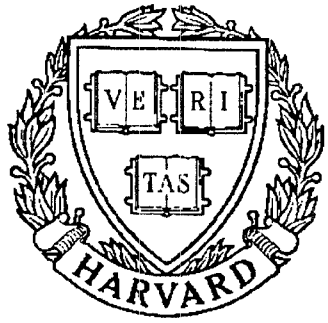
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



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

Finite-State Vector Quantization for Noisy Channels

by Y. Hussain and N. Farvardin

TR 92-69 r1

Finite-State Vector Quantization for Noisy Channels[†]

Y. Hussain and N. Farvardin
Electrical Engineering Department
and
Institute for Systems Research
University of Maryland
College Park, MD 20742

Abstract

Under noiseless channel conditions and for sources with memory, finite-state vector quantizers (FSVQs) exhibit improvements over memoryless vector quantizers. It is shown, however, that in the presence of channel noise, the performance of FSVQ degrades significantly. This suggests that for noisy channels, the FSVQ design problem needs to be reformulated by taking into account the channel noise. Using the developments in channel-optimized vector quantization, we describe two different noisy-channel FSVQs. We show by means of simulations on the Gauss-Markov source and speech LSP parameters and for a binary symmetric channel that both schemes are fairly robust to channel noise. For the Gauss-Markov source, the proposed noisy channel FSVQs perform at least as well as or better than channel-optimized vector quantization, while for speech LSP parameters, they lead to a saving of 1.5-4 bits/frame over channel-optimized vector quantization depending on the level of noise in the channel.

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

1 Introduction

Vector quantization (VQ)[1] as a means for data compression has been the subject of much research over the last decade [1]-[6]. The main motivation behind the use of VQ resides in Shannon's result stating that VQ can attain a rate-distortion performance close to the "best possible" in the limit when the vector dimension goes to infinity [7]. In practice, however, only finite dimensions can be implemented and practical VQ systems fall short of achieving the best, while still performing much better than scalar quantizers.

One of the main advantages of VQ over scalar quantization is its ability to exploit the source sample-to-sample correlation. Although the rate-distortion performance of the vector quantizer improves by increasing its dimension, the resulting complexity places a practical limit on how large a dimension can be used and therefore how effectively the source memory can be exploited. An alternative method to improve the performance of VQ (for a given rate) is to incorporate memory in the VQ structure. One such technique is finite-state vector quantization (FSVQ) [2], [3], [8], [9], which can be thought of as a time varying VQ. An FSVQ is a finite-state machine with one VQ associated with each state; the encoder and the decoder share the same state space and the decoder can track the encoder state sequence based on the received encoder output sequence. Use of finite-state machines in general digital transmission systems was investigated in [10] and FSVQ was first introduced in [8]. Various forms of FSVQ have been suggested for applications in image coding [3], [11]-[14], speech waveform coding [3], [8], [14] and coding of speech spectral parameters [9].

An FSVQ can be thought of as a two-stage quantizer. The first stage utilizes the inter-vector memory by performing a rough quantization based on the current state (obtained from the previous output and state) and the second stage performs a finer quantization thus leading to a performance superior to that of a memoryless VQ operating at the same rate. However, it is only under noiseless channel conditions that the decoder can track the encoder state sequence. If the channel is noisy, even a single error in the transmitted encoder output can lead to an incorrect decoder state sequence over a period of time and therefore a significant degradation in performance. In this paper we study the performance of FSVQ in the presence of channel noise and devise new FSVQ designs for operation over a noisy channel.

Previous work on quantizer design for noisy channels includes scalar quantization [15], [16], full-searched VQ [17], [18] as well as tree-structured VQ (TSVQ) and multi-stage

[1]Throughout the paper we use the acronym VQ to denote vector quantization or vector quantizer. The usage will be clear from the context.

VQ (MSVQ) in [19]. Furthermore, necessary conditions of optimality for trellis encoding systems have been developed in [20] and a design algorithm is described in [21]. An FSVQ can be looked upon as a trellis encoding system with unit search depth and a general next-state function (as opposed to a shift-register based next-state function). However, due to the use of a general next-state function in an FSVQ, the approach of [20] for the trellis encoding system cannot be used as such. We shall elaborate on this in Section 2.3.

In this work, using some simplifying assumptions on the FSVQ structure we describe two noisy-channel FSVQ systems. In the first scheme, we assume that a "protected" encoder state is transmitted periodically to the decoder, while in the second scheme, we modify the FSVQ structure in such a manner that without too big a loss in performance (as compared to the FSVQ of [8] used over a noiseless channel), the codewords explicitly contain the state information. This implies that a correct reception of the codeword is tantamount to a correct decoder state. In both schemes the state codebooks are designed based on a noisy channel assumption.

An important application that we will consider in this paper is that of quantizing speech LSP parameters [22]. Currently, there is a growing interest in encoding speech LSP parameters [23]-[26], both in vocoders [27] and in hybrid speech coders [28]-[31]. In many speech communication situations, it is necessary that the speech coder be robust against transmission errors. An important application is digital cellular networks where large channel error rates may arise from various sources such as multipath fading and interference from other channels [32]. We will show that the noisy-channel FSVQ systems proposed here present an interesting alternative for such situations. The proposed schemes efficiently utilize the interframe and intraframe correlation of LSP parameters while providing a fairly robust performance in the presence of channel noise.

The paper is organized as follows. Section 2 includes a brief description of FSVQ, a discussion of its sensitivity to channel noise and the formulation of the noisy-channel FSVQ design problem. In Sections 3 and 4, the description and design algorithms of the proposed noisy-channel FSVQs are provided. Section 5 includes simulation results for the Gauss-Markov source and speech LSP parameters followed by a summary and conclusions in Section 6.

2 Preliminaries

2.1 Definition of FSVQ

An L-dimensional K-state code [8] is specified by a state space $\mathcal{S} = \{0, 1, \ldots, K-1\}$ and the mappings:

$\alpha : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{N}$: finite-state encoder,

β: $\mathcal{N} \times \mathcal{S} \rightarrow \hat{\mathcal{A}}$: finite-state decoder,

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

where $\mathcal{N} \triangleq \{0, 1, \ldots, N-1\}$ is the channel alphabet of size N and $\hat{\mathcal{A}}$ is the reproduction space.

Let $\{\mathbf{x}_n\}_{n=0}^{\infty}$ denote the input vector sequence, where $\mathbf{x}_n \in \mathbb{R}^L$. Similarly, let $\{u_n\}_{n=0}^{\infty}$, $\{s_n\}_{n=0}^{\infty}$ and $\{\hat{\mathbf{x}}_n\}_{n=0}^{\infty}$ denote the channel symbol sequence, state sequence and reproduction vector sequence, respectively. Given an initial state s_0, the input sequence determines the sequence of channel symbols, reproduction vectors and states according to:

$$u_n = \alpha(\mathbf{x}_n, s_n), \tag{1}$$

$$\hat{\mathbf{x}}_n = \beta(u_n, s_n), \tag{2}$$

$$s_{n+1} = f(u_n, s_n), \quad n = 0, 1, \ldots. \tag{3}$$

The next state depends only on the present state and the output channel symbol; therefore, given the initial state and correct channel symbol sequence, the decoder can track the state sequence. Here, $\mathcal{C}_s \triangleq \{\beta(u, s), u \in \mathcal{N}\}$ is the codebook associated with state s and $\hat{\mathcal{A}} = \bigcup_{s=0}^{K-1} \mathcal{C}_s$. Consider the distortion measure $d : \mathbb{R}^L \times \hat{\mathcal{A}} \rightarrow [0, \infty)$ which assigns a non-negative cost $d(\mathbf{x}, \hat{\mathbf{x}})$ to reproducing the input vector $\mathbf{x}$ as $\hat{\mathbf{x}}$. As defined in [8], a *finite-state vector quantizer* (FSVQ) is a finite-state code with α given by the minimum distortion rule

$$\alpha(\mathbf{x}, s) = \arg\min_{u \in \mathcal{N}} d(\mathbf{x}, \beta(u, s)), \; s \in \mathcal{S}. \tag{4}$$

The average distortion incurred in an FSVQ system is given by $\frac{1}{L} E[d(\mathbf{X}, \hat{\mathbf{X}})]$, where the expectation is taken with respect to the source distribution.[2] The rate is given by $R = \log_2 N$, bits/vector.

[2]We have assumed that the joint source-reproduction process is stationary and ergodic implying that the expected distortion can be approximated from the long-term sample average distortion. However, even under milder assumptions (see [8] and references cited therein) on the source, similar conclusions can be made. Therefore, throughout the paper, in the analysis we use the expected distortion, but for the design and simulations the sample average distortion based on a long sequence of source vectors will be used.

2.2 Performance of FSVQ in the presence of channel noise

Due to its built-in memory, FSVQ is capable of exploiting inter-vector correlation leading to improved performance over a memoryless VQ. However, FSVQ is prone to channel error propagation due to the presence of a feedback structure in its encoder. In the absence of channel noise, the feedback eliminates the need to transmit the encoder state to the receiver side. However, the ability of the decoder to track the encoder state sequence *critically* depends on the availability of the *exact* replica of the transmitted codewords. Even a single error occurring in the transmitted codeword can lead to an incorrect decoder state. Once the decoder state is different from the encoder state, the decoder state sequence can remain "derailed" for a long time. In practice, since there is only a finite, and usually small number of states, the encoder and decoder states resynchronize after some time. Nevertheless, the performance degradation is severe because once the decoder loses track of the encoder state, its state sequence essentially becomes random and the input vectors get mapped to random reproduction vectors till the two state sequences resynchronize. This statement will be confirmed by the simulation results provided later. In order to control the performance of FSVQ under noisy channel conditions, it is essential to minimize the derailing of the decoder. This can be achieved by either (i) transmitting some "protected version" of the encoder state sequence periodically or (ii) modifying the FSVQ system so that it becomes self-tracking (i.e., decoder can retrack soon after derailing). We elaborate on these issues in Sections 3 and 4. In what follows, we formulate the FSVQ design problem under noisy channel conditions.

2.3 Problem formulation

Consider the FSVQ block diagram in Fig. 1. The encoder α can be described in terms of the partition $\{\mathcal{P}_{s,u};\ s \in \mathcal{S}, u \in \mathcal{N}\}$ such that $\alpha(\mathbf{x}, s) = u$, if $\mathbf{x} \in \mathcal{P}_{s,u},\ s \in \mathcal{S}, u \in \mathcal{N}$. The output u of the encoder α is transmitted over a discrete memoryless channel (DMC) described by a random mapping $\gamma : \mathcal{N} \rightarrow \mathcal{N}$ and the transition matrix

$$Q(v|u) = Pr(\gamma(u) = v),\ u, v \in \mathcal{N}, \tag{5}$$

where v is the output of the DMC. The decoder uses the received index v and its state $\hat{s}$ to generate the reproduction vector $\hat{\mathbf{x}} = \beta(v, \hat{s})$. The average distortion is given by

$$D(\alpha, \beta, f) = \frac{1}{L} E[d(\mathbf{X}, \hat{\mathbf{X}})], \tag{6}$$

where the expectation is taken with respect to the source and channel distributions. The problem is to minimize D by appropriate design of α, β and f.

