A combined vector and scalar codebook for robust quantization of LPC parameters

Ravi P. Ramachandran[1], M. Mohan Sondhi[2], N. Seshadri[2] and B. S. Atal[2]
[1]Caip Center, Rutgers University, Piscataway, New Jersey, 08855
[2]AT&T Bell Laboratories, Murray Hill, New Jersey, 07974

ABSTRACT

An important problem in speech coding is the quantization of linear predictive coefficients (LPC) with the smallest possible number of bits while maintaining robustness to a large variety of speech material and transmission media. Since direct quantization of LPCs is known to be unsatisfactory, we consider this problem for an equivalent representation, namely, the line spectral frequencies (LSF). To achieve an acceptable level of distortion a scalar quantizer for LSFs requires a 36 bit codebook. We derive a 30 bit two-quantizer scheme which achieves a performance equivalent to this scalar quantizer. This equivalence is verified by tests on data taken from various types of filtered speech, speech corrupted by noise and by a set of randomly generated LSFs. The two-quantizer format consists of both a vector and a scalar quantizer such that for each input, the better quantizer is used. The vector quantizer is designed from a training set that reflects the joint density (for coding efficiency) and which ensures coverage (for robustness). The scalar quantizer plays a pivotal role in dealing better with regions of the space that are sparsely covered by its vector quantizer counterpart. A further reduction of 1 bit is obtained by formulating a new adaptation algorithm for the vector quantizer and doing a dynamic programming search for both quantizers. The method of adaptation takes advantage of the ordering of the LSFs and imposes no overhead in memory requirements. Subjective tests in a speech coder reveal that the 29 bit scheme produces equivalent perceptual quality to that when the parameters are unquantized.

Keywords: Robust, quantizer, codebook, line spectral frequencies, adaptation, dynamic programming

1 INTRODUCTION

A linear predictive (LPC) analysis [1] of a speech signal, based on the model that a speech sample is a weighted linear combination of p previous samples, results in the set of weights $a(i)$. These weights correspond to the direct form coefficients of a nonrecursive filter $A(z) = 1 + \sum_1^p a(i)z^{-i}$. Passing the speech signal through the filter $A(z)$ results in the removal of the near-sample correlations and produces an LPC residual. In addition, the magnitude spectrum of $1/A(z)$ describes the spectral envelope of the speech being analyzed. Therefore, transforming the speech into the LPC residual also has the frequency domain interpretation that the spectral envelope is removed.

In predictive speech coders, the LPC residual and LPC parameters $a(i)$ are quantized and coded for transmission. In this paper, we investigate the quantization of only the LPC parameters. The goal of LPC parameter quantization is to minimize the spectral distortion (SD) or the root mean-square deviation between the true log magnitude spectrum of $1/A(z)$ and the spectrum that results after quantization. Quantization of the parameters $a(i)$ is known not to be efficient and can lead to an unstable filter $1/A(z)$. We therefore transform the LPC coefficients to an equivalent set of parameters, namely, the line spectral frequencies (LSF) $f(i)$ [2, 3, 4, 5]. Line spectral frequencies have several properties that make them more suitable for quantization. They are approximately related to the formant frequencies

0-8194-1601-0/94/$6.00

and bandwidths and show a localized spectral sensitivity. Also, it is possible to define a tractable distortion measure in terms of LSFs, which is closely related to the SD. Finally, it is possible to quantize LSFs without destroying an ordering property which guarantees stability of $1/A(z)$. We have chosen the order of the LPC analysis to be $p = 10$, which gives us ten LSFs to quantize.

Several interrelated issues affect the design of a quantizer: the type of quantizer involved (scalar or vector), the distortion measure used, the inclusion or exclusion of memory in the quantizer, the search complexity, the codebook design, the desired number of bits, memory requirements for storing the codebook, robustness to a wide class of inputs and robustness to channel errors. A study of all these issues is beyond the scope of this paper. We focus our investigation on the issue of quantizer robustness to various filtering and noise conditions. Our objective is to realize a quantization scheme that results in a prescribed level of distortion, with the lowest possible number of bits, for a wide class of inputs, under the assumption that there are no channel errors.

2 ISSUES IN QUANTIZER DESIGN

The quantizer performance is evaluated by the spectral distortion (SD) which, in dB, is defined as

$$\mathrm{SD} = \sqrt{\frac{1}{B}\int_R \left[10\log\left(|A_q(e^{j2\pi f})|^2/|A(e^{j2\pi f})|^2\right)\right]^2 df} \tag{1}$$

where A and A_q refer to the filters resulting from the original and quantized parameters, respectively. Throughout our study, the speech was sampled at 8 kHz, the region R taken to be the frequency band from 125 to 3400 Hz and B taken to be the bandwidth represented by R. It is generally accepted that a quantizer may be considered transparent if its average spectral distortion (AVSD) is about 1 dB, the percentage of Type 1 outliers (SD from 2 to 4 dB) is less than 2 percent and there are no Type 2 outliers (SD greater than 4 dB) [6]. For the codebook design, we use a computationally more tractable measure than the SD. The metric is the weighted squared Euclidean distance given by

$$d(\mathbf{f},\hat{\mathbf{f}}) = \sum_{i=1}^{p} w(i)[f(i) - \hat{f}(i)]^2, \tag{2}$$

where $f(i)$ is the $i-th$ component of the original LSF vector $\mathbf{f}$ and $\hat{f}(i)$ is the corresponding component of the quantized vector $\hat{\mathbf{f}}$. The weights are chosen to be [7]

$$w(i) = \frac{1}{f(i) - f(i-1)} + \frac{1}{f(i+1) - f(i)}. \tag{3}$$

These weights add some emphasis to the formant frequencies thus making the metric d correspond better to the SD than the unweighted squared Euclidean distance. The codebook design method is based on the Linde-Buzo-Gray (LBG) algorithm with binary splitting [8] in which an input set of training data is used to determine the codewords such that the expected distortion is minimized. Note that in comparison to the SD, the determination of the Voronoi cells and the centroid computation is much simpler for the weighted squared Euclidean distance thereby making it computationally feasible.

