ADAPTIVE TRANSFORM CODING OF SPEECH

by

David G. Sloan, B. Eng. (Hons. Electrical)

Department of Electrical Engineering
McGill University
Montreal, Canada

A thesis submitted to the Faculty

of Graduate Studies and Research in partial fulfillment

of the requirements for the degree of Master of Engineering

July, 1979

Electrical Engineering M. Eng.

ABSTRACT

ADAPTIVE TRANSFORM CODING OF SPEECH

by

David G. Sloan

This thesis investigates adaptive discrete transform coding of speech signals. The thesis addresses the problem of adaptive quantization of the transform coefficients. An all-pole estimate of the signal energy spectrum results in a quantization strategy which outperforms previously reported techniques. Additional perceptual improvements are obtained by pre-emphasizing the input signal to better reproduce high frequency formants and by windowing the input to reduce block boundary discontinuities. Results from computer simulations of this coding technique are presented. At 16 kb/sec the proposed scheme yields high quality speech. At the rate of 9.6 kb/sec, the coded speech is completely intelligible but contains a slight warbling sound.

Génie électrique Maîtrise

Résumé

CODAGE ADAPTIF DE LA VOIX PAR TRANSFORMÉE

par

David G. Sloan

Cette thèse traite du codage adaptif des signaux de parole et utilise une méthode de transformées discrètes. La thèse étudie le problème de la quantification adaptive des coefficients de la transformée. Un modèle autorégressif du spectre d'énergie du signal permet d'élaborer une stratégie de quantification qui surpasse les techniques décrites jusqu'alors. On obtient une amélioration perceptuelle additionnelle en pré-accentuant le signal d'entrée, ce qui permet une reproduction plus fidèle des formants hautes fréquences, et en effectuant une pondération du signal d'entrée pour réduire les discontinuités entre les frontières des blocs. On présente les résultats de cette technique de codage, obtenus par simulation sur ordinateur. La méthode proposée fournit un signal vocal de haute qualité pour un taux de transmission de 16 kb/sec. Au taux de transmission de 9.6 kb/sec, le signal de parole codé est totalement intelligible mais présente une légère sonorité gazouillante.

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor, Dr. P. Kabal, for his guidance in both the experimental aspect of this work and the preparation of this report.

I am also grateful to Dr. P. Mermelstein and Dr. M. Kaplan for their helpful discussions and critical comments.

TABLE OF CONTENTS

PAGE

I Introduction

In recent years increasing emphasis has been placed on the digital encoding of analogue signals. In this study, we examine one strategy for the efficient digital encoding of speech signals. Efficiency, in this context, denotes striving for high quality signal reproduction at low transmission rates. Our objective is to optimize the parameters of the coding scheme to achieve the best speech quality for a given transmission rate.

Transmission rate and coder design determine the fidelity of the signal reproduced at the receiver. Increasing the transmission rate improves the accuracy of the signal representation and thus implies a higher fidelity received signal. Alternatively, for a fixed transmission rate, improved fidelity can be obtained by using a coding scheme which removes signal redundancies before transmission. The coding efficiency is sustained by transmitting only the important signal attributes.

To achieve high fidelity at low transmission rates, it is often necessary to increase the complexity of the coder so it can better exploit the signal characteristics. Greater complexity usually implies higher cost. Consequently, for a given fidelity requirement, there are tradeoffs between coder complexity, transmission rate, and cost.

The goal of decreasing communication costs has motivated the current interest in digital transmission. Advances in digital circuit technology have been accompanied by dramatically decreasing costs. Large scale

integration techniques are applicable to the development of coders. Thus the investigation of even relatively complex coding schemes may give rise to practical and economic coders in the future.

The transmission of signals in a digital format has several advantages. A digital format facilitates signal transformation and digital signals can be received and retransmitted without loss of signal quality. In addition, some communication networks (notably the telephone network) are moving toward digital switching and control of signals in the network.

Digital encoding is also useful in several special applications. One is communication security (e.g. in police and military systems). The digital format allows for easy encryption of secure messages. Another application is to the reduction of storage requirements in message store-and-forward systems, in which voice messages are stored digitally for later retrieval. As the size of the system (number of users, number of messages) grows, memory demands increase. Coding before storage reduces memory requirements.

Traditionally, speech coders have been divided into two distinct families: waveforms coders and vocoders (a concatenation of the words voice coders). Waveform coders try to approximate the input waveform. They are generally designed without reference to a specific speech generation model. For speech inputs, waveform coders tend to be robust, in the sense that there is little degradation in performance due to varying speaker characteristics and background noise. Waveform coders can operate in either the time or frequency domain. The PCM family [1]

exemplifies time domain waveform coding. Sub-band coding [2] is an example of frequency domain waveform coding.

Vocoders, on the other hand, attempt to model speech production. They parameterize the signal according to this model and transmit only the parameters. In general, vocoder performance is more sensitive to speaker variations and background noise. In addition, the decoded speech often has a unnatural or synthetic quality. Vocoders, however, can transmit intelligible speech at a much smaller transmission rate than can be achieved with waveform coding. Linear predictive coding [3] is an example of a time domain vocoder. It models the vocal tract as a time varying linear filter excited by either a periodic source (for voiced sounds) or a noise source (for fricatives and unvoiced stops). The model requires that physical attributes such as pitch and voiced/unvoiced decisions be extracted from the signal. The formant vocoder is an example of a frequency domain vocoder. It transmits speech sounds by sending formant frequencies and bandwidths.

The approach taken in this study lies between pure waveform coding and pure vocoding. The coding scheme, known as transform coding, encodes a transformation of the speech signal. The adaptive form of the scheme includes some speech-specific modelling to achieve better performance.

The purpose of this investigation is to examine various transform coding strategies for speech communication. Our starting point is a technique recently reported by Zelinski and Noll [4]. We examine their basic scheme with the aim of increasing the quality of coded speech at low bit

rates. We consider a single transformation, the discrete cosine transform, and concentrate on improving quantization techniques. Transmission issues such as channel errors are not considered.

This study addresses the problem of adaptive quantization of the transform coefficients. In adaptive quantization schemes extra information, termed side information, must be communicated to the receiver for proper signal reconstruction. A side information strategy which represents the spectral envelope by an all-pole model outperforms previously reported techniques. Additional perceptual improvements are obtained by pre-emphasizing the input signal to better reproduce high frequency formants and by windowing the input to reduce block boundary discontinuities. At 16 kb/sec this coding scheme yields high quality speech. At 9.6 kb/sec, the coder speech is completely intelligible but contains a slight warbling sound.

Early work in transform coding was done by Campanella and Robinson [5], who compared the performance of several transformations for speech coding. Wintz [6] considered transform coding for pictures by investigating the choice of transformation and quantization strategy. Recently, several papers have appeared on transform coding of speech [7], [8]. There is a review in [9] of a variety of speech coding techniques, including transform coding, and a comparison of waveform coders and vocoders.

This report is organized into five chapters. Following the introduction, two chapters deal with the theory of transform coding. Chapter II describes the basic structure of a transform coder and discusses the

choice of transform and vector quantization. Chapter III deals more specifically with issues in adaptive transform coding. The fourth chapter describes a computer simulation of adaptive transform coding and gives the results of the simulation. Chapter V lists the conclusions of the study.

II Theory of Transform Coding

In this section we discuss the fundamentals of transform coding (TC). After describing the basic structure of a transform coder, we motivate the choice of the discrete cosine transform (DCT) and present a vector quantization strategy.

2.1 Basic Structure

The basic structure of a transform coder is shown in Figure 2.1. At the transmitter an analogue signal is digitized by sampling above its Nyquist rate. The input signal is assumed bandlimited to a frequency range $[0, F_{max}]$, so that sampling at or above the Nyquist rate, $2 F_{max}$, is a lossless operation. The samples are buffered and grouped into blocks of length N. Denote one such block by $\underline{X}$. The vector $\underline{X}$ is transformed into a new vector $\underline{Y}$, also of length N. The transformation is linear, and can be represented by the matrix equation

$$\underline{Y} = A \underline{X} .$$

A quantizer Q operates on $\underline{Y}$ yielding $\hat{\underline{Y}}$. The quantized vector $\hat{\underline{Y}}$ is then transmitted across the channel to the receiver.

At the receiver the original signal is reconstituted. As a first step $\hat{\underline{Y}}$ is inverse transformed to give $\hat{\underline{X}}$ as an approximation to $\underline{X}$. The samples in $\hat{\underline{X}}$ are then buffered and passed through a D/A converter to regenerate an analogue signal.

