01 Jan 1988

# New Directions in Subband Coding

Y. Shoham

Steven L. Grant
*Missouri University of Science and Technology*, sgrant@mst.edu

N. Seshadri

R. V. Cox

*et. al. For a complete list of authors, see https://scholarsmine.mst.edu/ele_comeng_facwork/1776*



## Recommended Citation

Y. Shoham et al., "New Directions in Subband Coding," *IEEE Journal on Selected Areas in Communications*, Institute of Electrical and Electronics Engineers (IEEE), Institute of Industrial Engineers (IIE), Jan 1988.
The definitive version is available at https://doi.org/10.1109/49.615

# New Directions in Subband Coding

RICHARD V. COX, SENIOR MEMBER, IEEE, STEVEN L. GAY, YAIR SHOHAM, MEMBER, IEEE, SCHUYLER R. QUACKENBUSH, MEMBER, IEEE, NAMBIRAJAN SESHADRI, AND N. S. JAYANT, FELLOW, IEEE

***Abstract*—Subband coding has been shown to provide high quality speech at rates of 16 kbit/s and above. Most previous subband coders have used scalar quantization and have been designed for fixed bit rates. In addition, most of the best of these coders have had fairly substantial delay. In this paper we describe two very different subband coders. The first is a modified Dynamic Bit Allocation Subband Coder (D-SBC) designed for variable rate coding situations, and it is easily adaptable to noisy channel environments. It can operate at rates as low as 12 kbit/s and still give good quality speech. Two key enhancements have been made to this coder. The relative importance of all bits is established as a byproduct of the dynamic bit allocation. By structuring the bitstream in a prioritized fashion, unequal bit error protection can be efficiently accomplished for those channels requiring it. In addition, identification of the relative importance of the bits in the bitstream makes it possible to make the coder embedded. The second enhancement is the use of a novel set of embedded nonlinear quantizers which allows flexibility for rate changes. The resulting coder is embedded in increments of 1 kbit/s. The second coder is a 16 kbit/s waveform coder, based on a combination of subband coding and vector quantization (VQ-SBC). The key feature of this coder is its short coding delay which makes it suitable for real-time communication networks. In the proposed coding system, gain-shape VQ, controlled by a codebook allocation algorithm, was applied to subband signals generated by a GQMF. Using this framework, a class of VQ-SBC coders was realized by varying the number of subbands which achieved a delay-performance tradeoff. The paper focuses on a 4-band VQ-SBC for which very good communication quality was achieved (segmental signal-to-noise ratio of about 20 dB), with a coding delay of about 15 ms. The speech quality of both of these coders has been enhanced by the use of adaptive postfiltering. The coders have been implemented on a single AT&T DSP32 signal processor.**

## I. Introduction

SUBBAND coding of speech is a relatively mature form of waveform coding of speech. The speech is first subdivided into a number of subbands which are then individually encoded. The underlying principle for the coder is that the bit allocation can be weighted so that those subbands with the most important information get the most bits. The advantage of subband coding may be viewed from several different angles. The most common explanation focuses on the perceptual merits of this technique. Since the human auditory mechanism responds differently to coding noises in different spectral regions, it is clearly advantageous to be able to control the spectral shape of the noise. This is achievable by coding the speech in subbands. Also, confining the coding noise that is generated in a certain region of the spectrum to that region activates the auditory masking effect which makes the noise less noticeable. Another aspect of the subband technique is more fundamental from a data compression point of view. As in transform coding, this method transforms the speech signal into a new domain in which the structure of the signal is manifested by a generally unequal energy pattern of the different bands. This energy pattern is used for efficiently controlling the allocation of the available bit resources. See, for example, [1] for a discussion of the subband coding gain. The initial subband coders used fixed bit allocations. These bit allocations were made based on the average spectrum of the speech. They tended to reflect a bias for voiced speech and gave more bits to the low frequencies than the high ones. Typical of this generation of coders was one by Crochiere [2].

In 1982 Ramstad introduced the idea of dynamically changing the bit allocation [3]. His idea was to quantize and transmit the rms value of each of the bands for a frame of speech. Based on the quantized values, the remaining bits could be allocated among the subbands in an optimal fashion. Most recently, Honda and Itakura [4] and Soong, Cox, and Jayant [5] have proposed dynamically allocating bits in both time and frequency. Their work produces very high quality speech. The complexity of these algorithms is very high, however.

Vector quantization (VQ) is a powerful coding method that has been proven to be very efficient for speech coding [6]. Vector quantization provides yet another motivation for using subband analysis–synthesis. The VQ coding gain increases with the vector dimension and becomes significant for large vector dimension. However, for medium rate coding, the computational and storage requirements are usually impractical, unless the vector dimension is small, which implies inefficient VQ. The subband approach offers a means of efficient (but suboptimal) VQ by splitting the speech signal into several signals, each with relatively low intersignal correlation, which are then vector-quantized independently. Since the subband signals are uncorrelated, the loss in performance due to independently quantizing each subband is conceptually small and this technique is achieved with a substantially reduced complexity. By viewing the subband structure and the vector quantizer as one integral scheme and by optimizing the vector quantizer for the subband signals, we were able to produce good quality coded speech.

Manuscript received May 31, 1987; revised October 23, 1987.
R. V. Cox, Y. Shoham, S. R. Quackenbush, N. Seshadri, and N. S. Jayant are with AT&T Bell Laboratories, Murray Hill, NJ 07974.
S. L. Gay is with AT&T Bell Laboratories, Holmdel, NJ 07733.
IEEE Log Number 8718685.

0733-8716/88/0200-0391$01.00 

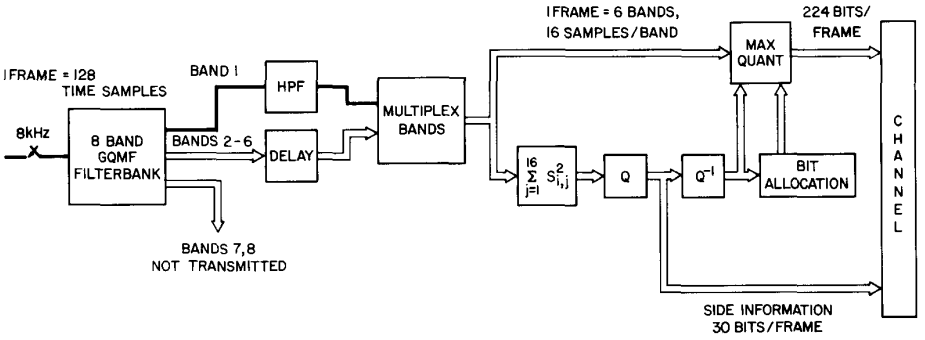



Fig. 1. Block diagram of the D-SBC transmitter.

In the work reported here, we have investigated two very different versions of subband coding. The first coder considered is based on the ideas proposed by Ramstad. The speech is divided into six 500 Hz wide subbands and into 16 ms frames. Once every frame, the rms value of the speech in each band is quantized and transmitted. This "side information" uses 2 kbit/s of the bit rate. Based on the side information, an adaptive bit allocation can be performed. Assuming the side information is correctly received, the same bit allocation can be derived from the side information at both the transmitter and receiver. Nonlinear PCM is used to quantize the subband samples. No in-band prediction is used. We shall refer to this coder as dynamic bit allocation SBC or D-SBC. The earlier coder of Crochiere [2] will be referred to as fixed bit allocation SBC or F-SBC. The side information can also be used to set the quantizer step size for each band. The main information, consisting of the quantized values of the samples of each subband, make up the majority of the information on the channel. Fig. 1 is a block diagram of the transmitter portion of this coder.

A particularly attractive feature of this coder is that the dynamic bit allocation algorithm tells us the priority of the bits used in the subband quantization. For example, the first bit allocated is more important than the second, etc. The side information bits are the most important because without these, the remainder of the encoded frame cannot be decoded. This information on the relative importance of the various data can be used to identify those bits which are in most need of error protection. This makes this coder an ideal candidate for applications involving noisy channels where a portion of the channel bandwidth is allocated for error correction. In Section III-A-2 we describe this feature more fully, and we present results on the relative sensitivities of the bits in the digital bitstream.

D-SBC has another attractive feature because of the adaptive bit allocation. It can easily be made into an embedded coder. An embedded coder has the property that a lower rate encoding can be obtained from a higher rate encoding by simply stripping the appropriate bits from the coded speech frame. Thus, the lower rate coder is embedded within the higher rate coder. For some applications, such as packet speech, it may be necessary to change bit rates dynamically according to channel conditions. An embedded coder is desirable because it allows us to encode at the highest rate and then reduce the bit rate by stripping the extra bits. This eliminates the need for decoding and then reencoding the speech at the lower rate. In the case of the D-SBC described in this paper, a 3 kbit/s coder is embedded in the 4 kbit/s coder, which in turn is embedded in the 5 kbit/s coder, etc. In order to do this, the quantizers must be embedded themselves. In Section III we describe how nonlinear embedded scalar quantizers were designed and how embedded subband coding is accomplished.

The second subband coder described in this paper is based on the combination of subband waveform coding and vector quantization. Our motivation was to exploit the advantages of both coding methods and to develop a high communication quality 16 kbit/s waveform coder with an acceptable level of complexity and a short delay. The basic approach for achieving a short delay was to use a small block size and to restrict the number of subbands (hence, the length of the filter impulse response). The second coder is a 4-band VQ-SBC which uses 2.5 ms (20 samples) data blocks. The overall coding delay of this coder is about 15 ms. However, since a larger number of (narrower) filters is advantageous from a data compression point of view, we realized the same VQ-SBC with 6 and 13 subbands and obtained higher performance at the price of longer delays. Working with a short data block created a very tight bit allocation budget which, in turn, emphasized the importance of using vector quantization. This is because VQ is an efficient way of utilizing the small number of available bits.

An additional attractive feature of both D-SBC and VQ-SBC is that because they both use a form of dynamic bit allocation, they tend to produce spectrally flat, or white, noise. Because of the side information, the relative levels of the individual bands are maintained. This makes both coders good candidates for adaptive postfiltering. If the noise were correlated with the speech, this would make the postfiltering less effective. Two different types of adaptive postfiltering were explored. The first is based on the transmitted energy of the subbands. The second is a backward adaptive method, similar to the adaptive postfiltering for ADPCM described by Ramamoorthy and Jayant [7]. In Section IV we describe both of these methods.

Both of these coders were implemented in real-time hardware based on the AT&T DSP32 signal processor. In Section V we describe the implementations of these coders. We have also had the opportunity to compare these coders to each other and to other medium bit rate speech coders. In Section VI we compare a group of medium bit rate coders on the basis of complexity, delay, and speech quality. In Section III we begin with a discussion of the possible analysis and synthesis filterbanks which were considered. Section II discusses the basic coding algorithms for each of the coders.

## II. Analysis–Synthesis Filterbanks

Subband coding algorithms can generally be considered in two parts. The first part is the filterbank structure used and the second part is the quantization of the subband signal and any side information. In this section we will discuss the filterbank structures used and the reasons for making the choices we have made.

Early subband coders, such as [2], used quadrature mirror filterbanks to divide the speech signal into subbands. The original quadrature mirror filters were based on a two-band design. To produce filterbanks with more than two bands, these filters were cascaded. As a result, filterbanks with equal sized bands could only be implemented when the number of bands was a power of two. In 1983 Rothweiler proposed a technique for designing and implementing a filterbank with an arbitrary number of bands, provided that they were all equally spaced [8]. Cox expanded on this technique [9] and called this technique generalized quadrature mirror filters (GQMF's), because this technique generalizes the two-band QMF concept to multiple bands. The design technique for GQMF filters begins with the design of a low-pass prototype. This prototype filter must have a nominal bandwidth of one-half of the desired subband bandwidth. The subband filters are obtained by modulating the low-pass filter with pairs of differently phased sinusoids. The resulting bandpass filters have the property that if a full-band signal is first passed through the entire filterbank, decimated to the Nyquist frequency in each subband, interpolated back to full-band, and resynthesized using the synthesis version of the filterbank, the resulting signal can be an arbitrarily close replica of the input signal. The phases of the sinusoids are chosen so that in the combination of the analysis–decimation–interpolation–synthesis process, the aliasing from adjacent bands will be cancelled. The frequency response of the prototype is chosen so that the frequency response of the overall system is flat to any desired accuracy.

