A Low Delay 16 kb/s Speech Coder

Vasu Iyengar and Peter Kabal, *Member, IEEE*

Abstract—**This paper studies a speech coder using a tree code generated by a stochastic innovations tree and backward adaptive synthesis filters. The synthesis configuration uses a cascade of two all-pole filters: a pitch (long time delay) filter followed by a formant (short time delay) filter. Both filters are updated using backward adaptation. The formant predictor is updated using an adaptive lattice algorithm. The multipath (M, L) search algorithm is used to encode the speech. A frequency weighted error measure is used to reduce the perceptual loudness of the quantization noise. The speech coder has low delay (1 ms) and subjective tests show that the speech quality is equally preferred to 7-b log-PCM.**

I. Introduction

THIS paper addresses the problem of low delay coding of speech signals at an encoding bit rate of 16 kb/s. Such a coder has applications in the switched telephone network. Encoding delay is an important consideration in the design of speech coding algorithms for use as part of a terrestrial common carrier network, due to the problem of disturbing echoes generated at the hybrid interface between two-wire and four-wire lines. In the presence of echoes, the perceptual degradation of the transmitted speech increases in severity as the round-trip transmission delay increases. Both the encoding delay and the propagation time contribute to the round-trip delay. Traditionally, this problem has been combated using echo control techniques. The amount of echo suppression required for satisfactory performance increases with the round-trip delay. The approach taken here is to avoid the echo control problem by using coding techniques with a small encoding/decoding delay. If end-to-end links are not entirely digital, tandem coding/decoding processes are necessary. Transmission across digital links may also involve several stages of digital transcodings to and from existing log-PCM techniques. Hence it is in general desirable to keep single stage encoding delays as low as possible. Also, it is important that the coder achieve toll quality for widespread acceptability as part of the switched telephone network.

Traditionally, toll quality coders have been waveform coders. Among waveform coders are the log-PCM coders, and the more complex ADPCM coding techniques. ADPCM schemes employ an adaptive predictor and an adaptive quantizer matched to the short-term modes of stationarity of the input speech. These strategies therefore exhibit improved performance over PCM schemes at the expense of greater complexity. The CCITT has formally approved an ADPCM coding algorithm that provides toll quality speech at 32 kb/s [1]. This coding algorithm utilizes backward adaptation schemes for the predictor and quantizer, i.e., the adaptations are based on an analysis of the past quantized data. This data is also available at the decoder, and therefore the receiver does not require side information. The use of backward adaptation allows for very low encoding delay. In contrast, in coders utilizing forward adaptation, the predictor and quantizer parameters are determined from an analysis of the coder input data. Since the analysis usually uses a 16–20 ms block of data, the input data has to be buffered. This buffering results in a large encoding delay.

If the encoding rate of the CCITT algorithm is reduced much below 32 kb/s, the quality produced drops off significantly. This occurs because of the interaction between the predictor update algorithm and the quantizer. When the quantizer noise increases, the performance of the predictor (which is determined by the reconstructed samples) drops, further increasing the quantization noise effects. Most previous attempts at high quality coding at rates near 16 kb/s have therefore used forward adaptation for the predictor and quantizer. However, this strategy tends to introduce large amounts of coding delay. Also, the receiver has to rely on side information to keep track of the evolution of the predictor and quantizer. This necessitates more complex framing schemes for correct multiplexing of the quantizer output and side information bit streams, thus increasing overall coder complexity.

A major advance towards improving the performance of waveform coders at low bit rates comes with the use of multipath tree search algorithms with differential waveform coders. A characteristic of differential coders is that the possible quantized output sequences are arranged in the form of a tree code. The quantizer and predictor pair essentially plays the role of a tree code generator. Encoding in conventional schemes proceeds by a single path search of this code tree to find the best output sequence. This has been identified as being a clear shortcoming of conventional DPCM and ADPCM. Much can be gained with the use of delayed decision schemes which employ multipath searches to make more efficient use of the tree code. The delays are usually of the order of a few samples, and can be kept within network echo delay constraints.

Manuscript received November 19, 1989; revised May 15, 1990.

V. Iyengar is with the Department of Electrical Engineering, McGill University, Montreal, Que., H3A 2A7 Canada.

P. Kabal is with the Department of Electrical Engineering, McGill University, Montreal, Que., H3A 2A7 and INRS-Telecommunications, Université du Québec, Verdun, Que., H3E 1H6 Canada.

IEEE Log Number 9042727.

Tree coding involves two basic issues. The first is the choice of an effective code tree, and the second involves the choice of a search algorithm to search the code tree for the output sequence that best matches the input. Multipath tree searching of codes generated by a fixed deterministic quantizer and a fixed predictor was first studied by Anderson and Bodie [2]. Jayant and Christensen [3] have studied the use of multipath searching of a code generated by a backward adaptive quantizer and a fixed predictor. Although delayed decision coding in the above studies provides gains in terms of both perceived speech quality and measurable signal-to-noise ratio over conventional differential encoders at a rate of 16 kb/s, the output speech quality was still reported to be characterized by easily perceived quantization noise. This is due to two main shortcomings. First, the encoding algorithm used fixed and not adaptive prediction. Second, the tree code was a deterministic tree code. Much can be gained with the use of so-called stochastic tree codes as will be shown later. Multipath tree searching with forward adaptive prediction has been studied in [4]–[6]. Unfortunately, forward adaptation of the predictor usually entails a large amount of encoding delay, of the order of 10–20 ms. Recently, Gibson and Haschke [7] have studied deterministic code trees with backward adaptive formant synthesis filters, but without a pitch loop.