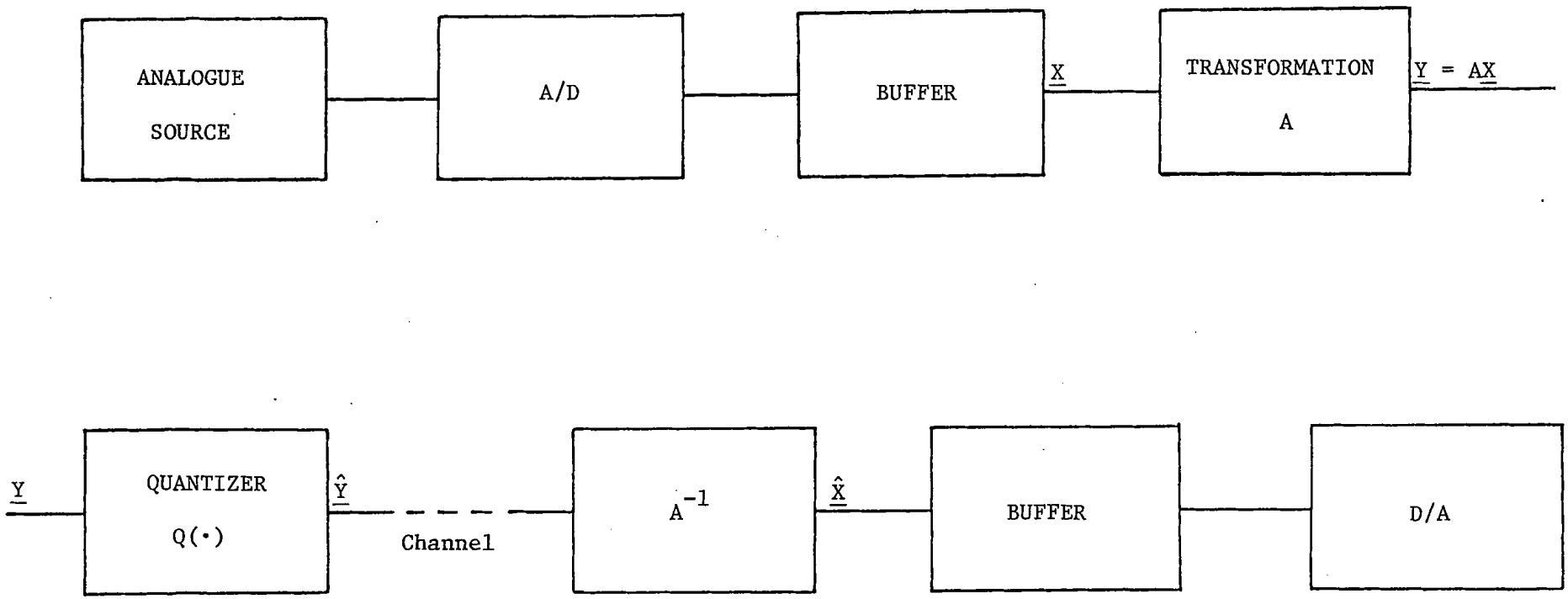


Figure 2.1 Transform coding block diagram.

We proceed to specify the transformation and the quantizer.

2.2 Orthogonal Transformation

The transformations of interest are orthogonal transformations; that is, if A is the matrix representing the transformation.

$$A^{-1} = A^{t}$$

where superscript t signifies complex conjugate transpose. Such transformations are important for digital signal processing in general [10]. For coding their utility is two-fold.

First, from a statistical viewpoint, TC can decorrelate the input samples; that is, the transform coefficients exhibit less correlation that the original data. Decorrelation facilities efficient quantization in the transform domain.

Second, TC can be advantageous in exploiting perceptual knowledge. It is often the case that the transform coefficients have a frequency domain interpretation. In the case of speech signals, much of the current perceptual theory is based on frequency domain parameters (formants, pitch, etc). TC can facilitate the understanding and the elimination of perceptual distortions resulting from coding.

2.2.1 Karhunen-Loève Transformation

An example of an orthogonal transformation is the Karhunen-Loève transformation (KLT). The KLT is a data dependent transformation whose basis vectors are the eigenvectors of the autocorrelation matrix $R_{\underline{X}}$ of the $\underline{X}$ process. The KLT diagonalizes the autocorrelation matrix of the transformed vector $\underline{Y}$. Thus the components of $\underline{Y}$ are uncorrelated.

The KLT is "optimal" in a mean square error (MSE) sense. We digress briefly to explain why MSE is an appropriate distortion measure for speech coding.

Minimizing MSE is equivalent to maximizing the signal-to-noise ratio (SNR)[1]. SNR can be used to rank the performance of coders. In particular, modifications to a coding scheme which increase SNR usually imply better perceived quality. For speech signals maximization of SNR on a block by block basis (known as segmental or block SNR) is known to be a good objective correlate of subjective preference [11], [12]. The SNR measure also makes the mathematical analysis more tractable.

The KLT is optimal in the sense of approximating $\underline{Y}$ by discarding some of its components; that is, approximating $\underline{Y}$ by some $\underline{Y}'$ in a lower dimensional space. It can be shown that the MSE in representing $\underline{X}$ by $\underline{Y}'$ (in place of $\underline{Y}$) is simply the sum of the variances of the discarded

[1] SNR is defined as the ratio of signal energy to the energy in the difference signal between the original and the coded waveforms.

transform coefficients. Therefore, if only the lowest variance coefficients are thrown away, the approximation is the best (for fixed dimension of the approximation) in a MSE sense.

The KLT has two disadvantages. First, the transform depends on $R_{\underline{X}}$. This is undesirable because we may not know $R_{\underline{X}}$ a priori and even if we approximate $R_{\underline{X}}$ by a long term average, the signal may be non-stationary. In the latter case the transformation changes with time. This brings up the second disadvantage. The KLT is computationally burdensome. Its calculation requires $O(N^2)$ operations[1]. Furthermore, numerical solution of eigenvector problems is difficult in the sense that the solutions can become unstable. Thus we are motivated to look for alternatives to the KLT.

2.2.2 Discrete Cosine Transformation

One alternative to the KLT is the discrete cosine transformation (DCT). The DCT approximates the KLT and is easier to implement. The DCT is defined as

$$Y(k) = \frac{2c(k)}{N} \sum_{n=0}^{N-1} x(n)\cos[\pi(2n+1)k/2N] \qquad k = 0,1,\ldots,N-1 \qquad (2.1)$$

[1] $O(.)$ is a measure of the complexity of algorithms, $O(N^2)$ indicates complexity increasing as the square of N.

where

$$c(k) = 1/\sqrt{2} \qquad k = 0$$

$$= 1 \qquad k = 1,2,\ldots,N-1$$

Eq. (2.1) the x(n), n=0,1,...,N-1 are the N components of $\underline{X}$ and Y(k), k=0,1,...,N-1, the N components of $\underline{Y}$. The inverse DCT (IDCT) is given by

$$x(n) = \sum_{k=0}^{N-1} c(k)Y(k)\cos[\pi(2n+1)k/2N] \qquad n = 0,1,\ldots,N-1$$

The DCT converges in mean-square to the KLT in the limit of large block length N for a first order stationary Markov autocorrelation matrix [13]. One study [4] has examined several candidate transformations for use in transform coding of speech and concluded that the DCT approximates well the KLT's performance.

To further justify the choice of the sub-optimal DCT we appeal to practical considerations. From (2.1) it is obvious that the form of the DCT is independent of the signal statistics. Furthermore, fast algorithms exist for both DCT and IDCT ($O(N\log_2 N)$ operations when calculated using a fast Fourier transform (FFT) algorithm). Appendix A presents an in-place version of this technique; "in-place" designating a procedure whereby the components of $\underline{X}$ are replaced by those of $\underline{Y}$ with minimal auxiliary storage.

Another advantage of the DCT is that it has a frequency domain interpretation analogous to that of the discrete Fourier transform (DFT). The analysis is from [7]. Define

$$u(n) = x(n) \qquad n = 0,1,\ldots,N-1$$

$$= 0 \qquad n = N,N+1,\ldots,2N-1$$

Consider the 2N point DFT of u(n)

$$U(k) = \sum_{n=0}^{2N-1} u(n)\exp[-j(2\pi/2N)nk]$$

$$= \sum_{n=0}^{N-1} u(n)\exp[-j(2\pi/2N)nk] \qquad (2.2)$$

$$= \exp[jk\pi/2N] \sum_{n=0}^{N-1} u(n)\exp[-j(2\pi/2N)(2n+1)k]$$

Comparing eq. (2.2) with eq. (2.1) Y(k) can be expressed as

$$Y(k) = \mathrm{Re}\{\frac{2c(k)}{N} \exp[-jk\pi/2N]U(k)\}$$

$$= \frac{2c(k)}{N} \mid U(k) \mid \cos[\mathrm{Arg}\{U(k)\}-k\pi/2N]$$

It follows that the envelope of the DCT spectrum is that of the DFT modulated by the term $\cos[\mathrm{Arg}\{U(k)\}-k\pi/2N]$. Moreover, the DCT is bounded by the DFT envelope. Thus the DCT will contain the same perceptual information as the DFT spectrum (e.g. formant structure, pitch striations).

We have presented the DCT as a good choice for the transform in TC. We next examine the problem of efficient transmission of the DCT coefficients.

2.3 Quantization of Vector Sources

Quantization is defined as the process of representing a continuous, possibly infinite range of values by elements in a finite set. If the number of values in the set (usually referred to the number of quantizer levels) is less than 2^m we can represent the variable by an m bit word. As a consequence of this representation we incur a distortion known as quantization error. It is reasonable to require minimum distortion for a fixed number of quantizer levels.

Quantization of vector sources (block quantization) is an extension of the single variable case. A vector of variables is to be suitably represented by a vector in some finite set. The design of such quantizers is greatly simplified if the component variables can be quantized independently without increased distortion. The following represents such a quantization scheme for jointly Gausssian random variables (in vector form) where the distortion measure is MSE. The results are drawn from [14], [15].

When the variables are Gaussian, the statistical independence required among the vector components reduces to the requirement that the components be uncorrelated. Correlated random variables can be transformed to uncorrelated ones by the KLT. KLT decorrelation of the

vector components allows each single variable quantizer to be designed independently [15]. It only remains to find the minimum MSE single variable quantizer for a given probability distribution.

The problem of designing minimum MSE quantizers was solved independently by Lloyd and Max [16], [17]. Lloyd-Max quantizers are in general non-uniform, i.e. the quantizer output levels are not uniformly spaced. Moreover, the design of such quantizers usually involves iterative techniques to solve the equations resulting from the minimum MSE formulation.

Figure 2.2 illustrates the optimal quantization scheme. The matrix G represents the KLT and H its inverse. Since G is orthogonal, $H = G^t$. The quantizer Q consists of a separate Lloyd-Max quantizers for each transform coefficient. For reasons stated previously the DCT can replace the KLT with only a small loss in performance.