Toward designing a noisy-channel FSVQ, we first consider the following theorem which establishes a formula for simplifying the average distortion D.

Theorem 1: Consider an FSVQ defined according to (1)-(3) and a DMC described by (5). Define $d'_s(\mathbf{x}, u)$ by

$$d'_s(\mathbf{x}, u) = \sum_{\hat{s},v} Q(v|u) Pr(\hat{s}|s) d(\mathbf{x}, \beta(v, \hat{s})), \tag{7}$$

where $\mathbf{x}, s, \hat{s}, u$ and v denote, respectively, the source vector, encoder state, decoder state, channel input index and channel output index all at the same time instant, say n. Then, the average distortion $D(\alpha, \beta, f)$ can be written as

$$D(\alpha, \beta, f) = \frac{1}{L} \sum_{s,u} Pr(s) \int_{\mathcal{P}_{s,u}} d'_s(\mathbf{x}, u) p(\mathbf{x}) d\mathbf{x}. \tag{8}$$

Proof: See Appendix A.

Here $d'_s(\mathbf{x}, u)$ represents the expected distortion (with respect to channel distribution) when the source vector $\mathbf{x}$ is encoded by an index u, given that the encoder is in state s. This implies that α, or equivalently the partition, can be obtained from a generalization of equation (4) according to

$$\alpha(\mathbf{x}, s) = \arg\min_{u \in \mathcal{N}} d'_s(\mathbf{x}, u), \; s \in \mathcal{S}. \tag{9}$$

Thus, for a fixed f and β, the encoder α is described by (9).[3]

Also, it is clear from (7) and (8) that for a fixed α and f, the optimum β is given by

$$\beta^*(v, \hat{s}) = \arg\min_{\hat{\mathbf{x}} \in \mathbb{R}^L} \sum_{s,u} Pr(s) Q(v|u) Pr(\hat{s}|s) \int_{\mathcal{P}_{s,u}} d(\mathbf{x}, \hat{\mathbf{x}}) p(\mathbf{x}) d\mathbf{x}, \; \hat{s} \in \mathcal{S}, v \in \mathcal{N}. \tag{10}$$

For a fixed α and β, determining the optimum f is not easy but some *ad hoc* rules can be used to get good next-state maps, leading to an overall suboptimal system just as in the noiseless-channel FSVQ of [8].

Notice that determining both the encoder in (9) and optimum decoder in (10) require the knowledge of $Pr(\hat{s}|s)$. Unfortunately, for a general next-state function, it is not clear how s and $\hat{s}$ are related to each other, thus making the computation of $Pr(\hat{s}|s)$ difficult. Next, we present two special cases in which certain assumptions about the system are made which make the computation of $Pr(\hat{s}|s)$ straightforward.

[3]Since the encoder given by (9) involves no delay, it is not optimal for the given decoder. However, if the encoder is allowed to look ahead (i.e., introduce delay) then it will become nearly optimal for the given decoder and the resulting FSVQ will then become a trellis encoding system with a vector alphabet. A similar observation is made in [8] for FSVQs without channel noise.

3 Description and Design of NC-FSVQ1

In this section we consider the first special case in which we assume that the encoder state sequence is known to the decoder (by transmitting the protected state indices). Assuming that $\hat{s} = s$ at all times (i.e., $Pr(\hat{s}|s) = 1$, if $\hat{s} = s$ and $Pr(\hat{s}|s) = 0$, otherwise), we have

$$d'_s(\mathbf{x}, u) = \sum_v Q(v|u) d(\mathbf{x}, \beta(v, s)), \tag{11}$$

which, for each state s, is the same as the modified distortion measure used for the design of a channel-optimized VQ (CO-VQ) in [18]. Therefore, for a fixed β and f, the encoder α is given by (9) with d'_s given by (11). This exactly corresponds to determining the optimum partition for CO-VQ in [17], [18] within each state $s \in \mathcal{S}$. Similarly for a fixed α and f, assuming that $\hat{s} = s$ at all times, the optimum β is obtained using (10) and can be expressed as

$$\beta^*(v, \hat{s}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^L} \sum_u Q(v|u) \int_{\mathcal{P}_{\hat{s},u}} d(\mathbf{x}, \hat{\mathbf{x}}) p(\mathbf{x}) d\mathbf{x}, \ \hat{s} \in \mathcal{S}, v \in \mathcal{N}. \tag{12}$$

Equation (12) corresponds to determining the optimum reproduction vectors in [17], [18] (for a fixed partition) within each state $s \in \mathcal{S}$. Finally, for a fixed α and β, f is determined in an *ad hoc* fashion as described in the following noisy-channel FSVQ design algorithm.

Algorithm:

1. Design an LBG-VQ [1] with K codevectors for the given training sequence. This is referred to as the state-label VQ, $\mathcal{C} = \{\mathbf{c}(s), s \in \mathcal{S}\}$.

2. For each s, use the algorithm of [18] to design a codebook $\mathcal{C}_s = \{\beta(u, s), u \in \mathcal{N}\}$ on the subsequence composed of all successors to vectors for which the state-label VQ chooses s, i.e., the subsequence $\{\mathbf{x}_n : s = \arg\min_{k \in \mathcal{S}} d(\mathbf{x}_{n-1}, \mathbf{c}(k))\}$.

3. As in [8], define a next-state function f by[4]

$$f(u, s) = \arg \min_{k \in \mathcal{S}} d(\beta(u, s), \mathbf{c}(k)), \ s \in \mathcal{S}, u \in \mathcal{N}. \tag{13}$$

4. Attempt to improve the state codebooks $\{\mathcal{C}_s, s \in \mathcal{S}\}$ by encoding (as in [18]) the training sequence using the next-state function obtained in step (3) and updating each codevector by the generalized centroid of its corresponding cell [17], [18]. Also update the state-label VQ $\mathcal{C}$.

[4]It might appear that since we are explicitly transmitting the encoder state to the decoder (i.e., the system is omniscient in the sense of [8]), it is unnecessary to define the next-state function based on the reproduction vectors. But as we will see later, this definition of next-state function helps in reducing the overhead information by transmitting the next-state indices only periodically.

The difference between the above algorithm and the one in [8] is that here the state codebooks are channel-optimized codebooks and the encoding is done as described in [18] within each state. Also, it is assumed that the decoder has perfect knowledge of the encoder state sequence. This implies that a "protected" encoder state sequence needs to be transmitted to the receiver. Since transmission of the protected encoder state for each vector can lead to a large overhead, the encoder state index is transmitted only periodically, say, every p frames; given the state indices at times k and $k+p$ (s_k and s_{k+p}) and the received codewords $\{v_i\}$ at times $i = k, \ldots, k+p-1$, a maximum a posteriori (MAP) estimate of the encoder state at times $i = k+1, \ldots, k+p-1$, is obtained at the decoder. Specifically, upon denoting a generic vector $(u_i, u_{i+1}, \cdots, u_j)$ by $\mathbf{u}_i^j$, we try to determine $\mathbf{u}^{*k+p-1}_k$ according to

$$\mathbf{u}^{*k+p-1}_k = \arg \max_{\mathbf{u}_k^{k+p-1}} Pr(\mathbf{u}_k^{k+p-1} | \mathbf{v}_k^{k+p-1}, s_k, s_{k+p}). \tag{14}$$

The probability in the right hand side of (14) can be written as

$$Pr(\mathbf{u}_k^{k+p-1} | \mathbf{v}_k^{k+p-1}, s_k, s_{k+p}) =$$

$$Pr(\mathbf{v}_k^{k+p-1} | \mathbf{u}_k^{k+p-1}, s_k, s_{k+p}) Pr(s_{k+p} | \mathbf{u}_k^{k+p-1}, s_k) Pr(\mathbf{u}_k^{k+p-1} | s_k) \frac{Pr(s_k)}{Pr(\mathbf{v}_k^{k+p-1}, s_k, s_{k+p})}. \tag{15}$$

Therefore, it suffices to maximize the product of the first three terms in (15). Due to the fact that the channel is memoryless, we have

$$Pr(\mathbf{v}_k^{k+p-1} | \mathbf{u}_k^{k+p-1}, s_k, s_{k+p}) = Pr(\mathbf{v}_k^{k+p-1} | \mathbf{u}_k^{k+p-1}) = \prod_{i=k}^{k+p-1} Q(v_i | u_i). \tag{16}$$

Given s_k and $\mathbf{u}_k^{k+p-1}$, s_{k+p} is known (by successive use of the next-state function) and can be written as a deterministic function $s_{k+p} = h(s_k, \mathbf{u}_k^{k+p-1})$. Thus,

$$Pr(s_{k+p} | \mathbf{u}_k^{k+p-1}, s_k) = \delta(s_{k+p}, h(s_k, \mathbf{u}_k^{k+p-1})), \tag{17}$$

where $\delta(.,.)$ is the Kronecker delta function. Finally, using the chain rule, we have

$$Pr(\mathbf{u}_k^{k+p-1} | s_k) = Pr(u_k | s_k) \prod_{i=k+1}^{k+p-1} Pr(u_i | s_k, \mathbf{u}_k^{i-1}). \tag{18}$$

However, assuming that the FSVQ is a reasonable one in that the state at any time instant captures essentially all the memory conveyed by the transmitted sequence, we can approximate $Pr(u_i | s_k, \mathbf{u}_k^{i-1})$ by $Pr(u_i | h(s_k, \mathbf{u}_k^{i-1})$. Using this approximation and combining (14)-(18), $\mathbf{u}^{*k+p-1}_k$ can be obtained by maximizing

$$Pr(u_k | s_k) Q(v_k | u_k) \prod_{i=k+1}^{k+p-1} Q(v_i | u_i) Pr(u_i | h(s_k, \mathbf{u}_k^{i-1})) \delta(s_{k+p}, h(s_k, \mathbf{u}_k^{k+p-1})), \tag{19}$$

which can be achieved by a Viterbi-type [33] algorithm. Having computed $\mathbf{u}^{*k+p-1}_k$, the state estimates are obtained recursively from $s^*_i = f(u^*_{i-1}, s^*_{i-1})$, $i = k+1, \ldots, k+p-1$, with $s^*_k = s_k$. This procedure introduces a decoding delay of p frames.

In the above procedure, there is a nonzero probability that some states are estimated incorrectly. However, the noisy-channel FSVQ described so far assumes that the encoder and decoder are in the same state and only the codeword index can be corrupted by the channel noise. If the decoder state differs from the encoder state at a certain time instant, then the resulting error can be very large even if the codeword is received correctly. To reduce the resulting distortion in such situations, we perform a judicious indexing of the codevectors *among* the states. The basic idea is as follows: The codevectors among those states that are close in the Euclidean sense are assigned binary codewords that are close in the Hamming sense; the highest priority is given to those states which are most likely to be confused with each other. The details of the index assignment algorithm can be found in [34]. This algorithm leads to a more robust performance especially for large values of p. In the sequel, we refer to this modified FSVQ system (with protected encoder state transmission, state estimation and codeword reassignment) as noisy-channel FSVQ1 (NC-FSVQ1). For a noiseless channel, NC-FSVQ1 reduces to noiseless-channel FSVQ [8], and will be referred to as FSVQ1.