The basic parameters of the analysis–synthesis scheme are the number of bands and their corresponding bandwidths, the flatness of the combined analysis–synthesis frequency response, and the attenuation in the stopband region of the individual filters. In general, the noise due to quantization will be on the order of 1 dB or greater. Consequently, our rule of thumb is to design filterbank systems which have a maximum passband ripple of 0.1 dB or less. With this low level of distortion, the difference between the original speech and that passed through the complete filterbank will be imperceptible. GQMF filters are approximately free of aliasing. The assumption is made that the stopband attenuation is sufficiently large that the only contributions to aliasing in any one band are made from its two nearest neighbors. For coding applications, we have found that if the stopband attenuation is greater than 40 dB, there will be no perceived aliasing. All of the filters discussed in this section meet these two criteria.

The remaining parameters which can be varied are the length of the prototype FIR filter and the number of bands. From a data compression point of view, the larger the number of bands, the greater is the potential for rate reduction, since the finer spectral resolution allows for better bit allocation. However, increasing the number of bands also increases the analysis–synthesis complexity and the coding delay. We have found that there is another key factor in choosing a filterbank structure. While the GQMF structure cancels aliasing in the absence of quantization, once quantization is performed this is no longer true. Moreover, much of the noise attributed to quantization is actually due to the uncancelled aliasing. In [10] filters with less aliasing (but more delay) were investigated and the resulting coders were reported to have less perceptible noise. The ratio of the prototype FIR filter length divided by the number of bands was found to be a good measure of the width of the transition bands which cause the aliasing. The larger the ratio, the smaller the transition bands can be made, and therefore, the smaller the amount of aliasing. Rothweiler in [8] originally recommended that this ratio be 5 or higher. As described below, the filterbanks we considered have ratios between 8 and 12.

Three different analysis-synthesis filterbanks were considered. In order of their size, the first was a 5-band 60-tap filterbank. Each band was 800 Hz wide and only the bottom 4 bands were used for speech coding. Omission of the uppermost band was not found to effect the quality of telephone bandwidth speech. This filterbank is the same one actually selected in [10] as a compromise between delay and the effects of aliasing. Because only 4 bands were used, we refer to this filterbank as the 4-band filter. The second filterbank was an 8-band 64-tap filterbank. Each band was 500 Hz wide and only the bottom 6 bands were used for speech coding. Omission of the uppermost two bands had only a slight effect on the quality of telephone bandwidth speech, primarily for female voices. While the amount of interband aliasing is increased over the first filterbank, the increased number of bands helps coding efficiency, as does the slightly smaller bandwidth (3000 versus 3200 Hz). Its delay is only $\frac{1}{2}$ ms greater. Because only 6 bands were used, we refer to this filterbank as the 6-band filter. The third filterbank was a 16-band 128-tap filterbank, obtained by interpolating the 64-tap prototype filter used for the second filterbank. Its relative amount of interband aliasing is therefore equivalent

to that of the second filterbank, but with twice the delay. We did not seriously consider using this filterbank for an implementation because of its complexity, but it made an interesting choice for comparison purposes when doing simulations. For coding purposes, the uppermost 3 bands were omitted, leaving a bandwidth of 3250 Hz. We refer to this filterbank as the 13-band filter. This paper focuses on coders using the first two filters, primarily due to delay and complexity. However, the best perceptual quality was realized using the 13-band system.

One additional detail worth noting concerns high-pass filtering. We have found that including a high-pass filter to remove dc and 60 Hz components is absolutely essential for real-time coder implementations. The GQMF filterbank provides a nice mechanism for realizing a high-pass filter. The phase of the sinusoids used to generate the lowest band filter in the filterbank can be chosen so that a null at dc results. (If the number of bands is even, the resulting filterbank will also have a null at 4000 Hz, but this is unimportant for our application.) Thus, a performance-delay tradeoff can be obtained by varying the number subbands.

## III. The Coding Algorithms

In this section we describe the details of the two subband coding algorithms. Section III-A describes D-SBC, and Section III-B describes VQ-SBC.

### A. The Dynamic Bit Allocation Subband Coder

The block diagram of the D-SBC transmitter is shown in Fig. 1. The quantization scheme is based on a frame size of 16 ms. Over the frame, speech is generally stationary, although this is not always the case. Since our quantization scheme can be described as *block adaptive*, the more stationary the speech is, the better the coder will perform. Each of the subbands produces 16 samples per frame. Thus, there are a total of 96 samples to quantize with a total of 256 bits. A side information scheme of the type proposed by Ramstad [3] was used. The energy for each of the subbands is computed and quantized. There are 6 bands, and 5 bits are used for quantizing the energy of each band, totaling 30 bits of information. (Since the remaining bits will be allocated in groups of 16, 2 bits are available for synchronization and/or signaling.) The quantizer reconstruction levels are proportional to the square root of the energies, which gives us an estimate of the standard deviation for each of the bands. This estimate is available at both the transmitter and the receiver and is the basis for the quantization of the subband signals. Quantization is essentially logarithmic over a 72 dB range.

Bit allocation is derived from the quantized energies using an iterative procedure. At each iteration, 16 bits (one per subband sample) are allocated to one of the subbands. Each iteration consists of finding the subband with the largest rms value, halving this value, storing the result in an rms table, and allocating 16 bits to that subband. There is one additional proviso—no frequency band can be allocated more than a specified maximum, typically either 4 or 5 bits per sample. When a band is allocated the maximum, its rms value is set to zero, ensuring that no more bits will be allocated to that subband. Each iteration represents 1 kbit/s of information, since each iteration represents the allocation of 1 bit per sample and each sample represents 1 ms. A nonuniform embedded quantizer optimized for a Gaussian input is used to quantize the individual subband samples. The step sizes of this quantizer are adjusted according to the quantized rms value of the band.

In comparing this algorithm to F-SBC [2], D-SBC has two main advantages. First, it has a superior filterbank structure. D-SBC has six 500 Hz bands versus two 500 Hz and two 1000 Hz bands in F-SBC. The interband aliasing of the GQMF filterbank structure of D-SBC limits the aliasing to adjacent bands. In the tree structure of F-SBC this is not the case—the aliasing from an earlier split can be present in several of the smaller bands. This leads to a noisier sounding coder. Second, the dynamic bit allocation leads to a definite improvement in speech quality. This is due to the inherent nonstationary nature of speech. Because the shape of the power spectral density of speech can vary widely over time, a frame oriented bit allocation scheme will always outperform one which uses a fixed, compromise bit allocation. The results from the subjective testing show that the combination of these two advantages is considerable.

D-SBC has two disadvantages as well. One disadvantage is that its additional complexity requires the use of more powerful DSP's like the DSP32. The other disadvantage is delay. The GQMF filterbank has a delay of 8 ms. The inherent one-way delay of D-SBC is 56 ms, however, because of the 16 ms frames which are processed, as shown in Table I. The tree structure filterbank used in [2] has a delay of 16 ms. Since the quantization is backward adaptive, there is no block procesing delay. We did not experiment with a backward adaptive coder for D-SBC. Such a coder would eliminate approximately 30 ms of the delay. Earlier work done by Soong, Cox, and Jayant [10] indicated that such an approach contains pitfalls. In particular, it is not clear how to reduce the bit allocation for one subband to one bit per subband sample and still know when to increase the allocation for that band in the future. Reducing the bit allocation to zero is impossible for the same reason. Thus, any dynamic bit allocation approach must use either side information or restrict the flexibility of its bit allocation.

If a coder is used in a packet network, or for applications such as voice store and forward, it is quite natural to process the speech in frames. This automatically adds a delay proportional to the frame length of the data. Since D-SBC is a block processing algorithm, it exploits this delay to provide higher quality speech at lower rates. Backward adaptive coders, such as F-SBC and ADPCM, do not exploit this delay. One of the other advantages of D-SBC is that its frames are memoryless—all of the side information is transmitted within each frame. In packet

TABLE I

| DELAY BUDGET | | |
|---|---|---|
| **Speech Coder** | **D-SBC** | **VQ-SBC** |
| Filterbanks | 8 msec | 7.5 msec |
| Buffer (Xmit & Rcvr) | 16 msec | 2.5 msec |
| Encode & Decode | 16 msec | 2.5 msec |
| Transmission | 16 msec | 2.5 msec |
| **Speech Coding Delay** | **56 msec** | **15 msec** |

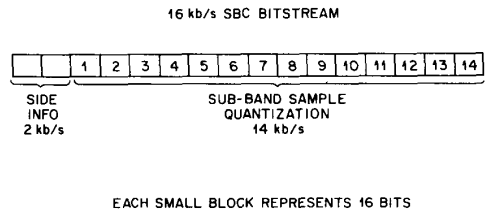



Fig. 2. Bitstream for 16 kbit/s D-SBC.

systems when a frame is lost, it may take several additional frames for backward adaptive coders to regain synchronization between the adaptive predictor coefficients of the transmitter and receiver. In the case of D-SBC, only the single missing frame would be lost. Thus, the additional delay of D-SBC is somewhat offset by its usefulness for certain applications, such as packet switching, and the fact that the side information is memoryless is an important asset.

In the next subsection we discuss techniques used to make this basic coder an embedded coder. This is possible, in part, due to the memoryless feature discussed above. It is also a result of the embedded quantizers to be discussed in the next section. However, the key feature which makes it all possible is the discovery that the dynamic bit allocation algorithm can be used to produce a prioritized bitstream. As will be seen in Section III-A-2, this is also a key to channel error protection.

As an example of how the prioritized bitstream can be constructed, consider 16 kbit/s D-SBC. For a 16 kbit/s coder, 16 ms of speech sampled at 8 kHz is represented by 256 bits. These 256 bits can be thought of as 16 individual 16-bit words. At the beginning of the stream are the 30 bits of side information. These 30 bits are packed together with two other bits into the first two 16-bit words. One of the other two bits is a synchronization bit which will contain a distinct synchronization pattern from frame to frame. The remaining bit could be used for another function such as: signaling, an indication of a change of coding rate, or an additional synchronization bit. The next 16-bit word is occupied by the sign bits of the 16 samples from the band allocated the first bit by the bit allocation algorithm. The fourth 16-bit word is occupied by the bits representing the second bit assigned by the bit allocation algorithm, and so forth. Fig. 2 shows the bitstream structure for the 16 kbit/s algorithm.

At the receiver, the side information is decoded first. Based on the side information, the bit allocation can be determined. Once this is completed, the other 224 bits of information can be decoded. The inverse quantizer is applied and the 96 subband samples are obtained. The synthesis filterbank can then be computed using this input and the resulting output is the decoded speech.

*1) Embedded Coding with D-SBC:* As mentioned previously, the bit rate in the channel that will be available to the speech coder may possibly be variable. An embedded coder would provide the ability to change rates in the network by simply "snipping" off the undesired bits from the end of the coded frame. Thus, a speech coder which is embedded has an advantage over a nonembedded coder for such a situation. The embedded coder can simply snip off the bits which can no longer be transmitted. As long as the receiver knows that those bits are not present, it can decode the bitstream. However, a nonembedded coder does not have this property. Instead, the speech coder must be notified when to reduce its rate and a second, lower rate coder must be substituted. For a nonembedded D-SBC, this would simply amount to changing the number of iterations of the bit allocator. However, this approach precludes the flexibility of changing rates once the coded speech is in the network.

Dynamic Bit Allocation SBC can be made into an embedded coder quite easily. The first step is to prioritize the bitstream, which we have already described. The second step is to use embedded quantizers for the quantization. For example, if the higher rate coder assigns 4 bits to a particular band, but the lower rate coder only assigns 3 bits, we would like the first 3 bits of the 4-bit quantizer to point to the correct output level for the 3-bit quantizer as well. A family of uniform quantizers has this property provided that they all have the same range. The main difficulty with this is that they all must have the same range. This is a serious shortcoming if the data are not uniformly distributed. Instead, we would like the larger quantizers to have a larger range because we know that the sample values being quantized do not come from a uniform distribution.

There are two requirements for embedded quantizers. The first is that the numbering scheme for the output levels must be embedded. Many numbering schemes will work, such as sign-magnitude, 2's-complement, or a natural binary code starting with 0 as the lowest level and then increasing by 1 for each new level. The second requirement is that the input thresholds for the $n + 1$ bit quantizer must be made up of the input thresholds *and output values* of the $n$ bit quantizer. We can begin with any size nonuniform quantizer and then build a family of embedded quantizers. Table II shows a family of nonuniform quantizers based on the 1-bit Max quantizer [11] for a normal distribution. In this case, the threshold levels for the 2-bit quantizer had to be $-0.798$, 0, and $+0.798$. Using Max's equations, the output levels were then computed to be optimal for a normal distribution. Those output levels were then combined with the input thresholds to form the set of thresholds for the 3-bit quantizer.

