Trellis Coded Quantization of Memoryless and Gauss–Markov Sources

MICHAEL W. MARCELLIN AND THOMAS R. FISCHER

Abstract—**Exploiting the duality between modulation for digital communications and source coding, trellis coded quantization (TCQ) is developed and applied to the encoding of memoryless and Gauss–Markov sources. The theoretical justification for the approach is alphabet constrained rate distortion theory, which is a dual to the channel capacity argument that motivates trellis coded modulation (TCM). We adopt the notions of signal set expansion, set partitioning, and branch labeling of TCM, but modify the techniques to account for the source distribution, to design TCQ coders of low complexity with excellent mean squared error (MSE) performance.**

For a memoryless uniform source, TCQ provides a MSE within 0.21 dB of the distortion rate bound at all positive (integral) rates. The performance is superior to that promised by the coefficient of quantization for all of the best lattices known in dimensions 24 or less. For a memoryless Gaussian source, the TCQ performance at rates of 0.5, 1, and 2 bits/sample, is superior to all previous results we have found in the literature, including stochastically populated trellis codes and entropy coded scalar quantization.

The encoding complexity of TCQ is very modest. In the most important case, the encoding for an N-state trellis requires only 4 multiplications, $4 + 2N$ additions, N comparisons, and 4 scalar quantizations per data sample.

TCQ is incorporated into a predictive coding structure for the encoding of Gauss–Markov sources. Simulation results for first-, second-, and third-order Gauss–Markov sources (with coefficients selected to model sampled speech) demonstrate that for encoding rates of 1, 2, or 3 bits/sample, predictive TCQ yields distortions ranging between 0.75 dB and 1.3 dB from the respective distortion rate bounds.

I. Introduction

THERE are numerous parallels between the research areas of modulation theory and source coding. Both areas rely heavily on signal space concepts, both have benefitted tremendously from block and trellis coding formulations, both areas find important application for lattices, and theoretical results in each area are dominated by the Euclidean distance measure due to the typical assumption of Gaussian channel noise in the analysis of modulation systems and the typical choice of mean squared error (MSE) as the distortion measure in source coding. The past ten years have witnessed a renewed interest in modulation theory spawned by Ungerboeck's [1] formulation of coded modulation using trellis coding, and culminating quite recently in general lattice based approaches [2]–[5] to coded modulation. It is the purpose of this paper to develop the natural dual[1] to this trellis coded modulation, namely, trellis coded quantization.

Trellis coding is a proven technique for source coding, with a long history [6]–[15]. A number of theorems (e.g., [6], [7]) demonstrate the asymptotic optimality (in a rate distortion sense) of the method. Unfortunately, the source coding theorems are generally nonconstructive (except for stochastically populating the trellis) and the complexity of traditional implementations has limited the approach to generally small encoding rates (typically, rates less than about two bits/sample).

The novel feature of the trellis coded quantization approach is the use of a structured codebook with an expanded set of quantization levels. Based on Ungerboeck's notion of set partitioning, the trellis structure then prunes the expanded number of quantization levels down to the desired encoding rate. By employing a deterministic codebook, a computationally simple encoding structure is achieved. The encoder uses the Viterbi algorithm [16] and, in the most important case, the encoding requires only 4 multiplications, 4 additions, and 4 scalar quantizations per source sample, plus 2 additions and 1 comparison per trellis state per source sample. Significantly, this encoding complexity is roughly independent of the encoding rate, and is far less than the complexity of corresponding stochastically populated trellis coders.

The main contribution of the paper is to describe the construction and performance of a good class of source codes for memoryless and Gauss–Markov sources. In comparing TCQ to other results in the literature, we find that TCQ is comparable and often superior in mean squared error performance to all previous results, but with very modest computational complexity. Further, an analysis of the effect of channel errors shows that TCQ can be constructed to be surprisingly insensitive to channel errors.

The remainder of the paper begins by considering the alphabet constrained rate distortion theory formulated by Pearlman and his coauthors [8]–[10]. Motivated by this theory and Ungerboeck's trellis coded modulation [1], [17], [18], TCQ systems are then designed for memoryless uniform, Gaussian, and Laplacian sources. The computational complexity of TCQ is then described, followed by an analysis of the effect of channel errors. Finally, a predictive TCQ structure is formulated and used to design coders for first-, second-, and third-order Gauss–Markov sources.

II. Alphabet Constrained Rate Distortion Theory

Alphabet constrained rate distortion theory is, in many respects, a mixture of the theory for discrete and continuous sources. For a discrete source, a specific reproduction alphabet must be chosen in order to compute the rate distortion function, while in the continuous case, the reproduction alphabet is implicitly the entire real line. In many cases, such an uncountable output alphabet is impractical or even impossible to implement in a finite complexity source coder.

Alphabet constrained rate distortion theory was developed in a series of papers by Pearlman and his coauthors [8]–[10]. The basic

Paper approved by the Editor for Quantization, Speech/Image Coding of the IEEE Communications Society. Manuscript received February 17, 1987; revised October 10, 1988. This work was supported in part by the National Science Foundation under Grant NCR-8821764. This paper was presented in part at the 1988 International Symposium on Information Theory, Kobe City, Japan.

M. W. Marcellin is with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721.

T. R. Fischer was with the Department of Electrical Engineering, Texas A&M University. He is now with the Department of Electrical and Computer Engineering, Washington State University, Pullman, WA 99164.

IEEE Log Number 8932067.

[1] Our use of the word "dual" is as a synonym for "counterpart," and should not be confused with the mathematically more formal use of dual in dual codes or dual lattices.

0090-6778/90/0100-0082$01.00

idea is to find an expression for the best achievable performance for encoding a continuous source using a finite reproduction alphabet. The theory has been developed to the extent that the choice of an output alphabet is quite flexible. The options available when choosing an output alphabet are as follows: i) choosing only the size of the alphabet (the number of elements), ii) choosing the size and the actual values of the alphabet, iii) choosing the size, values, and the probabilities with which the values are to be used. Since case ii) will be sufficient for our purposes, we present only a simple derivation [of case ii)] that we feel is more straightforward and intuitive than that in [10].

Let X be a source, producing independent and identically distributed (i.i.d.) outputs according to some continuous probability density function (pdf) fx. Consider prequantizing X with a high rate scalar quantizer to obtain the "source" U taking values in $\{a_1, a_2, \cdots, a_K\}$ with probabilities $P(a_1), P(a_2), \cdots, P(a_K)$. Consider further, encoding U as $\hat{X}$ where $\hat{X}$ takes values in $\{b_1, b_2, \cdots, b_J\}$. For notational convenience, we define the prequantization noise as $Q = U - X$. The distortion of the overall system is then

$$E[(X-\hat{X})^2] = E[(U-Q-\hat{X})^2]$$
$$= E[Q^2] + E[(U-\hat{X})^2] - 2E[Q(U-\hat{X})].$$

Clearly,

$$E[Q(U-\hat{X})] = \sum_{k=1}^{K}\sum_{j=1}^{J}\int q(a_k - b_j) f(q|a_k, b_j) P(a_k, b_j)\, dq.$$

Since $f(q|a_k, b_j) = f(q|a_k)$, we have

$$E[Q(U-\hat{X})] = \sum_{k=1}^{K} P(a_k) \sum_{j=1}^{J} (a_k - b_j) P(b_j|a_k) \int q f(q|a_k)\, dq$$
$$= \sum_{k=1}^{K} P(a_k)(a_k - E[\hat{X}|a_k]) E[Q|a_k]. \quad (1)$$

For the Lloyd-Max quantizer [19], [20], $E[Q|a_k] = 0$ and hence,

$$E[(X-\hat{X})^2] = E[(X-U)^2] + E[(U-\hat{X})^2].$$

Using the Blahut algorithm for discrete sources [21], we can compute the distortion rate function $D_U(R)$ for the discrete source U and reproduction alphabet $\{b_1, b_2, \cdots, b_J\}$. We define the alphabet constrained distortion rate function as

$$D_C(R) = E[(X-U)^2] + D_U(R).$$

It is a simple matter to show that a source coder exists that achieves performance arbitrarily close to $D_C(R)$ while encoding X using symbols in $\{b_1, b_2, \cdots, b_J\}$, because the rate distortion theory of discrete sources guarantees the existence of a coder to encode U using symbols in $\{b_1, b_2, \cdots, b_J\}$ while achieving performance arbitrarily close to $D_U(R)$. So, the desired encoder for X is constructed by simply cascading a Lloyd-Max quantizer with such a coder.

Figs. 1 and 2 show the alphabet constrained rate distortion functions for the i.i.d. uniform and Laplacian sources with 8 bit Lloyd-Max prequantization. The alphabet constrained rate distortion functions for the i.i.d. Gaussian source are similar. In each case, the output alphabet (of size J) is the set of output points for the J level Lloyd-Max quantizer. Also shown in these figures are the corresponding Shannon lower bounds and Lloyd-Max quantizer performance at various points of interest.

The alphabet constrained rate distortion function can be computed in a similar manner for non-Lloyd-Max prequantizers. The quantity in (1) will no longer be zero, but can be computed using information available from the Blahut algorithm used to compute $D_U(R)$. For the fixed output alphabet case examined here, our development is more general than that in [10] where the prequantization was taken to be uniform.

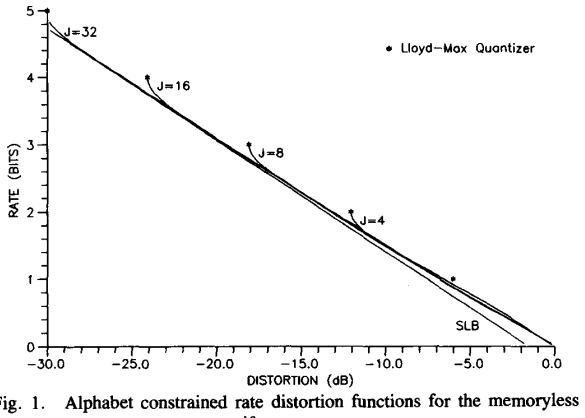


Fig. 1. Alphabet constrained rate distortion functions for the memoryless uniform source.

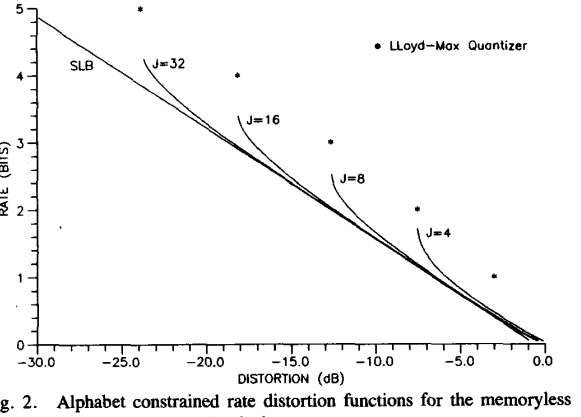


Fig. 2. Alphabet constrained rate distortion functions for the memoryless Laplacian source.