4 Description and Design of NC-FSVQ2

Although NC-FSVQ1 offers robustness against channel noise (to be discussed in Section 5), it suffers from a decoding delay that might be objectionable in low-delay systems. The decoder receives the protected encoder states say at times k and $k+p$ and estimates, based on the received codewords, the states at intermediate times $k+1, k+2, \ldots, k+p-1$. Thus the decoder must wait till time $k+p$ to estimate the encoder state at time $k+1$, leading to a maximum delay of p source vectors. Furthermore, there appears to be some waste of overhead bits in encoding the protected encoder state. After all, the encoder next state is implicitly contained (at least partially) in the current transmitted codeword and it seems that protecting the codeword (or part of it) instead of the state (as in NC-FSVQ1) might lead to improved performance.

Clearly it would be desirable to have a system in which upon observing the received codeword the next state can be inferred straightforwardly. In such a case, the presence of channel noise will not have as detrimental an effect simply because upon receiving any codeword correctly, the state will be corrected instantly and derailing will be avoided. In

what follows, we proceed to design a noisy-channel FSVQ system based on this principle.

To understand the basic idea, consider an FSVQ with $K = 2^l$ states and $R = \log_2 N$ bits/vector (assume $K \leq N$) and the following index (codeword) assignment procedure. In each state s, we group the codevectors based on the index of the next-state to which they are mapped so that group s' contains all codevectors in $\mathcal{C}_s$ that are mapped to the next-state s'. If the number of transitions to each state is the same, for each state there would be exactly N/K codevectors in each group. Let the binary codeword $\mathbf{b} = (b_1, b_2, \cdots, b_R)$ associated with each codevector in state s be such that the first l bits $\mathbf{b}_1 = (b_1, b_2, \cdots, b_l)$ are the binary representation of the codevector's group number (or next-state) s' and the next $R - l$ bits $\mathbf{b}_2 = (b_{l+1}, b_{l+2}, \cdots, b_R)$ represent the specific codevector within the group. With this construction, the channel symbol u can be written as $u \equiv (u^1, u^2)$ where u^1 and u^2 correspond, respectively, to the decimal representation of $\mathbf{b}_1$ and $\mathbf{b}_2$ where $u^1 = s'$. If we repeat the same process for all states, then regardless of the current state the first l bits of the received codeword uniquely specify the next-state, or simply, $f(u, s) = u^1$, for all $s \in \mathcal{S}$. With such an FSVQ used over a noisy channel, even if the current decoder state is different from the encoder state, the next decoder state will be correct if the current codeword is received correctly. Additionally, if the state information (i.e., u^1) is to be protected before transmission, as in NC-FSVQ1, it will also partially protect the transmitted codewords.

While the above idea seems to suggest an approach for assigning codewords to the FSVQ codevectors guaranteeing that $f(u, s) = u^1$ (merely a function of u), it assumes that the number of transitions to each state is the same. This assumption, in general, does not hold and hence the suggested codeword assignment is not possible. In what follows, we describe how to design an FSVQ system in which the next-state map satisfies a relation similar to $f(u, s) = u^1$. We will then extend this idea to a noisy-channel FSVQ.

First we construct a K-state FSVQ of rate $\log_2 K$ bits/vector with a next-state function satisfying $f(u^1, s) = u^1$. To do this, using the first three steps of the FSVQ design algorithm [8], we design an FSVQ with K states and K codevectors in each state codebook and denote the channel space by $\mathcal{N}_1$ with $\mathcal{N}_1 = \mathcal{S}$ and the state codebooks by $\mathcal{C}_s = \{\beta_1(u^1, s), u^1 \in \mathcal{N}_1\}$. The next-state function is given by

$$f(u^1, s) = \arg\min_{k \in \mathcal{S}} d(\beta_1(u^1, s), \mathbf{c}(k)),\ s \in \mathcal{S}, u^1 \in \mathcal{N}_1, \tag{20}$$

where $\{\mathbf{c}(k),\ k \in \mathcal{S}\}$ describes the state-label VQ.

For each state $s \in \mathcal{S}$, we perform the following reindexing of the codevectors in $\mathcal{C}_s$. First, we group all the codevectors in $\mathcal{C}_s$ that have the same value of $f(., s)$ (i.e., they lead to the same next state, say, $\tilde{s}$). Then in each such group, the codevector with highest probability of occurrence is reassigned the common index $\tilde{s}$. The remaining codevectors

from each group are finally reassigned, in an arbitrary fashion, a "unique" index in $\mathcal{N}_1$ which has not yet been assigned to any codevector. Note that after reassignment[5] if state s can transit to state s' then the next-state function will satisfy $f(s', s) = s'$.

In each state codebook $\mathcal{C}_s$, we retain only those codevectors whose indices satisfy $\delta(u^1, f(u^1, s)) = 1$, $\forall u^1 \in \mathcal{N}_1$. We denote the reduced version of $\mathcal{C}_s$ by $\mathcal{C}'_s$ which consists of $\sum_{u^1 \in \mathcal{N}_1} \delta(u^1, f(u^1, s))$ codevectors and use f' to denote the modified next-state function. The codewords of $\mathcal{C}'_s$ are binary tuples of length $l = \log_2 K$ with some of the l-tuples possibly not used (corresponding to the codevectors that are not retained and thereafter discarded). The original K-state FSVQ is modified to a new K-state FSVQ with state codebooks $\mathcal{C}'_s$, next-state function f' and rate $\log_2 K$ bits/vector. Note that if the codeword transmitted at time n is u_n^1, then the state at time $n+1$, s_{n+1}, is $f'(u_n^1, s_n) = u_n^1$, irrespective of s_n.

The complete design algorithm, leading to a scheme called FSVQ2, is provided next.

4.1 Design of FSVQ2 under noiseless conditions

The following describes an algorithm for the design of a K-state, R bits/vector ($2^R > K$) FSVQ with a next-state function satisfying $f(u, s) = u^1$ using a training sequence $\{\mathbf{x}_n\}$.

Algorithm:

1. Using the method described above design a K-state, $\log_2 K$ bits/vector FSVQ with a next-state function satisfying $f'(u^1, s) = u^1$; denote the FSVQ by $\{\{\mathcal{C}'_s\}_{s \in \mathcal{S}}, f'\}$.

2. Attempt to improve the state codebooks $\{\mathcal{C}'_s, s \in \mathcal{S}\}$ by encoding the training sequence using the next-state function f' and updating each codevector by replacing it by the centroid of the associated cell. Also update the state-label VQ $\mathcal{C}$ similarly.

3. Repeat steps (1) and (2) for some fixed number of iterations and choose the best case (in step (1), just update the next-state function f'). The resulting FSVQ clearly has a next-state function which satisfies $f'(u^1, s) = u^1$.

4. The FSVQ system designed so far has a rate $\log_2 K$ bits/vector. To operate at the desired rate of R bits/vector, we encode the training sequence using the FSVQ system $\{\{\mathcal{C}'_s\}_{s \in \mathcal{S}}, f'\}$ and then design[6] an LBG-VQ of rate $(R - \log_2 K)$ bits/vector for the

[5]It should also be noted that under noiseless channel conditions, the reindexing of the codevectors will not affect the performance of the FSVQ.

[6]Note that we are designing one LBG-VQ for each pair (s, u^1) in FSVQ2 (and its noisy version to be discussed in a later section) of rate $(R - \log_2 K)$ bits/vector as opposed to just one LBG-VQ of rate R bits/vector for each state in FSVQ1 (and its noisy version NC-FSVQ1); this is done to ensure that the memory requirements for FSVQ1 and FSVQ2 (and their noisy versions) are the same.

subtraining sequence associated with each pair (s, u^1).

The resulting modified FSVQ system can be described by the block diagram of Fig. 2 in which the K-state FSVQ encoder is completely specified by the next-state function f' and a mapping α_1, referred to as the primary encoder, which is described in terms of the partition $\{\mathcal{P}_{s,u^1}, s \in \mathcal{S}, u^1 \in \mathcal{N}_1\}$ according to

$$\alpha_1(\mathbf{x}, s) = u^1, \text{ if } \mathbf{x} \in \mathcal{P}_{s,u^1}. \tag{21}$$

Recall that $\mathcal{N}_1 = \mathcal{S}$. The LBG-VQ encoder in Fig. 2 is specified by a mapping α_2, referred to as the secondary encoder, which is described in terms of the partition $\{\mathcal{P}^{u^2}_{s,u^1}, s \in \mathcal{S}, u^1 \in \mathcal{N}_1, u^2 \in \mathcal{N}_2\}$ according to

$$\alpha_2(\mathbf{x}, s, u^1) = u^2, \text{ if } \mathbf{x} \in \mathcal{P}^{u^2}_{s,u^1}. \tag{22}$$

Here the channel space $\mathcal{N}_2$ is related to $\mathcal{N}$ and $\mathcal{N}_1$ by $\mathcal{N} = \mathcal{N}_1 \times \mathcal{N}_2$ and therefore the cardinality of $\mathcal{N}_2$ is $N_2 = 2^R/K$. We also have

$$\bigcup_{u^2} \mathcal{P}^{u^2}_{s,u^1} = \mathcal{P}_{s,u^1}, \; s \in \mathcal{S}, u^1 \in \mathcal{N}_1. \tag{23}$$

The FSVQ decoder and the LBG-VQ decoder can be jointly specified by f' and a mapping β_2 described by $\beta_2 : \mathcal{N}_1 \times \mathcal{N}_2 \times \mathcal{S} \rightarrow \hat{A}$. The decoder β_2 looks at the received codewords from the primary and secondary encoders and depending on its current state maps them into a reconstruction vector.

The modified FSVQ system described above, referred to hereafter as FSVQ2, is the special case of what we will call NC-FSVQ2 for the noisy channel. Since FSVQ2 is a restricted version of the FSVQ described in [8], we expect some performance loss under noiseless channel conditions. As will be seen in the simulation results section, this loss decreases as the encoding rate increases. On the other hand, when the channel is noisy, the structure of FSVQ2 allows for the design of a joint source/channel code without the need for explicit transmission of the protected state information. As a consequence, NC-FSVQ2 does not have any additional delay like NC-FSVQ1. Next we pose the problem of designing FSVQ2 in the presence of channel noise and provide a design algorithm.

4.2 Noisy-channel FSVQ2 problem statement

The block diagram in Fig. 3 illustrates the operation of FSVQ2 over noisy channels. The output of the primary encoder, u^1, is transmitted over a DMC described by a random mapping $\gamma_1 : \mathcal{N}_1 \rightarrow \mathcal{N}_1$ and the transition matrix

$$Q_1(v^1|u^1) = Pr(\gamma_1(u^1) = v^1), \; u^1, v^1 \in \mathcal{N}_1, \tag{24}$$

where v^1 is the output of γ_1. Similarly, the output of the secondary encoder, u^2, is transmitted over another DMC (assumed to be independent of γ_1) described by the random mapping $\gamma_2 : \mathcal{N}_2 \rightarrow \mathcal{N}_2$ and the transition matrix

$$Q_2(v^2|u^2) = Pr(\gamma_2(u^2) = v^2),\ u^2, v^2 \in \mathcal{N}_2, \tag{25}$$

where v^2 is the output of γ_2. For simplicity, we are considering two separate channels. In practice, the outputs of the primary and secondary encoders can be multiplexed and transmitted over the same channel.

