Vector Quantization in Speech Coding

JOHN MAKHOUL, FELLOW, IEEE, SALIM ROUCOS, MEMBER, IEEE, AND HERBERT GISH, MEMBER, IEEE

Invited Paper

***Quantization**, the process of approximating continuous-amplitude signals by digital (discrete-amplitude) signals, is an important aspect of **data compression** or **coding**, the field concerned with the reduction of the number of bits necessary to transmit or store analog data, subject to a distortion or fidelity criterion. The independent quantization of each signal value or parameter is termed **scalar** quantization, while the joint quantization of a block of parameters is termed block or **vector** quantization. This tutorial review presents the basic concepts employed in vector quantization and gives a realistic assessment of its benefits and costs when compared to scalar quantization. Vector quantization is presented as a process of redundancy removal that makes effective use of four interrelated properties of vector parameters: linear dependency (correlation), nonlinear dependency, shape of the probability density function (pdf), and vector dimensionality itself. In contrast, scalar quantization can utilize effectively only linear dependency and pdf shape. The basic concepts are illustrated by means of simple examples and the theoretical limits of vector quantizer performance are reviewed, based on results from rate-distortion theory. Practical issues relating to quantizer design, implementation, and performance in actual applications are explored. While many of the methods presented are quite general and can be used for the coding of arbitrary signals, this paper focuses primarily on the coding of speech signals and parameters.*

I. INTRODUCTION

Current projections for world-wide communications in the 1990s and beyond, point to a proliferation of digital transmission as a dominant means of communication for voice and data. Digital transmission is expected to provide flexibility, reliability, and cost effectiveness, with the added potential for communication privacy and security through encryption. The costs of digital storage and transmission media are generally proportional to the amount of digital data that can be stored or transmitted. While the cost of such media decreases every year, the demand for their use increases at an even higher rate. Therefore, there is a continuing need to minimize the number of bits necessary to transmit signals while maintaining acceptable signal fidelity or quality. The branch of electrical engineering that deals with the latter problem is termed *data compression* or *coding*. When applied to speech, it is known as *speech compression* or *speech coding*.

The conversion of an analog (continuous-time, continuous-amplitude) source into a digital (discrete-time, discrete-amplitude) source, consists of two parts: *sampling* and *quantization*. Sampling converts a continuous-time signal into a discrete-time signal by measuring the signal value at regular intervals of time. Quantization converts a continuous-amplitude signal into one of a set of discrete amplitudes, thus resulting in a discrete-amplitude signal that is different from the continuous-amplitude signal by the quantization error or noise. In this paper, we shall assume that our signals are adequately sampled (see, for example, [107]) so that the only loss in fidelity is attributable to quantization.

When each of a set of parameters (or a sequence of signal values) is quantized separately, the process is known as *scalar quantization*. When the set of parameters is quantized jointly as a single vector, the process is known as *vector quantization* (also known as block quantization or pattern-matching quantization). We shall often abbreviate vector quantization in this paper as VQ.

A. *Purpose and Scope*

The main purpose of this paper is to present the reader with information that can be used in making a realistic assessment of the benefits and costs of vector quantization relative to scalar quantization, especially in speech coding applications. The emphasis is on the exposition of basic principles rather than the elaboration of various techniques and their variations for which references to the literature are provided. Vector quantization is presented as a process of redundancy removal that makes effective use of four interrelated properties of vector parameters: linear dependency (correlation), nonlinear dependency, shape of the probability density function (pdf), and vector dimensionality itself. We shall see that linear dependency and pdf shape can be employed quite effectively with scalar quantization while the other two properties cannot. Nonlinear dependency plays a significant role in the quantization of speech spectral parameters, while dimensionality is important for waveform quantization. Because of the relatively large cost of vector quantization (generally exponential in the number of dimensions and the number of bits per dimension), given today's computation and storage technologies, the major benefits of vector quantization are

Manuscript received January 15, 1985; revised July 2, 1985.

The authors are with BBN Laboratories Inc., Cambridge, MA 02238, USA.

0018-9219/85/1100-1551$01.00 ©1985 IEEE

realized largely at transmission rates of about 1 bit per parameter or less, which is exactly the range where the performance of scalar quantizers degrades sharply. While the concepts presented are quite general, we shall focus in this paper on the low-rate coding of speech (below 8 kbits/s) as an application.

Vector quantization for the purpose of speech coding was used by Dudley [31] in the 1950s and Smith [127] in the 1960s. However, it was not until the introduction of linear predictive coding (LPC) [8], [71], [86], [90] to speech coding that VQ has had significant activity, starting with the work of Kang and Coulter [77], but spurred on mainly by the work of Buzo *et al.* [18], [81]. Until recently, the main purpose for the use of VQ in speech coding has been to reduce the transmission rate of 2400-bit/s vocoders (voice coders) to operate at much lower rates while maintaining acceptable speech intelligibility and quality. Speech coding at very low rates, in the range of 200–800 bits/s, has attracted substantial interest [18], [32], [53], [56], [76], [77], [98], [100], [111], [119], [137], [138] for use in both government and commercial applications. At such low rates, it is important to maximize the cost effectiveness of every bit that is transmitted. Vector quantization has been instrumental in retaining sufficient speech intelligibility to make such systems of actual utility. Today, very-low-rate coding of speech remains one of the major successful applications of VQ. More recently, a mushrooming research activity in the application of VQ to speech waveform coding at somewhat higher data rates has been taking place. While much of the activity has focused on the 8–16-kbit/s range [1], [24], [25], [34], [37], [42], [49], [50], [56], [63], [83], [109], [124], some work has started at data rates below 8 kbits/s [7], [117], which points to exciting possibilities for high-quality speech coding at low rates.

We should point out that, in a sense, VQ has been used regularly and effectively in pattern-recognition type of speech applications, such as in speech and speaker recognition (see, for example, [27], [80], [95], [104], [106], [118], [126]). After all, the VQ problem is part of the general pattern-recognition problem of the classification of data into a discrete number of categories that optimize some fidelity criterion. Indeed, in the design of vector quantizers, one often employs well-known techniques from pattern recognition. However, the basic theory underlying VQ stems from information theory and has wider implications for the transmission of information.

The theoretical foundations of data compression and vector quantization lie in a branch of information theory known as rate-distortion theory, originally set forth by Shannon [122]. Also, most of the theoretical developments since Shannon have taken place as part of the information theory discipline. Of particular relevance is the book by Berger [14] on rate-distortion theory, as well as other information theory texts, such as [46], [92]. Because a full development of vector quantization theory would be highly mathematical, we have chosen in this paper to concentrate on presenting the basic notions and the major results with just enough mathematics that would allow us to be complete without being obscure, we hope.

For further reading, we list a few key references which also contain other references to the literature. The books of collected papers edited by Jayant [73] and Davisson and Gray [26] are devoted to data compression and cover aspects of speech compression. Two relatively recent special issues, the IEEE TRANSACTIONS ON INFORMATION THEORY of March 1982 and the IEEE TRANSACTIONS ON COMMUNICATIONS of April 1982, are devoted to quantization and to speech coding, respectively. The most comprehensive treatment of waveform coding, with applications to speech and video, is the recent book by Jayant and Noll [74]. The review articles by Gold [52], Flanagan *et al.* [39], and Makhoul [87] cover various aspects of speech coding, while the review articles by Gersho and Cuperman [49] and Gray [56] describe some recent work in vector quantization.

B. Paper Outline

In Section II we present the basic VQ problem and several distortion measures that are utilized, along with the basic system design and associated computational and storage costs. The section ends with a VQ model that is introduced, with examples, to aid the reader in visualizing the different processes at work in VQ and in assessing the relative merits of vector and scalar quantization. The remainder of the paper can be viewed as an elaboration of the basic model, supported by theoretical and practical results. Section III contains some of the major theoretical results known about VQ performance from rate-distortion theory. Section IV is devoted to the design of scalar quantizers for vector sources; it includes a comparison between scalar and vector quantization for low-rate speech coding. In Section V we present important practical considerations for vector quantizer design, including methods for reducing computational and storage costs at some loss in performance, and issues of robustness in terms of expected differences between design performance and operational performance. One can take advantage of long-term time-related signal dependencies to reduce the bit rate further without sacrificing signal fidelity; several time-dependent VQ methods are summarized in Section VI. The main paper ends in Section VII with a brief discussion of VQ in speech waveform coding and an outlook to the future.

C. Speech Coding

Before we start the main presentation, we describe the main components of a general speech coding system and two paradigms that we shall use in this paper as a basis for our examples from speech coding.

Fig. 1 shows the basic components of a data compression system appropriate for speech coding. The first component analyzes the discrete-time signal $s(n)$[1] and extracts a vector of unquantized parameters $\boldsymbol{x}(n)$. The set of parameters $\boldsymbol{x}(n)$ is quantized into the vector $\boldsymbol{y}(n)$, which is then encoded into a sequence of bits $\boldsymbol{c}(n)$ and transmitted through the transmission channel or stored in some storage medium. (The quantizer includes any prediction and feedback loops that are an integral part of the quantization process.) In general, the output of the channel $\boldsymbol{c}'(n)$ will be different from $\boldsymbol{c}(n)$ if there are channel errors. At the receiver, the decoder converts the sequence of bits $\boldsymbol{c}'(n)$ into parameter values $\boldsymbol{y}'(n)$, which are then used as input

[1]The dependence of the discrete-time signals on the sampling period will be assumed but not shown explicitly.

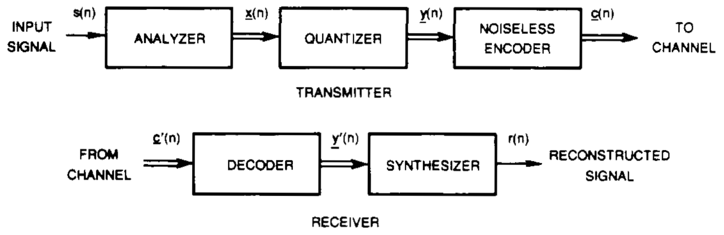

Fig. 1. Basic components of a data compression system for speech coding.

to the synthesizer. The output $r(n)$ is the reconstructed signal which will be an approximation to the input signal $s(n)$.

If there are no channel errors, then $c'(n) = c(n)$ and $y'(n) = y(n)$. The subject of channel errors is important, but will be treated only briefly in this paper. Unless otherwise noted, we shall assume no channel errors.

The nature of the synthesizer in Fig. 1 determines the type of voice coder and dictates the type of analysis to be performed. Fig. 2 shows an example of a synthesis model that is in general use. The model has two major components: an excitation (or source) and a spectral shaping filter. Having chosen a particular synthesis model, any reduction in transmission rate is accomplished by the quantizer and the encoder in Fig. 1. The encoder[2] is assumed to be noiseless, i.e., it does not introduce any additional noise or loss in fidelity. It assigns bits to $y(n)$ in such a way as to minimize the transmission rate, without any loss in fidelity, and may include additional bits to protect the transmitted bit stream against channel errors.

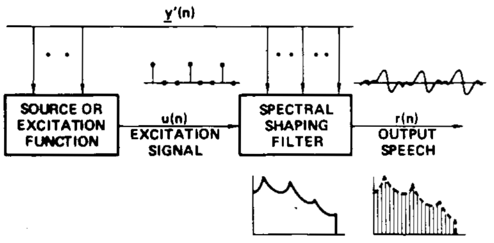

Fig. 2. Major components of the synthesizer in many speech coding systems.

The distortion in the output $r(n)$ relative to the input $s(n)$ may be the result of two processes: modeling and quantization. The modeling effected by the analysis/synthesis system in Fig. 1 may introduce a certain amount of distortion, even in the absence of any quantization (see, for example, the pitch-excited model described below). In this paper, we shall focus only on the distortion caused by the quantization process.

The speech coding task then is to design a system that minimizes the transmission rate while maintaining a certain speech quality, or conversely to maximize speech quality (minimize distortion) for a given transmission rate, subject to certain system cost constraints. For a given choice of analysis/synthesis system, the distortion and transmission rates are determined by the quantizer and the encoder.

[2]The terms "encoder" and "coding" are often used to refer to the whole compression process, as in speech coding. The noiseless encoder in Fig. 1 refers to a very specific part of the compression process. It is hoped that it will be clear from the context which of the two usages is meant.

Speech coding paradigms: We shall employ two basic paradigms as representative of the applications in speech coding, a low-rate coding paradigm and a medium-rate paradigm. The terms low-rate (or narrow-band) and medium-rate (or medium-band) have been used in the literature to denote a wide variety of systems. We shall use the term *medium-rate* for systems operating in the range 8–16 kbits/s, and *low-rate* for systems operating below that range, typically at or below 2400 bits/s. The term *very-low-rate* is often used to denote low-rate systems operating below about 1000 bits/s. Below, we give the basic paradigms that are used in this paper as examples of current systems that operate in the two main ranges.

In our paradigms, the synthesis model shown in Fig. 2 is based on a short-term spectral analysis of speech, where the speech signal is modeled as the output of an all-pole spectral shaping filter

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + \sum_{k=1}^{N} a(k) z^{-k}} \tag{1}$$

which is excited by a source with a flat spectral envelope. This is the well-known linear-predictive coding (LPC) model for speech. The gain G and the predictor coefficients $\{a(k), 1 \leqslant k \leqslant N\}$ are computed on a short-term basis over a *frame* of about 20–30 ms in which the speech signal can be considered to be approximately stationary. The coefficients are obtained as a result of minimizing the energy of the *prediction residual* obtained by filtering the input signal $s(n)$ through the all-zero filter $A(z)$. Ideally, the *excitation* signal $u(n)$ in Fig. 2 should match the residual signal so that the reconstructed signal $r(n)$ will match the input $s(n)$. In many medium-rate systems, the excitation used is exactly a quantized version of the prediction residual. This is true of many predictive waveform coding systems such as adaptive predictive coding (APC) [10], [89] and certain implementations of adaptive transform coding (ATC) [16]. We shall refer to such systems as *residual-excited* systems. (In Section VII, we shall include another filter that models the periodicity in the speech signal.)

At low rates, without using VQ it becomes necessary to have a simple model of the residual to maintain adequate speech intelligibility and quality. The most popular model is the *pitch-excited* model, where the speech in each frame is declared as either *voiced* or *unvoiced*. (Vowels and nasals are examples of voiced sounds, while consonants such as p, t, k, f, s, are unvoiced.) If a voiced determination is made, i.e., the sound is quasi-periodic at that point, the *pitch*, or fundamental frequency is measured and transmitted as well.

At the receiver, voiced sounds are synthesized by exciting the spectral shaping filter by a sequence of pulses separated by the period (Fig. 2 depicts that situation). For unvoiced sounds, a white random noise source is used as excitation. In either case, the gain G of the filter $H(z)$ is set such that the short-term energy in the output is equal to that of the input speech. Note that the pitch-excited model causes a certain amount of modeling distortion in the output, which can be heard even with no quantization of the model parameters.

II. Vector Quantization

This section begins with a formulation of the vector quantization problem, followed by a discussion of the more common distortion measures that are employed. Next we present the basic VQ system design and its associated computational and storage costs. The section ends with the introduction of a VQ model that is aimed at giving the reader a view of the various processes at work in vector quantization.

A. *Problem Formulation*

We assume that $\boldsymbol{x} = [x_1 \ x_2 \ \cdots \ x_N]^T$ is an N-dimensional vector whose components $\{x_k, 1 \leqslant k \leqslant N\}$ are real-valued, continuous-amplitude random variables. (The superscript T denotes transpose.) In vector quantization, the vector $\boldsymbol{x}$ is mapped onto another real-valued, discrete-amplitude, N-dimensional vector $\boldsymbol{y}$. We say that $\boldsymbol{x}$ is quantized as $\boldsymbol{y}$, and $\boldsymbol{y}$ is the quantized value of $\boldsymbol{x}$. We write

$$\boldsymbol{y} = q(\boldsymbol{x})$$

where $q(\cdot)$ is the quantization operator. $\boldsymbol{y}$ is also called the *reconstruction vector* or the *output vector* corresponding to $\boldsymbol{x}$. Typically, $\boldsymbol{y}$ takes on one of a finite set of values $\boldsymbol{Y} = \{\boldsymbol{y}_i, 1 \leqslant i \leqslant L\}$, where $\boldsymbol{y}_i = [y_{i1} \ y_{i2} \ \cdots \ y_{iN}]^T$. The set $\boldsymbol{Y}$ is referred to as the reconstruction *codebook*, or simply the codebook, L is the *size* of the codebook, and $\{\boldsymbol{y}_i\}$ are the set of *code vectors*. The vectors $\boldsymbol{y}_i$ are also known in the pattern-recognition literature as the reference patterns or *templates*. The size L of the codebook is also called the *number of levels*, a term borrowed from scalar quantization terminology. Thus one talks about an L-level codebook or L-level quantizer. To design such a codebook, we partition the N-dimensional space of the random vector $\boldsymbol{x}$ into L regions or *cells* $\{C_i, 1 \leqslant i \leqslant L\}$ and associate with each cell C_i a vector $\boldsymbol{y}_i$. The quantizer then assigns the code vector $\boldsymbol{y}_i$ if $\boldsymbol{x}$ is in C_i

$$q(\boldsymbol{x}) = \boldsymbol{y}_i, \qquad \text{if } \boldsymbol{x} \in C_i. \tag{2}$$

The codebook design process is also known as *training* or *populating the codebook*. A method for designing the codebook will be presented in Section II-C.

Fig. 3 shows an example of a partitioning of two-dimensional space ($N = 2$) for the purpose of vector quantization. The region enclosed by the bold lines is the cell C_i. Any input vector $\boldsymbol{x}$ that lies in the cell C_i is quantized as $\boldsymbol{y}_i$. The positions of the code vectors corresponding to the other cells are shown by dots. The total number of code vectors in the example of Fig. 3 is $L = 18$.

For $N = 1$, vector quantization reduces to scalar quantization. Fig. 4 shows an example of a partitioning of the real line for scalar quantization. The code values (output or reconstruction levels) are shown by dots. Here, also, any input value x that lies in the interval C_i is quantized as y_i. The number of levels in Fig. 4 is $L = 10$. Scalar quantization has the very special property that while cells may have different sizes, they all have the *same shape*, namely, they are all intervals on the real line. In comparison, note how in Fig. 3 the cells in two dimensions actually have different shapes. This freedom of having various cell shapes in multidimensional space gives vector quantization an advantage over scalar quantization, as we shall see below.

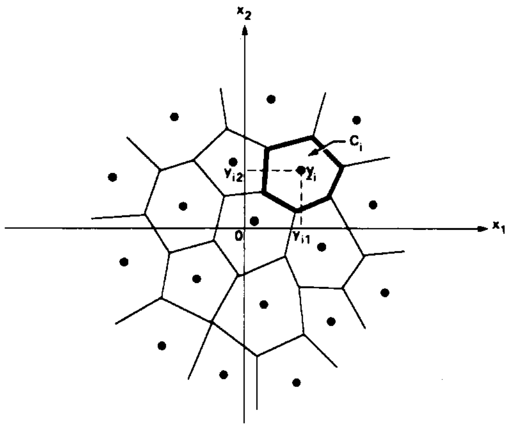

Fig. 3. Partitioning of two-dimensional space ($N = 2$) into $L = 18$ cells. All input vectors in cell C_i will be quantized as the code vector $\boldsymbol{y}_i$. The shapes of the various cells can be very different.

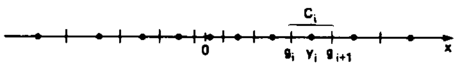

Fig. 4. Partitioning of the real line into $L = 10$ cells or intervals for scalar quantization ($N = 1$).

When $\boldsymbol{x}$ is quantized as $\boldsymbol{y}$, a quantization error results and a *distortion measure* $d(\boldsymbol{x}, \boldsymbol{y})$ can be defined between $\boldsymbol{x}$ and $\boldsymbol{y}$. $d(\boldsymbol{x}, \boldsymbol{y})$ is also known as a dissimilarity measure or distance measure. As the vectors $\boldsymbol{y}(n)$ at different times n are transmitted, one can define an overall average distortion

$$D = \lim_{M \to \infty} \frac{1}{M} \sum_{n=1}^{M} d[\boldsymbol{x}(n), \boldsymbol{y}(n)]. \tag{3}$$

If the vector process $\boldsymbol{x}(n)$ is stationary and ergodic,[3] the sample average in (3) tends in the limit to the expectation

$$\begin{aligned} D &= \mathscr{E}[d(\boldsymbol{x}, \boldsymbol{y})] \\ &= \sum_{i=1}^{L} P(\boldsymbol{x} \in C_i) \mathscr{E}[d(\boldsymbol{x}, \boldsymbol{y}_i) | \boldsymbol{x} \in C_i] \\ &= \sum_{i=1}^{L} P(\boldsymbol{x} \in C_i) \int_{\boldsymbol{x} \in C_i} d(\boldsymbol{x}, \boldsymbol{y}_i) p(\boldsymbol{x}) \, d\boldsymbol{x} \end{aligned} \tag{4}$$

where $P(\boldsymbol{x} \in C_i)$ is the discrete probability that $\boldsymbol{x}$ is in C_i, $p(\boldsymbol{x})$ is the multidimensional probability density function (pdf) of $\boldsymbol{x}$, and the integral is taken over all components of the vector $\boldsymbol{x}$.

For purposes of transmission, each vector $\boldsymbol{y}_i$ is encoded

[3] Ergodicity allows us to substitute sample (or time) averages for ensemble averages [28].

into a codeword of binary digits (bits) $\boldsymbol{c}_i$ of length B_i bits. In general, the different codewords will have different lengths. The transmission rate T is then given by

$$T = BF_c \text{ bits/s} \tag{5}$$

where

$$B = \lim_{M\to\infty} \frac{1}{M} \sum_{n=1}^{M} B(n) \qquad \text{bits/vector} \tag{6}$$

is the average codeword length, $B(n)$ is the number of bits used to code the vector $\boldsymbol{x}(n)$ at time n, and F_c is the number of codewords transmitted per second. It will also be useful to define the average number of bits per parameter or per dimension[4]

$$R = \frac{B}{N} \text{ bits/dimension.} \tag{7}$$

For a codebook of size L, the maximum number of bits needed to code each vector is

$$B_{\max} = \log_2 L. \tag{8}$$

In designing a data compression system, one attempts to design the quantizer such that the distortion in the output is minimized for a given transmission rate. One major decision in designing a quantizer is what distortion measure to use. This is discussed next.

B. Distortion Measures

To be useful, a distortion measure must be tractable, so that one can analyze it and compute it, and be subjectively relevant, so that differences in distortion values can be used as indicating similar differences in speech quality. Most distortion measures in use today are certainly tractable and, to some extent, subjectively relevant. However, many researchers have experienced the frustration which accompanies their discovery that a few decibels of decrease in the distortion is quite perceivable by the ear in one situation but not in another. The careful researcher has learned that, while objective distortion measures are necessary and useful tools in the design of speech coding systems, periodic subjective quality testing is indispensable to making an informed decision on directions for improving system performance. Below we list some of the major distortion measures in use today.

1) Mean-Square Error: By far the most common distortion measure is the mean-square error (mse)

$$d_2(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{N}(\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y}) = \frac{1}{N}\sum_{k=1}^{N}(x_k - y_k)^2 \tag{9}$$

where the distortion is defined per dimension. The popularity of the mse lies mainly in its simplicity and mathematical tractability. A more general distortion measure based on the L_r norm is defined by

$$d_r(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{N}\sum_{k=1}^{N}|x_k - y_k|^r. \tag{10}$$

[4]Note that R is the bit rate per dimension, B is the bit rate per vector, and T is the bit rate per second. We shall often use the generic term *bit rate* to refer to any or all of the three meanings. We trust the intended meaning will be clear from the context.

Note that (10) is equal to (9) for $r = 2$. The two other most popular values of r are $r = 1$ and $r = \infty$. d_1 represents the average absolute error and d_∞ tends towards the maximum error. In fact, one can show that

$$\lim_{r\to\infty} \left[d_r(\boldsymbol{x},\boldsymbol{y})\right]^{1/r} = \max\{|x_k - y_k|, 1 \leqslant k \leqslant N\}. \tag{11}$$

Minimizing D for $r = \infty$ would be equivalent to minimizing the maximum quantization error.

For speech coding, d_2 has been the most popular distortion measure, with d_1 and d_∞ being used occasionally.

2) Weighted Mean-Square Error: In the mse d_2 we assumed that the distortions contributed by quantizing the different parameters $\{x_k\}$ were weighted equally. In general, unequal weights can be introduced to render certain contributions to the distortion more important than others. A general weighted mse is then defined by

$$d_W(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T\boldsymbol{W}(\boldsymbol{x}-\boldsymbol{y}) \tag{12}$$

where $\boldsymbol{W}$ is a positive-definite weighting matrix. $\boldsymbol{W} = N^{-1}\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix, results in $d_W = d_2$.

One choice for $\boldsymbol{W}$ that is popular in many pattern classification applications is $\boldsymbol{W} = \boldsymbol{\Gamma}^{-1}$, where $\boldsymbol{\Gamma}$ is the covariance matrix of the random vector $\boldsymbol{x}$

$$\boldsymbol{\Gamma} = E\left[(\boldsymbol{x}-\bar{\boldsymbol{x}})(\boldsymbol{x}-\bar{\boldsymbol{x}})^T\right], \qquad \bar{\boldsymbol{x}} = \mathscr{E}[\boldsymbol{x}]. \tag{13}$$

In this case, d_W reduces to

$$d_W(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T\boldsymbol{\Gamma}^{-1}(\boldsymbol{x}-\boldsymbol{y}) \tag{14}$$

and is known as the Mahalanobis distance [85].

If, in addition to being positive definite, the weighting matrix $\boldsymbol{W}$ is symmetric (as in the Mahalanobis distance), one can factor $\boldsymbol{W}$ as follows:

$$\boldsymbol{W} = \boldsymbol{P}^T\boldsymbol{P}. \tag{15}$$

The vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ can be transformed into a new set of vectors $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$

$$\tilde{\boldsymbol{x}} = \boldsymbol{Px} \qquad \tilde{\boldsymbol{y}} = \boldsymbol{Py} \tag{16}$$

and

$$\begin{aligned} d_W(\boldsymbol{x},\boldsymbol{y}) &= (\boldsymbol{Px}-\boldsymbol{Py})^T(\boldsymbol{Px}-\boldsymbol{Py}) \\ &= (\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{y}})^T(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{y}}) \\ &= d_2(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}}). \end{aligned} \tag{17}$$

Thus the weighted mse between the original vectors is equal to the mse between the transformed vectors. Therefore, for computational purposes, it may be advantageous to perform the transformation in (16) on all the data before vector quantization is performed.