The scope of this paper is as follows. First, a generalized predictive coder is briefly described. This configuration allows the use of a frequency weighted error measure. Subsequent sections then view this coding scheme in the more general tree coding context. The predictive coding scheme is then extended to the multipath search case with a stochastically populated innovations tree. Multipath searching is done with the (M, L) algorithm [8]. Both objective and subjective test results of the coding algorithm are presented.

II. Generalized Predictive Coder

The block diagram of a generalized predictive coder with a short-term or formant predictor is shown in Fig. 1 [9]. This configuration allows for adaptive adjustment of the noise spectrum in relation to the speech spectrum. The shape of the noise spectrum in relation to the speech spectrum is important from the point of view of perceived distortion in the output speech. Noise in the formant regions is partially or totally masked by the speech signal, since the speech power is high in the formant regions. The perceived noise in the output speech therefore comes from noise in those frequency ranges where the signal level is low. In Fig. 1 $F(z)$ and $N(z)$ are given by

$$F(z) = \sum_{i=1}^{P} a_i z^{-i}, \qquad N(z) = \sum_{i=1}^{P} b_i z^{-i}. \tag{1}$$

For the configuration shown in the Fig. 1, the spectrum of the reconstruction error is given by

$$S(z) - \hat{S}(z) = Q(z)\,\frac{1 - N(z)}{1 - F(z)} \tag{2}$$

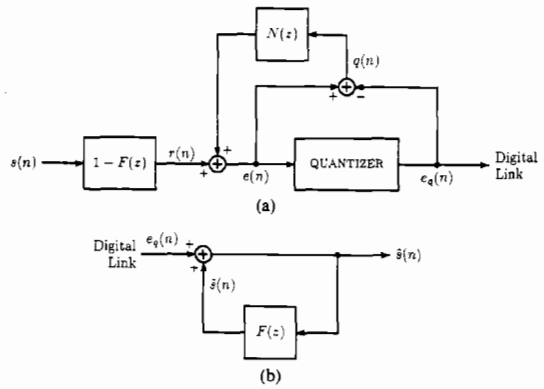


Fig. 1. Generalized predictive coder. (a) Encoder. (b) Decoder.

where the quantization error spectrum is given by $Q(z)$. With the usual assumptions of uncorrelated quantization noise, the quantization error has a flat power spectral density. The shape of the reconstruction error spectrum can be controlled by choosing $N(z)$ appropriately. It is usual to choose $N(z)$ as a bandwidth expanded version of $F(z)$, i.e., $N(z) = F(z/\mu)$, where $0 \le \mu \le 1$. A value of $\mu = 1$ gives a white reconstruction error spectrum, while $\mu = 0$ gives an error spectrum which has the same shape as the signal spectrum. The value of μ is usually chosen to be between 0.75 and 0.9. Such a value of μ has the effect of decreasing the noise power in the valleys (regions between the formants) of the speech spectral envelope, and increasing the noise power in the formant regions (where it is more tolerable).

Fig. 2 includes a long-term or pitch predictor, in addition to a formant predictor. The use of pitch prediction is motivated by the fact that voiced speech segments exhibit considerable similarity between adjacent pitch periods. Even in unvoiced segments, some additional prediction gain can be obtained with the use of a pitch predictor. In this work, a 3-tap pitch predictor is used, having the system function of

$$P(z) = \beta_1 z^{-M_p+1} + \beta_2 z^{-M_p} + \beta_3 z^{-M_p-1} \tag{3}$$

where M_p is the estimated pitch period in samples.

Adaptation of pitch predictors is necessary, since both the pitch lag and the predictor coefficients have to be fine tuned to the analysis segment. Pitch prediction is conventionally achieved with forward adaptation. The use of backward adaptation is considered here.

A. Formant Filter Adaptation

Since the signal spectrum is time varying, the coefficients of the filter $F(z)$ must be time varying also, to obtain a small prediction error variance. The update algorithm used for the formant predictor in this work is an adaptive lattice algorithm [10]. The prediction error filter $1 - F(z)$ can be formulated as a lattice filter as shown in Fig. 3. The reflection coefficients K_m are allowed to vary with time to track the modes of stationarity of the input

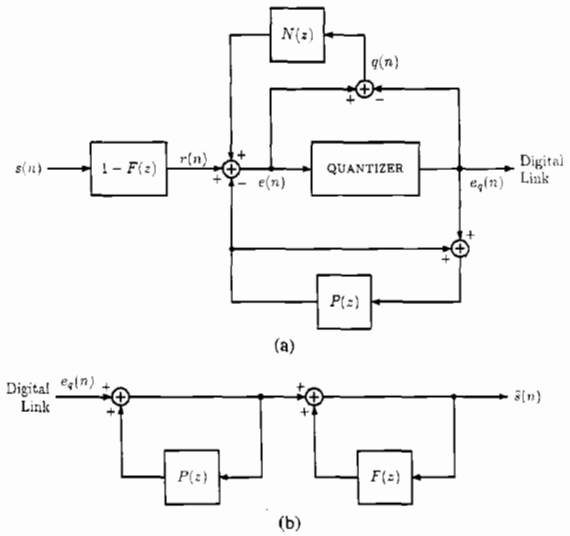


Fig. 2. Generalized predictive coder with pitch prediction. (a) Encoder. (b) Decoder.

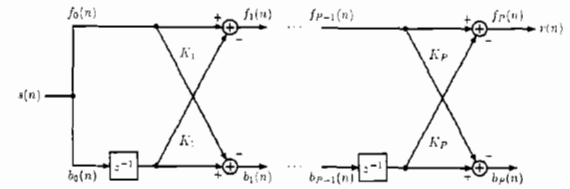


Fig. 3. Lattice filter of order P.