The decoder β_2 depends on both outputs v^1 and v^2 and is described by at most K^2 different codebooks $\mathcal{C}_{s,u^1} = \{\beta_2(u^1, u^2, s), u^2 \in \mathcal{N}_2\},\ s \in \mathcal{S}, u^1 \in \mathcal{N}_1$. We will use $\mathcal{C}^{\text{sup}} = \{\mathcal{C}_{s,u^1}\}_{s\in\mathcal{S},u^1\in\mathcal{N}_1}$ to denote the collection of all the codebooks.

Our problem is to minimize the average distortion $D(\alpha_1, \alpha_2, \beta_2) = \frac{1}{L}E[d(\mathbf{X}, \hat{\mathbf{X}})]$ by appropriate design of α_1, α_2 and β_2 for given values of K and N_2.

The average distortion is given by

$$D(\alpha_1, \alpha_2, \beta_2) = \frac{1}{L}\sum_{s,u^1,u^2}\sum_{\hat{s},v^1,v^2} E[d(\mathbf{X}, \hat{\mathbf{X}})|\hat{s}, v^1, v^2, s, u^1, u^2]Pr(\hat{s}, v^1, v^2, s, u^1, u^2), \tag{26}$$

where, s and $\hat{s}$, which in general can be different, are, respectively, the primary encoder state and the decoder state. It is easily shown that

$$E[d(\mathbf{X}, \hat{\mathbf{X}})|\hat{s}, v^1, v^2, s, u^1, u^2] = \frac{\int_{\mathcal{P}^{u^2}_{s,u^1}} d(\mathbf{x}, \beta_2(v^1, v^2, \hat{s}))p(\mathbf{x})d\mathbf{x}}{Pr(u^1, u^2|s)}. \tag{27}$$

In this scheme the encoder state at any time is just the codeword at the output of the primary encoder at the previous time instant and the decoder state is the corresponding received codeword. Using this fact and assuming that u^1 (and therefore effectively s) is transmitted over γ_1 and u^2 over γ_2 and that the two channels are independent of each other, we can use a sequence of arguments similar to those in Appendix A to write the average distortion as

$$D(\alpha_1, \alpha_2, \beta_2) = \frac{1}{L}\sum_{s,u^1,u^2} Pr(s)\int_{\mathcal{P}^{u^2}_{s,u^1}} d''_s(\mathbf{x}, u^1, u^2)p(\mathbf{x})d\mathbf{x}, \tag{28}$$

where $d''_s(\mathbf{x}, u^1, u^2)$ is a modified distortion measure given by

$$d''_s(\mathbf{x}, u^1, u^2) = \sum_{\hat{s},v^1,v^2} Q_1(v^1|u^1)Q_1(\hat{s}|s)Q_2(v^2|u^2)d(\mathbf{x}, \beta_2(v^1, v^2, \hat{s})). \tag{29}$$

This modified distortion can be interpreted as the expected distortion in encoding $\mathbf{x}$ given that the primary encoder is in state s, and u^1 and u^2 are transmitted over DMC's γ_1 and γ_2; the average is taken with respect to the channels' distributions.

Using the modified distortion measure d''_s, we can obtain a generalization of (4) which describes a joint encoder

$$\alpha(\mathbf{x}, s) = \arg \min_{u^1 \in \mathcal{N}_1, u^2 \in \mathcal{N}_2} d''_s(\mathbf{x}, u^1, u^2) \equiv (u^{1*}, u^{2*}). \tag{30}$$

Clearly the encoder α defined by (30) involves no delay and therefore is not optimal for the given decoder; we made a similar statement in footnote 3. For the design of the system under consideration, to be presented shortly, we actually used a delayed-decision encoding.

For a given encoder α (i.e., for a given partition $\{\mathcal{P}^{u^2}_{s,u^1}\}$), for each $\hat{s}, v^1, v^2$, the optimum decoder β_2 is given by

$$\beta^*_2(v^1, v^2, \hat{s}) = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^L} \sum_{s,u^1,u^2} Pr(s) Q_1(v^1|u^1) Q_1(\hat{s}|s) Q_2(v^2|u^2) \int_{\mathcal{P}^{u^2}_{s,u^1}} d(\mathbf{x}, \hat{\mathbf{x}}) p(\mathbf{x}) d\mathbf{x}. \tag{31}$$

Before providing the design algorithm, in what follows we present simplifications of the modified distortion in (29) and the optimum decoder in (31) for the squared-error criterion.

For $d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||^2$, we have

$$d''_s(\mathbf{x}, u^1, u^2) = \sum_{\hat{s}, v^1, v^2} Q_1(v^1|u^1) Q_1(\hat{s}|s) Q_2(v^2|u^2) ||\mathbf{x} - \beta_2(v^1, v^2, \hat{s})||^2. \tag{32}$$

Upon defining

$$\mathbf{y}_{s,u^1,u^2} \triangleq \sum_{\hat{s}, v^1, v^2} Q_1(v^1|u^1) Q_1(\hat{s}|s) Q_2(v^2|u^2) \beta_2(v^1, v^2, \hat{s}), \tag{33}$$

and

$$\eta_{s,u^1,u^2} \triangleq \sum_{\hat{s}, v^1, v^2} Q_1(v^1|u^1) Q_1(\hat{s}|s) Q_2(v^2|u^2) ||\beta_2(v^1, v^2, \hat{s})||^2, \tag{34}$$

the modified distortion measure can be expressed as [19]

$$d''_s(\mathbf{x}, u^1, u^2) = ||\mathbf{x}||^2 - 2\langle \mathbf{x}, \mathbf{y}_{s,u^1,u^2} \rangle + \eta_{s,u^1,u^2}. \tag{35}$$

For a given $\mathcal{C}^{\text{sup}}$, the terms $\mathbf{y}_{s,u^1,u^2}$ and η_{s,u^1,u^2} can be precomputed and stored for all $s \in \mathcal{S}, u^1 \in \mathcal{N}_1, u^2 \in \mathcal{N}_2$.

Also, for the squared-error distortion measure, it can be easily seen that the optimum decoder β_2 for a given partition $\{\mathcal{P}^{u^2}_{s,u^1}\}$ satisfies

$$\begin{aligned} \beta^*_2(v^1, v^2, \hat{s}) &= E[\mathbf{X}|\hat{s}, v^1, v^2] \\ &= \frac{\sum_{s,u^1,u^2} Q_1(\hat{s}|s) Q_1(v^1|u^1) Q_2(v^2|u^2) \int_{\mathcal{P}^{u^2}_{s,u^1}} \mathbf{x} p(\mathbf{x}) d\mathbf{x}}{\sum_{s,u^1,u^2} Q_1(\hat{s}|s) Q_1(v^1|u^1) Q_2(v^2|u^2) \int_{\mathcal{P}^{u^2}_{s,u^1}} p(\mathbf{x}) d\mathbf{x}}. \end{aligned} \tag{36}$$

The above arguments assume that the source distribution is known. In practice, we use the training sequence approach and replace integrals by appropriate sums. For the squared-error criterion, a design algorithm for a K-state NC-FSVQ2 of rate R bits/vector is provided below.

4.3 Design algorithm for NC-FSVQ2

0. Set $m = 0$ (iteration index) and $D^{(0)} = \infty$. Using the algorithm described in Section 4.1 design an FSVQ2, to be used as the initial NC-FSVQ2, with encoder partition $\{\mathcal{P}_{s,u^1}^{u^2(0)}\}$, codebook $\mathcal{C}^{\text{sup}(0)}$ and f' as the next-state function. In the following steps, keep the next-state function unchanged.

1. Set $m = m + 1$.

2. Compute $\mathbf{y}_{s,u^1,u^2}$ and η_{s,u^1,u^2} as given by (33) and (34) using $\mathcal{C}^{\text{sup}(m-1)}$.

3. Consider the trellis corresponding to the current NC-FSVQ2 with the modified distortion measure of (35) as the branch metric. Encode the training sequence using the Viterbi algorithm. This will result in a partition $\{\mathcal{P}_{s,u^1}^{u^2}\}$, where $\mathcal{P}_{s,u^1}^{u^2}$ is the set of all training vectors that correspond to the encoder state s and output pair (u^1, u^2).

4. Compute the optimum codebook $\mathcal{C}^{\text{sup}*}$ for the partition in Step (3) and using (36).

5. Compute the average distortion D using $\mathcal{C}^{\text{sup}*}$, $\{\mathcal{P}_{s,u^1}^{u^2}\}$ and Eqs. (28) and (35). Set $D^{(m)} = D$, $\mathcal{C}^{\text{sup}(m)} = \mathcal{C}^{\text{sup}*}$, $\mathcal{P}_{s,u^1}^{u^2(m)} = \mathcal{P}_{s,u^1}^{u^2}$.

6. If $m < m_{max}$ (prescribed maximum number of iterations), go to Step (1). Otherwise, stop with the encoder described by the final codebook and partition.

While in the above design algorithm a delayed-decision encoding in assumed with a joint selection of (u^1, u^2), the actual encoder used in our simulations is a simplified version which selects u^1 and u^2 sequentially and introduces no delay. Precisely, we define

$$\alpha_1(\mathbf{x}, s) = u^1, \text{ if } \mathbf{x} \in \mathcal{P}_{s,u^1}, \tag{37}$$

$$\alpha_2(\mathbf{x}, s, u^1) = u^2, \text{ if } \mathbf{x} \in \mathcal{P}_{s,u^1}^{u^2}, \tag{38}$$

where $\mathcal{P}_{s,u^1} = \bigcup_{u^2} \mathcal{P}_{s,u^1}^{u^2}$. Associated with the final partition $\mathcal{P}_{s,u^1}$, we design a channel-optimized extension of $\{\mathcal{C}_s'\}$ which is used, as in [18], to characterize $\mathcal{P}_{s,u^1}$ in the encoding process. This modification gives us the final NC-FSVQ2 for which we have reported our simulation results. We must mention that due to the two-stage structure of the encoding operation, the computational complexity is comparable to that of a two-stage VQ with K codevectors in the first stage and $2^R/K$ codevectors in the second stage.

5 Simulation Results

In all the experiments we have considered a binary symmetric channel with bit error rates $\epsilon = 0.0, 0.005, 0.01, 0.05, 0.1$. Performances of NC-FSVQ1 (FSVQ1 when $\epsilon = 0.0$) and NC-FSVQ2 (FSVQ2 when $\epsilon = 0.0$) are presented in this section for the Gauss-Markov (G-M) source and the speech LSP parameters. First, we provide the results for the G-M source and then consider the LSP parameters.

5.1 Gauss-Markov source

We used 200,000 vectors of dimension 4 from the G-M source with $\rho = 0.9$ as the training sequence. The test sequence consisted of 100,000 4-dimensional vectors from the same source not included in the training sequence. The distortion measure used was squared-error; the performance results are reported in signal-to-noise ratio (SNR) in dB.

5.1.1 Performance of FSVQ1

For different bit rates (denoted by b bits/vector) an 8-state FSVQ1 was designed. The performance results are tabulated in Table 1. For comparison, LBG-VQ performance results are also included. These results, which are in agreement with those of [8], indicate that FSVQ1 outperforms LBG-VQ at all rates by at least 1 dB.