With the quantizer structure specified, the one issue that remains is the assignment of the number of quantizer levels to each coefficients. This problem, termed bit assignment, is constrained by the total number of bits per block B. A fraction of B bits is allocated to each coefficient so as to minimize MSE.

Given a bit rate r, a sampling frequency f_s, and a block size N, the number of bits per block is

$$B = \frac{Nr}{f_s}$$

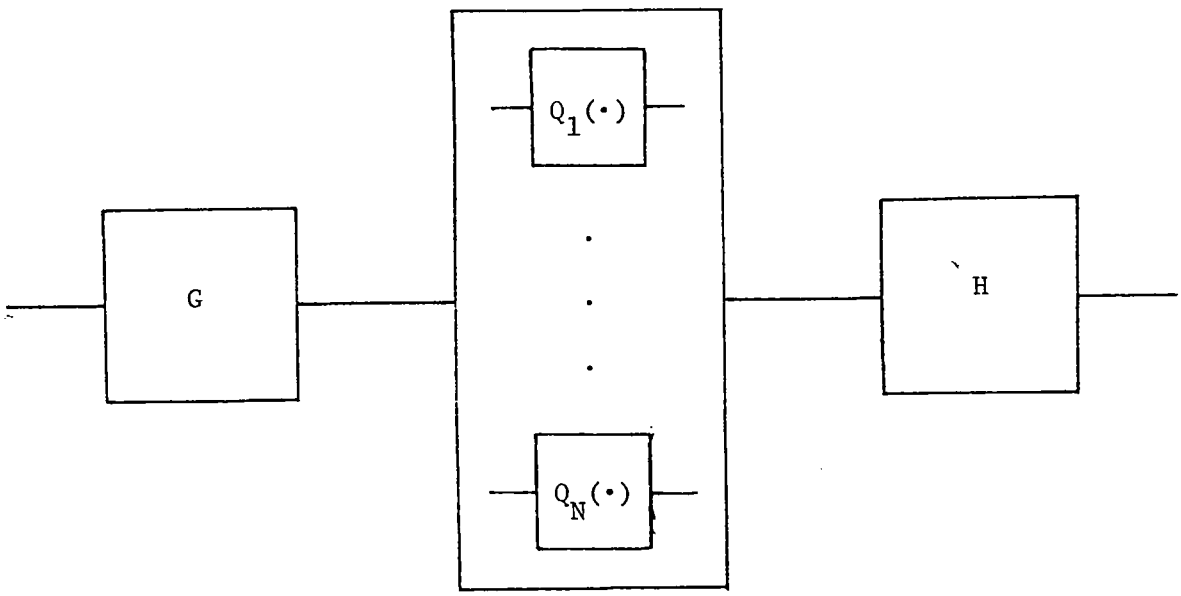


Figure 2.2 Optimal quantization scheme
(G represents KLT transformation).

The parameter r specifies the data rate over the transmission channel. This is the physical constraint which determines B.

The solution to the bit assignment problem is from [15]. Let $(\sigma_i)^2$ be the variance of the i-th transform coefficients and let b_i be the (real valued) number of bits assigned to i. Then

$$b_i = B/N + \tfrac{1}{2} \log_2 \left[\frac{\sigma_i^2}{\left[\prod_{j=1}^{N} \sigma_j^2 \right]^{1/N}} \right] \qquad i = 1,2,\ldots,N$$

Reference [15] also gives an optimal bit assignment technique when b_i are constrained to be integers. Let D(j) be the MSE resulting from the quantization of a (Gaussian distributed) coefficient with a j bit Lloyd-Max quantizer. The values D(j) are tabulated in [16]. The technique is to calculate the marginal returns,

$$P_{ij} = \sigma_i^2 \cdot [D(j) - D(j+1)] \qquad \begin{matrix} i = 1,2,\ldots,n \\ j = 0,1,\ldots \end{matrix} \qquad (2.3)$$

arrange P_{ij} in decreasing order, and assign bits one by one in the sorted order. In this manner the global minimum MSE is achieved.

The development above required that the quantizer input be Gaussian. In the case of present interest, when the quantizer input consists of the DCT coefficients of a speech segment, the exact form of the multivariate distribution of the transform coefficients is not known. For speech and other signals, unimodal distributions are common and we do not anticipate

the transformation to alter the distribution drastically. Hence we expect the quantization scheme developed here to result in improvements over single variable quantization even when the distributions involved are not Gaussian.

III Adaptive Transform Coding

Adaptive transform coding (ATC) is the name for a class of modified TC schemes. The goal of ATC is to improve on the basic TC method.

First consider a fixed scheme. The basic TC approach of Chapter II has been examined for speech signals [4]. Coefficient statistics estimated from long utterances are used to design a set of quantizers. The quantizers are non-adaptive in that they are fixed irrespective of changes in the input signal. This approach gives unsatisfactory results. One problem is that speech sounds vary greatly from quasi-periodic (vowels) to noise-like (fricatives). Thus speech is not a stationary process and a fixed quantization scheme based on the stationarity assumption yields poor performance. A second difficulty is that a fixed design optimized for a single speaker may be inappropriate for a different speaker. ATC is a dynamic strategy to extend the usefulness of TC.

The adaptivity in ATC refers to adjusting the parameters of the coder to suit changing signal characteristics. In TC, if we assume the transform is not altered, the only flexibility rests in the quantization strategy. Ideally, we want a scheme that can adapt to those changes and, at the same time, can be described by few parameters.

The latter is necessary because the dynamics of the scheme must be communicated to the receiver for proper signal reconstruction. This additional information that must be transmitted is termed "side information".

Consider an adaptive quantization approach based on the transform coefficient variances. We have already seen in eq. (2.3) that the bit assignment depends on the distribution of these variances. Furthermore, for a given form of probability density function the Lloyd-Max quantizer is determined solely by the variance of the distribution. Thus we can design unit variance quantizers corresponding to different numbers of levels and simply scale the quantizers by the standard deviation of the variable. Equivalently, we may scale the variable to unit variance, quantize and rescale the quantized variable. To keep pace with changing signal properties we update the scheme on a block by block basis. We need, therefore, an estimate of the coefficient variances which reflects the signal dynamics.

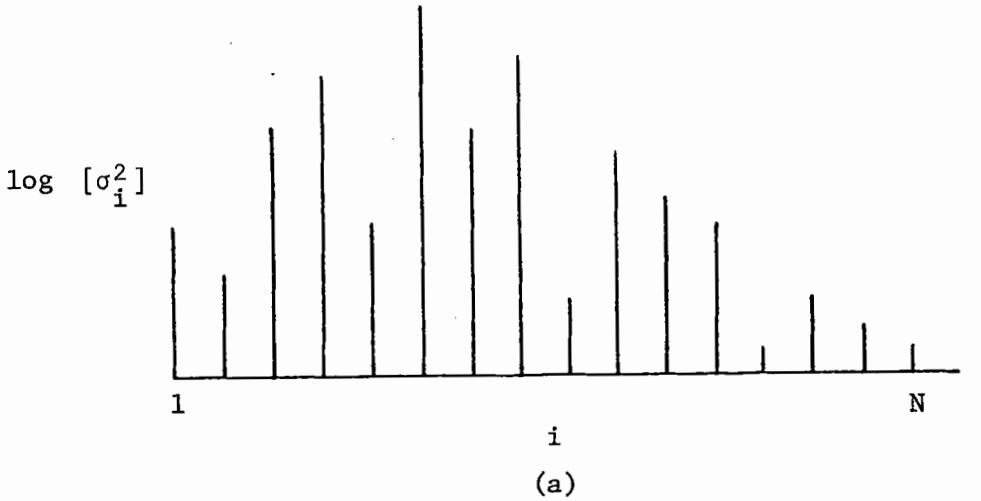
3.1 Basis Spectrum Estimation

Zelinski and Noll [4] have coined the term "basis spectrum" for the distribution of the transform coefficient variances. The problem is first to estimate the basis spectrum and then to transmit it efficiently to the receiver as side information. Their technique is described below.

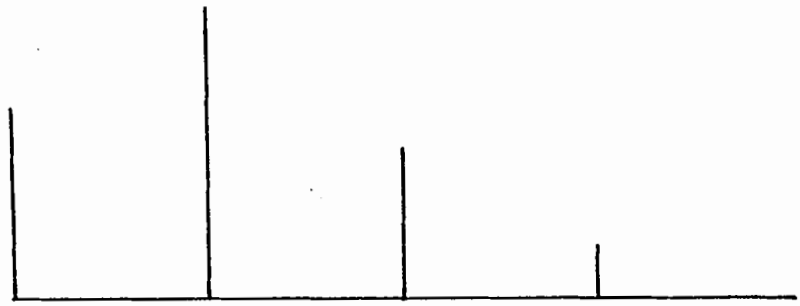
3.1.1 Smoothing Technique

The method of Zelinski and Noll is a smoothing technique. It attempts to take advantage of the similarity of adjacent transform coefficients.