It should be noted that the alphabet constrained rate distortion function is not actually a rate distortion function in the strictest sense: we have shown that a coder exists with performance arbitrarily close to $D_C(R)$, but we have not established that no coder can have better performance. In fact, choosing a finer prequantizer (higher rate) generally decreases $D_C(R)$. However, with sufficiently high rate prequantization, $D_C(R)$ is very near the unconstrained rate distortion function for rates and reproduction alphabet sizes that will be of interest here. It should also be pointed out that this formulation is actually more indicative of practical digital communication systems where prequantization is inherent in the sampling process.

III. Trellis Coded Quantization of Memoryless Sources

Trellis coded quantization (TCQ) uses Ungerboeck's set partitioning ideas from trellis coded modulation [1], [17], [18] to achieve performance comparable (superior in most cases) to that of conventional trellis coding [8]–[12] with a fraction of the complexity of conventional trellis coding.

To transmit one of 2^m symbols per signaling interval using trellis coded modulation, the traditional 2^m point signal constellation is doubled (to 2^{m+1} points) and then partitioned into $2^{\tilde{m}+1}$ subsets where $\tilde{m}$ is an integer less than or equal to m. $\tilde{m}$ of the input bits are expanded by a rate $\tilde{m}/(\tilde{m}+1)$ convolutional code and used to select which of the subsets the channel symbol for the current signaling interval will come from. The remaining $m - \tilde{m}$ bits are used to select one of the $2^{m-\tilde{m}}$ channel symbols in the selected subset. The channel is assumed to corrupt the transmitted signal with additive

white Gaussian noise and Viterbi decoding [16] is used to find the sequence of symbols which minimizes the probability that the transmitted and received sequences are different. This coded modulation scheme performs better than conventional modulation techniques because the convolutional code and set partition are chosen in such a manner as to increase the Euclidean distance between allowable sequences of channel symbols over that in traditional modulation schemes.

Consider the decoder used for trellis coded amplitude modulation. The Viterbi algorithm is used to find the allowable sequence of channel symbols that is closest in Euclidean distance to the received sequence at the channel output. Note that the Euclidean distance between two sequences of length n, say $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$, is given by

$$d_E(\boldsymbol{x},\hat{\boldsymbol{x}}) = \sqrt{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}$$

and hence, given $\boldsymbol{x}$, finding the sequence $\hat{\boldsymbol{x}}$, that minimizes $d_E(\boldsymbol{x},\hat{\boldsymbol{x}})$ is equivalent to finding the sequence that minimizes $(1/n)d_E^2(\boldsymbol{x},\hat{\boldsymbol{x}}) = \rho_n(\boldsymbol{x},\hat{\boldsymbol{x}})$, which is the familiar squared error distortion measure. Since any set of sequences $C = \{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \cdots, \hat{\boldsymbol{x}}_K\}$, each of length n, defines a source code, the set of all allowable channel sequences and the Viterbi decoder from the coded modulation formulation can be used as a source code and corresponding source coder. Specifically, given an input sequence $\boldsymbol{x}$, the Viterbi algorithm finds the sequence in C that minimizes $\rho_n(\boldsymbol{x},\hat{\boldsymbol{x}})$.

IV. TCQ of the Memoryless Uniform Source

It can be argued that the source coder described above should perform well for the i.i.d. uniform source as follows. Consider encoding a sequence of independent realizations of a uniform random variable. Clearly, such realizations (thought of as vectors in n-dimensional space) lie in an n-dimensional hypercube and hence, the output sequences of a source coder should also lie in this hypercube. For a fixed encoding rate of R bits per sample, there are 2^{Rn} output sequences. Since source vectors are no more likely to be in one region within the hypercube than any other (of equal volume), these output sequences should be placed uniformly throughout the hypercube. A moment of thought will show that a step toward achieving this goal is to maximize the distance between output sequences (with appropriate boundary constraints due to the hypercube "sides"). The convolutional codes and set partitions in coded modulation are chosen in an attempt to do just that.

From a theoretical standpoint, trellis coded quantization of the memoryless uniform source is justified by alphabet constrained rate distortion theory. Examination of Fig. 1 will show, that for a given encoding rate of R bits per sample, it is possible to obtain nearly all of the gain theoretically possible over the R bit per sample Lloyd–Max quantizer by using an encoder with an output alphabet consisting of the output points of the $R+1$ bit per sample Lloyd–Max quantizer. Those more familiar with coded modulation will recognize the fact that the alphabet constrained rate distortion curves for source coding at a rate of R bits per sample using 2^{R+i} ($i > 0$ an integer) output symbols are a direct counterpart to Ungerboeck's channel capacity curves for transmitting data through a channel at a rate of m bits per modulation interval using 2^{m+i} channel symbols [1].

Consider encoding the memoryless source distributed uniformly on the interval $[-A, A]$ (i.e., $f_X(x) = 1/2A, x \in [-A, A]$). The Lloyd–Max (minimum MSE) scalar quantizer is the uniform quantizer with output points $\pm(2i-1)A/2^R, i = 1, 2, \cdots, 2^{R-1}$, and corresponding MSE

$$D_{SQ}(R) = \sigma_X^2 2^{-2R} \tag{2}$$

where $\sigma_X^2 = A^2/3$ is the variance of X, and R is the encoding rate in bits per sample. Comparing $D_{SQ}(R)$ to the known values of the distortion rate function [22] at encoding rates of 1, 2, and 3 bits per sample, there is only a potential for improvement over (2) of 0.77 dB, 1.17 dB, and 1.36 dB, respectively. Asymptotically (as R gets large) there is a potential for 1.53 dB improvement. In the past, improvement over (2) has been obtained by using vector quantization (using either lattices [23], [24] or the clustering algorithm [25]), tree coding, and trellis coding. Since the uniform quantizer outputs are equally likely, entropy coding is of no benefit.

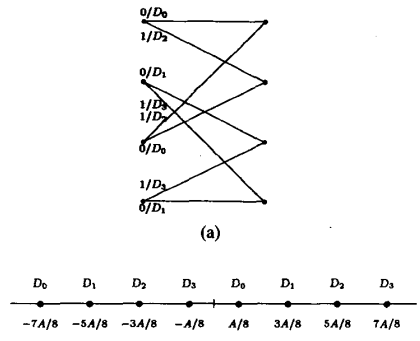


Fig. 3. (a) Ungerboeck's four-state amplitude modulation trellis. (b) Output points and partition for 2 bit per sample TCQ.

TABLE I
Trellis Coded Quantization Performance for the Memoryless Uniform Source Using Rate-$(R+1)$ Lloyd–Max Output Points. (Values are Listed as SNR in dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Lloyd-Max Quantizer	Distortion Rate Function
1	5.78	5.96	6.06	6.13	6.19	6.29	6.33	6.02	6.79
2	12.47	12.60	12.69	12.76	12.83	12.90	12.93	12.04	13.21
3	18.77	18.90	18.98	19.04	19.10	19.16	19.20	18.06	19.42
4	24.95	25.05	25.13	25.19	25.24	25.30	25.34	24.08	N/A
8	49.16	49.24	49.32	49.38	49.44	49.49	49.53	48.16	49.69
12	73.24	73.35	73.40	73.48	73.53	73.58	73.61	72.25	73.78

For encoding the memoryless uniform source at an encoding rate of R bits per sample using TCQ with any of Ungerboeck's amplitude modulation trellises [1], [18], the output alphabet can be chosen as the set of output points of the rate-$(R+1)$ Lloyd–Max quantizer. These output points are partitioned into four subsets by starting with the leftmost (most negative) point and proceeding to the right, labeling consecutive points $D_0, D_1, D_2, D_3, D_0, D_1, D_2, D_3, \cdots$ until the rightmost (most positive) point is reached. As an example, consider an encoding rate of 2 bits per sample. The output points and corresponding partition are shown in Fig. 3 along with Ungerboeck's four-state amplitude modulation trellis. Given a data sequence $\boldsymbol{x}$, the Viterbi algorithm is used to find the allowable sequence of output symbols $\hat{\boldsymbol{x}}$, that minimizes $\rho_n(\boldsymbol{x},\hat{\boldsymbol{x}})$.

The sequence of output symbols chosen by the Viterbi algorithm can be represented by the bit sequence specifying the corresponding trellis path (sequence of subsets) in addition to the sequence of $R-1$ bit codewords necessary to specify symbols from the chosen subsets. These bit sequences are transmitted through the channel (assumed error free for now) and used to reconstruct the output sequence as follows: the bit sequence that specifies the trellis path is used as the input to the convolutional coder; the output of the convolutional coder selects the proper subset; and the sequence of $R-1$ bit code words is used to select the correct symbol from each subset.

Simulation results for encoding the memoryless uniform source using TCQ with Ungerboeck's amplitude modulation trellises and the Lloyd–Max quantizer output alphabets are listed in Table I, along with the corresponding Lloyd–Max quantizer performance and distortion rate function values for various rates of interest. Each simulation was carried out by encoding 100 sequences (each of length 1000) of independent realizations from a uniformly distributed pseudo-random number generator. The values listed in Table I reflect the sample average of the 100 computed distortions.

The sample variance was also computed for these 100 distortion samples and used to compute confidence intervals on the true average distortion for each coder. Since each distortion ($\rho_n(\boldsymbol{x},\hat{\boldsymbol{x}}) =$

TABLE II
TRELLIS CODED QUANTIZATION PERFORMANCE FOR THE MEMORYLESS UNIFORM SOURCE USING DOUBLED AND OPTIMIZED OUTPUT ALPHABETS. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Lloyd-Max Quantizer	Distortion Rate Function
1	6.22	6.33	6.39	6.44	6.48	6.55	6.58	6.02	6.79
2	12.62	12.73	12.80	12.85	12.91	12.97	13.00	12.04	13.21
3	18.83	18.94	19.01	19.08	19.13	19.18	19.23	18.06	19.42

$(1/n)\sum_{i=1}^{n}(x_i - \hat{x}_i)^2)$ is a sample average itself, we can appeal to the central limit theorem and assume that the distribution of the computed distortions is approximately Gaussian. Under this assumption, confidence intervals can be computed for the true average distortion of each coder. For each coder listed in Table I, the 95 percent confidence interval is no worse than ± 0.02 dB. That is, with probability 0.95, the true average distortion for each coder is within 0.02 dB of the value listed in Table I.