$s(n)$. The reflection coefficients K_m are therefore a function of time n, and shown explicitly by writing $K_m(n)$, $m = 1, 2, \cdots, P$. The update method is based on the minimization of a weighted error of the form

$$E_m(n) = \sum_{k=-\infty}^{n} w(n-k)\, e_m^2(k) \tag{4}$$

where $e_m^2(k)$ is a sum of forward and backward residual energies

$$e_m^2(k) = (1-\gamma) f_m^2(k) + \gamma b_m^2(k), \qquad 0 \le \gamma \le 1 \tag{5}$$

and $w(n)$ is a causal window function. Minimizing $E_m(n)$ with respect to $K_m(n)$ yields the updated value for the reflection coefficient

$$K_m(n+1) = \frac{\sum_{k=-\infty}^{n} w(n-k) f_{m-1}(k) b_{m-1}(k-1)}{\sum_{k=-\infty}^{n} w(n-k)[\gamma f_{m-1}^2(k) + (1-\gamma) b_{m-1}^2(k-1)]} = \frac{C_m(n)}{D_m(n)}. \tag{6}$$

The sufficient conditions for stability of the synthesis filter, $1/(1 - F(z))$ are that 1) $\gamma = 0.5$ and 2) $w(n) \ge 0$ for $n \ge 0$ [10].

The window $w(n)$ is used to form a weighted sum of a function of the forward and backward residual energies at each stage of the lattice. The shape of the window $w(n)$ should be such that the residual energy over the immediate past is weighted more than the residual energy over the more distant past. This ensures that the predictor evolves in accordance with the changing modes of stationarity of the input speech. The time frame over which the error is minimized is also an important design factor. A time frame that is too long results in averaging over regions where the characteristics of the signal are changing, whereas a time frame that is too short will result in a filter that is too finely tuned to the signal in the backward window, this being inappropriate for predicting the signal which follows the window. Important parameters in choosing a window are therefore the shape of the window, and the effective length of the window. A third consideration in choosing a window is the effect on the computational complexity of the update algorithm. A way of reducing computational complexity in the update procedure is to use windows $w(n)$ that can be considered as the impulse response of a causal finite order recursive digital filter. The use of such windows leads to the possibility of obtaining recursive update equations for $C_m(n)$ and $D_m(n)$ in (6). In general, if $W(z)$ is of the form

$$W(z) = \frac{1 - \sum_{i=1}^{N_z} \alpha_i z^{-i}}{1 - \sum_{i=1}^{N} \beta_i z^{-i}} \tag{7}$$

then $C_m(n)$ satisfies the following recursive relationship:

$$C_m(n) = \sum_{i=1}^{N} \beta_i C_m(n-i) - \sum_{i=1}^{N_z} \alpha_i f_{m-1}(n-i) b_{m-1}(n-1-i) + f_{m-1}(n) b_{m-1}(n-1) \tag{8}$$

and $D_m(n)$ satisfies

$$D_m(n) = \sum_{i=1}^{N} \beta_i D_m(n-i) - \sum_{i=1}^{N_z} \alpha_i [\gamma f_{m-1}^2(n-i) + (1-\gamma) b_{m-1}^2(n-1-i)] + [\gamma f_{m-1}^2(n) + (1-\gamma) b_{m-1}^2(n-1)]. \tag{9}$$

Windows that have desirable shape and length characteristics are considered in later sections.

B. Pitch Filter Adaptation

The pitch predictor adaptation scheme is a nonrecursive procedure operating on past quantized formant residual samples. Adaptation of the pitch predictor requires both the pitch lag and pitch coefficients to be updated. Conventionally, pitch prediction uses forward adaptation. In this work, we deviate from this practice and employ a novel backward adaptive pitch filter. The update is based

on the covariance formulation of linear prediction [11]. Minimizing the mean-square error over a frame of length N samples results in the following system of linear equations:

$$\sum_{n=1}^{N} r(n)r(n - M_p + 2 - i) = \sum_{j=1}^{3} \beta_j \sum_{n=1}^{N} r(n - M_p + 2 - i)r(n - M_p + 2 - j) \tag{10}$$

for $i = 1, 2$, and 3. The sequence $r(n)$ is a formant predicted residual signal. This above equation can be written in matrix form as

$$\begin{bmatrix} \phi(M_p - 1, M_p - 1) & \phi(M_p - 1, M_p) & \phi(M_p - 1, M_p + 1) \\ \phi(M_p, M_p - 1) & \phi(M_p, M_p) & \phi(M_p, M_p + 1) \\ \phi(M_p + 1, M_p - 1) & \phi(M_p + 1, M_p) & \phi(M_p + 1, M_p + 1) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} \phi(0, M_p - 1) \\ \phi(0, M_p) \\ \phi(0, M_p + 1) \end{bmatrix} \tag{11}$$

where $\phi(i, j)$ is given by

$$\phi(i, j) = \sum_{n=1}^{N} r(n - i)r(n - j). \tag{12}$$

This, in turn, can be written compactly in matrix form as $\mathbf{\Phi\beta} = \boldsymbol{\alpha}$, where $\mathbf{\Phi}$ is the 3 by 3 correlation matrix. In a backward adaptation scheme, the sequence $r(n)$ would correspond to the past quantized formant residual signal.

In operation, first an estimate of the pitch period M_p is obtained using one of the methods described in [12] (method 1). This method computes the optimal lag for a three tap pitch predictor using a simplified formulation for the predictor. This simplification involves neglecting the off-diagonal terms of $\mathbf{\Phi}$, which is justified if formant prediction is done first. Once the lag M_p is computed, (11) is used to obtain the set of predictor coefficients β_i.