5.1.2 Performance of NC-FSVQ1 over noisy channels

FSVQ1 was simulated in the presence of channel noise and the results are presented in Table 2. Severe degradation in the performance of FSVQ1 is observed. For very noisy channels, larger bit rates often result in lower SNR.

The SNR performances of NC-FSVQ1 and CO-VQ are displayed in Fig. 4. Additional results on NC-FSVQ2 are also included in Fig. 4 and will be discussed later. When $0.005 \leq \epsilon \leq 0.01$ ($0.05 \leq \epsilon \leq 0.1$), a rate 1/2 (1/3) convolutional code was used to protect the state index and $p = 6$ ($p = 3$). The bit rates in Fig. 4 include the overhead information associated with the transmission of the protected encoder state. When an 8-state NC-FSVQ1 is used, the overhead is given by $3/(pr_c)$, bits/vector, where r_c is the rate of the channel code. These results indicate that NC-FSVQ1 and CO-VQ perform similarly when $\epsilon \leq 0.05$. At $\epsilon = 0.1$, NC-FSVQ1 outperforms CO-VQ by 0.4-0.9 dB. However, as compared to FSVQ1, NC-FSVQ1 performs significantly better for all $\epsilon > 0.0$.

5.1.3 Performance of FSVQ2

For different bit rates, an 8-state FSVQ2 was designed using the algorithm of Section 4.1; Table 1 includes the performance results. Clearly, FSVQ2 outperforms LBG-VQ by 0.56-0.85 dB depending on the bit rate. As compared to FSVQ1 (FSVQ [8]), the performance of FSVQ2 is inferior by about 0.5 dB due the structure imposed on the system. FSVQ2 was also designed with 16 states but the improvement over the 8-state version was only marginal.

5.1.4 Performance of NC-FSVQ2 over noisy channels

For different values of b and ϵ, NC-FSVQ2 was designed using the design algorithm of Section 4.3. The SNR performance of the 8-state NC-FSVQ2 is included in Fig. 4 which shows that NC-FSVQ2 outperforms CO-VQ for all values of b and ϵ considered. At $\epsilon = 0.005$, the SNR gain over CO-VQ at the same bit rate is about 0.4-0.8 dB. As ϵ increases, the gain increases; at $\epsilon = 0.1$, the gain is about 0.7-1.0 dB. NC-FSVQ2 performs equal to or better than NC-FSVQ1 in all cases considered. In terms of computational complexity, NC-FSVQ2 has the complexity of a 2-stage VQ, while NC-FSVQ1 (operating at the same rate) has the much higher complexity of a full-searched VQ. In addition, there is a decoding delay involved in NC-FSVQ1.

5.2 Speech LSP parameters

The speech database used for training consisted of 120 minutes of speech (from the TIMIT database [36]) sampled at 8 KHz and uttered by several male and female speakers. A 10th-order LPC analysis based on the standard autocorrelation method was performed every 22.5 msec with a 30 msec analysis window. Thus we had 320,000 LSP vectors in the training sequence. The test sequence consisted of 2,261 vectors not in the training sequence (also used in [37]).

The LSP source exhibits relatively high intra- and inter-frame correlation and is therefore a good candidate for FSVQ type systems that can efficiently exploit these correlations. The performance of the LSP quantization system is expressed in terms of the average spectral distortion (SD) given by

$$\text{SD} = \frac{1}{T}\sum_{n=1}^{T}[\int_{-\pi}^{\pi}(10\log S_n(\omega) - 10\log \hat{S}_n(\omega))^2\frac{d\omega}{2\pi}]^{\frac{1}{2}}, \text{ (dB)}. \tag{39}$$

Here, $S_n(\omega)$ and $\hat{S}_n(\omega)$ are the original and quantized spectra, respectively, associated with

the n^{th} frame and T is the number of frames. The average spectral distortion is a useful measure of performance in LSP quantization schemes [22].

Earlier work on memoryless VQ of LSP parameters [25] suggests that for transparent quantization[7] of LSP parameters, an encoding rate in the neighborhood of 24 bits/frame is required. To achieve this kind of rate, FSVQ state codebooks will need to have 2^{24} codevectors which is prohibitively large. To reduce the complexity, we split each 10-dimensional LSP vector into 3 subvectors (dimensions 3, 3 and 4) and design an FSVQ of rate $R/3$ for each subvector (R is the overall rate in bits/frame); the three subvectors are encoded separately using different FSVQs[8]. Performance results of such a system are provided next and it is shown that noticeable gains are obtained over the case where a memoryless LBG-VQ is used to encode each subvector (such a system will be called a split-VQ system).

5.2.1 Performance of FSVQ1

An FSVQ1 was designed for each of the three LSP subvectors, resulting in a quantization system referred to as split-FSVQ1. The squared-error distortion measure was used for the design but encoding was done using the inverse harmonic mean distortion measure introduced in [37]. Table 3 summarizes the performance results of split-FSVQ1 (8 states for each subvector) and split-VQ (using the same 3 subvectors) at various bit rates. The results indicate that split-FSVQ1 yields a saving of approximately 3 bits/frame over the split-VQ system and achieves the 1 dB average spectral distortion (SD) at 24 bits/frame; at this rate the outlier rate (OL) is 0.75%. The performance of split-FSVQ1 can be improved by increasing the number of states; we found that a 16-state FSVQ1 (for each subvector) achieves transparent quantization at 23 bits/frame.

5.2.2 Performance of NC-FSVQ1 over noisy channels

The split-FSVQ1 system was simulated in the presence of channel noise and the results are tabulated in Table 4. As expected, the performance degradation is severe. With 24 bits/frame, under noiseless conditions, split-FSVQ1 achieves a 1 dB average SD but when

[7]The quantization is said to be transparent if the resulting average spectral distortion is less than 1 dB and the fraction of frames with spectral distortion over 2 dB (called the outliers) is less than 2% with no frame having spectral distortion greater than 4 dB.

[8]In [25], the LSP vectors are split into 2 subvectors and a 1 dB average spectral distortion is reported with a 12-bit VQ for each subvector. We could not, however, repeat this result with 2 subvectors. We suspect that use of the modified covariance LPC analysis in [25] (as opposed to the standard autocorrelation method used here) and the different database used could be two possible reasons for this discrepancy. An additional reason may be that [25] considers the telephone band (200-3300 Hz) frequencies for computing SD as opposed to the full-band (0-4000 Hz) used in this paper. Similar findings are reported in [26].

$\epsilon = 0.01$, the average SD rises up to 2.97 dB.

The performance results of NC-FSVQ1[9] (designed for $\epsilon > 0.0$) are provided in Fig. 5 for $p = 4$ and $p = 8$. To protect the encoder state information, we found that a rate 1/2 (1/3) convolutional code with constraint length 4 is sufficient when $0.005 \leq \epsilon \leq 0.01$ ($0.05 \leq \epsilon \leq 0.1$). The decoder is implemented using the Viterbi algorithm and the estimated codeword is released at the end of each frame. In Fig. 5 the bit rates for NC-FSVQ1 include the overhead associated with the transmission of the protected encoder state sequence. When 3 subvectors and 8-state encoders are used, the overhead is easily computed to be $(3 \times 3)/(pr_c)$, bits/frame, where r_c is the rate of the channel code used.

For comparison, the results for the split CO-VQ system (each subvector is quantized using a CO-VQ) are also included in Fig. 5. It can be seen that with $p = 4$, NC-FSVQ1 performs close to split CO-VQ when $\epsilon = 0.005$ but for larger ϵ it outperforms split CO-VQ with a saving of about 1.5 bits/frame at $\epsilon = 0.01$ and 4.25 bits/frame at $\epsilon = 0.1$. For $p = 8$ (larger delay), the gains are even larger. Also comparison with the MSVQ-based scheme of [19], [37] (see Fig. 5) shows significant gains at larger values of ϵ (> 0.01) and comparable or better performance at $\epsilon = 0.005$. At $\epsilon = 0.005$, NC-FSVQ1 achieves a 1 dB average SD with 31.5 bits/frame ($p = 4$) and 29.25 bits/frame ($p = 8$).

5.2.3 Performance of FSVQ2

As in split-FSVQ1, an FSVQ2 was designed for each LSP subvector, resulting in an encoder called split-FSVQ2. Again, squared-error distortion was used for the design, but encoding was done using the inverse harmonic mean distortion measure. Table 3 includes the results of the 8-state split-FSVQ2 system. The results show that as compared to the split-VQ system, split-FSVQ2 achieves a saving of approximately 2 bits/frame (1 bit/frame less than split-FSVQ1 for higher values of b). Split-FSVQ2 achieves close to 1 dB average SD at 25 bits/frame with an outlier rate of 1.1%. The performance of split-FSVQ2 improves as the number of states increases; a 16-state split-FSVQ2 achieves transparent quantization at 24 bits/frame.

5.2.4 Performance of NC-FSVQ2 over noisy channels

The performance results of NC-FSVQ2[10] (for $\epsilon > 0.0$) are illustrated in Fig. 6 for the 8- and 16-state cases. NC-FSVQ2 is designed for each of the three subvectors using the

[9] As in the noiseless case, we are using a different NC-FSVQ1 for each subvector, but for brevity we use the terminology NC-FSVQ1 instead of split-NC-FSVQ1.

[10] Again, as in the noiseless case, we are using a different NC-FSVQ2 for each subvector, but for brevity we use the terminology NC-FSVQ2 instead of split-NC-FSVQ2.

design algorithm of Section 4.3. For comparison purposes, the results of split CO-VQ are also plotted in Fig. 6. It can be seen that 8-state NC-FSVQ2 outperforms split CO-VQ by over 1 bit/frame when $\epsilon = 0.005$; this gain increases to over 3 bits/frame at $\epsilon = 0.1$. Comparison of the 8-state and 16-state NC-FSVQ2 shows that unlike in the case of G-M source, for speech LSP parameters the performance improvement with the increase in the number of states is noticeable for higher values of ϵ. When $\epsilon = 0.05$, the improvement in going from 8 to 16 states is about 0.5 bits/frame, while for $\epsilon = 0.1$, the gain can be as high as 1.5 bits/frame. The gain is insignificant for $\epsilon \leq 0.01$.

5.2.5 NC-FSVQ1 versus NC-FSVQ2

Comparison of Figs. 5 and 6 shows that at low values of ϵ (< 0.01), NC-FSVQ2 performs better than NC-FSVQ1 and vice versa for high values of ϵ (> 0.05), with comparable results for $0.01 \leq \epsilon \leq 0.05$. However, NC-FSVQ2 does not suffer from any delay. In addition, the decoder for NC-FSVQ2 is simpler and there is no need for a separate channel code in NC-FSVQ2. The memory requirements are the same for NC-FSVQ1 and NC-FSVQ2 operating at the same bit-rate and with the same number of states.