TABLE II

POSITIVE INPUT THRESHOLDS AND OUTPUT LEVELS FOR EMBEDDED QUANTIZERS OPTIMIZED FOR 1 BIT PERFORMANCE. (*ITALIC* INDICATES INPUT THRESHOLD, **BOLD** OUTPUT LEVEL.) EXAMPLE: IF A SAMPLE VALUE OF 0.95 IS TO BE QUANTIZED BY A 3-BIT QUANTIZER, IT FALLS BETWEEN THE THRESHOLDS 0.798 AND 1.366, SO IT IS QUANTIZED AS 1.053. IF THE VALUE WERE −0.95, IT WOULD BE QUANTIZED AS −1.053

| Quantizer Size (Bits) | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| | | | | **2.569** |
| | | | **2.219** | *2.219* |
| | | | | **1.997** |
| | | **1.825** | *1.825* | *1.825* |
| | | | | **1.687** |
| | | | **1.568** | *1.568* |
| | | | | **1.462** |
| | **1.366** | *1.366* | *1.366* | *1.366* |
| | | | | **1.280** |
| | | | **1.200** | *1.200* |
| | | | | **1.125** |
| | | **1.053** | *1.053* | *1.053* |
| | | | | **0.986** |
| | | | **0.921** | *0.921* |
| | | | | **0.858** |
| **0.798** | *0.798* | *0.798* | *0.798* | *0.798* |
| | | | | **0.741** |
| | | | **0.686** | *0.686* |
| | | | | **0.632** |
| | | **0.580** | *0.580* | *0.580* |
| | | | | **0.528** |
| | | | **0.477** | *0.477* |
| | | | | **0.427** |
| | **0.378** | *0.378* | *0.378* | *0.378* |
| | | | | **0.330** |
| | | | **0.282** | *0.282* |
| | | | | **0.234** |
| | | **0.187** | *0.187* | *0.187* |
| | | | | **0.140** |
| | | | **0.093** | *0.093* |
| | | | | **0.047** |

TABLE III

OPTIMAL QUANTIZER INPUT THRESHOLDS AND OUTPUT LEVELS FOR A NORMAL DISTRIBUTION (FROM [10]). (*ITALIC* INDICATES INPUT THRESHOLD, **BOLD** OUTPUT LEVEL.) EXAMPLE: IF A SAMPLE VALUE OF 0.95 IS TO BE QUANTIZED BY A 3-BIT QUANTIZER, IT FALLS BETWEEN THE THRESHOLDS 0.5 AND 1.05, SO IT IS QUANTIZED AS 0.756. IF THE VALUE WERE −0.95, IT WOULD BE QUANTIZED AS −0.756

| Quantizer Size (Bits) | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| | | | | **3.263** |
| | | | **2.733** | *2.977* |
| | | | | **2.692** |
| | | **2.152** | *2.401* | *2.505* |
| | | | | **2.319** |
| | | | **2.069** | *2.174* |
| | | | | **2.029** |
| | **1.510** | *1.748* | *1.844* | *1.908* |
| | | | | **1.788** |
| | | | **1.618** | *1.682* |
| | | | | **1.577** |
| | | **1.344** | *1.437* | *1.482* |
| | | | | **1.387** |
| | | | **1.256** | *1.299* |
| | | | | **1.212** |
| **0.798** | *0.982* | *1.050* | *1.099* | *1.130* |
| | | | | **1.049** |
| | | | **0.942** | *0.972* |
| | | | | **0.895** |
| | | **0.756** | *0.800* | *0.821* |
| | | | | **0.747** |
| | | | **0.657** | *0.676* |
| | | | | **0.605** |
| | **0.453** | *0.500* | *0.522* | *0.536* |
| | | | | **0.467** |
| | | | **0.388** | *0.399* |
| | | | | **0.331** |
| | | **0.245** | *0.258* | *0.265* |
| | | | | **0.198** |
| | | | **0.128** | *0.132* |
| | | | | **0.066** |

Table III gives the optimal Max nonuniform quantizers for 1–5 bits assuming a normal distribution. Of particular interest are the maximum output values for this family of quantizers compared to the maximum values for Table II. We see that the effective ranges of the quantizers in Table III are greater than those in Table II. This is why those in Table III are optimal and those in Table II are less efficient.

Now that we have selected embedded quantizers for the coder, all of the elements are in place for an embedded coder. The minimum bit rate is 2 kbit/s, the bit rate assigned to the side information. Once the side information is transmitted, the order of assigning the bits is the same regardless of how many bits are to be assigned. For example, if the transmitter were run at 12 kbit/s, we would have a bitstream which contained 192 bits per frame. If we only wished to transmit 11 kbit/s, we could throw away the last 16 bits of the bitstream (because it is prioritized) and the result would be the 11 kbit/s coder. In this way the D-SBC coder is embedded in 1 kbit/s chunks.

*2) Dynamic Bit Allocation and Bit Errors:* The basic dynamic bit allocation scheme was described in Section III-A. As discussed there, the scheme is designed not only to devise an optimal bit allocation, but the order in which the allocation is performed is believed to be optimal as well. Thus, the first bit allocated is believed to be more important than the second bit, which in turn is more important than the third, etc. If this is the case, then the bitstream described in Section III-A is prioritized on a 16-bit word basis. That is, an error in the third word should be more apparent than an error in the fourth word, etc.

This notion of determining the relative importance of different bits in the bitstream was pioneered by Rydbeck and Sundberg [12]. By their method one can determine the importance of individual bits empirically via computer simulation. The same sentence is processed over and over again with a different bit in error each time. Using the Segmental Signal-to-Noise Ratio (SNR) as a basis for comparison, the relative importance of each of the bits in the bitstream can be determined.

In order to design a more robust coder, more error protection should be applied to the most important bits in order to optimize the overall performance of the coder for the expected channel conditions. It is worth noting that a model for the expected channel performance must be chosen. If the model is overly pessimistic, the peak quality

of the coder will be limited because more bits will be allocated to error protection and less to the speech coder. If the model is overly optimistic, the bits allocated to error protection will be insufficient and the resulting performance will be poor. For example, in the case of cellular mobile radio, the channel performance changes depending on the location and speed of the car. If the coder could be designed to be variable rate, so that bits could be dynamically allocated between speech coding and error protection, then the resulting performance would be close to optimal for every channel condition encountered. In fact, Goodman and Sundberg reported on such a strategy for an ADPCM system in [13]. We have not done this for D-SBC, as there are a number of practical questions concerning the desirability of such a coder, but with the appropriate communications between transmitter and receiver, it ought to be possible.

However, the purpose of this section is to discuss error protection and how the bit allocation algorithm lends itself to this task. Fig. 3 shows the bitstream for the 12 kbit/s D-SBC coder. The main difference from Fig. 2 is that there are only 10 kbit/s available for quantizing the subband samples. In considering this bitstream, it is readily apparent that the side information is the most important. Without that information, the remainder cannot be decoded.

The method described in [12] was employed to determine the importance of each of the bits in the bitstream. In this method, the same bit is set in error for every frame and the resulting segmental SNR is measured. Comparing the relative SNR's of the different bits determines their importance. Fig. 4 shows the results for the 30 bits of side information. It is arranged by bands and then the order of bits for each band. The leftmost bit represents the most significant bit for band 1. The rightmost bit represents the least significant bit for band 6. As can be seen, the most significant bits in a band have the most impact on performance. While it is true that the side information bits are of unequal significance, it is our opinion that all of them must be protected to prevent the loss of an entire frame of data.

Fig. 5 shows the results for the bits used to quantize the subband samples. In each word there are 16 bits which could be in error. Simulations were run where just one of the 16 bits was set in error, and the results from the 16 simulations were averaged in order to produce Fig. 5. The monotonically increasing nature of Fig. 5 confirms our belief that the optimality of the order of the bit allocation is correct. It is true that bits in word 3 are more important than bits in word 4, etc. At the same time, the fact that the curve flattens out so quickly indicates that there is a great deal of inefficiency in D-SBC. A more efficient coder would produce a curve in which the bits assigned later produced a bigger change than the one observed here.

Channel errors were generated by simulating a typical mobile radio channel. In the case where only the side information was protected, error protection was provided by a rate 1/3 maximum free distance convolutional code of constraint length 5. In the case where both the side information and the most significant 2 kbit/s of main information were protected, error protection was provided by a rate 1/2 maximum free distance convolutional code of constraint length 7. Generator polynomials for these codes can be found in [14]. More sophisticated codes that can provide unequal error protection across the various

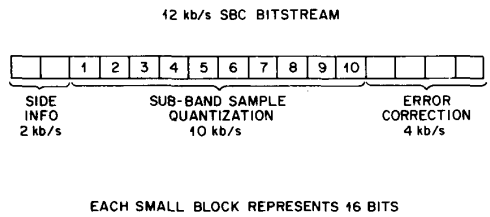



Fig. 3. Bitstream for 12 kbit/s D-SBC.

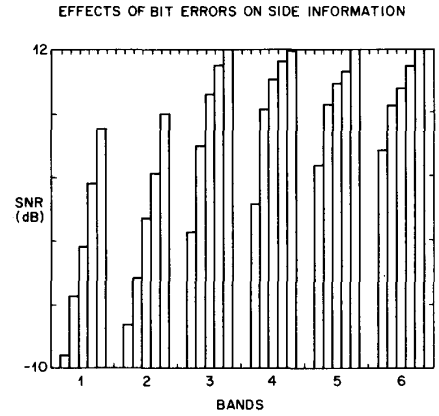



Fig. 4. Bit error sensitivity for side information. Along the $X$-axis, the order is band 1 (bits 1-5), band 2 (bits 1-5), . . . , band 6 (bits 1-5). The $Y$-axis shows the received signal-to-noise ratio when the particular bit on the $X$-axis is always in error.

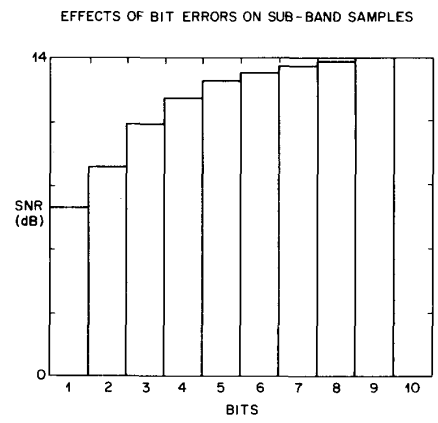



Fig. 5. Subband quantizer error sensitivity. There are 10 different priority classes of bits, as shown in Fig. 4. The $X$-axis shows the 10 different classes. The $Y$-axis shows the received signal-to-noise ratio when one bit from a particular class is always in error.

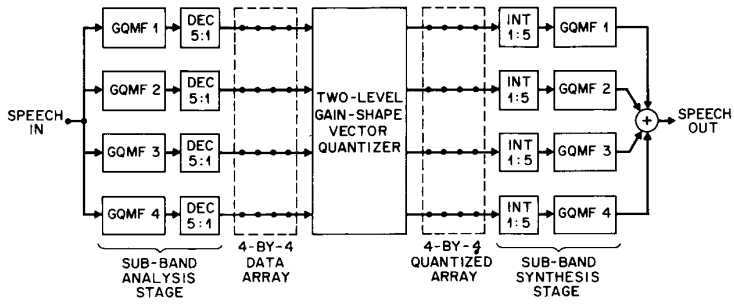



Fig. 6. Simplified block diagram of the VQ-based subband coder.

words of the side and main information would be more appropriate [15].

In listening to simulations of the coder with channel errors, we found that when the side information is unprotected, the noise caused by the errors can be quite loud, even when there is no speech input to the coder. If we protect the side information, the speech is quite intelligible, but the degradation is still apparent. If we protect at least two of the 16-bit words, for a total of 4 kbit/s, the resulting speech is quite like that of the clear channel (no bit errors) case. Some degradation occurs but the effect of the errors is much less apparent. The perceived effects of the errors are granular rather than impulsive in nature.

A key point to make is that because of the dynamic bit allocation we know exactly which bits are most critical in each frame. Dynamic bit allocation allows us to dynamically change where to put the error protection so that the most critical bits are always protected. This is a very important feature for a coder which must provide robust performance in harsh environments. In the case of D-SBC, bit prioritization is inherent to the algorithm.