If the pitch predictor is backward adapted, the frame over which the mean-square prediction error is minimized does not correspond to the frame over which the pitch predictor is applied. While this scheme works well in segments where the lag is relatively constant, it does not perform as well in transition regions. This is because the pitch lag and coefficients are too finely tuned to the analysis frame. Some of the adverse effects of backward pitch predictor adaptation can be tempered by "softening" the pitch predictor. Conceptually, this is achieved by adding uncorrelated white noise to the input to the pitch predictor and using this perturbed signal to solve for the pitch lag and pitch predictor coefficients. This approach is implemented by adding a noise term to the diagonal elements of the correlation matrix $\mathbf{\Phi}$—the diagonal elements $\phi(i, i)$ of the matrix $\mathbf{\Phi}$ are replaced by $(1 + \alpha)\phi(i, i)$, where α is a small factor. If α is equal to zero, the pitch predictor coefficients are exactly tuned to the analysis segment. On the other hand, if α is very large, the pitch predictor coefficients tend to zero. In practice, a small value of α is used to achieve some untuning to the analysis segment.

III. Tree Codes

Conventional differential encoders such as the generalized predictive coder described above, can be viewed in a more general setting as so-called tree coding algorithms. These tree codes can be classified into two categories, deterministic and stochastic tree codes. The tree codes associated with conventional waveform coders such as PCM and ADPCM are examples of deterministic tree codes. With differential encoders, it is useful to make a distinction between the innovations code tree which represents the quantized residual signal and the reconstruction code tree which represents the reconstruction output signal. The reconstruction code tree is obtained by passing each of the quantized residual sequences of the innovations code tree through the synthesis filter. The nodes of the reconstruction code tree are then populated with the output values of the synthesis filter. The synthesis filter may be either fixed, backward adaptive, or forward adaptive. With a forward adaptive synthesis filter, the reconstruction code tree is completely specified by the innovations code tree and the side information giving the values of the synthesis filter parameters for each analysis frame. With a backward adaptive synthesis filter, the reconstruction code tree is completely specified by the filter (update algorithm and initial conditions) and the innovations code tree. The encoder configuration in Fig. 1 reflects the use of a frequency weighted error measure in searching through the code tree.

In a deterministic tree, each quantized residual sample can take on one of 2^R values, where R is the number of bits per sample used to encode the input. The process of populating the innovations tree can be considered to be as follows. Each node is assigned a value from a dictionary of size 2^R, the branch number of the node being used as an index into the dictionary. These innovations codes also typically include the effect of a backward adaptive gain. The use of Jayant multipliers [13] is one possibility. In this case, the value assigned to a node is the indexed dictionary value multiplied by a gain value. Backward adaptation of the gain is achieved by multiplying the gain with a multiplier, the value of which depends on the dictionary value indexed. Recently, deterministic codebooks with more than 2^R values have been considered in a trellis rather than a tree structured code [14], [15].

Stochastic trees are richer in that they allow more generality in populating the innovations tree. The nodes of stochastic innovations trees are populated with values as follows. Each node in the tree is associated with a unique path map or branch number sequence from the root up to that particular node. The N_b most recent bits of the path map are used as an index into a dictionary containing 2^{N_b} values. Thus by choosing N_b to be greater than R, each

node in a stochastic tree is no longer restricted to one of 2^R values. Such code trees are said to be stochastically populated because the dictionary is populated with random numbers from a certain distribution. For proper decoding of the quantized residual sequence at the receiver, identical copies of the dictionary must be available to both the transmitter and the receiver.

The effect of a backward adaptive gain can be included in stochastic trees by multiplying the indexed dictionary value with a backward adaptive gain. Two adaptation schemes are proposed. In the first method, the gain is updated based on an exponentially averaged variance estimate. The dictionary value assigned to a node is multiplied by the node gain G to yield an innovations sample e. The node gain is then updated according to

$$\hat{G}^2 = \delta G^2 + (1 - \delta)e^2, \qquad 0 < \delta < 1 \qquad (13)$$

where $\hat{G}$ is the new gain value, and δ is a parameter that controls the effective memory of the gain update. This is a simple gain adaptation scheme, generalizations of which can be obtained through the use of general windows in place of the simple exponential window used in (13).

The second gain update method is an extension of the Jayant update for the deterministic tree case. It is achieved by first partitioning the effective amplitude range of the numbers in the dictionary into subranges. Each of the subranges is then assigned a multiplier value. As in the case of a deterministic tree, the gain is then adapted by multiplication with a value which depends on the subrange occupied by the indexed dictionary value.

The approach taken here towards obtaining good tree codes for speech signals is to capture the statistics of the formant and pitch predicted residual signal in a stochastically populated innovations tree. The effect of a backward adaptive gain is included in the stochastic innovations tree through the use of (13). The essential pitch and formant structure is then inserted by passing the innovations sequence through backward adaptive pitch and formant synthesis filters, to produce the final output reconstruction tree. The next section will address the tree search algorithm issue.

IV. Tree Search Algorithms

Any tree coding scheme involves two basic issues, the choice of an effective code tree, and the choice of a search algorithm to search through the code tree. We address the latter issue in this section. Among search algorithms, a distinction can be made between single path searches and multipath searches. Single path searches proceed through the code tree along one line of decisions, whereas multipath searches consider several paths in parallel, and choose among them at a later time. Let the optimum path map sequences be P_1 and P_2 given by consideration of the input sequence up to time t_1 and t_2, respectively, where $t_1 < t_2$. The path map P_2 up to time t_1 need not in general be the same as P_1. Single path searches neglect this possibility by making irreversible instantaneous decisions about the best path map. Multipath searches therefore yield better performance than single path searches. Given an input sequence of length L samples to be encoded, the optimal multipath search scheme is one that considers all possible output sequences of length L, i.e., the encoder performs an exhaustive search of all possible output sequences. This is an impossibly complex and unnecessary procedure in practice. Also, the choice of L is limited in practice by an encoding delay constraint. One therefore has to consider suboptimal (nonexhaustive) but less computationally expensive multipath search schemes. A highly efficient multipath search scheme is the (M, L) algorithm. This search algorithm is described in the next section, and was used to search the code tree considered in this paper.