6 Summary and Conclusions

In this paper, we have considered the quantization of sources with memory using FSVQ and memoryless VQ encoders under noiseless as well as noisy channel conditions. Two sources, namely the Gauss-Markov source with $\rho = 0.9$ and speech LSP parameters were considered. Under noiseless channel conditions, comparisons were made between FSVQ1, FSVQ2 and LBG-VQ systems for both sources. It was concluded that in all cases both FSVQ systems outperform LBG-VQ with FSVQ1 performing slightly better than FSVQ2. For noisy channels, FSVQ1 collapsed for all values of $\epsilon > 0.0$, while FSVQ2 collapsed for $\epsilon > 0.01$ making it necessary to redesign these FSVQ systems by taking the channel noise into account. Two systems, namely NC-FSVQ1 (noisy version of FSVQ1) and NC-FSVQ2 (noisy version of FSVQ2) were developed. For both sources considered, the noisy-channel FSVQ systems offered higher robustness than their noiseless channel counterparts. In particular, when LSP parameters were quantized using NC-FSVQ1, the saving over split CO-VQ was 5 bits/frame for very noisy channels with a decoding delay of 8 frames; NC-FSVQ2 achieved a saving of up to 3 bits/frame (for the 8-state case) over split CO-VQ.

Based on our results, it appears that the proposed noisy-channel FSVQs are good candidates for speech encoding systems such as in mobile communication where channel

noise is an important factor. If a delay can be tolerated, then NC-FSVQ1 is a better choice especially for the highly noisy case (ϵ close to 0.1); when delay is objectionable, NC-FSVQ2 is a better candidate for all levels of channel noise.

In the NC-FSVQ1 and NC-FSVQ2 schemes, each of the three subvectors of each frame are treated independently; we can expect performance improvements if we exploit the memory present between the three subvectors. Also, if we split the LSP vectors into just two subvectors, then the overhead needed in NC-FSVQ1 will be reduced; in addition, the intraframe correlation will be better utilized than in the 3 split case.

We must point out that if delay is not an issue, using a delayed-decision encoding in NC-FSVQ1 and NC-FSVQ2, as in trellis coding systems, will lead to improved performance. Such a delayed-decision FSVQ encoder was considered in [35] for the noiseless channel case and performance improvements were observed over the ordinary FSVQ.

Appendix A
Proof of Theorem 1

We wish to show that the average distortion is given by

$$D(\alpha,\beta,f)=\frac{1}{L}\sum_{s,u}Pr(s)\int_{\mathcal{P}_{s,u}}\{\sum_{\hat{s},v}Q(v|u)Pr(\hat{s}|s)d(\mathbf{x},\beta(v,\hat{s}))\}p(\mathbf{x})d\mathbf{x}. \tag{A.1}$$

We note that

$$D(\alpha,\beta,f)=\frac{1}{L}\sum_{s,u}\sum_{\hat{s},v}E[d(\mathbf{X},\hat{\mathbf{X}})|s,\hat{s},u,v]Pr(s,\hat{s},u,v), \tag{A.2}$$

and $Pr(s,\hat{s},u,v)$ can be written as

$$Pr(s,\hat{s},u,v)=Pr(v|s,u)Pr(\hat{s}|s,u,v)Pr(s,u). \tag{A.3}$$

We know that for a given s_0, the encoder state at time n, s_n, depends only on $\mathbf{u}_0^{n-1}\equiv(u_0,u_1,\ldots,u_{n-1})$ and the decoder state $\hat{s}_n$ depends only on $\mathbf{v}_0^{n-1}\equiv(v_0,v_1,\ldots,v_{n-1})$. Then for a memoryless channel we have

$$Pr(v_n|s_n,u_n)=Pr(v_n|u_n)=Q(v_n|u_n). \tag{A.4}$$

Also,

$$Pr(\hat{s}_n|s_n,u_n,v_n)=Pr(\hat{s}_n|s_n). \tag{A.5}$$

Equation (A.5) follows from the fact that the channel is memoryless and s_n and $\hat{s}_n$ do not depend on u_n and v_n and are completely determined from $\mathbf{u}_0^{n-1}$ and $\mathbf{v}_0^{n-1}$, respectively. Thus

$$Pr(s,\hat{s},u,v)=Q(v|u)Pr(\hat{s}|s)Pr(s,u). \tag{A.6}$$

Next consider the simplification of $E[d(\mathbf{X},\hat{\mathbf{X}})|s,\hat{s},u,v]$. Using the definition of expectation,

$$E[d(\mathbf{X},\hat{\mathbf{X}})|s,\hat{s},u,v] = \int_{\mathbb{R}^L} p(\mathbf{x}|s,\hat{s},u,v)d(\mathbf{x},\beta(v,\hat{s}))d\mathbf{x},$$
$$= \int_{\mathbb{R}^L} p(\mathbf{x}|s,u)d(\mathbf{x},\beta(v,\hat{s}))d\mathbf{x}. \qquad (A.7)$$

In obtaining (A.7), we used $p(\mathbf{x}|s,\hat{s},u,v) = p(\mathbf{x}|s,u)$, which is a direct consequence of the fact that given s and u, $\mathbf{x}$ is independent of $\hat{s}$ and v. Finally, we note that

$$p(\mathbf{x}|s,u) = \begin{cases} \frac{p(\mathbf{x})}{Pr(u|s)} & \text{if } \mathbf{x} \in \mathcal{P}_{s,u}, \\ 0 & \text{otherwise.} \end{cases} \qquad (A.8)$$

Combining (A.2), (A.6), (A.7) and (A.8) yields (A.1).

References

[1] Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84-95, Jan. 1980.

[2] R. M. Gray, "Vector Quantization," *IEEE ASSP Mag.*, pp. 4-29, Apr. 1984.

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

[4] H. Abut, R. M. Gray and G. Rebolledo,"Vector Quantization of Speech and Speech-like Waveforms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 423-435, Jun. 1982.

[5] A. Buzo, A. H. Gray, R. M. Gray and J. D. Markel, "Speech Coding Based upon Vector Quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 562-574, Oct. 1980.

[6] N. M. Nasrabadi and R. A. King, "Image Coding Using Vector Quantization: A Review," *IEEE Trans. Commun.*, vol. 36, pp. 957-971, Aug. 1988.

[7] C. E. Shannon, "Coding Theorems for a Discrete Source with Fidelity Criterion," *IRE National Convention Record*, Part 4, pp. 142-163, 1959.

[8] 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.

[9] M. O. Dunham and R. M. Gray, "An Algorithm for the Design of Labeled-Transition Finite-State Vector Quantizers," *IEEE Trans. Commun.*, vol. COM-33, pp. 83-89, Jan. 1985.

[10] N. T. Gaarder and D. Slapian, "On Optimal Finite-State Digital Transmission Systems," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 167-186, March 1982.

[11] R. Aravind and A. Gersho, "Low-Rate Image Coding with Finite-State Vector Quantization," *Proc. ICASSP*, pp. 137-140, Apr. 1986.

[12] H. H. Shen and R. L. Baker, "A Finite State/Frame Difference Interpolative Vector Quantizer for Low Rate Image Sequence Coding," *Proc. ICASSP*, pp. 1188-1191, Apr. 1988.

[13] T. Kim, "New Finite State Vector Quantizers for Images," *Proc. ICASSP*, pp. 1180-1183, Apr. 1988.

[14] Y. Hussain and N. Farvardin, "Variable-Rate Finite-State Vector Quantization and Applications to Speech and Image Coding," *IEEE Trans. Speech and Audio Processing*, vol. 1, pp. 25-38, January 1993.

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

[16] N. Farvardin and V. Vaishampayan, "Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding," *IEEE Trans. Inform. Theory*, vol. 33, pp. 827-838, Nov. 1987.

[17] H. Kumazawa, M. Kasahara and T. Namekawa, "A Construction of Vector Quantizers for Noisy Channels," *Electron. Eng. Japan*, vol. 67-B, no. 4, pp. 39-47, 1984.

[18] N. Farvardin and V. Vaishampayan, "On the Performance and Complexity of Channel-Optimized Vector Quantizers," *IEEE Trans. Inform. Theory*, vol. IT-37, pp. 155-160, Jan. 1991.

[19] N. Phamdo, N. Farvardin and T. Moriya, "A Unified Approach to Tree-Structured and Multi-Stage Vector Quantization for Noisy Channels," *IEEE Trans. Inform. Theory,* vol. 39, pp. 835-850, May 1993.

[20] J.G. Dunham and R. M. Gray, "Joint Source and Noisy Channel Trellis Encoding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 516-519, Jul. 1981.

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

[22] N. Sugamura and F. Itakura, "Speech Data Compression by LSP Speech Analysis-Synthesis Technique," *IECE Trans.*, vol. J64-A, pp. 599-605, Aug. 1981 (in Japanese).

[23] N. Sugamura and N. Farvardin, "Quantizer Design in LSP Speech Analysis-Synthesis," *IEEE Journ. Selected Areas Commun.*, vol. 6, pp. 432-440, Feb. 1988.

[24] F. K. Soong and B. H. Juang, "Optimal Quantization of LSP Parameters," *IEEE Trans. Speech and Audio Processing*, vol. 1, pp. 15-24, January 1993.

[25] K. K. Paliwal and B. S. Atal, "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame," *IEEE Trans. Speech and Audio Processing*, vol. 1, pp. 3-14, January 1993.

[26] J. Pan and T. Fischer, "Vector Quantization of Speech Line Spectrum Pair Parameters and Reflection Coefficients," submitted to *IEEE Trans. Speech and Audio Proc.*, June 1993.

[27] D. P. Kemp, J.S. Collura and T. E. Tremain, "Multi-frame coding of LPC parameters at 600-800 bps," *Proc. ICASSP,* pp. 609-612. Toronto, 1991.

[28] M. S. Schroeder and B. S. Atal, "Code-Excited Linear Prediction (CELP): High Quality Speech at Very Low Bit Rates," *Proc. ICASSP*, pp. 937-940, Tampa, FL, Mar. 1985.

[29] G. Davidson and A. Gersho, "Complexity Reduction Methods for Vector Excitation Coding," *Proc. ICASSP*, pp. 3055-3058, Tokyo, Japan, Apr. 1986.

[30] T. Moriya and M. Honda, "Transform Coding of Speech Using a Weighted Vector Quantizer," *IEEE Journ. Selected Areas Commun.*, vol. 6, pp. 425-431, Feb. 1988.

[31] J.P. Campbell, V.C. Welch and T.E. Tremain, "An Expandable Error-Protected 4800 BPS CELP Coder (U.S. Federal Standard 4800 BPS Voice Coder)," *Proc. ICASSP*, pp. 735-738, Glasgow, Scotland, May 1989.

[32] M.J. Mclaughlin and P.D. Rasky, "Speech and Channel Coding for Digital Land-Mobile Radio," *IEEE Journ. Selected Areas Commun.*, vol. 6, pp. 332-344, Feb. 1988.

[33] G.D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, Mar. 1973.

[34] Y. Hussain, *Design and Performance Evaluation of a Class of Finite-State Vector Quantizers*, Ph.D. Dissertation, Electrical Engineering Department, Univ. of Maryland, College Park, MD, 1992.

[35] C. Bei, and R. M. Gray, "Simulation of Vector Trellis Encoding Systems," *IEEE Trans. Commun.*, vol. COM-34, pp. 214-218, Mar. 1986.

[36] NTIS Federal Computer Products Center, "DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus," U.S. Department of Commerce.