3) Linear Prediction Distortion Measures: In LPC analysis, the predictor coefficients $\{a(k)\}$ are obtained as a result of minimizing the energy of the prediction residual. One can show [86] that the solution for the optimum $A(z)$ in (1) is unique and is computed from the set of simultaneous linear equations

$$\sum_{k=1}^{N} a(k)\phi(i-k) = -\phi(i), \qquad 1 \leqslant i \leqslant N \tag{18}$$

where $\{\phi(i), 0 \leqslant i \leqslant N\}$ are the short-term autocorrelation coefficients of the speech signal over a single frame. (For a speech signal band-limited to 5 kHz, for example, the number of coefficients N is typically set to 12–14.) The gain G of the filter $H(z)$ is set so that when excited by a unit-variance source the output energy will be equal to $\phi(0)$. That can be accomplished by setting

$$G^2 = \phi(0) + \sum_{k=1}^{N} a(k)\phi(k) \tag{19}$$

which is equal to the minimum residual energy. The parameters of the filter $H(z)$ are computed every frame, quantized, and transmitted.

The gain G is usually quantized on a logarithmic scale and transmitted separately. (Some work has been done in joint quantization of the gain and the LPC parameters [108].) Because the quantization of predictor coefficients can lead to instability of the resulting all-pole filter, they are usually transformed to another set of parameters known as the reflection coefficients $\{K_k, 1 \leq k \leq N\}$ or partial correlation (PARCOR) coefficients [69]. Reflection coefficients result as a byproduct of solving (18) or can be computed recursively from the predictor coefficients (see, for example [86], [90]). For a stable $H(z)$, the reflection coefficients have the property

$$|K_k| < 1, \qquad 1 \leq k \leq N. \tag{20}$$

Because for values of $|K_k|$ approaching 1 the poles approach the unit circle, small changes in K_k can result in large changes in the spectrum. Therefore, for quantization purposes, the reflection coefficients are usually transformed to another set of coefficients that exhibit lower spectral sensitivity as K approaches 1. Two popular transformations are [78]

$$S_k = \frac{2}{\pi}\sin^{-1} K_k, \qquad 1 \leq k \leq N \tag{21}$$

$$G_k = \frac{1}{2}\log\frac{1+K_k}{1-K_k} = \tanh^{-1} K_k, \qquad 1 \leq k \leq N. \tag{22}$$

The parameters G_k are known as log-area-ratios (LARs) from the acoustic tube analogy of the vocal tract [8], [90] and possess the property that small changes in G_k are approximately proportional to corresponding changes in the log spectrum of $H(z)$ [131]. The mse d_2 as well as the minimax error d_∞ have been used to quantize S_k and G_k. The quantization properties of K_k, S_k, and G_k have been studied by several researchers [61], [72], [131],

An alternative distortion measure used in quantizing predictor coefficients was proposed by Itakura and Saito [68], [70]; it derives from maximum-likelihood principles. A modified form of the Itakura–Saito distortion between one vector of predictor coefficients $\boldsymbol{x} = [a(1)\ a(2) \cdots a(N)]^T$ and another vector of predictor coefficients $\boldsymbol{y}$ is given by (see [57] for variations on the Itakura–Saito distortion)

$$d_I(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T \boldsymbol{\Phi}_x (\boldsymbol{x}-\boldsymbol{y}) \tag{23}$$

where

$$\boldsymbol{\Phi}_x = \{\phi(i-k)/\phi(0), 0 \leq i,k \leq N-1\} \tag{24}$$

is the normalized autocorrelation matrix whose coefficients $\phi(i-k)$ were used in computing the vector of predictor coefficients $\boldsymbol{x}$ in (18). Since the autocorrelation coefficients in (24) are normalized by $\phi(0)$, one can show that the matrix $\boldsymbol{\Phi}_x$ and the vector $\boldsymbol{x}$ uniquely determine each other. It is important to note that $\boldsymbol{\Phi}_x$ in (23) is effectively a weighting matrix but, unlike in (12) where $\boldsymbol{W}$ is fixed, here $\boldsymbol{\Phi}_x$ changes value as $\boldsymbol{x}$ changes. Since $\boldsymbol{\Phi}_x \neq \boldsymbol{\Phi}_y$ for $\boldsymbol{x} \neq \boldsymbol{y}$, the Itakura–Saito distortion is not symmetric with respect to its arguments, i.e., $d_I(\boldsymbol{x},\boldsymbol{y}) \neq d_I(\boldsymbol{y},\boldsymbol{x})$. The distortion measure is not a distance or a metric. By contrast, the weighted mse distortion is a symmetric distance and metric.

Even though the computation of d_I in (23) implies a matrix multiply, the computation can be simplified considerably and reduced to a scalar (dot) product [68].

4) Perceptually Motivated Distortion Measures: For very small distortions, and therefore high bit rates, most reasonable distortion measures, including those mentioned above, all exhibit similar behavior, by linearity arguments. Furthermore, they would all be expected to correlate well with subjective judgements of speech quality. However, as bit rate decreases and distortion increases, simple distortion measures have not always correlated well with perceptual judgements. Since VQ is expected to be especially useful at low bit rates, it becomes more important to develop and use distortion measures that are correlated better with human auditory behavior. A number of perceptually based distortion measures, and others that correlate well with subjective judgements, have been used in speech coding (see, for example, [74, Appendix E], [11], [12], [94], [100], [101], [134]). If high speech quality at a given bit rate is the most important consideration in a coder design, then one would do well to consider the use of a distortion measure that correlates well with human perception.

C. Codebook Design

As mentioned above, to design our L-level codebook, we partition N-dimensional space into L cells $\{C_i, 1 \leq i \leq L\}$ and associate with each cell C_i a vector $\boldsymbol{y}_i$. The quantizer then assigns the code vector $\boldsymbol{y}_i$ if $\boldsymbol{x}$ is in C_i. A quantizer is said to be an optimal (minimum-distortion) quantizer if the distortion in (4) is minimized over all L-level quantizers. There are two necessary conditions for optimality. The first condition is that the optimal quantizer is realized by using a minimum-distortion or nearest neighbor selection rule

$$q(\boldsymbol{x}) = \boldsymbol{y}_i, \qquad \text{iff } d(\boldsymbol{x},\boldsymbol{y}_i) \leq d(\boldsymbol{x},\boldsymbol{y}_j), \quad j \neq i, \quad 1 \leq j \leq L. \tag{25}$$

That is, the quantizer chooses the code vector that results in the minimum distortion with respect to $\boldsymbol{x}$. (Ties are decided by some rule.) The second necessary condition for optimality is that each code vector $\boldsymbol{y}_i$ is chosen to minimize the average distortion in cell C_i. That is, $\boldsymbol{y}_i$ is that vector $\boldsymbol{y}$ which minimizes

$$D_i = \mathscr{E}[d(\boldsymbol{x},\boldsymbol{y})\,|\,\boldsymbol{x} \in C_i] = \int_{\boldsymbol{x}\in C_i} d(\boldsymbol{x},\boldsymbol{y})p(\boldsymbol{x})\,d\boldsymbol{x}. \tag{26}$$

We call such a vector the *centroid* of the cell C_i, and we write

$$\boldsymbol{y}_i = \text{cent}(C_i). \tag{27}$$

Computing the centroid for a particular region will depend on the definition of the distortion measure. (The cells thus defined are known as nearest neighbor cells, Voronoi cells, or Dirichlet regions [48].)

In practice, we are given a set of training vectors $\{x(n), 1 \leqslant n \leqslant M\}$. A subset M_i of those vectors will be in cell C_i. The average distortion D_i is then given by

$$D_i = \frac{1}{M_i} \sum_{x \in C_i} d(x, y_i). \tag{28}$$

For either the mse or the weighted mse criterion, one can show that D_i is minimized by

$$y_i = \frac{1}{M_i} \sum_{x \in C_i} x(n) \tag{29}$$

or y_i is simply the sample mean of all the training vectors contained in C_i. For the Itakura–Saito distortion d_I, one can show that y_i is computed by first averaging the normalized autocorrelations corresponding to the sample vectors

$$\phi_{y_i}(k) = \frac{1}{M_i} \sum_{x \in C_i} \phi_x(k), \qquad 0 \leqslant k \leqslant N \tag{30}$$

where $\phi_x(k)$ are normalized such that $\phi_x(0) = 1$. The vector y_i is then obtained as the solution to (18) with $\phi_{y_i}(k)$ as the autocorrelation coefficients.

One method for codebook design is an iterative clustering algorithm known in the pattern-recognition literature as the *K-means algorithm*.[5] In our problem here, $K = L$. The algorithm divides the set of training vectors $\{x(n)\}$ into L clusters[6] C_i in such a way that the two necessary conditions for optimality are satisfied. Below, m is the iteration index and $C_i(m)$ is the ith cluster at iteration m, with $y_i(m)$ its centroid. The algorithm is as follows:

K-Means Algorithm

Step 1: *Initialization:* Set $m = 0$. Choose by an adequate method a set of initial code vectors $y_i(0)$, $1 \leqslant i \leqslant L$. (See Section V-E.)

Step 2: *Classification:* Classify the set of training vectors $\{x(n), 1 \leqslant n \leqslant M\}$ into the clusters C_i by the nearest neighbor rule

$$x \in C_i(m), \qquad \text{iff } d[x, y_i(m)] \leqslant d[x, y_j(m)], \quad \text{all } j \neq i.$$

Step 3: *Code Vector Updating:* $m \leftarrow m + 1$. Update the code vector of every cluster by computing the centroid of the training vectors in each cluster

$$y_i(m) = \text{cent}(C_i(m)), \qquad 1 \leqslant i \leqslant L.$$

Step 4: *Termination Test:* If the decrease in the overall distortion $D(m)$ at iteration m relative to $D(m - 1)$ is below a certain threshold, stop; otherwise go to Step 2.

Any other reasonable termination test may be substituted for Step 4 above.

The above algorithm can be shown to converge to a local optimum (see, for example, [5], [81]). Furthermore, any such solution is, in general, not unique [33], [58]. Global optimality may be achieved approximately by initializing the code vectors to different values and repeating the above algorithm for several sets of initializations and then choosing the codebook that results in the minimum overall distortion. In Section V-E we shall discuss methods for performing initialization.

D. Computational and Storage Costs

Having designed a codebook as described above, one can then use it to quantize each input vector $x(n)$. The quantization is performed as shown in (25) by computing the distortion between $x(n)$ and each of the code vectors, then choosing the code vector with the minimum distortion as the quantized value of $x(n)$. This type of quantization is known as a *full search* since all code vectors are tested for quantizing each input vector. For an L-level quantizer, the number of distortion computations needed to quantize a single input vector is L. While a distortion computation can be arbitrarily complex, we shall assume here that each distortion computation requires a total of N multiply-adds (this is true for the mse and one version of the Itakura–Saito distortion). Therefore, the *computational cost* for quantizing each input vector is

$$\mathscr{C} = NL. \tag{31}$$

If we encode each code vector into $B = RN = \log_2 L$ bits for transmission, then

$$\mathscr{C} = N 2^{RN}. \tag{32}$$

Thus computation cost grows exponentially with the number of dimensions and the number of bits per dimension.

Another important part of the quantization cost is *memory* or *storage cost*, i.e., how much storage is needed to store the code vectors. We shall measure storage assuming one storage location per vector parameter. It is clear that storage cost is then given by

$$\mathscr{M} = NL = N 2^{RN}. \tag{33}$$

Like computational cost, storage cost is exponential in the number of dimensions and the number of bits per dimension.

The costs given above are for the full search algorithm. In Section V we shall present methods that reduce computational costs substantially at the cost of relatively small loss in performance and/or increase in storage.

While we place heavy emphasis on the costs associated with the quantization process itself, it is also important to

[5]The algorithm presented here was described by Forgy in 1965 [41] and is the clustering algorithm described most in the pattern-recognition literature [5], [30], [64], [99], [129]. The name *K-means* comes from MacQueen [84], who actually describes a different algorithm. In an unpublished paper in 1957, Lloyd had independently developed the same algorithm as Forgy's but for the scalar quantization problem and a known distribution (Lloyd's paper has recently been published [82].) The application of this algorithm to a training sequence and the VQ case has been termed in some of the information theory literature as the generalized Lloyd algorithm [56]. Linde, Buzo, and Gray [81] have shown that the algorithm works with a large class of distortion measures, including measures that are not metric, and so the algorithm has also been called the LBG algorithm.

[6]We use the same symbol C_i to represent both the cluster and the cell corresponding to code vector y_i. Cell C_i is that region of N-dimensional space that is closest to y_i based on the nearest neighbor rule, while cluster C_i is that subset of training vectors which are closest to y_i based on the same rule, i.e., cluster C_i is the set of training vectors contained in cell C_i.

keep in mind the costs associated with design of the codebook in the first place. In the K-means algorithm described above, most of the computations result from the classification step; code vector updating presents a negligible amount of computation by comparison. For an L-level quantizer, M training vectors, and I iterations, the computational cost for training is

$$\mathscr{C}_T = NLMI = N2^{NR}MI. \tag{34}$$

The storage cost, including the storage needed to store all the training vectors, is

$$\mathscr{M}_T = N(L + M). \tag{35}$$

For reliable design of the codebook, it has been our experience that one needs at least 10 and preferably about 50 training vectors per code vector, so that M is on the order of $10L$ or more (see Section V-E). So, storage cost for training is largely dominated by the amount of training vectors needed.

E. Vector Quantization Model

Before we delve into more mathematics and examine the detailed workings of vector quantization, we shall present a simple model of vector quantization, which we hope will give the reader a basic understanding of the various processes at work. The concepts presented will be illustrated by simplified examples.

1) Basic Model: Our VQ model identifies four properties of vector components which, when utilized appropriately in codebook design, result in optimal performance. The four properties of vector components are: *linear dependency*, *nonlinear dependency*, *pdf shape*, and *dimensionality*. These four properties are interrelated to a certain extent. For example, even though the multidimensional pdf shape completely specifies any dependencies among vector components, we shall see that pdf shape still plays an important role in determining optimal performance, even when all dependencies among components are removed. For a given codebook size, VQ takes advantage of the four properties above by proper *placement* of the code vectors in N-dimensional space. By code vector placement we mean having the freedom to place code vectors where they are needed most so as to minimize the given distortion measure. For example, one would not place code vectors in regions of zero probability. There are two aspects of code vector placement that are of interest: code vector *spacing* or density and *cell shape*. Code vector spacing refers to how close the code vectors are to each other. In general, one would expect closer spacing (higher density) in high pdf regions (i.e., where $p(\boldsymbol{x})$ is large) and wider spacing of code vectors in regions of low pdf. Once the code vectors are specified, then the cell shapes are automatically determined by the distortion measure. Conversely, once all the cell shapes and positions are specified, then the code vectors are automatically determined as the cell centroids. We shall see below that it will be beneficial to think in terms of cell shapes to gain a better understanding of the workings of VQ, for it is the freedom we have to choose different cell shapes in higher dimensions ($N > 1$) that allows us to exploit dimensionality to minimize distortion in a way that is not possible with scalar quantization. Below, we demonstrate how a vector quantizer chooses its code vector placements and cell shapes, taking advantage of the four vector properties to optimize performance. The examples are designed to expose the processes at work in VQ in a simple way and to show how they may differ from scalar quantization.

2) Dependency: Data compression is largely a process of redundancy removal; it is not necessary to waste bits in transmitting redundant information. Redundancy usually implies some sort of dependency among transmission parameters. We shall classify statistical dependency into two types: *linear dependency* and *nonlinear dependency*. These terms are explained below.

Linear dependency is what we normally think of as *correlation*. Two random variables that are correlated are linearly dependent. If the variables are uncorrelated, they are no longer linearly dependent, but they may still be statistically dependent. The latter "residual" dependency we call nonlinear dependency; it is whatever dependency remains after the linear dependency is removed. Two (zero-mean) variables x_1 and x_2 are uncorrelated if the expected value of their product is zero

$$\mathscr{E}[x_1 x_2] = 0 \quad \text{(uncorrelated)}. \tag{36}$$

But x_1 and x_2 are independent if and only if their joint pdf is equal to the product of the individual (marginal) densities of x_1 and x_2

$$p(x_1, x_2) = p(x_1)p(x_2), \qquad \text{for all } x_1, x_2 \quad \text{(independent)}. \tag{37}$$

If x_1 and x_2 are uncorrelated but dependent (i.e., (36) applies but not (37)), then that dependency we call nonlinear. We shall now demonstrate how one can take advantage of both types of dependency in reducing the necessary bit rate for transmission. In the examples below we shall use the mse as our distortion measure.

EXAMPLE 1

x_1 and x_2 are two random variables whose joint pdf $p(x_1,x_2)$ is shown in Fig. 5. The density is uniform and nonzero inside the rectangle and zero outside the rectan-

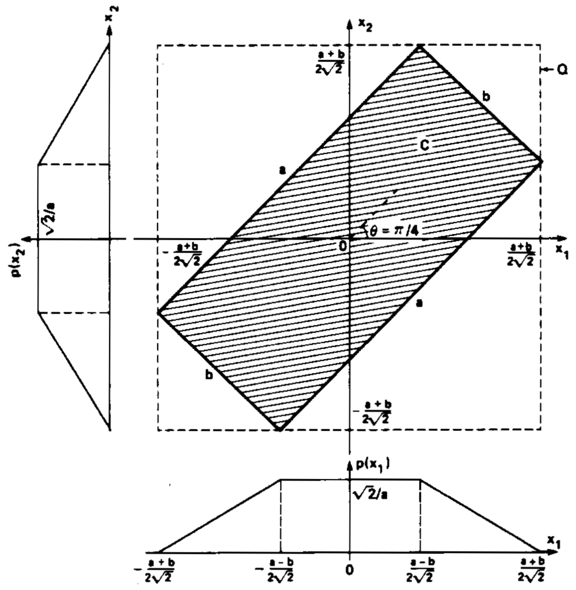

Fig. 5. An example of a uniform pdf $p(x_1, x_2)$ in two dimensions; the density is zero outside the region C. Shown also are the marginal densities $p(x_1)$ and $p(x_2)$. x_1 and x_2 in this example are correlated.

gle. Let the region inside the rectangle be denoted by C, then

$$p(x_1,x_2) = p(\boldsymbol{x}) = \begin{cases} \dfrac{1}{ab}, & \boldsymbol{x} \in C \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

The rectangle has sides a and b, and is oriented at an angle $\theta = \pi/4$. Shown also in Fig. 5 are the marginal densities $p(x_1)$ and $p(x_2)$, which in this example happen to be equal. It is clear from (38) and the marginal densities in Fig. 5 that (37) does not apply, and so x_1 and x_2 are dependent. One can also show that x_1 and x_2 are correlated, so that (36) is not true either.

Now, let us use a scalar *uniform quantizer* to quantize x_1 and x_2 independently. A uniform quantizer is one whose quantizing intervals C_i are of equal length. (Such a quantizer minimizes the mse only for a uniform density and, therefore, would not be optimal for quantizing x_1 and x_2 in this example.) We shall quantize x_1 and x_2 using quantizing intervals equal to Δ. Since x_1 and x_2 have values that range between $-(a+b)/2\sqrt{2}$ and $(a+b)/2\sqrt{2}$, the number of levels needed to quantize each of x_1 and x_2 is equal to

$$L_1 = L_2 = \frac{a+b}{\sqrt{2}\,\Delta}. \tag{39}$$

x_1 and x_2 can be coded using $R_1 = \log_2 L_1$ bits and $R_2 = \log_2 L_2$ bits, respectively. The vector $\boldsymbol{x}$ can be coded, then, using

$$B_x = R_1 + R_2 = \log_2 L_1 L_2 = \log_2 \frac{(a+b)^2}{2\Delta^2} \text{ bits.} \tag{40}$$

The two scalar quantizers correspond to using a vector quantizer with a total number of levels

$$L_x = L_1 L_2 = \frac{(a+b)^2}{2\Delta^2}. \tag{41}$$

Indeed, such a quantizer can be obtained by first demarking the extreme values of x_1 and x_2 by the dashed rectangle (it is a square in this example) enclosing the region Q in Fig. 5, and then drawing a rectangular grid with the separation between grid lines equal to Δ in both directions (see Fig. 8). Such a quantizer would have quantization cells in the form of squares, each of area Δ^2. Clearly, the total number of such squares inside the dotted region Q is obtained by dividing the total area by Δ^2. The result is equal to L_x in (41). Such a vector quantization code is known as a *product code* because it is the Cartesian product of the codes for quantizing x_1 and x_2 separately. The use of a product code for this example is clearly wasteful of bits since regions of zero probability are assigned some of the bits.

We have seen above that, with a product code, the total number of quantization levels is proportional to the area of the whole quantization region (region Q in Fig. 5). Therefore, if the area of the quantization region can somehow be reduced, the number of quantization levels and the corresponding bit rate will also be reduced. For our example, one can perform a coordinate transformation via a rotation that transforms Fig. 5 into Fig. 6. The vector $\boldsymbol{x}$ is transformed into another vector $\boldsymbol{u}$. One can show that the new coordinates u_1 and u_2 are, in fact, uncorrelated. With the proper rotation, any set of random variables can be rendered uncorrelated, as we shall see in Section IV. In the special case of the example in Fig. 6, we see from the marginal densities shown in the figure that

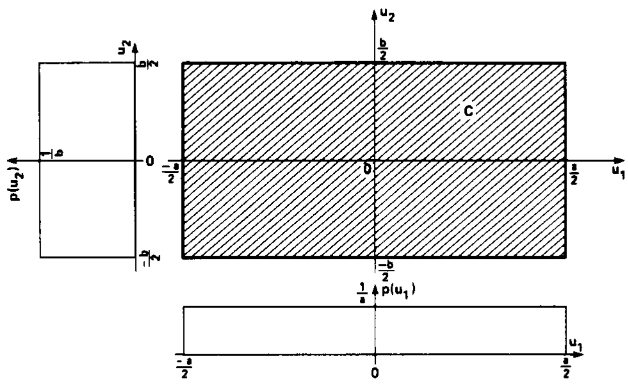

Fig. 6. The pdf in Fig. 5 after a rotation of coordinates. u_1 and u_2 are now uncorrelated and independent.

$$p(u_1,u_2) = p(u_1)p(u_2), \qquad \text{for all } u_1, u_2$$

and, therefore, u_1 and u_2 are also statistically independent. Performing uniform scalar quantization in this case with a quantizing interval of length Δ yields a number of levels equal to

$$L_1 = \frac{a}{\Delta} \qquad L_2 = \frac{b}{\Delta}$$

$$L_u = L_1 L_2 = \frac{ab}{\Delta^2}. \tag{42}$$

The corresponding number of bits is

$$B_u = \log_2 \frac{ab}{\Delta^2}. \tag{43}$$

The difference in the number of bits needed to code $\boldsymbol{x}$ and $\boldsymbol{u}$ can be seen from (40) and (43) to be

$$B_x - B_u = \log_2 \frac{(a+b)^2}{2ab}. \tag{44}$$

For example, for $a = 2b$

$$B_x - B_u = 1.17 \text{ bits} \qquad (a = 2b). \tag{45}$$

Therefore, the rotation saves us in excess of 1 bit per transmitted vector. Such a difference can become significant at low data rates.

In comparing bit rates in (44) we made an implicit assumption that the total quantization distortion is the same for both examples of Figs. 5 and 6. Otherwise, comparing bit rates as such is not meaningful. In fact, for small Δ and, hence, large B_x and large B_u, one can show from (4) that the total distortion in both cases is about the same. Differences in distortion arise as Δ increases and boundary effects near the edges of the rectangle become significant. Under such circumstances one must be careful to equalize the distortions before comparing bit rates. At such low bit rates, (45) would not be expected to hold; it would most likely decrease.

EXAMPLE 2

Example 1 demonstrated how one can take advantage of decorrelation through rotation to reduce the bit rate in scalar quantization of a vector. In this example, we shall demonstrate how, through vector quantization, one can take advantage of nonlinear dependencies to reduce the bit rate.

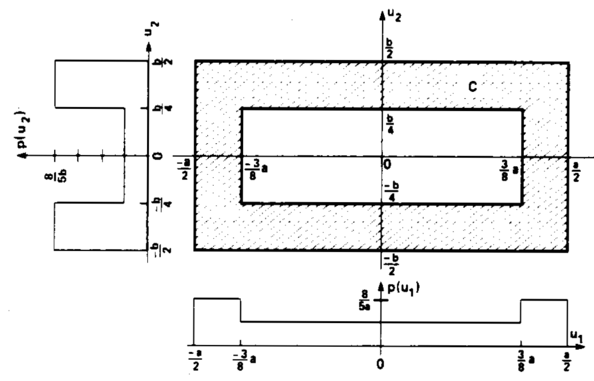

Fig. 7. An example where u_1 and u_2 are uncorrelated but dependent; this type of dependency is termed *nonlinear.*

Fig. 7 shows a different example of a pdf that is nonzero and uniform inside the hatched region C and zero otherwise. Since the area of the hatched region is $5ab/8$, we have

$$p(\mathbf{u}) = \begin{cases} \dfrac{8}{5ab}, & \mathbf{u} \in C \\ 0, & \text{otherwise.} \end{cases} \tag{46}$$

One can show that, like the example in Fig. 6, u_1 and u_2 are uncorrelated but, unlike in Fig. 6, u_1 and u_2 here are not independent, as is clear from (46) and the marginal densities in Fig. 7. Such statistical dependency is termed nonlinear; it cannot be removed by a process of decorrelation. A scalar quantizer designed for u_1 and u_2 in Fig. 7 will yield the same bit rate B_u in (43). To take advantage of the nonlinear dependency we must use a different vector quantizer that partitions only the hatched area in Fig. 7 and does not waste bits inside the small rectangle. One such quantizer would divide only the hatched area into squares of equal area Δ^2. (Such a quantizer would yield the same distortion as the scalar quantizer for small Δ.) The number of levels and bits for this vector quantizer would be

$$L'_u = \frac{5}{8}\frac{ab}{\Delta^2} \qquad B'_u = \log_2 \frac{5ab}{8\Delta^2}. \tag{47}$$

The reduction in bit rate between a scalar quantizer and a vector quantizer in this case is the difference between (43) and (47)

$$B_u - B'_u = \log_2 \frac{8}{5} = 0.68 \text{ bits}. \tag{48}$$

Therefore, taking advantage of nonlinear dependency in this case saved us 0.68 bits/vector.

We have seen how, with proper cell or code vector placement, a vector quantizer was able to take advantage of nonlinear dependencies to reduce the bit rate. It should be clear that any vector rotation will not affect the vector quantizer's ability to locate its cells properly. Therefore, VQ can make effective use of all dependencies (linear or nonlinear) simply by proper code vector placement.