A. (M, L) Algorithm

The (M, L) algorithm is controlled by two parameters, M and L [8]. The maximum number of paths kept in contention at any stage is at most M. The length of these paths is equal to L. Each of the saved paths is first extended to the nodes corresponding to the next sampling instant. The cumulative errors for each of the paths are then calculated according to some suitable error measure, and the extended path with the lowest error is identified. This lowest error path will extend from a node L time samples back. The branch number for this node is then transmitted. This corresponds to an incremental mode of operation where each search involving sequences of length $L + 1$ is followed by the release of one branch number. Only valid paths that extend from the chosen node L time samples back are in contention, the others are eliminated. Among the valid paths, at most M lowest error paths are kept and saved for the next stage.

V. Encoding Algorithm

The nodes of the innovations tree were populated from a dictionary containing 4096 samples from a Laplacian pseudorandom number generator. The long-term distribution of normalized formant and pitch predicted residual samples is approximately Laplacian in nature. Backward gain adaptation is included in the innovations tree through the use of (13) with a value of 0.86 for the parameter δ. It was found experimentally that this value gives the best results. The reconstruction tree is obtained by passing this innovations tree through synthesis filter(s).

With the (M, L) algorithm, a certain number of paths are retained at each stage. At the start of the encoder operation, the coding algorithm requires the existence of a certain number of saved paths. The saved paths should reflect the "rest state" of the coder before start up. The most obvious choice for the initial saved paths is to have one saved innovations path of length L samples, whose nodes are all populated with samples of value zero. The initial gain value for this path can be assigned any reasonable value. The initial gain value was found not to be critical in this work. The filter coefficients for the initial

path are also set to zero. The initial innovations path and the gain value are assumed to be known to the decoder.

We will now consider a synthesis filter configuration in which only a formant synthesis filter is used. The filter $F(z)$ along each path is updated at each time instant using the backward adaptive algorithm described. If there is no delay in the update, the formant filter $F(z)$ will evolve differently along different paths of the tree. Proceeding along different parallel paths of the code tree will therefore involve a separate update of the formant filter along each path. By exploiting the fact that all saved paths stem from a single released node L time samples back, the computational complexity of the encoding scheme can be reduced. If the formant filters are updated with a delay of L samples, the filters for each of the saved paths evolve in an identical way. All the saved paths are therefore associated with a single filter which evolves via a delay of L samples. Furthermore, for the exponential adaptive lattice window, experiments show that the prediction gain actually increases with increasing update delays (up to delays of 8 samples) and then levels off. A plot of prediction gain versus update delay for a particular sentence is shown in figure Fig. 4. Other sentences show very similar behavior.

With the inclusion of a pitch synthesis filter, the encoder and decoder configurations of Fig. 2 are used along each path of the tree. The pitch predictor is updated using the past released quantized residual samples, i.e., using the output of the pitch synthesis filter. The pitch filter is updated every 20 samples. The pitch lags are constrained to lie between 20 and 120 samples, and a frame length of 100 samples is used in solving for the pitch predictor coefficients. It was found experimentally that a frame length of 100 gave the best results.

VI. Objective Test Results

The encoding algorithm was simulated on a VAX 8600 computer using floating point arithmetic. A simple one-pole or exponential window given by

$$W(z) = \frac{1}{1 - \beta z^{-1}} \tag{14}$$

is used to form the weighted sum of the forward and backward residual signals in (4). The update equations for $C_m(n)$ and $D_m(n)$ are then given by

$$\begin{aligned} C_m(n) &= \beta C_m(n-1) + f_{m-1}(n) b_{m-1}(n-1) \\ D_m(n) &= \beta D_m(n-1) + \gamma f_{m-1}^2(n) \\ &\quad + (1-\gamma) b_{m-1}^2(n-1). \end{aligned} \tag{15}$$

The value of β is chosen to be 0.986. The stability constant γ of the adaptive lattice is fixed at 0.5 in order to ensure stability of the synthesis filter. The order P of the formant predictor is fixed at 8. A value of μ equal to 1.0 is used in the noise feedback filter $N(z)$ implying the use of a squared error distortion measure. Adaptation of the formant filter is done using the reconstructed speech samples. The resulting reflection coefficients are converted to direct form coefficients for use in the filtering operations.

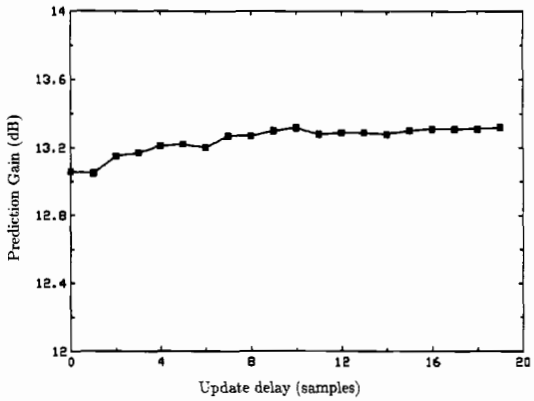


Fig. 4. Variation of prediction gain with update delay.