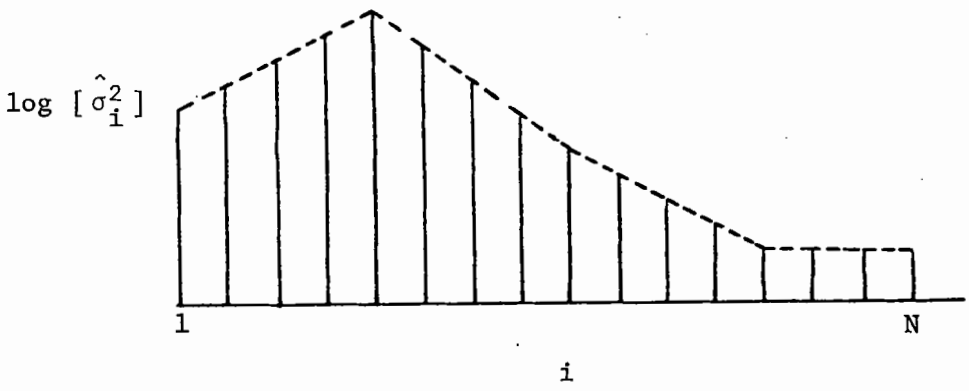
With reference to Figure 3.1, a crude variance estimate is generated by squaring the DCT coefficients of a specific block. The number of squared



(a)



(b)



(c)

Figure 3.1 (a) DCT spectrum, (b) support values, (c) basis spectrum estimate.

values is reduced by averaging over adjacent values. By the averaging operation it is hoped that the remaining values, called support values, will be better estimates of their respective coefficient variances. This method, therefore, assumes a certain smoothness in the basis spectrum of the signals under consideration. The support values comprise the side information which is then sent to the receiver. At the receiver the remaining basis spectrum values are calculated by linearly interpolating between the logarithm of the support values. Interpolation between the logarithms of the values results in a smoother basis spectrum estimate than using the support values directly.

3.1.2 All-Pole Model Technique

We propose on alternative technique for basis spectrum estimation which involves modelling the basis spectrum by an all-pole model. This type of spectral modelling is well known in linear predictive coding (LPC). A spectrum is approximated as

$$\hat{\sigma}_i^2 = \frac{1}{\left| P(\exp[j\omega_i]) \right|^2}$$

where $(\hat{\sigma}_i)^2$ is an estimate of the variance of the i-th coefficient, P is a polynomial of order M, and ω_i is the normalized radian frequency corresponding to coefficient i.

$$\omega_i = \frac{\pi(i-1)}{N} \qquad i = 1,2,\ldots,N$$

The polynomial order is selected according to the number of peaks (formants) expected in the spectrum. The coefficients of the polynomial are chosen so that the MSE between the model and the given spectrum is minimized.

The all-pole model of the spectrum has been found to be a good representation for the spectral envelope of speech sounds. Complex pole pairs give rise to peaks in the spectrum which correspond to formants in speech sounds. LPC based on the all-pole model is a viable coding scheme in its own right. Markel and Gray in [18] discuss linear predictive coding of speech and give efficient algorithms for the computation of the best fit polynomial (Levinson's recursion).

The procedure for using the all-pole model in ATC[1] is as follows. The DCT coefficients of a block are squared as in the smoothing technique. This gives a simple but crude estimate of the basis spectrum. The estimate is inverse (discrete) Fourier transformed to obtain an autocorrelation-like function. The function is then used to obtain the best M-th order polynomial fit in an MSE sense [18]. The M coefficients of the polynomial then form a compact description of the basis spectrum and are transmitted as side information to the receiver. A practical advantage of this representation is that efficient methods of quantizing the polynomial coefficients have been developed for LPC [19].

[1] Tribolet and Crochiere have recently proposed the same technique [8].

Modifications to the all-pole model technique will be described in Chapter IV. These modifications will be introduced to combat undesirable quantization effects.

IV Coder Simulation and Results

In this chapter we describe a computer simulation of ATC coders of the type discussed, giving details of the experimental setup and the input signal characteristics.

4.1 Experimental Setup

The experimental investigation of coding techniques is facilitated by computer simulation of the coder. The flexibility of a software implementation permits easy modification coder parameters and experimental optimization. For example, tradeoffs between complexity and performance can thus be thoroughly and inexpensively studied before resorting to a less flexible hardware design.

The simulation procedure requires three steps. The first step is to digitize the analogue signal and place the digitized signal in secondary storage (e.g. on a random access device like a disk). The software then simulates the coder action in non-real-time, producing output which is again placed in storage. The third step regenerates the processed analogue signal in real-time from the stored digital signal.

The simulation procedure outlined above is performed on a PDP-11/45 mini-computer. A 15 bit A/D, D/A converter combination allows for real-time signal acquisition and playback under computer control. The facility offers variable sampling rate as well as analogue processing (amplifiers, analogue filters) on input and output. The coder simulation

runs in approximately 100 times real-time. Appendix B gives a listing of the FORTRAN programs and subroutines relevant to the simulation.

All of the coder simulations to be described embody two basic assumptions. The first of these is that the transmission channel is error-free; in practice the output from the source coder is further protected against transmission errors by channel coding. The second assumption is more closely related to ATC in particular. We assume that the problem of coding and quantizing the side information can be separated from the coding of the transform coefficients. Therefore, in the simulation, the side information is not quantized. We do need, however, an estimate of the extra bit rate required to send this side information if meaningful comparisons with other coders are desired. For the smoothing technique, we refer to an estimate of the side information rate in [4] and restrict 2 kb/sec of any bit rate stated to be allocated to side information transmission. For schemes based on the all-pole model, we use the same estimate. Linear predictive coders, which use the all-pole model, can perform reasonably well at 2.4 kb/sec. Considering that in this case, we want to code only the side information, 2 kb/sec seems a generous estimate of the rate needed. We note in passing that quantization techniques developed for LPC (reflection coefficients, log area coefficients, etc.) are directly applicable to coding the side information for the all-pole model case.

The simulation uses speech waveforms as input. The scope of the experiment comprises the processing of 2 utterances each spoken by 3 speakers (2 male, 1 female). The coder rates considerated are 9.6, 12,

and 16 kb/sec. The input speech, in all cases, is sampled at a rate of 8000 samples/sec. The samples are then band-pass filtered by an FIR (finite impulse response) digital filter. The pass-band is 200 Hz to 3200 Hz which approximates the frequency response of a telephone channel. Block sizes of N=128 (16 msec) and N=256 (32 msec) are utilized in the ATC processing. Additionally the input signals can undergo pre-processing operations such as pre-emphasis and/or windowing.

4.2 Simulation Results

The simulation results are presented in three sub-sections. The first deals with ATC combined with the smoothing technique of 3.1.1. The second deals with ATC in connection with the all-pole technique of 3.1.2. The third sub-section contains a number of modifications to and refinements of the basic scheme.

4.2.1 ATC using the Smoothing Technique

We examine the performance of ATC by several means. First, we present plots of waveforms at various points in the coder for a given input block of speech samples. Second, coder performance is measured objectively by using segmental SNR as a distortion measure. Third, the coders are evaluated subjectively according to the results of informal listening tests.

The input block of speech samples is taken from the middle of an utterance. It is a high energy voiced segment from the vowel part of the

word "depth". At this point in the utterance the pitch is constant. Formants 1,3, and 4 are steady, with formant 2 dropping from about 1500 to 1000 Hz.

Figure 4.1 (a) shows the DCT spectrum for a particular input block. Recall that the DCT is not equal to the Fourier spectrum. It is, however, bounded by the Fourier spectrum. For future reference the unqualified term spectrum implies the DCT spectrum. From the figure, we see that the spectrum exhibits a formant structure and a pitch structure analogous to the Fourier spectrum. Note the presence of energy below 200 Hz and above 3200 Hz resulting from the fact that the input block is obtained from the input signal by rectangular windowing.

The basis spectrum estimate, derived by the smoothing technique of 3.1.1, is illustrated in Figure 4.1 (b). Just 16 support values determine the curve. The interpolated segments appear as straight lines on the logarithmic dB scale. Notice that the estimate captures the gross spectral structure. It does not, however, follow the fine structure. Also the fit is poor at low frequencies. Figure 4.2 (a) superimposes the estimate and the signal spectrum.

We next consider the bit assignment based on this estimate. For illustrative purposes, we use a bit rate of 9.6 kb/sec. This rate allows for sufficiently coarse quantization to make the coder effects obvious. For this example 9.6 kb/sec is equivalent to 121 bits/block (block size = 128 samples). Referring to Figure 4.3 (a), we note the step structure of the bit assignment curve. Also note that no bits are assigned above a

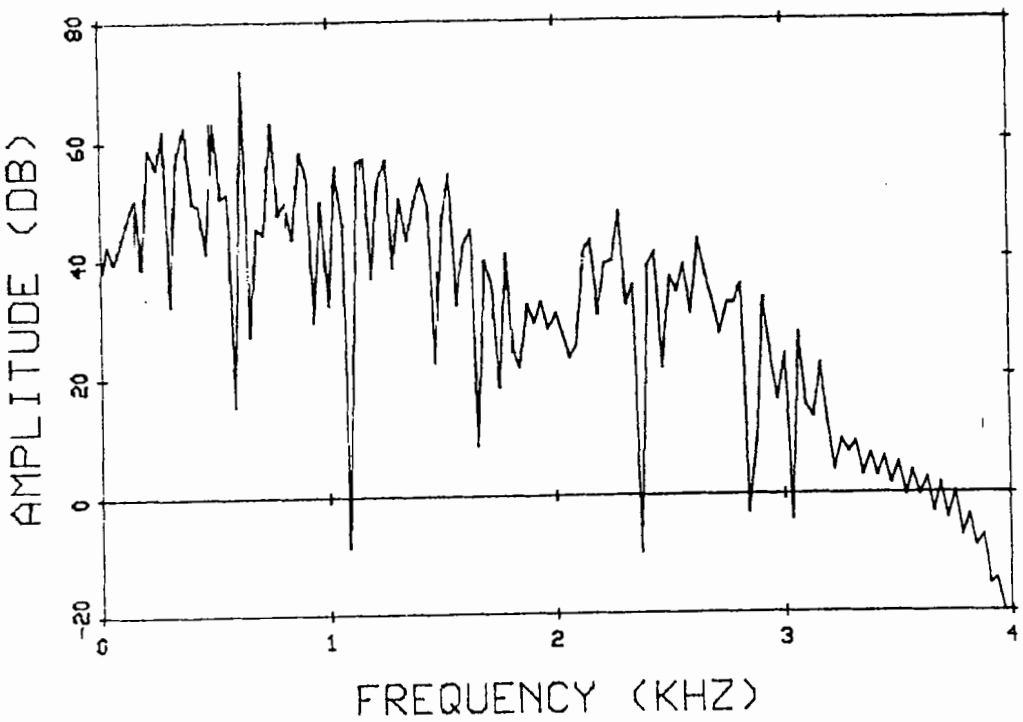


Figure 4.1(a) Input DCT spectrum (0 dB is arbitrary on all logarithmic plots).

