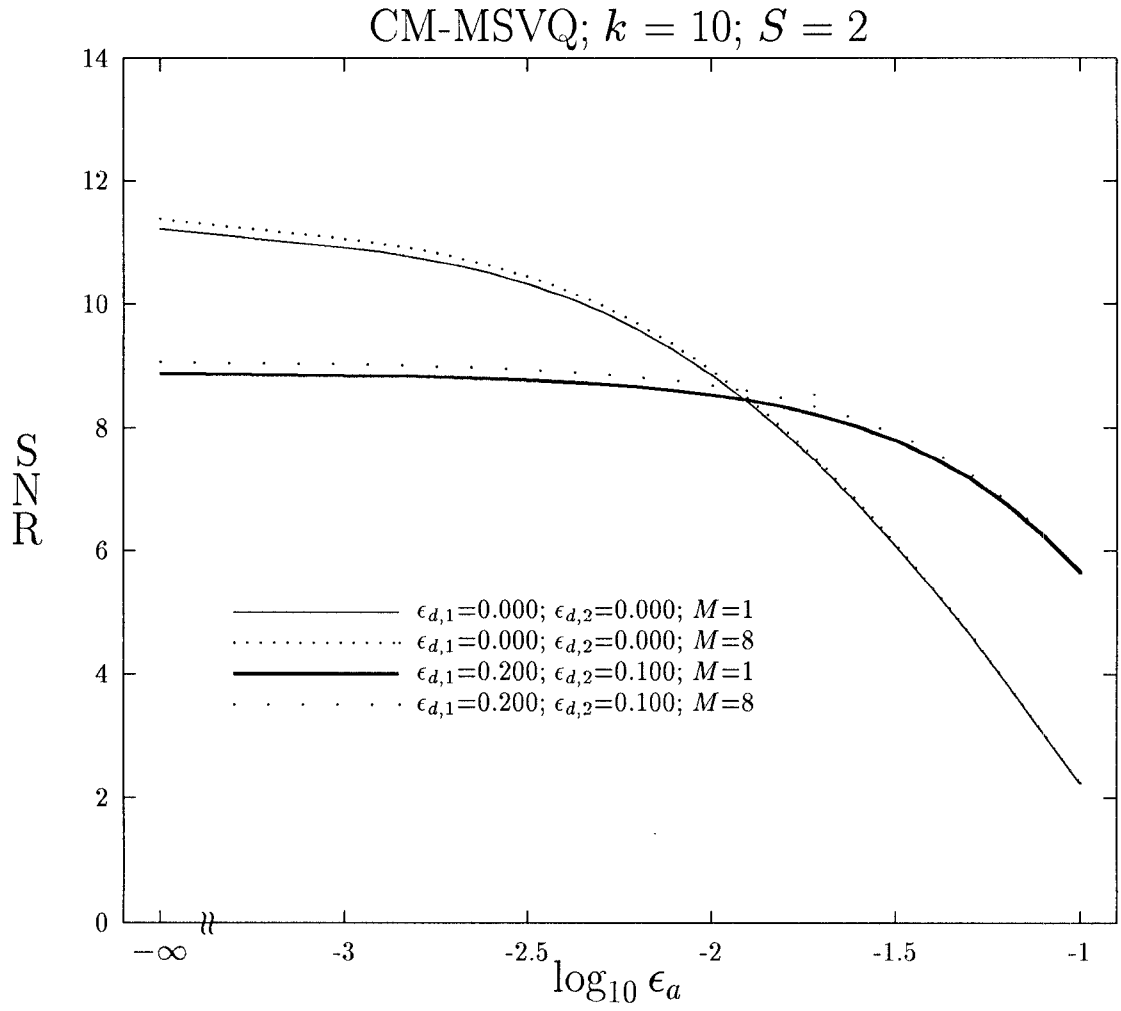


Figure 10: SNR (in dB) Performances of the Multiple Candidate CM-MSVQ for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; ϵ_a = Actual Channel BER; $\epsilon_{d,1}$ = Design BER for the First Stage; $\epsilon_{d,2}$ = Design BER for the Second Stage; M = Number of Candidates; k = Dimension; S = Number of Stages.