Although Table I shows significant gain over Lloyd–Max quantization, there is still potential for improvement. For a given trellis, TCQ performance deteriorates as the encoding rate decreases. For example, with the 256 state trellis, TCQ performance is within 0.17 dB of the distortion rate function at high rates but falls off to 0.46 dB away at an encoding rate of 1 bit per sample. One explanation for this problem is, that at low rates, a large percentage of the allowable output sequences lie near the sides of the previously discussed hypercube. This is because a large portion of the points in the output alphabet are exterior points: that is, they have only 1 neighbor. For example, in the case of 1 bit per sample TCQ, the output points at $\pm 3A/4$ have only 1 neighbor each. There are only four output points for this case and hence 50 percent of the output points are exterior.

In an attempt to improve TCQ performance, we have developed a training sequence based numerical optimization procedure for output alphabet design. Training sequence design techniques have been used extensively in the source coding literature with, perhaps, the clustering algorithm for vector quantizer design [25] being the most widely known of such algorithms. The principle behind training sequence design algorithms is to find a source coder that works well for a given set of data samples that is representative of the source to be encoded.

Given a TCQ system (trellis, output alphabet, and partition) and a set of *fixed* data sequences to encode (a training set), the average distortion incurred by encoding these sequences can be thought of as a function of the output alphabet. For an alphabet of size $J = 2^{R+1}$, the average distortion is a function of the J symbols in the output alphabet and so maps $\mathcal{R}^J$ to $\mathcal{R}$ where $\mathcal{R}^J$ and $\mathcal{R}$ are the J-dimensional and one-dimensional Euclidean (or real) spaces, respectively. In this frame-work, optimization of the output alphabet can be carried out by any numerical algorithm which solves for a vector in $\mathcal{R}^J$ that minimizes a scalar function of J variables. Note that each time the numerical algorithm updates the output alphabet estimate, the training sequences must be reencoded to compute the resulting distortion. Hence, the design process is extremely computationally intense. For this reason, we constrained the output alphabets to be symmetric about the origin in all of our designs. This restriction cuts the number of free variables in half and greatly increases convergence speed. Since the uniform pdf is symmetric and Ungerboeck's trellises exhibit symmetry with respect to subset sequences, there is no obvious reason to suspect that this assumption will degrade performance.

Each coder was designed using a training set of 100 sequences of 1000 samples each. A quasi-Newton method subroutine was used for the optimization process. The output alphabets obtained in this manner are listed in [26] with selected cases also listed in the Appendix. For the curious reader, just to design the rate-3 coders (with the symmetry constraint) for trellis sizes 2^i, $i = 1, 2, \cdots, 8$, required more than 25 h of CPU time on a Digital Equipment Corporation VAX 8800 super-mini-computer.

Simulation results for encoding the memoryless uniform source using TCQ with the optimized output alphabets appear in Table II. Again, 100 sequences of length 1000 (different than the ones used as training data) were processed by each coder and used to compute sample average distortion and sample distortion variance. The 95 percent confidence intervals were computed and found to be no worse than ± 0.02 dB. Note that the SNR's in Tables I and II are comparable for an encoding rate of 3 bits per sample. Optimization at an encoding rate of 4 bits per sample produced no significant performance increase.

TABLE III
MAXIMUM ACHIEVABLE PERFORMANCE INCREASE OVER LLOYD–MAX QUANTIZATION FOR THE MEMORYLESS UNIFORM SOURCE AT HIGH ENCODING RATES. (VALUES ARE LISTED AS SNR IN dB)

Lattice D_4	E_8	Λ_{16}	Λ_{24}	Distortion Rate Function
.37	.65	.86	1.03	1.53

TABLE IV
TRELLIS CODED QUANTIZATION PERFORMANCE FOR THE MEMORYLESS GAUSSIAN SOURCE USING RATE-$(R+1)$ LLOYD–MAX OUTPUT POINTS. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Lloyd-Max Quantizer	Distortion Rate Function
1	4.64	4.78	4.86	4.93	4.99	5.06	5.08	4.40	6.02
2	10.19	10.31	10.37	10.44	10.50	10.55	10.59	9.30	12.04
3	15.86	15.96	16.02	16.10	16.17	16.21	16.25	14.62	18.06
4	21.66	21.78	21.84	21.92	21.96	22.02	22.05	20.22	24.08

The performance of TCQ with the optimized output alphabets (for encoding rates of 3 bits per sample and less) and the Lloyd–Max alphabets (for higher rates) is very good. For the simple four-state trellis, the sample average distortion is within 0.59 dB of the distortion rate function at all rates tested. This performance is better than is theoretically possible for any VQ of dimension less than 15. This is easily verified by evaluating the asymptotic quantizer bound [38], [39],

$$D_n(R) = \frac{e[\Gamma(1+n/2)]^{2/n}D(R)}{(1+n/2)}$$

for $n = 1, 2, \cdots, 15$ where $\Gamma(\cdot)$ is the familiar gamma function, n is the VQ dimension, and $D(R)$ is the distortion rate function. For the 256 state trellis, the sample average distortion is within 0.21 dB of the distortion rate function at all rates tested. This performance is better than all results we have found in the literature. In particular, the performance is better than that promised by the best lattices discovered to date in up to 24 dimensions [27] and is superior to trellis coding results for stochastic codebooks obtained by Wilson [11]. Table III lists the asymptotic performance of vector quantizers based on the D_4, E_8, Λ_{16}, and Λ_{24} lattices [27]. The values are listed as gains above Lloyd–Max quantization performance. It should be pointed out that these results are only valid for high rate coders and a certain amount of deterioration in performance is expected at low rates, due to the "edge effects" of truncating the lattice. Note that the results in Tables I and II are superior to the asymptotic lattice performance for rates as small as 3 bits per sample. In fact, evaluation of the asymptotic VQ bound indicates that no VQ of dimension less than 69 can exceed the MSE performance of 256 state TCQ.

V. TCQ OF THE MEMORYLESS GAUSSIAN AND LAPLACIAN SOURCES

Although we have no intuitively pleasing distance property arguments to justify using TCQ for memoryless Gaussian and Laplacian sources, alphabet constrained rate distortion theory indicates that a substantial performance increase over the Lloyd–Max quantizer is possible. For encoding the memoryless Gaussian and Laplacian sources at an encoding rate of R bits per sample, we can again choose the respective rate-$(R+1)$ Lloyd–Max quantizer output points as the output alphabet for TCQ. The results of encoding simulations carried out in the same manner as in the uniform case using the appropriate pseudo-random number generators are listed in Tables IV and V. Here also, 95 percent confidence intervals were calculated and found to be no worse than ± 0.03 dB and ± 0.10 dB for the Gaussian and Laplacian results, respectively.

TABLE V
TRELLIS CODED QUANTIZATION PERFORMANCE FOR THE MEMORYLESS LAPLACIAN SOURCE USING RATE-$(R+1)$ LLOYD–MAX OUTPUT POINTS. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Lloyd-Max Quantizer	Distortion Rate Function
1	3.66	3.75	3.83	3.91	3.96	4.00	4.04	3.01	6.62
2	8.81	8.93	9.01	9.08	9.14	9.18	9.23	7.54	12.66
3	14.35	14.47	14.55	14.62	14.68	14.74	14.77	12.64	18.68
4	20.11	20.23	20.31	20.38	20.43	20.49	20.53	18.13	N/A

TABLE VI
TRELLIS CODED QUANTIZATION PERFORMANCE FOR THE MEMORYLESS GAUSSIAN SOURCE USING DOUBLED AND OPTIMIZED OUTPUT ALPHABETS. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Lloyd-Max Quantizer	Distortion Rate Function
1	5.00	5.19	5.27	5.34	5.43	5.52	5.56	4.40	6.02
2	10.56	10.70	10.78	10.85	10.94	10.99	11.04	9.30	12.04
3	16.19	16.33	16.40	16.47	16.56	16.61	16.64	14.62	18.06

TABLE VII
TRELLIS CODED QUANTIZATION PERFORMANCE FOR THE MEMORYLESS LAPLACIAN SOURCE USING DOUBLED AND OPTIMIZED OUTPUT ALPHABETS. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Lloyd-Max Quantizer	Distortion Rate Function
1	4.34	4.47	4.56	4.63	4.72	4.78	4.83	3.01	6.62
2	9.41	9.56	9.64	9.70	9.79	9.85	9.90	7.54	12.66
3	14.87	15.00	15.03	15.13	15.22	15.27	15.37	12.64	18.68

TABLE VIII
PERFORMANCE OF VARIOUS SOURCE CODING SCHEMES FOR THE MEMORYLESS GAUSSIAN SOURCE. (VALUES ARE LISTED AS SNR IN dB)

Rate	TCQ 4 Sts.	TCQ 256 Sts.	Ent. Cod. Quant.	Clust. Alg. VQ (Dim. 6)	Wilson 32 Sts.	Wilson 128 Sts.	Pearlman 512 Sts.
1	5.00	5.56	4.64	4.79	4.76	5.47	5.50
2	10.56	11.04	10.55	N/A	10.32	10.87	N/A
3	16.19	16.64	16.56	N/A	15.58	16.78	N/A

The optimization procedure discussed in the previous section was also implemented for the Gaussian and Laplacian TCQ systems. The resulting output alphabets are tabulated in [26] with selected cases listed in the Appendix. The results of encoding simulations using these alphabets are listed in Tables VI and VII. The 95 percent confidence intervals are no worse than ± 0.05 dB and ± 0.15 dB for the Gaussian and Laplacian results, respectively.

The performance of the optimized output alphabet coders for the Gaussian source is comparable and in most cases superior to all results we have seen in the literature. Performance results for several source coding schemes for the memoryless Gaussian source are listed in Table VIII. The systems included are: TCQ; entropy coded quantization [28]; clustering algorithm based vector quantization [29]; and stochastically populated trellis coders designed by Wilson [11] and Pearlman [9].

The only source coder which outperforms the 256 state TCQ system is Wilson's 128-state trellis coder for an encoding rate of 3 bits per sample. This trellis coder was populated stochastically from a continuous output alphabet. As we shall see, this leads to a complexity that is significantly higher than that of TCQ. It should also be pointed out that TCQ performs quite well for modest trellis sizes. For example, the four-state TCQ systems outperform Wilson's 32-state systems. In fact, four-state TCQ outperforms entropy coded quantization at encoding rates of 1 and 2 bits per sample.

The performance of the Laplacian coders is rather disappointing. Although substantial improvement has been made over the Lloyd–Max quantizer, the average distortion is still far from the distortion rate function. For both the Gaussian and Laplacian sources, the performance diverges away from the distortion rate function as the rate grows. Since the analogy between TCQ and coded modulation weakens for nonuniform sources, this result is not especially surprising.