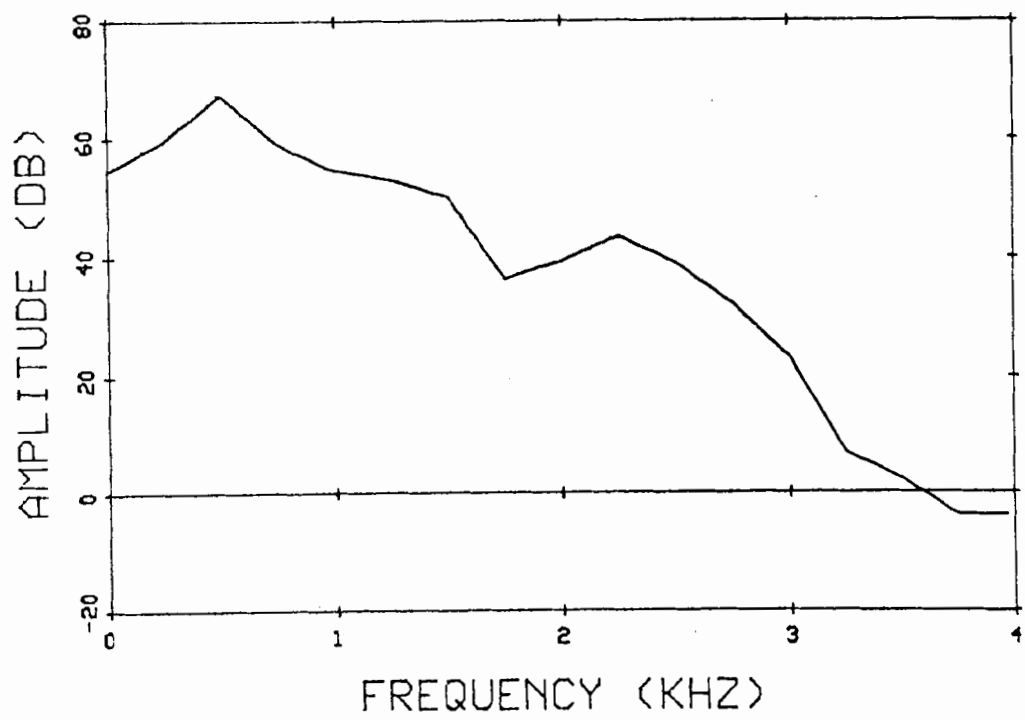


Figure 4.1(b) Basis spectrum estimate using the smoothing technique of 3.1.1.

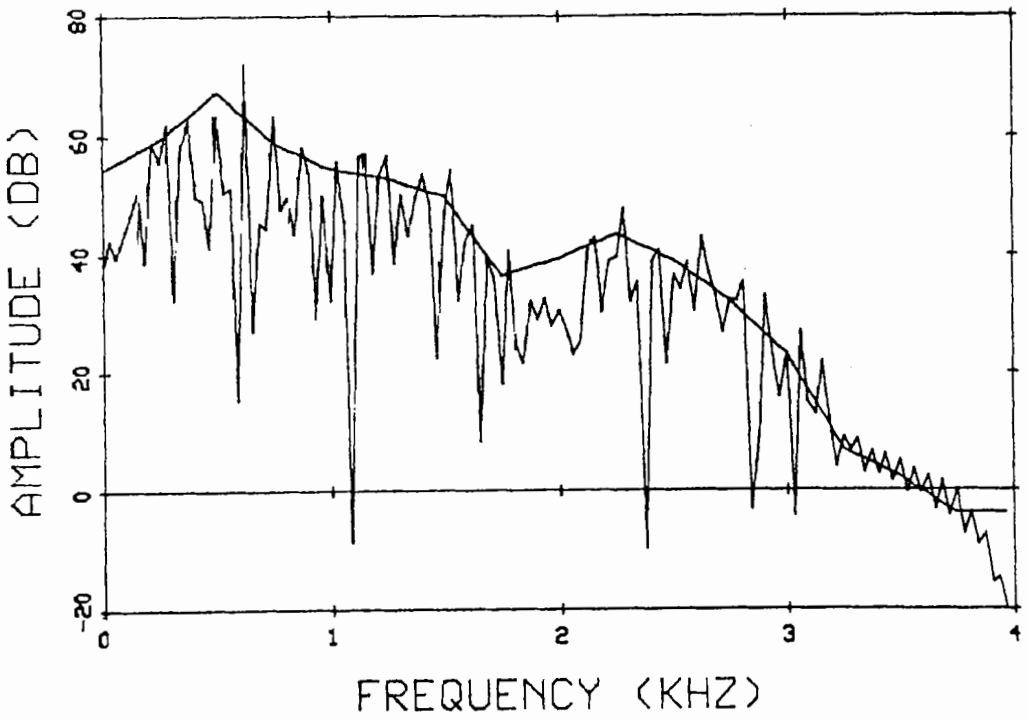


Figure 4.2(a) Superimposed plot of input DCT spectrum and estimate.

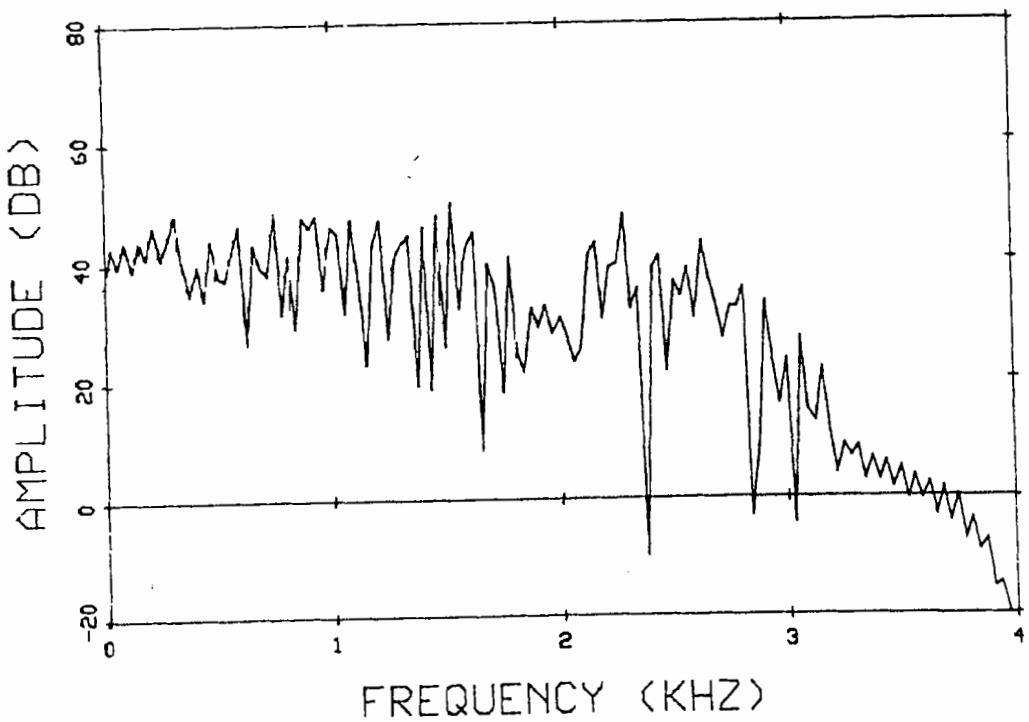


Figure 4.2(b) Quantization noise spectrum.

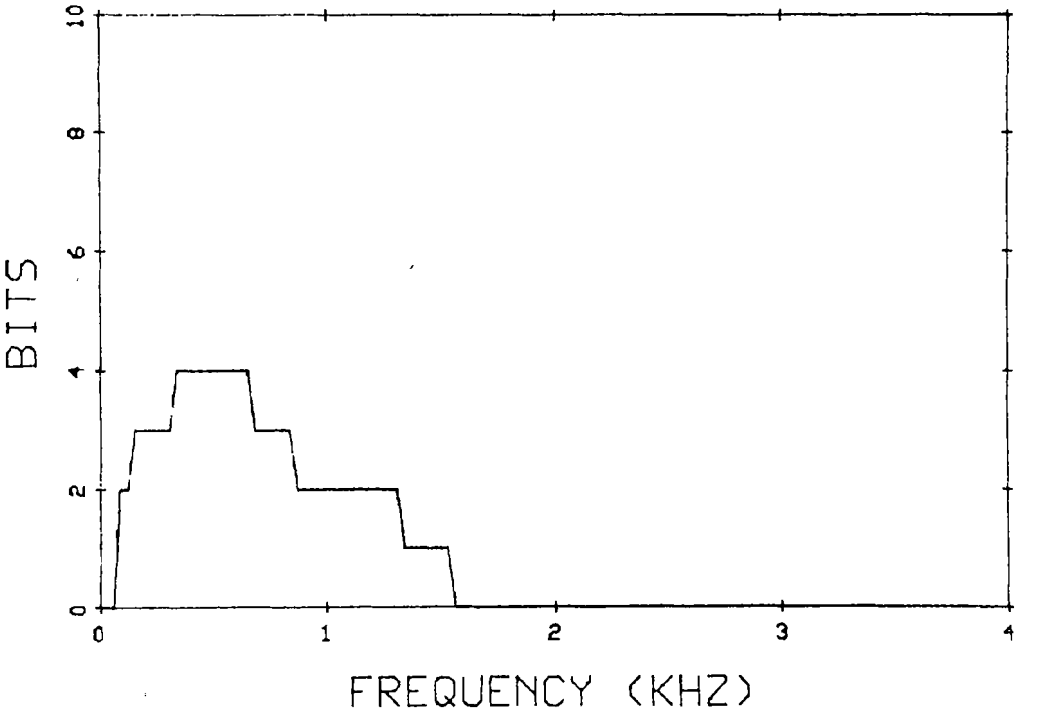


Figure 4.3(a) Bit assignment curve.