[37] N. Phamdo, N. Farvardin and T. Moriya, "Combined Source-Channel Coding of LSP Parameters Using Multi-Stage Vector Quantization," *Research Meeting of the Institute of Electronics, Information and Communication Engineers* (IEICE, Japan), vol. SP90-52, DSP90-83, pp. 63-70, Nagoya, Japan, Oct. 1990.

b	LBG-VQ	FSVQ1	FSVQ2
1	3.57	5.26	-
2	6.53	8.14	-
3	8.24	9.97	9.07
4	10.15	11.31	10.84
5	11.60	12.68	12.36
6	12.98	14.04	13.63
7	14.31	15.34	14.92
8	15.69	-	16.25

Table 1: Performance of LBG-VQ, FSVQ1 and FSVQ2 under noiseless channel conditions at various bits rates (bits/vector); Gauss-Markov source with $\rho = 0.9$; $L = 4$; $K = 8$.

b	$\epsilon = 0.000$	$\epsilon = 0.005$	$\epsilon = 0.010$	$\epsilon = 0.050$	$\epsilon = 0.100$
1	5.26	4.21	3.30	0.40	-0.87
2	8.14	4.65	2.99	-0.79	-1.84
3	9.97	5.40	3.38	-0.93	-2.07
4	11.31	4.30	1.98	-1.67	-2.53
5	12.68	4.99	2.66	-1.50	-2.48
6	14.04	4.78	2.36	-1.74	-2.62
7	15.34	3.74	1.54	-2.27	-2.94

Table 2: Performance of FSVQ1 for various levels of channel noise and different bit rates (bits/vector); Gauss-Markov source with $\rho = 0.9$; $L = 4$; $K = 8$.

	Split-VQ		Split-FSVQ1		Split-FSVQ2	
b	SD	OL%	SD	OL%	SD	OL%
21	1.51	9.69	1.27	3.80	1.39	6.28
22	1.43	6.59	1.17	2.60	1.30	4.80
23	1.33	3.80	1.09	1.60	1.20	2.73
24	1.25	1.95	1.02	0.75	1.10	1.78
25	1.18	1.37	0.97	0.53	1.02	1.10
26	1.11	0.57	0.91	0.44	0.95	0.70
27	1.03	0.18	0.86	0.35	0.89	0.51

Table 3: Performance of split-VQ, split-FSVQ1 and split-FSVQ2 under noiseless channel conditions at various bit rates (bits/frame); speech LSP parameters; $K = 8$.

	$\epsilon = 0.005$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$
b	SD	SD	SD	SD
21	2.36	3.23	6.50	7.66
22	2.24	3.08	6.38	7.81
23	2.19	3.13	6.46	7.85
24	1.90	2.97	6.40	7.66
25	1.77	2.86	6.26	7.60
26	1.80	2.87	6.11	7.52
27	1.79	2.70	6.03	7.41

Table 4: Performance of split-FSVQ1 for various levels of channel noise and bit rates (bits/frame); speech LSP parameters; $K = 8$.

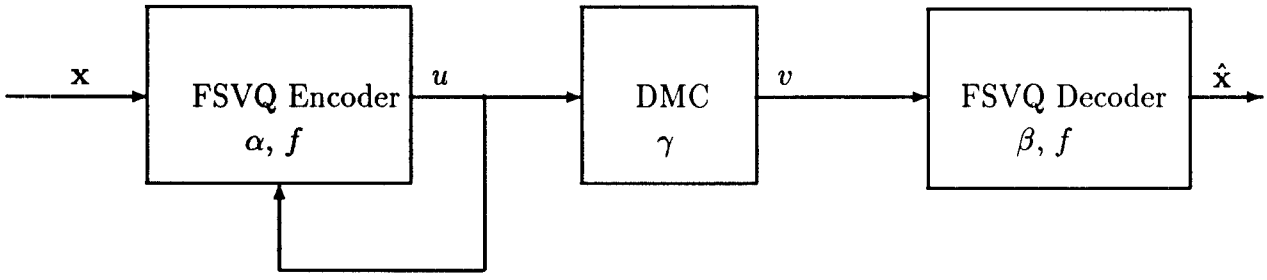
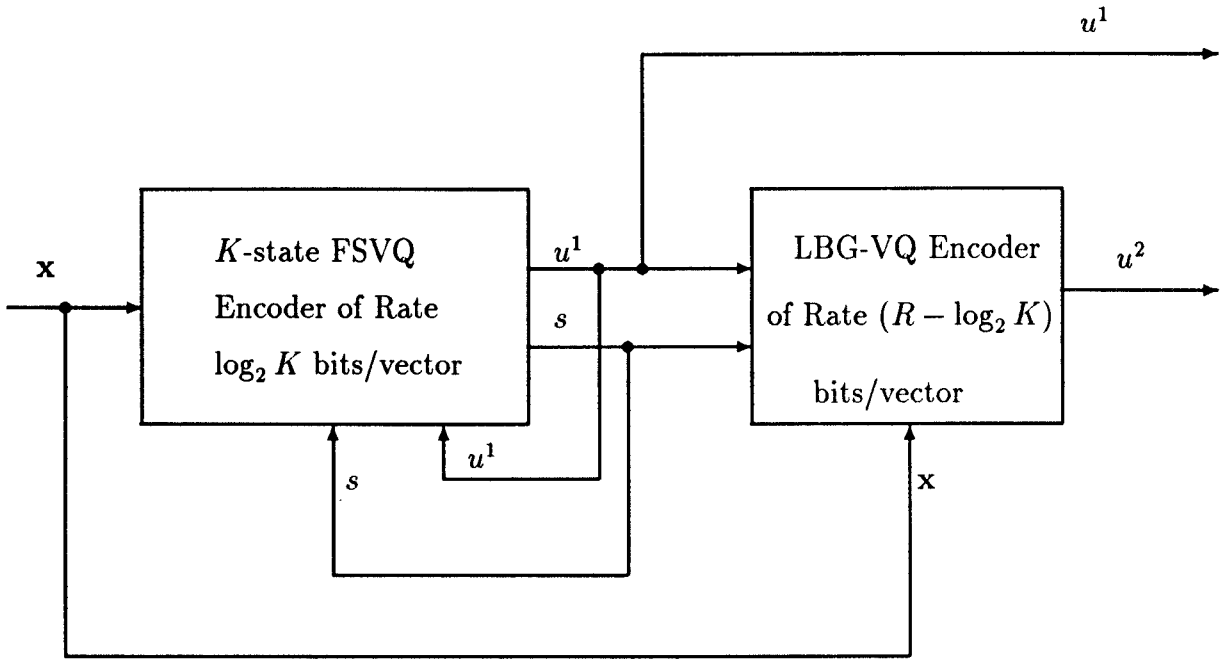
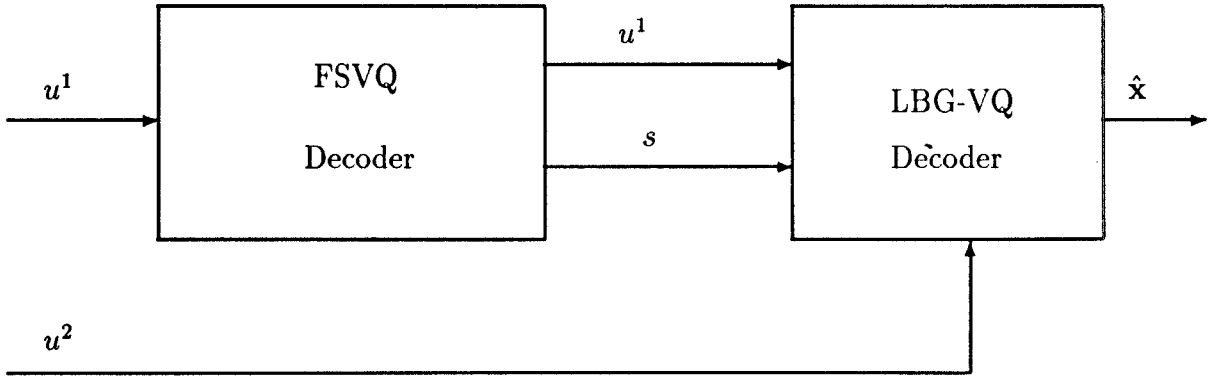


Figure 1: Block diagram of an FSVQ system with transmission over a noisy channel.



BLOCK ENCODER



BLOCK DECODER

Figure 2: Block diagram of the encoder and the decoder of the FSVQ2 system.

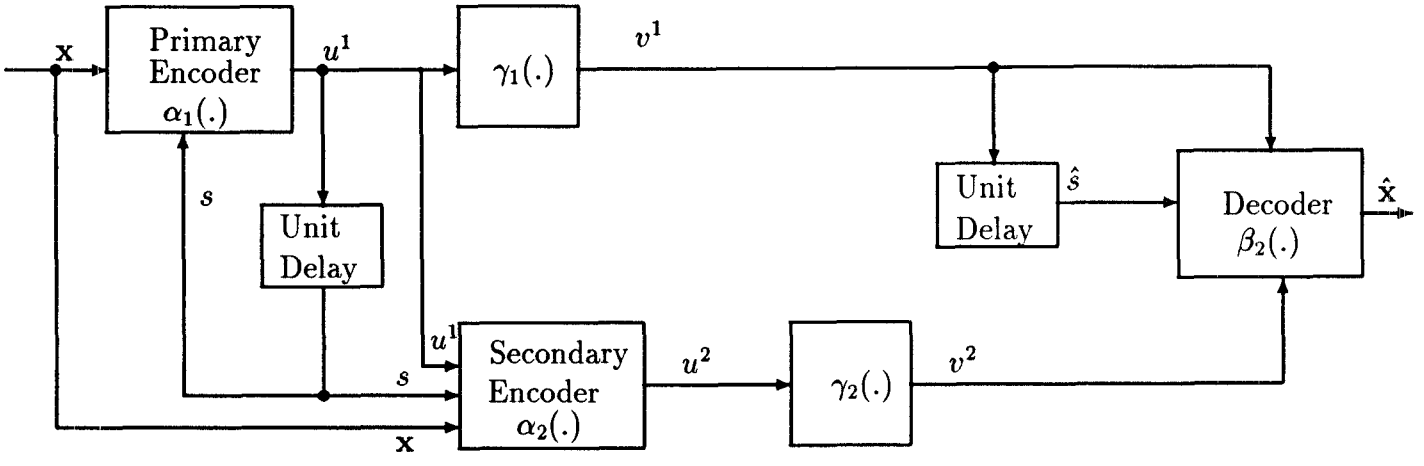


Figure 3: Block diagram of NC-FSVQ2 system with transmission over a noisy channel.

