SRC TR 88-15

A Study of Vector Quantization for Noisy Channels

by

N. Farvardin

A Study of Vector Quantization for Noisy Channels

by

Nariman Farvardin †

Electrical Engineering Department

Institute for Advanced Computer Studies

and

Systems Research Center

University of Maryland

College Park, MD 20742

ABSTRACT

While there is ample evidence that vector quantization is a very useful technique for data compression, little is known about its sensitivity to channel and/or storage device errors. In this paper, several issues related to vector quantization for noisy channels are addressed. An algorithm based on simulated annealing is developed for assigning binary codewords to the vector quantizer codevectors. It is shown that this algorithm could result in dramatic performance improvements as compared to randomly selected codewords. Also, a modification of the simulated annealing algorithm for binary codeword assignment is developed for the case where the bits in the codeword are subjected to unequal error probabilities (resulting from unequal levels of error protection). An algorithm for the design of an optimal vector quantizer for a noisy channel is briefly discussed and its robustness under channel mismatch conditions is studied. Numerical results for a stationary first-order Gauss-Markov source and a binary symmetric channel are provided. It is concluded that the channel-optimized vector quantizer design algorithm, if used carefully, can result in a fairly robust system with no additional delay. Finally, the case in which the communication channel is nonstationary (as in mobile radio channels) is studied and some preliminary ideas for quantizer design are presented.

---

† Consultant, AT&T Bell Laboratories, June-July, 1987.

I. Introduction

The objective in a data compression system is to represent a sequence of analog (or finely discretized) samples by a relatively *low rate* sequence for subsequent *transmission* over the communication channel or *storage* in the storage device. Vector quantization, as a means of data compression, has received a tremendous amount of attention in the past decade. Utilizing vector quantization instead of scalar quantization has resulted in dramatic performance improvements (in terms of bit rate reduction for the same level of fidelity) in various speech and image coding situations. An informative tutorial survey of vector quantization including the underlying principles, the design algorithms, some applications and implementation issues can be found in [1]. Also, a tutorial survey of vector quantization with emphasis on speech coding applications is provided in [2].

An efficient data compression system removes, to the extent possible, the redundancy in the source and retains the useful (nonredundant) part in an effort to reduce the data rate. This removal of redundancy, in turn, introduces a great deal of sensitivity to the transmission noise or the storage device errors. Unless some corrective measures are taken, the more efficient the data compression system becomes, the more sensitive it will be to the channel and/or storage device noise. While the issues of sensitivity to the channel noise are studied in some depth for scalar quantization [3]-[5], there seems to be very little work on channel error effects and possible design procedures for vector quantizers. Exceptions are the works of De Marca and Jayant [6], Chen et al. [7] and Zeger and Gersho [8] on assigning binary codewords to the vector quantizer codevectors, as well as an algorithm for the design of vector quantizer for a noisy channel [9], [10], [23].

In this paper, we will study several issues related to vector quantization for noisy channels. Specifically, we will develop an algorithm based on *simulated annealing* for assigning binary codewords to the codevectors of a vector quantizer. A modification of this algorithm which reflects an intended hierarchy in the bit positions will also be discussed. This modified scheme is particularly suitable for subsequent unequal error protection through some type of channel coding. Furthermore, we will briefly describe an algorithm for the design of vector quantizers for a noisy channel and investigate its robustness under mismatch conditions. It will be shown, as a conclusion of our study, that given some information about the statistical properties of the channel, it is possible to improve the performance of the vector quantizer through a judicious,

joint design of the codebook and the binary codeword assignment. Numerical results in support of this claim will be provided.

The rest of this paper is organized as follows. Section II is devoted to the development of notation and some preliminary analysis. In Section III, an algorithm based on simulated annealing for the assignment of binary codewords to the codevectors of the vector quantizer is provided. A modification of this algorithm for subsequent unequal error protection is also described in Section III. Section IV includes a brief discussion of an algorithm for vector quantizer design for noisy channels. Further, issues of robustness of the channel-optimized vector quantizer under mismatch conditions, as well as the nonstationarity of the channel are addressed in this section. In Section V numerical results for the stationary Gaussian source (with and without memory) and a binary symmetric channel (possibly nonstationary) are provided. Finally, Section VI includes a summary and conclusions.

II. Preliminaries

We assume that the source to be encoded is a real-valued, discrete-time, continuous-amplitude, stationary process $\{X_t; t = 0, 1, \ldots\}$ with zero mean and variance σ_X^2.

The source is to be encoded by means of a vector quantizer. A k-dimensional vector quantizer is a mapping q, that assigns to a typical source output vector $\mathbf{x} = (x_{nk}, x_{nk+1}, \ldots, x_{nk+k-1})$, a vector $\mathbf{y} = q(\mathbf{x})$, drawn from a finite reproduction alphabet (codebook) $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_M\}$. The vector quantizer q is completely described by the codebook $\mathcal{C}$ together with the partition $\mathcal{P} = \{S_1, S_2, \ldots, S_M\}$ of the input vector space into the sets $S_i, i = 1, 2, \ldots, M$. Specifically, the mapping $q(\cdot)$ is described by

$$q(\mathbf{x}) = \mathbf{c}_i, \text{ if } \mathbf{x} \in S_i, \ i = 1, 2, \ldots, M \ . \tag{1}$$

Let us assume that the distortion caused by representing a source vector $\mathbf{x}$ by a reproduction vector (also called codevector) $\mathbf{y}$, is given by a nonnegative distortion measure $d(\mathbf{x}, \mathbf{y})$. The performance of a vector quantizer is generally measured by the *average distortion* per sample $D_S(q)$ described by

$$D_S(q) = \frac{1}{k} \sum_{i=1}^{M} \int_{S_i} p(\mathbf{x}) d(\mathbf{x}, \mathbf{c}_i) d\mathbf{x}, \tag{2.a}$$

where $p(\mathbf{x})$ is the k-fold probability density function of the source, and the *rate* is

$$R = \frac{1}{k} \log_2 M, \text{ bits/sample.} \tag{2.b}$$

For a given source and a given rate R, a vector quantizer q^* is said to be *optimum* if

$$D_S(q^*) \leq D_S(q), \text{ for all } q \, . \tag{3}$$

Various algorithms for the design of "good" vector quantizers have been developed in the past decade [1],[2],[11]-[14]. Among these, the algorithm that has had a very important impact is that developed by Linde, Buzo and Gray [11], hereafter referred to as the LBG algorithm, in which the partition P and the codebook C are iteratively updated. For the sake of the present discussion, unless specifically stated, we will assume that a vector quantizer q is designed and fixed.

In any practical situation, the codevectors of the vector quantizer should be mapped to *binary codewords*[1] for subsequent transmission or storage. The length of these binary codewords is $n = kR = \log_2 M$. Let us assume that $b(\mathbf{c}_i)$, $i = 1, 2, \ldots, M$, is a nonnegative *integer* whose n-bit binary representation is the binary codeword assigned to $\mathbf{c}_i$. From now on, we will refer to $b(\mathbf{c}_i)$ as the *index* associated with the codevector $\mathbf{c}_i$.

Suppose that the indices (or, equivalently, binary codewords) representing the output of the vector quantizer, are to be sent over a noisy channel. We assume that the channel is memoryless with finite input and output alphabets given by $\{0, 1, \ldots, M-1\}$. Let $p(j|i)$, $i, j = 0, 1, \ldots, M-1$, denote the probability that the index j is received given that the index i is transmitted. Therefore, for a given vector of indices, say $b \equiv (b(\mathbf{c}_1), b(\mathbf{c}_2), \ldots, b(\mathbf{c}_M))$, the average distortion per source sample, caused by the channel noise is given by

$$D_C(b) = \frac{1}{k} \sum_{i=1}^{M} \sum_{j=1}^{M} P(\mathbf{c}_i) P(b(\mathbf{c}_j)|b(\mathbf{c}_i)) d(\mathbf{c}_i, \mathbf{c}_j), \tag{4}$$

where $p(\mathbf{c}_i)$ is the a priori probability of the codeword $\mathbf{c}_i$.

[1] Here, we confine our attention to fixed-length binary codewords. The study of variable-length binary codewords over noisy channels is far more difficult due to error propagation issues.

Notice that the overall distortion caused by the vector quantizer and the channel is given by

$$D(q;b) = \frac{1}{k}\sum_{i=1}^{M}\sum_{j=1}^{M} P(b(\mathbf{c}_j)|b(\mathbf{c}_i)) \int_{S_i} p(\mathbf{x}) d(\mathbf{x},\mathbf{c}_j) d\mathbf{x}, \tag{5}$$

which, in general, is *not* equal to the sum of the source encoder distortion $D_S(q)$ and the channel distortion $D_C(b)$. However, if the distortion criterion is the widely used squared-error distortion, i.e.,

$$d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|^2, \tag{6}$$

equation (5) can be written as

$$\begin{aligned} D(q;b) = \frac{1}{k}\sum_{i=1}^{M}\sum_{j=1}^{M} P(b(\mathbf{c}_j)|b(\mathbf{c}_i)) \int_{S_i} p(\mathbf{x})\|\mathbf{x}-\mathbf{c}_i\|^2 \, d\mathbf{x} \\ +\frac{1}{k}\sum_{i=1}^{M}\sum_{j=1}^{M} P(b(\mathbf{c}_j)|b(\mathbf{c}_i))P(\mathbf{c}_i)\|\mathbf{c}_i-\mathbf{c}_j\|^2 \\ +\frac{2}{k}\sum_{i=1}^{M}\sum_{j=1}^{M} P(b(\mathbf{c}_j)|b(\mathbf{c}_i)) \int_{S_i} p(\mathbf{x})(\mathbf{x}-\mathbf{c}_i)^T(\mathbf{c}_i-\mathbf{c}_j) d\mathbf{x}. \end{aligned} \tag{7}$$

The third term on the right hand side of (7) vanishes if $\mathbf{c}_i$ is chosen as the centroid of S_i, $i = 1,2,\ldots,M$, in which case we will have

$$D(q;b) = D_S(q) + D_C(b). \tag{8}$$

Equation (8) is the vector extension of the similar result for scalar quantization [4], [5], suggesting the separation of the overall distortion to the sum of the quantization distortion and the channel distortion for the squared-error distortion measure. To enjoy this nice property, we will continue to assume in the sequel that the distortion measure is squared-error and that the codevectors are the centroid of their respective regions.[2]

In view of the above argument, we will focus our attention on minimizing the channel distortion $D_C(b)$ by an appropriate choice of the index assignment function b.

[2] Notice that for equation (8) to hold we do not need to make any assumptions on the partition $\mathcal{P}$.

The problem of channel error effects on the performance of source encoders has recently received some attention. Specifically, in a recent paper, De Marca and Jayant [6] developed an algorithm for assigning indices to the codevectors of the vector quantizer to minimize the channel distortion. It is shown in [6] that a substantial reduction in channel distortion can be obtained through a judicious assignment of indices rather than a random assignment. In two other recent papers Chen et al. [7] and Zeger and Gersho [8] have studied a somewhat similar algorithm for index assignment.

In what follows we will present an algorithm based on simulated annealing for the index assignment. We will provide some justification for the choice of this approach; primarily, it is supported by recent theoretical results on the behavior of simulated annealing for combinatorial optimization problems.

III. Simulated Annealing for Index Assignment

Many combinatorial optimization problems belong to a class of problems which are "difficult" to solve, generally referred to as the class of *NP*-complete problems [15]. This is the class of problems for which there is no known algorithm whose worst-case complexity is bounded by a polynomial in the size of the input. Usually, heuristic algorithms are used to find a "good" or "close-to-optimal" solution for these problems as the complexity of exhaustive search becomes prohibitive when the problem size is large. Most heuristic algorithms perform a deterministic search in a set of admissible configurations and often terminate in a *local* minimum.