### *B. General Description of the VQ-SBC Coder*

Fig. 6 shows the combined transmitter–receiver coding scheme for the proposed VQ-SBC coder. The basic structure of this scheme is quite simple. The input is first analyzed by the subband filterbank. This filterbank contains four uniformly spaced, equal-bandwidth bandpass filters which span the spectral range from 0 to 3200 Hz. After decimation by a factor of 5, to convert the sampling rate from 8000 to 1600 Hz, the stream of the four parallel band outputs is cut into successive $4 \times 4$ arrays. These data arrays are then fed to the vector quantizer which produces the corresponding quantized arrays. The quantized arrays are interpolated by a factor of 5 and filtered by the synthesis filterbank which outputs the final coded speech.

The coding rate is set to 16 kbit/s, which corresponds to 2 bits per input sample at a sampling rate of 8000 sample/s. Each data array at the analysis output contains 16 samples and corresponds to 20 samples at the analysis input (that is, 20 original speech samples). This means that the vector quantizer has 40 bits available for coding the 16 samples of each array.

The VQ subsystem performs the quantization function via the gain-shape technique, supported by a dynamic codebook allocation, as discussed in Section III-B-1.

At the transmitter, the vector quantizer performs a search over a set of codebooks and transmits the addresses of those codebook entries selected. At the receiver, these addresses are acquired from the channel and the corresponding codebook entries are retrieved from the codebook set by a simple table lookup to reconstruct the array.

Note that, apart from the subband analysis–synthesis, no other technique is employed for redundancy removal. This means that the vector quantizer is the only unit responsible for taking care of the correlation in the subband outputs. In more elaborate schemes, this task can be shared by VQ and other techniques such as adaptive prediction. Following the results of [10] and [5], we conjecture that embedding the VQ in an adaptive predictive loop will result in improved quality. However, the problem of designing the VQ for this situation is yet to be solved. Further research in this direction is recommended.

The filterbank contains four 60-tap FIR filters. The combined analysis–synthesis filterbank is a linear-phase system which produces an overall delay of 60 samples, corresponding to 7.5 ms. The VQ block size is 20 samples (2.5 ms). A delay of about 3 blocks is needed before the receiver can output the first synthesized sample. The overall coding delay is, therefore, about 15 ms which is considered to be a low delay for 16 kbit/s speech coders.

*1) Two-Level Vector Quantization:* As noted earlier, the main problem associated with the implementation of vector quantization is the coding complexity. One way of circumventing this problem (at the expense of some reduction in performance) is the decomposition of the input signal into two, loosely correlated (ideally uncorrelated) new signals and coding these signals with two, individually optimal, vector quantization operations. A coding method which generalizes this concept to any number of decomposed signals is called Hierarchical Vector Quantization (HVQ) [1], [16]. Following the HVQ terminology, we associate each decomposed signal with a certain coding level.

We chose the gain-shape approach [17] for defining the input decomposition, where the gain (first level) is simply the Euclidean norm of the input vector and the shape (second level) is the gain-normalized input vector. The reasons for this choice are twofold. First, the gains are

needed for the dynamic bit allocation which is crucial for efficient quantization of the data. Since the gains must be transmitted to the receiver, decoupling of the gains from the rest of the data is necessary in order to avoid coding the same information twice (hence, wasting bits). Second, after decimation, the shape of the decimated signal has almost no significant structure and is very loosely correlated with its energy. Therefore, gain and shape can be coded separately, and the product structure greatly reduces the VQ search complexity. Also, the structureless shapes are more "universal" (in the sense that it has no specific characteristics) which renders the VQ system more robust to varying source characteristics.

The two-level VQ is incorporated into the subband framework in the following fashion. The four parallel streams of data samples, emitted from the GQMF filters, are partitioned into successive $4 \times 4$ arrays. Let us denote such an array by $A_n = \{x_{i,j,n}\}_{i,j=1}^{4}$ where the row vector $X_{i,n} = (x_{i,n+1}, \cdots, x_{i,n+4})$ corresponds to the output of the $i$th filter at time instances $n+1, \cdots, n+4$ and $n$ is an array time index. A scalar gain is extracted from each row vector in the array, defined as

$$g_{i,n} = \|X_{i,n}\|^2 = \sum_{j=1}^{4} x_{i,j,n}^2. \tag{1}$$

The vector $G_n = (g_{1,n}, \cdots, g_{4,n})$ provides a coarse description of input spectral pattern at time index $n$. The relation between the array $A_n$ and the vector $G_n$ is shown in Fig. 7.

The two-level gain-shape quantization is, now, carried out in the following steps. First, the gain vector $G_n$ is vector quantized, using a *gain codebook*, to yield the quantized vector $\hat{G}_n = (\hat{g}_{1,n}, \cdots, \hat{g}_{4,n})$. Then, each row vector is normalized by its corresponding quantized gain, that is,

$$Y_{i,n} = \frac{X_{i,n}}{\hat{g}_{i,n}}; \qquad i = 1, 4. \tag{2}$$

Now, each of the normalized vectors are vector quantized using a set of *shape codebooks*, one codebook for each band. The reason for using a dedicated shape codebook for each band is the desire to capture the residual bandpass structure in the signal. Indeed, as mentioned earlier, most of this structure is destroyed by the decimation operation. Nevertheless, the decimated subband signals often have some residual pattern which justifies using separate codebooks. We established experimentally that a common shape codebook deteriorated the SNR performance by 0.5 to 1 dB. It was decided to retain the advantage of a band-specific codebook system even at a price of increasing the required memory by a factor of 4.

The final step, performed at the receiver, is the reconstruction of the row vectors by

$$\hat{X}_{i,n} = \hat{g}_{i,n}\hat{Y}_{i,n}; \qquad i = 1, 4. \tag{3}$$

The information sent to the receiver includes one codebook index (address) for the gain vector and four codebook indexes for the row vectors. The transmission rate, in bits per array, is the sum of the base 2 logarithms of the codebook sizes. The VQ system is depicted schematically in Fig. 8.

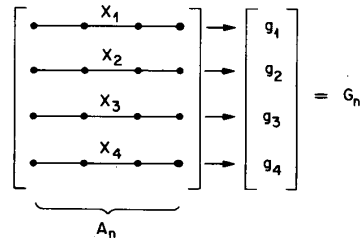



Fig. 7. The $4 \times 4$ data array at the GQMF output and the associated gain vector.

The gain and shape vector quantizers are very different in structure. The gain is quantized by a predictive vector quantizer. The shape vectors are quantized by a variable rate VQ with dynamic codebook allocation. Out of the total 40 bits available for each array, 10 bits (25 percent) are used to quantize the gain vector. The remaining 30 bits are assigned to the shape quantizer and are dynamically allocated to the various bands. The two subsystems are discussed next, starting with the gain predictive VQ.

*2) Gain Predictive Vector Quantization:* The vector sequence $G_n$ provides a rough description of the input spectrum as it evolves in time. Since this is usually a slowly varying process, one should expect high correlation between successive vectors. The gain vector, in addition, exhibits a distinct internal structure, being a replica of the short-term input power spectrum. Therefore, applying VQ to a group of gain vectors (in a form of matrix quantization) would result in an efficient gain VQ. However, this would extend the block of input data to be processed at any given time and would increase coding delay. To minimize the delay, only one array (hence, only one gain vector) is processed at a time. The intervector correlation can still be exploited using predictive coding.

The 4-dimensional gain vectors are coded by a vector predictive quantizer (VPQ) which is an extension of the common scalar predictive quantizer. One or more past input vectors are used for predicting the current gain vector using either a fixed or adaptive linear (vector) predictor. The error vector between the predicted and the actual vectors is then vector quantized and the corresponding codebook index is transmitted to the receiver. This vector predictive quantizer is shown in Fig. 9.

This approach was originally proposed in [18] for coding speech waveforms. The idea was to let the predictor take care of only the intervector correlation (i.e., between remote samples) while the VQ took care of the correlation inside the input vectors. In [18] the predictor was restricted to be of first order, namely, the current vector was predicted from the previous vector. One prediction matrix was used for linearly transforming the immediate past vector into a prediction vector. A third-order VPQ was used in here for higher prediction gain. Such a VPQ can

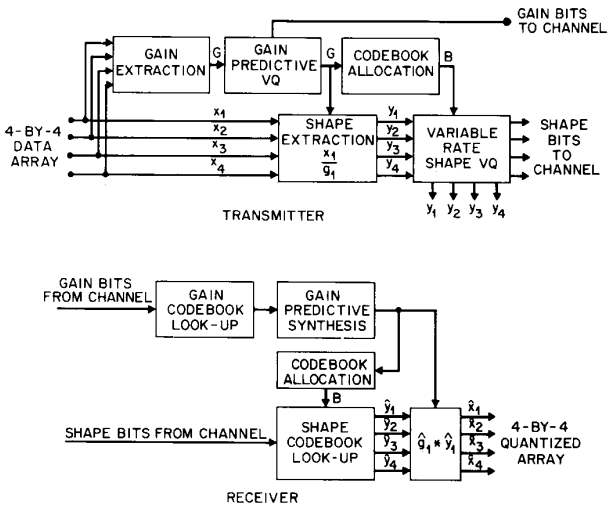



Fig. 8. Quantizer subsystem for shape and gain vectors.

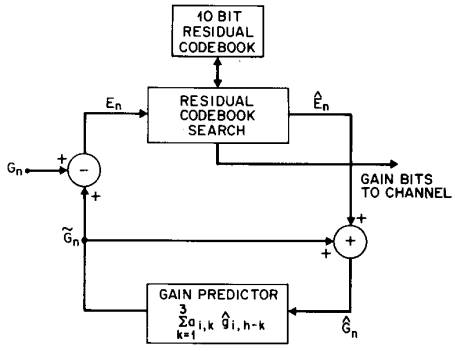



Fig. 9. Predictive gain VQ.

in general employ three 4 × 4 prediction matrices for maximum prediction gain. However, in order to reduce the complexity of such a VPQ, we employ a constrained system where each component of the gain vector is predicted independently of the other. In other words, we use in-band prediction and let the error vector quantizer take care of the interband (intercomponent) correlation.

The predictor for the $i$th band is represented by the coefficients $a_{i,1}, \cdots, a_{i,p}$ where $p = 3$, the prediction order, is the same for all bands. The predicted gain of the $i$th band at time $n$ is, therefore,

$$\tilde{g}_{i,n} = \sum_{k=1}^{3} a_{i,k} \hat{g}_{i,n-k}; \qquad i = 1, 4. \tag{4}$$

The 4 × 3 matrix $\{a_{i,k}\}$ represents the combined predictor of the VPQ system.

Typical prediction coefficients and the associated prediction gains are given in Table IV. These coefficients were optimized by minimizing the long-term average prediction error over a training set (TS) of 46 490 4-element gain vectors. This TS was extracted from speech material comprising 10 distinct utterances, spoken by 7 different speakers, 4 males and 3 females (a total of 70 different utterances). Recall that the data after the GQMF analysis are decimated by a factor of 5 and, thus, are loosely correlated. Nevertheless, Table IV shows that the *gain vectors* are still highly predictable; this is because the spectral evolution pattern is not destroyed by the decimation process. The data in Table IV are based on an unquantized gain sequence. However, as discussed in the following, using these parameters in a predictive loop resulted in a very small loss in the prediction gain.

TABLE IV
PREDICTION COEFFICIENTS AND PREDICTION GAINS OF THE GAIN-PREDICTIVE-VQ

| Band | Prediction Coefficients | Prediction Gain (db) |
|---|---|---|
| 1 | 0.04 0.33 0.60 | 9.27 |
| 2 | 0.14 0.24 0.56 | 7.38 |
| 3 | 0.10 0.35 0.48 | 6.54 |
| 4 | 0.17 0.37 0.39 | 6.21 |

Out of the 40 bits allocated to each data array, 10 bits are used to quantize the associated gain vector. The design of the required 10-bit codebook poses a problem. Since the VQ resides inside the prediction loop, the codebook has to be designed over the actual prediction error vector sequence which, in turn, is affected by codebook itself. Unfortunately, there is no known design algorithm which accounts for such a situation. Therefore, the codebook is designed in a suboptimal, "open-loop" fashion. First, a training set of error vectors is extracted, with the vector quantizer removed, that is, the quantized quantities in (4) are replaced by the unquantized ones. Then, the codebook is designed over this error-TS using the standard LBG [19] algorithm with the common mean-square error as the distortion measure. The quality of this design is assessed, simply by inserting the VQ back into the loop and comparing the resulting prediction gains to those obtained without quantization and the total SNR to that obtained in the open-loop design phase. Table V shows the SNR's and the prediction gains of each band, as well as the total SNR, for the "design" and "test" cases. As shown, the two cases differ by less than 1 dB.