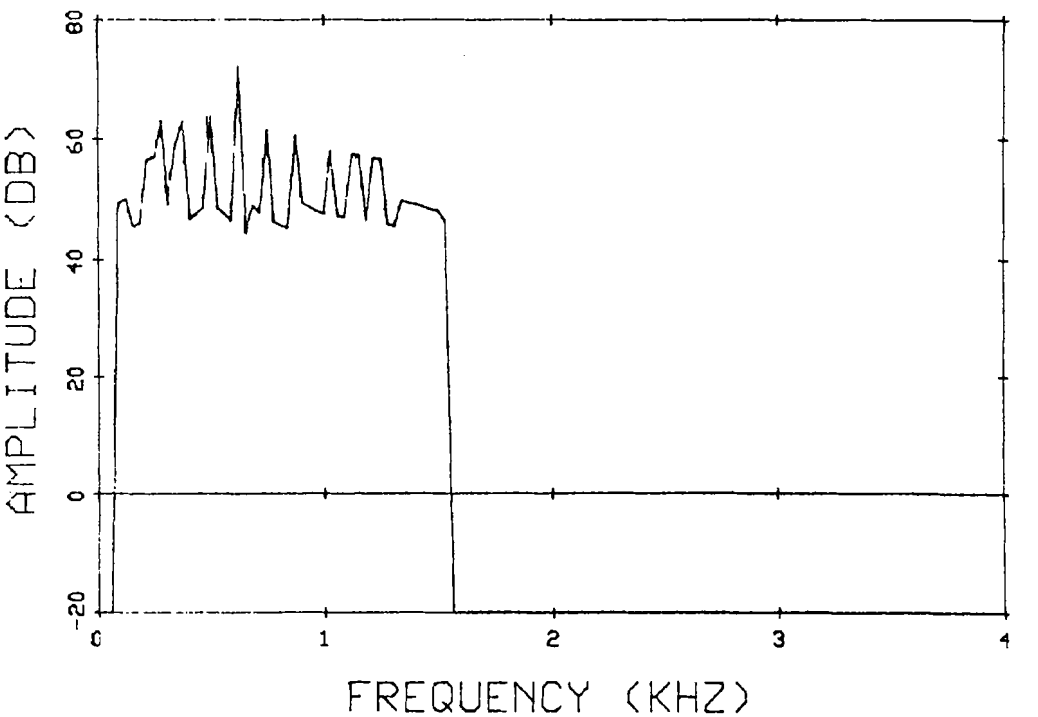


Figure 4.3(b) Receiver DCT spectrum.

frequency of 1.6 kHz. This effect is caused by the shortage of bits to be allocated, coupled with the large dynamic range in the spectrum. The small values of the estimate at high frequencies restrict the bit assignment to the more energetic low frequency coefficients.

The spectrum after quantization $\underline{\hat{Y}}$, that is the receiver spectrum, is shown in Figure 4.3 (b). The energy distribution strongly reflects the bit assignment in that coefficients assigned zero bits are quantized to zero. Because of the adaptive quantization, coefficients assigned even a single bit are present at the receiver. Figure 4.2 (b) shows the spectrum of the quantization noise. Theoretically the coefficients should undergo equal distortion in regions where bits are assigned. In practice, errors in the basis spectrum estimate and the integer bit assignment constraint give the noise spectrum a fine structure.

Figures 4.4 (a) and 4.4 (b) are the input and receiver time waveforms. Figure 4.5 (a) superimposes these two directly and Figure 4.5 (b) is the error waveform. Note the large errors near 0 and 16 msec, that is, at the block boundaries.

We now examine coder performance averaged over different sentences and speakers. The objective measures are SNR and segmental SNR (SEGSNR) [12]. Segmental SNR attempts to eliminate a bias in the conventional SNR measure. Conventional SNR tends to give more weight to high amplitude segments. Segmental SNR tries to alleviate this bias by averaging SNR (expressed in dB) over short intervals (10-30 msec). Let $\underline{X}_k$ be the k-th input block of time samples, $\underline{\hat{X}}_k$ the k-th decoded block, and K the number

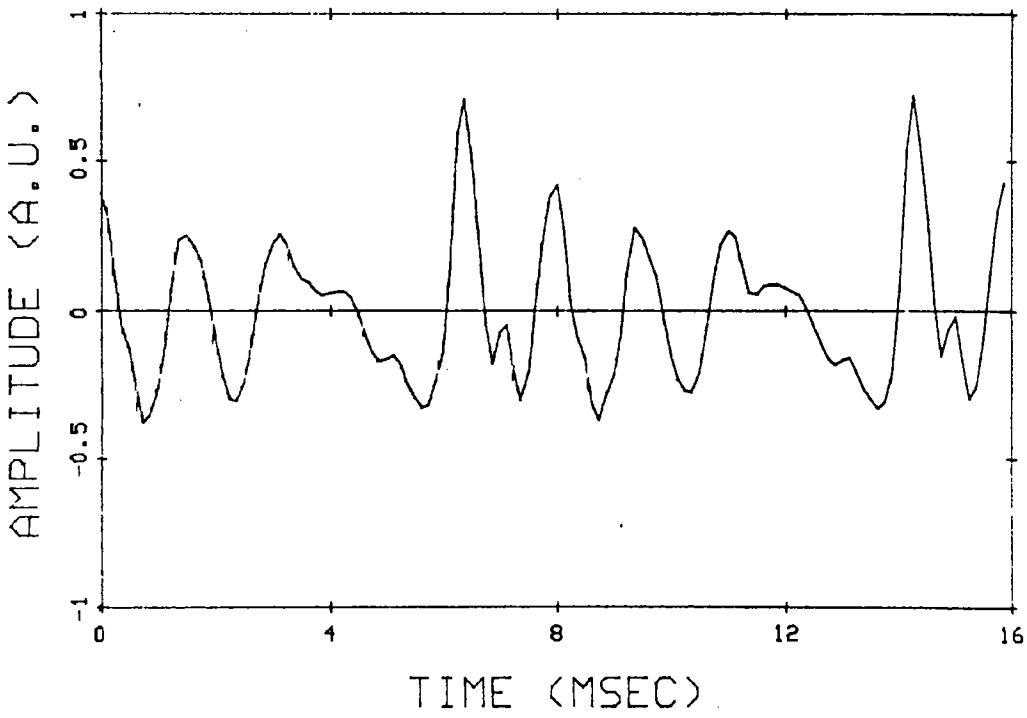


Figure 4.4(a) Input time waveform (A.U. denotes arbitrary units).

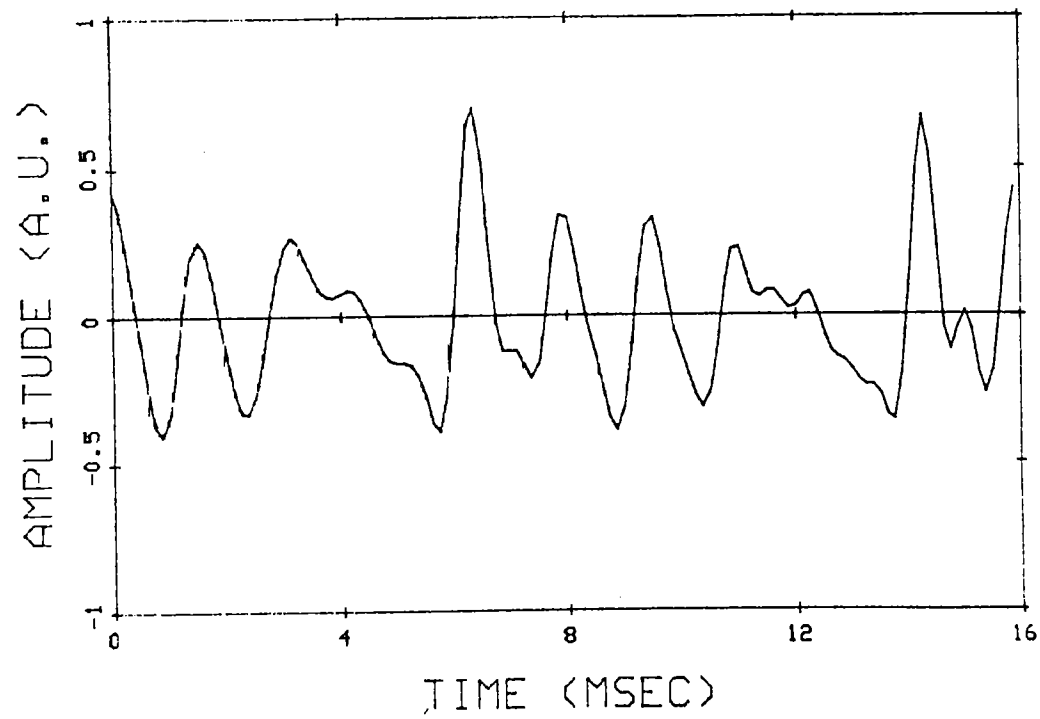


Figure 4.4(b) Receiver time waveform.