To avoid this behavior, a class of randomized algorithms [16] have been devised in which (i) the next state configuration is generated *randomly* and (ii) "hill climbing" is allowed, i.e., moves that result in configurations of higher cost than the present one are accepted. Simulated annealing, as proposed by Kirkpatrick et al. [18] is one such random search algorithm in which "hill climbing" is allowed according to a certain criterion which takes into consideration the state of the search process.

The process of *annealing* is well-known in growing crystals. It consists of melting the material by increasing the temperature to the melting point and then lowering the temperature (or, cooling) slowly until the crystal is formed. The slow cooling stage allows the formation of a molecular configuration with minimum stored potential energy. The process of annealing is numerically simulated by the so-called Metropolis algorithm [17] to gain an understanding of

the ground state configuration.

A typical combinatorial optimization problem seeks the minimum of a given objective (or, cost) function of many variables. The variables are usually subject to intertwining constraints and interact with each other in a complicated way — similar to the interaction between the molecules in a physical system. Simulated annealing is simply a Monte Carlo Metropolis algorithm for solving a combinatorial optimization problem. It mimics the objective function associated with the combinatorial optimization problem as the *energy* associated with the physical system, and by slowly reducing an appropriately defined *effective temperature*, seeks the minimum energy state.

Simulated annealing has been applied to various combinatorial optimization problems with varying degrees of success [18]-[20]. Perhaps most relevant to our problem is the work in [20] in which simulated annealing is used for designing source codes as well as channel codes. Some of the specifics of the cooling schedule employed in our work are in fact borrowed from [20]. In what follows we will describe the simulated annealing algorithm in more precise terms and discuss its application to our index assignment problem.

Let us recall that the objective function to be minimized is $D_C(b)$ as described in (4). From now on we will refer to the index assignment function $b = (b(\mathbf{c}_1), b(\mathbf{c}_2), \ldots, b(\mathbf{c}_M))$ as the *state* and to the channel mean squared-error (MSE) $D_C(b)$ as the *energy* of a hypothetical system. Then, the simulated annealing algorithm works as follows:

Algorithm:

0) Define an effective temperature T and set $T = T_0$, an initial high temperature; randomly choose an initial state b.

1) Randomly choose a state b', a perturbation of the state b and let $\Delta D_C = D_C(b') - D_C(b)$.

   1.a) If $\Delta D_C < 0$, replace b by b', go to step (2).

   1.b) If $\Delta D_C \geq 0$, replace b by b' with probability $\exp\{-\frac{\Delta D_C}{T}\}$, go to step (2).

2) If in step (1) the number of energy drops exceeds a prescribed number or if too many unsuccessful perturbations (perturbations not resulting in energy drops) occur, go to step (3). Otherwise, go to step (1).

3) Lower temperature T. If the temperature T is below some prescribed freezing temperature T_f or if it appears that a stable state is reached, stop, with b describing the state. Otherwise, go to step (1).

While the above algorithm describes the basics of the operation of the simulated annealing algorithm, it requires some elaboration. A few comments are in order.

In step (1) of the algorithm, perturbing the state b means "jiggling" it, i.e., taking two components of the index vector b randomly and interchanging them. For instance $b' = (1, 0, 3, 2)$ is a perturbation of $b = (0, 1, 3, 2)$, in which the first and the second indices in the state vector are interchanged. Notice that this choice of perturbation allows us to move from *any* state to *any* state in a *finite* number of perturbations.

In step (1.a), $\Delta D_C < 0$ corresponds to a decrease of the energy and hence it is accepted, as the final goal is to minimize the energy. On the other hand $\Delta D_C \geq 0$ corresponds to an increase of the energy. This is still accepted with a probability which decreases as the effective temperature is decreased. Thus, in the beginning when the system is "hot", almost all perturbations (causing energy drops or energy increases) are accepted. However, as the temperature is reduced those perturbations which cause an increase of the energy will be accepted with diminishingly small probabilities. This process allows the algorithm to climb out of local minima when the temperature is high in the hope that as the system is cooled the state falls in the global minimum (or, somewhere close to it) out of which the algorithm should not be able to climb. It is easy to show that the sequence of states in the simulated annealing process forms a discrete-time nonhomogeneous (nonstationary) Markov chain. This fact has been used to show that, with a suitable perturbation scheme and a sufficiently slow cooling schedule, the algorithm converges in probability to the *global* minimum. Specifically, it is shown in [21] and [22] that if the initial melting temperature T_0 is sufficiently large, a cooling schedule described by $T_k = c/\log(k+1)$, insures convergence in probability to the global minimum. Here, T_k is the value of the temperature after the kth temperature drop.

Since this cooling schedule is very slow, faster cooling schedules such as $T_k = \alpha T_{k-1}$, where $0 < \alpha < 1$, have been used in practical applications of simulated annealing [18], [20]. Details of the simulated annealing algorithm as applied to the index assignment problem, along with some numerical results are presented in Section V.

Before we close this section, we need to make a few comments about the channel transition probabilities in (4). If the channel used for transmission of binary codewords is a binary symmetric channel with crossover probability ϵ, the transition probabilities in (4) will be described by

$$P(j\,|i\,) = (1-\epsilon)^{n-d_H(i,j)}\epsilon^{d_H(i,j)}, \quad i,j = 0,1,\ldots,M-1, \tag{9}$$

where $d_H(i,j)$ is the Hamming distance between the n-bit binary codewords represented by integers i and j. In this case, clearly, all bits in the binary codeword suffer from the same probability of error. The index assignment obtained through the simulated annealing algorithm, therefore, does not place any particular hierarchy on the bits depending on their position in the codeword, as they are all transmitted through the same channel.

When the communication channel becomes very noisy, it is common practice to resort to some type of channel coding scheme to reduce the effective channel bit error probability. Since channel coding requires the transmission of some redundant bits, it should be used very carefully. In practical situations, it is desirable to use channel coding in such a way as to provide the highest level of protection on the most significant bit[3] (MSB) followed by lower levels of protection on other bit positions and finally the lowest level of protection (or, perhaps no protection at all) on the least significant bit (LSB).

If such an unequal error protection (UEP) on different bits is to be used in encoding the binary codewords, the expression in equation (9) will not be valid any more. To be more precise, assume that UEP is used so that the effective bit error rate on the leftmost bit is ϵ_1, followed by ϵ_2 for the adjacent bit and finally ϵ_n for the rightmost bit. Then, the objective function to be minimized is again described by (4) in which

$$P(j\,|i\,) = \prod_{k=1}^{n}(1-\epsilon_k)^{1-\ell_k(i,j)}\epsilon_k^{\ell_k(i,j)}, \quad i,j = 0,1,\ldots,M-1, \tag{10}$$

where $\ell_k(i,j) = 1$ if the binary codewords represented by i and j differ in the kth position and $\ell_k(i,j) = 0$ otherwise.

It can be seen from the above formulation that if $\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_n$, the leftmost bit suffers from a smaller channel error than the rightmost bit. Therefore, in assigning indices to the codevectors, the leftmost bit may be used to distinguish between two codevectors with large

[3] Here, by the most significant bit we mean that bit position in which the occurrence of a channel error causes a larger MSE in the reconstructed output than in any other bit position.

Euclidean distance, whereas, this should be avoided, as much as possible, for the rightmost bit.

This unequal error protection followed by simulated annealing results in a natural *prioritization* of bits in the codeword so that the leftmost bit becomes the MSB and the rightmost bit becomes the LSB. This is a desirable property in any communication system where a limited amount of error control coding is to be utilized. We will provide a specific example in Section IV to demonstrate the usefulness of such UEP.

IV. Vector Quantizer Design for Noisy Channels

In this section we will provide a brief description of an iterative algorithm described in [9],[10] for the design of a vector quantizer for noisy channels. We will also address some issues that arise as a result of channel mismatch and channel nonstationarity.

Let us recall that for a given quantizer q and a given index assignment b, the average distortion is given by (5). To simplify our notation, we will further assume that the codevectors are rearranged so that $b(\mathbf{c}_i) = i - 1,\ i = 1, 2, \ldots, M$. With this assumption then, for a given codebook $C = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_M\}$ and a given partition $P = \{S_1, S_2, \ldots, S_M\}$, the average distortion is given by

$$D(C; P) = \frac{1}{k} \sum_{i=1}^{M} \sum_{j=1}^{M} P(j-1|i-1) \int_{S_i} p(\mathbf{x}) \|\mathbf{x} - \mathbf{c}_j\|^2 \, d\mathbf{x}. \tag{11}$$

The algorithm for optimizing the codebook and the partition, which is a straightforward vector extension of the algorithm described in [5], iterates by updating the codebook C for a fixed partition P, followed by updating P for a fixed codebook C.

More specifically, if the partition P is fixed, the best codevector $\mathbf{c}_j$ is the conditional expectation of $\mathbf{X}$ given that the index $(j-1)$ is received, i.e.,

$$\mathbf{c}_j = E(\mathbf{X}|j-1) = \frac{\sum_{i=1}^{M} P(j-1|i-1) \int_{S_i} \mathbf{x} p(\mathbf{x}) d\mathbf{x}}{\sum_{i=1}^{M} P(j-1|i-1) \int_{S_i} p(\mathbf{x}) d\mathbf{x}}. \tag{12.a}$$

Similarly, for a fixed codebook C, it is easy to show that the best partition is that in which [9], [10]

$$S_i = \{\mathbf{x} : \sum_{j=1}^{M} P(j-1|i-1)\|\mathbf{x}-\mathbf{c}_j\|^2 \leq \sum_{j=1}^{M} P(j-1|\ell-1)\|\mathbf{x}-\mathbf{c}_j\|^2, \forall \ell\}, \; i = 1, 2, \ldots, M \,. \quad (12.b)$$

A successive application of equations (12.a) and (12.b) will result in a sequence of quantizers for which the corresponding average distortions form a non-increasing sequence of nonnegative numbers which has to converge. This algorithm, similar to its scalar version [5], trades quantization accuracy for smaller channel distortion in an attempt to minimize the overall distortion. From now on, we will refer to this algorithm as the generalized Lloyd algorithm and the resulting vector quantizer will be referred to as the *channel-optimized* vector quantizer. Notice that the index assignment is a byproduct of this algorithm as we always associate the index $(i-1)$ with the codevector $\mathbf{c}_i$, $i = 1, 2, \ldots, M$.

IV.A. Channel Mismatch

While the above algorithm yields a locally optimum vector quantizer when the channel transition probability is known, it would be useful to study its robustness when applied to a channel with a different transition probability.

To assess the performance of the channel-optimized vector quantizer under channel mismatch conditions it would be desirable to separate the source encoder distortion from the channel distortion. It is easy to show that if $\hat{\mathbf{c}}_i$ represents the centroid of the region S_i, the overall distortion is given by

$$D = \frac{1}{k}\sum_{i=1}^{M}\int_{S_i} p(\mathbf{x})\|\mathbf{x} - \hat{\mathbf{c}}_i\|^2 d\mathbf{x} + \frac{1}{k}\sum_{i=1}^{M}\sum_{j=1}^{M}\hat{P}_i P(j-1|i-1)\|\hat{\mathbf{c}}_i - \mathbf{c}_j\|^2 \,, \qquad (13)$$

in which the first term can be thought of as the source encoder distortion while the second term can be interpreted as the channel distortion. Here, $\hat{P}_i$ is the probability that the source vector $\mathbf{X}$ falls in S_i.

The advantage of separating the overall distortion according to (13) is that now the source encoder distortion is a quantity which is fixed and depends only upon the characteristics of the assumed channel, while the second term, whose computation is simple, bears the effects of the active channel. Numerical results for the performance of the channel-optimized quantizer under channel mismatch conditions are provided in Section V.