The large differences in the SNR's between the various bands demonstrate the advantage of using predictive vector quantization. The optimal bit allocation, performed implicitly by the vector quantizer, produces an equal contribution to the distortion from all the bands. The equal-distortion condition indicates a state of optimality. Since the bands widely differ in their energies, while having about the same distortion, the corresponding SNR's differ accordingly.

The LBG design algorithm is basically the iterative Lloyd algorithm, extended to the vector case. It assumes some initial codebook of size $N$ and iteratively updates this codebook by alternately performing clustering and centroid calculation operations on a training set of open-loop gain error vectors. The total SNR in Table V reflects

TABLE V
PREDICTION GAINS AND SNR'S FOR THE GAIN-PREDICTIVE-VQ IN THE DESIGN AND TEST CASES

| Band | Prediction Gain (db) | | SNR (db) | |
|---|---|---|---|---|
| | Design | Test | Design | Test |
| 1 | 9.27 | 9.26 | 32.61 | 32.28 |
| 2 | 7.37 | 7.33 | 20.92 | 20.48 |
| 3 | 6.54 | 6.38 | 15.18 | 14.57 |
| 4 | 6.21 | 5.84 | 11.91 | 10.96 |
| Overall | | | 27.27 | 26.71 |

the performance of the gain VQ designed by this algorithm.

*3) Codebook Allocation for the Shape Quantizer:* Vector quantization implies optimal bit allocation over the vector being quantized. Therefore, optimal allocation exists in the gain domain and in the time domain within each individual band. However, optimal bit allocation should be performed explicitly across the bands since each row is quantized separately. This is a necessary operation if one wants to take advantage of the signal structure within the entire 2-D subband array. The objective is to distribute the available bit resources among the four bands so as to minimize the overall average quantization error. The allocation should be based on the available information about the distribution of the energy across the bands. The only such information, available at both the transmitter and the receiver, is that given by the quantized gain vector. This information is used by the allocation algorithm as described below.

Let $T_i = \{X_{i,n}\}_{n=1}^{L}$ be a long training set of vectors of the $i$th band. Let $D_i(b)$ be the average distortion at band $i$ assuming that a fixed number of bits $b$ is allocated to that band. We define the *Normalized Quantizer Function* (NQF) of the $i$th band as the function $Q_i(b)$ satisfying

$$D_i(b) = \sum_{n=1}^{L} \| \hat{g}_{i,n} \hat{Y}_{i,n} - X_{i,n} \|^2. \tag{5}$$

The distortion $D_i(b)$ can be found by designing a codebook of size $b$ for the normalized vectors $Y_{i,n}$ and calculating (5). Then, the NQF $Q_i(b)$ is found simply by

$$Q_i(b) = \frac{D_i(b)}{\sum_{n=1}^{L} \hat{g}_{i,n}^2}. \tag{6}$$

Reliable estimation of $Q_i(b)$ for all $i$ and $b$ requires designing many codebooks (for all admissible values of $b$) over a very long training set. This can be a very tedious operation. Experiments with a few values of $b$ have shown that the NQF can be approximated by

$$Q_i(b) = q_i 2^{-2(b/k)} \tag{7}$$

which agrees with known results about the general behavior of a VQ distortion versus rate function [20]. $q_i$ is the VQ constant for band $i$, and $k$ is the dimension ($k = 4$ in our case). The values of $q_i$ were found experimentally to be 5, 1.7, 1.3, and 1.1 for bands 1, 2, 3, and 4, respectively. We have avoided the effort of carefully estimating the NQF's and used the approximation of (7).

With the aid of the NQF's, the overall distortion can be approximated by

$$D = \sum_{n=1}^{L} \sum_{i=1}^{4} \hat{g}_{i,n}^2 Q_i(b_{i,n}) \tag{8}$$

where $b_{i,n}$ is the number of bits allocated to the $i$th band at time $n$. Observe that the distortion is the simple mean-square error between the original and coded vectors of all the bands. The minimization of $D$ results in nearly equal contributions to the distortion from each of the bands. This is a simple performance criterion, which is not necessarily the best one, from a perceptual point of view. In a future work, other performance criteria may be employed, like the one producing equal SNR's, in contrast to equal distortions.

The distortion (8) is minimized by minimizing the inner summation for each time $n$. Let $R$ be the available number of bits per array for coding the shape vectors (30, in our case). Then one must maintain $\Sigma_{i=1}^{4} b_{i,n} \leq R$ for any $n$. Let $S_i$ be the set of admissible bit values for the $i$th band. In our case $S_i$ is defined by all integers values between (and including) some minimum and maximum nonnegative integers. Let $S$ be the set of all vectors $B = (b_1, \cdots, b_4)$ such that $b_i \in S_i$. The allocation problem can be stated as follows. Given the set of NQF's, the quantized vector $\hat{G}$, the shape rate $R$, and the admissible set $S$, find

$$\min_{B \in S} \sum_{i=1}^{4} \hat{g}_i^2 Q_i(b_i) \tag{9a}$$

subject to

$$\sum_{i=1}^{4} b_i \leq R. \tag{9b}$$

The algorithm used in this work for solving this problem is described in detail in [21] and [1]. It is based on the Lagrangian method for integer allocations. This algorithm performs an efficient allocation for any set of NQF's and, thus, it is suitable for solving the allocation problem for the actual NQF's as may be derived from the data.

Applying the algorithm to a large training set of gain vectors, we found that the allocations were generally within the ranges [4–27], [0–16], [0–16], and [0–15] for bands 1, 2, 3, and 4, respectively. Therefore, we set the admissible ranges $S_i$ to be equal to these intervals, respectively, in the design and simulation of the coder.

Fig. 10 depicts a typical bit allocation pattern which was generated by the allocation algorithm. The top plot in the figure shows the speech segment to which the predictive gain VQ and the allocation algorithm were applied. The figure shows two distinctly different allocation patterns, one for the unvoiced and the other for the voiced sections of the waveform. As expected, most of the bits go to the lower frequency bands in the voiced region. In

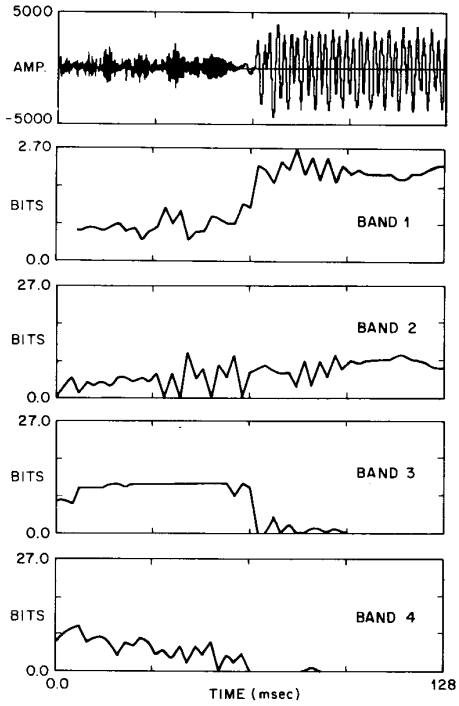



Fig. 10. Bit allocation versus time in all bands for the speech segment shown in the top figure.

the unvoiced region, the bits are more uniformly distributed across the bands, with a tendency to have more bits allocated to the third band. In fact, band 3 exhibits bit saturation which may occasionally happen due to the constraints on the admissible bit values.

*4) Multistage Shape Quantizer:* The shape quantizer subsystem has three inputs: the 4 × 4 data array, the quantized 4-dimensional gain vector $\hat{G}_n$, and a 4-dimensional bit allocation vector $B_n$. The $i$th row vector $X_{i,n}$ is normalized by the corresponding gain $\hat{g}_{i,n}$ as in (2). The resulting vector $Y_{i,n}$ is vector quantized using codebook of size $b_{i,n}$ (in bits).

To fully utilize the compression potential of the subband signals, reflected by the large dynamic range of the gain components, the bit allocations were allowed to be in the intervals $S_i$, as specified in the previous section. For higher numbers of bits, the codebook sizes and the associated search intensity become totally impractical. To obtain a realizable VQ, a suboptimal multistage [21] codebook system was constructed. This system is built of 4 subsystems, one for each band. The system of band 1 is built of 3 sections. The system of bands 2, 3, and 4 each contains 2 sections. Each section contains 9 codebooks of size 1–9 bits. The allocation value $b_i$, provided by the allocation unit, determines which of the codebooks from the $i$th codebook system is to be used in coding the shape vector $y_i$. The structure of the quantizer for band 1 is shown Fig. 11.

The coding procedure is carried out in three stages. In stage 1, $Y_i$ is coded with the first section of the codebook

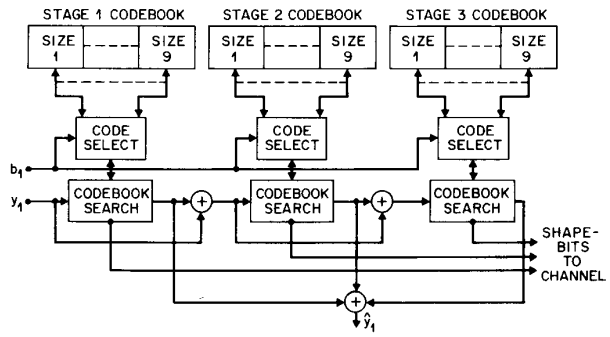



Fig. 11. Three-stage variable size VQ.

whose size is $b = \min(b_i, 9)$ and the nearest codevector $C_i^{(1)}$ is selected. If $b = 0$, no quantization is done and $C_i^{(1)}$ is set to zero. Otherwise, the algorithm proceeds to stage 2.

In stage 2, the error vector $E_i^{(1)} = X_i - C_i^{(1)}$ is coded with the second section of the codebook whose size is $b = \min(b_i - 9, 9)$ and the nearest codevector $C_i^{(2)}$ is selected. Note that the error vector at the second stage is always extracted with the aid of the 9-bit first section codebook. If $b \leq 0$ no quantization is done in stages 2 and $C_i^{(2)}$ is set to zero. Otherwise, the algorithm proceeds to stage 3.

In this last stage (actually performed only in band 1), the error vector $E_i^{(2)} = Y_i - C_i^{(1)} - C_i^{(2)}$ is coded with the third section of the codebook whose size is $b = b_i - 18$ and the nearest codevector $C_i^{(3)}$ is selected. If $b \leq 0$ no quantization is done in stage 3 and $C_i^{(3)}$ is set to zero.

The final output from the shape quantizer is, therefore,

$$\hat{Y}_i = \sum_{j=1}^{3} C_i^{(j)}; \qquad i = 1, 4. \tag{10}$$

It is clear that the required codebook storage and the search intensity are dramatically reduced, compared to the case of a full-search VQ. As an example, a full-search VQ for the highest bit allocation of 27 bits would require a codebook with $2^{27}$ codevectors and $2^{27}$ distance calculations per input vector. This is, of course, totally impractical. The three-stage VQ, employed here, requires a codebook of 1536 codevectors only, and the same number of distance calculations per vector.

The major issue of this scheme is the design of the codebook sets. The codebooks are to be optimized individually for each band and each bit allocation. Considering the allocation ranges given above, a total of 71 different codebooks have to be designed. The total size of this codebook system is 6752 codevectors, or 27 008 scalar values.