3) Dimensionality:

EXAMPLE 3

The vector quantizer in Example 2 employed the square as the shape of all its cells. The square shape was inspired from the scalar quantizers used earlier. But one property of vector quantizers in higher dimensions is that one has the freedom to choose other cell shapes and not be restricted to the N-dimensional cubes suggested by scalar quantization. Let us examine the effect of using a different shape, such as the *hexagon,* to partition our two-dimensional space. Fig. 8 shows a covering of space by squares and Fig. 9 shows a space covering by regular hexagons. Let each

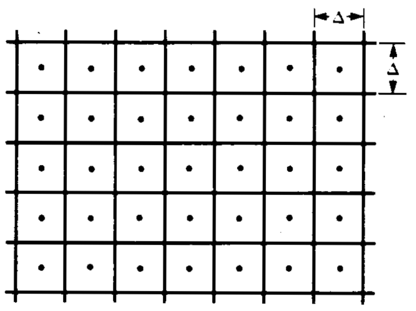

Fig. 8. Packing of two-dimensional space with squares.

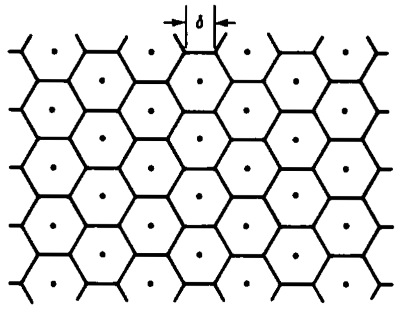

Fig. 9. Packing of two-dimensional space with regu hexagons. For the same number of cells in a given area w a uniform pdf, hexagons have a lower mse than squares.

hexagon have sides of size δ. The area of the hexagon is then

$$A_H = \frac{3\sqrt{3}}{2}\delta^2. \tag{49}$$

To compare the performance of the square quantizer to the hexagon quantizer, we must compare the quantization mse for both quantizers. If we assume that the code vectors are located in the center of the square and the hexagon in each case (as shown by the dots in Figs. 8 and 9), one can show that the mse for each cell is given by

$$E_S = \frac{\Delta^4}{6} \qquad \text{(square)} \tag{50}$$

$$E_H = \frac{5\sqrt{3}}{8}\delta^4 \quad \text{(hexagon)} \tag{51}$$

where the subscripts denote the respective quantizers. The total mse is then obtained by multiplying (50) and (51) by the number of cells (levels). If we make the area of the hexagon equal to the area of the square, i.e., set $A_S = \Delta^2 = A_H$ in (49), then, neglecting edge effects, both quantizers will have the same number of cells covering any given area and, therefore, they will have the same bit rate. However, the ratio of the distortions can be shown to be

$$\frac{E_H}{E_S} = \frac{5\sqrt{3}}{9} = 0.962 \qquad (A_S = A_H) \tag{52}$$

which is equivalent to -0.167 dB. That is, the hexagon quantizer will yield a lower mse than the square quantizer by a fraction of a decibel. Conversely, if we equalize the distortions, i.e., set $E_S = E_H$, then the ratio of the areas will be

$$\frac{A_H}{A_S} = \frac{3}{\sqrt[4]{3}\sqrt{5}} = 1.0194 \qquad (E_S = E_H). \tag{53}$$

This means that, for the same distortion, the hexagon has a slightly larger area than the square and, therefore, will require proportionately fewer cells to cover the same space. Therefore, the bit rate for the hexagon will be lower by an amount equal to $\log_2 A_H/A_S$, which from (53) is equal to 0.028 bits.

The gain of 0.028 bits resulting from the use of a hexagon quantizer can be obtained in the examples of Figs. 6 and 7 in addition to any other gains obtained by taking advantage of linear and nonlinear dependencies. This means that, even in the case when two random variables are independent, as in Fig. 6, vector quantization can squeeze out an additional small fraction of a bit by taking advantage of the higher dimensionality using an appropriate cell shape.

4) PDF Shape: In the examples above, the cells always had not only the same shape throughout the quantization region but also the same size. Effectively, the cell *spacing* was uniform. Uniform cell spacing is certainly a reasonable choice with a uniform pdf. However, for nonuniform pdf shapes, one would expect that some form of nonuniform cell spacing would be needed for optimal performance. We shall see in Section III how scalar and vector quantizers make effective use of pdf shape to reduce the bit rate.

In this section, we presented a model for vector quantization and gave a few examples to illustrate some of the basic principles. In the following sections, we shall review some of the theoretical foundations of vector quantization and give practical examples from speech coding. However, the basic concepts presented in the vector quantization model above and in the simplified examples will still hold when we deal with real-world data.

III. Theoretical Vector Quantizer Performance

In this section we review some of the important *theoretical* tools that are used to help estimate the performance of vector quantizers, along with some of the known results. The presentation is under three headings: rate-distortion theory, scalar quantization, and asymptotic vector quantization. Rate-distortion theory and scalar quantization will give us lower and upper bounds, respectively, on the minimum bit rate achievable by vector quantizers for any given distortion.

This section is theoretical in nature and is rather long. At first reading, we recommend that the reader take only a cursory look at the contents of this section to gain familiarity with the basic definitions and concepts presented, and move quickly to Section IV and the remainder of the paper.

A. Rate-Distortion Theory

Since a major purpose in performing data compression is to minimize the bit rate for a desired level of distortion, it is important in a particular situation to know the theoretical lower bound on the bit rate for any quantizer. By knowing such a bound, one can compare the performance of different quantizers to that bound and decide whether to search for other possibly more complex quantizers that might approach that bound more closely. Rate-distortion theory, a branch of information theory, deals with obtaining such lower bounds without requiring the design of actual quantizers. For a given distortion D, one can compute either $R(D)$, the *rate-distortion function*, defined as the minimum achievable rate (per dimension) for a given distortion D, or its inverse $D(R)$, the *distortion-rate function*, defined as the minimum achievable distortion for a given rate R. The performance limit provided by $D(R)$ or $R(D)$ applies to all methods of source coding, not just vector quantization. It specifically allows for coders that incorporate arbitrarily long delays. In light of the complexity of encoding permitted by this theory, not being able to come very close to the optimal performance when investigating practical vector quantizers may not be indicative of a meaningful disparity. It is when the performance of practical coders is relatively close to the rate-distortion limits that the theory is most useful. Below, we present a summary of the relevant results from rate-distortion theory, preceded by an introduction to the concept of *entropy* from information theory.

1) Coding of Quantizer Output: We mentioned in Section II that one can code the quantizer output vectors $\{y_i, 1 \leq i \leq L\}$ with $\log_2 L$ bits each. If L is a power of 2, the coding is very simple, but if L is not a power of 2, then one can group several vectors together and then perform the coding. For example, if $L = 5$, then to code a single vector would require 3 bits instead of the desired $\log_2 5 = 2.32$ bits, since one cannot transmit a fractional number of bits as such. However, if we group three vectors together, the total number of levels is equal to $L^3 = 125 \cong 2^7$, then each triplet can be coded using 7 bits, for an average of 2.33 bits which is close to the desired average. Henceforth, we shall disregard the problem of fractional bits and simply assume that values can be grouped together, if desired, to achieve the needed rate.

In general, coding L vectors with $\log_2 L$ bits each is actually the maximum rate needed for coding. The minimum average achievable rate to code the vectors $\{y_i\}$ is given by the *entropy* of $\{y_i\}$ [46], defined as

$$H(y) = -\sum_{i=1}^{L} P(y_i) \log_2 P(y_i) \tag{54}$$

where $H(y)$ is the entropy of the discrete-amplitude variable y and $P(y_i)$ is the discrete probability of y_i. $H(y)$ is also called *self-information*, and it is a measure in bits of the information in $\{y_i\}$. Since all discrete probabilities must obey

$$0 \leq P(y_i) \leq 1 \qquad \sum_{i=1}^{L} P(y_i) = 1 \tag{55}$$

one can show [46] that entropy is bounded by

$$0 \leq H(y) \leq \log_2 L. \tag{56}$$

Each vector y_i is coded using

$$B_i = -\log_2 P(y_i) \quad \text{bits} \tag{57}$$

so that vectors with different probabilities will have different wordlengths. The resulting code will be a *variable-length* code, with an average rate equal to the entropy $H(y)$ in (54). This type of coding is known as *entropy coding*. The Huffman code [66] is a well-known, straightforward method for performing entropy coding. With variable-length coding, appropriate buffering schemes would need to be implemented if transmission is over a fixed-rate channel. Any such buffering would, of course, introduce a delay in the system. Also, variable-length coding is particularly sensitive to channel errors. *Permutation codes* [15] offer an alternative for entropy coding using fixed-length codes; however, the codewords can be very long, which also results in delays. In practice, variable-length codes with buffering are used.

One can show that maximum entropy is achieved when all vectors $\mathbf{y}_i$ have equal probability [46], i.e., $P(\mathbf{y}_i) = 1/L$, in which case

$$H_{\max}(\mathbf{y}) = B_{\max} = \log_2 L \quad \text{bits.} \tag{58}$$

For this special case, a fixed-length code is, in fact, optimal.

Entropy coding is one form of *noiseless coding* in that the coding does not introduce any additional noise or distortion beyond that introduced by the quantization process; it merely takes advantage of the probability distribution to minimize the bit rate. (The purpose of the noiseless encoding box in Fig. 1 is to minimize the bit rate without introducing extra distortion.)

One important and useful property of entropy coding is that, even if the number of levels L is infinite, the entropy can still be finite. (Values with very small probability contribute very little to the entropy since the limit as $P \to 0$ of $-P\log_2 P$ is zero.) Therefore, one can partition N-dimensional space into a countably infinite number of finite (nonzero) cells, and the entropy of the resultant codebook will be finite. In practice, L must be finite, but it could be made large without substantially increasing the bit rate, provided entropy coding is used.

As the cells in the quantization process are made smaller so that their size goes to zero, the quantizer discrete outputs $\{\mathbf{y}_i\}$ tend to the original continuous-amplitude source $\mathbf{x}$; the quantization distortion goes to zero; and the entropy becomes infinite. In other words, it takes infinitely many bits to transmit a continuous-amplitude source. For such a source, it will prove useful to define its *differential entropy* [14]

$$h(\mathbf{x}) = -\int_{\mathbf{x}} p(\mathbf{x}) \log_2 p(\mathbf{x})\, d\mathbf{x} \tag{59}$$

where $h(\mathbf{x})$ is the differential entropy of the vector $\mathbf{x}$ and the integral is over all the components of $\mathbf{x}$. Unlike *absolute* entropy, which was defined in (54), differential entropy may be positive or negative. As such, differential entropy values are not meaningful in and of themselves; they are meaningful only relative to other differential entropies. Thus the difference between two differential entropies is a meaningful measure of the difference in information (in bits) between the corresponding sources.

For a random variable x with a given variance σ^2, one can show that $h(x)$ is bounded above by [122]

$$h(x) \leqslant h_G(x) = \tfrac{1}{2}\log_2(2\pi e \sigma^2) \tag{60}$$

where $h_G(x)$ is the differential entropy of a Gaussian pdf with variance σ^2. We shall see that Gaussian sources play a special role in bounding the performance of coding systems.

2) Rate-Distortion Theory Results: Most of the results of rate-distortion theory have been obtained for a scalar source that is defined as a function of a discrete variable (such as sampled time). Let $x(n)$ be a discrete-time, continuous-amplitude random source, i.e., a stochastic sequence. If all samples of $x(n)$ are independent and identically distributed, we say that the source is *memoryless* and is completely specified by a single pdf, $p(x)$. It would appear that in quantizing a memoryless source, vector quantization should not have any advantages over scalar quantization. However, we have already seen in Example 3 that VQ indeed can achieve better performance even in the case of a memoryless source. Therefore, in investigating rate-distortion limits we group samples of the source in blocks or vectors and determine the conditions that achieve those limits.

We shall group N consecutive values of $x(n)$ into a single vector $\mathbf{x}$, so that $\mathbf{x}$ is a random vector. The vector $\mathbf{x}$ is then quantized into a vector $\mathbf{y} = q(\mathbf{x})$, where $\mathbf{y}$ is one of the vectors in the set $\{\mathbf{y}_i, 1 \leqslant i \leqslant L\}$, with L possibly infinite. The average distortion D in representing $\mathbf{x}$ as $\mathbf{y}$ is given by $\mathscr{E}[d(\mathbf{x},\mathbf{y})]$, where $d(\mathbf{x},\mathbf{y})$ is the distortion per dimension (i.e., per sample of $x(n)$). The vectors $\mathbf{y}$ can be transmitted at an average bit rate of $R = H(\mathbf{y})/N$ bits per sample, where $H(\mathbf{y})$ is the entropy of $\mathbf{y}$ as defined in (54). The minimum achievable distortion $D_N(R)$ for a given rate R is given by

$$D_N(R) = \min_{q(\mathbf{x})} \mathscr{E}[d(\mathbf{x},\mathbf{y})], \qquad \text{with } \frac{1}{N}H(\mathbf{y}) \leqslant R \tag{61}$$

where the minimum is taken over all possible mappings $q(\mathbf{x})$. The distortion-rate function $D(R)$ is obtained in the limit as $N \to \infty$

$$D(R) = \lim_{N\to\infty} D_N(R). \tag{62}$$

$D(R)$ is the minimum attainable distortion in coding the source $x(n)$ at a rate R; it is a lower bound on the performance of any quantization scheme. The rate-distortion function $R(D)$ is the inverse of $D(R)$ and is defined similarly.[7] We shall have occasion to use both functions in this paper, although $D(R)$ is used more in practice since we are typically given the rate R and we design our quantizer to minimize the distortion.

The main result of rate-distortion theory that relates to VQ is that, *by using a vector quantizer, one can in principle approach the distortion-rate function arbitrarily closely by increasing the vector size N.* This is essentially implied in the definition of $D(R)$ in (62). Therefore, $D(R)$ is not merely a lower bound for any quantizer, it is actually achievable, in theory, by a vector quantizer of high dimension.

The distortion-rate function $D(R)$ has two important properties: it is *monotone decreasing* with R and it is *convex*. Furthermore, for the mse distortion, $D(R)$ decreases at the rate of about 6 dB/bit for large R. We shall exhibit these properties in specific examples below.

While defining $R(D)$ or $D(R)$ is straightforward, neither is simple to evaluate analytically, except for a few special cases. For a memoryless (zero-mean) Gaussian source with variance σ^2 and a mse distortion, $R(D)$ and $D(R)$ are

[7]The general definition of $R(D)$ in rate-distortion theory is given in terms of the concept of *mutual information* [14]. We have chosen here a somewhat narrower definition that is sufficient for our purposes.

given by [14]

$$R_G(D) = \max\left\{0, \tfrac{1}{2}\log_2 \frac{\sigma^2}{D}\right\}$$
$$= \begin{cases} \tfrac{1}{2}\log_2 \sigma^2/D, & 0 \leqslant D \leqslant \sigma^2 \\ 0, & D > \sigma^2 \end{cases} \tag{63}$$
$$D_G(R) = 2^{-2R}\sigma^2. \tag{64}$$

If $D > \sigma^2$ is given, then we need not transmit any information ($R = 0$) because we can always obtain $D = \sigma^2$ by using zeros for the quantized values of $x(n)$, since the quantization error is then equal to $x(n)$. By dividing (64) by σ^2 we obtain the *normalized distortion* which can be measured in decibels

$$\mathcal{D}_G(R) = 10\log_{10}\frac{D_G(R)}{\sigma^2} = -6.02R \quad \text{dB} \tag{65}$$

where *script* $\mathcal{D}$ is the normalized distortion in decibels. The negative of $\mathcal{D}$ is just the signal-to-noise ratio (SNR) in decibels of the quantizer

$$\text{SNR} = -\mathcal{D} = 10\log_{10}\frac{\sigma^2}{D} \quad \text{dB}. \tag{66}$$

For much of this paper we shall use $\mathcal{D}$ instead of SNR to measure quantizer mse distortion.

There are scant explicit $D(R)$ results for memoryless non-Gaussian sources (see the result for Laplacian densities by Berger [14]). However, lower and upper bounds exist

$$D^*(R) \leqslant D(R) \leqslant D_G(R). \tag{67}$$

Thus $D(R)$ is upper bounded by the distortion-rate function for a Gaussian source. The lower bound $D^*(R)$, known as the *Shannon lower bound*, for a mse distortion is given by

$$D^*(R) = \frac{1}{2\pi e}2^{2h(x)}2^{-2R} \tag{68}$$
$$R^*(D) = h(x) - \tfrac{1}{2}\log_2 2\pi e D \tag{69}$$

where $h(x)$ is the differential entropy of the memoryless source. The Shannon lower bound is achievable for many sources only as $R \to \infty$. From (60) and (68), we can write $\mathcal{D}^*(R)$ as a normalized distortion in decibels

$$\mathcal{D}^*(R) = -6.02R - 6.02[h_G(x) - h(x)] \quad \text{dB} \tag{70}$$

or

$$\mathcal{D}_G(R) - \mathcal{D}^*(R) = 6.02[h_G(x) - h(x)] \quad \text{dB}$$
$$= 6.02[R_G(D) - R^*(D)] \quad \text{dB}. \tag{71}$$

Since $h_G(x) > h(x), \mathcal{D}^*(R)$ is less than the Gaussian distortion-rate function by an amount equal to the difference between the source and Gaussian differential entropies (in bits) multiplied by 6.02 dB/bit. Equation (70) makes it very clear that the asymptotic behavior of the distortion-rate function for many sources is expected to decrease at a rate of -6.02 dB/bit as $R \to \infty$.

Table 1 shows four pdfs that are common models used for certain signal distributions. Shown are the differential entropies, the difference $R_G(D) - R^*(D)$, and the corresponding difference in distortion. The Gamma pdf shows the greatest deviation from the Gaussian; it is by far the sharpest or most peaked of the four pdfs, and it becomes unbounded at $x = 0$. The Gamma density is often mentioned as a good model for the first-order pdf of speech [74, p. 32]. However, this model is only good for long-term statistics. Medium-term statistics (on the order of 100 ms), where the speech amplitudes are normalized with respect to the medium-term energy, show that the Laplacian pdf becomes a better model of speech [96]. Short-term statistics (on the order of 20 ms) show the Gaussian pdf to be a good first-order model [96]. The Gaussian pdf also appears to be a good short-term first-order model for the prediction residual in adaptive predictive coding [7], [74]. Since any speech coding system operating at 2 bits/sample or less would need to be normalized with respect to short-term energy to minimize distortion, we shall assume in this paper that the first-order pdf for speech and the prediction residual is essentially Gaussian.

Table 1 Four Common PDFs and their Differential Entropies $h(x)$. Column 4 gives the difference in bits between the Gaussian rate-distortion function $R_G(D)$ and the Shannon lower bound $R^*(D)$ for each pdf. Column 5 shows the corresponding difference in distortion in decibels; it is obtained by multiplying column 4 by 6.02. Column 6 is the asymptotic (high bit rate) difference in bits between the rate of a Lloyd–Max quantizer R_{LM} and the Shannon lower bound, and column 7 is the corresponding difference for the distortion. (From Jayant and Noll [74].)

pdf	$p(x)$	$h(x)$	$R_G(D) - R^*(D) = h_G(x) - h(x)$ (bits)	$\mathcal{D}_G(R) - \mathcal{D}^*(R)$ (dB)	$R_{LM} - R^*$ (bits)	$\mathcal{D}_{LM} - \mathcal{D}^*$ (dB)
Gaussian	$\frac{1}{\sqrt{2\pi}\,\sigma}\exp[-x^2/2\sigma^2]$	$\frac{1}{2}\log_2(2\pi e\sigma^2)$	0	0	0.722	4.35
Uniform	$\frac{1}{2\sqrt{3}\,\sigma}$, $\lvert x\rvert \leqslant \sqrt{3}\,\sigma$; 0, otherwise	$\frac{1}{2}\log_2(12\sigma^2)$	0.255	1.53	0.255	1.53
Laplacian	$\frac{1}{\sqrt{2}\,\sigma}\exp[-\sqrt{2}\lvert x\rvert/\sigma]$	$\frac{1}{2}\log_2(2e^2\sigma^2)$	0.104	0.63	1.190	7.17
Gamma	$\frac{\sqrt[4]{3}}{\sqrt{8\pi\sigma\lvert x\rvert}}\exp[-\sqrt{3}\lvert x\rvert/2\sigma]$	$\frac{1}{2}\log_2(4\pi e^{1-C}\sigma^2/3)$ C = Euler's constant = 0.5772	0.709	4.27	1.963	11.82

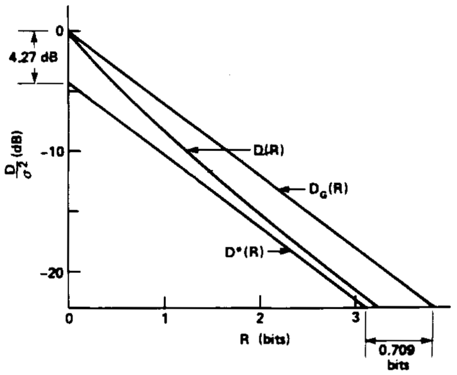

Fig. 10. Distortion-rate function $D(R)$ for a memoryless Gamma source and a mse distortion measure, bounded above by the Gaussian distortion-rate function $D_G(R)$ and below by the Shannon lower bound $D^*(R)$, from Noll and Zelinski [97]. $D_G(R)$ and $D^*(R)$ are always parallel straight lines in the decibel scale with a slope of -6.02 dB/bit. $D(R)$ tends toward a slope of -6.02 dB/bit at high bit rates (usually $R > 3$ bits).

Fig. 10 shows a plot of $D(R)$ for the Gamma pdf along with the Gaussian upper bound $D_G(R)$ and the Shannon lower bound $D^*(R)$. (The reason for choosing the Gamma pdf is because it shows very clearly the departure from the Gaussian.) The $D(R)$ plot was obtained using Blahut's algorithm [17]. Note that the $D(R)$ curve is monotone decreasing and convex. Also, as R increases beyond a few bits, $D(R)$ decreases at the rate of about 6.02 dB/bit which is the slope of the upper and lower bounds. The slope of -6.02 dB/bit will recur over and over again for many types of quantizers as R increases.

Thus far we have considered only memoryless sources. For sources with memory, where the samples are dependent, explicit $D(R)$ results are even more meager. What little is known is for the Gaussian case where the dependence is linear and can be specified completely by the spectral density $\Phi(\omega)$ of the stochastic sequence $x(n)$. For this special case, $D(R)$ is given parametrically by [14]

$$D_G(\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \min\{\theta, \Phi(\omega)\}\, d\omega \tag{72}$$

$$R_G(\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \max\left\{0, \frac{1}{2}\log_2 \frac{\Phi(\omega)}{\theta}\right\} d\omega. \tag{73}$$

For the case of small distortions defined by

$$\theta \leqslant \min_{\omega}\{\Phi(\omega)\} \tag{74}$$

$D_G(R)$ is given by

$$D_G(R) = \gamma 2^{-2R}\sigma^2 \tag{75}$$

where

$$\gamma = \frac{1}{\sigma^2}\exp\left[\frac{1}{2\pi}\int_{-\pi}^{\pi}\log\Phi(\omega)\, d\omega\right] = \frac{\mathrm{GM}\{\Phi(\omega)\}}{\mathrm{AM}\{\Phi(\omega)\}} \leqslant 1. \tag{76}$$

GM and AM are the geometric mean and arithmetic mean, respectively. (σ^2 = arithmetic mean of the spectrum.) γ is a *spectral flatness measure* which is a nonnegative quantity that is equal to one if and only if the spectrum is flat [86], [90]. For a flat spectrum, i.e., a white source, the Gaussian signal values are independent and $D_G(R)$ in (75) becomes equal to (64). Therefore,

$$D_G(R)|_{\text{correlated source}} = \gamma D_G(R)|_{\text{white source}} \tag{77}$$

for distortions obeying (74). Note that for this case of small distortions, the distortion in (72) is equal to θ for all frequencies, which means that the reconstruction error will have a flat spectral density: a result that we shall meet again below.

Equation (77) states that $D(R)$ for a correlated Gaussian source is less than that for the corresponding white (memoryless) source. The difference for small distortions (or high rates) is equal to $-10\log_{10}\gamma$ decibels. Equation (77) quantifies the intuitive notion that one can transmit dependent sources at a lower distortion than independent sources. This general notion applies to non-Gaussian sources as well. The $D(R)$ function for such sources is still bounded above and below as in (67), where the Shannon lower bound is still defined by (68). However, $h(x)$ is now the differential entropy for the dependent source, defined as

$$h(x) = \lim_{N\to\infty}\frac{1}{N}h(\boldsymbol{x})$$

which will be smaller than the differential entropy for the corresponding memoryless source.

B. *Scalar Quantization*

We showed above how the distortion-rate function $D(R)$ provides a *lower bound* on the minimum distortion achievable by a vector quantizer at a given bit rate. Scalar quantizers provide us with an *upper bound* on the minimum achievable distortion. The difference between the two bounds gives the reduction in distortion that is potentially attainable by vector quantization. In this section we shall consider the scalar quantization of memoryless sources. The scalar quantization of correlated sources is discussed in Section IV.

1) Lloyd–Max Quantization: A scalar quantizer may be designed using the K-means algorithm described in Section II-C. However, because in one dimension the cells are restricted to be adjacent line segments, as shown in Fig. 4, one can instead use the well-known *Lloyd–Max quantizer*[8] [82], [91]. Given a pdf $p(x)$ and a number of levels L, this quantizer determines the intervals C_i and the reconstruction values y_i that minimize the average mse. The necessary conditions for the minimum are obtained by straightforward differentiation with respect to y_i and the interval boundary values g_i. The necessary conditions for optimality can be shown to be

$$g_i = \tfrac{1}{2}(y_i + y_{i-1}), \qquad 2 \leqslant i \leqslant L \tag{78}$$
$$g_1 = -\infty, \qquad g_{L+1} = \infty$$
$$y_i = \mathrm{cent}(C_i), \qquad 1 \leqslant i \leqslant L \tag{79}$$

where the centroid of C_i is simply the mean value of x in that interval. Equation (78) states that the interval boundaries must lie halfway between the reconstruction values. Note that (78) corresponds to (25) in the general case and (79) is the same as (27). For $L > 3$, (78) and (79) are solved itera-

[8]Also known as the Lloyd quantizer or the Max quantizer.

tively to obtain a set of optimal values. Those values may indicate only a local optimum. For the case when $p(x)$ is log-concave, the above necessary conditions are also sufficient for optimality [40] and the optimum will be global. In general, the optimum quantizer will not be uniform, i.e., the interval lengths will not be the same.[9] The interval spacing tends to be smaller where the pdf is larger. Performance, typically, must be evaluated by numerical methods, except when the number of quantization levels is large there is the asymptotic formula given by Algazi [4] for the rth-power distortion

$$D_r = \frac{2^{-r}}{r+1} L^{-r} \left[\int_{-\infty}^{\infty} [p(x)]^{1/(r+1)}\, dx \right]^{r+1} \quad \text{(Lloyd–Max)}. \tag{80}$$

For $R = \log_2 L$ and $r = 2$, (80) can be written in decibels as

$$\mathscr{D}_2 = 10 \log_{10} D_2 = -6.02R + F_{LM}(p) \quad \text{dB} \tag{81}$$

where script $\mathscr{D}_2$ is the distortion in decibels and $F_{LM}(p)$ is a constant that depends on the pdf shape. Note again the -6.02-dB/bit behavior for large R.