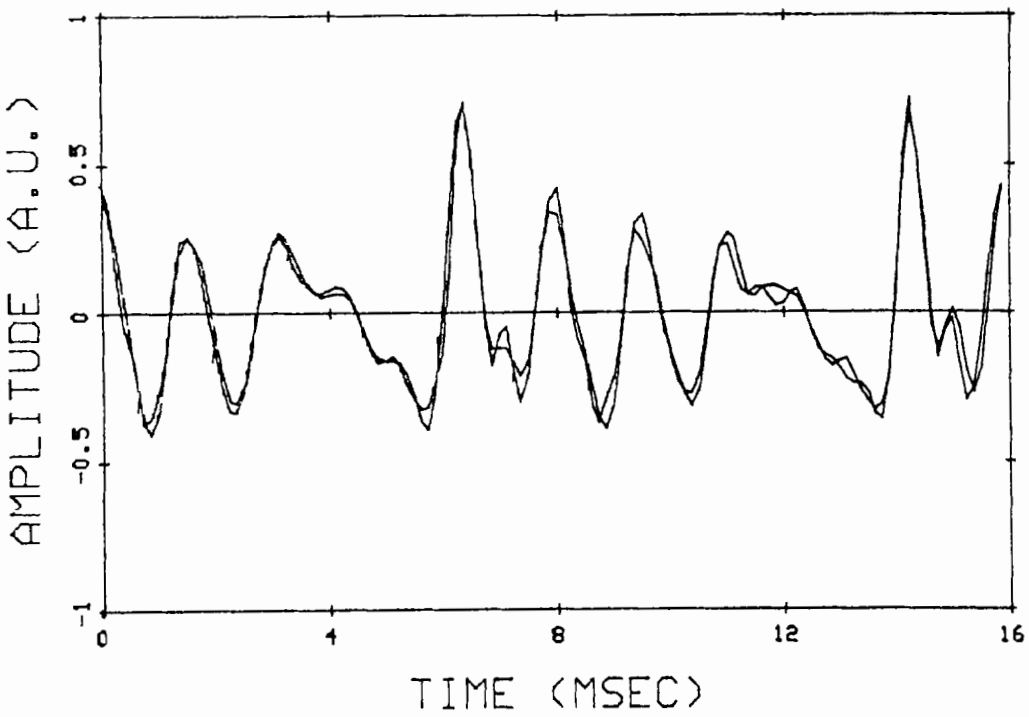


Figure 4.5(a) Superimposed plot of input and receiver time waveforms.

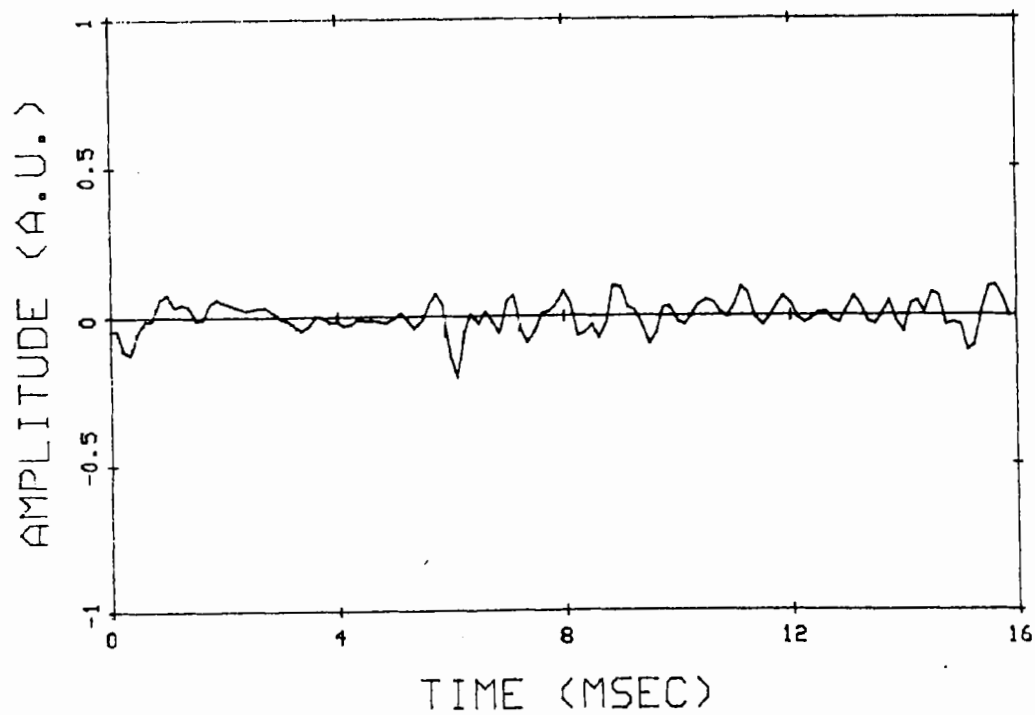


Figure 4.5(b) Quantization error time waveform.

of blocks in an utterance, then

$$\text{SNR} \equiv 10 \log_{10} \left[\frac{\sum_{k=1}^{K} |\underline{X}_k|^2}{\sum_{k=1}^{K} |\underline{X}_k - \hat{\underline{X}}_k|^2} \right]$$

and $$\text{SEGSNR} \equiv 10 \log_{10} q^2$$

with $$\log_{10} (1+q^2) = \frac{1}{K} \sum_{k=1}^{K} \log_{10} \left[1 + \frac{|\underline{X}_k|^2}{|\underline{X}_k - \hat{\underline{X}}_k|^2} \right]$$

Results using both measures are presented in tabular format in Table 4.1. Figures 4.6 (a) and 4.6 (b) are plots of SEGSNR versus bit rate. With reference to Figure 4.6 (a) we note that the SEGSNR drops almost linearly with bit rate. At a given rate, differences between speaker can account for up to 1½ dB variation in SEGSNR.

Also from Figure 4.6 (b), a change in SEGSNR of up to 2½ dB can occur between the two sentences used in the experiment. Average SEGSNR values are 11.0, 13.0 and 15.9 dB for rates of 9.6, 12, and 16 kb/sec respectively.

SNR (dB) / SEGMENTAL SNR (dB)

Speaker	Sentence	9.6 kb/sec	12 kb/sec	16 kb/sec
1	A	9.9 / 11.6	12.0 / 14.0	14.1 / 17.3
	B	9.5 / 9.2	11.0 / 10.9	12.6 / 13.6
2	A	9.5 / 12.1	10.9 / 14.2	13.0 / 17.1
	B	8.1 / 10.8	9.1 / 12.6	10.5 / 15.2
3	A	11.8 / 12.2	13.6 / 14.2	16.2 / 17.2
	B	10.1 / 10.1	11.5 / 11.9	13.2 / 14.7
Overall Average:		9.8 / 11.0	11.4 / 13.0	13.3 / 15.9
By Speaker:				
1		9.7 / 10.4	11.5 / 12.5	13.3 / 15.5
2		8.8 / 11.5	10.0 / 13.4	11.7 / 16.2
3		10.9 / 11.1	12.6 / 13.0	14.7 / 16.0
By Sentence:				
	A	10.4 / 12.0	12.2 / 14.1	14.4 / 17.2
	B	9.2 / 10.0	10.5 / 11.8	12.1 / 14.5

Speaker 1: Male
Speaker 2: Female
Speaker 3: Male

Sentence A: It's easy to tell the depth of a well.
Sentence B: The birch canoe slid on the smooth planks.

Table 4.1 SNR Performance of the Smoothing Technique

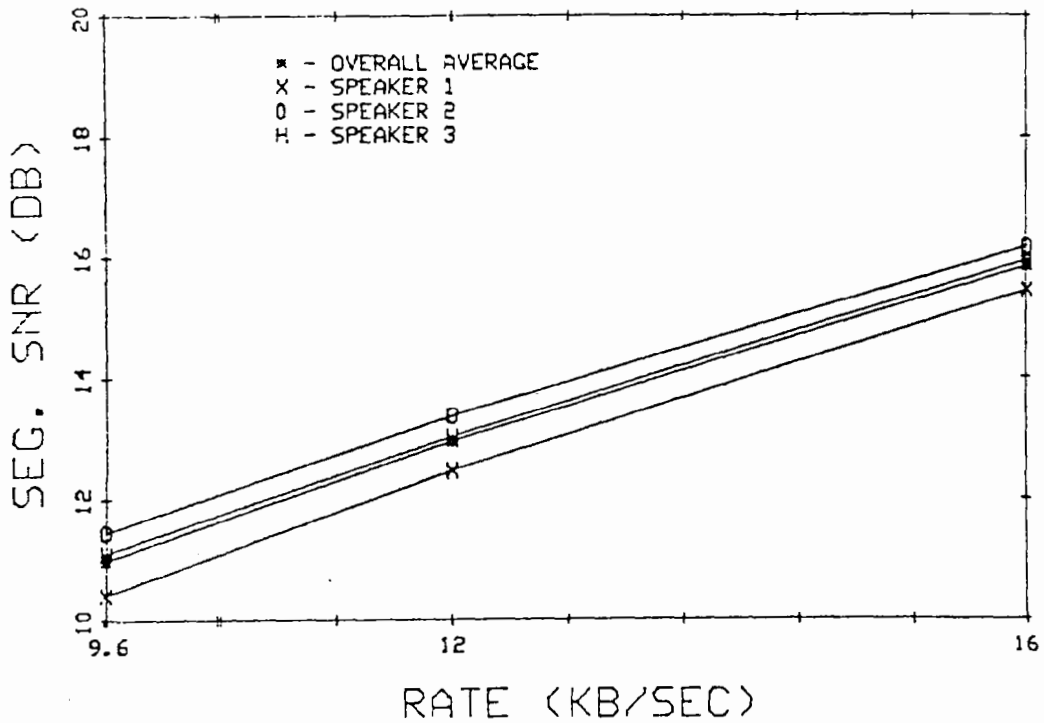


Figure 4.6(a) Sensitivity of the smoothing technique to speaker variation.