IV.B. Nonstationary Channels

In many practical situations the communication channel is inherently nonstationary. A good example is the land mobile radio channel, in which as the vehicle is driven around, due to various obstructions, deep fades in the received signal strength may cause large error probabilities for short periods of time, while at other times the error probability may be fairly small. In such a situation the following important question arises naturally: if a channel-optimized vector quantizer is to be employed for operation over a nonstationary channel, what error probability should it be designed for?

Let us suppose that the channel is a binary symmetric channel. Let ϵ_d and ϵ_a denote the crossover probability (error probability) of the channel for which the quantizer is designed and the crossover probability of the active channel, respectively. Let $D(\epsilon_a, \epsilon_d)$ denote the corresponding mismatch condition distortion. Now suppose ϵ_a itself is a random quantity with a specified distribution. Then the average value of the distortion is given by

$$\bar{D}(\epsilon_d) = E(D(\epsilon_a, \epsilon_d)) \ , \tag{14}$$

where the expectation is with respect to the probability distribution of ϵ_a. We wish to determine that ϵ_d which minimizes $\bar{D}(\epsilon_d)$. This problem, in its most generally, is difficult to solve. However, if we assume that ϵ_a is restricted to take on very small values ($n\epsilon_a \ll 1$), so that only single-error patterns can be considered, we will have from (11)

$$D(\epsilon_a, \epsilon_d) = \frac{1}{k}\sum_{i=1}^{M}(1 - n\epsilon_a)\int_{S_i} p(\mathbf{x})\|\mathbf{x} - \mathbf{c}_i\|^2 d\mathbf{x} + \frac{\epsilon_a}{k}\sum_{i=1}^{M}\sum_{j\in\tau_i}\int_{S_i} p(\mathbf{x})\|\mathbf{x} - \mathbf{c}_j\|^2 d\mathbf{x} \ , \tag{15}$$

in which τ_i is the set of all integers j satisfying $d_H(i,j) = 1$. When averaged with respect to ϵ_a, we will obtain

$$\bar{D}(\epsilon_d) = D(\bar{\epsilon}_a, \epsilon_d) \ , \tag{16}$$

where $\bar{\epsilon}_a$ is simply the average value of the active channel crossover probability. Clearly, $D(\bar{\epsilon}_a, \epsilon_d) \geq D(\bar{\epsilon}_a, \bar{\epsilon}_a)$, suggesting that for the best average performance, we must choose $\epsilon_d = \bar{\epsilon}_a$.

It is important to mention that the reason behind this simplification resides in the fact that when $n\epsilon_a \ll 1, D(\epsilon_a, \epsilon_d)$ can be approximated as a linear function of ϵ_a. If the assumption of $n\epsilon_a \ll 1$ does not hold, the linearly property collapses as a result of which the above result does not hold any more.

V. Numerical Results

This section will present numerical results for the algorithms discussed in Sections III and IV.A.

V.A. Codes from the Simulated Annealing Algorithm

In all of our results, the source to be quantized is assumed to be a first-order autoregressive Gaussian source described by

$$X_n = \rho X_{n-1} + W_n, \quad n = 1, 2, \ldots, \tag{17}$$

where $\{W_n\}$ is a sequence of independent and identically distributed Gaussian random variables with zero mean and variance σ_W^2, and the initial condition X_0 is chosen to guarantee the stationarity of the process $\{X_n\}$.

We have restricted our attention to rate $R = 1$ bit/sample vector quantizers. For a memoryless Gaussian source ($\rho = 0.0$) and a heavily correlated Gaussian source with $\rho = 0.9$, we have designed vector quantizers of dimensions $k = 2, 4, 6$, and 8. The LBG algorithm is used for vector quantizer design and the splitting algorithm is used for initialization [11]. As mentioned earlier, the distortion measure is the squared-error throughout.

The simulated annealing algorithm, as described in Section III, is run to minimize the average channel distortion $D_C(b)$. In fact, to simplify our numerical computations, we have assumed that the channel bit error rate (BER) is sufficiently small so that we can consider only single bit error patterns in a codeword. Specifically, let τ_i be the collection of all integers $j, 0 \leq j \leq 2^n - 1$, such that the binary representation of j is of Hamming disance one from the binary representation of i. Then, our channel is approximated by

$$P(j \mid i) = \begin{cases} \epsilon, & j \in \tau_i, \\ 1 - n\epsilon, & j = i, \\ 0, & \text{otherwise}, \end{cases} \tag{18}$$

where ϵ is the channel bit error rate. With this approximation for the channel, the average channel distortion, normalized to the channel bit error rate, is given by

$$\frac{D_C(b)}{\epsilon} = \frac{1}{k}\sum_{i=1}^{M} p(\mathbf{c}_i) \sum_{j:b(\mathbf{c}_j)\in\tau_{b(\mathbf{c}_i)}} d(\mathbf{c}_i, \mathbf{c}_j), \tag{19}$$

in which we have implicitly assumed that $d(\mathbf{c}_i, \mathbf{c}_i) = 0,\ i = 1, 2, \ldots, M$.

The simulated annealing algorithm is run with an initial (melting) temperature $T_0 = 10.0$ and a final (freezing) temperature $T_f = 2.5 \times 10^{-4}$. The *cooling schedule* is as follows. The temperature is reduced by a factor $\alpha = 0.97$ after *five* drops (not necessarily in a row) in the average distortion or after more than 200 perturbations, whichever happens first. The algorithm terminates when the temperature drops below T_f or if 50,000 consecutive perturbations do not result in a drop in the average distortion. It is important to mention that the above choices of parameters are obtained by experimentation and seem to yield satisfactory results.

Figures 1 and 2 illustrate the evolution of the simulated annealing algorithm for the memoryless and the correlated Gaussian source, respectively. In both cases the dimension of the vector quantizer is $k = 6$. In these figures, we have plotted the channel MSE as a function of the effective temperature. Every time that a perturbed assignment is accepted, a point indicating its energy and the corresponding effective temperature is registered on the plot. In the example of Figure 2, the benefit of the simulated annealing algorithm is a gain of about 4.5 dB in channel error power (reduction of MSE from about 15.0 to about 5.0).

As described in Section III, it is easy to see from Figs. 1 and 2 that when the effective temperature is high almost all perturbations are accepted. However, as the temperature is reduced, the general trend is to accept those perturbations which result in a drop of energy. Another interesting observation is the larger variation (when the temperature is high) in the value of the energy when the source is heavily correlated. This is due to the fact that for a heavily correlated Gaussian source, the codevectors of the vector quantizer are concentrated along a thin ellipsoid in the k-dimensional Euclidean space. Therefore, some codevectors are very close to each other, while some are very far. As a result, a small perturbation of the index assignment could result in a large variation in the energy. This phenomenon does not occur to the same extent when the source is memoryless, as in that case the codevectors are more uniformly distributed in the space and the variation in their distances is not as pronounced.

To examine the consistency of the results obtained by the simulated annealing algorithm, for each vector quantizer we have run the simulated annealing algorithm five times. These results, for different values of the correlation coefficient and different block sizes are summarized in Table 1. These results indicate that the simulated annealing algorithm converges to a point whose corresponding average distortion does not vary too much from run to run. Also, it is interesting to observe that for situations with low dimensionality, this variation is smaller (small number of local minima). Indeed, for the cases with $k = 2$ (four codevectors) the simulated annealing algorithm yielded the same result in all five runs.

To make a comparison against the performance of a system in which the indices are *randomly* chosen, it would be desirable to compute the ensemble average of the average channel distortion over all possible codeword assignments. Since the total number of different codeword assignments is $M!$, the ensemble average distortion, denoted by $\bar{D}_C$, is computed according to

$$\bar{D}_C = \frac{1}{M!} \sum_{\text{all } b} D_C(b) \ , \tag{20}$$

which for the channel of (18) can be written as

$$\bar{D}_C = \frac{1}{M!} \cdot \frac{\epsilon}{k} \sum_{i=1}^{M} p(\mathbf{c}_i) \sum_{\text{all } b} \sum_{j:b(\mathbf{c}_j) \in \tau_{b(\mathbf{c}_i)}} d(\mathbf{c}_i, \mathbf{c}_j) \ , \tag{21}$$

which, in turn, simplifies to

$$\bar{D}_C = \frac{\epsilon n}{k(M-1)} \sum_{i=1}^{M} p(\mathbf{c}_i) \sum_{j=1}^{M} d(\mathbf{c}_i, \mathbf{c}_j) \ . \tag{22}$$

For comparison purposes, this ensemble average of the average channel distortion is also included in Table 1.

The results summarized in Table 1 suggest that use of simulated annealing results in an improvement in average channel distortion compared with randomly chosen codewords, which varies between 1-5 dB. Generally speaking, the improvement is more substantial for larger dimensions (larger number of codevectors) and for more heavily correlated sources (nonuniformly dispersed codevectors).

An important observation to be made in Table 1 is the trend of the average channel distortion (obtained in the simulated annealing algorithm) as the dimension k of the vector quantizer increases. It is interesting to note that the rate of increase of the average channel distortion (as a function of k) is much greater for $\rho = 0.9$. The reason for this behavior may be the greater concentration of the multi-dimensional source distribution in a narrow region of k-space in the case of $\rho = 0.9$, which forces the codevectors to be placed in a concentrated region. As a result of this, for $k = 8$, the average channel distortion for $\rho = 0.9$ is smaller than that for $\rho = 0.0$, while the reverse is true for the smaller values of k illustrated in Table 1.

Since the overall MSE is the sum of the vector quantizer MSE and the channel MSE, to make an appropriate comparison, in Figs. 3-10, we have plotted the overall signal-to-noise ratio (in dB) of the vector quantization scheme followed by various binary codeword assignments.

The performance of the scheme utilizing simulated annealing for binary codeword assignment is indicated by a solid line in these figures and labeled as "Simulated Annealing." For comparison with randomly selected codewords, we have chosen, at random, 10 binary codeword assignments. The *average* performance of these sets of codewords as well as the *worst* case performance are also include in these figures and labeled as "Average" and "Worst" respectively.

The general conclusion that can be drawn from these figures is that an optimal choice of the binary codeword assignment improves the overall signal-to-noise ratio over a randomly selected codeword assignment. This improvement could be quite substantial when the channel is very noisy. Also, the improvements are more noticeable in higher dimensions and for more heavily correlated sources.

It is important to emphasize that while in the absence of channel noise, increasing the block size improves the performance of the vector quantizer, at the same time this increase of block size introduces more sensitivity to the channel noise.

V.B. Codes from the Splitting Algorithm for VQ Design

Recall that we have used the "splitting" algorithm [11] for the initialization of the vector quantizer design algorithm. One could use this initialization algorithm to obtain a binary codeword assignment as follows. To design a k-dimensional vector quantizer with $M = 2^{kR}$ codevectors, we first compute the centroid of the entire training sequence, say $\mathbf{y}_0$. Then, we split $\mathbf{y}_0$ into $\mathbf{y}_0$ and $(1+\epsilon)\mathbf{y}_0$. This will result in partitioning the set of training vectors into

two sets, namely those training vectors which are closer to $\mathbf{y}_0$ and those which are closer to $(1+\epsilon)\mathbf{y}_0$. We compute the centroid of these two regions, call them $\mathbf{y}_{11}$ and $\mathbf{y}_{12}$ and label them by "0" and "1", respectively. Then, in the same manner, we split each of $\mathbf{y}_{11}$ and $\mathbf{y}_{12}$ and compute four new centroids called $\mathbf{y}_{21}$, $\mathbf{y}_{22}$, $\mathbf{y}_{23}$, $\mathbf{y}_{24}$ and labeled "00", "01", "10" and "11", respectively. Clearly, if we continue in the same manner, we will end up with the desired M codevectors with a natural binary code (NBC) assignment. The results of this algorithm for codeword assignment is also included in Figs. 3-10 and labeled as "Splitting."