Fig. 11 shows the normalized mse obtained by the Lloyd–Max quantizer as a function of the rate $R = \log_2 L$ for four pdfs. While the Gamma pdf has the lowest $D(R)$ of

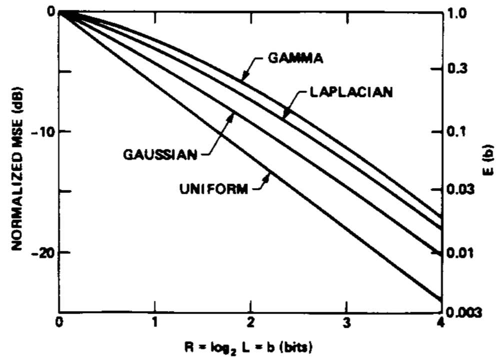

Fig. 11. Normalized mse for four memoryless sources using Lloyd–Max quantization ($R = \log_2 L$), plotted from Jayant and Noll [74, Table 4.4].

the four pdfs, it results in the highest distortion when using the Lloyd–Max quantizer. Columns 6 and 7 of Table 1 show the asymptotic difference between the Lloyd–Max quantizer and the distortion-rate function. These differences are the maximum that can be gained potentially from vector quantization.

2) Constrained-Entropy Quantization: In Lloyd–Max quantization, the number of levels L is fixed and the bit rate is defined simply as $\log_2 L$. However, if the reconstruction values y_i are not equally probable, we can use entropy coding to reduce the rate below $\log_2 L$ to $H(y)$. One can go even further and restate the optimization problem to minimize the distortion *subject to a given entropy* $H(y) = R$. We shall call the resulting quantizer a *constrained-entropy quantizer*. The number of levels L can now be any desired value including infinity. Gish and Pierce [51] have shown that, *at high bit rates, the uniform quantizer is the optimum constrained-entropy quantizer.* With very mild restrictions, this result is true for all pdfs and distortion measures. More recently, Farvardin and Modestino [33] showed experimentally that, with mse distortion, the uniform quantizer is very close to optimal even at very low rates. The obvious conclusion is that, if one is willing to perform variable-rate entropy coding, then the uniform quantizer is always the scalar quantizer of choice.

[9]Note that a uniform quantizer is one whose interval lengths are equal. However, depending on the pdf shape and the distortion measure, the output levels of a minimum-distortion uniform quantizer will not be equally spaced, in general.

The asymptotic (high bit rate) performance for the constrained-entropy quantizer is given in [51], and can be written for an rth-power distortion measure as

$$D_r = \frac{2^{-r}}{r+1} 2^{rh(x)} 2^{-rH_{\min}} \quad \text{(constrained entropy)} \tag{82}$$

where $H_{\min}$ is the entropy of the constrained-entropy quantizer outputs and $h(x)$ is the differential entropy of the source. Equation (82) can be written in decibels for $r = 2$

$$\mathscr{D}_2 = -6.02 H_{\min} + F_{CE}(p) \quad \text{dB} \tag{83}$$

where F_{CE} is another constant that depends on the pdf shape.

It is interesting that, for the mse distortion, there is a simple relationship between $\log_2 L$ of the Lloyd–Max quantizer and $H_{\min}$. In particular, we have from [51] that, for large L

$$H_{LM} - H_{\min} = \tfrac{1}{3}(\log_2 L - H_{\min}) \tag{84}$$

where H_{LM} is the entropy of the Lloyd–Max quantizer outputs. This relation shows the additional benefit of constrained-entropy quantization over entropy coding the outputs of the Lloyd–Max quantizer.

Another significant result obtained by Gish and Pierce [51] is that the constrained-entropy quantizer approaches the rate-distortion bound within a fixed constant that is dependent only on the distortion measure and not on the pdf. This constant, derived for an rth-power distortion measure, is given by

$$H_{\min} - R(D_r) = \frac{1}{r} \log_2 \frac{re}{1+r} + \log_2 \Gamma\left(1 + \frac{1}{r}\right) \tag{85}$$

where Γ is the Gamma (factorial) function. For the mse distortion, (85) reduces to

$$H_{\min} - R(D_2) = \frac{1}{2} \log_2 \frac{\pi e}{6} = 0.255 \text{ bits}. \tag{86}$$

Fig. 12 shows a plot of (85) as a function of r. The horizontal scale has been chosen to illustrate the important result that for $r = \infty$, which corresponds to the minimax criterion, the curve goes to zero, and $H_{\min} = R(D_\infty)$. Therefore, *a uniform scalar quantizer with entropy coding can achieve the rate-distortion function for the minimax criterion.* For $r = 1$, the constrained-entropy quantizer can approach $R(D)$ within 0.443 bits, and within 0.255 bits for $r = 2$.

The low-rate region ($R < 3$ bits) has been explored by a number of researchers (see [74]), with the results of Farvardin and Modestino [33] being the most useful for our purposes. For the mse distortion, the constrained-entropy quantizer at low rates is even closer to $D(R)$ than 0.255 bits, especially for the more peaked pdfs such as the Laplacian and the Gamma densities. In fact, of all the nonuniform pdfs tested, the Gaussian pdf registers the least gain and

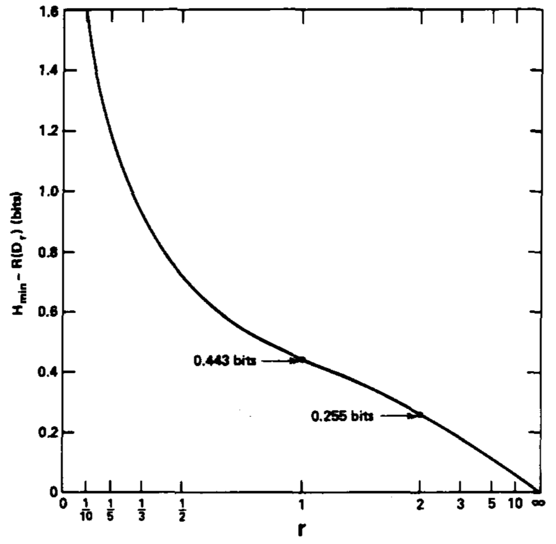

Fig. 12. The difference between the entropy of a constrained-entropy (uniform) quantizer and the rate-distortion function at high bit rates for an rth-power distortion function. This difference is independent of the pdf of the memoryless source; it depends only on r. For $r = \infty$, the minimax distortion, that difference is zero. The data for the plot were taken from Gish and Pierce [51].

lies the furthest away from its $D(R)$ function at low rates. Figs. 13 and 14 show plots of the mse distortion with the constrained-entropy quantizer (solid curves) compared to Lloyd–Max quantization and the distortion-rate functions for the Gaussian (Fig. 13) and the Laplacian (Fig. 14) memoryless sources. The constrained-entropy curve for the

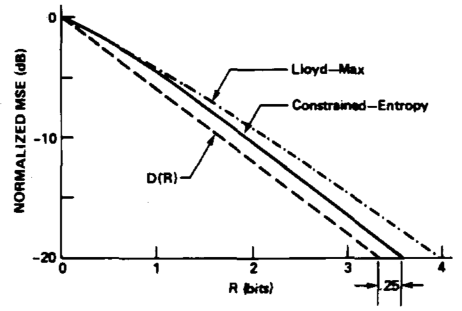

Fig. 13. Constrained-entropy quantization for a memoryless Gaussian source, compared to Lloyd–Max quantization and the distortion-rate function. (From Farvardin and Modestino [33].)

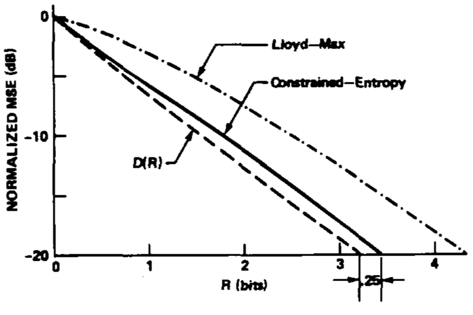

Fig. 14. Same as Fig. 13 for a memoryless Laplacian source. (From Farvardin and Modestino [33].)

Laplacian pdf is much closer to its distortion-rate function than the Gaussian curve is to its $D(R)$, while the opposite is true for the Lloyd–Max curves.

We saw above how constrained-entropy quantization (with entropy coding) takes advantage of the pdf shape to minimize the bit rate at a given distortion. The difference between the solid curves and the $D(R)$ curves in Figs. 13 and 14 is what is not achievable by scalar quantization. Vector quantization with very large N is capable of closing that gap without the need for entropy coding. At the low data rate of $R = 1$ bit in Fig. 13, being able to reach the distortion-rate bound for a Gaussian source would mean a reduction of about 25 percent in bit rate over the scalar quantizers. This result will be important for the coding of the speech prediction residual since it is modeled well as a memoryless Gaussian source.

C. Asymptotic Vector Quantization

We now present some of the known theoretical results for vector quantization. Most of the results below were derived under asymptotic conditions: either L is large (i.e., small distortion) and the vector size N is arbitrary, or N is large and L is arbitrary. We begin by discussing the optimum L-level quantizer in N dimensions, which is analogous to Lloyd–Max quantization in the scalar case, followed by constrained-entropy quantization in the vector case. It is important to notice that, unlike the results of the previous section which applied only to memoryless sources, *the VQ results below apply to sources with arbitrary pdfs, including linear and nonlinear dependencies*, unless noted otherwise.

The asymptotic formulas for scalar quantizer performance mentioned in Section III-B have analogues for vector quantizers. Zador [141] showed that, for large L, the optimum (minimum rth-power distortion) L-level quantizer in N dimensions has a distortion

$$D_r = A(r, N) L^{-r/N} \left[\int_{-\infty}^{\infty} [p(\mathbf{x})]^{N/(N+r)} \, d\mathbf{x} \right]^{(N+r)/N} \tag{87}$$

where $p(\mathbf{x})$ is the N-dimensional pdf of the vector process $\mathbf{x}$, and $A(r, N)$ is a term that is independent of the pdf; it represents how well cells can be packed in N-dimensional space for the rth-power distortion measure. Equation (87) is analogous to (80) in the scalar case, but it is important to note that (80) was derived for a memoryless source while (87) is true for an arbitrary source. For the mse distortion, (87) may be written in decibels as

$$\begin{aligned} \mathcal{D}_2 &= -\frac{6.02}{N} B + F_{VQ}(p, N) \quad \text{dB} \\ &= -6.02R + F_{VQ}(p, N) \quad \text{dB} \end{aligned} \tag{88}$$

where $B = \log_2 L$ is the number of bits per vector, R is the number of bits per dimension, and $F_{VQ}(p, N)$ is a term that depends on the pdf $p(\mathbf{x})$ and the number of dimensions.

The difficulty in utilizing the above expressions is that $A(r, N)$ is known exactly for very few cases. For other than $N = 1$, only $A(2, 2)$ is known exactly. For two dimensions and the mse criterion [35], [47], optimal vector quantizer performance (without entropy coding) is obtained with hexagonal regions, for large L. Note that, except for a uniform pdf, these hexagons will not be regular and will not have the same size in general. For higher dimensions, there are a number of upper and lower bounds on $A(r, N)$, but such bounds, though theoretically interesting, appear to be of little practical value. Recently, Conway and Sloane [23] conjectured a lower bound on $A(2, N)$ that seems interesting and useful (see Fig. 15). Other useful information on performance bounds is contained in [47], [139].

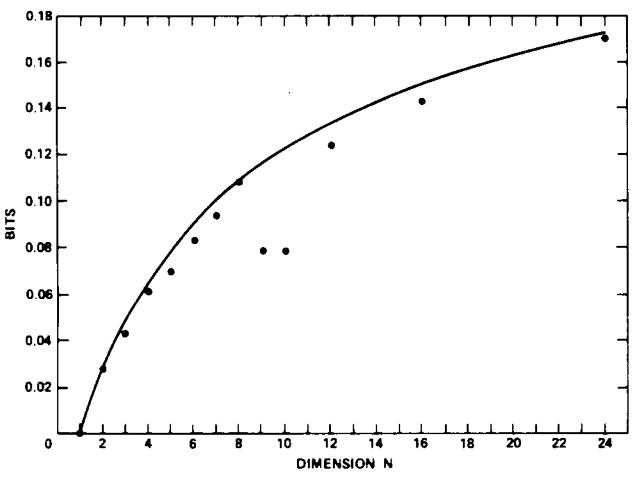

Fig. 15. The savings in bits when using certain N-dimensional lattice quantizers compared to quantization with N-dimensional cubes for a uniform pdf and high bit rate. The data were computed from Conway and Sloane [23, Table I] as follows:

$$\text{BITS} = [10 \log_{10} 0.08333/G_n]/6.02$$

where G_n is as defined in [23]. The solid curve is a conjectured bound given by Conway and Sloane [23] and the dots are the savings for the best known lattice quantizer for each N.

Although the optimal N-dimensional quantizer and its performance are not known in most cases, some understanding has been obtained about its structure. Zador [141] has shown that the optimum density of output values to use for random quantizer selection is proportional to $p^{N/(N+r)}$, where p is the pdf of the vector $\boldsymbol{x}$. (A related result is also obtained by Gersho [47].) For very large N ($N \gg r$), therefore, the output values should have a density similar to $p(\boldsymbol{x})$ for optimum performance.

In a manner similar to scalar quantizer design, one can constrain the entropy rather than the number of levels for the vector quantizer; we call the resulting quantizer the *constrained-entropy vector quantizer.* For such a quantizer, Zador [141] gives the asymptotic rth-power distortion as

$$D_r = B(r, N) 2^{rh(\boldsymbol{x})/N} 2^{-rH_{\min}/N} \tag{89}$$

where $B(r, N)$ is a term similar to $A(r, N)$ in (87). Equation (89) is analogous to (82) in the scalar case. One could write (89) for $r = 2$ to show the -6.02-dB/bit behavior.

Again, values of $B(r, N)$ are known for only a few cases. However, if entropy coding is to be used, there are suboptimal solutions for which the performance is known for large L. One solution, considered by Gish and Pierce [51], is analogous to the solution in the scalar case; namely, apply a uniform scalar quantizer to each of the vector dimensions and perform *joint entropy coding* on the vector outputs. The quantization interval should be the same in all dimensions, which is equivalent to specifying an N-dimensional cube for the cells of the vector quantizer. The result for this coding procedure is that the bit rate will differ from the rate-distortion function by an amount equal to that shown in Fig. 12 for an rth-power distortion measure, independent of the shape of the multidimensional pdf. For the mse, this difference is again 0.255 bits per dimension. This result is true at high bit rates; experiments similar to those performed by Farvardin and Modestino [33] for the scalar case will need to be done for the vector case to test the performance at low bit rates. The 0.255-bit differential that exists for the mse criterion when quantizing with cubes can be reduced by using different cell shapes for different dimensions, as we shall see below in the discussion under space packing.

While the VQ method outlined above is very simple indeed and requires a minimum amount of design and computations, it requires a very large amount of storage. Note that the entropy coding needs to be performed in N-dimensional space. Now, when we use entropy coding we typically use a large number of levels per dimension, therefore, the total number of levels in N-dimensional space will be very large, and the entropy coding will require a storage location for each of the codewords. In essence, we are substituting storage for computation and the amount of storage needed may be excessive for many applications.

Another important result from rate-distortion theory is that, for fixed L and for very large N, the entropy of the vector quantizer approaches $\log_2 L$ [123]. This implies that the output values $\boldsymbol{y}_i$ must have equal probability. (Note that the output values may have a *density* proportional to $p(\boldsymbol{x})$ but still have *discrete* probabilities that are equal.) This result is satisfying in that it confirms the fact that a vector quantizer can achieve its optimal performance for high dimensions without the need for entropy coding. In other words, the fixed-L and constrained-entropy quantizers achieve the same performance for high dimensions. This statement is in sharp contrast to the scalar case ($N = 1$) where the two types of quantizers give different performance.

Space packing: An interesting set of results have been obtained in studying the benefits of packing N-dimensional space with different types of polytopes (volumes bounded by planes) for the mse criterion. In fact, in a manner similar to the hexagon result mentioned above for $N = 2$, it has been conjectured by Gersho [47] that, asymptotically for large L and for *any pdf*, the optimum fixed-L vector quantizer partitions the space into cells whose shapes are derived from a specific polytope for each N.

A special set of polytopes, known as parallelohedra, have been studied by Gersho [47] and by Conway and Sloane [21] for the purpose of quantizing a uniform pdf in N dimensions. The centers of these parallelohedra lie on a *lattice* in N-dimensional space.[10] The lattice points are the output values of the quantizer (i.e., they constitute the codebook), and they form a regularly spaced array of points in N-dimensional space. Such a quantizer is termed a *lattice quantizer.* One very attractive feature of lattice quantizers is

[10]The concept of a lattice is of importance in a number of diverse areas, including the geometry of numbers [19], solid-state physics [143], and image processing [29].

that, given an input vector $\boldsymbol{x}$, it is relatively straightforward to compute the nearest lattice output value [22], without having to compare $\boldsymbol{x}$ against all codebook values and without having to store all codebook values. Conway and Sloane [23] have investigated the use of a number of lattices in the vector quantization of a uniform pdf for large L and using the mse criterion. Fig. 15 shows the saving in bits when those lattices are used relative to using cubes as quantization regions. For each dimension, the lattice used is different. For $N = 1$ we have, of course, the uniform cube quantizer. For $N = 2$, it is the hexagonal lattice, and for $N = 3$, the polyhedron is known as the truncated octahedron. Note that the saving in bit rate increases with N for the different lattices except for $N = 9$ and $N = 10$ for which no lattices have been found that perform better than $N = 8$. The value of 0.028 bits for $N = 2$ is the same as that obtained in Example 3 earlier.

Although the results in Fig. 15 were obtained for a uniform pdf, they have wider applicability. One can show that, if entropy coding is used, then for *any pdf*, the lattice quantizers with large L achieve an entropy less than the cube quantizer by an amount equal to the rates shown in Fig. 15. Since from the results of Gish and Pierce the bit rate of the cube quantizer is 0.255 bits above the rate-distortion function, the bit rate for the lattice quantizers is even closer to $R(D)$ by an amount equal to that shown in Fig. 15 for different dimensions. For example, for $N = 8$, the lattice quantizer with entropy coding will have a bit rate that is only $0.255 - 0.109 = 0.146$ bits greater than $R(D)$ for large L. However, because of the potentially excessive amount of storage needed for the codewords, this method may not be practical. In the remainder of the paper we shall discuss methods that do not require entropy coding.

Experimental results: Much of this section has been devoted to theoretical asymptotic vector quantization results. Few attempts have been made to see how these results may apply in practice and for relatively small values of L and N. For a memoryless Gaussian source, the best VQ results with $R = 1$ bit/sample show practically no improvement over scalar quantization for $N = 2$, a reduction in mse of 0.09 dB (about 0.015 bits) for $N = 3$, and a reduction of about 0.52 dB (about 0.087 bits) for $N = 6$ [36], [58], [81], which is a good improvement for $R = 1$ bit. These results were obtained with no entropy coding.

Much greater gains are achievable for certain non-Gaussian sources. For example, for a memoryless Gamma source and $R = 1$ bit/sample, Fischer and Dicharry [36] obtained a substantial reduction in mse for $N = 6$. However, the resulting mse was still approximately 2 dB higher than $D(R)$ and well above the distortion of the constrained-entropy uniform quantizer, which was found by Granzow and Noll [55] to be only 0.7 dB above $D(R)$. However, this result serves to illustrate that, for cases where Lloyd–Max quantization performs poorly relative to $D(R)$, and entropy coding is not desired or cannot be used, one can make good use of vector quantization to improve performance substantially.

Results with a single-pole Gaussian source have also been obtained at $R = 1$ bit/sample, which show that VQ could outperform differential PCM (DPCM) coding with vector lengths $N > 3$ [60]. But, for $N = 7$, the VQ mse was still 1.6 dB higher than $D(R)$.

IV. Scalar versus Vector Quantization of Vector Sources

We indicated in Section II that the redundancy owing to linear dependency (correlation) can be utilized effectively by vector as well as scalar quantization, provided the latter is performed after the appropriate coordinate transformation. In this section we show how best to use scalar quantizers in the quantization of vector components. The purpose for this presentation is twofold. We wish to give the reader tools to help assess how much of the vector quantization performance in a particular application takes advantage of linear dependencies, and to help evaluate whether a scalar or a vector quantizer is more cost effective in that application, given other real-world constraints.

Throughout this section we use the mse as the distortion measure. The first two subsections are devoted to the scalar quantization of vector sources and the last subsection compares scalar to vector quantization. First, we show that, given a total number of bits to allocate to scalar coding of the various vector components, it is best (minimum mse) to allocate different number of bits to components with different variances. A bit-allocation procedure is described which yields the minimum mse. We then show that, if the vector components are correlated, one can reduce the mse further by first performing a decorrelating transformation or rotation on the vector, followed by independent (scalar) quantization of the components of the rotated vector, utilizing the bit-allocation procedure. Finally, we compare the performance of scalar and vector quantizers in a particular application.

The subject matter of this section is very applicable to the coding of linear prediction parameters (such as log-area ratios), which are generally correlated and have different probability distributions and different variances (see Section IV-C). It is also applicable to the transform coding of speech and other waveforms.

A. Bit Allocation

We are given a random vector $\boldsymbol{x}$ whose components x_k have identical pdf shapes but generally unequal variances σ_k^2. (The case of different pdf shapes will be treated below.) We wish to allocate a given number of bits B among the components and use scalar quantizers to quantize each component x_k independently, in such a way as to minimize the average vector distortion. Let R_k be the number of bits allocated to component x_k and let the distortion in quantizing that component be

$$D_k = \mathscr{E}(x_k - y_k)^2 \tag{90}$$

where y_k is the quantized value of x_k. The problem then is to determine the set $\{R_k\}$ so as to minimize the average distortion

$$D = \frac{1}{N}\sum_{k=1}^{N} D_k \tag{91}$$

subject to

$$\frac{1}{N}\sum_{k=1}^{N} R_k = \frac{B}{N} = R \tag{92}$$

where R is the average bit rate per component.

Let us assume for the present that we are operating in the high bit-rate region of the quantization distortion curve for each of the components (usually $R > 3$). Then we can write from the results of Section III-B

$$R_k = \rho + \frac{1}{2}\log_2 \frac{\sigma_k^2}{D_k} \tag{93}$$

where ρ is a constant value that depends on the pdf shape and on the particular scalar quantization method used. (For example, for a Lloyd–Max quantizer, we see from Fig. 11 that $\rho = 0$ for a uniform pdf and $\rho = 0.722$ for a Gaussian pdf.) From (91) and (93) we have

$$D = \frac{1}{N}\sum_{k=1}^{N} \sigma_k^2 2^{-2(R_k-\rho)}. \tag{94}$$

(Note that D in (94) has a slope of $-6.02/N$ decibels/bit, as expected for high bit rates.) The problem now is to find the R_k values that minimize D in (94) subject to (92). Using Lagrange multipliers, one can show that the distortions D_k must be equal

$$D_{\min} = D_k = 2^{-2(R-\rho)}\left[\prod_{k=1}^{N}\sigma_k^2\right]^{1/N}, \qquad \text{for all } k \tag{95}$$

and the bit allocation is given by

$$R_k = R + \frac{1}{2}\log_2 \frac{\sigma_k^2}{\left[\prod_{k=1}^{N}\sigma_k^2\right]^{1/N}}. \tag{96}$$

If the components have different variances, the result implies that the number of bits used to quantize each component will be different.

By allocating a different number of bits to different components we realize a lower distortion than we would, had we used an equal number of bits R to quantize each component. In the latter case the total distortion would be

$$D_0 = 2^{-2(R-\rho)}\frac{1}{N}\sum_{k=1}^{N}\sigma_k^2. \tag{97}$$

Therefore, the ratio of the minimum distortion $D_{\min}$ to the equal allocation distortion D_0 is

$$\frac{D_{\min}}{D_0} = \frac{\left[\prod_{k=1}^{N}\sigma_k^2\right]^{1/N}}{\frac{1}{N}\sum_{k=1}^{N}\sigma_k^2} = \frac{\mathrm{GM}\{\sigma_k^2\}}{\mathrm{AM}\{\sigma_k^2\}} = \gamma \leqslant 1 \tag{98}$$

which is the ratio of the geometric mean to the arithmetic mean of the component variances, with equality iff all variances are equal. Thus with bit allocation, one would have an increase in SNR of $-10\log_{10}\gamma$ decibels. (Note that (98) was derived assuming the same pdf and high bit rates for all components.)

If $\gamma \ll 1$, implying that the component variances have a large dynamic range, then the bit-allocation scheme in (96) would result in a wide range of bit rates R_k. Some of these may be very low (even though the average may be high) and some may even be negative. Low rates would violate our assumption of high rates that allowed us to write (93), and negative rates are, of course, unallowable. The latter condition results for components whose variances are less than $D_{\min}$ in (95). For such cases, one would not transmit anything because by simply using the mean value at the receiver one would have at worst a distortion equal to the signal variance. If such negative bit rates are obtained from the bit-allocation scheme, the corresponding components are allocated zero bits. When negative values of R_k are thus increased to zero, the rates of other components will have to be modified to maintain the same average rate R. Several procedures have been developed which reoptimize the allocation. Segall [120] has solved the above optimal *continuous* bit allocation problem, assuming independent Gaussian components. Fox [43] had earlier presented an algorithm for optimal *integer* bit allocation in the general case where the pdfs of the components may be different. Below we give a brief description of the algorithm and the conditions for its applicability.

Optimum Integer Bit Allocation: Let us assume that we wish to have a bit allocation where all rates R_k are integers. We associate with each component x_k a normalized (unit-variance) quantization distortion function $E_k(b)$, which gives the distortion as a function of the number of bits b, assuming x_k has unit variance. (We assume that each x_k may have a different pdf.) Clearly then, the distortion of x_k is given by

$$D_k(b) = \sigma_k^2 E_k(b). \tag{99}$$

$E_k(b)$ is an actual curve obtained by applying a specific quantizer to the random variable x_k with its specific pdf. The quantizer can be a Lloyd–Max quantizer or a constrained-entropy quantizer or any other quantizer of interest. The main thing is to generate the curve $E_k(b)$ for each component. Of course, if $E_k(b)$ is not optimized in any way to the component x_k, then the results may be suboptimal. (Fig. 11 shows $E(b)$ for a Lloyd–Max quantizer for four different pdfs.)