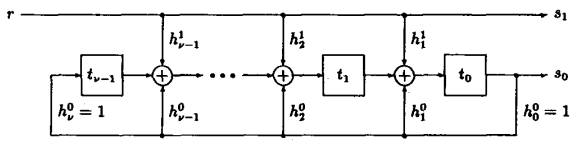


Fig. 4. Rate-1/2 systematic convolutional encoding circuit.

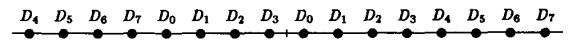


Fig. 5. Output alphabet and partition for 2 bit per sample quadrupled alphabet TCQ.

This deterioration of performance at high rates was the motivation to examine trellises other than those in [1], [18]. Ungerboeck's amplitude modulation trellises were obtained by searching over all convolutional codes with encoding circuits of the form shown in Fig. 4, where $\boldsymbol{h}^0 = (h^0_\nu, h^0_{\nu-1}, \cdots, h^0_0)$ and $\boldsymbol{h}^1 = (h^1_\nu, h^1_{\nu-1}, \cdots, h^1_0)$ are the parity check coefficients and 2^ν is the number of trellis states. Encoders of this form are called *systematic* because one of the output bits is identical to the input bit. The parity check coefficients of the codes with the best distance properties for coded modulation are tabulated in [1], [18] (the table in [1] contains some errors).

After performing simulations for all coders of the type in Fig. 4 with up to 64 states, we concluded that little could be gained over Ungerboeck's trellises. In several cases, there were trellises that performed better than Ungerboeck's, but the improvement was insignificant.

VI. QUADRUPLED OUTPUT ALPHABETS FOR THE LAPLACIAN SOURCE

Careful examination of Fig. 2 will show that for the Laplacian source, doubling the signal constellation is not sufficient to achieve performance close to the rate distortion function. For example, at a rate of 2 bits per sample, the alphabet constrained rate distortion function with $2^3 = 8$ output values is roughly 1 dB away from the rate distortion function. It is clear that quadrupling the size of the output alphabet brings the alphabet constrained rate distortion function very near the rate distortion function. For this reason, an attempt was made to design codes with a four-fold output alphabet expansion.

Following Ungerboeck's formulation with the number of output points quadrupled gives rise to an eight-way partition and a rate-1/3 convolutional code. Unfortunately, labeling the output points of the rate-$R+2$ Lloyd–Max quantizer $D_0, D_1, \cdots, D_7, D_0, D_1, \cdots$ from left to right and searching over a large class of convolutional codes led to no substantial performance increase over the doubled alphabet systems. It is our feeling that this problem arises from the fact that all output symbols receive equal emphasis in the resulting coders. Clearly, for nonuniform sources, this is undesirable. The set partitioning and trellis branch labeling need to be modified so that less probable output values are assigned to fewer trellis branches than more probable output values. This approach is akin to that used in stochastically populated trellis coders and entropy coding. For the Laplacian source, the outermost symbols (furthest from the origin) should be used less frequently than the innermost symbols (closest to the origin).

Consider encoding the Laplacian source at a rate of 2 bits per sample using the output points of the rate-4 Lloyd–Max quantizer. In order to implement the ideas discussed above, we partition the output alphabet into 8 subsets and divide these subsets into two classes. D_0-D_3 are assigned to the "more probable" class and D_4-D_7 are assigned to the "less probable" class. This grouping leads to a labeling such as that shown in Fig. 5. Similar alphabets and partitions can be constructed for encoding rates of 1 bit per sample and 3 bits per sample.

It is easily shown that for the alphabet and partition of Fig. 5, realizations of the Laplacian source are closer to a point in the more probable class than to any point in the less probable class about 85 percent of the time. Using subsets from the more probable class seven times more frequently than subsets from the less likely class corresponds to using the more likely class 87.5 percent of the time. A ratio of 6 to 1 matches the 85 percent probability more closely but the 7 to 1 ratio implies that 7 out of every 8 branch labels would come from the more likely class while 1 out of every 8 branch labels would come from the less likely class. Since the number of branches labeled $D_i (i = 0, 1, 2, 3)$ in all of Ungerboeck's amplitude modulation trellises with 16 or more states is a multiple of 8, it is possible that good codes might result by replacing 1 out of every 8 subsets labeled $D_i (i = 0, 1, 2, 3)$ by a subset labeled $D_j (j = 4, 5, 6, 7)$.

After careful scrutiny of Ungerboeck's trellises and many simulation studies, a number of interesting properties of TCQ become apparent. After using the Viterbi algorithm to encode a lengthy source sequence, the distortions associated with all the survivor paths are very nearly equal. Under the assumption that the distortions of all survivors are exactly equal at each stage of the encoding process, we will show that the step in the Viterbi algorithm, in which all paths passing through the same node but the survivor are eliminated, is equivalent to a scalar quantization process. This assumption is not exactly valid, but is approximately true after the algorithm has progressed far into the trellis. We make this assumption here because it leads to valuable insight into how trellises should be labeled for use with a quadrupled output alphabet.

In the Viterbi algorithm, $\rho(x, D_{ij})$ [where D_{ij} is the jth element of subset D_i] is calculated for each branch and then added to the distortion of the survivor path from which it emanates. All but one of the paths that go to a common next state are eliminated with the one having the lowest overall distortion being the one retained. Under the assumption that all previous survivor distortions are equal, the path retained is simply the one with the smallest branch distortion $\rho(x, D_{ij})$. This is simply a scalar quantization with the branch letters of all the branches entering a common node used as output points.

Examination of all the TCQ systems discussed previously will show that there are only two of these "quantizers" in each system. One consists of the output points of D_0 together with those of D_2 and the other consists of the output points of D_1 and D_3. Thus, each "quantizer" consists of every other point from the overall output alphabet and hence, each is still reasonable as a scalar quantizer. From the partition in Fig. 5, "reasonable quantizers" might consist of Q_0) the points from D_0 and D_2; Q_1) the points from D_1 and D_3; Q_2) the points from D_0 and D_4; Q_3) the points from D_1 and D_5; Q_4) the points from D_2 and D_6; and Q_5) the points from D_3 and D_7.

TCQ trellises for use with quadrupled output alphabets were designed by replacing 1 out of every 8 of the subsets labeled D_0-D_3 by one labeled D_4-D_7 in such a manner as to only have "quantizers" of the type Q_0-Q_5. Many such labelings are possible but the ones which achieve the best performance seem to be those that exhibit a high degree of symmetry among the resulting subset sequences (a property also present in the doubled alphabet TCQ systems).

The characteristics of the trellises designed in this manner are tabulated in [26]. It should be noted that although these trellises do not necessarily correspond to a convolutional code, they can still be specified by a finite state machine. In fact, the only difference between the machines to implement these new trellises and the machines to implement Ungerboeck's trellises is the output function: for a given binary input sequence, the state transitions are identical.

The results of simulation studies carried out using 100 sequences of length 1000 each appear in Table IX. In this case, the 95 percent confidence intervals are no worse than ± 0.10 dB. Note that an increase in performance over the doubled output alphabet coders has been obtained in every case. In fact, the performance of the quadrupled alphabet TCQ systems using Lloyd–Max quantizer values exceeds that of the doubled output alphabet coders (even using optimized alphabets) by at least 0.4 dB in every case with some coders showing improvements of nearly 1 dB.

TABLE IX
TRELLIS CODED QUANTIZATION PERFORMANCE FOR THE MEMORYLESS LAPLACIAN SOURCE USING RATE-$(R+1)$ LLOYD–MAX OUTPUT POINTS. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 16	32	64	128	256	Lloyd-Max Quantizer	Distortion Rate Function
1	4.90	5.13	5.29	5.44	5.47	3.01	6.62
2	10.08	10.35	10.66	10.77	10.86	7.54	12.66
3	15.36	15.65	16.15	16.16	16.33	12.64	18.68

TABLE X
TRELLIS CODED QUANTIZATION PERFORMANCE FOR THE MEMORYLESS LAPLACIAN SOURCE USING QUADRUPLED AND OPTIMIZED OUTPUT ALPHABETS. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 16	32	64	128	256	Lloyd-Max Quantizer	Distortion Rate Function
1	4.92	5.13	5.35	5.49	5.54	3.01	6.62
2	10.47	10.73	10.98	11.16	11.22	7.54	12.66
3	16.20	16.43	16.79	16.84	16.96	12.64	18.68

TABLE XI
PERFORMANCE OF VARIOUS SOURCE CODING SCHEMES FOR THE MEMORYLESS LAPLACIAN SOURCE. (VALUES ARE LISTED AS SNR IN dB)

Rate	TCQ 16 Sts.	TCQ 256 Sts.	Ent. Cod. Quant.	Clust. Alg. VQ (Dim. 6)	Wilson* 64 Sts.	Wilson* 128 Sts.	Pearlman 512 Sts.
1	4.92	5.54	5.76	4.98	5.21	N/A	5.15
2	10.47	11.22	11.31	N/A	N/A	11.45	N/A
3	16.20	16.96	17.20	N/A	15.47	16.84	N/A

* within accuracy of graph in [11]

Of course, the optimization procedure can be used here to obtain performance even better than that of the quadrupled alphabet Lloyd–Max TCQ systems. The resulting output alphabets are listed in [26]. The results of encoding simulations using these optimized output alphabets are presented in Table X. The corresponding 95 percent confidence intervals are no worse than ± 0.09 dB.

The performance of the quadrupled and optimized output alphabet coders for the Laplacian source is comparable and in most cases superior to other results we have seen in the literature. Performance results for several source coding schemes for the memoryless Laplacian source are listed in Table XI. The systems included are: quadrupled alphabet TCQ; entropy coded quantization [28]; clustering algorithm based vector quantization [29]; and stochastically populated trellis coders designed by Wilson [11] and Pearlman [9].

For each rate examined, the performance of entropy coded quantization exceeds the 256 state TCQ performance by roughly 0.2 dB. It should be pointed out, however, that entropy coded quantization is particularly prone to error propagation and also to buffer overflow in practical implementations [30]. As will be shown in subsequent sections, these factors are not a significant problem for TCQ. Wilson's 128-state trellis coder is slightly superior in performance to the 256-state TCQ system at an encoding rate of 2 bits per sample, but again, the complexity is substantially higher for Wilson's scheme. As in the Gaussian case, TCQ performance is quite good even for modest size trellises.

VII. COMPUTATIONAL BURDEN

In this section, we derive expressions for the computational requirements of TCQ systems. As we shall see, the computational burden of TCQ is far less than that of most source coders with comparable performance.