It is interesting to note that the binary codeword assignment based on the splitting algorithm performs very well in most cases. This is especially true when the source has memory. In this case, as mentioned earlier, the codevectors will be concentrated on a thin ellipsoid in the k-dimensional space and the binary partitions obtained by the splitting algorithm, are such that, "most" vectors in the same region are closer to one another than to those in other regions, and therefore, to a great extent, binary codewords of small Hamming distance will be assigned to codevectors of small Euclidean distance. A side advantage of the codewords obtained through the splitting algorithm is that, for sources with memory, they result in a hierarchy in bit positions. In other words, a channel error in the leftmost bit causes a larger distortion than that in the rightmost bit. This property can be exploited if subsequent channel coding is to be used.

V.C. Unequal Error Protection

To investigate the potential advantages of error control coding in improving the robustness of the vector quantizer against channel noise, we have considered a simple example of unequal error protection. Specifically, to design a k-dimensional quantizer with rate $R = 1$ bit/sample we can have up to $M = 2^k$ codevectors. Instead, we have considered designing k-dimensional vector quantizers with $M' = 2^{k-1}$ codevectors, and hence rate $R' = (k-1)/k$ bits/sample. Then, we have taken the leftmost bit and we used a rate 1/2 channel code for encoding it. The other bits are transmitted without any protection. Notice that the effective encoding rate is again 1 bit/sample. The rationale behind considering this scheme is to investigate the potential advantages of trading source coding accuracy for increased channel error protection.

In order to obtain the optimum index assignment for this UEP scheme, we assume that the effective bit error rate for the protected bit is *two orders of magnitude* smaller than that for other bits. With this assumption, the modified version of the simulated annealing algorithm

described in Section III is used to obtain the optimal index assignment. The resulting signal-to-noise ratio (SNR) performance is included in Figs. 3-10 and labeled as "UEP."

Clearly, the UEP scheme offers robustness against channel errors at the cost of some performance degradation for the noiseless channel. The usefulness of the UEP scheme becomes clearer for larger values of k. We should mention, however, that the results suggested by UEP curves in Figs. 3-10 are very *optimistic* in the region where the channel is very *noisy*. This is due to the fact that, for a very noisy channel ($\epsilon \geq 0.01$), gaining two orders of magnitude improvement in the probability of error using a rate 1/2 channel code is a very difficult (if not impossible) task. The concomitant delay and complexity of such a powerful channel code will cause serious implementation problems. Nevertheless, these UEP curves are useful as they determine a performance bound on what can be done by heavily protecting the MSB.

V.D. Channel-optimized Vector Quantizer

Finally, in Figs. 3-10 we have included the signal-to-noise ratio (in dB) of the channel-optimized vector quantizer described in Section IV, for four different channels, namely $\epsilon = 0.005$, $\epsilon = 0.01$, $\epsilon = 0.05$ and $\epsilon = 0.1$. These results are indicated by triangles and labeled as "Channel-Optimized." These results are interesting as they offer performance improvements (in some cases substantial) over the optimum vector quantizer designed for the noiseless channel and followed by an optimum index assignment. This is done at the cost of no additional delay. However, the comparison between the "Simulated Annealing" curves and the "Channel-Optimized" triangles is not entirely fair. Specifically, the vector quantizer used in the "Channel-Optimized" category is based on the exact characteristics of the channel while the vector quantizer in the "Simulated Annealing" examples is designed based on a noiseless channel.

To make a fair comparison, we have obtained the SNR performance results of the channel-optimized quantizers designed for $\epsilon_d = 0.005$, $\epsilon_d = 0.01$, $\epsilon_d = 0.05$, and $\epsilon_d = 0.1$ under channel mismatch conditions. These results for $k = 8$ are illustrated in Figures 11 and 12 (broken curves). For comparison purposes, the SNR performance of the vector quantizer designed for the noiseless channel and followed by an optimum index assignment is also included (solid curve) and labeled as "Simulated Annealing, $\epsilon_d = 0.0$."

It can be seen from results in Figs. 11 and 12 that the channel-optimized vector quantizers demonstrate a fair degree of robustness. Indeed, depending on the practical situation and the

active channel characteristics, a channel-optimized vector quantizer can be chosen to obtain the highest degree of robustness. It is also very interesting to note that by choosing an appropriate channel-optimized quantizer, a performance close to that indicated by UEP in Figs. 9 and 10 can be obtained. This observation is very important because, as we had indicated earlier, the UEP results in Figs. 3-10 are too optimistic for heavily noisy channels. Furthermore, the use of a channel code for the protection of the MSB results in added complexity and delay. These results suggest that to obtain the desired level of robustness, it is preferable to use a judiciously designed source encoder rather than using a conventional source encoder followed by a channel code.

VI. Summary and Conclusions

We have studied various design and performance issues related to vector quantization for noisy channels. First, we have developed a simulated annealing type algorithm for the optimal assignment of binary codewords to the codevectors of a vector quantizer. The performance of the vector quantizer with this codeword assignment is compared against other types of codeword assignment. In particular, random codewords and codeword assignment based on the splitting algorithm are considered. It is shown that for large dimensions and for heavily correlated sources, an optimal choice of the codeword assignment could result in substantial performance improvements. The algorithm based on splitting performs well and almost in all cases within 1 dB of the optimum performance.

We have also considered the possibility of redesigning the quantizer and the codeword assignment simultaneously in an effort to obtain the optimum performance for a given noisy channel. Again, it is shown that for very noisy channels and for high dimensionalities, the channel-optimized quantizer offers substantial improvements over the vector quantizer designed for the noiseless channel. Finally, the issue of robustness of the channel-optimized vector quantizer against channel mismatch conditions is investigated. It is concluded that a high degree of robustness can be obtained through an appropriate choice of the vector quantizer at the cost of no additional delay and only a modest increase in complexity.

Acknowledgement: The author is grateful to Drs. N. S. Jayant and C.-E. Sundberg from AT&T Bell Laboratories, Murray Hill, New Jersey, for many fruitful suggestions and discussions.

REFERENCES

1. R. M. Gray, "Vector Quantization," *IEEE Acoust., Speech, Signal Processing Magazine*, Vol., 1, pp. 4-29, April 1984.
2. J. Makhoul, S. Roucos and H. Gish, "Vector Quantization in Speech Coding," *IEEE Proc.*, Vol. 73, pp. 1551-1588, Nov. 1985.
3. A. Kurtenbach and P. Wintz, "Quantizing for Noisy Channels," *IEEE Trans. Commun. Technology*, Vol. COM-17, pp. 291-302, April 1969.
4. N. Rydbeck and C. W. Sundberg, "Analysis of Digital Errors in Nonlinear PCM Systems," *IEEE Trans. Commun.*, Vol. COM-24, pp. 59-65, Jan. 1976.
5. N. Farvardin and V. Vaishampayan, "Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding," *IEEE Trans. Inform. Theory*, Vol. IT-33, pp. 827-838, Nov. 1987.
6. J. R. B. De Marca and N. S. Jayant, "An Algorithm for Assigning Binary Indices to the Codevectors of a Multidimensional Quantizer," *Proc. IEEE Int. Comm. Conf.*, Seattle, WA, pp. 1128-1132, June 1987.
7. J.-H. Chen, G. Davidson, A. Gersho and K. Zeger, "Speech Coding for the Mobile Satellite Experiment," *Proc. IEEE Int. Comm. Conf.*, Seattle, WA, pp. 756-763, June 1987.
8. K. A. Zeger and A. Gersho, "Zero Redundancy Channel Coding in Vector Quantization," *IEE Electronics Letters*, Vol. 23, pp. 654-655, June 1987.
9. 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.
10. N. Farvardin and V. Vaishampayan, "An Algorithm for Quantizer Design for Noisy Channels," in preparation.
11. Y. Line, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Commun.*, Vol. COM-28, pp. 84-95, Jan. 1980.
12. A. Gersho, "Asymptotically Optimal Block Quantization," *IEEE Trans. Inform. Theory*, Vol. IT-25, pp. 373-380, July 1979.
13. J. H. Conway and N. J. A. Sloane, "Voronoi Regions of Lattices, Second Moments of Polytopes and Quantization," *IEEE Trans. Inform Theory*, Vol. IT-28, pp. 211-226, March 1982.

14. 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.

15. M. R. Garey and D. S. Johnson, *Computers and Interactability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

16. J. Schwartz, "Fast Probabilistic Algorithms for Verification of Polynomial Indentities," *J. Assoc. Comput.*, 27, pp. 701-717, March, 1980.

17. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines," *J. Chem. Phys.*, Vol. 21 pp. 1087-1091, June 1953.

18. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, pp. 671-680, May 1983.

19. F. Romeo, C. Sechen and A. Sangiovanni-Vincentelli, "Simulated Annealing Research at Berkeley," *Proc. IEEE Int. Conf. Computer Design*, pp. 652-657, 1984.

20. A. A. El Gamal, L. A. Hamachandra, I. Shperling and V. Wei, "Using Simulated Annealing to Design Good Codes," *IEEE Trans. Inform. Theory*, Vol. IT-33, pp. 116-123, Jan. 1987.

21. B. Hajek, "A Tutorial Survey of Theory and Applications of Simulated Annealing," *Proc. of IEEE Conf. on Decision and Control*, Ft. Lauderdale, FL, pp. 755-760, December 1985.

22. D. Mitra, F. Rameo and A. Sangiovanni-Vincentelli, "Convergence and Finite-Time Behavior of Simulated Annealing," *Adv. Appl. Prob., 18*, pp. 747-771, Sept. 1986.

23. 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.

List of Tables

Table 1: Channel MSE (normalized to the channel bit error probability) for different vector quantizers in five different runs of the simulated annealing algorithm and ensemble average of $M!$ codes (normalized to the channel bit error probability); Gaussian source, $\sigma_X^2 = 1$.

List of Figures

	$\rho = 0.0$				$\rho = 0.9$			
	k=2	k=4	k=6	k=8	k=2	k=4	k=6	k=8
Run #1	2.56	3.56	4.94	6.73	4.35	5.42	5.86	6.00
Run #2	2.56	3.56	4.81	6.70	4.35	5.34	5.48	5.97
Run #3	2.57	3.54	5.07	6.68	4.35	5.40	5.44	5.82
Run #4	2.56	3.57	4.92	6.63	4.35	5.42	5.45	5.88
Run #5	2.56	3.52	4.73	6.72	4.35	5.40	5.51	6.20
Random	3.39	5.68	8.40	11.41	5.29	10.60	15.06	20.15

Table 1

Channel MSE (normalized to the channel bit error probability) for different vector quantizers in five different runs of the simulated annealing algorithm and ensemble average of $M!$ codes (normalized to the channel bit error probability); Gaussian source, $\sigma_X^2 = 1$.