The bit-allocation procedure simply assigns the next bit to the component that causes the maximum incremental decrease in the overall distortion. The procedure is as follows:

Step 1: For each bit rate b, calculate the incremental decrease in the distortion when a bit is added to each component:

$$\Delta_k(b) = \sigma_k^2[E_k(b-1) - E_k(b)], \qquad 1 \leqslant k \leqslant N, \quad b = 1, 2, \cdots. \tag{100}$$

Step 2: Sort the values $\Delta_k(b)$ in decreasing order.

Step 3: Assign the given bits one by one according to the resulting order.

(In actual implementation, the values $\Delta_k(b)$ are computed sequentially as they are needed in allocating the bits.) This algorithm can also be applied to assigning an integral number of *levels* to each component. The variable b would then refer to the number of levels.

The above algorithm gives the optimal solution when all quantization distortion functions $E_k(b)$ are monotone decreasing and convex, i.e., successive bits decrease the quantization error by an amount smaller than earlier bits allocated to that component. For the general case, where the quantization distortion functions are nonconvex or even nondecreasing, Shoham and Gersho [125] present an algorithm that finds the optimum bit allocation.

Note that in the case where the distortions do not obey (94), the minimum distortions for the different components after bit allocation will generally not be equal.

B. Vector Rotation for Correlated Sources

We saw above how one can take advantage of differences in component variances to decrease the distortion, for a given rate, over that of equal bit allocation. The development did not make any assumptions about possible dependence among components. We present below a procedure that takes advantage of any correlation that may exist among the vector components.

Fig. 16 shows a block diagram for the scalar quantization of a vector source, which includes a vector rotation $\boldsymbol{A}$ before quantization, followed by an inverse rotation $\boldsymbol{B}$ at the receiver. $\{q_k, 1 \leqslant k \leqslant N\}$ are N independent scalar quantizers that quantize the components of the rotated vector $\boldsymbol{u}$. In Section IV-A we considered the case of $\boldsymbol{A} = \boldsymbol{B} = \boldsymbol{I}$. Here we ask the question: What are the rotations $\boldsymbol{A}$ and $\boldsymbol{B}$ and the quantizers $\{q_k\}$ that minimize the distortion at the output, subject to a given bit rate? The answer to this question is not known for general distributions. However, for a source $\boldsymbol{x}$ whose components are jointly Gaussian, Huang and Schultheiss [65] and then Segall [120] showed that the optimal rotation $\boldsymbol{A}$ is given by a matrix $\boldsymbol{S}$ whose rows are the normalized eigenvectors of $\boldsymbol{\Gamma}_x$, the covariance matrix of the vector $\boldsymbol{x}$ (see (13)), and $\boldsymbol{B}$ is the inverse of $\boldsymbol{A}$

$$\boldsymbol{A} = \boldsymbol{S} \qquad \boldsymbol{B} = \boldsymbol{S}^{-1} \tag{101}$$

with

$$\boldsymbol{S}^{-1} = \boldsymbol{S}^T \tag{102}$$

and

$$\boldsymbol{S}\boldsymbol{\Gamma}_x\boldsymbol{S}^T = \boldsymbol{\lambda} = \operatorname{diag}[\lambda_1\lambda_2 \cdots \lambda_N] \tag{103}$$

where $\boldsymbol{\lambda}$ is a diagonal matrix whose elements are the eigenvalues of the covariance matrix $\boldsymbol{\Gamma}_x$. Equation (103) is a well-known property of a symmetric matrix [13]. The eigenmatrix $\boldsymbol{S}$ is known to be an orthogonal matrix. The rotated vector $\boldsymbol{u}$

$$\boldsymbol{u} = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{S}\boldsymbol{x} \tag{104}$$

will then have a covariance matrix

$$\begin{aligned}\boldsymbol{\Gamma}_u &= \mathscr{E}\left[(\boldsymbol{u} - \bar{\boldsymbol{u}})(\boldsymbol{u} - \bar{\boldsymbol{u}})^T\right] \\ &= \boldsymbol{S}\boldsymbol{\Gamma}_x\boldsymbol{S}^T = \boldsymbol{\lambda}.\end{aligned} \tag{105}$$

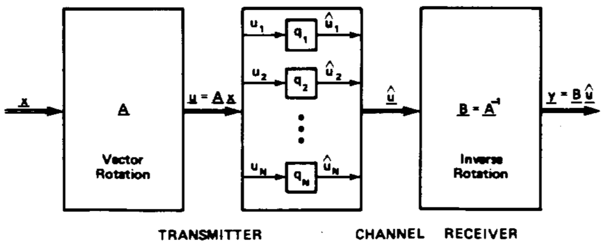

Fig. 16. Scalar quantization of a vector source. The eigenvector rotation $\boldsymbol{A}$ converts the input $\boldsymbol{x}$ into a vector $\boldsymbol{u}$ with uncorrelated components. An inverse rotation after the scalar quantizers gives $\boldsymbol{y}$, the quantized value of $\boldsymbol{x}$.

Therefore, the covariance of $\boldsymbol{u}$ is a diagonal matrix whose elements are the eigenvalues of the covariance $\boldsymbol{\Gamma}_x$. Since $\boldsymbol{\Gamma}_u$ is diagonal, the vector $\boldsymbol{u}$ has uncorrelated components, and since $\boldsymbol{x}$ is Gaussian, $\boldsymbol{u}$ will be Gaussian with independent components. The variance of each component u_k is then equal to the corresponding eigenvalue λ_k.

Using property (102) of an orthogonal matrix, one can show that the distortion at the receiver is equal to the distortion in quantizing the vector $\boldsymbol{u}$. Since

$$\boldsymbol{y} = \boldsymbol{S}^{-1}\hat{\boldsymbol{u}} \tag{106}$$

we have from (104), (106), and (102)

$$D = \frac{1}{N}\mathscr{E}\left[(\boldsymbol{x} - \boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y})\right] = \frac{1}{N}\mathscr{E}\left[(\boldsymbol{u} - \hat{\boldsymbol{u}})^T(\boldsymbol{u} - \hat{\boldsymbol{u}})\right]. \tag{107}$$

Therefore, minimum distortion is obtained by designing quantizers that will minimize the average distortion in quantizing $\boldsymbol{u}$. The optimal scalar quantization scheme is then to quantize the components u_k using the bit-allocation scheme described in Section IV-A.

For the special case of high bit rates, the minimum distortion and the bit allocation are given by (95) and (96), respectively, with λ_k substituted for σ_k^2. The ratio of the minimum distortion after rotation $D'_{\min}$ to the equal bit allocation distortion D_0 is then given by

$$\frac{D'_{\min}}{D_0} = \frac{\left[\prod_{k=1}^{N}\lambda_k\right]^{1/N}}{\frac{1}{N}\sum_{k=1}^{N}\sigma_k^2}. \tag{108}$$

Note that, since [74]

$$\mathrm{GM}\{\lambda_k\} \leqslant \mathrm{GM}\{\sigma_k^2\} \tag{109}$$

we have from (98) and (108)

$$D'_{\min} \leqslant D_{\min} \tag{110}$$

with equality iff the components of the Gaussian vector $\boldsymbol{x}$ are uncorrelated and, hence, the variances are equal to the eigenvalues. In fact, the eigenvector rotation is the transform that minimizes the geometric mean of the resulting component variances. The same transformation, therefore, minimizes the quantization distortion. From (104) and (102), we can also write

$$\begin{aligned}\frac{1}{N}\sum_{k=1}^{N}\sigma_k^2 &= \frac{1}{N}\mathscr{E}\left[(\boldsymbol{x} - \bar{\boldsymbol{x}})^T(\boldsymbol{x} - \bar{\boldsymbol{x}})\right] \\ &= \frac{1}{N}\mathscr{E}\left[(\boldsymbol{u} - \bar{\boldsymbol{u}})^T(\boldsymbol{u} - \bar{\boldsymbol{u}})\right] \\ &= \frac{1}{N}\sum_{k=1}^{N}\lambda_k.\end{aligned} \tag{111}$$

Substituting in (108), we have

$$\frac{D'_{\min}}{D_0} = \frac{\mathrm{GM}\{\lambda_k\}}{\mathrm{AM}\{\lambda_k\}} \leqslant 1. \tag{112}$$

We note that there are many possible decorrelating rotations that can be applied to $\boldsymbol{x}$ to obtain a vector $\boldsymbol{u}$ whose components are uncorrelated. The eigenmatrix $\boldsymbol{S}$ is the one decorrelating matrix that results in the minimum distortion.

Optimal Scalar Quantization: Therefore, for a Gaussian vector process, the scalar quantization procedure that minimizes the mse for a given bit rate is

Step 1: Using the eigenmatrix $\boldsymbol{S}$ derived from the covariance of the vector process, transform (rotate) the input vector $\boldsymbol{x}$ to form another vector $\boldsymbol{u}$.
Step 2: Quantize the components of $\boldsymbol{u}$ using some optimal scalar quantizer and the bit-allocation procedure in Section IV-A. The result is the quantized vector $\hat{\boldsymbol{u}}$.
Step 3: Perform an inverse transform (rotation) on $\hat{\boldsymbol{u}}$ with the matrix $\boldsymbol{S}^T$, resulting in the output vector $\boldsymbol{y}$.

The above procedure is known to be optimal for Gaussian sources, but experience shows that the procedure works well for other distributions. In Example 1, we transformed the vector $\boldsymbol{x}$ in Fig. 5 into the vector $\boldsymbol{u}$ in Fig. 6. One can show that the rotation used is in fact an eigenvector rotation. Note how the marginal densities for the vector components change shape as we go from $\boldsymbol{x}$ to $\boldsymbol{u}$, and the components u_1 and u_2 in this case become independent. With some computation, one can show the interesting result in this example that the reduction in distortion achieved by the eigenvector rotation and the bit allocation is far greater than that predicted by (112). The reason, we believe, is the change in the shape of the marginal densities effected by the rotation.

Transform Coding: The transformation introduced by the eigenvector rotation is sometimes referred to as the *Karhunen–Loève Transform* (KLT) (see [74]), or simply the *eigenvector transform*. The transform involves the computation of the eigenvectors and eigenvalues of the covariance matrix. For a relatively small number of dimensions N, the computation of the eigenvectors can be accomplished via any of a number of standard routines [67]. The eigenvector transform has been used, for example, in quantizing the outputs of a filter bank in a speech vocoder [79] and in quantizing LPC log-area-ratio parameters [45], [112], [116] (see Section IV-C). However, as N increases to values in the hundreds, the amount of computation may make computing the eigenvectors prohibitively expensive. In such cases, one often resorts to simpler orthogonal transforms, such as the discrete cosine transform [3], which, even if not optimal, has been shown to produce very good results for speech and other data [74]. Adaptive transform coding (ATC) [16], [130], [142] is one of the well-known methods for the coding of speech waveforms in the range 8–16 kbits/s.

Distortion Weighting: In Section II-B we mentioned the possibility of using a weighting matrix $\boldsymbol{W}$ to weight the errors in specific components differently. The reason for utilizing such weighting is often guided by perceptual considerations. Minimizing the weighted mse in quantizing $\boldsymbol{x}$ is equivalent to minimizing the mse in quantizing the transformed vector $\tilde{\boldsymbol{x}}$ given in (16). The quantization procedure then follows Fig. 16 with $\tilde{\boldsymbol{x}}$ instead of $\boldsymbol{x}$, and $\boldsymbol{A}$ is the eigenmatrix corresponding to the covariance of $\tilde{\boldsymbol{x}}$, which can be shown to be

$$\boldsymbol{\Gamma}_{\tilde{x}} = \boldsymbol{P}\,\boldsymbol{\Gamma}_x\,\boldsymbol{P}^T \qquad (113)$$

where $\boldsymbol{P}$ is given by (15).

C. Comparisons with Vector Quantization

The performance of VQ in any specific application can be compared to scalar quantization to help assess the benefits accrued by VQ. Such comparisons are particularly important because of the substantial storage and computational costs typically associated with VQ. In this section we present an example to help illustrate the concepts presented and to see the specific benefits in an important application in speech coding.

EXAMPLE 4

In very-low-rate speech coding employing LPC analysis, one generally transmits for each frame a set of LPC coefficients, a gain term, a voiced/unvoiced decision, and a pitch value if the sound is voiced. At an analysis rate of 40 frames/s (i.e., every 25 ms) and 20 bits/frame, for example, the total bit rate will be 800 bits/s. In this paper we shall focus only on the quantization of spectral (LPC) parameters, since that is where vector quantization is mostly used. Typically, about 10–13 bits/frame are used to code the spectrum. Coincidentally, computational and storage costs tend to become unacceptable for applications requiring significantly more bits per frame to code the spectrum.

Fig. 17 shows the relative performance of scalar and vector quantizers for LPC parameters. Speech, low-pass filtered at 5 kHz and sampled at 10 kHz, was recorded from ten male speakers, 1 min of speech from each speaker. LPC analysis with 14 coefficients was performed at a rate of 100 frames/s. The resulting 60 000 frames of data were used to train the scalar and vector quantizers. The LPC coefficients were represented by log-area-ratios (LARs) G_k defined in (22) and the distortion measure was the mse. The mse in Fig. 17 is plotted relative to the distortion at zero bits, which is equal to the average squared value of the LARs for all the data. Fig. 17 shows three plots of mse as a function of bit rate. Plot (a) was obtained using a separate Lloyd–Max

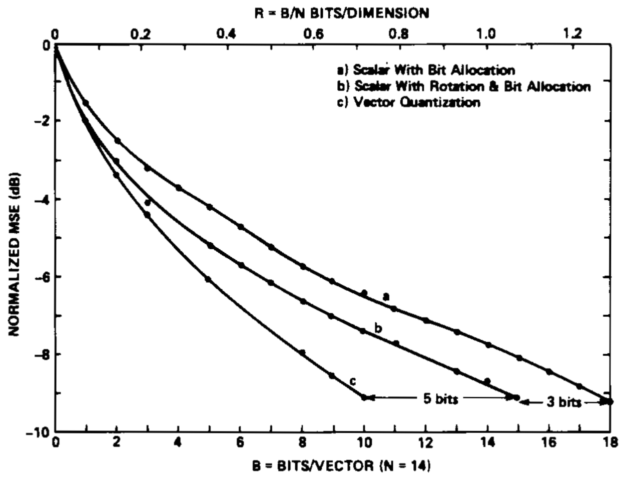

Fig. 17. Normalized mse in quantizing log-area-ratios (LARs) using three methods: (a) scalar quantization with bit allocation; (b) scalar quantization with bit allocation, preceded by eigenvector rotation; and (c) vector quantization. The 3-bit reduction from (a) to (b) takes advantage of linear dependencies (correlation), and the additional 5-bit reduction from (b) to (c) takes advantage largely of nonlinear dependencies.

scalar quantizer for each LAR with the integer bit-allocation scheme described in Section IV-A. Plot (*b*) was obtained by first performing a fixed eigenvector rotation, as described in Section IV-B, followed by bit allocation and Lloyd–Max scalar quantization. Plot (*c*) shows the performance of a vector quantizer whose codebook was designed using the *K*-means algorithm described in Section II-C. For the same mse obtained for the vector quantizer with 10 bits, the scalar quantizer with rotation requires 15 bits, while the scalar quantizer without rotation requires 18 bits. The benefit of 5–8 bits with vector quantization at $B = 10$ bits would be expected to increase as the number of bits is increased, until the slope of the vector quantization curve reaches -0.43 dB/bit ($6.02/14 \approx 0.43$). However, in terms of percentage of total bit rate, the relative benefit of vector over scalar quantization decreases as the bit rate increases.

The 3-bit reduction in bit rate with eigenvector rotation shows the benefit of taking advantage of linear dependency. Much of the additional 5-bit reduction effected by vector quantization takes advantage of nonlinear dependencies among LARs. The existence of such significant nonlinear dependencies in the space of LARs (or other spectral parameters) for speech is what makes vector quantization truly attractive in this application.

The 10-bit vector quantization system and the 15-bit scalar quantization system with rotation were also compared using subjective listening tests. The analysis/synthesis employed variable-frame-rate transmission (see Section VI-A) at an average of 30 frames/s. Pitch, gain, and duration were unquantized. Upon informal listening, no clear difference could be detected between the two systems.

The bit allocation for the scalar quantizers utilized the pdfs of the different components. Fig. 18 shows the estimated (unnormalized) pdfs for G_1, G_2, and G_{14}. The pdfs for G_1 and G_2 are quite asymmetric and their variances are significantly larger than the other LARs. All pdfs for G_4 to G_{14} tend to be Gaussian in shape, as in the pdf of G_{14} shown in the figure. Table 2 shows the bit allocation before and after eigenvector rotation for the first 15 bits. The table shows the parameter index to which each additional bit is allocated. The parameters are the LARs for the case with no rotation and the transformed parameters after rotation (the index in the latter case is that of the corresponding eigenvector with the eigenvalues ordered in decreasing value). For example, the seventh bit was allocated to the sixth LAR in one case and to the parameter corresponding to the fourth eigenvector in the other. Table 3 shows the total number of bits allocated to each parameter for the 15-bit case. Generally, the parameters with larger variance are allocated more bits. Note in both cases how not all parame-

Table 2 Bit Allocation Before and After the Eigenvector Rotation of LARs. The table shows the parameter index to which each bit is allocated.

Bit Number	Parameter Index (*k*) No Rotation	Parameter Index (*k*) With Rotation
1	1	1
2	2	2
3	1	1
4	2	2
5	4	3
6	3	1
7	6	4
8	1	5
9	5	6
10	3	3
11	8	7
12	4	8
13	2	4
14	10	2
15	6	9

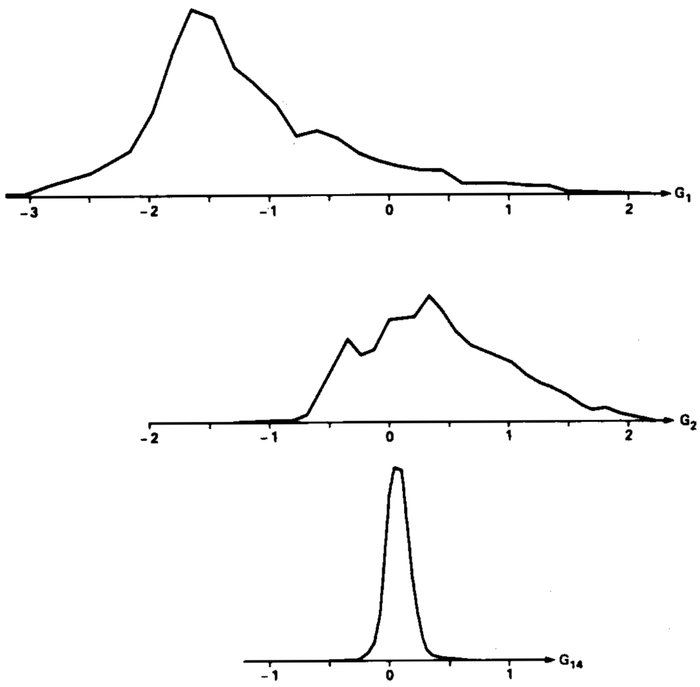

Fig. 18. Histograms of LARs G_1, G_2, and G_{14}. Histograms for G_4 through G_{14} can be modeled well by Gaussian pdfs.

Table 3 Total Number of Bits Allocated to Each Parameter for the 15-Bit Case

Parameter Index (k)	Number of Bits: No Rotation	Number of Bits: With Rotation
1	3	3
2	3	3
3	2	2
4	2	2
5	1	1
6	2	1
7	0	1
8	1	1
9	0	1
10	1	0
11–14	0	0
	15 bits	15 bits

ters are allocated bits. Parameters whose variances are smaller than the expected distortion are allocated zero bits.

Distortion Measure: In the example above, the mse was used in measuring the distortion of quantizing LARs. The same distortion measure was used for the vector and the scalar quantizers. Using the *same* distortion measure to assess the relative performance of quantizers is very important; otherwise, one can obtain misleading results. However, one freedom we have in vector quantization is to choose distortion measures that are not usually defined for the scalar case, such as the Itakura–Saito distortion in (23). It would be unfair, for example, to use the Itakura–Saito distortion to compare the performance of vector and scalar quantizers unless they were both designed using that distortion measure. Unfortunately, it is not simple to design a scalar quantizer using the Itakura–Saito distortion.[11] One, of course, could always compare the performance of any two quantizers fairly by subjective listening tests. We have designed LPC vector quantizers using the Itakura–Saito distortion as well as the mse on LARs and found no clear difference in subjective speech quality between the two methods.

Dimensionality and Bit Allocation: It was clear in the example above that if sufficient bits are not available, then certain vector components may not be allocated any bits with scalar quantization. For cases where vector components are uncorrelated and of equal variance, and $R < 1$ bit, the bit-allocation procedure will have to assign 1 bit to each of several components in some arbitrary fashion and assign zero bits to the other components, such that the average bit rate is R. The major limitation we are witnessing here is that a scalar quantizer cannot assign a fraction of a bit to a component; it must assign either 1 bit or none corresponding to either a two-level or a one-level quantizer (a fractional number of levels is not possible). (The only exception is if entropy coding is used, in which case fractional bits are possible.) A major advantage of the dimensionality of vectors is that fractional bits come in a very natural way, and all vector components are allocated bits in an equitable fashion. It is this aspect of dimensionality, in addition to the ability to specify arbitrary cell shapes, that makes VQ especially important at low rates. These properties of dimensionality are what makes it possible to code the prediction residual at $R < 1$ bit/sample and maintain good speech quality (see Section VII-B).

Vector Sources: The sequence of LPC-parameter vectors in Example 4 constitutes what one might consider as a "naturally occurring" vector source. Each vector is not merely a collection of scalars from a scalar source, but somehow represents a single entity, namely, the short-term spectrum. The VQ of this vector source as described above takes advantage of the dependencies among vector parameters *in parameter space*. VQ theory allows us, however, to form longer vectors from sequences of LPC vectors and perform VQ on the longer vectors, thereby taking advantage of dependencies *in time* between adjacent LPC vectors, and benefiting from the higher dimensionality. (This is precisely what is done in segment quantization, as described in Section VI.) However, the basic VQ process and its properties remain the same whether it is performed on an inherently scalar or vector source.

[11]Strictly speaking, one cannot design an optimal scalar quantizer using the Itakura–Saito distortion, because scalar quantization implies the independent quantization of vector parameters and the Itakura–Saito distortion is not separable into distortions on the individual parameters. However, one could design a product code instead (see Section V-C).

V. Codebook Design

It should be clear now that VQ can offer substantial performance advantages over scalar quantization at very low rates, especially for sources that exhibit significant nonlinear dependencies. Unfortunately, these advantages are obtained at considerable computational and storage costs, as we saw in Section II-D. For full-search coding, the costs are exponential in the number of bits per vector. Computational and storage costs double for each increase of 1 bit in the rate. In Example 4, to perform 10-bit quantization of vectors of 14 LARs, it takes $14 \times 2^{10} = 14\,336$ multiply-adds to quantize *each* input vector. The codebook requires an equal number of storage locations. This number doubles if 11-bit quantization is desired. Very quickly, computational and storage costs become prohibitively expensive as the number of bits/vector increases.

A number of fast-search algorithms have been proposed in the pattern-recognition literature [44], [121], [140] and more recently in VQ [20], [48], which are designed to reduce the computations in a full search. Most of the algorithms are based on geometrical notions in Euclidean spaces, they require preprocessing of the codebook, and tend to trade off multiplications with comparisons and with increased storage requirements. The number of multiplications can be reduced by as much as an order of magnitude.

In this section we present variations on the basic VQ scheme which are intended to reduce computational costs in a very significant manner (linear rather than exponential growth with the number of bits per vector), but at some reduction in performance. Other methods that are designed to reduce storage costs result in relatively greater reduction in performance. The section ends with a discussion of training and testing issues and codebook robustness.

A. Binary Search

With the K-means algorithm (with $K = L$ levels), a *full search* of the L code vectors is required to quantize each input vector. *Binary search* [18], [112], known in the pattern-recognition literature as hierarchical clustering [5], [64], is a method for partitioning space in such a way that the search for the minimum-distortion code vector is proportional to $\log_2 L$ rather than L. Specifically, N-dimensional space is first divided into two regions (using the K-means algorithm with $K = 2$), then each of the two regions is divided further into two subregions, and so on, until the space is divided into L regions or cells. Here, L is restricted to be a power of 2, $L = 2^B$, where B is an integral number of bits. (A relaxation of this condition will be discussed below.) Associated with each region at each binary division is its centroid. Fig. 19 is a schematic of binary division of space into $L = 8$ cells. At the first binary

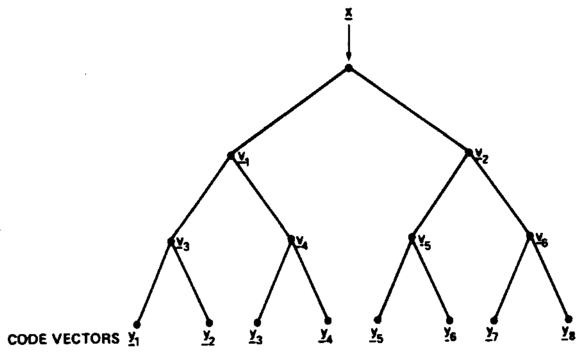

Fig. 19. Uniform tree for a binary-search vector quantizer. The vectors v_i are intermediate code vectors that are compared with the input vector x. The code vectors are the vectors y_i. The codebook size L is restricted to be a power of 2 in a uniform binary search.

division, v_1 and v_2 are the region centroids. At the second binary division, there are four regions with centroids v_3 through v_6. The centroids of the regions after the third binary division are the code vectors y_i. An input vector x is quantized by traversing (searching) the tree in Fig. 19 along a path that gives the minimum distortion at each node in the path. Thus x is compared to v_1 and v_2. If $d(x, v_2) < d(x, v_1)$, for example, then the path leading to v_2 is taken. Next x is compared to v_5 and v_6. If, for example, $d(x, v_5) < d(x, v_6)$, then the path to v_5 is taken and x is finally compared to y_5 and y_6. If $d(x, y_6) < d(x, y_5)$, then y_6 is chosen as the quantized value of x. Clearly, the total number of distortion computations is, in general, equal to $2 \log_2 L$. Again assuming N multiply-adds for each distortion computation, we have a total computational cost of

$$\mathscr{C} = 2N \log_2 L = 2NB \qquad \text{(binary search)} \qquad (114)$$

which is only *linear* with the number of bits. In the example of 10-bit quantization of 14 LARs, the number of computations is now only 280, as compared to 14 336 with full search. (The 280 figure is close to the cost of scalar quantization when rotation is used.) Thus a vast reduction in computation has been effected. The storage cost, however, has increased. Note in Fig. 19 that, in addition to storing the code vectors y_i, one must also store all the intermediate vectors v. The total storage cost can be shown to have approximately doubled:

$$\mathscr{M} = 2N(L - 2) \qquad \text{(binary search)}. \qquad (115)$$

Also, we shall see below that an additional price is that a certain loss in performance takes place.