Recall that for each coder discussed, every subset contains 2^{R-1} elements. Clearly, the 2^{R-1} parallel branches associated with each subset come from a common node, and hence, have the same "survivor distortion." Therefore, the best path within a given subset can be determined by a rate-$(R-1)$ scalar quantization operation. After this quantization operation has been performed for each subset, only two branches entering each node remain in contention and hence, each "new survivor" can be computed by adding the distortions from the

scalar quantizations to the associated "old survivors" and choosing the path associated with the smaller of these two sums.

For each of the doubled output alphabet and quadrupled output alphabet coders there are 4 and 8 subsets, respectively. In either case, the scalar quantization need only be performed once per subset per sample (not each time the subset appears in the trellis). Hence, for the doubled alphabet coders, the computations required per data sample are 4 rate-$(R-1)$ quantizations with associated distortion calculations, $2N$ adds and N two-way compares where N is the number of trellis states or nodes. In the quadrupled case, the 4 rate-$(R-1)$ quantizations and associated distortion calculations are replaced by 8 of the same. Each distortion calculation, $(x-D_{ij})^2$ can be computed with 1 add (subtract) and 1 multiply. Hence, the computational burden of TCQ is 4 rate-$(R-1)$ scalar quantizations, 4 multiplies, $2N+4$ adds, and N two-way compares in the doubled alphabet case and 8 rate-$(R-1)$ scalar quantizations, 8 multiplies, $2N+8$ adds, and N two-way compares in the quadrupled alphabet case.

The exponential growth in computations with R present in previous trellis coders is not a problem with TCQ. In fact, TCQ complexity is roughly independent of R. In recent times (and sometimes still), the number of multiplies per data sample was used as the sole measure of complexity. Under this criterion, TCQ requires only four multiplies per data sample for the memoryless uniform and Gaussian sources, and eight multiplies per data sample for the memoryless Laplacian source. Significantly, the number of multiplies in TCQ is independent of encoding rate *and* trellis size.

The computational burden of various source coding schemes is listed in Table XII. The systems included are: TCQ (doubled alphabet and quadrupled alphabet); clustering algorithm based vector quantization [25] with full search encoding, and (generally suboptimum) binary tree search encoding; and the stochastically populated trellis coders designed by Wilson [11] and Pearlman [9]. Each value in Table XII is listed as the number of operations per data sample with subtractions being counted as additions. The number of compares in each case represents the number of two-way compares required: it is easily shown that one I-way compare can be implemented with $I-1$ two-way compares and that a rate-R scalar quantization can be implemented with R two-way compares. As before, N is the number of trellis states, R is the encoding rate in bits per sample, and J is the alphabet size for Pearlman's trellis coders. Note that J must be greater than 2^R for performance exceeding that of the Lloyd-Max quantizer.

The complexity of TCQ is clearly far less than that of any other scheme listed in Table XII except for tree searched VQ (at low rates and dimensions). Recalling the performance results from previous sections we see that TCQ yields performance comparable (or superior) to previous source coders with a fraction of the complexity required by previous schemes. It should be pointed out that the computational requirements of entropy coded quantization are quite modest; however, error propagation and buffer overflow problems [30] are of great concern in many practical applications. It should also be noted that the complexity of the TCQ source coder/decoder pair is concentrated at the coder. The decoder can be implemented with a shift register and lookup table. This property is desirable in certain data storage systems such as the compact disk where source coding can be done "off-line" but play-back must be done in "real-time."

VIII. Noninteger Encoding Rates

TCQ can be extended to non-integer encoding rates in a straightforward manner. In principle, any non-negative rational number can be used as an encoding rate for TCQ. Rational rates are achieved by encoding multiple samples per trellis branch.

Consider a source code with block length n_b and size $K=2^{n_s+1}$. Consider also, partitioning the K sequences (codewords) from this source code into four subsets each containing 2^{n_s-1} sequences of length n_b and assigning the subsets to trellis branches as before. In the encoding process, the scalar quantizations associated with each subset become vector quantizations and n_b data samples are encoded per state transition. Since 1 bit per state transition is required to specify the sequence of subsets and n_s-1 bits are required to specify a sequence from each subset, the encoding rate is n_s/n_b bits per sample.

TABLE XII
Computational Requirements of Various Source Coding Schemes for Memoryless Sources

Coder Type	Multiplies	Adds	Compares
TCQ (doubled alphabet)	4	$2N+4$	$N+4(R-1)$
TCQ (quadrupled alphabet)	8	$2N+8$	$N+8(R-1)$
Clust. Alg. VQ (Dim. n)			
Full Search	2^{Rn}	2^{Rn}	$(2^{Rn}-1)/n$
Binary Tree Search	$2Rn$	$2Rn$	R
Wilson	$N2^R$	$N2^{R+1}$	$N(2^R-1)$
Pearlman	J	$N2^R+J$	$N(2^R-1)$

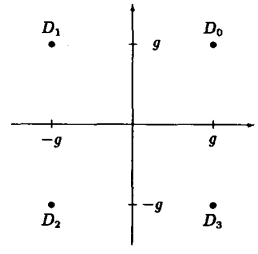


Fig. 6. Output sequences for rate-1/2 TCQ.

TABLE XIII
Rate-1/2 TCQ Performance for the Memoryless Uniform, Gaussian, and Laplacian Sources Using Rate-1 Lloyd-Max Output Points in 2-D Sequences. (Values are Listed as SNR in dB)

Source	Trellis Size (States)						
	4	8	16	32	64	128	256
Uniform	2.60	2.74	2.80	2.88	2.95	3.02	3.07
Gaussian	2.35	2.44	2.49	2.54	2.60	2.65	2.68
Laplacian	1.95	2.01	2.04	2.08	2.11	2.14	2.16

TABLE XIV
Rate-1/2 TCQ Performance for the Memoryless Uniform, Gaussian, and Laplacian Sources Using Optimized 2-D Sequences. (Values are Listed as SNR in dB)

Source	Trellis Size (States)						
	4	8	16	32	64	128	256
Uniform	2.84	2.96	3.01	3.09	3.14	3.21	3.24
Gaussian	2.49	2.57	2.61	2.66	2.71	2.75	2.78
Laplacian	2.01	2.07	2.09	2.12	2.15	2.18	2.20

For example, an encoding rate of 1/2 bit per sample can be achieved by choosing a source code with $K=4$ ($n_s=1$) output sequences each of length $n_b=2$. One choice of output sequences and corresponding partition is shown in Fig. 6. In this case, each subset contains only 1 sequence and hence, the corresponding vector quantization is trivial: whenever a particular subset is chosen, its only element is selected as the output sequence. In general, there will be 2^{n_s-1} sequences in each subset and the "vector quantizer" will find the one closest (in Euclidean distance) to the input sequence segment (of length n_b) currently being encoded.

Simulation studies were carried out with an encoding rate of 1/2 bit per sample for the memoryless uniform, Gaussian, and Laplacian sources. In each case, the output sequences of the vector quantizer were $\{(-g,-g),(-g,g),(g,-g),(g,g)\}$ where $\pm g$ are the output points of the corresponding rate-1 Lloyd-Max scalar quantizer. The results of these simulations are given in Table XIII. The 95 percent confidence intervals are no worse than ± 0.02, ± 0.025, and ± 0.04 dB for the uniform, Gaussian, and Laplacian cases, respectively.

In Fig. 6, each output sequence is the same distance from each axis (i.e., g from above). Each of the TCQ systems can be optimized with

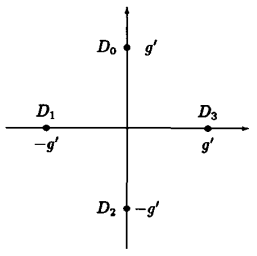


Fig. 7. Rotated output sequences for rate-1/2 TCQ.

respect to g in a manner similar to that of previous sections. The resulting values of g are listed in [26] and the corresponding simulation results appear in Table XIV. The 95 percent confidence intervals corresponding to the uniform, Gaussian, and Laplacian sources are no worse than ± 0.02, ± 0.03, and ± 0.045 dB, respectively.

The set of output sequences shown in Fig. 6 are reasonable for the uniform source where the two-dimensional pdf is constant over a square and also for the Gaussian source where the pdf is spherically symmetric. In the Laplacian case however, the contours on which the two-dimensional pdf is constant are diamonds (in higher dimensions, pyramids). This is the motivation for choosing the output sequences and partition shown in Fig. 7. We call the sequences in Fig. 7 the *rotated* two-dimensional sequences to distinguish them from those in Fig. 6. Note that the sequences in Fig. 7 can be obtained (within a scale factor) by rotating Fig. 6 by 45°.

In Fig. 7, the output sequences are $\{(0, g'), (0, -g'), (g', 0), (-g', 0)\}$ and hence, optimization over g' can be carried out as before. The resulting values of g' are given in [26] and the corresponding simulation results appear in Table XV. The 95 percent confidence intervals are no worse than ± 0.045 dB.

The performance of TCQ at rate-1/2 is quite good. In particular, the performance for the Gaussian source surpasses that of all results we have found in the literature ([31], [32], for example) with a fraction of the complexity of previous coders. For an encoding rate of 1/2 bit per sample, TCQ can easily be shown to require only two multiplies, $N+4$ adds and $(N/2)$ two-way compares per data sample, where again, N is the number of trellis states. Table XVI lists the performance and encoding complexity for Schroeder and Sloane's permutation code [31] and the Golay code based scheme proposed by Adoul and Lamblin [32]. The eight-state TCQ system achieves performance comparable to both of these schemes while requiring only two multiplies, 12 adds, and four compares per data sample. Considering the fact that commercially available digital signal processing chips are capable of performing multiplies as fast as adds and compares, the complexity of TCQ is far less than that of these other two schemes.

TABLE XV
RATE-1/2 TCQ PERFORMANCE FOR THE MEMORYLESS LAPLACIAN SOURCE USING THE ROTATED AND OPTIMIZED 2-D SEQUENCES. (VALUES ARE LISTED AS SNR IN dB)

Trellis Size (States)						
4	8	16	32	64	128	256
2.18	2.26	2.29	2.33	2.38	2.42	2.44

TABLE XVI
PERFORMANCE OF VARIOUS SOURCE CODING SCHEMES FOR THE MEMORYLESS GAUSSIAN SOURCE FOR AN ENCODING RATE OF 1/2 BIT PER SAMPLE. (VALUES ARE LISTED AS SNR IN dB)

Coder Type	SNR	Adds	Compares
Permutation	2.57	320	128
Golay	2.53	66	85

IX. ERROR PROPAGATION

In this section, we lift the assumption of error free channels and examine the effects of transmission errors on TCQ performance. We show that the trellises associated with Ungerboeck's coding circuits are particularly susceptible to error propagation when used for TCQ systems. This is not particularly a problem, however, because in each case, coding circuits exist with equivalent MSE performance while guaranteeing good error propagation properties.