The design of the codebook systems was performed over the training set described in Section III-B-2. This TS contained 46 490 4 × 4 decimated subband arrays, that is, 46 490 4-dimensional vectors for each of the 4 bands. A nice feature of this VQ scheme is that the shape codebooks are not optimized independently of the gains but, rather, by taking the already given quantized gains into

account. In other words, the shape quantizer strives to do its best for the pair of inputs $(X_i, \hat{g}_i)$ in the first stage, for the pair $(E_i^{(1)}, \hat{g}_i)$ in the second stage, and for the pair $(E_i^{(2)}, \hat{g}_i)$ in the third stage. For convenience, the error vectors will be represented by the vector $Z_i$, defined as

$$Z_i = \begin{cases} X_i; & 0 < b_i \leq 9 \\ E_i^{(1)}; & 9 < b_i \leq 18 \\ E_i^{(2)}; & 18 < b_i. \end{cases} \tag{11}$$

Using the main training set mentioned above, 71 different *pair training sets* were built. Let $T_i^b$ be a sub-TS of the $i$th band. This sub-TS contained all pairs $(Z_{i,n}, \hat{g}_{i,n})$ which were assigned $b$ bits. To actually get this set of sub-TS's, the quantized gain sequence $\{\hat{G}_n\}_{n=1}^{L}$ ($L = 46490$) was first extracted by the optimized predictive gain VQ discussed in Section III-B-2. Then, the allocation algorithm was applied to this sequence to yield the allocation sequence $\{B_n\}_{n=1}^{L}$. With the allocation known for each pair in the pair-TS, it was easy to construct the set $\{T_i^b\}$. Note, however, that the sub-TS's of any $b > 9$ depend on the 9-bit codebook of the previous stage for the extraction of the error vectors. Therefore, the design was carried out first for stage 1, then for stage 2, then for stage 3.

Each codebook was optimized over the corresponding sub-TS using the standard LBG algorithm with the MSE criterion. However, the centroids were modified to take into account the gain-shape structure. As in the standard algorithm, $T_i^b$ was partitioned into $2^b$ cells by applying the a nearest-neighbor rule to the sequence $\{Y_{i,n}\}$, using an initial (current) codebook. Let the $j$th cell in this partition be $T_i^{b,j}$. Then, it is not difficult to show (see, e.g., [21]) that the $j$th optimal codevector (centroid) is given by

$$C_i^{b,j} = \frac{\sum_{n:(Z_{i,n}, \hat{g}_{i,n}) \in T_i^{b,j}} \hat{g}_{i,n} Z_{i,n}}{\sum_{n:(Z_{i,n}, \hat{g}_{i,n}) \in T_i^{b,j}} \hat{g}_{i,n}^2}. \tag{12}$$

The shape codebooks, designed in this way, are optimized for the given gain quantizer and for the given allocation mapping (i.e., the mapping from the gain vector $\hat{G}$ to the bit vectors $B$). The converse problem of optimizing the gain quantizer, given a set of shape codebooks, is not simple and was not pursued in this study.

## IV. Adaptive Postfiltering for Subband Coding

The idea of adaptive postfiltering is to deemphasize those portions of the spectrum which contain the more obvious noise. Alternatively, we could consider it as emphasizing those portions of the spectrum which contain the most signal content. As mentioned in the Introduction, D-SBC and VQ-SBC produce an error signal which should be spectrally flat. Since the speech spectrum is not flat, spectral flatness in the error signal is good because it means that we can obtain an advantage with adaptive postfiltering. In the case of ADPCM, Ramamoorthy and Jayant [7] used the CCITT predictor to determine their adaptive postfilter. Since that predictor is determined in the transmitter by quantized parameters, it is influenced by the quantization noise in the DPCM system. Nevertheless, it has been shown to be of value by increasing the perceived quality of the output speech.

In the case of D-SBC, the side information is not influenced by the quantization noise of the subband signals. One possible method for designing an adaptive postfilter is to use the side information. The idea is to deemphasize those bands with lesser energy while emphasizing those with greater energies. The construction of the adaptive postfilter is done on a frame-by-frame basis.

Suppose that we have a digital filterbank consisting of bandpass filters with the same nominal bandwidth as the filters used in the subband coder. At the band edges, these filters have an attenuation of 6 dB. If they are added together, the result is an impulse response with the delay of the filterbank. (One way to obtain such a filterbank is to convolve the analysis and synthesis filterbank filters for each band. This results in a filter with 6 dB of attenuation at their band edges. Truncating the result to a reasonable length in order to minimize computation and delay will still leave an attenuation of 6 dB at the edges.) Once the filters are obtained, the following type of equation can be used to combine them into a single filter:

$$g(n) = \sum_{k=1}^{6} \left(A + (1 + A)\gamma E(k)/S\right) h_k(n). \tag{13}$$

In (13) $h_k(n)$ is the bandpass filter for the $k$th band, $E(k)$ is the quantized rms energy for band $k$, and $A$ is a constant used to weigh the relative amount of postfiltering. $S$ is the sum of $E(k)$. Typical values for $A$ and $\gamma$ are 0.7 and 2.5, respectively.

When we first tried this scheme, we were most impressed with the way it reduced the high-frequency noise that was produced at the output of the coder. At the same time, however, we could detect a periodic artifact caused by the filtering. Our suspicion is that the abrupt changes in the filter every 16 ms cause an audible frame rate noise. Nevertheless, this frame rate noise is very slight, and we feel that the coder sounds better with adaptive filtering than without it.

A second method for doing adaptive postfiltering uses a backward adaptive LPC analysis. In this method, the LPC coefficients are determined on the output speech. In a sense, this method is more similar to the Ramamoorthy and Jayant method because it is based on an adaptive predictor produced from the quantized output speech. We know that the presence of white noise in the output speech will reduce the effectiveness of this filter. Initially it was designed for the 16 kbit/s coder. Like the first postfilter, it significantly reduced the high-frequency "hiss" of the coder. In addition, it did not suffer from a frame rate noise. For the 16 kbit/s coder, we felt this gave it an advantage over the first postfilter.

The method used for computing the predictor coeffi-

cients is based on an LMS algorithm. Let $s(n)$ be the speech sample value at time $n$. The predictor filter is given by $\{h(1), h(2), \cdots, h(6)\}$. The prediction value for $s(n)$ is given by

$$\hat{s}(n) = h(1)\, s(n-1) + \cdots + h(6)\, s(n-6) \quad (14)$$

and the error, $e(n)$, is defined by

$$e(n) = s(n) - \hat{s}(n). \quad (15)$$

The new value of $\{h(k)\}$ is given by

$$h(k) = Bh(k) + u \operatorname{sgn}(e(n)) \operatorname{sgn}(s(n-k)). \quad (16)$$

Typically, $B = 0.996$ and $u = 0.008$. By using only the sign bit of the error and the previous sample values, the adaptation rate is slower. However, this works to our advantage. If the speech is noisy due to either quantization or bit errors, the effect of this noise on the operation of the postfilter is less pronounced. In using just the sign bit to update our predictors, we are following a convention also used in the CCITT ADPCM predictor updates. In addition to giving robust performance, it avoids a messy normalization problem which would be required if the actual values for $e(n)$ and $s(n-k)$ were used.

The output of the postfilter, $x(n)$, is given by

$$x(n) = Gs(n) + g^1 h(1)\, x(n-1) + \cdots + g^6 h(6)\, x(n-6). \quad (17)$$

The value of $G$ can be adjusted so that the overall output level of the filter sounds the same as that of the original speech. If $G$ is 1, the postfiltered speech usually sounds louder than the original speech. A setting in the range of 0.5–0.75 is usually about right. We have been using a value of about 0.7 for $g$ in our simulations at 12 kbit/s and 0.6 for simulations at 16 kbit/s. In general, the higher the value of $g$, the greater the effect of postfiltering. While postfiltering reduces obvious noise, it also tends to make the speech more muffled because these filters are low-pass filters much of the time. Thus, it is not advantageous to use too large a value for $g$.

At 12 kbit/s, D-SBC has more granular noise than 16 kbit/s D-SBC, but in spite of this, the LMS postfilter still provides a perceptual improvement. We have compared the performances of both postfilters on the same frames of speech. The results were quite remarkable in that they showed how subtle the filtering is! For voiced speech, the typical difference between the maximum boost and maximum deemphasis was less than 5 dB. For unvoiced speech this difference was even smaller. The typical difference between the two filters was less than 1 dB. Most listeners to our real-time simulation do not complain of any muffled quality in the output. When we turn off the adaptive postfilter, they find the increased high-frequency "hiss" readily apparent. We concluded that adaptive postfiltering, if used sparingly, can definitely enhance the perceived quality of D-SBC without degrading its intelligibility.

These filters require about 1 million instructions/s on the DSP32. This corresponds to approximately 25 percent of the device's real-time capability. If the device would otherwise be idle during that time, then adaptive postfiltering is definitely worthwhile. If, in the future, we can improve the D-SBC algorithm at the price of increased computational complexity, the amount of computation required for the postfilter may be considered a burden and we may need to simplify one of the techniques.

Finally, it is our opinion that when the channel errors can be confined to that part of the bitstream which causes granular noise, adaptive postfiltering helps to reduce this granular noise as well. We observed that even when simulating the coder over a noisy channel, the version using the adaptive postfiltering sounded better than the version without it. This was still the case even when only the side information was protected. Thus, we find that adaptive postfiltering is another element which can be used to make a more robust coder.

## V. Coder Implementations

At the very outset in the design of both D-SBC and VQ-SBC, one of the principal goals of the research was to design coders which could be implemented on a single AT&T DSP32. The motivation for this goal was that a full-duplex codec consisting of a DSP32 and a $\mu$-law or linear codec would be an attractive implementation, in that it would provide a low cost and small size implementation with sufficient flexibility for future enhancements. Both coders were successfully implemented full-duplex on a single 16 MHz DSP32 processor. In Sections V-A and B we describe the implementation of D-SBC and VQ-SBC, respectively.

### A. Implementation of D-SBC

Fig. 12 describes the D-SBC coder hardware as it was implemented in one prototype. The hardware consists of the DSP32, an INTEL 8051 microcomputer, an echo canceller, and a $\mu$-law codec. The microcomputer connects to the DSP32 via a parallel DMA link. It is used as an interface between the DSP32 and the RS-232 serial interface through which the 16 kbit/s data stream passes. (The DSP32 has a full-duplex serial port, but it is used for the codec.) The 8051 was chosen because it requires no external logic to pass signals between the two devices. This chip contains 128 words of RAM and 4K bytes of ROM. Its main function is to interface the 16 kbit/s data channel with the DSP32 and perform digital frame synchronization and telephone line control. The echo canceller performs 8 ms of echo cancellation. Its algorithm is similar to that described by Duttweiler [22].

The main effort in the hardware implementation was to prove that the coder could fit into the 40-pin version of the DSP32. In order to save program memory, this coder did not include the adaptive postfiltering. The coder was first developed on the DSP32 development system which uses the 100-pin package DSP32 and can address up to

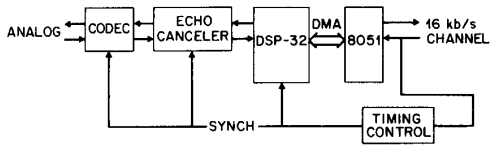



Fig. 12. Block diagram of the hardware implementation.

56K bytes of off-chip RAM for either instructions or data. As initially written, the coder did not fit into the more limited address space of the 40-pin package, which contains 4K bytes of RAM and 2K bytes of ROM; 90 percent of the on-chip RAM and 120 percent of the ROM were filled by the code. To cope with this overflow, a portion of the code can be stored in the ROM of the 8051 microcomputer. At startup or reset, this code can be loaded into the RAM of the DSP32 from the 8051 ROM. In this way, the entire code can fit into the combined internal ROM of the two devices.

With regard to real time, the full-duplex coder fits nicely into the 16 MHz DSP32. The coder requires 6.7 ms per frame for the encoder, and 4.7 ms for the decoder, leaving a real-time margin of 28 percent. The DSP32 is a 4 million instruction per second (MIPS) device. These figures indicate that full-duplex operation requires 2.85 MIPS. There appears to be enough real-time capacity left to improve the coder even further. The most limiting resource for the 40-pin DSP32 is memory. This could be alleviated by using the 100-pin DSP32 and using external memory.

A second real-time implementation of D-SBC containing all of the features described in earlier sections has been accomplished using the DSP32 signal processor development system. Specifically, it includes the prioritized bitstream, embedded coding, and the second type of adaptive postfiltering described in Section IV. This version runs at rates between 3 and 20 kbit/s, but only provides useful quality at rates of 10 kbit/s and above. Both encoding and decoding are accomplished on a single DSP32, with approximately 83 percent of its real-time capabilities utilized.

### *B. Implementation of VQ-SBC*

VQ-SBC has been implemented in real time on a single AT&T DSP32 signal processor integrated circuit, external memory, and a codec. However, in order to achieve real-time operation using this processor, some modifications to the algorithm had to be made. This section will, first, assess the algorithm complexity, discuss the tradeoffs in the real-time implementation, briefly describe the coder hardware, and give coder performance figures (as segmental SNR's).