Since the focus of our study is on robustness to various conditions, we examine the relationship of robustness to the type of quantizer and number of bits used. The advantage of vector quantizers over scalar quantizers is that they require fewer bits to achieve a given level of distortion by exploiting the multidimensional probability density and allowing for Voronoi regions of arbitrary convex shape [9]. Since the true probability density of the LSFs derived from all possible speech material is not known, a vector quantizer design is based on a large set of empirical training data. Due to the sampling variability of the training set, there is an incomplete description of the multidimensional probability density. Therefore, the vector quantizer is sensitive to the distribution of the data on which it is tested thereby diminishing its robustness. In contrast, a scalar quantizer is configured by designing codebooks for each dimension separately. It can adequately cover a designated multidimensional space. Therefore, it performs well even for data with a distribution

different from that of the training set. However, to achieve this robustness, the scalar quantizer requires more bits than a vector quantizer.

Our goal is to design a quantization scheme that compromises between the conflicting requirements of using few bits and robustness. A scalar quantizer that results in about a 1 dB AVSD for various test data serves as a benchmark. Then, we develop a quantizer with as few bits as possible and with a performance no worse than that of the scalar quantizer. The performance is evaluated in terms of AVSD and number of Type 1 and Type 2 outliers. Note that we attach considerable importance to reducing the number of outliers especially if the AVSD is already about 1 dB or less.

Our quantization scheme is memoryless, i.e., each vector is coded independently of past or future actions of the encoder or decoder [10]. Finally, search complexity and memory requirements are important practical considerations. Although we do not focus on these, neither will be made prohibitively large.

3 PARAMETER VARIATION

Since our aim is to get a robust quantizer, we first study the effects of various practically reasonable filtering conditions on the behavior of LSFs. This will give us information about the dynamic range of each LSF and the boundaries of the 10-dimensional space that have to be considered for the design. The LPC analysis is performed every 20 ms by the modified covariance method [11] with error weighting [12] by a 25 ms Hamming window, a high frequency compensation factor of 0.05 and a bandwidth expansion of 10 Hz. The speech is taken from the TIMIT database and the data comes from 518 different sentences spoken by 209 speakers.

The TIMIT database consists of speech sampled at 16 kHz. We study the effects of applying 3 different lowpass filters (with cutoff frequencies at 3600, 3400 and 3200 Hz, respectively) to the speech, followed by downsampling by 2. It is observed that as the cutoff frequency decreases, $f(9)$ and $f(10)$ shift downward and the rest remain essentially the same. After lowpass filtering to 3400 Hz and downsampling by 2, we study the effect of applying 3 additional filter weightings on LSF behavior. Applying a 300 Hz highpass filter is equivalent to examining the limitation of telephone bandwidth. In this case, the major influence is on $f(1)$ which shifts upward. Also $f(2)$ and $f(3)$ shift upward but to a lesser extent than $f(1)$. The other LSFs do not change. Processing speech by a bandpass filter that conforms with the Intermediate Reference Mask (IRS), which is a CCITT standard, has a different impact. The upward shift of the first 3 LSFs is more than with the highpass filter. The frequencies $f(4)$, $f(5)$, $f(6)$ and $f(7)$ shift upward but not as much as the first 3 LSFs. The last two LSFs shift downward while $f(8)$ remains the same. We do a z to $-z$ transformation on the IRS filter to get another bandpass filter to observe the effect of different spectral tilts on the LSFs. Compared to the standard IRS filter, this other bandpass filter causes a larger upward shift of $f(1)$ and $f(2)$, about the same shift for $f(3)$ and a smaller upward shift of $f(4)$. There are practically no shifts for $f(5)$ through $f(8)$. The downward shift of $f(9)$ is about the same and the downward shift in $f(10)$ is less than for the IRS filter.

The above observations suggest the following:

1. An acceptable vector quantizer designed on training data from just one filtering condition (e.g., the 3400 Hz lowpass filter) may not work well for inputs from another condition (e.g., the IRS filter) because of the shifting behavior of many LSFs.
2. Even two bandpass filters related by a simple frequency transformation but having different spectral tilts influence the joint LSF distribution in different ways.
3. Since the dynamic ranges of some individual LSFs are affected by highpass and bandpass filter weighting, the overall 10 dimensional space that has to be considered for codebook design is larger than the space described for LSFs derived from 3400 Hz lowpass filtered speech only. This indicates that more bits are needed to accommodate the filter weightings.

Differential line spectral frequencies (DLSF) are defined as follows: $\Delta(1) = f(1)$ and $\Delta(i+1) = f(i+1) - f(i)$ for $i = 1$

to $p-1$. These parameters (except $\Delta(1)$) have the empirically observed property that the filtering conditions described above do not change their dynamic ranges. Also, the standard deviations of the DLSFs are generally lower than those of the LSFs. In view of this, the DLSFs are more suited to quantization. However, DLSFs have potential advantages only for scalar quantization. Vector quantization of DLSFs yields no advantage over that of LSFs as discussed later.