Recall that each trellis used in our doubled alphabet TCQ systems arises from a convolutional coder (finite state machine) of the form shown in Fig. 4. Recall also that the sequence of bits transmitted through the channel can be divided into two sequences, one that is used as the input to the state machine, and one that is used to select symbols (or vectors in the noninteger rate case) from the subset selected by the output of the state machine. Note that the bits used to select symbols from the subsets have no effect on the state of the machine and hence, no effect on future outputs from the source decoder. However, the bits used as the input to the state machine directly determine the next state and hence, the next subset. Clearly, a transmission error in these bits can affect multiple outputs from the source decoder. Examination of Fig. 4 will show that a channel error of this type has the potential to cause all subsequent source decoder outputs to be in error. This problem is due to feedback in the encoding circuit: any erroneous values in the binary storage elements are used (along with the input bits) to compute subsequent values of the storage elements. Clearly, errors can propagate in this manner indefinitely.

Fortunately, there is a solution to this problem. For every convolutional code used here, there exists a feedback free encoding circuit for which any given input bit can affect no more than $1 + \log_2 N$ outputs where N is the number of trellis states [33]. It should be noted that the definition of a convolutional code is similar to that of a source code in that only output sequences are specified. The input/output relationship of any given encoding circuit is not unique, but the set of allowable output sequences is. Hence, any encoding circuit will do for our purposes, because the set of output sequences of the encoding circuit specifies the source code. The only difference between TCQ systems using encoding circuits with feedback and those using encoding circuits without feedback is that the bit sequence used to specify a particular sequence of subsets is different for the two cases. This difference is enough, however, to guarantee that the corresponding TCQ system will handle channel errors in a graceful manner. For example, a channel error can affect no more than three source decoder outputs for a four-state trellis and no more than nine-source decoder outputs for a 256-state trellis. In this respect, TCQ is preferable to systems such as entropy coding and large dimensional vector quantization where a single channel error can destroy an entire block of data. Note also, that if it is desired that only 1 source decoder output be affected by any single channel error, only 1 out of every R bits must be protected.

For integer encoding rates of 2 bits per sample or greater, even these "short" subset sequence errors are less serious than might be expected. For example, consider the rate-2, doubled output alphabet case. The output alphabet and corresponding partition are shown in Fig. 8. Each subset contains a positive element and a negative element. If the bit labeling used for selecting elements from a given subset corresponds to the sign of the element (for example, 0 for a negative element and 1 for a positive element), then the sign of the element chosen by the decoder will be correct even if the subset is incorrect. Clearly, the magnitude of any such error is bounded by $(D_3^+ - D_0^+)$ where D_0^+ and D_3^+ are the positive elements of D_0 and D_3, respectively. For higher rates, the bit assignment can be chosen so that the problem of incorrect subset selection is even less serious.

It is a simple matter to show that similar results hold for rational rate TCQ and quadrupled alphabet TCQ. In the former case, we use the same trellises as in the doubled alphabet case and in the latter, the trellises arise from a simple modification of the output function of the convolutional encoding circuit in the doubled alphabet case.

X. PREDICTIVE TCQ OF SOURCES WITH MEMORY

A. The Predictive TCQ Structure

Consider the differential pulse code modulation (DPCM) structure. Each time a data sample is to be encoded, a prediction of that

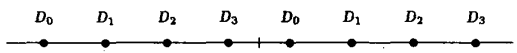


Fig. 8. Output alphabet and partition for rate-2 doubled alphabet TCQ.

data sample is formed using quantized versions of previous data samples. The difference between the data sample and the prediction (the prediction error or residual) is quantized, represented as bits, and transmitted through the channel. The quantized value of the current data sample is formed (by both the source coder and source decoder) by adding the quantized prediction residual to the prediction.

Recall from previous discussions on the TCQ of memoryless sources, that paths through the trellis specify sequences of output symbols. Hence, at each stage in the encoding, each survivor path specifies a sequence of output symbols. These output symbols are approximations of the corresponding data samples.

Consider now, a predictive TCQ system with each survivor path specifying a sequence of numbers which serve as approximations to previous data samples. Clearly, a prediction can be formed for each trellis node using the output symbols specified by the survivor path associated with that particular node. The prediction residual is formed at each node as the difference between the current data sample and the prediction associated with that node. For each branch emanating from a given node, a scalar quantization is performed to find the element of the corresponding subset which is closest to the prediction residual associated with that node. The distortions incurred by performing these scalar quantizations for each branch are added to the "old" survivor distortions and a two-way compare is carried out at each "next" node to determine the "new" survivors. Finally, the encoded value of the data sample is computed for each new survivor path as the sum of the predicted value from the appropriate old survivor and the corresponding quantized prediction residual.

Given a data sequence to be encoded, say $\boldsymbol{x} = \{x_1, x_2, \cdots, x_n\}$, the ith step in this encoding process can be described more precisely as follows. Let the survivor path ending at node k (at time $i-1$) be called *survivor-k* and let $\hat{x}_{i-j}^k$, $k = 1, 2, \cdots, N$, be the approximation (encoded value) of x_{i-j}, $j = 1, 2, 3, \cdots$, associated with survivor-k where N is the number of trellis states. Let $\hat{x}_{i|i-1}^k$ denote the predicted value of the current data sample x_i given $\hat{x}_{i-j}^k$, $j = 1, 2, 3, \cdots$, and let $d_i^k = (x_i - \hat{x}_{i|i-1}^k)$ be the prediction residual associated with survivor-k. Finally, let $\rho_{i-1}(\boldsymbol{x}, \boldsymbol{\hat{x}}^k)$ be the distortion associated with survivor-k. As before, there are two branches labeled with subset names entering and leaving each node. We denote the subset associated with the branch leaving node k and entering node l as D_l^k. For each such subset, a scalar quantization operation is performed to determine the subset element that is closest to d_i^k. This element is denoted by $\bar{D}_l^k$. As in the memoryless case, all the elements of each subset (parallel branches) are discarded except the one selected by the scalar quantization operation. After this procedure, there are two branches entering each next node. Clearly, when the next node is node l, these branches are labeled $\bar{D}_l^{k_1}$ and $\bar{D}_l^{k_2}$ where k_1 and k_2 are the nodes from which these two branches emanate. Finally,

$$\rho_i(\boldsymbol{x}, \boldsymbol{\hat{x}}^l) = \min_{k \in \{k_1, k_2\}} (\rho_{i-1}(\boldsymbol{x}, \boldsymbol{\hat{x}}^k) + (d_i^k - \bar{D}_l^k)^2) \tag{3}$$

and

$$\hat{x}_i^l = \hat{x}_{i|i-1}^{k'} + \bar{D}_l^{k'}$$

where k' is the value of k that achieves the minimum in (3). This recursion is carried out until the end of the data sequence is reached (i.e., $i = n$).

As in the memoryless case, 1 bit per data sample is required to specify the sequence of chosen subsets and $R-1$ bits per sample are used to specify which element from the chosen subset is closest to the corresponding prediction residual. The source decoder uses these bit sequences to produce the sequence of quantized prediction residuals corresponding to survivor-k' where survivor-k' is the survivor with the smallest distortion when the end of the data sequence is reached. This sequence of quantized prediction residuals is passed through the standard DPCM decoder to obtain the encoded sequence $\hat{x}_i^{k'}$, $i = 1, 2, \cdots, n$.

This system is quite similar to that in [13] for an encoding rate of 1 bit per sample, but is unique for higher encoding rates. It should be pointed out that this algorithm is not optimal. The suboptimality arises because of the prediction. Future values of prediction residual depend heavily on the quantized values of x_i used to compute predictions. In the step corresponding to (3), an entire path history is eliminated. It is conceivable that if not discarded, this path could have attained lower overall distortion than the one retained. However, if these paths are not discarded, the resulting structure is a tree rather than a trellis and no optimal algorithm exists other than exhaustive search. For this reason, we chose to use the stated procedure.

B. Predictive TCQ of Gauss-Markov Sources

In this section, the performance and complexity of predictive TCQ (PTCQ) systems for encoding Gauss-Markov sources are evaluated. Gauss-Markov sources are defined by

$$x_i = \sum_{j=1}^{L} a_j x_{i-j} + w_i$$

where $\{a_j : j = 1, 2, \cdots, L\}$ is a set of real coefficients and w_i is an element of an i.i.d. sequence of realizations of a zero-mean, Gaussian random variable. The coefficients of the Gauss-Markov sources to be studied are listed in Table XVII along with their distortion rate functions for encoding rates of 1, 2, and 3 bits per sample. We have chosen these sources because they are often used as models for sampled speech. The coefficients of the sources referred to as AR(1), AR(2), and $AR_1(3)$ in Table XVII were calculated from the long term sample autocorrelation function values of sampled speech presented in [34]. These coefficients are often referred to as McDonald's coefficients. The source labeled $AR_2(3)$ has been used in place of $AR_1(3)$ in several publications (see [13], [35], for example). The third coefficient a_3 of the $AR_2(3)$ source is in error. This minor transposition of digits seems to have originated in [35] and was subsequently copied in other publications. We include the $AR_2(3)$ source here for the purpose of comparison to published results.

Using notation from the previous section, the prediction (of the current data sample, x_i) associated with survivor k is

$$\hat{x}_{i|i-1}^k = \sum_{j=1}^{L} a_j \hat{x}_{i-j}^k. \tag{4}$$

Using (4) and the definitions in the previous section, it is easily shown that if $\hat{x}_i^k \approx x_i$, then $d_i^k \approx w_i$. Since w_i is a realization of a Gaussian random variable, say W, we have chosen to use the output alphabets that were optimized for the Gaussian source in Section V as the output alphabets in our PTCQ systems for Gauss-Markov sources. This choice can also be justified by the results of [36] where Kolmogorov-Smirnov goodness of fit tests were performed on prediction residuals of scalar DPCM systems. It was found that prediction residuals resulting from a Gauss-Markov input source can be reasonably modeled as realizations of a Gaussian random variable.

For low rates ($R < 2$), the distortion may be large enough to cause the assumption that $d_i^k \approx w_i$ to be questionable. To partially compensate for this fact, the sample standard deviation of d_i^k was computed during simulations of the PTCQ systems and used as a scale factor for the output alphabet in subsequent simulations. Iterating on this procedure a few times led to a sample standard deviation that closely matched the scale factor used for the output alphabet. Essentially, the result of this procedure is that the scale of the output alphabet is matched to the variance of the source it is meant to encode.