The objective results are given in terms of plots of segmental signal-to-noise ratio (segSNR) versus time, and in terms of averaged segSNR values, the average being taken over a whole sentence. In the former, the signal-to-noise ratio in decibels (dB) is calculated for successive overlapping (80 sample overlap) blocks of 100 samples (12.5-ms blocks), thus yielding a graphical display of the time variations in signal-to-noise ratio. Averaged segSNR values are obtained by calculating the signal-to-noise ratios in dB for nonoverlapping blocks 16 ms in duration, and then averaging the SNR values over all the blocks in a sentence. Thus,

$$\text{segSNR} = \frac{1}{K} \sum_i \text{SNR}_i \tag{16}$$

where SNR_i is the signal-to-noise ratio for the ith block in decibels, there being K such blocks.

A. Performance with M

The objective results using only a formant synthesis filter are presented first. An eighth-order formant filter was used. The plot in Fig. 5 shows the segSNR values versus M for a particular sentence with the stochastically populated innovations tree. The segSNR value increases rapidly with M at first, and then finally saturates with M to an almost constant value. Other sentences show a very similar behavior. Saturation is achieved with M equal to about 16. Note that the branching factor of the code tree is equal to four, and that with L equal to eight, there are 4^8 paths available, of which a maximum of M are considered at any stage. Saturation in performance with M is therefore attained with a value of M that is very much less than the total number of paths available. This is in keeping with the results of multipath search using the (M, L) algorithm of differential encoder code trees [2]–[4], [6]. Fig. 5 also shows the performance with M with a deterministic innovations tree. The underlying 4 level quantizer was made adaptive using a Jayant step size update

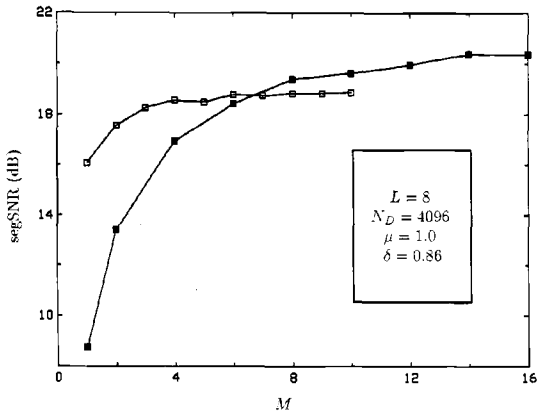


Fig. 5. Performance with M for the stochastic (shown by the filled squares) and deterministic (shown by the unfilled squares) cases.

[13] (multiplier values of 0.8 and 1.6). The results show that with the deterministic tree, the (M, L) algorithm achieves a performance similar to that for an exhaustive search [7].

Comparing the deterministic code with the stochastic code brings out some interesting points. First, saturation with M occurs at a much lower value of M with a deterministic tree than with a stochastic tree. Second, the segSNR value for $M = 1$ with a deterministic tree is larger than that obtained with a stochastic tree. The stochastic code performs well with a multipath search, but gives very poor performance with a single path search. Intuitively, the reason for this is as follows. In the stochastic tree code, the output sequences along different paths of the tree are highly dissimilar in nature. This is in contrast to the deterministic case, where the innovations tree is more structured than in the stochastic case. In particular, the dictionary size is equal to the branching factor of the tree in the deterministic case, thus rendering the code structure more amenable than a stochastic code to a single path search. With a stochastic code, typical candidate output sequences are not generated through a single path search. Because of the highly destructured nature of the stochastic code, in looking at several paths in parallel the chances of finding an output sequence very close to the input sequence, are higher than in the deterministic case. One therefore has to use a multipath search to reap the full benefits of the stochastic code. The segSNR value, once saturation with M is attained, is higher with a stochastic tree than with a deterministic tree. Subjective performance is also better with a stochastic tree, for large enough values of M. The results indicate that the choice of the code tree and the search algorithm are by no means independent design issues.

The objective performance with the inclusion of a pitch synthesis filter was also investigated. In solving for the pitch predictor coefficients, the diagonal elements of the correlation matrix were perturbed using a value of $\alpha = 0.01$. Fig. 6 shows a plot of segSNR versus time, both with and without the inclusion of a pitch synthesis filter. Note that an increase in segSNR of about 2–10 dB is attained during the voiced segments. The segSNR remains about the same during unvoiced segments. The improvement in SNR and segSNR over a whole sentence ranges from about 0.5 dB for some sentences to about 2.5 dB for others.

B. Performance with L

This section investigates the performance of the system with L for a male utterance which exhibits typical behavior. The parameter L is the length of the paths considered in the multipath search. Fig. 7 shows plots of segSNR with L for fixed M ($M = 16$). The performance tends to saturate with increasing L. The performance of the system with a lower order predictor saturates earlier than that for the higher order predictor. The value of L does not seem to be critical, provided it is high enough to account for the predictor order.

VII. Subjective Test Results

A subjective test of the coder was carried out by conducting a preference test between tree coded utterances and log-PCM coded utterances of various bit rates. The twelve listeners consisted of mostly "naive listeners" (students working in areas other than speech coding) and a few trained listeners (those working in the speech coding area). The "naive listeners" were more inclined towards the tree coded speech than "trained listeners." The following parameters were used for the tree coded sentences; a 3 tap pitch predictor with noise perturbation adaptation ($\alpha = 0.01$), an 8 tap formant predictor using a one-pole error window ($\beta = 0.986$), noise shaping ($\mu = 0.85$), and tree searching with $M = 16$ and $L = 8$. The tree coded sentences were compared with 5, 6, 7, and 8 b/sample log-PCM coded sentences. The subjective test presented pairs of sentences, a tree coded version and a log-PCM version, with both orderings represented. The various test pairs were randomly ordered in the presentation. Results of the subjective tests are shown in Fig. 8. The vertical axis shows the fraction of times that the tree coded sentences were preferred over the corresponding log-PCM coded sentences. For example, tree coded sentences were preferred over 5 b/sample log-PCM coded sentences every time. The equal preference point is achieved at about 7 b/sample log-PCM. One can therefore conclude that the tree coding scheme achieves a level of subjective quality equal to 7 b/sample log-PCM.