4 DESCRIPTION OF SCALAR QUANTIZER

In order to assess the savings in bits offered by our approach, we first design a benchmark scalar quantizer for LSFs by the LBG method. The training data consists of 333,355 LSF vectors collected from the TIMIT database. They correspond to equal contributions from 5 conditions, namely, (1) the lowpass filter at 3400 Hz, (2) the lowpass filter at 3200 Hz, (3) the 300 Hz highpass filter, (4) the IRS filter and (5) the transformed IRS filter. Two lowpass filters are used to provide some emphasis to the relatively lower values of the first 3 LSFs. The weightings (3)-(5) are applied after lowpass filtering to 3400 Hz and downsampling by 2. Let $f_{min}(i)$ and $f_{max}(i)$ denote the smallest and largest values of the ith LSF determined from the 333,355 data points. The overall space S is described by the cross-product of the intervals $[f_{min}(i), f_{max}(i)]$ under the usual constraint of ordering. The training set consisting of the 333,355 data points leads to a scalar quantizer that accomplishes a coverage of the space S. The codebook for each $f(i)$ is designed independently of the others.

The performance of the quantizer is evaluated by five sets of test data as follows:

1. 14,780 vectors from the 5 filtering conditions but different from the training set.
2. 14,780 vectors from LPC analysis of speech corrupted by additive white Gaussian noise with an SNR of 15 dB.
3. 14,780 vectors from LPC analysis of speech corrupted by car noise (at a speed of 120 km/h) with an SNR of 15 dB.
4. 14,780 vectors from LPC analysis of speech corrupted by multitalker babble noise with an SNR of 30 dB.
5. 10,000 randomly generated LSF vectors.

For the first four items, 18 sentences of speech from 18 speakers (1 sentence for each speaker) from the TIMIT database are used. They are different from the ones used to derive the LSFs that train the quantizer. The noise is added to the original 16 kHz sampled speech and filtered in the same way as the speech such that the resulting SNR (as mentioned above) is obtained just prior to LPC analysis.

To generate the test data for item 5, the procedure is as follows. We first transform the training data vectors to log-area ratio vectors, and compute the mean and variance of each component. To get one random log-area ratio vector, we use a Gaussian random number generator for each component with the computed mean and variance . Finally, we transform this log-area ratio vector into the corresponding random LSF vector.

A 36 bit scalar quantizer achieves about a 1 dB AVSD for the test conditions. The bit allocation is (4,4,4,4,4,4,3,3,3,3). Table 1 shows the performance results. These results were obtained by doing a sequential search for the best codebook entry for each LSF separately such that ordering is preserved. Better search strategies are discussed later.

As seen from Table 1, the scalar quantizer works reasonably well on a variety of input vectors. The outliers are mainly due to clipping of one or more LSF components. For speech degraded by noise, a somewhat better performance is achieved than for LSFs obtained from noiseless speech. This is because additive noise generally moves each $f(i)$ towards a region in $[f_{min}(i), f_{max}(i)]$ that is more densely populated by the training data, and at the same time reduces its variance. The case of lowest variance and best performance is for speech with white noise. The poorest performance is, as expected, for random LSFs. However, even for these, the resulting AVSD is only slightly higher than for the LSFs obtained from the various filtering conditions.

Table 1: Performance results for 36 bit scalar quantizer

Condition	AVSD (dB)	Type 1 Outliers (%)	Type 2 Outliers (%)
Different filters	0.99	2.17	0.03
White noise	0.80	0.13	0
Car noise	0.92	1.10	0.01
Babble noise	0.96	1.62	0.02
Random LSFs	1.16	6.51	0.13

A 34 bit DLSF scalar quantizer achieves about the same performance as the 36 bit LSF quantizer for all the testing conditions. The bit allocation is (4,4,4,4,3,3,3,3,3,3). The sequential search strategy is as outlined in [4]. The first DLSF is $\Delta(1) = f(1)$. It is quantized to $\hat{\Delta}(1) = \hat{f}(1)$. Then, for $i = 1$ to $p-1$, the value

$$\Delta_v(i+1) = f(i+1) - \hat{f}(i) \tag{4}$$

is quantized to $\hat{\Delta}(i+1)$. The reconstructed LSFs are given by

$$\hat{f}(i+1) = \hat{f}(i) + \hat{\Delta}(i+1) \tag{5}$$

for $i = 1$ to $p-1$. The design of a globally optimum scalar quantizer based on dynamic programming (as opposed to a locally optimum solution provided by the LBG method) is dealt with in [13, 14].

5 USE OF VECTOR QUANTIZATION

In the previous section, we saw that scalar quantization of DLSFs saves about 2 bits compared to scalar quantization of LSFs, for the same performance. Also, as pointed out earlier, the various filtering conditions do not appear to alter the distribution of the DLSFs. It appears to be natural, therefore, to consider a vector quantizer for DLSFs. However, unless the DLSFs are quantized sequentially, (see equations (4) and (5)), quantization errors propagate to the higher LSF components and yield a large spectral distortion. Inspection of this sequential quantization procedure shows that, except for a renaming of the components, vector quantization of DLSFs reduces precisely to the vector quantization of LSFs. When the sequential procedure is used as part of the LBG algorithm, the codebook for the DLSFs can be shown to be equivalent to that for the LSFs. Thus, for vector quantization there is no advantage in going to DLSFs.