The cost requirements in (114) and (115) can be cut in half if the mse is used as the distortion measure. In that case, instead of comparing x to two vectors, one can test where x lies relative to the hyperplane that separates the two regions. Such a test involves the computation of a single scalar dot product of two vectors.

We call the quantization tree depicted in Fig. 19 *uniform*, in that *all* regions at any stage are each divided into two subregions. When performing the actual training to obtain the quantization tree, there may result one or more clusters at certain points in the subdivision process which contain very few training samples, perhaps even one sample. Such clusters would be essentially wasting bits because any further subdivision of those clusters would not reduce the distortion. To ensure a lower average distortion and maximize the utilization of bits, we do not branch the tree uniformly. Specifically, in the training phase, at each point in the subdivision process, the total distortion contributed by each cluster is examined. The cluster that contributes the largest amount of distortion is subdivided next, and the process is repeated. The result is typically a *nonuniform* tree, as depicted in Fig. 20. Note that, in this case, the

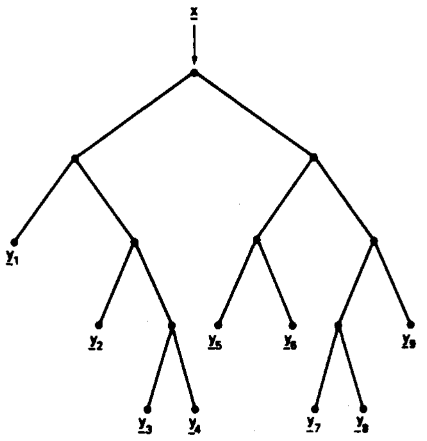

Fig. 20. Nonuniform tree for a binary-search vector quantizer. The codebook size in this case can be any integer.

number of levels can be any desired integer; it is not restricted to be an integral power of 2. A nonuniform tree with $L = 9$ is shown in Fig. 20.

To evaluate the relative performance of binary search (uniform and nonuniform) compared to the full search we give a specific example from very-low-rate speech coding. Fig. 21 shows the average distortion as a function of the number of bits per vector for the quantization of LPC frames of 14 LARs each. As expected, for a given distortion, full search has the lowest rate, followed by nonuniform binary, followed by uniform binary. For the highest rates shown in Fig. 21, the difference between full search and nonuniform binary is approximately 0.5 bits only. For many applications, this small difference in performance is well worth the savings in computation effected by binary search. The plots in Fig. 21 were obtained from the speech of 15 male speakers. For a single speaker, the difference between nonuniform binary and full search is typically larger (about 0.7 bits at $B = 8$ bits).

Binary search is a special case of a class of VQ methods known as *tree-searched* VQ, with the binary search being the simplest. In general, one could divide the space at each node in the tree into more than two subregions (using

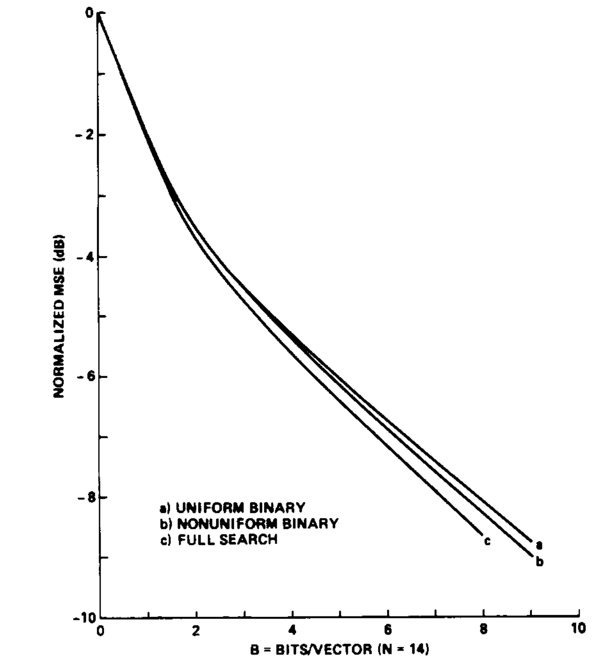

Fig. 21. Comparison of mse when quantizing LARs with three types of vector quantization: (*a*) uniform binary search; (*b*) nonuniform binary search; and (*c*) full search.

K-means with $K > 2$). Such a method would result in an increase in computation over binary search, with some increase in performance. However, because the performance of binary search is usually quite close to full search for many applications, it is typically most cost effective to use binary search with a nonuniform tree.

B. Cascaded Quantization

The major advantage of binary search was the substantial decrease in computational cost relative to full search, accompanied by a relatively small decrease in performance. However, the storage cost was not reduced; in fact, for non-mse distortions the memory cost doubled relative to full search. *Cascaded VQ* is a method intended to reduce storage as well as computational cost [76], [112]. As the name implies, cascaded (also known as multistage) VQ consists of a sequence of VQ stages, each operating on the "residual" of the previous stage. A two-stage cascade is depicted in Fig. 22. The input vector $\boldsymbol{x}$ is first quantized using a B_1-bit (L_1-level) vector quantizer (the quantizer can use full search or binary search, as desired). The residual or "error" $\boldsymbol{e}$ between $\boldsymbol{x}$ and its quantized value $\boldsymbol{z}_i$ is then used as the input to a B_2-bit (L_2-level) second VQ stage with output $\boldsymbol{w}_j$. (The matrix $\boldsymbol{A}_i$ plays a useful role that is explained below; here, we assume that $\boldsymbol{A}_i = \boldsymbol{I}$ and $\boldsymbol{u} = \boldsymbol{e}$.) The final quantized value of $\boldsymbol{x}$ is then simply the sum of the two vectors $\boldsymbol{z}_i$ and $\boldsymbol{w}_j$

$$q(\boldsymbol{x}) = \boldsymbol{y} = \boldsymbol{z}_i + \boldsymbol{w}_j \qquad (\boldsymbol{A}_i = \boldsymbol{I}). \tag{116}$$

During the training phase, the residuals $\boldsymbol{e}$ from the first stage are pooled together and treated as a new random vector that is to be quantized. The process can be repeated for as many stages as desired.

There can be confusion between tree-searched VQ and cascaded VQ. While both processes take place in stages, what one does at each stage is different in the two methods. In tree-searched VQ, one is trying to find the desired

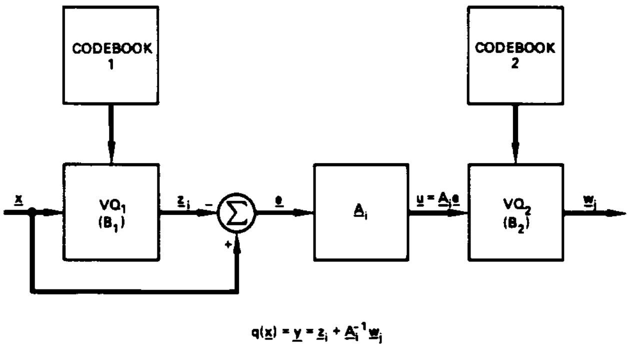

Fig. 22. A two-stage cascaded vector quantizer. In the first stage, a B_1-bit vector quantizer quantizes $\boldsymbol{x}$ into a particular vector $\boldsymbol{z}_i$. The "residual" vector $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{z}_i$ is then quantized using a B_2-bit quantizer. (The rotation $\boldsymbol{A}_i$ is used to "realign" all the residual vectors to reduce the distortion.) The quantized value of $\boldsymbol{x}$ is then given as the sum of the two code vectors, as shown in the figure.

code vector by searching the space in a systematic manner; at each step, one gets closer and closer to the desired code vector. The role played by the intermediate vectors $\boldsymbol{v}$ in Fig. 19 is simply to guide the search; the desired code vector is found at the end of the search. In cascaded VQ, each stage is a complete VQ search of the space which yields a separate code vector; the final quantized value of the input $\boldsymbol{x}$ is then the *sum* of the code vectors found at each of the stages. The input to each stage is the difference (residual) between the chosen code vector and the input for the previous stage.

The computational and storage costs for a two-stage cascaded vector quantizer (assuming *K*-means for each stage) are simply

$$\mathscr{C} = N(L_1 + L_2) \qquad \text{(cascade)} \tag{117}$$

$$\mathscr{M} = N(L_1 + L_2) \tag{118}$$

instead of NL_1L_2 for a single-stage full-search quantization. For the example of 10-bit quantization of 14 LARs, the cost of two-stage cascaded VQ with $B_1 = B_2 = 5$ bits is $14(2^5 + 2^5) = 896$ multiply-adds as compared to 14336 multiply-adds for single-stage VQ. Also, storage cost is reduced in the same proportion. What we are sacrificing for this significant reduction in cost is a loss in performance, as we shall see below.

We use as our example 14-LAR quantization in speech coding. Fig. 23 shows the mse distortion as a function of bit rate for several VQ schemes. Plot (*d*) shows the performance of a nonuniform binary quantizer. Plot (*a*) shows the performance of a cascaded vector quantizer with each stage quantizing to only 1 bit. Clearly, this constitutes the most extreme form of cascaded VQ, with a maximum reduction in cost. As can be seen from plots (*a*) and (*d*), the performance of this cascaded quantizer is significantly worse than the single-stage binary vector quantizer for each bit rate.

What causes this vast reduction in performance for the cascaded quantizer? Recall that during the training phase, after each stage, the residuals are pooled together to form the input to the next stage. If all the clusters in the first stage possess the same relative pdf structure within the cluster, the sets of residuals from the different clusters will have the same pdfs and pooling them together should not reduce performance. However, in general, the residual pdfs from the different clusters will be different and pooling them together will result in a single pdf that will lose many

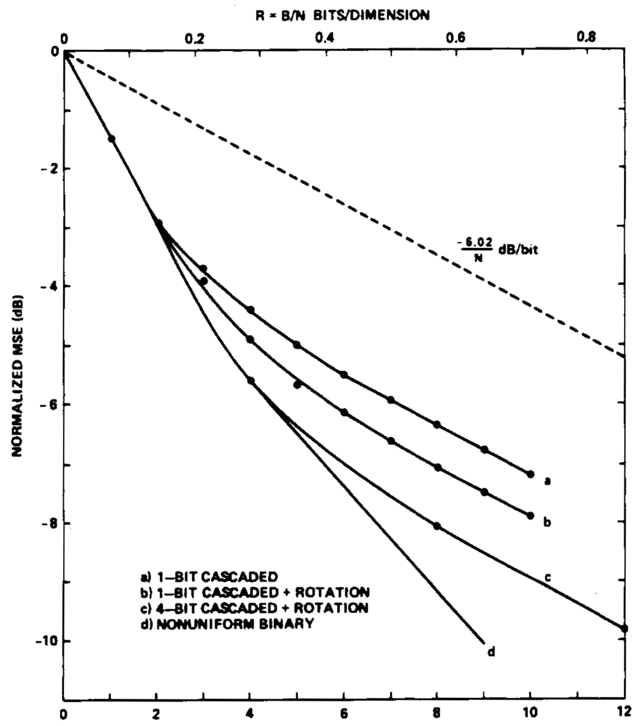

Fig. 23. A comparison of the mse in quantizing LARs using four vector quantizers: (*a*) multistage (10 stages shown) 1-bit cascaded vector quantization with no rotation between stages; (*b*) same as (*a*) but with rotation between stages; (*c*) 3-stage, 4-bit cascaded vector quantization with rotation (each stage used a nonuniform binary search); and (*d*) one-stage vector quantization with a nonuniform binary search.

of the dependencies that existed in the initial clusters. As a result, the distortion-rate curve will reach its asymptotic behavior of $-6.02/N$ decibels per bit at a higher distortion level. We see from plot (*a*) that it reaches its asymptotic behavior at 7 bits and a distortion of -6 dB.

In an effort to forestall the loss of dependence among residual components, we have inserted a vector rotation $\boldsymbol{A}_i$ after each stage [112], as shown in Fig. 22. The idea is to attempt to at least preserve some of the linear dependencies that exist among residual components before they are pooled together. The matrix $\boldsymbol{A}_i$ is set equal to the eigenmatrix corresponding to the covariance matrix of the residuals from the ith cluster. In this manner, the residuals from the different clusters are rotated so that their covariance matrices will have the same structure. Therefore, the covariance of $\boldsymbol{u} = \boldsymbol{A}_i\boldsymbol{e}$ resulting from each cluster will have the same structure. So, pooling the vectors $\boldsymbol{u}$ from the different clusters together will keep the same covariance structure. Fig. 23, plot (*b*), shows the performance of a 1-bit cascaded quantizer with vector rotation. Note how the asymptotic behavior begins at a lower distortion level. Plot (*b*) shows about a 1.8-bit advantage over plot (*a*) for the same distortion at the higher rates. This advantage with using rotation would be expected to be smaller if each stage in the cascade employed a larger number of bits.

Plot (*c*) in Fig. 23 shows the performance of a 4-bit, 3-stage, cascaded vector quantizer with eigenvector rotation after each stage. The 4-bit VQ in each stage utilized a binary tree search. (That is why plots (*c*) and (*d*) coincide up till $B = 4$ bits.) The asymptotic behavior is reached after the second stage at a distortion level of about -8 dB. The sharp departure of plot (*c*) from plot (*d*) at 4 bits is a clear indicator of the devastating effect that pooling residuals has, even if eigenvector rotation is utilized. The nonlinear dependencies among LARs are so important and they are virtually destroyed by the pooling of residuals. In fact, the performance for $B > 4$ bits in plot (*c*) is comparable to the performance of a scalar quantizer at that point. So, there may not be any need to go through the expense of additional vector quantizers after the first stage; a scalar quantizer (with rotation) could effectively replace all stages beyond the first.

The improvement in performance afforded by the vector rotation between the two stages of the cascade is obtained at a cost of additional computation and storage. The additional cost can be seen to be

$$\mathscr{C} = 2N^2 \qquad \text{(rotation)} \qquad (119)$$

$$\mathscr{M} = N^2 L_1. \qquad (120)$$

The storage cost is especially large relative to that for the cascade in (118). Because of these additional costs, the inclusion of the rotation becomes less attractive as a method for improving performance.

It appears, therefore, that the maximum that one can effectively utilize in practice is about two stages of cascaded quantization. The first stage would employ VQ and should be allocated the maximum number of bits that one can afford in terms of computations and storage. The second stage would then perform scalar quantization on the residuals from the first stage. However, if the bit rate for the second stage is less than 1 bit/parameter ($R < 1$), then VQ would be advisable for the second stage as well. Also, if the distortion measure is not conveniently defined in the scalar case (such as the Itakura–Saito distortion), then VQ would have to be used for all stages.

Storage Reduction: Cascaded quantization was designed chiefly for the purpose of reducing storage (though computations are reduced as well), for in tree-searched VQ we already have a very effective method for reducing computations with relatively little loss in performance and a modest increase in storage. In fact, with binary search, for example, one could claim with some justification that, for many applications, computational cost is no longer a major limitation in VQ. If storage cannot be reduced without a concomitant major reduction in performance, then one would have to conclude that *storage cost is ultimately the major limitation in VQ*, for it would prevent us from taking advantage of the power of very large codebooks. With binary search, many would not hesitate to use a 30-bit codebook, but the required storage would tax the addressing capabilities of most of today's computers. Furthermore, the amount of training data needed to design such a codebook can be a serious limitation as well. While much research has been devoted to reducing computational costs, relatively little effort has been expended specifically in reducing storage costs. And little or no work appears to have been done on reducing storage at the expense of increased computation instead of decreased performance. Below, we present another approach to reducing storage costs, namely, product codes. This approach also reduces computations and results in reduced performance, but it could serve as a more attractive solution for many applications.

C. Product Codes

Assume that we structure our parameter space such that two or more component vectors, each with its own codebook, jointly represent all points in the space. The Cartesian product of the component codebooks is known as a *product code*. If, for example, we structure parameter space in terms of two component vectors of dimensions N_1 and N_2, and let the corresponding codebooks be of sizes L_1 and L_2, respectively, then the product code of dimension N and size L, where

$$N = N_1 + N_2 \qquad L = L_1 L_2 \tag{121}$$

will need memory storage equal to the sum of the storage requirements of the two component codebooks

$$\mathcal{M} = N_1 L_1 + N_2 L_2 \qquad \text{(product code)}. \tag{122}$$

Compared to a storage cost of NL for an N-dimensional codebook of size L, we see that the storage costs for a product code can be substantially lower.[12]

While storage costs are reduced through the use of a product code, computational costs are reduced only by using certain types of distortion measures or by making simplifying assumptions and approximations. To illustrate the problem, assume we want to use a product code for LARs, i.e., have a separate codebook for each LAR (see Example 4), but instead of the mse use the Itakura–Saito distortion[13] to find the nearest code vector. Then the only way to quantize an input LAR vector would be to perform a *full search* of the product code for the nearest code vector, because unlike the mse, the Itakura–Saito distortion is not separable into distortions on the individual LARs. The result is that there is no reduction in computation relative to the full-search codebook designed by K-means.

There are at least two conditions under which computational costs for quantization with a product code can be reduced in a comparable manner to the storage cost reduction in (122): *independent* quantization and *sequential* quantization. Under independent quantization, the component vectors are quantized independently using their respective codebooks. If the distortion measure is *separable* into independent distortions on the individual vectors, then independent quantization will, in fact, give the nearest product code vector. The mse is one example of a distortion measure that is separable in this fashion.

Under sequential quantization, one of the component vectors is quantized first, then the quantized value of that vector is used in the quantization of a second component vector, and so on. (Clearly, sequential quantization includes independent quantization as a special case.) The LPC "gain–shape" product code [18], [114], using a different definition of the Itakura–Saito distortion, is one example where the nearest product code vector can be obtained by quantizing the LPC parameters (shape) first, then using the quantized LPC parameters to quantize the gain such that the overall distortion is minimized.

In practice, sequential quantization is often used as an approximate suboptimal solution so as to minimize computations. One example is in a residual-excited speech coder, where the product code is the product of the spectrum codebook and the residual codebook. In one of the known implementations, the spectral envelope is quantized first, then the residual is computed using the values of the quantized spectrum, and finally the residual is quantized. This sequential quantization procedure does not minimize any known overall distortion on the reconstructed speech waveform. (See Section VII-B for a different example of a residual-based product code.)

Thus far we have considered only the quantization process in using a product code. Another important consideration is the design of the component codebooks in the first place. If an optimal (minimum-distortion) product code is desired, then, in general, the component codebooks cannot be designed independently unless the component vectors are statistically independent *and* the distortion measure is separable. In practice, independent or sequential quantization is employed in the design process to reduce computations even if the final product code is suboptimal.

In structuring our parameter space into two or more component vectors, a question arises as how best to define the component vectors and allocate the bits among the respective codebooks. Ideally, one would want to attempt to *structure parameter space such that the component vectors are statistically independent*, as much as possible. For if the component vectors were independent, then the only reduction in performance would be owing to the reduced dimensionality of the vectors. Otherwise, we would expect an additional reduction in performance in proportion to the dependence between the component vectors. While it is possible to remove linear dependencies by proper vector rotations, as we saw in Section IV, removing nonlinear dependencies is a difficult task, in general.

To maximize performance, it should be clear that, as a general design criterion, the number of component codebooks should be as low as possible and the size of each codebook as large as can be afforded in storage. The problem of how to choose the sizes of the component codebooks is a bit-allocation problem similar to the one discussed in Section IV-A.

In comparing a two-vector product code and two-stage cascaded quantization, we note that if L_1 and L_2 in (122) are equal to L_1 and L_2 in (118), then the cascade storage would be greater than the product code storage. But a fair comparison between the performance of the two methods would be to design the component codebooks in each case such that the memory costs are equal, then compare the resulting distortion. From (118) and (122), that would mean that the sizes of the component codebooks in each case would have to be different. Which of the two methods would be expected to perform better in practice depends on the particular problem and the structure of the statistical dependencies.

Below we mention briefly two specific methods of using product codes for coding LPC spectra, which can be used as alternatives to cascaded quantization. In Section VII-B we present an effective method for utilizing product codes in speech waveform coding.

[12] It is possible to structure parameter space such that $N_1 + N_2$ is larger than the dimension of the original space. (One example is structuring a speech waveform into two vectors: one representing the spectral envelope and another representing the residual.) In that case one has to compare (122) with $N'L$, where N' is the dimension of the original space.

[13] Computing the Itakura–Saito distortion between two LAR vectors would necessitate converting the LAR vectors to predictor coefficients first and then using (23).

Split-Vector Codes: In this method, a vector (such as a LAR vector) is split into two or more subvectors and each subvector is coded independently using a codebook designed for that part of the vector. Consider again the coding of LAR vectors in Example 4 above using a total of 15 bits/vector for 14 LARs, and assume we are interested in comparing a product code with a two-stage cascade for a given storage cost. In particular, assume that the storage cost is the minimum possible for the cascade case, which, from (118) and (121), can be seen to occur for $L_1 = L_2 = 2^{7.5}$, i.e., 7.5 bits for each of the two codebooks, so that $\mathcal{M} = 5068$ for a 14-dimensional vector. A product code can be designed using the bit-allocation table in Table 3 so that the storage is similar. A good solution would be to split the 14-LAR vector into two subvectors and use two codebooks: a 4-dimensional, 10-bit codebook corresponding to parameters 1–4, and a 10-dimensional, 5-bit codebook corresponding to parameters 5–14. The resulting storage cost from (122) is 4416. One would then compare the distortions for the two separate cases. (We note from Table 3 that a similar product code could be designed for the rotated parameters, which may perform better than a product code on the LARs.) To our knowledge, the suggestion to use split-vector codes for LPC parameters is novel. As yet no comparisons have been made with cascaded quantization.

Split-Band Codes: In this method, the spectral envelope is split into two or more bands and each band is vector quantized separately. Copperi and Sereno [24] designed a residual-excited coder in which the signal was split into two equal frequency bands, and separate LPC model parameters for each band were vector quantized separately. Using the split-band approach allows the designer to choose lower order spectral models and to make use of the fact that human perception is generally less sensitive to high-frequency distortions than to low-frequency distortions. This method is reminiscent of sub-band coding [74] except that here we attempt to keep the number of bands to a minimum (preferably not more than two) and the spectrum in each band is modeled explicitly.

D. Random Codebooks

While it is important to reduce the costs associated with the vector *quantization* process, there are times when reducing the costs in the *training* process is of interest. One simple method to design a codebook with essentially no computational training cost is to choose the code vectors at random from a given set of training data. We call the resulting codebook a *random codebook*.

At first glance it would appear that a random codebook would not perform well for quantization purposes. However, results on the asymptotic (large L and N) optimality of the expected performance of random quantizers [115] and the use of random codebooks in proving the basic rate-distortion theorems is a strong motivation for the use of such codebooks. While the use of a random codebook is a reasonable choice for large L and N, it is nevertheless used in practice when these conditions are not satisfied. Random codebook selection and performance under these non-asymptotic conditions is a research area of current interest.

In Fig. 24 we show the relative performance of random and nonuniform binary codebooks for the quantization of 14-LAR vectors using the mse distortion measure. The training data consisted of 15 min of speech (1 min from each of 15 speakers), which comprised approximately 90 000 training vectors The random codebook was obtained by selecting a set of vectors from the training data at equal intervals. Thus if a 100-level random codebook was desired, we chose every 900th vector from the training data. The test data consisted of sentences from five male speakers not used in the training. Note how at low bit rates the random codebook performs significantly worse than the binary codebook. However, the two curves begin to converge as the bit rate increases. We would expect the curves to get even closer at higher rates, but asymptotically they would become parallel to each other with some separation in bits. The difference in performance of about 1.3 bits at 10-bit quantization is smaller than one might expect.

We must bear in mind that, although a random codebook is simple to design, it is a full-search codebook and so

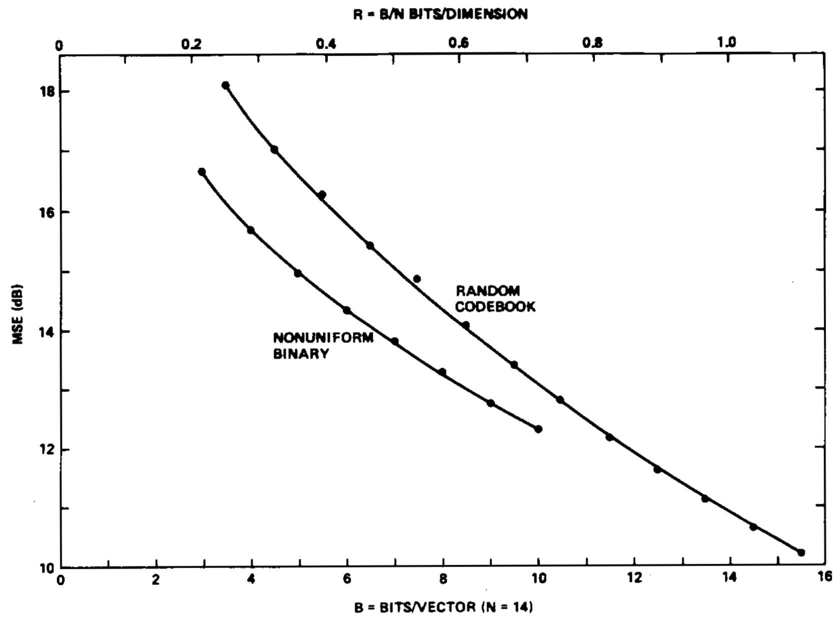

Fig. 24. The mse in quantizing LARs using a random codebook, compared with a nonuniform binary-structured codebook.

requires the same computation and storage. We find random codebooks to be useful in exploring how the VQ distortion decreases as a function of bit rate for higher data rates. In Section VI we mention another useful application of random codebooks.

E. Training and Testing

An important aspect of the design of any codebook is the training procedure used to populate the codebook, i.e., to fill its entries. In the K-means algorithm, the training procedure was given in Section II-C, except for a crucial initialization step where one chooses an initial set of code vectors. Below we describe initialization procedures that we have found useful for the VQ of LPC parameters in speech coding. The testing of the resulting codebook and its robustness are discussed next.

1) Training:

Binary clustering: In binary search we need at each stage a method to initialize the partitioning of space into two regions. Ideally, what we would like to do is to draw a hyperplane through the mean of the training data and orthogonal to the largest eigenvector (we assume a mse distortion). Such a hyperplane would divide space into two initial regions from which to start the iterative training procedure. For data where one dimension has a larger variance than the other dimensions, we approximate the eigenvector direction by the dimension with the largest variance. The centroids of the two clusters separated by the orthogonal hyperplane are used as the initial two code vectors. The K-means algorithm is then continued with $K = 2$ to determine the final code vectors. (We have found 5 to 10 iterations sufficient to obtain convergence.) After the first partition is completed, each of the subregions is divided into two using the same procedure, etc.