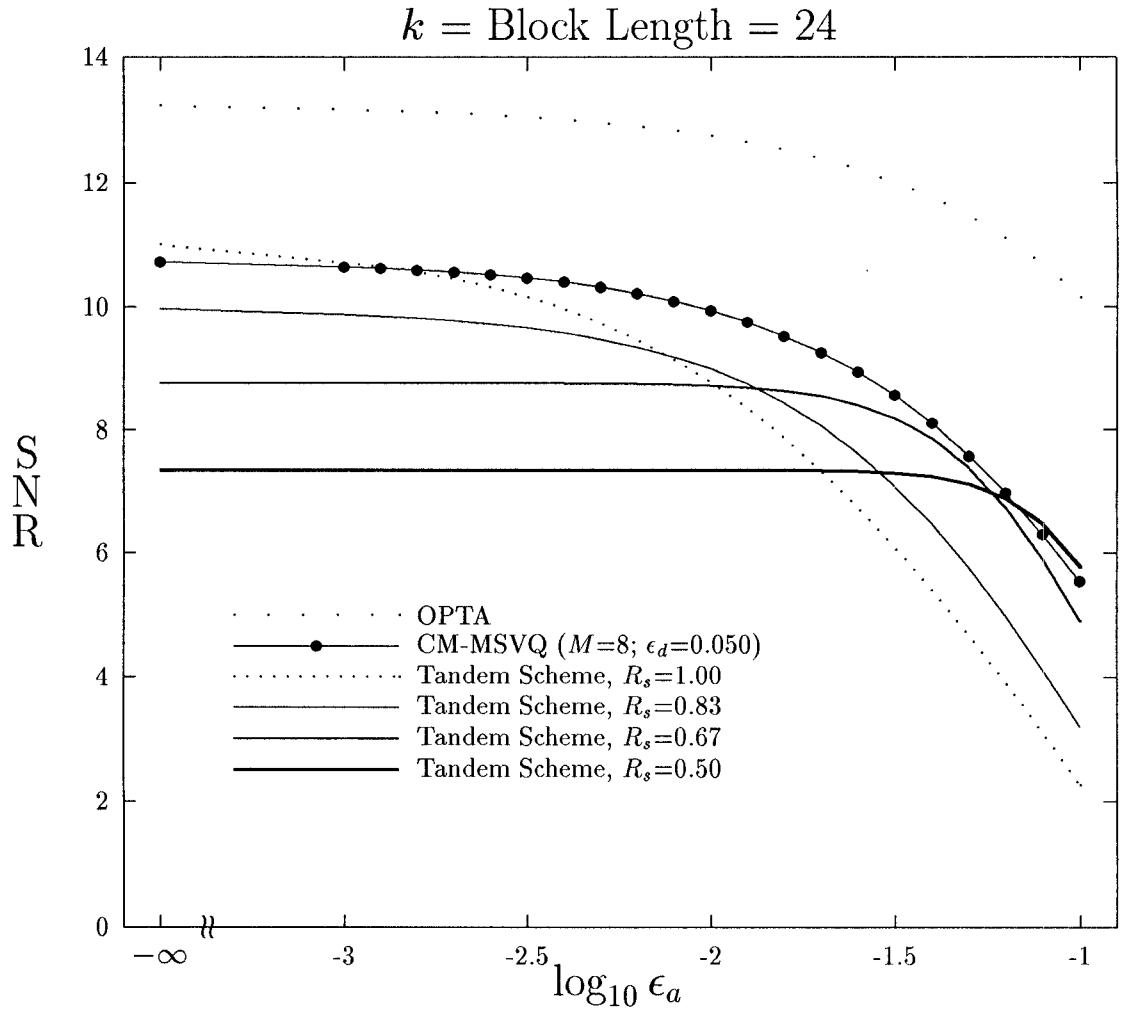


Figure 11: SNR (in dB) Performances of the CM-MSVQ and the Tandem Scheme for the Gauss-Markov Source with Correlation Coefficient 0.9 under Channel Mismatch Conditions; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; M = Number of Candidates; ϵ_d = Design BER for the CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; ϵ_a = Actual Channel BER; k = Effective Block Length of Both Schemes = 24.