In each case, the performance obtained when using the variance-matched output alphabets was superior to that obtained when using output alphabets matched to the variance of W (when the two vari-

TABLE XVII
COEFFICIENTS AND DISTORTION RATE FUNCTIONS FOR VARIOUS GAUSS-MARKOV SOURCES OF INTEREST. ($D(R)$ VALUES ARE LISTED AS SNR IN dB)

Source	Coefficients	$D(1)$	$D(2)$	$D(3)$
AR(1)	$a_1 = .9$	13.23	19.25	25.27
AR(2)	$a_1 = 1.515,\ a_2 = -.752$	14.96	21.64	27.66
$AR_1(3)$	$a_1 = 1.748,\ a_2 = -1.222,\ a_3 = .310$	14.52	22.06	28.10
$AR_2(3)$	$a_1 = 1.75,\ a_2 = -1.22,\ a_3 = .301$	14.59	22.16	28.18

ances differed). The scale factors used for each source, encoding rate, and trellis size are tabulated in [26]. Note that for an encoding rate of 3 bits per sample, the scale factor is unity in each case. This is as expected, since for encoding rates larger than about 2 bits per sample, the distortion is small enough that the prediction residual can be very closely modeled by w_i for even scalar DPCM.

As in previous sections, simulation studies were carried out using 100 sequences of 1000 samples each and Ungerboeck's amplitude modulation trellises. The results of these simulations are presented in Tables XVIII–XXI for the AR(1), AR(2), $AR_1(3)$, and $AR_2(3)$ sources, respectively. For comparison, scalar DPCM systems were also simulated using Gaussian Lloyd–Max quantizers to scale factors "optimized" as in the PTCQ coders. The 95 percent confidence intervals were computed and found to be no worse than ± 0.06 dB, ± 0.08, ± 0.09, and ± 0.09 dB for the PTCQ performance values in Tables XVIII–XXI, respectively. The 95 percent confidence intervals for the DPCM systems were slightly larger, but not more than a few hundredths of a dB.

The performance of PTCQ for these sources is quite good. For each of the 256-state PTCQ systems, the SNR is at least 2.30 dB better than the corresponding scalar DPCM system. In some cases, PTCQ outperforms scalar DPCM by nearly 5 dB. Even the four-state systems perform between 1.19 dB and 2.57 dB better than the corresponding scalar DPCM systems.

Farvardin and Modestino reported results for entropy coded DPCM systems in [37]. They found, that for highly correlated first-order Gauss–Markov sources, the performance was at least 1.5 dB away from the rate distortion function for all rates of interest here. Since the 256-state PTCQ performance for the AR(1) source is within 1.3 dB of the rate distortion function at encoding rates of 1, 2, and 3 bits per sample, we conclude that PTCQ performance is superior to that of entropy coded DPCM for this source. It should also be noted that the performance of entropy coded DPCM deteriorates at low rates while PTCQ performance improves at low rates for highly correlated first-order Gauss–Markov processes, such as the AR(1) source studied here.

As in the memoryless case, PTCQ achieves good performance with trellises of moderate size. For the AR(1) source, the performance with only 32 states is as good or better than entropy coded DPCM. PTCQ using an eight-state trellis surpasses the vector quantization (VQ) results reported in [38] for the AR(1) source at encoding rates of 1 and 2 bits per sample. PTCQ also compares favorably with the rate-1 predictive trellis waveform coders (PTWC) designed by Ayanoğlu and Gray [13] using the generalized Lloyd algorithm. For the AR(1) source, the PTCQ systems outperform all PTWC systems with trellises having 32 or less states while the PTWC systems perform slightly better for trellises having 64 and 128 states. The performance of the encoding schemes discussed above for the AR(1) source are presented in Table XXII. The systems included are PTCQ, DPCM, clustering algorithm VQ [38], and PTWC [13].

Although PTCQ performance is not uniformly better than that of Ayanoğlu and Gray's PTWC systems, the complexity of PTCQ grows much more gracefully as a function of rate than would comparable PTWC systems. As mentioned previously, PTCQ and PTWC are quite similar for an encoding rate of 1 bit per sample. In fact, it is easily shown that the computational requirements in both cases are $(L + 2)N$ multiplies, $(L + 5)N$ adds, and N compares per data sample where L is the predictor order and N is the number of trellis states. PTWC systems were only designed for encoding rates of 1 bit per sample in [13], but if PTWC were extended to higher encoding rates by the traditional method of expanding the trellis in such a manner that 2^R branches enter and leave each node, the resulting computational burden would be $(L + 2^R)N$ multiplies, $(L + 2^{R+1} + 1)N$ adds, and $(2^R - 1)N$ compares per data sample. This should be compared to the much smaller requirements of PTCQ. In particular, PTCQ requires only $(L + 2)N$ multiplies, $(L + 5)N$ adds, and $(2(R - 1) + 1)N$ compares per data sample.

Several encoding schemes for the $AR_2(3)$ source at an encoding rate of 1 bit per sample are compared in [13]. PTCQ is superior to most of them in performance. Among the schemes which outperform PTCQ are Ayanoğlu and Gray's PTWC systems and a tree coder designed by Wilson and Husain [35]. The performance of these two schemes for the $AR_2(3)$ source is listed in Table XXIII along with that for PTCQ. For the PTCQ and PTWC systems, N is the number of trellis states, and for the tree coder, N is the number of paths retained at each step of encoding while using the (M, L) algorithm for searching the tree.

Although the PTWC and tree code SNR's are larger than those for PTCQ for each trellis size (number of paths retained in the tree code case) shown, it should be noted that the difference is decreasing rapidly as the trellis size grows. For example, with $N = 4$, the corresponding PTWC and tree code SNR's are 0.44 and 0.97 dB

TABLE XVIII
PTCQ PERFORMANCE FOR THE AR(1) SOURCE. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Scalar DPCM	Distortion Rate Function
1	11.19	11.60	11.89	12.13	12.22	12.41	12.49	10.00	13.23
2	17.21	17.69	17.95	18.13	18.24	18.33	18.41	16.07	19.25
3	22.92	23.40	23.60	23.78	23.90	23.94	23.98	21.69	25.27

TABLE XIX
PTCQ PERFORMANCE FOR THE AR(2) SOURCE. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Scalar DPCM	Distortion Rate Function
1	11.58	12.14	12.68	13.05	13.26	13.47	13.64	9.39	14.96
2	18.92	19.57	20.01	20.29	20.48	20.60	20.72	17.31	21.64
3	25.00	25.63	25.92	26.10	26.27	26.36	26.40	23.74	27.66

TABLE XX
PTCQ PERFORMANCE FOR THE $AR_1(3)$ SOURCE. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Scalar DPCM	Distortion Rate Function
1	10.95	11.59	12.16	12.57	12.84	13.06	13.21	8.40	14.51
2	19.00	19.67	20.15	20.47	20.71	20.85	20.96	16.90	22.07
3	25.37	25.97	26.30	26.50	26.67	26.76	26.80	23.79	28.10

TABLE XXI
PTCQ PERFORMANCE FOR THE $AR_2(3)$ SOURCE. (VALUES ARE LISTED AS SNR IN dB)

Rate (bits)	Trellis Size (States) 4	8	16	32	64	128	256	Scalar DPCM	Distortion Rate Function
1	11.03	11.65	12.24	12.64	12.90	13.12	13.28	8.46	14.59
2	19.03	19.74	20.23	20.54	20.78	20.93	21.06	16.96	22.16
3	25.42	26.04	26.38	26.58	26.75	26.85	26.87	23.88	28.18

TABLE XXII
PERFORMANCE OF VARIOUS SOURCE CODING SCHEMES FOR THE AR(1) SOURCE. (VALUES ARE LISTED AS SNR IN dB)

Rate	TCQ 8 Sts.	TCQ 128 Sts.	Clust.Alg.VQ* Dim. 4	Clust.Alg.VQ* Dim. 7	PTWC 8 Sts.	PTWC 128 Sts.	DPCM
1	11.60	12.41	10.2	11.2	11.47	12.58	10.00
2	17.69	18.33	16.00	N/A	N/A	N/A	16.07
3	23.40	23.94	N/A	N/A	N/A	N/A	21.69

* within accuracy of graph in [38]

TABLE XXIII
PERFORMANCE OF VARIOUS SOURCE CODING SCHEMES FOR THE $AR_2(3)$ SOURCE. (VALUES ARE LISTED AS SNR IN dB)

Coder	N						
	4	8	16	32	64	128	256
PTCQ	11.03	11.65	12.24	12.64	12.90	13.12	13.28
PTWC	11.47	12.04	12.60	12.90	N/A	N/A	N/A
Tree Code[1,2]	12.00	12.40	12.45	N/A	N/A	N/A	N/A

1 values taken from table in [13]
2 (M, L) algorithm was used for the tree search with M=N

TABLE XXIV
DOUBLED AND OPTIMIZED OUTPUT ALPHABETS FOR THE MEMORYLESS UNIFORM SOURCE

Trellis Size (states)	Rate (bits)	Subset			
		D_0	D_1	D_2	D_3
4	1	−1.0509	−0.6680	0.6680	1.0509
	2	−1.4263	−1.1858	−0.6506	−0.2329
		0.2329	0.6506	1.1858	1.4263
	3	−1.5690	−1.4978	−1.2195	−1.0215
		−0.7916	−0.5480	−0.3420	−0.1113
		0.1113	0.3420	0.5480	0.7916
		1.0215	1.2195	1.4978	1.5690
256	1	−1.1172	−0.5597	0.5597	1.1172
	2	−1.4481	−1.1212	−0.6505	−0.2222
		0.2222	0.6505	1.1212	1.4481
	3	−1.5956	−1.4272	−1.1970	−0.9709
		−0.7482	−0.5369	−0.3273	−0.1192
		0.1192	0.3273	0.5369	0.7482
		0.9709	1.1970	1.4272	1.5956

better than PTCQ respectively, but for $N = 16$, these values have already decreased to 0.36 and 0.21 dB, respectively. Also, we feel that a great deal (if not all) of the difference in performance between PTCQ and PTWC for small trellis sizes can be eliminated by using iterative procedures for optimizing the predictor and output alphabet similar to those used in designing the PTWC systems in [13].

XI. SUMMARY

A new source coding scheme for memoryless sources called trellis coded quantization (TCQ) has been developed. Simulation results for encoding the memoryless uniform, Gaussian, and Laplacian sources using TCQ were presented and computational requirements were summarized. A procedure was given for optimizing TCQ systems for encoding memoryless sources. A complete listing of the designs resulting from this optimization procedure for the memoryless uniform, Gaussian, and Laplacian sources is given in [26] while performance results are reported within the text here. In many cases, TCQ outperforms all previous source coding techniques with a fraction of the computational requirements of many of them. The effects of channel errors were examined, and specific bounds on the number of data samples that can be affected by a single channel error were developed.