K-means initialization: Because the K-means is not guaranteed to result in a codebook that is globally optimum, it is often suggested that one repeat the algorithm with a number of different initial sets of code vectors. The codebook that results in the minimum distortion is then chosen for actual use. The pattern-recognition and VQ literatures contain a number of initialization methods for K-means [30], [56], [64], [99].

Because the performance of binary search is close to the optimal K-means (see Fig. 21), we have found the codebook generated by nonuniform binary search as a good initial codebook to start the K-means iteration. This initialization method has produced better results than using a random initial codebook.

2) Testing: After a codebook is designed with a given set of training data, it is important to test the performance of the codebook on independent data that were not used in the training. Testing only on the training data always presents an overly optimistic view of how the codebook will perform on operational data.

Fig. 25 shows the mse distortion for a 6-bit codebook ($L = 64$ levels) of 14-LAR vectors. The mse is plotted against the number of vectors in the training data. The training data were taken from the speech of 15 male speakers. The resulting codebook for each set of training data was tested on the same training data and on an independent set of test data taken from 5 male speakers who were not used in the training. With increased training, the convergence of the two plots increases. However, the convergence is very slow beyond 20 training vectors per codebook level (i.e., per

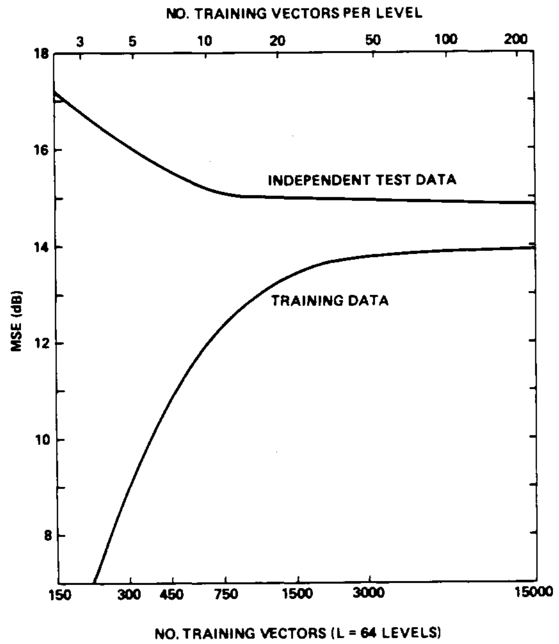

Fig. 25. The mse in quantizing LARs for a 6-bit codebook ($L = 64$ levels) as a function of the number of training data vectors, when tested on the training data and on an independent data set. The smaller the gap between the two curves, the more robust is the codebook.

code vector). As a rule of thumb, using 50 training data per code vector is considered sufficient for most applications. If sufficient data are not available or if the computation or storage becomes excessive, then using as few as 10 training data per code vector may be adequate. The remaining difference between the two plots in Fig. 25 is due to the fact that the test speakers were different from the training speakers. This issue relates to codebook *robustness*, which is discussed below.

In general, convergence of training and test performance curves with increased training set size is not guaranteed for arbitrary sources. One theoretical result of interest here is that, for reasonable distortion measures, such convergence is guaranteed for a class of sources known as *asymptotically mean stationary* [59], in which the stationarity condition is relaxed. Speech from a single speaker under a fixed set of environmental conditions, for example, exhibits short-term and long-term stationary properties and is well modeled as an asymptotically mean stationary source. However, the model may not apply as well under more variable, nonstationary conditions.

3) Codebook Robustness: Codebook robustness refers to the resistance of a codebook to degraded performance when tested on data whose distribution is different from that of the training data. Under operational conditions, one cannot usually predict all of the situations under which a quantizer will be used, and so the distribution of the operational data will in general be different from that of the training data. (Fischer and Dicharry [36] have studied the effects of designing codebooks for particular memoryless sources and testing on others with different pdfs.) We differentiate two major types of variabilities that impact the design and operational performance of a codebook: input signal variability and digital transmission channel errors. We discuss each of these below and compare VQ to scalar quantization.

For speech, signal variability can be classified further into speaker variability and environmental variability. Intraspeaker variability is that due to changes in each speaker's voice: normal everyday changes, changes due to different health conditions, and changes brought about by various emotional states. Interspeaker variability refers to voice differences across speakers. Environmental variability refers to the level and type of background noise that surrounds the speaker (e.g., office environment, outdoors, helicopter, etc.) and signal pickup characteristics, including the types of microphones and transmission facilities (e.g., telephone, radio). The performance of any codebook would be expected to deteriorate if used with types of signals for which it had not been designed. Thus if a codebook is designed (trained) for the voice of one speaker, it would not be expected to perform as well for other speakers. Likewise, a codebook trained in an office environment would not be expected to perform as well in a helicopter environment, for example. It is clear that, for optimal performance, one should train the codebook using data that are representative of what the VQ system will meet in actual operation.

The two plots of Fig. 25 would have converged even more had the test data been taken from the same speakers used in the training. The remaining difference of about 1 dB between training and test data in Fig. 25 is significant. Increasing the amount of training data from the same 15 male speakers will not reduce the difference appreciably, it will simply render the codebook better at quantizing the data from those same 15 speakers. If what is desired is the best possible speaker-independent performance, then the gap between training and test shown in Fig. 25 can be bridged further only by increasing the number of speakers in the training rather than by increasing the amount of speech from each speaker. If the final system is to be used with female speech as well, then including female speakers in the training becomes important.

For a given bit rate, a speaker-independent codebook cannot possibly perform as well as a speaker-dependent codebook. (We have seen differences in performance between 1 to 2 bits per vector for LAR quantization around 10 bits.) One interesting possibility for maximizing performance of a VQ system is to design a speaker-independent codebook initially. Then, as the system is used, it adapts to the speech of the new speaker. Such a system would have the extra advantage of also automatically adapting to the acoustic environment of the speaker. Codebook adaptation means that the code vectors will change in time, which necessitates transmitting the new code vectors to the receiver as well. Two such systems were designed by Paul [100] for real-time speech coding at 800 bits/s. In one system, a full-search codebook minimizes the maximum distortion between the input vectors and the code vectors. When the quantization distortion exceeds a certain threshold, the input vector is taken as a new code vector and the least used code vector up to that point is discarded. The new code vector is then transmitted to the receiver.

Scalar quantizers are generally more robust to signal variability than vector quantizers. For the same number of bits per vector, a scalar quantizer usually has fewer output levels than the vector quantizer. (For example, a 10-bit vector quantizer has 1024 levels, while 10 1-bit scalar quantizers have a total of only 20 output levels.) Therefore, given the same amount of training data, the scalar quantizer will have more training data per level and, hence, the resulting quantizer will be more robust. Another more important reason for the greater robustness of the scalar quantizer is exactly the same reason that renders its performance lower than the vector quantizer. The scalar quantizer, by "averaging" or "projecting" the data on one dimension at a time, is less able to model the detailed multidimensional dependencies in the data and, therefore, will be less susceptible to changes in those detailed dependencies. In other words, the one-dimensional pdfs are expected to change less between training and test than the corresponding multidimensional pdfs. Because vector quantizers are not in general as robust as scalar quantizers, it is very important to test vector quantizers extensively on independent data. Also, whenever changes in the algorithm are made, a new set of test data should be used. Otherwise, there might be a tendency to unconsciously change the algorithm to improve performance on the same test set, effectively rendering the test set part of the training data.

Transmission channel errors present a different type of problem to system robustness. Channel errors translate directly into distortions in the output; the higher the error rate, the greater the distortion. Again, in general, VQ systems tend to be less robust to random channel errors than scalar quantizers. (Burst errors may affect both types of quantizers equally.) A simple example will illustrate the difference. Let us take the example of the 10-bit vector quantizer versus the ten 1-bit scalar quantizers, and assume a channel error rate of 1 percent. Then, in 100 bits there will be 1 bit that is in error, on the average. In the scalar quantizer, that 1-bit error causes one value in one dimension to be wrong, while in the vector quantizer, the same 1-bit error causes a whole 10-bit vector to be wrong, which would result in larger distortions on the average.

By using extra transmission bits, standard error-correcting techniques can be employed to help correct for channel errors [103]. The use of such techniques is strongly recommended for high error rates (> 0.1-percent errors). In an unstructured codebook, all bits in the code vector would have to be protected in some fashion, while in structured codebooks (binary-structured or scalar), one can choose to protect only those bits that are most important to the fidelity of the output.

Channel errors can play havoc with any system that employs variable-length coding (such as entropy coding). A single bit error affects not only the one value in which it occurs but can also affect many succeeding values. The error causes the receiver to lose synchronization with the input and succeeding bits may be erroneously decoded. The use of "self-synchronizing codes" [105] reduces the magnitude of the problem significantly. However, independent synchronization schemes, involving the transmission of additional synchronization bits, would still be needed to avoid loss of synchronization.

It should be clear by now that simply optimizing system performance over a set of training data is only one aspect of choosing the best system for some application. Overall robustness of the system to the operational environment must be included in system performance assessment.

VI. Time-Dependent Vector Quantization

Thus far, we have assumed that each input vector is quantized independently of any other input vector. While we have considered dependencies among vector components (in short-term spectral space, for example), we have not yet taken advantage of longer term vector dependencies in time. Certainly, any further dependencies we can uncover should allow us to transmit the data at even lower rates for a given average distortion. Speech, of course, is a time-varying process that is rich in time-related dependencies. The statistical structures of the spectral sequences for each sound and of the sequences of sounds in each utterance are fertile sources of dependency. In this section, we survey briefly several methods in which time dependencies (beyond a single frame) are exploited to reduce the bit rate. We begin first with frame transmission schemes that select the frames to be transmitted judiciously, followed by time-dependent VQ methods. Most of the methods described below incorporate additional delays.

A. Selective Frame Transmission

Perhaps the simplest, and yet effective, method for selective frame transmission is known as *frame repeat* or *frame fill* [93], [101], [102]. In this method, only every other frame is transmitted; for the intervening frames, a 1-bit code is transmitted which specifies whether the missing frame should be replaced by the previous frame or the following frame. Which frame to repeat is determined by a nearest neighbor rule. A variation on this method is to send 1 or 2 bits which specify values for the missing frame which are a linear interpolation of the transmitted frames [119].

Another simple and very effective method for selective frame transmission is *variable-frame-rate* transmission [100], [119], [132]. In this method, information is transmitted only when the properties of the speech signal have changed sufficiently (in some predetermined sense) since the preceding transmission. The parameters of the untransmitted frames are regenerated at the receiver through linear interpolation between the parameters of the two adjacent transmitted frames. Thus parameter transmissions occur more frequently when speech characteristics are changing rapidly, as in transitions between sounds, and less frequently when speech characteristics are relatively constant, as in steady-state sounds.

B. Segment Quantization

Variable-frame-rate VQ of spectral parameters is a useful procedure for coding at rates as low as 400 bits/s. For data rates of 300 bits/s or lower, the linear interpolation assumption breaks down and the method is no longer adequate in maintaining reasonable intelligibility. At these lower rates, the method of choice is to perform VQ on a sequence of frames comprising a larger segment of speech. The method is known as *segment quantization* [111], [137], in contrast with frame quantization where each frame is quantized separately. VQ is now employed over a larger vector $\boldsymbol{X} = [\boldsymbol{x}(1)^T \boldsymbol{x}(2)^T \cdots \boldsymbol{x}(J)^T]^T$ comprising the elements of J consecutive frame vectors. In this manner, the dependence between consecutive frames is included implicitly in the larger vectors.[14] The duration J of the segment can be either fixed or variable. Clearly, a fixed J results in a simpler quantization problem. However, utilizing variable-duration segments has led to better speech quality.

[14]Segment quantization has also been termed *matrix quantization* [137]. We prefer to use the former term because matrix quantization could be taken to imply another method beyond scalar and vector quantization, while in reality it is simply VQ applied over a longer vector that comprises a larger segment of speech.

Segment quantization is a good example where a random codebook has been especially useful. Because of the relatively high-dimensional space ($10 \times 14 = 140$ if $J = 10$), one can argue that a random codebook should give good results. The overriding reason for using a random codebook, however, comes from perceptual considerations. We have found that the clustering and centroid computations in the K-means algorithm produce code vectors (segments) that sound muffled upon listening. So, even if a random codebook produces a larger mse distortion than the corresponding K-means codebook, the synthesized speech from the random codebook yields higher quality and intelligibility. This is yet another instance where an objective measure of distortion is not a good measure of perceptual distance. In the experiments performed thus far, a 13-bit codebook is used ($L = 8192$ segments). Because a random codebook is not structured, quantization of the speech input requires a substantial amount of computation.

Segment quantization is truly in the spirit of VQ theory. For, in principle, one should be able to approach the performance of the optimal coding system by simply using longer blocks for quantization. Unfortunately, if signal fidelity is to be maintained, exponentially larger codebooks will need to be used, which very quickly becomes impractical. An alternative solution is to include time dependence explicitly in the modeling (as opposed to implicitly, as in segment quantization). In this manner, the complexity of the vector quantizer can remain manageable. We have already described two such methods above, frame repeat and variable-frame-rate transmission. Below, we describe other models that adapt to the changing properties of the signal.

C. Adaptive VQ

Adaptive VQ implies a change in the codebook as a function of time. There are two types of adaptive vector quantizers: forward-adaptive and backward-adaptive. In forward-adaptive schemes, the system examines *future* data and decides whether a change in the codebook is necessary or not, and transmits to the receiver the desired changes using additional bits. Typically, such schemes incorporate some delay to allow for the inspection of upcoming data. Backward-adaptive schemes examine only *past* transmitted data and change the codebook accordingly. Since past information is available at both the transmitter and the receiver, the necessary codebook updating computation can be performed at both ends simultaneously, thus obviating the need for transmitting additional bits. Backward-adaptive systems have the added advantage of having no extra delay, but are quite susceptible to degraded performance under channel errors. If in forward-adaptive systems the additional bits are not counted, then such systems will result in lower quantization distortion than backward-adaptive systems. However, for an overall fixed data rate, which of the two types of systems performs better depends on the relative number of additional bits to the overall data rate. The principles of forward and backward adaptation are similar to those used in scalar quantization [74].

We should mention at this point that VQ, by its very nature, implies some delay to allow for the input vectors to be specified. Even a backward-adaptive vector quantizer will have this inherent delay. A forward-adaptive vector quantizer may have additional delays above and beyond the delays inherent in specifying the vectors in the first place.

An example of a forward-adaptive system is Paul's adaptive VQ system [100] described above. Most adaptive VQ schemes to date, however, are backward-adaptive and most such schemes are *feedback* VQ systems [56]. Two major classes of feedback VQ systems are *vector predictive* systems and *finite-state VQ* systems. Vector predictive systems [25] are a generalization of scalar predictive systems to the vector case and, thus far, they have been used for medium-rate coding of speech waveforms.

In finite-state VQ, we have a finite-state network with transitions among states and we associate with each state a codebook. If one is in a particular state, then the codebook associated with that state is used to quantize the input vector at that time. Depending on which code vector was used, a transition is made to another state (which could also be the same state). The codebook at that state is now used to quantize the next input vector, and so on. The union of all codebooks from all states is usually very large. By subsetting the larger codebook into smaller codebooks at each state, a smaller number of bits is used to quantize each input vector, so that the average bit rate is reduced from the case where the large codebook is used for every input vector. If the time-dependent structure of the data is extensive, substantial reduction in bit rate without loss in fidelity can be accomplished. Finite-state VQ schemes have been employed with segment quantization [113], frame quantization [32], [110], and waveform coding [42], [62].

VII. Speech Waveform Coding

Our understanding of the vector quantization of speech *spectral parameters* has advanced and its usefulness and practicability have been demonstrated to the point where real-time hardware implementation of very-low-rate speech coders is becoming a reality. The application of VQ to speech *waveform* coding is a more recent but intense research activity, and lends itself well to the development of a large number of techniques and variations. We believe it will be some time before the relative merits of the various techniques are worked out and fully appreciated. While some waveform VQ techniques are becoming practical now, especially for coding at about 16 kbits/s, other techniques for low-rate coding require significant computational resources that are not widely available. With additional research effort, the role of waveform VQ in speech coding will become clearer and the possibilities of achieving toll quality at low data rates (below 8 kbits/s) may become a reality.

Below, we give the reader a view of the difficulties inherent in waveform VQ and present a framework that we hope will help sharpen our understanding of the waveform VQ process. It is instructive first to take a brief look at the state of the art in the scalar quantization of speech waveforms, especially those methods that will have a bearing on our VQ discussion below.

A. *Scalar Waveform Quantization*

It is generally accepted that at 16 kbits/s, adaptive predictive coding (APC) and adaptive transform coding (ATC) result in speech that has approximately toll quality [74] (i.e., quality equal to a high-grade telephone line). In a particular configuration for both methods, one computes and transmits: 1) a representation of the short-term spectrum (typically in the form of LPC coefficients); 2) the short-term gain; 3) the pitch and parameters of a pitch filter; and 4) the prediction residual waveform.[15] The pitch filter is usually an all-pole filter with a maximum of three parameters; it is of the form $1/C(z)$, where

$$C(z) = 1 + c(\tau - 1)z^{-(\tau-1)} + c(\tau)z^{-\tau} + c(\tau + 1)z^{-(\tau+1)} \tag{123}$$

and τ is the estimated pitch period in samples. Pitch periods for adults vary from less than 3 ms for high-pitched females to over 12 ms for low-pitched males. (In certain system implementations, the inclusion of a pitch filter is found to be unnecessary or undesirable; see, for example [89], [133].) Parameter sets 1–3 above are computed and transmitted every frame as *side information* with the residual. At the receiver, the residual excites a filter of the form

$$H(z) = \frac{G}{A(z)\,C(z)} \tag{124}$$

to result in the output speech. The speech signal is usually sampled at either 6.67 or 8 kHz. At 6.67-kHz sampling, the residual is coded with 2 bits/sample, which leaves about 2.7 kbits/s to transmit the side information. At 8-kHz sampling, a 3-level quantizer is used to code the residual (or about 1.6 bits/sample), which leaves over 3 kbits/s to code the side information.

At 16 kbits/s, the short-term SNR achievable by APC and ATC is approximately equal to the gains predicted by rate-distortion theory. The SNR gain over simple quantization of the speech waveform itself is approximately equal to $-10 \log_{10} \gamma$, where γ is the ratio of the geometric mean to the arithmetic mean of the short-term speech spectrum as represented by $H(z)$ (see (76)). The short-term SNR, therefore, changes as the spectrum changes and, in speech, a typical range is 5–20 dB, with higher SNR values associated with vowel sounds and lower values with unvoiced sounds.

As the number of bits used to quantize the residual is reduced to about 1 bit/sample or less, speech quality deteriorates rapidly. A number of methods have been employed to improve the quality at transmission rates in the range 8–9.6 kbits/s. One technique known as *multipulse coding* [9] models the residual as a sequence of a relatively small number of pulses (smaller than the number of residual samples) whose amplitudes and time locations are optimized and transmitted each frame. Another class of coders, with a long history in speech coding, is known as *baseband-excited coders* [89] or *voice-excited coders* [38]. This is a frequency-selective method where only a low-pass band (known as the baseband) of the residual, for example, is transmitted. At the receiver, the high frequencies are obtained from the low frequencies by nonlinear high-frequency regeneration techniques. Though quite effective, these methods also break down as the bit rate goes much

[15]Most ATC systems operate directly on the speech waveform and not on the prediction residual [74]. However, if done appropriately, ATC on the prediction residual can produce similar results [16].

below 1 bit/sample. Vector quantization promises to carry speech coding with high quality into these lower rates.

B. Vector Waveform Quantization

Direct VQ of the speech waveform can be accomplished by simply dividing the waveform into consecutive blocks of N samples each and designing a codebook whose vectors are obtained by one of the clustering methods described earlier. At high data rates, such a method would give good results at relatively small values of N. However, as the bit rate is decreased, it becomes more and more important to increase the block length to minimize the distortion. As N increases, two problems arise. The first problem is the one we already know, and that is the exponential increase of needed resources. This problem can be dealt with partly by proper structuring of the codebook, as we shall see below. The second problem is less obvious; it concerns the concatenation of waveforms for synthesis. As we quantize each block independently to its closest code vector, we minimize the distortion over the whole block so that, for a long block, there is no guarantee that the end of one code vector will match the beginning of the next code vector in time. The result will be a discontinuity in the speech waveform, which may be heard as roughness in the speech. The problem is aggravated at low data rates since the choices for appropriate code vectors are relatively few. One solution to the concatenation problem is to quantize blocks that overlap by a small amount. Then, the quantized code vectors are overlapped and added with a decreasing weighting during the overlap region which goes to zero in both directions, so that the transition from one quantized block to the next is made smoother. This method is used effectively in ATC of the speech waveform to prevent waveform discontinuities from one block to the next [130]. Although the weighted-overlap method just mentioned would be expected to work well, it has the drawback that the overlap actually wastes bits since the bits for the overlapped regions are in effect transmitted twice. Another solution to the waveform discontinuity problem is to quantize the residual instead of the speech waveform itself, as will be described below. ATC of the residual has been used effectively in eliminating the need for overlap [16].

Much of the recent activity in applying VQ to the coding of speech waveforms has concentrated on the medium-rate range of 8–16 kbits/s (see, for example, [1], [24], [25], [34], [37], [42], [49], [50], [56], [63], [83], [109], [124]). In that range, scalar quantization methods already produce approximately toll quality speech at 16 kbits/s and communications quality [39] at 8–9.6 kbits/s. Of course, with the use of VQ, we have the potential of moving the toll quality boundary further down. An exposition of the various waveform VQ techniques under development is beyond the intended scope of this paper. Instead, we have chosen in this brief section to discuss a particular approach to speech coding at low rates (below 8 kbits/s), for it is at those rates that scalar methods tend to break down and VQ offers the potential of achieving toll quality. This approach was chosen because it forms a natural extension to existing well understood scalar speech coding techniques, and appears to have a potential for high-quality low-rate coding of speech.

The main problem in speech waveform coding is how to structure and design our codebook such that signal fidelity is maintained but computational and storage costs are reduced to manageable proportions. One relatively natural manner to structure the codebook is to factor out the short-term spectral and pitch dependencies and code them separately, in effect use a product code. By removing the short-term linear dependencies, one is left with a residual that is relatively white. Fig. 26 shows three waveforms: plot (a) is the speech waveform; plot (b) is the prediction residual after filtering the speech waveform with the spectral inverse filter $A(z)$; and plot (c) is the residual after filtering the speech waveform with the combined spectral and pitch inverse filter $A(z)C(z)$. (Note that in Fig. 26, waveform (b) was amplified 10 dB and waveform (c) 20 dB relative to the speech waveform.) The residual waveform (c) has most of its short-term linear dependencies removed; it certainly looks much more random than (a) or (b). The statistics of waveform (c) tend to be Gaussian [7]. Our codebook can now be structured in four parts corresponding to the spectral filter $A(z)$, the pitch filter $C(z)$, the gain G, and the residual (shown as waveform (c) in Fig. 26). The first three sets of parameters can be coded and transmitted as side information, exactly as in APC, except that here we can perform VQ on each set of parameters [2], [24] to reduce the bit rate. Also as in APC, the residual is *not quantized separately*, but rather as part of a synthesis procedure. If performed separately, time-domain quantization of the residual will result in no SNR gain over PCM coding of the waveform.

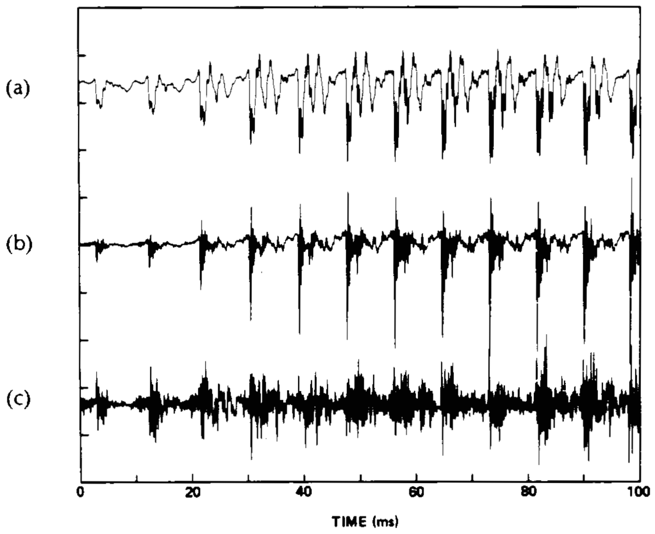

Fig. 26. (a) A speech waveform. (b) The prediction residual after filtering the speech with the all-zero filter $A(z)$. (The plot is amplified 10 dB relative to (a).) (c) The residual after filtering the speech with the combined filter $A(z)C(z)$, thus removing much of the pitch-related structure. (The plot is amplified 20 dB relative to (a).) (From Atal [7].)

There is a compelling reason why the product code suggested above would be expected to perform well. We mentioned in Section V-C that a product code would be expected to yield good results if the component vectors are chosen to be independent. One could argue with some justification that the short-term spectral envelope, the pitch, and the whitened residual are to some extent independent,

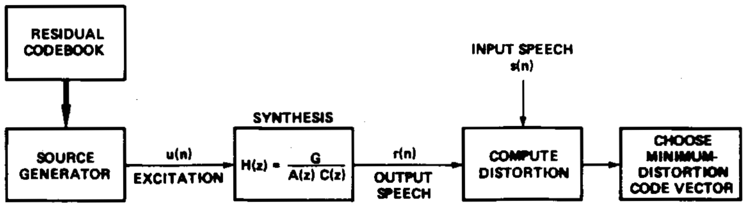

Fig. 27. A block diagram of a system for low-rate speech waveform coding which employs vector quantization. The spectral and pitch filter parameters are quantized and transmitted separately. The codebook shown is used to find the excitation that results in the minimum distortion. As shown in the figure, the excitation is white. If desired, $C(z)$ can be eliminated, in which case the excitation should have some pitch structure, as in the residual waveform (b) in Fig. 26.

so that the use of a product code in this case should reduce computation and storage at only a relatively small reduction in performance over a single large codebook obtained by K-means.

Fig. 27 shows a basic block diagram of the coding system. The first step is to compute and quantize the parameters of $A(z)$, $C(z)$, and G, then substitute the *quantized* values in the synthesis filter $H(z)$ in Fig. 27. Then, the source generator takes one code vector at a time from the residual codebook and sends the samples from the code vector sequentially in time as excitation to the filter $H(z)$. The distortion between the output speech and the input speech in the block to be coded is computed. The process is repeated for all code vectors in the residual codebook, each time comparing the output to the same block of input speech. The code vector that results in the minimum distortion is chosen for transmission. The whole process is then repeated for the next input block. Note that nowhere is there an explicit quantization of the actual prediction residual; the quantization takes place implicitly by searching all vectors in the residual codebook for the vector that minimizes the distortion in the output speech. (The distortion computation may benefit from perceptually based operations such as the use of spectral noise shaping [11], [88].) In product code terminology, the above is a sequential quantization procedure, whereby the side information is quantized first, then the quantized parameter values are used in a full-search of the residual codebook such that an overall distortion criterion is minimized.