Informal listening tests with and without the pitch filter, indicate that the pitch filter while not generating a substantial overall improvement in quality, is beneficial in certain crucial segments of the test utterances. Subsequent to the formal subjective tests, an adaptive postfilter [16] was added to the system. This adaptive postfilter helps in a small way to further improve the quality of the reconstructed speech.

The effect on subjective quality of other types of windows for the adaptive lattice was also considered. A class

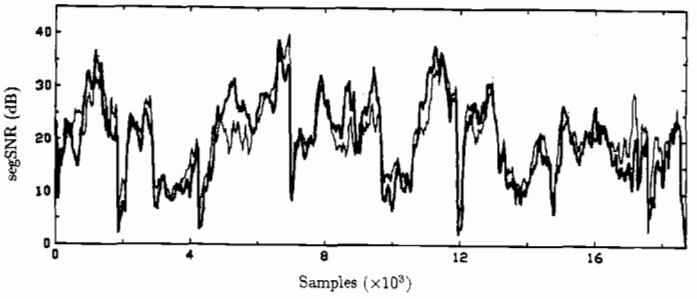


Fig. 6. segSNR with pitch prediction and noise addition. The thin line shows segSNR without pitch prediction. The thick line shows the segSNR with pitch prediction and noise addition.

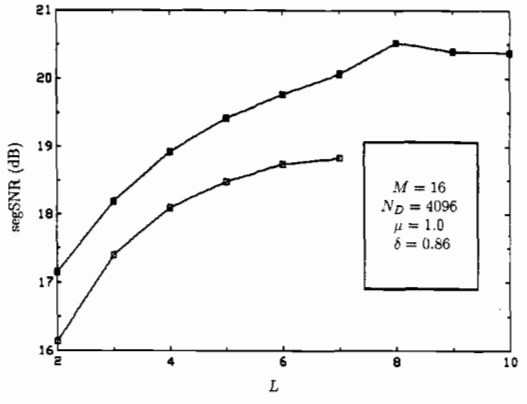


Fig. 7. Performance of the system with L with an eighth-order predictor (shown by the filled squares) and with a third-order predictor (shown by the unfilled squares).

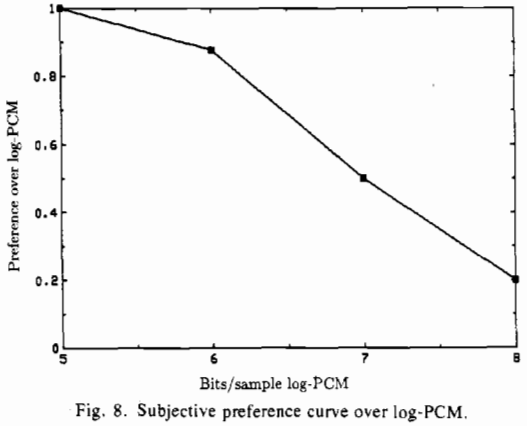


Fig. 8. Subjective preference curve over log-PCM.

of windows that are impulse responses of filters having one zero and two poles were tried. The window $w(n)$ is obtained as the sum of two decaying exponential sequences as given by

$$w(n) = (\beta_1)^n - a(\beta_2)^n. \tag{17}$$

The values of β_1, β_2 and a were chosen so as to ensure that $w(n) \geq 0$ for $n \geq 0$. The z transform of $w(n)$ is given by

$$W(z) = \frac{(1 - a) + (a\beta_1 - \beta_2)z^{-1}}{1 - (\beta_1 + \beta_2)z^{-1} + \beta_1\beta_2 z^{-2}}. \tag{18}$$

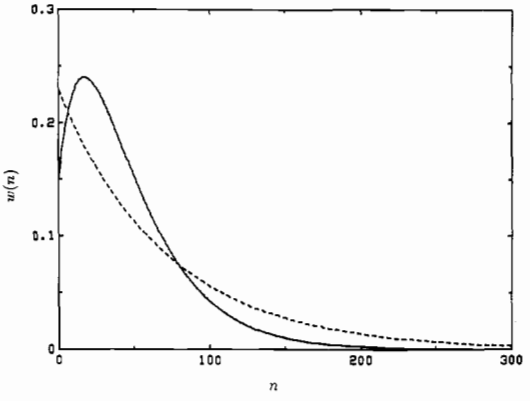


Fig. 9. Window obtained with $\beta_1 = 0.97$, $\beta_2 = 0.95$, and $a = 0.85$ (solid line). A simple exponential window with $\beta = 0.986$ and normalized to the same area is also shown (dashed line).

An example of such a window is shown in Fig. 9. For comparison, the first-order window used earlier is shown on the same plot. By carefully controlling the parameters, a window shape that is intermediate to the one and two-pole-types is obtained. For β_1 and β_2 equal to 0.97 and 0.95, respectively, choosing the parameter a close to zero results in a first-order all-pole window, and choosing a close to one results in a second-order all-pole window. Use of the window shown in Fig. 9 results in a 0.1–0.3 dB increase in signal-to-noise ratio, for various sentences. Subjectively, this window gives an output speech quality that is "crisper" than that obtained with a one-pole window.