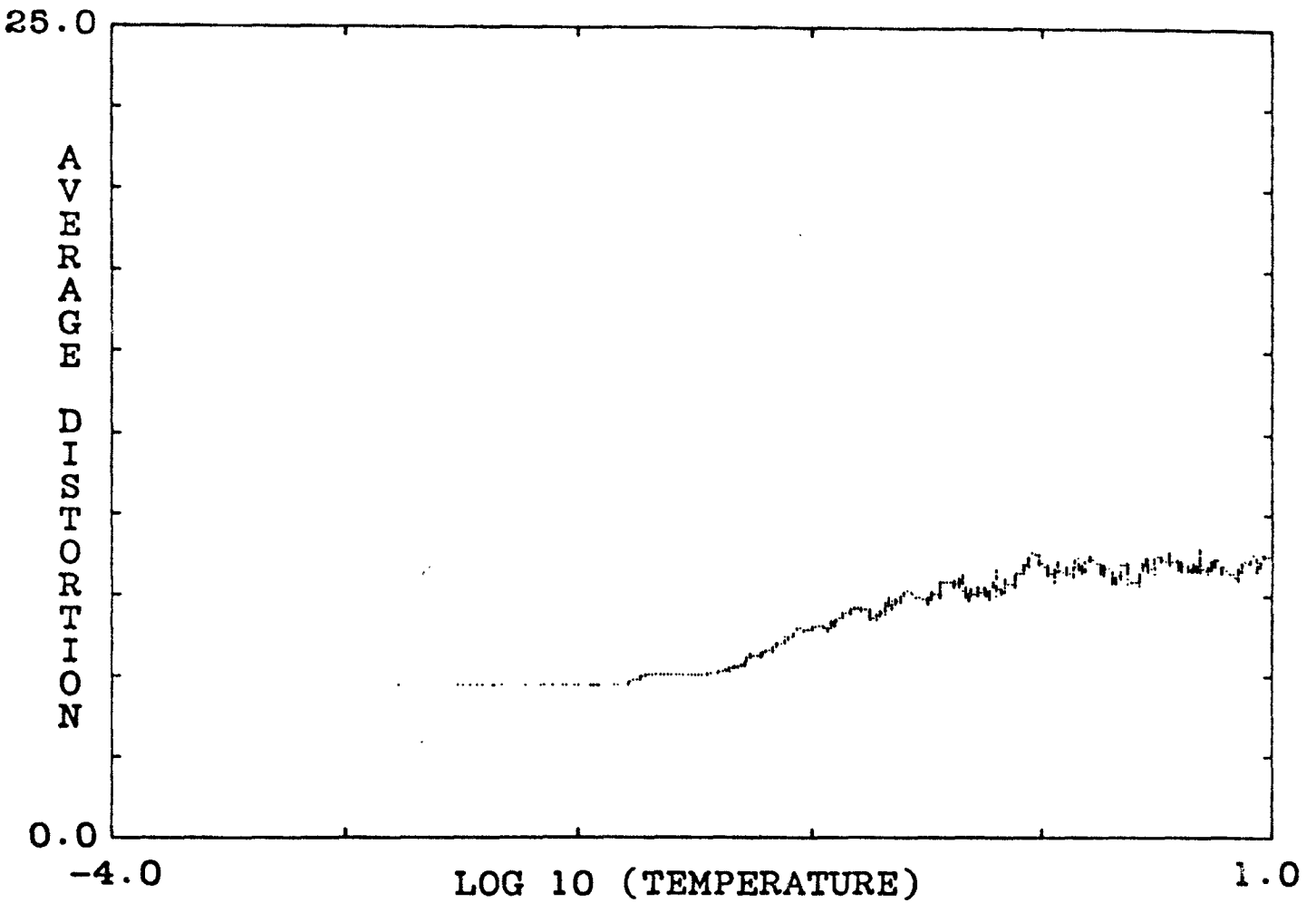

Figure 1

The Evolution of the Channel MSE During a Run of the Simulated Annealing Algorithm; $k = 6$, $\rho = 0$. (Memoryless Source).

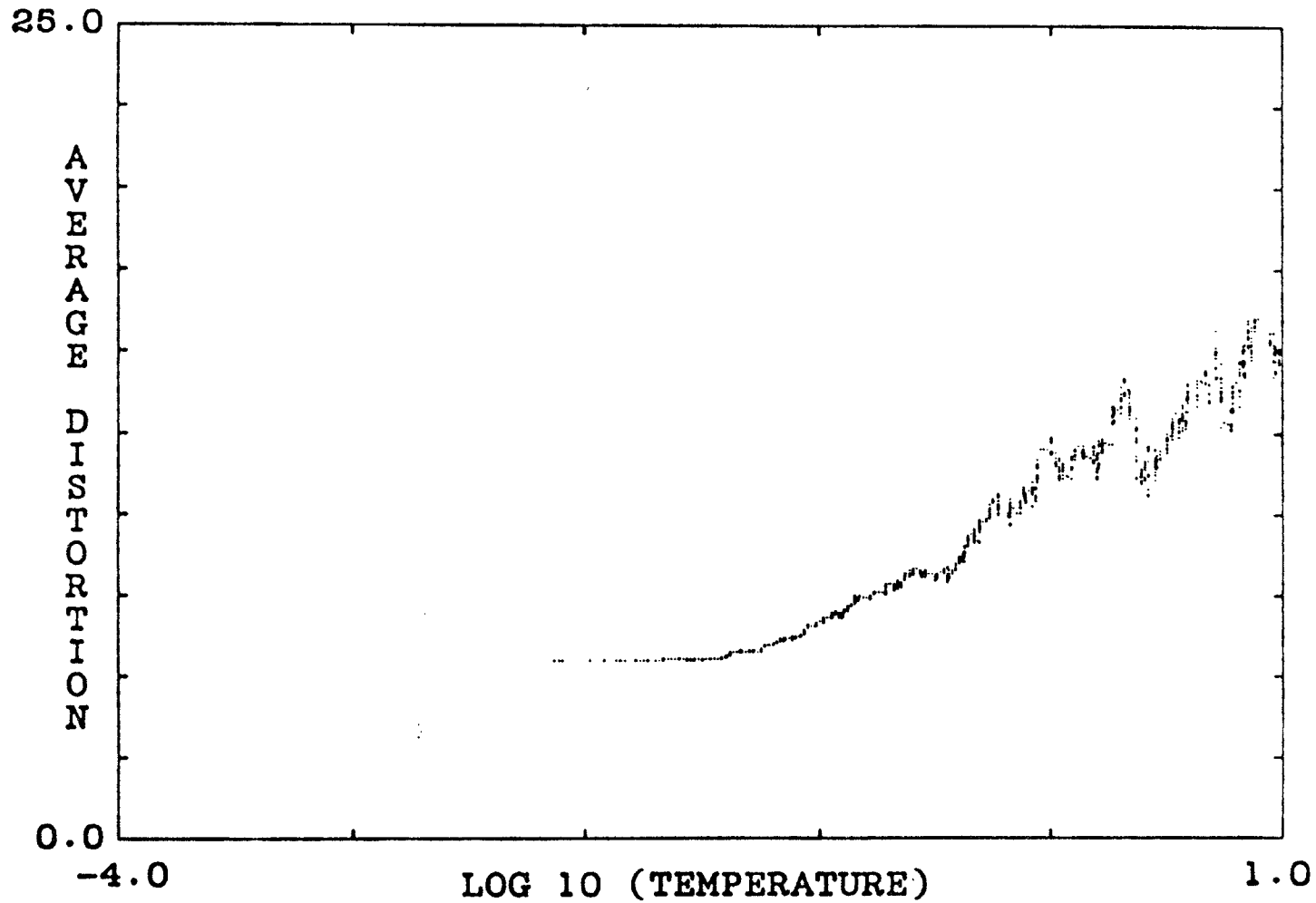

Figure 2

The Evolution of the Channel MSE During a Run of the Simulated Annealing Algorithm; $k = 6$, $\rho = 0.9$ (Correlated source).

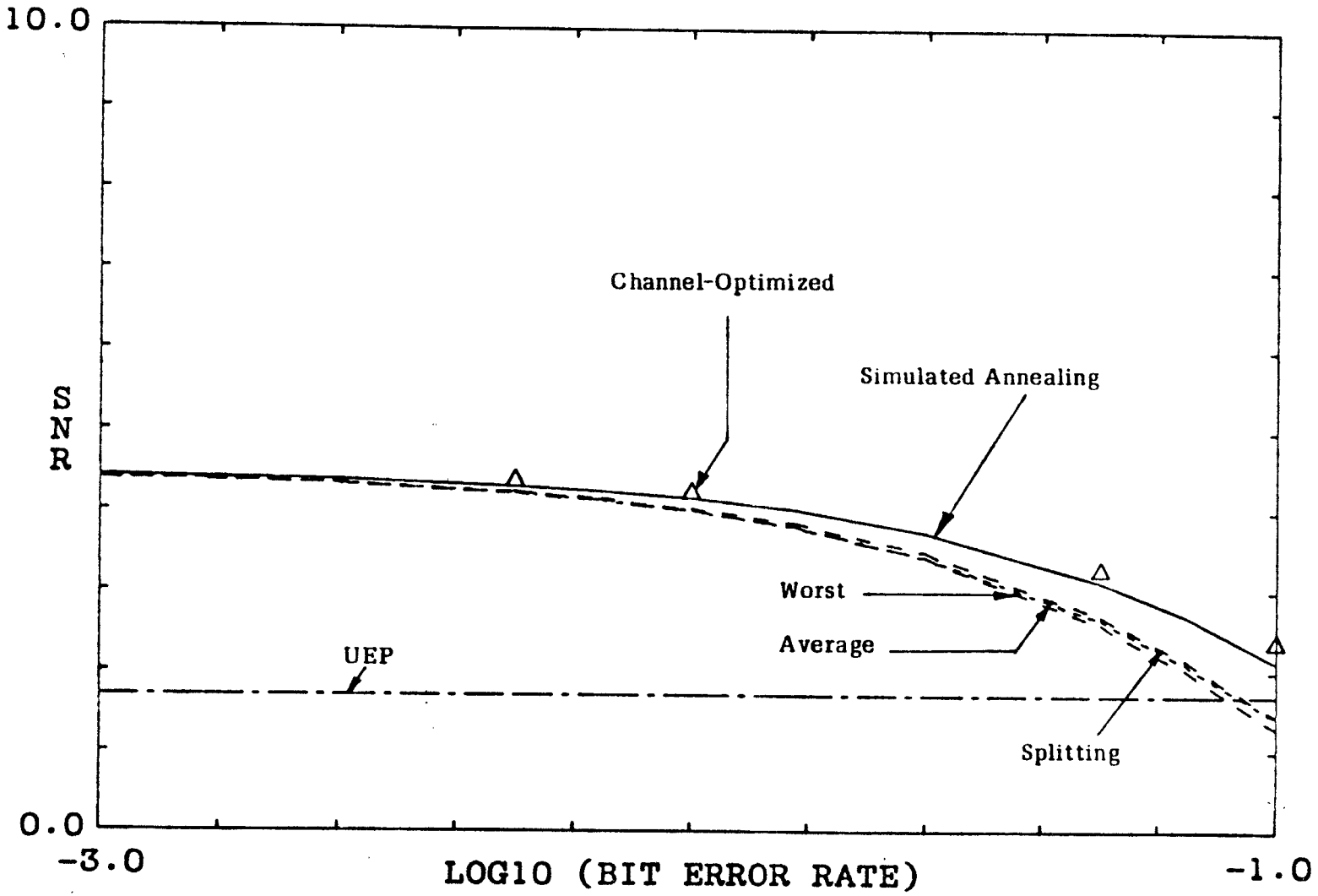

Figure 3

SNR (dB) Performance of Different Encoding Schemes on a Memoryless Gaussian Source; $k = 2$, $R = 1$ bit/sample.