Although the simulation studies discussed in Sections III–VI were limited to a few sources, we feel that similar results can be expected for other discrete memoryless sources. In particular, for sources with relatively light tailed probability density functions (such as the uniform and Gaussian sources), we would expect to achieve near-optimal performance (in a rate distortion theory sense) with modest design effort. For sources with relatively heavy tailed probability density functions (such as the Laplacian source), we would expect near-optimal performance to be more difficult to achieve.

In Section X, a predictive TCQ (PTCQ) structure was developed for encoding sources with memory. Simulation results were presented for encoding various Gauss–Markov sources that were chosen as models for sampled speech. Computational requirements were summarized and compared with those of other source coding schemes. PTCQ was shown to perform quite well in comparison with other schemes reported in the literature.

APPENDIX

This Appendix contains tables of doubled and optimized output alphabets for TCQ of memoryless sources. These alphabets were optimized using the procedure described in Section IV. Tables XXIV, XXV, and XXVI list alphabets for the uniform, Gaussian, and Laplacian sources, respectively. For each trellis size and encoding rate, the values are listed according to the subset to which they belong.

TABLE XXV
DOUBLED AND OPTIMIZED OUTPUT ALPHABETS FOR THE MEMORYLESS GAUSSIAN SOURCE

Trellis Size (states)	Rate (bits)	Subset			
		D_0	D_1	D_2	D_3
4	1	−1.1998	−0.3804	0.3804	1.1998
	2	−1.8768	−1.0728	−0.6292	−0.1775
		0.1775	0.6292	1.0728	1.8768
	3	−2.5390	−1.7992	−1.3484	−1.0248
		−0.7732	−0.5219	−0.3203	−0.1062
		0.1062	0.3203	0.5219	0.7732
		1.0248	1.3484	1.7992	2.5390
256	1	−1.2392	−0.2239	0.2239	1.2392
	2	−1.8873	−1.0620	−0.5601	−0.1852
		0.1852	0.5601	1.0620	1.8873
	3	−2.6300	−1.8753	−1.4154	−1.0332
		−0.7416	−0.5139	−0.2790	−0.1161
		0.1161	0.2790	0.5139	0.7416
		1.0332	1.4154	1.8753	2.6300

TABLE XXVI
DOUBLED AND OPTIMIZED OUTPUT ALPHABETS FOR THE MEMORYLESS LAPLACIAN SOURCE

Trellis Size (states)	Rate (bits)	Subset			
		D_0	D_1	D_2	D_3
4	1	−1.4017	−0.1422	0.1422	1.4017
	2	−2.5920	−1.2178	−0.5342	−0.1432
		0.1432	0.5342	1.2178	2.5920
	3	−3.9243	−2.5645	−1.7392	−1.1633
		−0.8010	−0.4752	−0.2769	−0.0384
		0.0384	0.2769	0.4752	0.8010
		1.1633	1.7392	2.5645	3.9243
256	1	−1.4143	−0.0239	0.0239	1.4143
	2	−2.6622	−1.2501	−0.5119	−0.0979
		0.0979	0.5119	1.2501	2.6622
	3	−4.5300	−2.8319	−1.9161	−1.2703
		−0.8201	−0.4813	−0.2587	−0.0449
		0.0449	0.2587	0.4813	0.8201
		1.2703	1.9161	2.8319	4.5300

REFERENCES

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

[2] G. D. Forney, Jr., "Coset codes—Part I: Introduction and geometrical classification," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1123–1151, Sept. 1988 (Invited Paper).

[3] ——, "Coset codes—Part II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1151–1187, Sept. 1988 (Invited Paper).

[4] A. R. Calderbank and N. J. A. Sloan, "New trellis codes based on lattices and cosets," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 177–195, Mar. 1987.

[5] L. F. Wei, "Trellis-coded modulation with multidimensional constellations," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 483–501, July 1987.

[6] R. M. Gray, "Time-invariant trellis encoding of ergodic discrete-time sources with a fidelity criterion," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 71–83, Jan. 1977.

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

[8] W. A. Finamore and W. A. Pearlman, "Optimal encoding of discrete-time continuous-amplitude memoryless sources with finite output alphabets," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 144–155, Mar. 1980.

[9] W. A. Pearlman, "Sliding-block and random source coding with constrained size reproduction alphabets," *IEEE Trans. Commun.*, vol. COM-30, pp. 1859–1867, Aug. 1982.

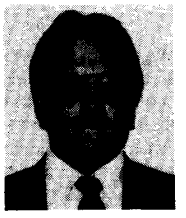
[10] W. A. Pearlman and A. Chekima, "Source coding bounds using quantizer reproduction levels," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 559–567, May 1984.

[11] S. G. Wilson, "Trellis encoding of continuous-amplitude, discrete-time information sources," Ph.D. dissertation, Univ. Washington, 1975.

[12] S. G. Wilson and D. W. Lytle, "Trellis encoding of continuous-

amplitude memoryless sources," *IEEE Trans. Inform. Theory,* vol. IT-28, pp. 211–226, Mar. 1982.
[13] E. Ayanoğlu and R. M. Gray, "The design of predictive trellis waveform coders using the generalized Lloyd algorithm," *IEEE Trans. Commun.,* vol. COM-34, pp. 1073–1080, Nov. 1986.
[14] J. B. Anderson and S. Mohan, "Sequential encoding algorithms: A survey and cost analysis," *IEEE Trans. Commun.,* vol. COM-32, pp. 169–176, Feb. 1984.
[15] L. C. Stewart, R. M. Gray, and Y. Linde, "The design of trellis waveform coders," *IEEE Trans. Commun.,* vol. COM-30, pp. 702–710, Feb. 1982.
[16] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE (Invited Paper),* vol. 61, pp. 268–278, Mar. 1973.
[17] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets—Part I: Introduction," *IEEE Commun. Mag.,* vol. 25, pp. 5–11, Feb. 1987.
[18] ——, "Trellis-coded modulation with redundant signal sets—Part II: State of the art," *IEEE Commun. Mag.,* vol. 25, pp. 12–21, Feb. 1987.
[19] S. P. Lloyd, "Least squares quantization in PCM," unpublished memorandum, Bell Laboratories, 1957; see also *IEEE Trans. Inform. Theory,* vol. IT-28, pp. 129–137, 1982.
[20] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Theory,* vol. IT-6, pp. 7–12, Mar. 1960.
[21] R. E. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Trans. Inform. Theory,* vol. IT-18, pp. 460–473, July 1972.
[22] N. S. Jayant and P. Noll, *Digital Coding of Waveforms.* Englewood Cliffs, NJ: Prentice-Hall, 1984.
[23] 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, Mar. 1982.
[24] ——, "Fast quantizing and decoding algorithms for lattice quantizers and codes," *IEEE Trans. Inform. Theory,* vol. IT-28, pp. 227–232, Mar. 1982.
[25] 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.
[26] M. W. Marcellin, "Trellis coded quantization: An efficient technique for data compression," Ph.D. dissertation, Texas A&M Univ., Dec. 1987.
[27] J. H. Conway and N. J. A. Sloane, "A lower bound on the average error of vector quantizers," *IEEE Trans. Inform. Theory,* vol. IT-31, pp. 106–109, Jan. 1985.
[28] N. Farvardin and J. W. Modestino, "Optimum quantizer performance for a class of non-Gaussian memoryless sources," *IEEE Trans. Inform. Theory,* vol. IT-30, pp. 485–497, May 1984.
[29] R. M. Dicharry, "An algorithm for adaptive vector quantizer design with real-time applications," M.S. thesis, Texas A&M Univ., Dec. 1984.
[30] N. Farvardin and J. W. Modestino, "On overflow and underflow problems in buffer-instrumented variable-length coding of fixed-rate memoryless sources," *IEEE Trans. Inform. Theory,* vol. IT-32, pp. 839–845, Nov. 1986.
[31] M. R. Schroeder and N. J. A. Sloane, "New permutation codes using Hadamard unscrambling," *IEEE Trans. Inform. Theory,* vol. IT-33, pp. 144–146, Jan. 1987.
[32] J.-P. Adoul and C. Lamblin, "A comparison of some algebraic structures for CELP coding of speech," *Conf. Proc. Int. Conf. Acoust., Speech, Signal Processing,* 1987, pp. 1953–1956.
[33] G. D. Forney, Jr., "Convolutional codes I: Algebraic structure," *IEEE Trans. Inform. Theory,* vol. IT-16, pp. 720–738, Nov. 1970.
[34] R. A. McDonald, "Signal-to-quantization noise ratio and idle channel performance of DPCM systems with particular application to voice signals," *Bell Syst. Tech. J.,* vol. 45, pp. 1123–1151, Sept. 1966.
[35] S. G. Wilson and S. Husain, "Adaptive tree encoding of speech at 8000 bits/s with a frequency-weighted error criterion," *IEEE Trans. Commun.,* vol. COM-27, pp. 165–170, Jan. 1979.
[36] Data compression class project for Dr. T. R. Fischer, Dep. Elec. Eng., Texas A&M Univ., Spring 1986.
[37] N. Farvardin and J. W. Modestino, "Rate-distortion performance of DPCM schemes for autoregressive sources," *IEEE Trans. Inform. Theory,* vol. IT-31, pp. 402–418, May 1985.
[38] 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.
[39] Y. Yamada, S. Tazaki, and R. M. Gray, "Asymptotic performance of block quantizers with difference distortion measures," *IEEE Trans. Inform. Theory,* vol. IT-26, pp. 6–14, Jan. 1980.

★



Michael W. Marcellin was born in Bishop, CA, on July 1, 1959. He received the M.S. and Ph.D. degrees in electrical engineering from Texas A&M University in 1985 and 1987, respectively. He graduated summa cum laude with the B.S. degree in electrical engineering from San Diego State University in 1983 where he was named the most outstanding student in the College of Engineering by the engineering faculty.

He is currently an Assistant Professor of Electrical and Computer Engineering at The University of Arizona. His research interests include digital communication theory, data compression, and signal processing.

Dr. Marcellin is a member of Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi.

★

Thomas R. Fischer received the Ph.D. degree in electrical and computer engineering from the University of Massachusetts, Amherst, in 1979, and the M.S. degree from the same university in 1978, and the Sc.B. degree magna cum laude from Brown University in 1975.

From June 1975 until August 1976, he was a Staff Engineer at the Charles Stark Draper Laboratory in Cambridge, MA. From September 1979 until December 1988 he was with the Department of Electrical Engineering, Texas A&M University. Since January 1989 he has been a Professor in the Department of Electrical and Computer Engineering at Washington State University. His current research interests include data compression, vector quantization, digital communications, and digital signal processing.