The most computationally intensive part of the coder algorithm is the vector search. The squared error distance per codevector can be rewritten as

$$\sum_{i=1}^{4} (x_i - y_i)^2 = \sum_{i=1}^{4} \{x_i^2 + y_i^2 - 2x_iy_i\} = \sum_{i=1}^{4} x_i^2 + \sum_{i=1}^{4} y_i^2 - \sum_{i=1}^{4} 2x_iy_i \quad (18)$$

where $x$ is the vector to match and $y$ is the codevector. This form is strictly a sum of products and is much more suitable for computing on the DSP32. The $\Sigma\, y_i^2$ term can be precomputed and stored as part of the codebook. The $\Sigma\, x_i^2$ remains constant for all codevectors in the search and can be neglected. In this manner, the minimum distance search can be done in 9 DSP32 instructions (2.25 microsec) per 4-dimension codevector, including test and branching overhead.

However, the gain codebook is 10 bits and the shape codebooks are, as a worst case, a series of three 9-bit books plus one 3-bit book for the four shape vectors. Even with the compact distance calculation, this corresponds to 3.6 million instructions per second (MIPS) for the gain vector search and 5.5 MIPS for the shape search. Since the design goal was to implement the coder on a single DSP32, a 4 MIPS machine, the vector search had to be simplified.

In order to get an idea of the complexity of the entire algorithm, two other computationally intensive portions of the algorithm need to be considered. These are the subband analysis/synthesis filtering which requires 0.86 MIPS, and determination of bit allocation which requires 0.84 MIPS for two executions (once in the transmitter and once in the receiver) This results in an algorithm which requires at least 10.9 MIPS, or 1362 operations per input sample.

Since the codebook search dominates the complexity, it was decided to use a tree search rather than full search codebook structure. An $N$-bit full search codebook was redesigned to be a two-level $m$-ary tree structure, with $m$ equal to $N/2$. It was felt that limiting the number of levels to two would minimize the suboptimality of the tree search relative to a full search. For the gain codebook, this resulted in a 5-bit level 1 codebook indexing into 32 5-bit level 2 codebooks. In the cases of $N$ being odd, the smaller number of bits was assigned to the level 1 codebook. So, for example, a 9-bit shape codebook was restructured into a 4-bit level 1 codebook and 16 5-bit level 2 codebooks. Since, depending on the bit allocation, some shape codebook may be already quite small, tree structures were used only for codebooks greater than 6 bits. All other books were fully searched.

The use of tree structure for the larger codebooks reduced the complexity of the gain vector search to 0.23 MIPS and the shape search to 0.89 MIPS. The entire algorithm would now require 2.83 MIPS, or 354 operations per input sample. Allowing for other coder operations and some overhead, this was within the computational power of a single DSP32.

However, codebook size was another important constraint of the implementation. Since the DSP32 is a floating-point machine, all codebooks were stored in 4-byte floating-point format. Furthermore, the two-level tree-structured codebooks are larger than full search codebooks by an amount equal to the number of vectors in the first-level codebook. Finally, the distance calculation of (18) requires the sum of the squares of the four codevector

TABLE VI
CODER PERFORMANCE OF A REAL-TIME HARDWARE IMPLEMENTATION COMPARED TO THAT OF A FORTRAN SIMULATION

| Implementation | SEGSNR |
|---|---|
| Fortran, full search | 21.90 dB |
| Fortran, tree search | 20.83 dB |
| real-time hardware | 20.44 dB |

elements to be stored as an effective fifth element of the vector. Therefore, each codevector requires a total of 20 bytes of storage. All this resulted in large codebooks: 32 level 1 codevectors and 1024 level 2 codevectors at 20 bytes/codevector totaled 21 120 bytes for the gain codebook, and, similarly, 138 880 bytes for the set of 71 books that made up the shape codebook.

Since the DSP32 has a 64 Kbyte address space, a memory management scheme was used to facilitate codebook access. The coder hardware has 256 Kbytes of memory which is organized into 128 pages of 2048 bytes each. In our hardware architecture, the DSP32 can access any 16 out of 128 pages, for a total of 32 Kbytes of external memory. This is done by manipulating the DSP32 address bits so that the 32 Kbyte address space is mapped into the 256 Kbyte physical address space. Since no data are actually moved in physical memory, a page of memory can be released and another mapped into its place in a single machine instruction cycle. The 32 Kbytes of directly addressable memory was sufficient for storage of program, data, and the single largest codebook (the gain codebook). Prior to the gain quantization and at each stage of the shape quantization, the required codebook is swapped into the address space. After the search of any codebook, it is mapped out and other codebooks mapped in as needed.

The coder suffered some modest performance degradation due to using tree-structured codebooks as opposed to the full search structure of the original algorithm. The performance results are summarized in Table VI. The Fortran simulation of the original algorithm produced a segmental SNR of 21.90 dB. Using a tree search structure for the books larger than 6 bits degraded the performance of the Fortran simulation approximately 1 dB, to 20.83 dB. It was gratifying to see that the real-time hardware produced a nearly identical segmental SNR of 20.44 dB. This is not unexpected, since the Fortran simulation used only single precision floating-point arithmetic, identical in precision to the floating-point arithmetic of the DSP32. In fact, the slight decrease in SNR could only be attributable to the difference between the floating-point number format of the DSP32 and the general-purpose computer used in the simulation.

## VI. PERFORMANCE

When we speak about the performance of a speech coder, there are a number of attributes which can be used to describe its performance. Probably the most important is the speech quality of the coder. A second important measure is its complexity, as this will largely determine the cost of the implementation. A third measure is the delay of the speech coder. As we have seen, low delay is important because any delay of greater than a few milliseconds will require the use of echo cancellers in the implementation. A fourth measure is the ability to pass data and signaling tones. In this section we will discuss how the two coders compare to each other as well as other medium bit rate speech coders for each of the first three categories. We have not tested for the fourth measure, but feel that both of these coders can reliably handle signaling tones.

One can divide speech quality measures into two broad categories, objective and subjective. A typical objective measure is segmental SNR, while Diagnostic Acceptability Measure and Mean Opinion Score are two subjective measures. Standard test sentences processed by the 16 kbit/s D-SBC coder (without adaptive postfiltering) were submitted for a Mean Opinion Score test, together with data from other coders including 16 kbit/s Multipulse LPC [23], 12 and 16 kbit/s ADPCM with adaptive postfiltering [7], [24] and 16 kbit/s F-SBC [2]. The VQ-SBC was not included in this test because the implementation was not yet complete at the time of the test.

One of the purposes of the test was to determine how robust coder performance was for multiple asynchronous encodings. (Adaptive postfiltering is only useful for single encodings, as the results for APF-ADPCM show.) There were four test conditions for each of the coders. In condition 0, the speech was processed by the hardware and algorithm, but the salient parameters and/or signals that would normally be quantized for transmission over a 16 kbit/s channel were left at the full precision of the processor. The purpose of this condition was to establish a baseline for the maximum performance for each algorithm. In condition 1, the speech was processed by a single encoding. In conditions 2 and 3, the speech was processed by 2 or 3 asynchronous encodings, respectively. (The APF-ADPCM coders were not intended for multiple tandems, as they were considered primarily for voice store and forward applications.) In addition to these coders, standard noise conditions were added.

All coders had identical material input to them. Most of the coders used an 8 bit $\mu$-law codec with its own internal antialiasing filters. The one exception was the fixed bit allocation subband coder, which used a 12-bit linear A/D and D/A. Also included in the processing of the material was a Rockland 48 dB/octave bandpass filter on the output of the coder. In addition to making the bandwidth of the test material uniform, the filter also suppressed the 8 kHz tone generated by the $\mu$-law codec. The nominal bandwidth of the filter was 200–3200 Hz.

Fig. 13 gives a comparison of the test results for the coders listed above. To put these results in perspective, the source material received an average score of 4.34 on the test. This material is equivalent to a single encoding of $\mu$-law PCM. The score for MPLPC with 0 encodings

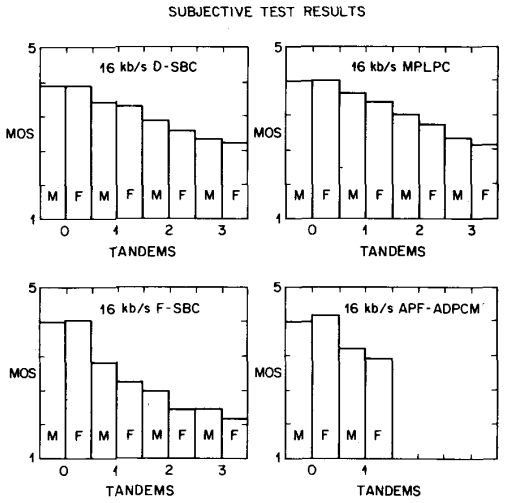



Fig. 13. Comparison of results for 16 kbit/s coders. Notes: Tandems refers to the number of tandem encodings. A zero tandem implies that the coder parameters were left at the full machine precision. **M** refers to results for male speakers while **F** refers to results for female speakers.

is 4.2 and is equivalent to two encodings with 64 kbit/s $\mu$-law PCM. The comparison of the dynamic bit allocation subband coder with multipulse LPC and the fixed bit subband coder are the most relevant for this discussion.

In comparing the performance of D-SBC to MPLPC, we see that the multipulse coder is slightly better with an average score of 3.8 versus 3.69 for the SBC. Both coders perform better for male talkers than female talkers. The difference between them for a single encoding, 0.11, is not statistically significant when the standard errors of the two coders are taken into account [16]. If we compare their scores for condition 0, we see that the advantage for multipulse is 0.08. This indicates that the reduced bandwidth of the SBC filterbank has only a small effect on quality. It also implies that most of the difference for the two coders for condition 1 is probably attributable to the slightly lower bandwidth of the subband coder. In terms of their performance for multiple encodings, the two coders performed equally well. Each fell 0.51 points for the second encoding and the two coders were almost equal for three encodings.

In comparing the performances of dynamic bit allocation SBC and fixed bit allocation SBC, D-SBC has a considerable performance advantage. The average advantage for one encoding is 0.66 quality points. For male speakers it is only 0.48 points, but for female speakers it is 0.84 points. For two encodings the difference is 0.81 points and for three encodings the difference is 0.76. These differences indicate that listeners found a relatively large difference between the two coders. We concluded that the quality of the dynamic bit allocation SBC represented a significant increase in quality over the fixed bit allocation SBC.

Because no subjective test results for VQ-SBC are available, we assessed the performance of this coder by the objective segmental signal-to-noise ratio (SNRSEG), as well as by subjective, informal listening to the coded speech.

The SNRSEG was measured over the entire design training set described earlier and over several different short utterances within the TS. These are called Inside-Training-Set ($I$) tests. Using the codebook set generated by the design procedure, many different speech segments not included in the design TS were coded. These are called Outside-Training-Set ($O$) tests. It was found that the SNRSEG varied in the range of 20 to 23 in the $I$-tests and in the range of 18.6 to 23 in the $O$-tests. As expected, the $O$-tests indicated some degradation in performance. However, this degradation (about 1.5 dB) does not seem to be perceptually severe. This suggests that the gain-shape VQ diminishes the seriousness of the VQ robustness problem, probably due to the fact that the shape (normalized) vectors are more "universal" in nature and less speech specific.

A few typical tests are summarized in Table VII. The test segments are identified by a file number and are typed $I$ or $O$ to denote inside and outside training-set cases. Also, the segment lengths (s) and the number of male and female speakers in each set are given. This table enables future references to this database for further studies and comparison to other coding schemes.

The performance of this coder, as indicated by the SNRSEG values, is promising, considering the short coding delay of 15 ms. This also indicates that the vector quantizer performs very efficiently when used as the only means of redundancy removal. Such a conclusion can be made by comparing the performance of this coder to that reported in [5] and [10].

Listening to the coded speech indicates that the naturalness and the richness of the speech material was perfectly preserved and there was no loss of intelligibility. However, a low-level quantization distortion was sometimes noticeable in the voiced sounds of female speakers. We conjecture that this perceivable noise is a result of imperfect aliasing cancellation due to quantization distortion.

In our judgment, this coder may be rated as producing very good communication quality speech. Based on a comparison to D-SBC, we estimate the mean opinion score (MOS) quality rating to be about 3.5–3.7 on a scale of 1–5. The addition of adaptive postfiltering should raise this score a small amount, perhaps 0.2.