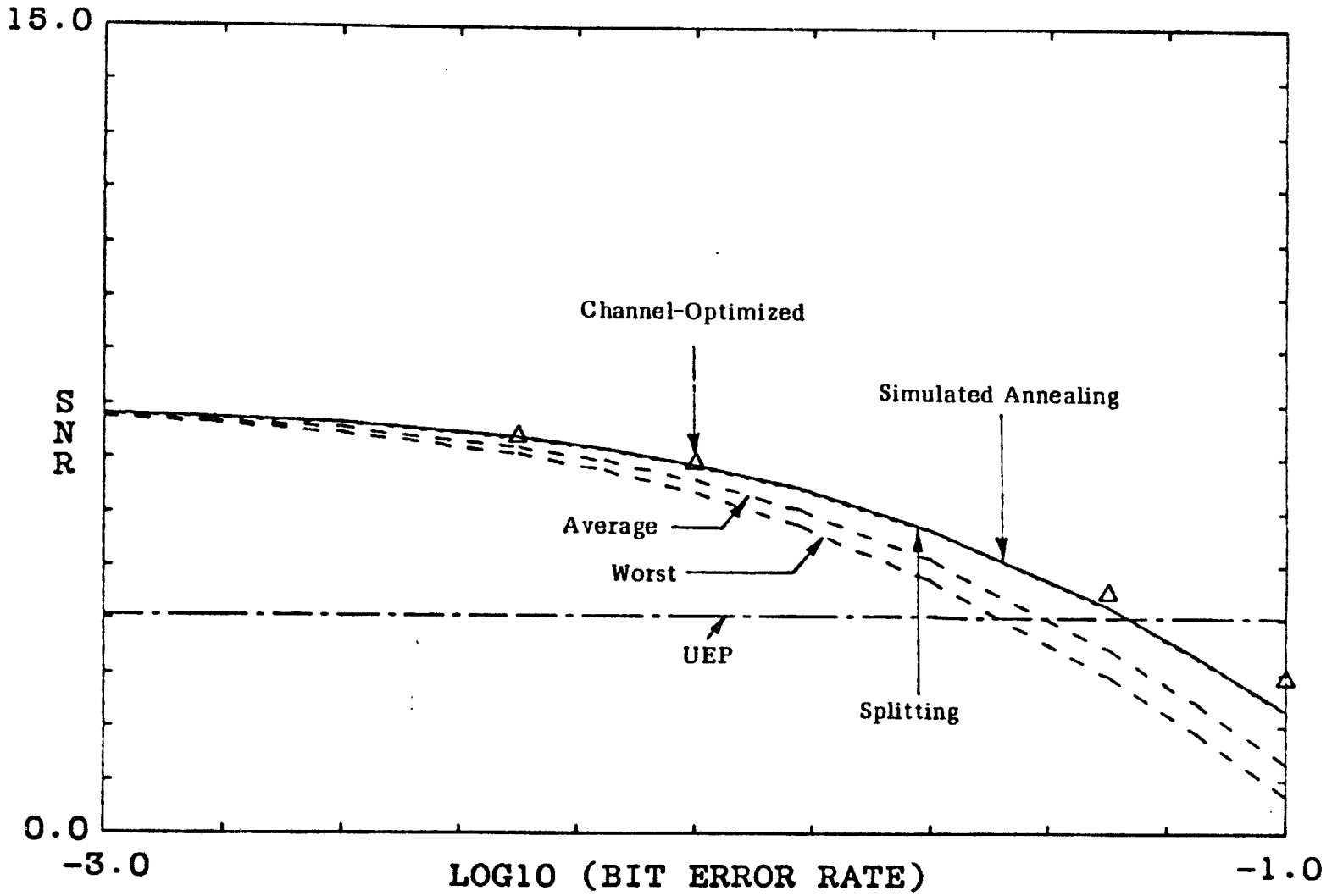

Figure 4

SNR (dB) Performance of Different Encoding Schemes on a First-Order Gauss-Markov Source; $k = 2$, $\rho = 0.9$, $R = 1$ bit/sample.

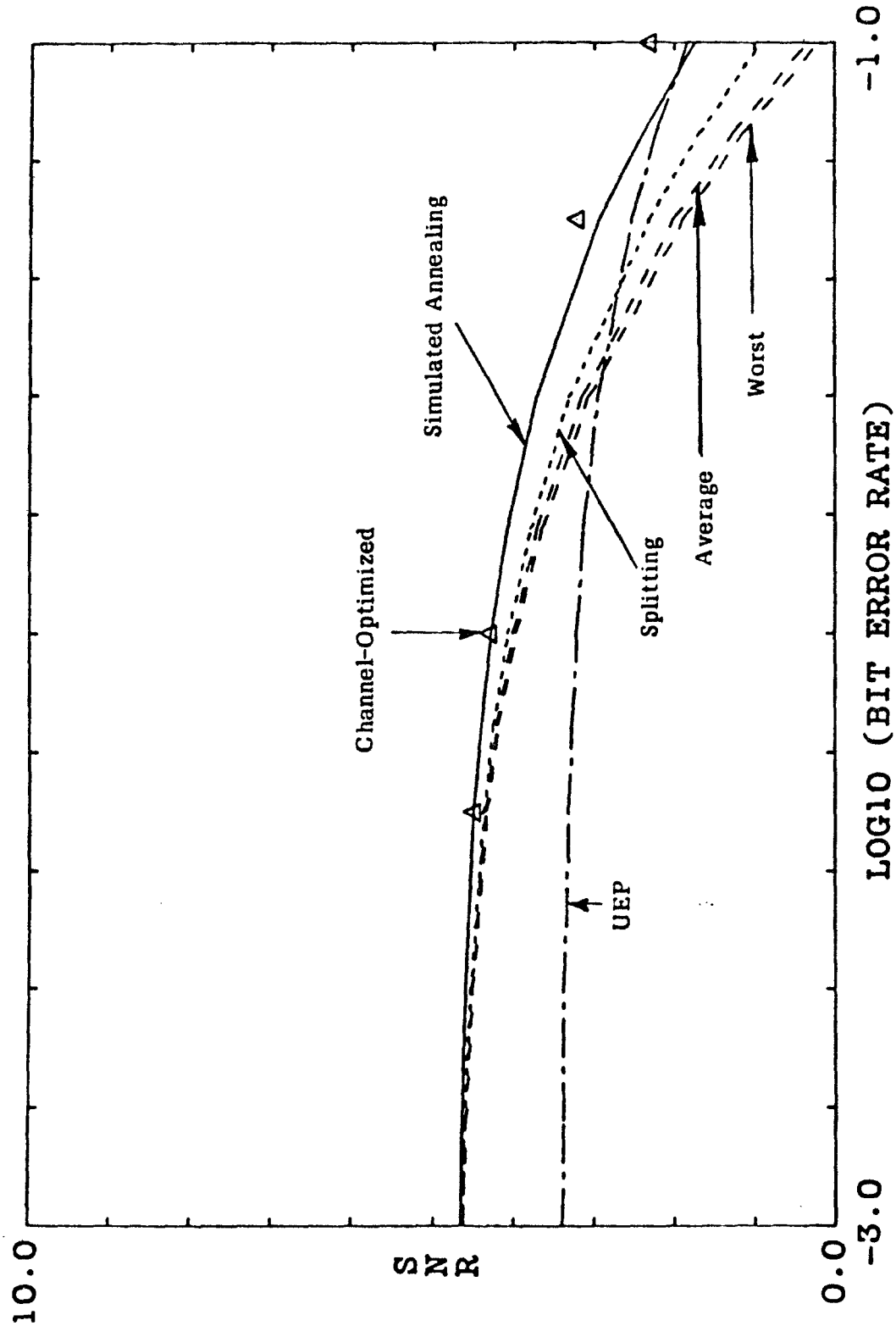

Figure 5

SNR (dB) Performance of Different Encoding Schemes on a Memoryless Gaussian Source; $k = 4$, $R = 1$ bit/sample.

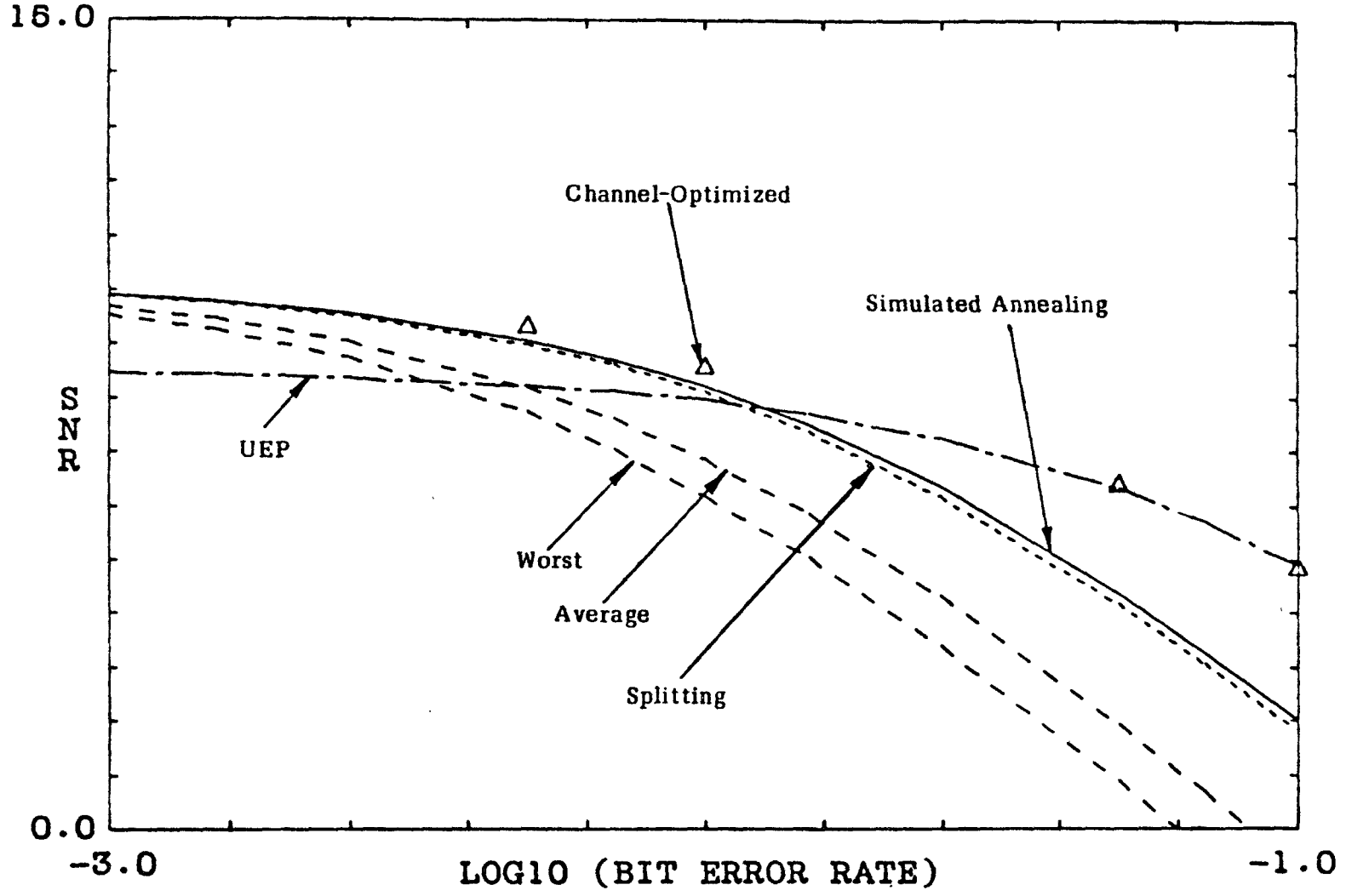

Figure 6

SNR (dB) Performance of Different Encoding Schemes on a First-Order Gauss-Markov Source; $k = 4$, $\rho = 0.9$, $R = 1$ bit/sample.