The only remaining issues are to choose the block length N and the bit rate R, and to decide how to populate the residual codebook for a given N and R. There is evidence from experiments by Atal [7] that block lengths of about at least 4 ms are needed for good results. This is equivalent to $N = 32$ samples at 8-kHz sampling. At $R = 1$ bit/sample, the codebook contains 32×2^{32} residual samples or about one-half year of speech! At $R = 0.5$ bits/sample, the equivalent amount of speech is just over 4 min. However, the size of the codebook is still $L = 2^{16} = 65\,536$ vectors, which means that the process in Fig. 27 will need to be repeated that many times for each block of 32 samples, if a full search is desired. These illustrative numbers place in perspective the magnitude of the computational problem in VQ. To render the computations more manageable, one will need to either go to lower rates, use shorter blocks, or structure the residual codebook in some fashion.

Because the number of dimensions N here is relatively large, a random codebook would be expected to do well. This is especially so because the residual is relatively white. Therefore, one could populate the codebook from a random sample of residual vectors. Since the residual is relatively Gaussian and white, another possibility would be to simply use a random Gaussian number generator to populate the codebook. Both methods have been used by Atal and Schroeder with good results. In recent simulations on a Cray-1 computer, with $N = 40$ and $R = 0.25$ bits/sample for the residual (excitation), good speech quality was reported with no quantization of the side information [117]. It is important to note that even if the excitation consists of random numbers, each code vector will have a different detailed spectrum and time-domain structure. The quantization process chooses the particular code vector that minimizes the distortion between the original and reconstructed signals.

The use of a pitch filter $1/C(z)$ may not always be desirable. If a pitch filter is not used, then the residual codebook will need to contain wave shapes similar to waveform (b) in Fig. 26. Because of the need for the "pitch pulses," one can no longer use a random Gaussian source to populate the residual codebook. A codebook will have to be designed from a set of residual training data. One could, of course, use any of the other methods we mentioned in Section V. No one, to our knowledge, has attempted to use a more optimized residual codebook for large block lengths.

Tree and trellis coding: Another approach to structuring the codebook to reduce computations and storage is to have code vectors share some of their elements, i.e., some of the elements will have the same values. The two most well-known methods in this class are *tree coding* and *trellis coding* [74], [75], [135]. In tree coding, the vector elements are on a tree, while in trellis coding, the elements are on a trellis. Both types of coding permit the use of longer blocks because of the reduced computational requirements. However, the computational and memory savings are obtained at the cost of reduction in performance, with trellis coding achieving the greatest computational savings but at reduced performance relative to tree coding and to VQ. A number of studies have utilized tree and trellis coding of speech waveforms [6], [7], [34], [49], [54], [56], [74], [128], [136], but there has not been a systematic study comparing these two methods against VQ at lower data rates.

VIII. Conclusions

In the coding of signals at a given rate, it is always possible to reduce the distortion further by using vector quantization instead of scalar quantization. However, because of the substantial computational and memory costs associated with vector quantization, it is most advantageous to use it at low data rates of less than 1 bit/parameter or signal value to be transmitted. In addition, vector quantizers, on the whole, may not be as robust as scalar quantizers when used in an operational environment, especially under conditions of channel errors. However, at low data rates, vector quantization may still be the method of choice when all factors are considered.

Speech presents a fertile signal for the application of vector quantization. Speech parameters are rich with linear and nonlinear dependencies that are utilized effectively by vector quantizers to optimize performance. Much of the success in applying vector quantization to speech coding has been in the coding of spectral parameters for low rate coding of intelligible speech below 800 bits/s. Recent results in the vector quantization of speech waveforms point to exciting possibilities for high-quality speech coding at 8 kbits/s and less.

Acknowledgment

The authors wish to thank the following friends and colleagues who have read and commented on an earlier version of this paper: B. Atal, M. Ostendorf Dunham, A. Gersho, N. Jayant, B-H. Juang, R. Mercer, P. Noll, D. Paul, and Y. Shoham. Their comments have been very valuable in enhancing the accuracy of the presentation. The authors also thank L. Willson for her assistance in the preparation of the manuscript.

References

[1] H. Abut and S. A. Luse, "Vector quantizers for subband coded waveforms," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (San Diego, CA, Mar. 1984), paper 10.6.

[2] J-P. Adoul, J-L. Debray, and D. Dalle, "Spectral distance measure applied to the optimum design of DPCM coders with L predictors," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Denver, CO, Apr. 1980), pp. 512–515.

[3] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.

[4] V. R. Algazi, "Useful approximations to optimum quantization," *IEEE Trans. Commun. Technol.*, vol. COM-14, pp. 297–301, 1966.

[5] M. R. Anderberg, *Cluster Analysis for Applications.* New York, NY: Academic Press, 1973.

[6] J. B. Anderson and J. B. Bodie, "Tree encoding of speech," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 379–381, July 1975.

[7] B. S. Atal, "Predictive coding of speech at low bit rates," *IEEE Trans. Commun.*, vol. COM-30, no. 4, pp. 600–614, Apr. 1982.

[8] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *J. Acoust. Soc. Amer.*, vol. 50, no. 2, pp. 637–655, Aug. 1971.

[9] B. S. Atal and J. R. Remde, "A new model of LPC excitation for producing natural-sounding speech at low bit rates," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Paris, France, May 1982), pp. 614–617.

[10] B. S. Atal and M. R. Schroeder, "Adaptive predictive coding of speech signals," *Bell Syst. Tech. J.*, vol. 49, pp. 1973–1986, Oct. 1970.

[11] _____, "Predictive coding a speech signals and subjective error criteria," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, no. 3, pp. 247–254, June 1979.

[12] T. P. Barnwell, III, "Correlation analysis of subjective and objective measures for speech quality," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Denver, CO, Apr. 1980), pp. 706–709.

[13] R. E. Bellman, *Introduction to Matrix Analysis.* New York, NY: McGraw-Hill, 1960.

[14] T. Berger, *Rate-Distortion Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

[15] T. Berger, "Minimum entropy quantizers and permutation codes," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 2, pp. 149–157, Mar. 1982.

[16] M. Berouti and J. Makhoul, "An adaptive-transform baseband coder," in *Speech Communication Papers: 97th Meeting of the Acoustical Society of America*, J. J. Wolf and D. H. Klatt, Eds. (Cambridge, MA, June 1979), pp. 377–380.

[17] R. E. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 460–473, July 1972.

[18] A. Buzo, A. H. Gray Jr., R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, no. 5, pp. 562–574, Oct. 1980.

[19] J. W. S. Cassels, *An Introduction to the Geometry of Numbers.* Berlin, Germany: Springer-Verlag, 1959.

[20] D. Y. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham, "Fast search algorithms for vector quantization and pattern matching," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (San Diego, CA, Mar. 1984), paper 9.11.

[21] 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, no. 2, pp. 211–226, Mar. 1982.

[22] _____, "Fast quantizing and decoding algorithms for lattice quantizers and codes," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 2, pp. 227–232, Mar. 1982.

[23] _____, "A lower bound on the average error of vector quantizers," *IEEE Trans. Inform. Theory*, vol. IT-31, no. 1, pp. 106–109, Jan. 1985.

[24] M. Copperi and D. Sereno, "9.6 kbit/s piecewise LPC residual excited coder using multiple-stage vector quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (San Diego, CA, Mar. 1984), paper 10.5.

[25] V. Cuperman and A. Gersho, "Vector predictive coding of speech at 16 kbits/s," *IEEE Trans. Commun.*, vol. COM-33, pp. 685–696, July 1985.

[26] L. D. Davisson and R. M. Gray, Eds., *Data Compression.* Stroudsburg, PA: Dowden Hutchinson & Ross, 1976.

[27] N. R. Dixon and T. B. Martin, Eds., *Automatic Speech and Speaker Recognition.* New York, NY: IEEE PRESS, 1979.

[28] J. L. Doob, *Stochastic Processes.* New York, NY: Wiley, 1953.

[29] E. Dubois, "The sampling and reconstruction of time-varying imagery with application in video systems," *Proc. IEEE*, vol. 73, no. 4, pp. 502–522, Apr. 1985.

[30] R. O. Duda and R. E. Hart, *Pattern Classification and Scene Analysis.* New York, NY: Wiley, 1973.

[31] H. Dudley, "Phonetic pattern recognition vocoder for narrow-band speech transmission," *J. Acoust. Soc. Amer.*, vol. 30, no. 8, pp. 733–739, Aug. 1958.

[32] M. O. Dunham and R. M. Gray, "An algorithm for design of labeled-transition finite-state vector quantizers," *IEEE Trans. Commun.*, vol. COM-33, no. 1, pp. 83–89, Jan. 1985.

[33] N. Farvardin and J. W. Modestino, "Optimum quantizer performance for a class of non-Gaussian memoryless sources," *IEEE Trans. Inform. Theory*, vol. IT-30, no. 3, pp. 485–497, May 1984.

[34] H. G. Fehn and P. Noll, "Multipath search coding of stationary signals with applications to speech," *IEEE Trans. Commun.*, vol. COM-30, no. 4, pp. 687–701, Apr. 1982.

[35] L. Fejes Toth, "Sur la representation d'une population infinie par un nombre fini d'elements," *Acta Math. Acad. Scient. Hung.*, vol. 10, pp. 299–304, 1959.

[36] T. R. Fischer and R. M. Dicharry, "Vector quantizer design for memoryless Gaussian, Gamma, and Laplacian sources," *IEEE Trans. Commun.*, vol. COM-32, no. 9, pp. 1065–1069, Sept. 1984.

[37] T. R. Fischer and K. T. Malone, "Contour vector quantization and waveform coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Tampa, FL, Mar. 1985), pp. 1707–1710.

[38] J. L. Flanagan, *Speech Analysis Synthesis and Perception*, 2nd ed. New York: Academic Press, 1972.

[39] J. L. Flanagan, M. R. Schroeder, B. S. Atal, R. E. Crochiere, N. S. Jayant, and J. M. Tribolet, "Speech coding," *IEEE Trans. Commun.*, vol. COM-27, no. 4, pp. 710–737, Apr. 1979.

[40] P. E. Fleischer, "Sufficient conditions for achieving minimum distortion in a quantizer," in *1964 IEEE Int. Conv. Rec.*, pt. 1, pp. 104–111, 1964.

[41] E. W. Forgy, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications," *Biometrics*, vol. 21, p. 768, abstract, 1965.

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

[43] B. Fox, "Discrete optimization via marginal analysis," *Manag. Sci.*, vol. 13, no. 3, pp. 210–216, Nov. 1966.

[44] J. H. Friedman, F. Bashett, and L. J. Shustek, "An algorithm for finding nearest neighbors," *IEEE Trans. Comput.*, vol. C-24, pp. 1000–1006, Oct. 1975.

[45] J. W. Fussell, "The Karhunen-Loéve transform applied to the log area ratios of a linear predictive speech coder," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Denver, CO, Apr. 1980), pp. 36–39.

[46] R. G. Gallager, *Information Theory and Reliable Communication.* New York, NY: Wiley, 1968.

[47] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. Inform. Theory*, vol. IT-25, no. 4, pp. 373–380, July 1979.

[48] _____, "On the structure of vector quantizers," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 157–166, Mar. 1982.

[49] A. Gersho and V. Cuperman, "Vector quantization: A pattern-matching technique for speech coding," *IEEE Commun. Mag.*, vol. 21, pp. 15–21, Dec. 1983.

[50] A. Gersho, T. Ramstad, and I. Versvik, "Fully vector-quantized subband coding with adaptive codebook allocation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (San Diego, CA, Mar. 1984), paper 10.7.

[51] H. Gish and J. N. Pierce, "Asymptotically efficient quantizing," *IEEE Trans. Inform. Theory*, vol. IT-14, no. 5, pp. 676–683, Sept. 1968.

[52] B. Gold, "Digital speech networks," *Proc. IEEE*, vol. 65, no. 12, pp. 1636–1658, Dec. 1977.

[53] _____, "Experiments with a pattern-matching channel vocoder," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Atlanta, GA, Apr. 1981), pp. 32–35.

[54] A. J. Goldberg, "Predictive coding with delayed decision," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Hartford, CT, May 1977), pp. 405–408.

[55] W. Granzow and P. Noll, "Quantization of a memoryless gamma source," submitted to *IEEE Trans. Commun.*

[56] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, pp. 4–29, Apr. 1984.

[57] R. M. Gray, A. Buzo, A. H. Gray, Jr., and Y. Matsuyama, "Distortion measures for speech processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, no. 4, pp. 367–376, Aug. 1980.

[58] R. M. Gray and E. D. Karnin, "Multiple local optima in vector quantizers," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 2, pp. 256–261, Mar. 1982.

[59] R. M. Gray and J. C. Kieffer, "Asymptotically mean stationary measures," *Ann. Probability*, vol. 8, pp. 962–973, Oct. 1980.

[60] R. M. Gray and Y. Linde, "Vector quantizers and predictive quantizers for Gauss-Markov sources," *IEEE Trans. Commun.*, vol. COM-30, no. 2, pp. 381–389, Feb. 1982.

[61] A. H. Gray, Jr. and J. D. Markel, "Distance measures for speech processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, no. 5, pp. 380–391, Oct. 1976.

[62] A. Haoui and D. G. Messerschmitt, "Predictive vector quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (San Diego, CA, Mar. 1984), paper 10.10.

[63] _____, "Embedded coding of speech: A vector quantization approach," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Tampa, FL, Mar. 1985), pp. 1703–1706.

[64] J. A. Hartigan, *Clustering Algorithms.* New York: Wiley, 1975.

[65] J. Huang and P. Schultheiss, "Block quantization of correlated Gaussian random variables," *IEEE Trans. Commun. Syst.*, vol. CS-11, pp. 289–296, Sept. 1963.

[66] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sept. 1952.

[67] IMSL Library, International Mathematical and Statistical Libraries, Inc., Houston, TX.

[68] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, no. 1, pp. 67–72, Feb. 1975.

[69] F. Itakura, S. Saito, T. Koike, H. Sawabe, and M. Nishikawa, "An audio response unit based on partial autocorrelation," *IEEE Trans. Commun.*, vol. COM-20, pp. 792–797, Aug. 1972.

[70] F. Itakura and S. Saito, "Analysis synthesis telephony based on the maximum likelihood method," in *Proc. 6th Int. Congr. Acoust.* (Tokyo, Japan, 1968), pp. C17–C20.

[71] _____, "A statistical method for estimation of speech spectral density and formant frequencies," *Electron. Commun. Japan*, vol. 53-A, pp. 36–43, 1970.

[72] _____, "On the optimum quantization of feature parameters in the PARCOR speech synthesizer," in *Conf. Rec., IEEE Conf. Speech Communication and Processing* (Newton, MA, Apr. 1972), pp. 434–437.

[73] N. S. Jayant, Ed., *Waveform Quantization and Coding.* New York, NY: IEEE PRESS, 1976.

[74] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Englewood Cliffs, NJ: Prentice-Hall, 1984.

[75] F. Jelinek, "Tree encoding of memoryless time-discrete sources with a fidelity criterion," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 584–590, Sept. 1969.

[76] B-H. Juang and A. H. Gray, Jr., "Multiple stage vector quantization for speech coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Paris, France, May 1982), pp. 597–600.

[77] G. S. Kang and D. C. Coulter, "600 bps voice digitizer," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Philadelphia, PA, Apr. 1976), pp. 91–94.

[78] N. Kitawaki and F. Itakura, "Nonlinear coding of PARCOR coefficients," in *Meeting of the Acoustical Society of Japan* (in Japanese), pp. 449–450, Oct. 1973.

[79] H. P. Kramer and M. V. Mathews, "A linear encoding for transmitting a set of correlated signals," *IRE Trans. Inform. Theory*, vol. IT-2, pp. 41–46, Sept. 1956.

[80] S. E. Levinson, "Structural methods in automatic speech recognition," this issue, pp. 1625–1650.

[81] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, no. 1, pp. 84–95, Jan. 1980.

[82] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[83] Ph. Mabilleau and J-P. Adoul, "Medium band speech coding using a dictionary of waveforms," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Atlanta, GA, Mar. 1981), pp. 804–807.

[84] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. on Math., Statist., and Prob.* Berkeley, CA: Univ. of California Press, 1967, pp. 281–297.

[85] P. C. Mahalanobis, "On the generalized distance in statistics," *Proc. Indian Nat. Inst. Sci.* (Calcutta), vol. 2, pp. 49–55, 1936.

[86] J. Makhoul, "Linear prediction: A tutorial review," *Proc. IEEE*, vol. 63, no. 4, pp. 561–580, Apr. 1975.

[87] _____, "Speech coding and processing," in *Modern Signal Processing*, T. Kailath, Ed. New York, NY: Hemisphere/Springer-Verlag, 1985, pp. 211–248.

[88] J. Makhoul and M. Berouti, "Adaptive noise spectral shaping and entropy coding in predictive coding of speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, no. 1, pp. 63–73, Feb. 1979.

[89] _____, "Predictive and residual encoding of speech," *J. Acoust. Soc. Amer.*, vol. 66, no. 6, pp. 1633–1641, Dec. 1979.

[90] J. D. Markel and A. H. Gray, Jr., *Linear Prediction of Speech.* New York, NY: Springer-Verlag, 1976.

[91] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Theory*, vol. IT-6, no. 2, pp. 7–12, Mar. 1960.

[92] R. J. McEliece, *The Theory of Information and Coding: A Mathematical Framework for Communication.* Reading, MA: Addison-Wesley, 1977.

[93] E. McLarnon, "A method for reducing the transmission rate of a channel vocoder by using frame interpolation," in *Proc.*

IEEE Int. Conf. Acoust., Speech, Signal Processing (Tulsa, OK, Apr. 1978), pp. 458–461.

[94] P. Mermelstein, "Evaluation of a segmental SNR measure as an indicator of the quality of ADPCM coded speech," *J. Acoust. Soc. Amer.*, vol. 66, no. 6, pp. 1664–1667, Dec. 1979.

[95] A. Nadas, R. L. Mercer, L. R. Bahl, R. Bakis, P. S. Cohen, A. G. Cole, F. Jelinek, and B. L. Lewis, "Continuous speech recognition with automatically selected acoustic prototypes obtained by either bootstrapping or clustering," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Atlanta, GA, Apr. 1981), pp. 1153–1155.

[96] P. Noll, "Adaptive quantizing in speech coding systems," in *Proc. Int. Zurich Seminar on Digital Commun.* (Zurich, Switzerland, Mar. 1974), paper B3.

[97] P. Noll and R. Zelinski, "Bounds on quantizer performance in the low bit-rate region," *IEEE Trans. Commun.*, vol. COM-26, no. 2, pp. 300–304, Feb. 1978.

[98] J. J. O'Donnell, "A system for very low data rate speech communication," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Atlanta, GA, Apr. 1981), pp. 8–11.

[99] E. A. Patrick, *Fundamentals of Pattern Recognition.* Englewood Cliffs, NJ: Prentice-Hall, 1972.

[100] D. B. Paul, "An 800 bps adaptive vector quantization vocoder using a perceptual distance measure," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Boston, MA, Apr. 1983), pp. 73–76.

[101] ____, "The Lincoln low-rate vocoder: A 1200/2400 bps LPC-10 voice terminal," Tech. Rep. 676, Lincoln Lab., Lexington, MA, Mar. 1984, DCC AD-A141291/5.

[102] D. B. Paul and P. E. Blankenship, "Two distance measure-based vocoder quantization algorithms for very-low-data rate applications: Frame-fill and spectral vector quantization," in *Proc. IEEE Int. Conf. Commun.* (June 1982), pp. 1–6.

[103] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed. Cambridge, MA: M.I.T. Press, 1972.

[104] R. Pieraccini and R. Billi, "Experimental comparison among data compression techniques in isolated word recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Boston, MA, Apr. 1983), pp. 1025–1028.

[105] S. U. H. Qureshi and G. D. Forney, Jr., "Adaptive residual coder—An experimental 9.6/16 kb/s speech digitizer," in *EASCON Rec.*, pp. 29A–29E, 1975.

[106] L. Rabiner, S. E. Levinson, and M. M. Sondhi, "On the application of vector quantization and hidden Markov models to speaker-independent isolated word recognition," *Bell Syst. Tech. J.*, vol. 62, pp. 1075–1105, Apr. 1983.

[107] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals.* Englewood Cliffs, NJ: Prentice-Hall, 1978.

[108] L. R. Rabiner, M. M. Sondhi, and S. E. Levinson, "A vector quantizer incorporating both LPC shape and energy," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (San Diego, CA, Mar. 1984), paper 17.1.

[109] H. Reininger and D. Wolf, "Speech and speaker independent codebook design in VQ coding schemes," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Tampa, FL, Mar. 1985), pp. 1700–1702.

[110] S. Roucos, J. Makhoul, and R. Schwartz, "A variable-order Markov chain for coding of speech spectra," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Paris, France, May 1982), pp. 582–585.

[111] S. Roucos, R. Schwartz, and J. Makhoul, "Segment quantization for very-low-rate speech coding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Paris, France, May 1982), pp. 1565–1569.

[112] ____, "Vector quantization for very-low-rate coding of speech," in *Proc. IEEE Global Telecommunications Conf.* (Miami, FL, Nov. 29–Dec. 2, 1982), pp. 1074–1078.

[113] ____, "A segment vocoder at 150 B/S," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Boston, MA, Apr. 1983), pp. 61–64.

[114] M. J. Sabin and R. M. Gray, "Product code vector quantizers for waveform and voice coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, no. 3, pp. 474–488, June 1984.

[115] D. J. Sakrison, "A geometric treatment of the source encoding of a Gaussian random variable," *IEEE Trans. Inform. Theory*, vol. IT-14, no. 3, pp. 96–101, May 1968.

[116] M. R. Sambur, "An efficient linear-prediction vocoder," *Bell Syst. Tech. J.*, vol. 54, pp. 1693–1723, Dec. 1975.

[117] M. R. Schroeder and B. S. Atal, "Code-excited linear prediction (CELP): High-quality speech at very low bit rates," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Tampa, FL, Mar. 1985), pp. 937–940.

[118] R. Schwartz, Y. Chow, S. Roucos, M. Krasner, and J. Makhoul, "Improved hidden Markov modeling of phonemes for continuous speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (San Diego, CA, Mar. 1984), paper 35.6.

[119] R. Schwartz and S. Roucos, "A comparison of methods for 300–400 b/s vocoders," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Boston, MA, Apr. 1983), pp. 69–72.

[120] A. Segall, "Bit allocation and encoding for vector sources," *IEEE Trans. Inform. Theory*, vol. IT-22, no. 2, pp. 162–169, Mar. 1976.

[121] I. K. Sethi, "A fast algorithm for recognizing nearest neighbors," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, no. 3, pp. 245–248, Mar. 1981.

[122] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, 1948.

[123] ____, "Coding theorems for a discrete source with a fidelity criterion," in *IRE Nat. Conv. Rec.* (pt. 4), pp. 142–163, 1959. (Also in *Information and Decision Processes*, R. E. Machol, Ed. New York, NY: McGraw-Hill, 1960, pp. 93–126.)

[124] Y. Shoham and A. Gersho, "Pitch synchronous transform coding of speech at 9.6 kb/s based on vector quantization," in *Proc. IEEE Int. Conf. Commun.* (Amsterdam, The Netherlands, May 1984), pp. 1179–1182.

[125] ____, "Efficient codebook for an arbitrary set of vector quantizers," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Tampa, FL, Mar. 1985), pp. 1696–1699.

[126] J. E. Shore, D. Burton, and J. Buck, "A generalization of isolated word recognition using vector quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Boston, MA, Apr. 1983), pp. 1021–1024.

[127] C. P. Smith, "Perception of vocoder speech processed by pattern matching," *J. Acoust. Soc. Amer.*, vol. 46, no. 6 (pt. 2), pp. 1562–1571, June 1969.

[128] L. C. Stewart, R. M. Gray, and Y. Linde, "The design of trellis waveform coders," *IEEE Trans. Commun.*, vol. COM-30, no. 4, pp. 702–710, Apr. 1982.

[129] J. T. Tou and R. C. Gonzales, *Pattern Recognition Principles.* Reading, MA: Addison-Wesley, 1974.

[130] J. M. Tribolet and R. E. Crochiere, "Frequency domain coding of speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, no. 5, pp. 512–530, Oct. 1979.

[131] R. Viswanathan and J. Makhoul, "Quantization properties of transmission parameters in linear predictive systems," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, no. 3, pp. 309–321, June 1975.

[132] V. Viswanathan, J. Makhoul, R. Schwartz, and A. W. F. Huggins, "Variable-frame-rate transmission: A review of methodology and application to narrow-band LPC speech coding," *IEEE Trans. Commun.*, vol. COM-30, no. 4, pp. 674–686, Apr. 1982.

[133] V. Viswanathan, W. Russell, A. Higgins, M. Berouti, and J. Makhoul, "Speech-quality optimization of 16 kb/s adaptive predictive coders," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Denver, CO, Apr. 1980), pp. 520–525.

[134] V. Viswanathan, W. Russell, and A. W. F. Huggins, "Objective speech quality evaluation of mediumband and narrowband real-time speech coders," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Boston, MA, Apr. 1983), pp. 543–546.

[135] 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.

[136] S. G. Wilson and S. Hussain, "Adaptive tree encoding of rion," *IEEE Trans. Commun.*, vol. COM-27, pp. 165–170, Jan. 1979.

[137] D. Y. Wong, B. H. Juang, and D. Y. Cheng, "Very low data rate speech compression with LPC vector and matrix quanti-

zation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Boston, MA, Apr. 1983), pp. 65–68.

[138] D. Y. Wong, B. H. Juang, and A. H. Gray, Jr., "An 800 bit/s vector quantization LPC vocoder," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, no. 5, pp. 770–780, Oct. 1982.

[139] Y. Yamada, S. Tazaki, and R. M. Gray, "Asymptotic performance of block quantizers with difference distortion measures," *IEEE Trans. Inform. Theory*, vol. IT-26, no. 1, pp. 6–14, Jan. 1980.

[140] T. P. Yunck, "A technique to identify nearest neighbors," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, pp. 678–683, Oct. 1976.

[141] P. L. Zador, "Asymptotic quantization error of continuous signals and the quantization dimension," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 2, pp. 139–149, Mar. 1982.

[142] R. Zelinski and P. Noll, "Adaptive transform coding of speech signals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, no. 4, pp. 299–309, Aug. 1977.

[143] J. M. Ziman, *Principles of the Theory of Solids.* Cambridge, UK: Cambridge Univ. Press, 1972.