It is clear from our discussion in Sections 2 and 3 that robustness is not guaranteed for a vector quantizer even if the same training data is used as for a scalar quantizer. The reason is that the LBG algorithm (or any other reasonable algorithm for that matter) allocates large numbers of codevectors to regions of S that are densely populated by the training data, and very few to the sparse regions. This gives rise to a large spectral distortion whenever a *test* vector occurs in one of the sparse regions. This problem can be alleviated by configuring a training set with two components. The first component is the set of LSF vectors corresponding to the 5 filtering conditions discussed in Section 4. This data provides the empirical estimate of the probability density. The second component is a set of uniformly distributed vectors in S (which we will call a "uniform sheet"). This set increases robustness by covering the sparse regions of the first component. The relative proportion of vectors from these two components is determined experimentally to get the best performance.

It is, of course, not feasible to design a single codebook of more than 12 bits because of the prohibitively large memory and computational requirements. Two suboptimum approaches are the multistage vector quantizer [15, 16] and the split vector quantizer [6]. For both these methods, the codebooks at each stage have fewer bits than when using a single codebook. In the multistage technique, a given test vector is quantized with the first codebook. For the remaining stages, the quantization error from the previous stage is quantized with its particular codebook. The final quantized version of the test vector is obtained by summing the codevectors of all the stages. Note, however,

Table 2: Performance results for 32 bit vector quantizer using random LSFs

Fraction of uniform sheet	AVSD (dB)	Type 1 Outliers (%)	Type 2 Outliers (%)
0.0	1.18	9.27	0.16
0.1	1.16	5.28	0.01
0.2	1.18	3.99	0.0
0.3	1.18	3.12	0.0
0.4	1.20	3.07	0.01
0.5	1.22	2.85	0.01
0.6	1.23	2.69	0.0
0.7	1.26	2.72	0.0
1.0	1.49	6.36	0.0

Table 3: Performance of vector quantizers for LSFs obtained from speech

Condition	Lowest bit rate	Fraction of uniform sheet	AVSD (dB)	Type 1 Outliers (%)	Type 2 Outliers (%)
Different filters	27	0.0	1.02	1.31	0.0
White noise	28	0.0	0.82	0.07	0.0
Car noise	30	0.1	0.95	0.51	0.0
Babble noise	28	0.0	0.98	0.77	0.0

that the final quantized vector is not guaranteed to satisfy the ordering property. Also, it is not feasible to generate uniform sheets to adequately cover 10 dimensional regions. For LSF vectors, therefore, we are forced to use a special case of a multistage vector quantizer, called a split vector quantizer. Here, the codebook at each stage quantizes only a subvector. We use a (3,3,4) split for our 10 dimensional vectors. (i.e., the first stage quantizes (f_1, f_2, f_3), the second stage quantizes (f_4, f_5, f_6) and the last stage quantizes (f_7, f_8, f_9, f_{10})). Clearly with this scheme, it is possible to enforce ordering of the quantized vectors. The ordering property also allows us to use codebook adaptation which is discussed later. Finally, the reduced number of dimensions for each codebook makes it feasible to generate the uniform sheets mentioned in the previous paragraph.

We evaluate the performance of the different vector quantizers designed by using different proportions of the two training data components. A sequential search is used to find the best codebook entry for each subvector subject to the ordering constraint. By far the most stringent test condition is that of the random LSFs. Table 2 shows the results of experimentation with different proportions of the uniform sheet for a 32 bit quantizer for this test condition. The bit allocation for the three subvectors was chosen to be (11,10,11). ¿From Table 2, it is seen that a fraction of 0.6 for the uniform sheet results in the fewest number of outliers. In comparison to the 36 bit scalar quantizer, a matching AVSD and significantly fewer outliers are obtained for fractions of 0.2 and 0.3. For this condition, we clearly cannot go less than 32 bits.

Table 3 summarizes the results of similar experimentation with the other testing conditions. When tested on LSFs obtained from the 5 filtering conditions, additive white noise and babble noise, it turns out that the best performance is obtained when *no* uniform sheet is used. The criteria for transparent quantization (see Section 2) are met and the performance is no worse than that of the scalar quantizer. For the filtering conditions alone, we need only 27 bits. An additional bit is needed for white noise and babble noise. For the case of car noise, the best performance is obtained if 10 percent of the training data is from the uniform sheet. A 30 bit codebook is required.

To achieve the performance of the 36 bit scalar quantizer, we need 32 bits if we design the codebook in the manner outlined above. Of course, the procedure of introducing the uniform sheet may not be the best way of providing coverage of sparse regions of the training data. Indeed, as we show in the next section, a scheme using two quantizers accomplishes the coverage more efficiently.

Table 4: Performance results for 30 bit two-quantizer format using random LSFs

Fraction of uniform sheet	AVSD (dB)	Type 1 Outliers (%)	Type 2 Outliers (%)	Vector quantizer selected (%)
0.0	1.15	4.56	0.01	54
0.1	1.16	4.07	0.01	50
0.2	1.18	3.92	0.0	46
0.3	1.18	3.55	0.0	45
0.4	1.20	3.81	0.0	41
0.5	1.21	3.94	0.01	38
0.6	1.22	4.03	0.0	36
0.7	1.23	3.83	0.0	33
1.0	1.28	5.53	0.0	19

6 TWO-QUANTIZER FORMAT

Another way to improve coverage of sparse regions of the training data is to use *both* a vector quantizer *and* a scalar quantizer. The number of bits for each quantizer is the same and one additional bit specifies which quantizer is used. The encoding algorithm for each input vector is as follows:

1. Find the codeword for the vector quantizer that minimizes $d(\mathbf{f}, \hat{\mathbf{f}})$.
2. Find the codeword for the scalar quantizer that minimizes $d(\mathbf{f}, \hat{\mathbf{f}})$.
3. Select the codeword that results in the lowest SD.