The two coders can be compared in complexity to other coders such as F-SBC, MPLPC, and ADPCM. In a rank ordering of coders by complexity, ADPCM would be the least complex, followed by F-SBC, D-SBC, MPLPC, and VQ-SBC. In a recent ICASSP paper, up to 8 half-duplex ADPCM coders were implemented on a single chip [25]. F-SBC was implemented on two first generation DSP chips. The implementations of D-SBC and VQ-SBC have already been described. MPLPC was also implemented on a DSP32 but required external RAM. It could not fit on the 40-pin DSP32. In addition, for certain computations MPLPC required the floating-point capability of the

TABLE VII
TYPICAL SNRSEG VALUES FOR INSIDE AND OUTSIDE TRAINING-SET UTTERANCES CODED BY THE VQ-BASED SUBBAND CODER

| File # | Type | Length (sec.) | Speakers | SNRSEG (db) |
|---|---|---|---|---|
| 1 | I | 116.2 | 4M,3F | 23.10 |
| 2 | I | 2.05 | 1F | 21.36 |
| 3 | I | 1.79 | 1F | 22.93 |
| 4 | I | 2.30 | 1M | 20.10 |
| 5 | I | 1.79 | 1M | 20.30 |
| 6 | O | 3.87 | 1F | 19.88 |
| 7 | O | 2.08 | 1M | 22.35 |
| 8 | O | 1.79 | 1M | 21.20 |
| 9 | O | 10.24 | 2M,2F | 20.80 |
| 10 | O | 2.75 | 1M | 21.29 |
| 11 | O | 3.26 | 1M | 22.61 |
| 12 | O | 2.56 | 1M | 19.70 |
| 13 | O | 4.18 | 1F | 21.58 |

DSP32. Both D-SBC and VQ-SBC could be implemented on fixed-point DSP chips because neither algorithm requires floating-point computation.

One of the most important characteristics of a speech coder is its delay. Unless the delay is almost zero, echo cancellers will be needed, thus increasing the cost of the implementation. If the delay is too great, holding a conversation becomes an onerous chore.

Table I gives the delay budget for D-SBC and VQ-SBC. It also illustrates the delay structure of block coders in general. The first delay is due to the filterbanks. This delay is evenly divided between the encoder and the decoder and is unavoidable for a subband coder. For LPC-based coders which use a window larger than their block size, there is a corresponding delay. For example, the MPLPC described in [23] had a window size of 20 ms and a block size of 10 ms. It incurs a delay of one-half the excess window size, e.g., 5 ms. The next delay is due to the buffers at both transmitter and receiver. Speech must be collected before analysis can begin at the transmitter. At the receiver, the speech is decoded and then stored in a buffer until it is played out. The total amount of time spent in these two buffers must be equal for all samples. The third delay is due to the processor. It takes a finite amount of time to process a block of speech. Since the DSP32 does both encode and decode on a single chip, the total processing time is under the block size. This amount could be reduced further by a faster processor. However, faster processors cost more and would just be sitting idle for a greater amount of the time. The fourth delay is due to the transmission time. We assume that the channel is matched to the coding rate and that the time spent transmitting the block is equal to the block length. It is worth noting that for every 1 ms the block size is reduced, the overall delay is reduced by 3 ms.

The delay of D-SBC is almost four times that of VQ-SBC, 56 versus 14 ms. By way of comparison, ADPCM has almost no delay, just 0.125 ms, and MPLPC as implemented in [23] has a delay of 35 ms.

## VII. SUMMARY AND CONCLUSIONS

In this work we have investigated two very different ideas in subband coding. D-SBC uses embedded scalar quantization and a prioritized bitstream. VQ-SBC uses multistage vector quantization. Both coders were implemented in real-time hardware based on the AT&T DSP32 signal processor. We compared both coders to three other coders, fixed bit allocation subband coding (F-SBC), multipulse LPC (MPLPC), and ADPCM with adaptive postfiltering.

To summarize our comparisons, we found that in terms of performance, the dynamic bit allocation subband coder is comparable to multipulse LPC at 16 kbit/s. However, the embedded feature of D-SBC gives it more flexibility and it can be implemented using fixed point which should provide a cost advantage as well. This combination of performance, flexibility, and implementation cost make D-SBC a more attractive coder than MPLPC for 12–16 kbit/s applications. In comparison to other 16 kbit/s coders, such as F-SBC and APF-ADPCM, D-SBC has a significant edge in performance and the cost of implementation is comparable, depending in large part on the intended application.

It is not our intention to propose D-SBC for use throughout the telephone network. Its delay, implementation cost, and performance are all stumbling blocks which will keep it from widespread use in the network. While its performance is better than other coders, it is still not up to the level of $\mu$-law PCM or the 32 kbit/s CCITT G.721 standard. Its delays are large enough to require echo cancellers which add to its implementation cost. (To be fair, it would seem that the delay of any high quality 16 kbit/s coder will be large enough to require an echo canceller.) However, its implementation cost, good quality, and "memorylessness" do make it a good candidate for use in limited applications.

VQ-SBC is based on a combination of subband coding and vector quantization. A two-level shape-gain vector quantizer, controlled by a dynamic codebook allocation, was incorporated into a subband analysis-synthesis framework, built of generalized quadrature mirror filters. Good communication quality was achieved, with a coding delay of only 15 ms. We have demonstrated that, in spite of some modifications and implementation tradeoffs, the real-time coder delivers essentially the same performance as that of its software counterpart.

The results of this study show that VQ can successfully perform the combined tasks of redundancy removal and coding of the subband signals. This encourages further investigations of more advanced VQ structures for the purpose of subband coding. In the following, we briefly outline a few possible directions for further research.

VQ can be embedded in an adaptive predictive loop in each subband. Vector quantization of the prediction residual (instead of the waveform itself) may result in higher perceptual quality.

In comparing D-SBC and VQ-SBC, we see that their strengths complement each other. The greater coding efficiency of vector quantization and the low delay it makes possible would be useful additions to the flexibility of D-SBC. Perhaps a combination of the two coders could be produced in the future. It would require some sort of

embedded vector quantization which would still be relatively efficient.

## References


[1] Y. Shoham, "Hierarchical vector quantization with application to speech waveform coding," Ph.D. dissertation, Dep. Elec. Comput. Eng., UCSB, 1984.

[2] R. E. Crochiere, R. V. Cox, and J. D. Johnston, "Real-time speech coding," *IEEE Trans. Commun.*, vol. COM-30, pp. 621-634, Apr. 1982.

[3] T. A. Ramstad, "Sub-band coder with a simple adaptive bit allocation algorithm," in *Proc. ICASSP '82*, pp. 203-207.

[4] M. Honda and F. Itakura, "Bit allocation in time and frequency domain for predictive coding of speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 465-473, June 1985.

[5] F. K. Soong, R. V. Cox, and N. S. Jayant, "A high quality sub-band speech coder with backward adaptive predictor and optimal time-frequency bit assignment," in *Proc. ICASSP '86*, pp. 2387-2390.

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

[7] V. Ramamoorthy and N. S. Jayant, "Enhancement of ADPCM speech by adaptive postfiltering," in *Proc. ICC '85*, pp. 29.2.1-4.

[8] J. Rothweiler, "Polyphase quadrature mirror filters—A new sub-band coding technique," in *Proc. ICASSP '83*, pp. 1280-1283.

[9] R. V. Cox, "The design of uniformly and nonuniformly spaced pseudoquadrature mirror filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1090-1096, Oct. 1986.

[10] F. K. Soong, R. V. Cox, and N. S. Jayant, "Sub-band coding of speech using backward adaptive prediction and bit allocation," in *Proc. ICASSP '85*, pp. 1672-1675.

[11] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Theory*, pp. 7-12, Mar. 1960.

[12] N. Rydbeck and C. E. Sundberg, "Analysis of digital errors in nonlinear PCM systems," *IEEE Trans. Commun.*, vol. COM-24, pp. 59-65, Jan. 1976.

[13] D. J. Goodman and C. E. Sundberg, "Combined source and channel coding for matching the speech transmission rate to the quality of the channel," *BSTJ*, vol. 62, no. 7, pp. 2017-2036, Sept. 1983.

[14] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, 1983.

[15] J. Hagenauer, "Rate compatible punctured convolutional codes," in *Proc. ICC '87*, pp. 1032-1036.

[16] A. Gersho and Y. Shoham, "Hierarchical vector quantization of speech with dynamic codebook allocation," in *IEEE Proc. Int. Conf. ICASSP-84*, pp. 10.9.1-10.9.4.

[17] 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, pp. 474-488, June 1984.

[18] V. Cuperman and A. Gersho, "Adaptive differential vector coding of speech," in *Conf. Rec., GLOBECOM 82*, Dec. 1982, pp. 1092-1096.

[19] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizing design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84-95, Jan. 1980.

[20] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 373-380, July 1979.

[21] B. H. Juang and A. H. Gray, Jr., "Multiple stage vector quantization for speech coding," in *IEEE Proc. Int. Conf. ICASSP-82*, pp. 597-600.

[22] D. L. Duttweiler, "A twelve-channel digital voice echo canceller," *IEEE Trans. Commun.*, vol. COM-26, pp. 647-653, May 1978.

[23] H. Alrutz, "Implementation of a multipulse coder on a single chip floating-point signal processor," in *Proc. ICASSP '86*, pp. 2367-2370.

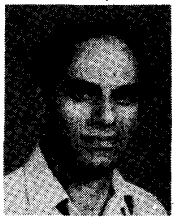
[24] R. V. Cox, "A family of ADPCM coders implemented on real-time hardware," in *Proc. ICASSP '87*, pp. 964-967.

[25] J. D. Beatty, R. D. Calder, Jr., P. Farazi, D. P. Kelly, and J. L. Melsa, "Custom VLSI design of a single chip multi-channel ADPCM processor," in *Proc. ICASSP '87*, pp. 479-482.


**Richard V. Cox** (S'69-M'70-SM'87), for a photograph and biography, see this issue, p. 382.

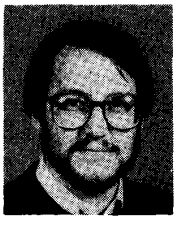
**Steven L. Gay** received the B.S.E.E. degree from the University of Missouri-Rolla in 1979. He joined Bell Laboratories as a member of the Technical Staff in 1980 and, participating in Bell Laboratories One-Year-On-Campus Program, received the M.S. degree in electrical engineering from the California Institute of Technology in 1981.

In 1983 he joined ITT—Defense Communications Division's Voice Processing Department where his activities included speech coding research and systems engineering. He returned to Bell Laboratories in 1985. His principle research interests lie in the areas of speech and image processing. He is currently pursuing the Ph.D. degree in electrical engineering at Rutgers University, Piscataway, NJ.



**Yair Shoham** (S'82-M'85) received the B.Sc. and M.Sc. degrees from the Technion-Israel Institute of Technology, Haifa, Israel, in 1969 and 1973, respectively, and the Ph.D. degree in 1985 from the University of California at Santa Barbara.

In 1969 he joined RAFAEL, the Armament Development Authority, in Israel, where he was involved in the development of special-purpose tactical communication circuits and systems. From 1978 to 1980 he was a Group Head, responsible for various projects in the area of point-to-point communication links. In 1985-1986 he was a consultant for AT&T Bell Laboratories, and in 1986 he joined the Signal Processing Research Department of Bell Laboratories. His current interests are in the area of digital signal processing, vector quantization algorithms, with applications to digital speech coding.



**Schuyler R. Quackenbush** (S'74-M'86) received the B.S. degree from Princeton University, Princeton, NJ, in 1975. After four years in industry as a Design Engineer, he entered the Georgia Institute of Technology, where he received the M.S. degree in electrical engineering in 1980, and the Ph.D. degree in electrical engineering in 1985.

For the latter half of 1985, he was a Staff Research Associate at Georgia Tech. He joined AT&T Bell Laboratories in 1986, where his research interests have been in speech quality assessment, speech coding algorithms, and signal processing hardware.

**Nambirajan Seshadri** received the B.E. degree in electronics and communications engineering from the University of Madras, India, in 1982, and the M.S. and Ph.D. degrees in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1982 and 1986, respectively.

He is currently a member of the Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ, in the Signal Processing Research Department. His research interests include digital communications, combined source and channel coding, and multiaccess techniques.

**N. S. Jayant** (M'69-SM'77-F'82), for a photograph and biography, see this issue, p. 382.