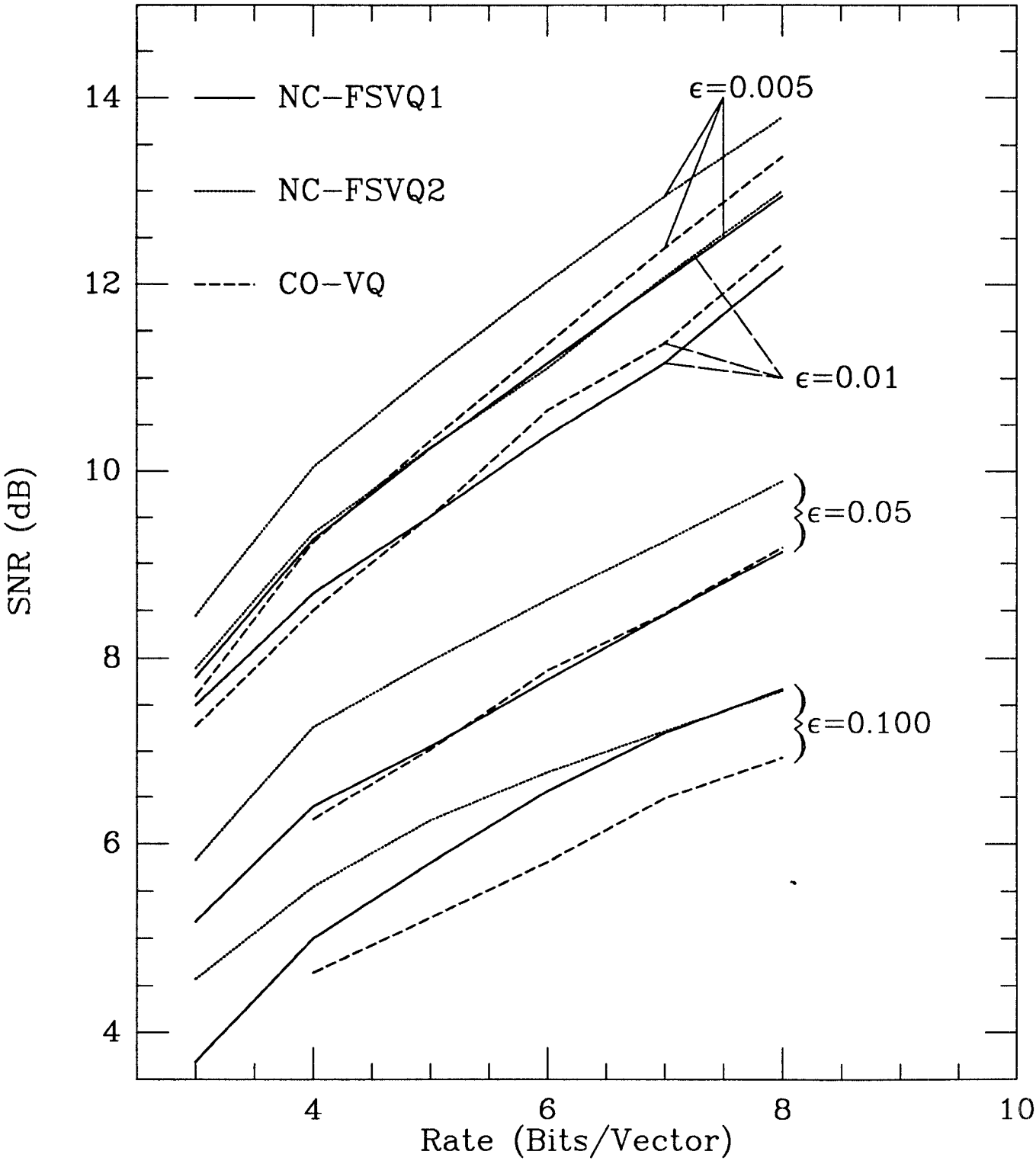


Figure 4: Performance of NC-FSVQ1, NC-FSVQ2 and CO-VQ for various levels of channel noise and bit rates; $p = 6$, when $\epsilon = 0.005$ and 0.010, while $p = 3$, when $\epsilon = 0.050$ and 0.100; Gauss-Markov source with $\rho = 0.9$; $L = 4$; $K = 8$.

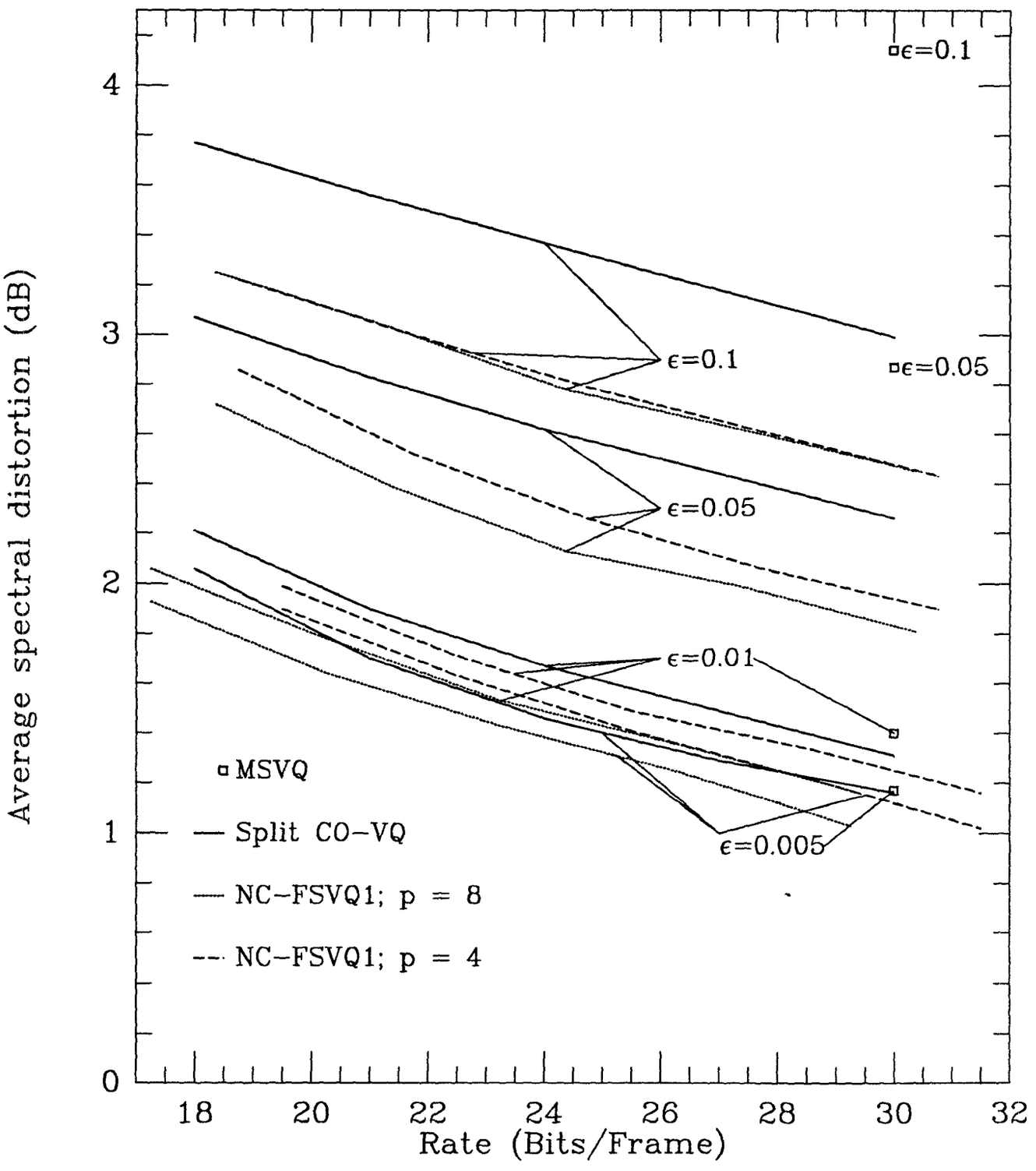


Figure 5: Performance of NC-FSVQ1 and split CO-VQ for various levels of channel noise and bit rates; speech LSP parameters; $K = 8$; $p = 4$ and $p = 8$, MSVQ results are taken from [37].

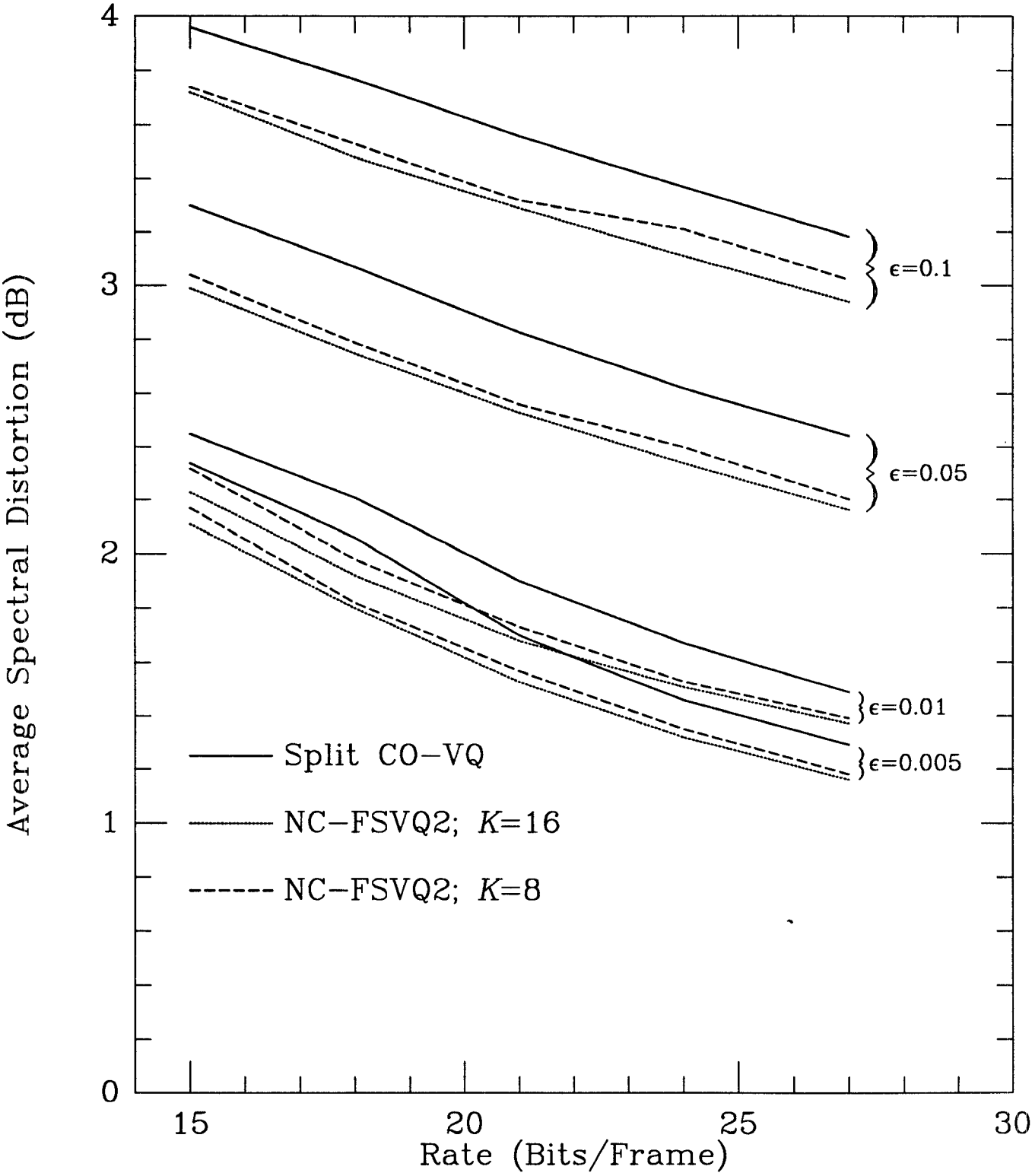


Figure 6: Performance of NC-FSVQ2 and split CO-VQ for various levels of channel noise and bit rates; speech LSP parameters; $K = 8$ and $K = 16$.