The training data for the vector quantizer is as before. The 333,355 vectors obtained from the 5 filtering conditions are used to train the DLSF scalar quantizer.

We discuss next the performance of a 30 bit scheme (29 bits for each quantizer + 1 bit to specify the selected quantizer). For the vector quantizer, the bit allocation is (10,9,10). In the case of the scalar quantizer, the bit allocation is (3,3,3,3,3,3,3,3,3,2). By examining the quantized vectors, some benefits of this scheme are understood. The two quantizers can complement each other. The vector quantizer deals better with vectors whose components are clipped by the scalar quantizer. The scalar quantizer can be better for inputs in regions of the space that are otherwise sparsely covered by the vector quantizer codebook. Using two quantizers has the added advantage that the final selection can be based directly on the SD. Generally, this yields a slight reduction in the SD and can reduce the number of Type 1 outliers by 0.5 percent.

Table 4 shows the performance of this scheme on the random LSFs as the testing condition. The 30 bit two-quantizer scheme outperforms the 36 bit scalar quantizer even when no uniform sheet is used. However, using a sheet (fraction of 0.3) leads to the fewest number of outliers and about the same AVSD as the 36 bit scalar quantizer. The next best situation is when the fraction of the sheet is 0.2. When the fraction of the uniform sheet is 0.2 or 0.3, the 30 bit scheme is comparable to a 32 bit vector quantizer. It is interesting to note from the last column of Table 4 that the scalar quantizer is chosen for quite a large percentage of the test data.

Table 5 shows the performance for the other testing conditions when the uniform sheet comprises 20 percent and 30 percent of the training data. It is seen that the two conditions are almost equivalent. The fraction 0.2 is better for most of the conditions. For both the fractions, the criteria for transparent quantization are met. Compared to the 36 bit scalar quantizer, the number of outliers are much less at the expense of a higher AVSD. However, the latter difference is not as significant since the AVSD is already 1 dB or less. specified criteria.

Table 5: Performance of 30 bit two-quantizer format for LSFs obtained from speech. The uniform sheet constitutes 20 or 30 percent of the training data for the vector quantizer.

Condition	Fraction of uniform sheet	AVSD (dB)	Type 1 Outliers (%)	Type 2 Outliers (%)	Vector quantizer selected (%)
Different filters	0.2	1.01	0.64	0.0	67
Different filters	0.3	1.03	0.76	0.0	62
White noise	0.2	0.86	0.01	0.0	73
White noise	0.3	0.89	0.05	0.0	68
Car noise	0.2	0.99	0.77	0.0	61
Car noise	0.3	1.00	0.38	0.0	58
Babble noise	0.2	0.99	0.45	0.0	65
Babble noise	0.3	1.01	0.54	0.0	60

7 CODEBOOK ADAPTATION

The investigation continues by attempting to further lower the bit rate without sacrificing performance. Therefore, we no longer do a comparison with the 36 bit scalar quantizer. Rather, the results in Tables 4 and 5 serve as the benchmark. The (3,3,4) split vector quantization described above does not take advantage of any dependence among the subvectors. As far as the design of the codebooks is concerned, it is not easy to take advantage of this dependence. The codebook for each subvector has to be designed independently of the others because a joint design is not feasible. However, in the quantization of a given vector it is possible to capitalize on the dependence. The basic idea is as follows. Suppose the ith subvector is quantized to $\hat{\mathbf{f}}_i$. Then the ordering property of LSFs restricts the region in which the $(i+1)$st quantized vector $\hat{\mathbf{f}}_{i+1}$ may lie. Rather than restricting the search of the $(i+1)$st codebook to the allowed region, the entire codebook may be mapped to that region, thus providing a finer sampling of it. Note that if the mapping is completely specified by the quantized vector $\hat{\mathbf{f}}_i$, there is no cost in terms of bits. Also, since the mapping starts with one fixed codebook, there is no overhead in terms of memory requirements. This idea, called codebook adaptation, was first proposed for a scalar quantizer in [14]. Our method is a multidimensional generalization of this approach. The complete procedure is outlined below.

Recall that the ith LSF is in the interval $[f_{min}(i), f_{max}(i)]$ (see Section 4). Suppose there are N subvectors $\mathbf{f}_i$ of dimension d_i to be quantized by N corresponding codebooks C_i. Then, the quantized vector is $\hat{\mathbf{f}} = [\hat{\mathbf{f}}_1 \mid \hat{\mathbf{f}}_2 \mid \cdots \mid \hat{\mathbf{f}}_N]$. The subvector $\mathbf{f}_1$ is quantized to $\hat{\mathbf{f}}_1 = [\hat{f}(1)\ \hat{f}(2)\ \cdots\ \hat{f}(d_1)]$ without any adaptation. Let the jth entry of C_2 be $\hat{\mathbf{f}}_{j,2} = [\hat{f}_j(d_1{+}1)\ \hat{f}_j(d_1{+}2)\ \cdots\ \hat{f}_j(d_1{+}d_2)]$. For each j, it must be ensured that $\hat{f}_j(d_1{+}1) > \hat{f}(d_1)$. If $\hat{f}(d_1) < f_{min}(d_1{+}1)$, this naturally holds and no adaptation is performed. Otherwise, we transform codebook C_2 to C_2^t such that C_2^t covers a smaller space that is, in general, not similar in shape to the space covered by C_2. Our transformation maps each component of the jth entry separately. The mapped first component $\hat{f}_j^t(d_1+1)$ is given by