VIII. Summary

A code tree generated by a stochastically populated innovations tree with a backward adaptive gain, and backward adaptive synthesis filters was considered. The code tree was searched using the multipath (M, L) search algorithm. For large values of M, the stochastic code tree

was found to give better performance than the deterministic tree, both objectively and subjectively. The addition of the pitch filter gives 2–10 dB increase in segSNR in the voiced segments. Subjective testing has shown that the coder attains a subjective quality equivalent to 7 b/sample log-PCM, with an encoding delay of 8 samples (1 ms with an 8 kHz sampling rate).

Robustness to noisy channel conditions was not considered in designing the coder presented above. Indeed in many network applications, the error rate is extremely low. However, a new class of applications is emerging with more severe error environments, a good example being that of mobile radio. In such situations, it is important that the effects of channel errors be localized. The extent to which this is true depends on the nature of the reconstruction code tree. A single bit error will lead to consecutive errors in the indexed dictionary value. These errors will also have an effect on the evolution of the gain and the synthesis filter parameters which will in turn affect the values of the future output samples. The adverse effects of bit errors can be reduced by reducing the size of the dictionary and also by control of the memory in the gain update and the synthesis filter updates.

Since the time of the publication of a shorter conference version of this paper [17] there has been an accelerated interest in low delay speech coding to the extent that a CCITT study group is considering the formulation of an international standard for low delay coding at 16 kb/s.

References

[1] W. R. Daumer, P. Mermelstein, X. Maitre, and I. Tokizawa, "Overview of the ADPCM coding algorithm," in *Proc. IEEE Global Telecommun. Conf.* (Atlanta, GA), Nov. 1984, pp. 774–777.

[2] J. B. Anderson and J. B. Bodie, "Tree encoding of speech," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 379–387, July 1975.

[3] N. S. Jayant and S. A. Christensen, "Tree encoding of speech using the (M, L)-algorithm and adaptive quantization," *IEEE Trans. Commun.*, vol. COM-26, pp. 1376–1379, Sept. 1978.

[4] S. G. Wilson and S. Husain, "Adaptive tree encoding of speech at 8000 b/s with a frequency-weighted error criterion," *IEEE Trans. Commun.*, vol. COM-27, pp. 165–170, Jan. 1979.

[5] T. Svendsen, "Tree coding of the LPC residual," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (San Diego, CA), Apr. 1984, pp. 10.11.1–10.11.4.

[6] H. G. Fehn and P. Noll, "Multipath search coding of stationary signals with applications to speech," *IEEE Trans. Commun.*, vol. COM-30, pp. 687–701, Apr. 1982.

[7] J. D. Gibson and G. B. Haschke, "Adaptive code generators for tree coding of speech," in *Proc. IEEE Int. Conf. Commun.* (Seattle, WA), June 1987, pp. 1142–1146.

[8] F. Jelinek and J. B. Anderson, "Instrumentable tree encoding of information sources," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 65–76, Jan. 1971.

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

[10] J. I. Makhoul and L. K. Cosell, "Adaptive lattice analysis of speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 654–659, June 1981.

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

[12] R. P. Ramachandran and P. Kabal, "Pitch prediction filters in speech coding," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-37, pp. 467–478, Apr. 1989.

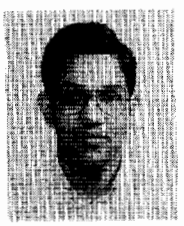
[13] N. S. Jayant, "Adaptive quantization with a one word memory," *Bell Syst. Tech. J.*, vol. 52, pp. 1119–1144, Sept. 1973.

[14] E. Ayanoglu and R. M. Gray, "The design of predictive trellis waveform coders using the generalized Lloyd algorithm," *IEEE Trans. Commun.*, vol. COM-34, pp. 1073–1080, Nov. 1986.

[15] M. W. Marcellin, T. R. Fischer, and J. D. Gibson, "Predictive trellis coded quantization of speech," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (New York, NY), Apr. 1988, pp. 247–250.

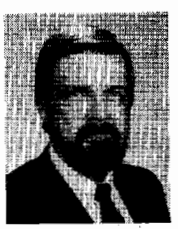
[16] J.-H. Chen and A. Gersho, "Real-time vector APC speech coding at 4800 bps with adaptive postfiltering," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Dallas, TX), Apr. 1987, pp. 2185–2188.

[17] V. Iyengar and P. Kabal, "A low delay 16 kb/s speech coder," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (New York, NY), Apr. 1988, pp. 243–246.



Vasu Iyengar was born in New Delhi, India, in January 1964. He received the B.Eng(Hons) degree in electrical engineering in 1986, and the M.Eng degree in electrical engineering in 1987, both from the Department of Electrical Engineering, McGill University, Montreal, Canada.

Since May 1986, he has been carrying out his research at INRS-Télécommunications, Verdun, Que., Canada. His research interests include speech coding and adaptive filtering.



Peter Kabal (S'70–M'75) received the B.A.Sc., M.A.Sc., and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, Ont., Canada.

He is an Associate Professor in the Department of Electrical Engineering at McGill University, Montreal, Que., Canada, and a Visiting Professor at INRS-Telecommunications (a research institute affiliated with the Université du Québec), Verdun, Quebec. From September 1982 to September 1983 he was a full-time consultant to the Speech Communications Group at Bell-Northern Research, Verdun, Quebec. From September 1987 to June 1988 he spent a sabbatical year as a Visiting Professor at the University of California, Santa Barbara. His current research interests focus on signal processing as applied to speech coding, adaptive filtering, and data transmission.