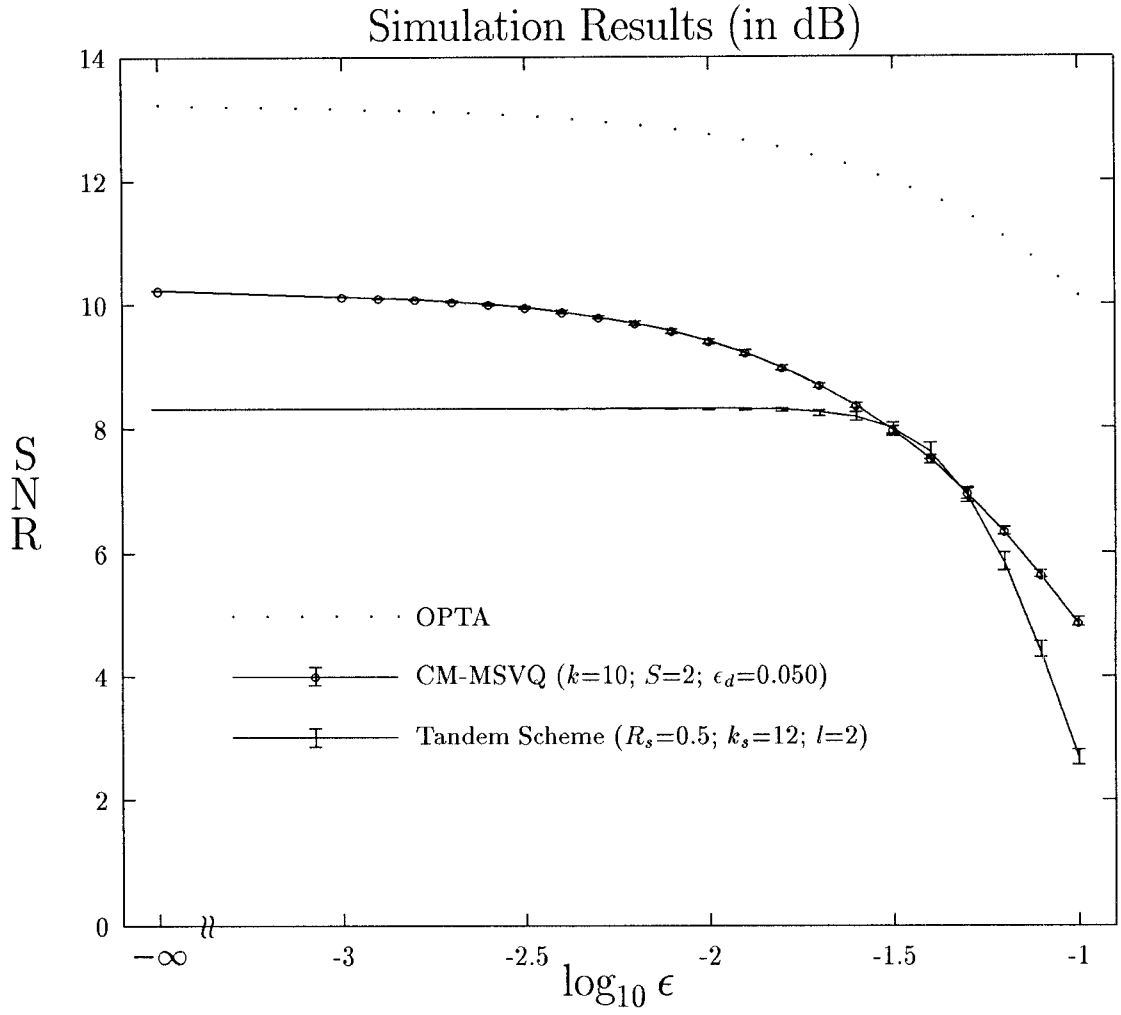


Figure 12: Comparisons Between CM-MSVQ and Tandem Scheme with the Same Complexity; Gauss-Markov Source with $\rho = 0.9$; Overall Rate = 1 Bit/Sample; OPTA = Optimum Performance Theoretically Attainable; k = Block Length of CM-MSVQ; S = Number of Stages; ϵ_d = Design BER of CM-MSVQ; R_s = Rate of Source Code of Tandem Scheme; k_s = Block Length of Source Code of Tandem Scheme; l = Number of Source Blocks per Channel Block; Error Bars Indicate the Minimum and Maximum SNRs of the Simulations.