$$\hat{f}_j^t(d_1+1) = \frac{[(\hat{f}_j(d_1+1) - f_{min}(d_1+1)][f_{max}(d_1+1) - \hat{f}(d_1)]}{f_{max}(d_1+1) - f_{min}(d_1+1)} + \hat{f}(d_1) \tag{6}$$

Note that this is the mapping used in [14] for the scalar quantizer. Successive components $\hat{f}_j(d_1+r)$ for $r = 2$ to d_2 are sequentially mapped as outlined below. Regarding the original codebook C_2, let

$$f_m(d_1+r) = \begin{cases} f_{min}(d_1+r) & \text{if } f_{min}(d_1+r-1) \le \hat{f}_j(d_1+r-1) \le f_{min}(d_1+r) \\ \hat{f}_j(d_1+r-1) & \text{if } \hat{f}_j(d_1+r-1) > f_{min}(d_1+r) \end{cases} \tag{7}$$

For the transformed codebook C_2^t, let

$$f_m^t(d_1+r) = \begin{cases} f_{min}(d_1+r) & \text{if } f_{min}(d_1+r-1) \le \hat{f}_j^t(d_1+r-1) \le f_{min}(d_1+r) \\ \hat{f}_j^t(d_1+r-1) & \text{if } \hat{f}_j^t(d_1+r-1) > f_{min}(d_1+r) \end{cases} \tag{8}$$

Table 6: Performance of 30 bit two-quantizer format with adaptation. The uniform sheet constitutes 20 or 30 percent of the training data for the vector quantizer.

Condition	Fraction of uniform sheet	AVSD (dB)	Type 1 Outliers (%)	Type 2 Outliers (%)	Vector quantizer selected (%)
Different filters	0.2	0.99	0.39	0.0	70
Different filters	0.3	1.01	0.42	0.0	66
White noise	0.2	0.85	0.01	0.0	77
White noise	0.3	0.87	0.02	0.0	72
Car noise	0.2	0.98	0.64	0.0	64
Car noise	0.3	0.99	0.21	0.0	60
Babble noise	0.2	0.97	0.27	0.0	68
Babble noise	0.3	0.99	0.26	0.0	63
Random LSFs	0.2	1.14	2.70	0.0	53
Random LSFs	0.3	1.14	2.31	0.0	52

Then, the mapping is given by

$$\hat{f}_j^t(d_1+r) = \frac{[(\hat{f}_j(d_1+r) - f_m(d_1+r)][f_{max}(d_1+r) - f_m^t(d_1+r)]}{f_{max}(d_1+r) - f_m(d_1+r)} + f_m^t(d_1+r) \tag{9}$$

Note the dependence on the previous component of the original codeword and the mapped codeword. This is needed to maintain the ordering property of the entries of C_2^t. The next subvector $\mathbf{f_2}$ is quantized to $\hat{\mathbf{f}}_2$ using C_2^t. Then, C_3 is mapped in a similar fashion as outlined above for the quantization of $\mathbf{f_3}$. The process continues until all the N subvectors are quantized.

In implementing the codebook transformation, we have considered two options. Option 1 is to map every codeword of C_i ($i = 2$ to N). This is based on the heuristic claim that the conditional probability density $p[f(\sum_{k=1}^{i-1} d_k + 1), \cdots, f(\sum_{k=1}^{i} d_k) | \hat{f}(\sum_{k=1}^{i-1} d_k)]$ is similar in shape to the probability density $p[f(\sum_{k=1}^{i-1} d_k + 1), \cdots, f(\sum_{k=1}^{i} d_k)]$. Option 2 is to keep the codewords of C_i that maintain the ordering property intact and only map the other codewords. This guarantees a lower distortion for any input vector if $N = 2$. Experiments show that for our vector quantizers in which $N = 3$, the performance is much better if Option 2 is invoked.

Consider the case of a 30 bit vector quantizer alone in which the fraction of the uniform sheet is either 0.2 or 0.3. For LSFs taken from speech, the AVSD consistently decreases by about 0.02 dB. Depending on the quantizer and the testing condition, the ratio of the number of outliers with and without adaptation ranges from 0.33 to 0.91 (average value is 0.57). In the case of random LSFs, the AVSD decreases by 0.09 dB and there are 0.58 as many outliers. Table 6 gives the results for a 30 bit two-quantizer system with adaptation. The consequences of adaptation are more apparent in terms of reducing outliers than in diminishing the AVSD. Note that although adaptation improves performance, the number of bits is not lowered.

8 DYNAMIC PROGRAMMING SEARCH

The results generated so far are obtained by a sequential search. For a split vector quantizer, a sequential search yields the optimal codevector if the subvectors are independent. However, for ordered data (like LSFs), where $\hat{\mathbf{f}}_{i+1}$ depends on $\hat{\mathbf{f}}_i$, this is not true. The search is suboptimal even with codebook adaptation, because the transformed codebook C_{i+1}^t is different for each entry of C_i^t. We use a dynamic programming technique called the A^* algorithm [18] to get the optimal codevector for a given split vector quantizer. The A^* algorithm can be used with or without codebook adaptation. We illustrate the technique when codebook adaptation is used.