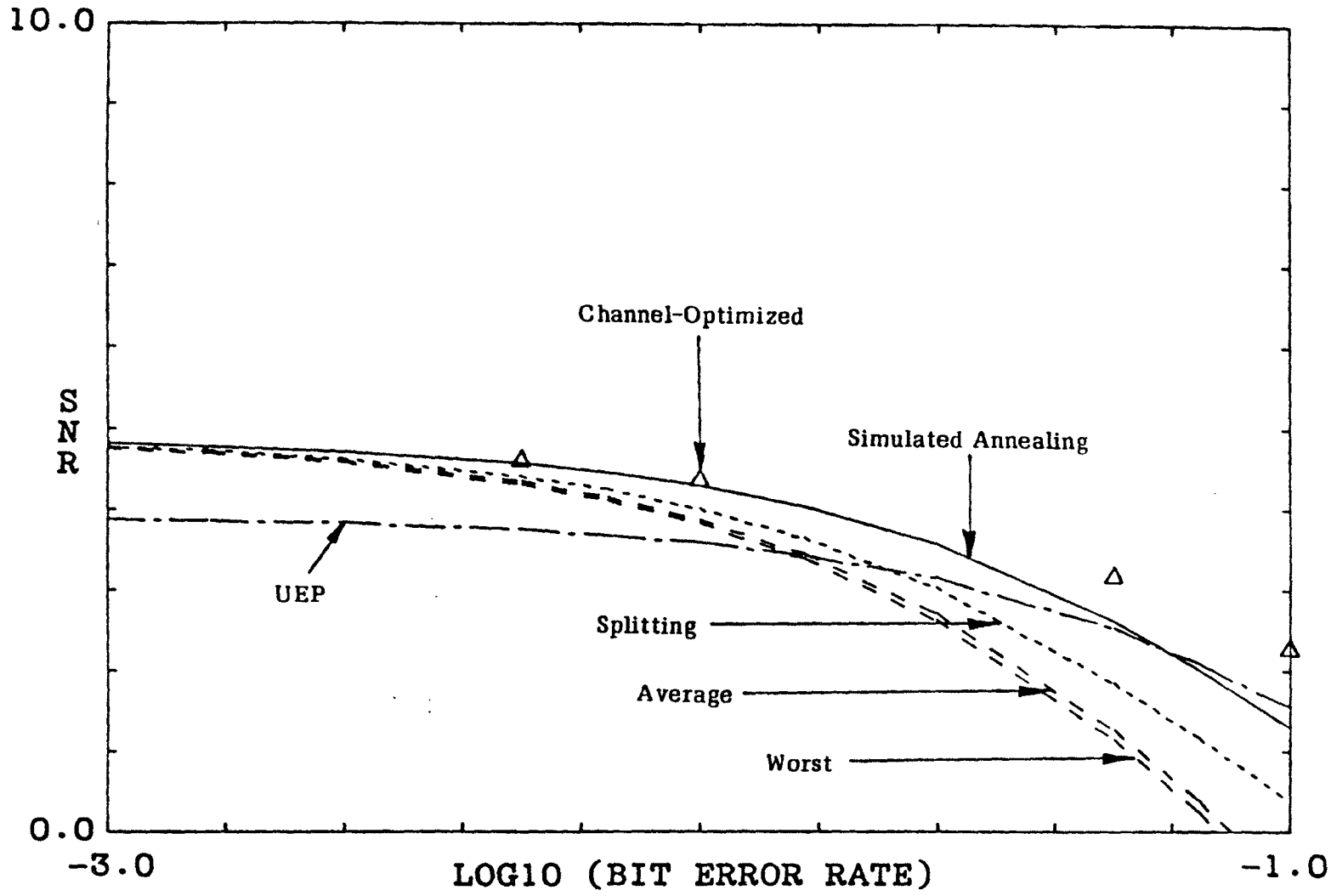

Figure 7

SNR (dB) Performance of Different Encoding Schemes on a Memoryless Gaussian Source; $k = 6$, $R = 1$ bit/sample.

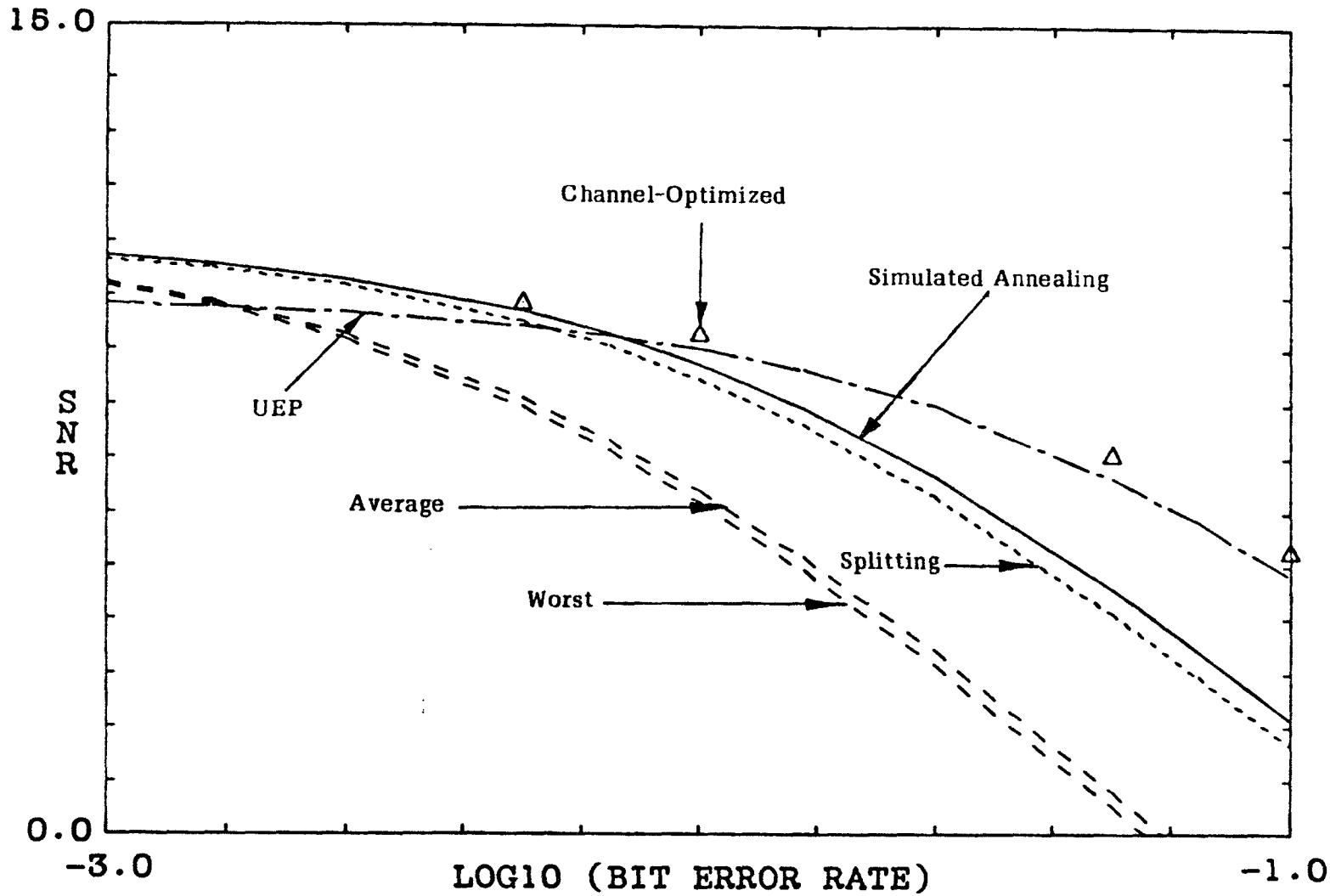

Figure 8

SNR (dB) Performance of Different Encoding Schemes on a First-Order Gauss-Markov Source; $k = 6$, $\rho = 0.9$, $R = 1$ bit/sample.

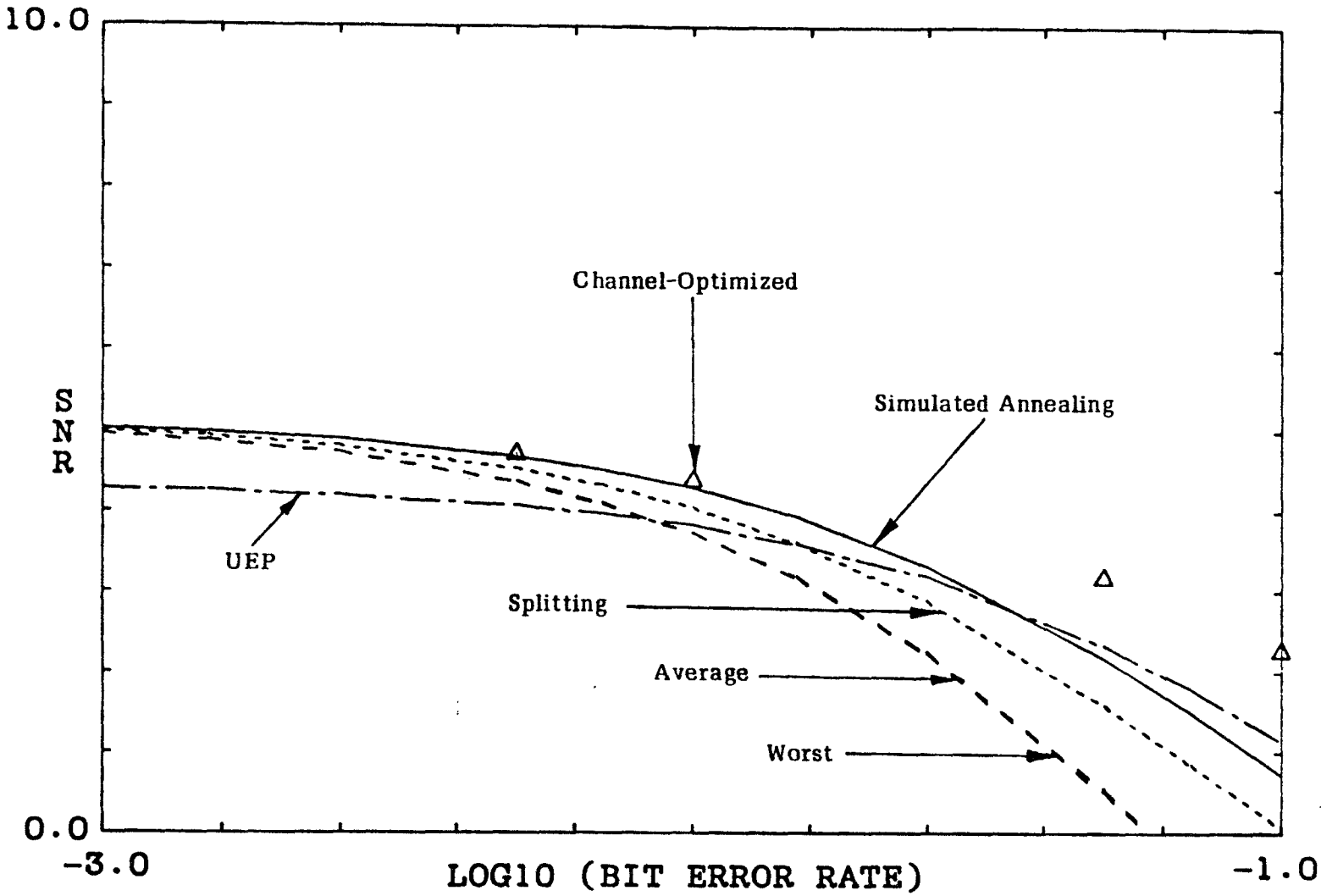

Figure 9

SNR (dB) Performance of Different Encoding Schemes on a Memoryless Gaussian Source; $k = 8$, $R = 1$ bit/sample.