1. Initialization: We have a split vector quantizer with N codebooks C_i each having a size n_i and dimension

Table 7: Performance of 29 bit two-quantizer format with adaptation and hybrid search technique. The uniform sheet constitutes 20 or 30 percent of the training data for the vector quantizer.

Condition	Fraction of	AVSD (dB)	Type 1	Type 2	Vector quantizer
Different filters	0.3	1.04	0.32	0.0	56
White noise	0.2	0.88	0.01	0.0	71
White noise	0.3	0.91	0.01	0.0	66
Car noise	0.2	1.02	0.51	0.0	55
Car noise	0.3	1.02	0.21	0.0	51
Babble noise	0.2	1.01	0.21	0.0	58
Babble noise	0.3	1.02	0.25	0.0	53
Random LSFs	0.2	1.15	2.35	0.0	41
Random LSFs	0.3	1.15	1.82	0.0	41

d_i. The input vector is $\mathbf{f} = [f(1)\ f(2)\ \cdots\ f(p)]$ where $p = \sum_i^N d_i$. The subvectors of $\mathbf{f}$ are denoted by $\mathbf{f_i}$. The jth entry of C_i is $\hat{\mathbf{f}}_{\mathbf{j,i}} = [\hat{f}_j(\sum_{r=1}^{i-1} d_r + 1)\ \cdots\ \hat{f}_j(\sum_{r=1}^{i} d_r)]$. The corresponding entry of C_i^t is $\hat{\mathbf{f}}^{\mathbf{t}}_{\mathbf{j,i}} = [\hat{f}_j^t(\sum_{r=1}^{i-1} d_r + 1)\ \cdots\ \hat{f}_j^t(\sum_{r=1}^{i} d_r)]$.

2. Quantize the first subvector $\mathbf{f_1}$ with every entry of C_1. Arrange the results in a stack. The jth element of the stack corresponds to the jth entry of C_1. The path length is $p(j) = 1$, the accumulated distortion is $d_q(j) = \sum_{i=1}^{d_1} w(i)[f(i) - \hat{f}_j(i)]^2$ and the index matrix entry is $I(j,1) = j$. The number of stack elements is $n_{\mathrm{tot}} = n_1$.

3. Given n_{tot} stack elements, find the stack element k corresponding to the smallest accumulated distortion subject to a positive path length.

4. If $p(k) = N$, the algorithm terminates as the encoding is complete.

5. Now, $p(k) = m < N$. Depending on the quantized subvector from the transformed codebook C_m^t, adapt C_{m+1} to C_{m+1}^t. Quantize $\mathbf{f_{m+1}}$ using each entry of C_{m+1}^t. Accumulate the stack with n_{m+1} entries. For these entries ($l = n_{\mathrm{tot}} + 1$ to $n_{\mathrm{tot}} + n_{m+1}$ corresponding to the indices $j = 1$ to n_{m+1}), $p(l) = m+1$, $I(l,i) = I(k,i)$ for $i = 1$ to m, $I(l, m+1) = j$. The accumulated distortion is $d_q(l) = d_q(k) + \sum_{i=u}^{v} w(i)[f(i) - \hat{f}_j^t(i)]^2$ where $u = \sum_{r=1}^{m} d_r + 1$ and $v = \sum_{r=1}^{m+1} d_r$. Update the value of n_{tot}. Set $p(k) = 0$. Go back to step 3.

The $A^\star$ method has been successfully applied to the scalar quantization of DLSFs [19]. However, due to the mismatch between the weighted squared Euclidean distance and the SD, a hybrid scheme is suggested in [19]. For any input vector, there are two possible codevectors, one from a sequential search and one from an $A^\star$ search. The codevector that yields the smallest SD is chosen. We use the hybrid technique for both the vector and scalar quantizers to select the best codevector in terms of SD among four possibilities. The first two are the codewords from the vector quantizer resulting from a sequential and $A^\star$ search. The same procedure is repeated for the scalar quantizer to get the other two possibilities. Table 7 shows the results for the resulting 29 bit scheme. For the vector quantizer, the bit allocation is (10,9,9). In the case of the scalar quantizer, the bit allocation is (3,3,3,3,3,3,3,3,2,2).

Compared to a 30 bit system with no adaptation (see Tables 4 and 5), the AVSD is about the same and the number of outliers are reduced. The $A^\star$ algorithm offers much more of an improvement for the scalar quantizer than for the vector quantizer. This leads to an increased usage of the scalar quantizer for all the testing conditions as seen in Tables 6 and 7. Transparent quantization is achieved by a 29 bit system for LSFs obtained from speech. The results are generally about the same whether we use a 20 percent or 30 percent uniform sheet to train the vector quantizer. However, note that the 30 percent sheet is better for random LSFs and can bring the number of outliers below 2 percent. The combination of codebook adaptation and the $A^\star$ algorithm results in a savings of 1 bit.

9 SUMMARY AND CONCLUSIONS

In this paper, we have proposed a 30 bit quantization scheme that is robust to a wide variety of inputs. The performance is at least equal to that of a 36 bit scalar quantizer for various test conditions that include a set of randomly generated LSFs. The scheme is based on a two-quantizer format involving both a vector and a scalar quantizer. For each input, the quantizer that achieves the lower distortion is used.

The vector quantizer is designed with training data that has two components. The first component, consisting of LSFs derived from speech passed through various filters in common use, provides coding efficiency. The second component, consisting of a set of vectors uniformly distributed over the space S described by the LSFs in the first component, provides robustness. The scalar quantizer is designed with training data from the first component only. The two key factors in achieving robustness are our configuration of the training set for the vector quantizer, and the option of using the scalar quantizer. The scalar quantizer plays an important role by dealing better with regions of S that are sparsely covered by the vector quantizer codebook.

A further savings of 1 bit is obtained by codebook adaptation and a dynamic programming search. We present a new codebook adaptation scheme for the split vector quantizer that is a generalization of a method for scalar quantizers. The adaptation is based on transforming the codebook for a particular subvector in a manner that depends on the previously quantized subvector and is such that ordering is preserved. This reduces the SD without an increase in bits and without additional memory requirements. A dynamic programming search using the A^* algorithm alleviates the suboptimality of a sequential search. Although more complex than the sequential search, the A^* method is much more efficient than a prohibitive exhaustive search.

To evaluate our 29 bit quantizer subjectively, we used a CELP coder [20] to code several sentences with quantized as well as unquantized LPC parameters. The experiments showed that the output speech signals obtained with and without quantization are indistinguishable to the ear.

10 ACKNOWLEDGEMENTS

The authors thank Peter Kroon for supplying the software to run the CELP coder and for furnishing the quantizer tables for the adaptive and stochastic codebook gains.

References

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

[2] F. Itakura, "Line spectrum representation of linear predictor coefficients of speech signals", *J. Acoust. Soc. Amer.*, vol. 57, S35(A), 1975.

[3] H. Wakita, "Linear prediction voice synthesizers: Line spectrum pairs (LSP) is the newest of several techniques", *Speech Technology*, Fall 1981.

[4] F. K. Soong and B.-H. Juang, "Line spectrum pair (LSP) and speech data compression", *IEEE Int. Conf. on Acoust., Speech and Signal Processing*, San Diego, California, pp. 1.10.1–1.10.4, March 1984.

[5] G. S. Kang and L. J. Fransen, "Application of line spectrum pairs to low bit rate speech encoders", *IEEE Int. Conf. on Acoust., Speech and Signal Processing*, Tampa, Florida, pp. 7.3.1–7.3.4, April 1985.

[6] K. K. Paliwal and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame", *IEEE Trans. on Speech and Audio Processing*, vol. 1, pp. 3–14, Jan. 1993.

[7] R. Laroia, N. Phamdo and N. Farvardin, "Robust and efficient quantization of speech LSP parameters using structured vector quantizers", *IEEE Int. Conf. on Acoust., Speech and Signal Processing*, Toronto, Canada, pp. 641–644, May 1991.

[8] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design", *IEEE Trans. on Comm.*, vol. COM-28, pp. 84–95, Jan. 1980.

[9] J. Makhoul, S. Roucos and H. Gish, "Vector quantization in speech coding", *Proc. IEEE*, vol. 73, pp. 1551–1558, Nov. 1985.

[10] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, Kluwer Academic Publishers, 1992.

[11] B. S. Atal and M. R. Schroeder, "Predictive coding of speech signals and subjective error criteria", *IEEE Trans. on Acoust., Speech and Signal Processing*, vol. ASSP-27, pp. 247–254, June 1979.

[12] S. Singhal and B. S. Atal, "Improving performance of multi-pulse LPC coders at low bit rates", *IEEE Int. Conf. on Acoust., Speech and Signal Processing*, San Diego, California, pp. 1.3.1–1.3.4, March 1984.

[13] F. K. Soong and B.-H. Juang, "Optimal quantization of LSP parameters", *IEEE Trans. on Speech and Audio Processing*, vol. 1, pp. 15–24, Jan. 1993.

[14] N. Sugamara and N. Farvardin, "Quantizer design in speech analysis-synthesis", *IEEE J. on Sel. Areas in Comm.*, vol. 6, pp. 432–440, Feb. 1988.

[15] W.-Y. Chan, S. Gupta and A. Gersho, "Enhanced multistage vector quantization by joint codebook design", *IEEE Trans. on Comm.*, vol. 40, pp. 1693–1697, Nov. 1992.

[16] W. P. LeBlanc, V. Cuperman, B. Bhattacharya and S. A. Mahmoud, "Efficient search and design procedures for robust multi-stage VQ of LPC parameters for 4 kb/s speech coding", *IEEE Trans. on Speech and Audio Processing*, vol. 1, pp. 373–385, Oct. 1993.

[17] E. Paksoy, W.-Y. Chan, and A. Gersho, "Vector quantization of speech LSF parameters with generalized product codes", *Int. Conf. on Spoken Lang. Proc.*, Banff, Canada, pp. 33–36, Oct. 1992.

[18] N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, 1971.

[19] F. K. Soong and B.-H. Juang, "Optimal quantization of LSP parameters using delayed decisions", *IEEE Int. Conf. on Acoust., Speech and Signal Processing*, Albuquerque, New Mexico, pp. 185–188, April 1990.

[20] P. Kroon and K. Swaminathan, "A high-quality multirate real-time CELP coder", *IEEE J. on Sel. Areas in Comm.*, vol. 10, pp. 850–857, June 1992.