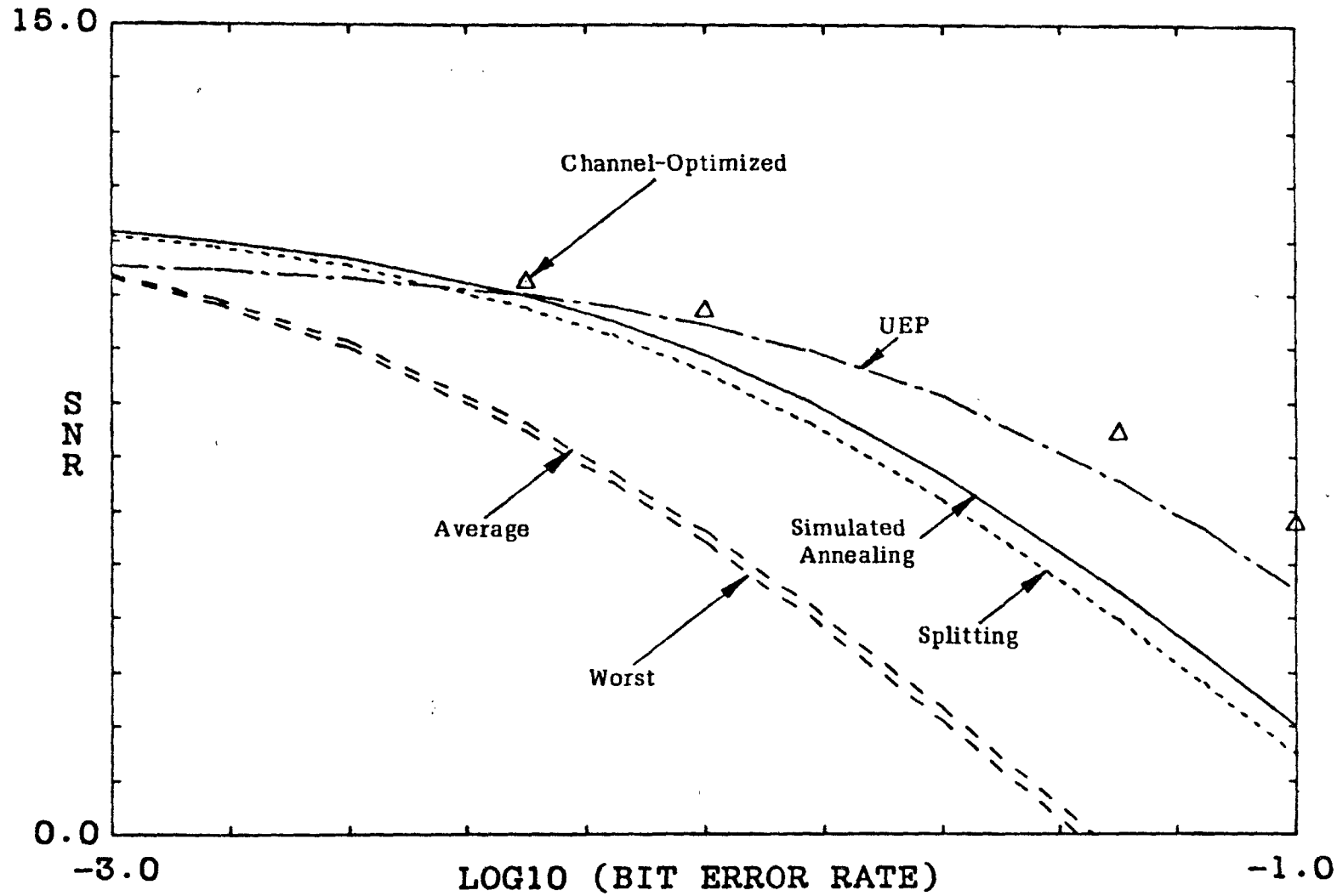

Figure 10

SNR (dB) Performance of Different Encoding Schemes on a First-Order Gauss-Markov Source; $k = 8$, $\rho = 0.9$, $R = 1$ bit/sample.

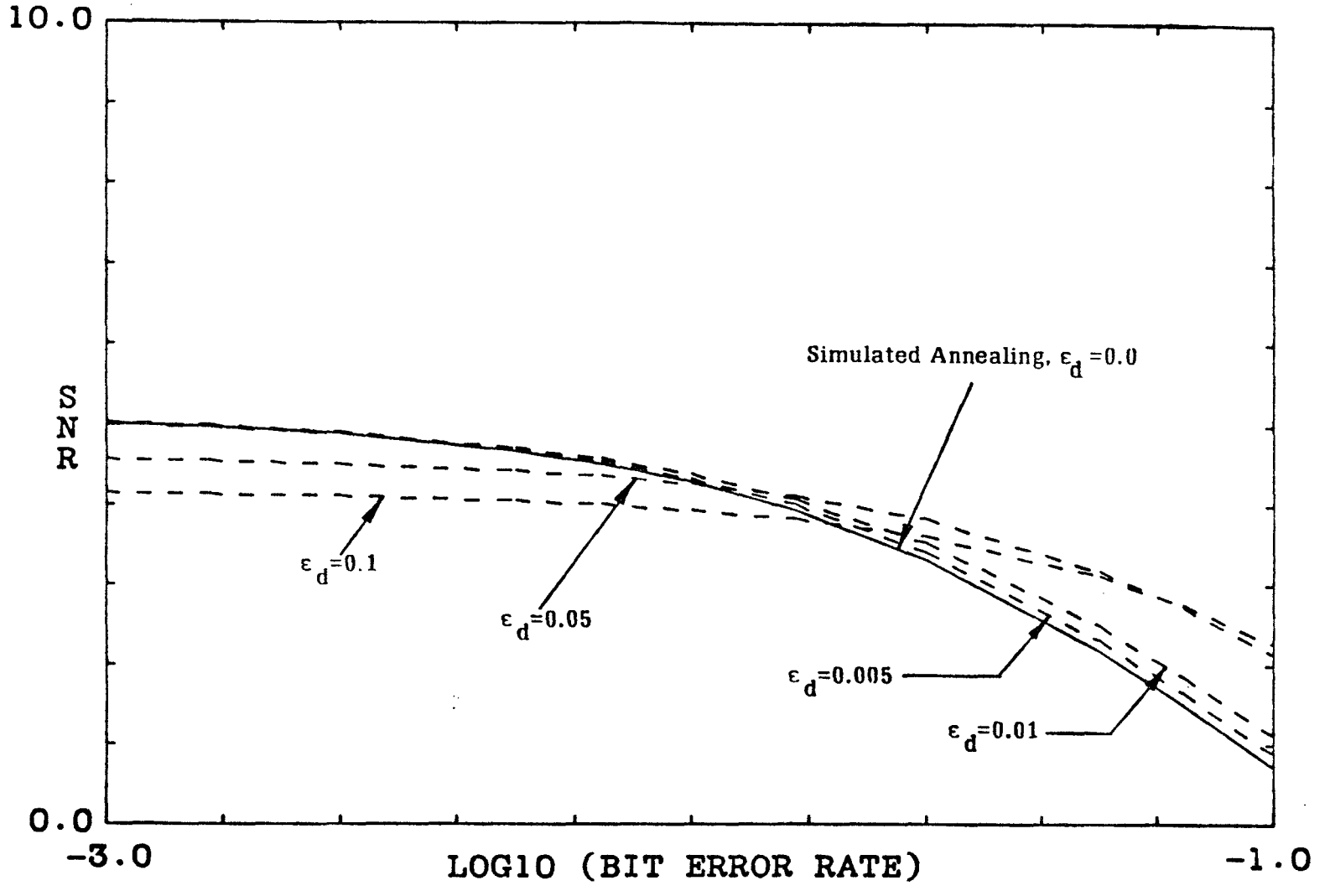

Figure 11

SNR (dB) Performance of Channel-Optimized Vector Quantizers Under Channel Mismatch Conditions for a Memoryless Gaussian Source; $k = 8$, $R = 1$ bit/sample.

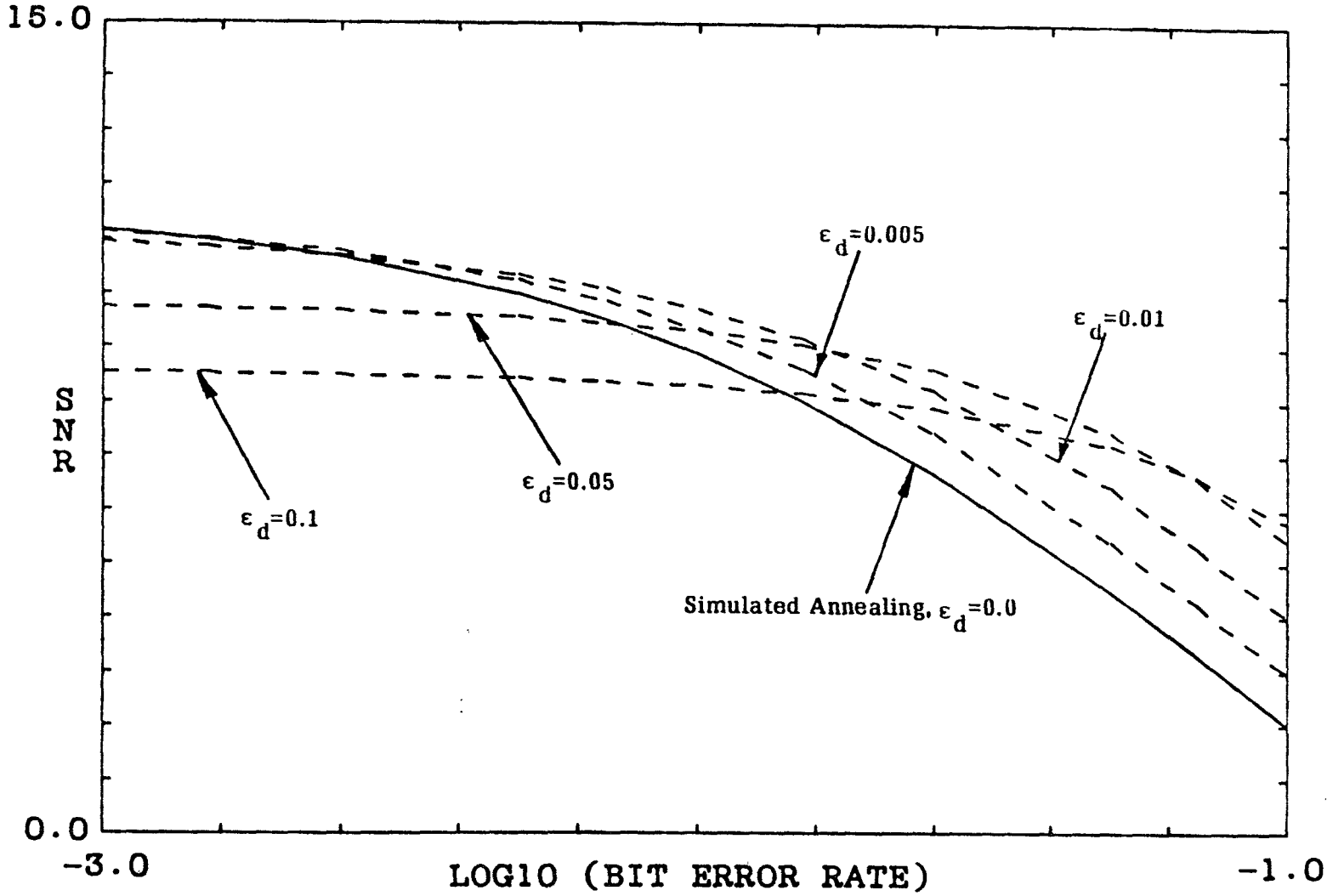

Figure 12

SNR (dB) Performance of Channel-Optimized Vector Quantizers Under Channel Mismatch Conditions for a First-Order Gauss-Markov Source; $\rho = 0.9$, $k = 8$, $R = 1$ bit/sample.