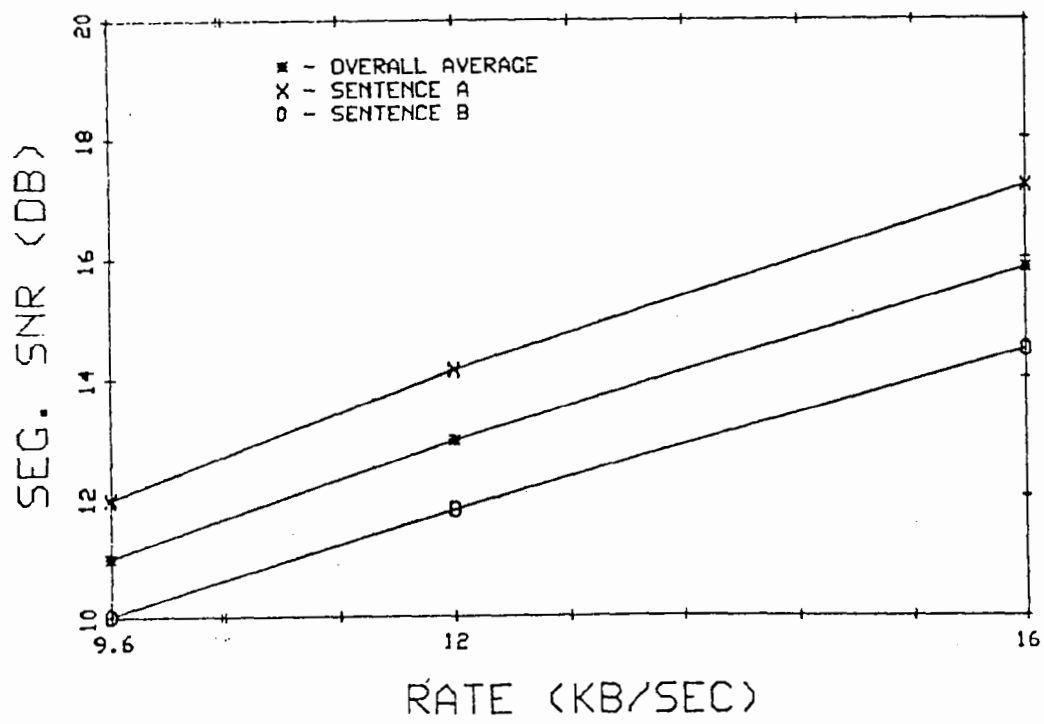


Figure 4.6(b) Sensitivity of the smoothing technique to sentence material.

The SNR results have been supplemented by subjective evaluation to censure a realistic assessment of signal quality. The subjective tests involved seven experienced listeners.

The distortions in the coded speech were described as rough or harsh. The distortions are annoying but are perceptually distinct from the speech material and thus do not affect intelligibility. The distortions increased with decreasing bit rate. At 9.6 kb/sec, the speech exhibits a warbling noise which makes the speech sound unnatural. High level clicks are clearly audible. In addition, for some speakers, the speech sounds muffled due to the loss of high frequency components.

4.2.2 ATC using the All-Pole Model Technique

We now examine ATC in combination with the all-pole model technique of Section 3.1.2. The samples contained in the input block of size 256 include the 128 samples of the previous example. In addition the block is pre-emphasized by a simple first order filter and windowed by a non-rectangular window. These refinements will be discussed in more detail in the next section.

Figure 4.7 (a) shows the time waveform. Notice that the increased block size spans 4 pitch periods for this speaker. Figure 4.7 (b) shows the corresponding autocorrelation function, which is needed to calculate the all-pole estimate of the basis spectrum. In the following section, we will see how to extract even a better estimate from this function. For the present, notice the peaks at 8 msec intervals which result from the pitch periodicity in the time waveform.

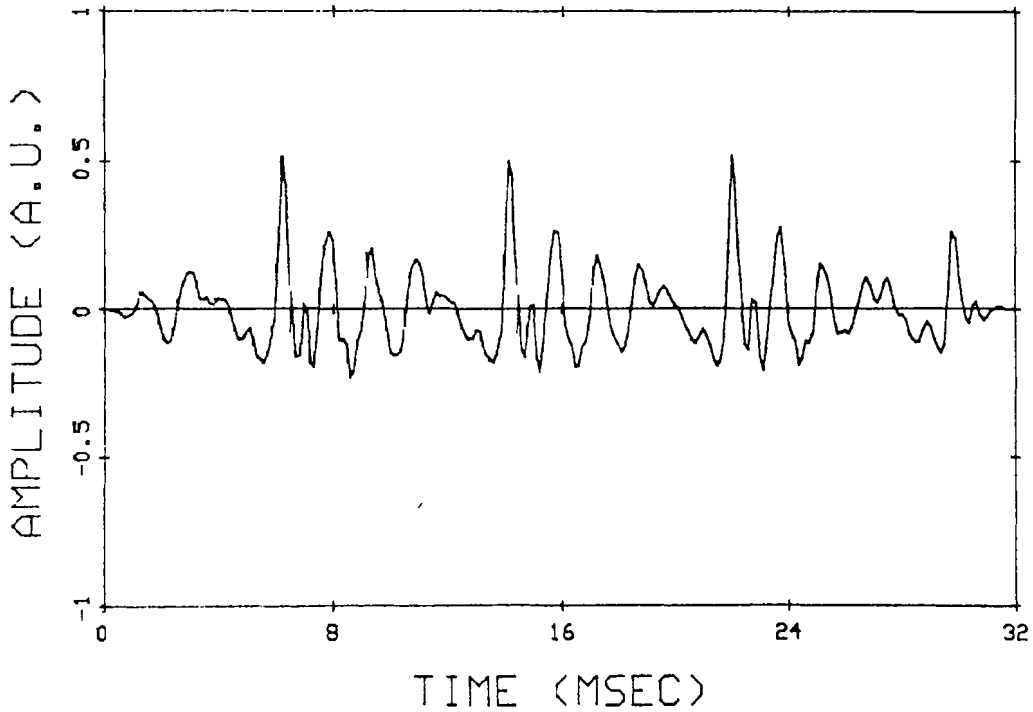


Figure 4.7(a) Input time waveform.

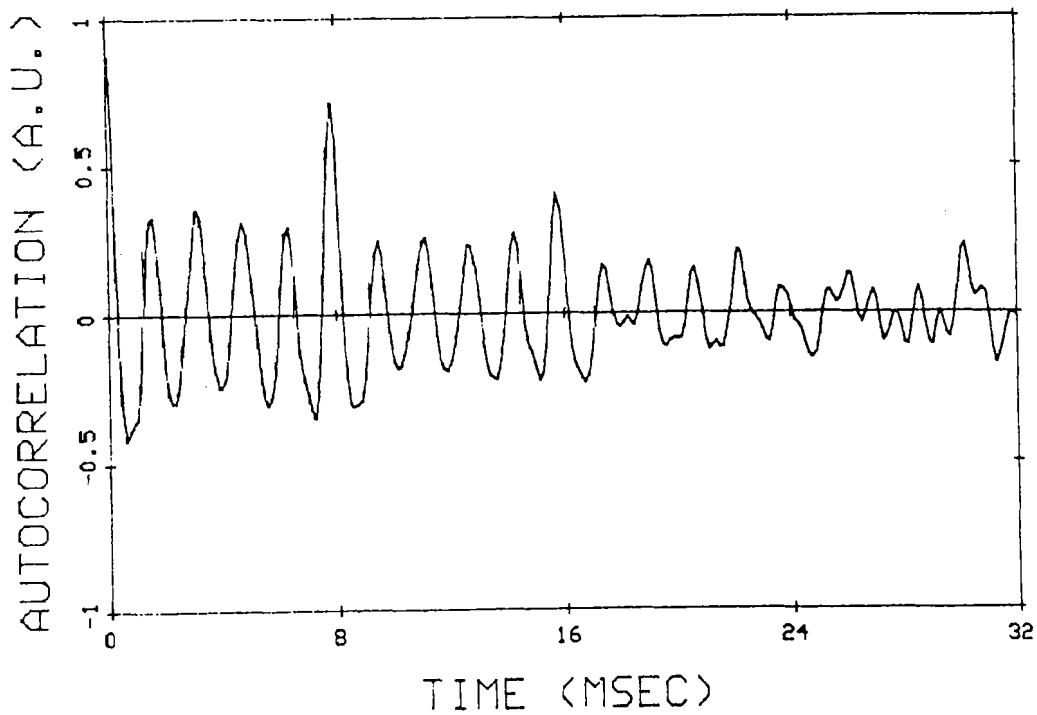


Figure 4.7(b) Autocorrelation function of Figure 4.7(a).

The block of time samples is transformed, yielding the spectrum in Figure 4.8 (a). Windowing the time samples results in less spectral spill-over in the DCT domain. We see decreased energy in the 0-200 Hz range and the near absence of energy in the 3200-4000 Hz range (cf Figure 4.1 (a)).

Figure 4.8 (b) illustrates the all-pole model estimate for this block. The analysis uses 11-th order filter (12 parameters including filter gain). The filter order is chosen to include four formants (8 poles) plus three poles to approximate the speech spectrum roll-off. Note the smoothness of the estimate and the good overall fit to the spectrum. The all-pole model is known to fit formant peaks very well. For Figure 4.9 (a) the spectrum and the estimate are superimposed. Notice that the estimate fits the valleys more poorly. The match is especially bad at the extremes of the spectrum. As the model is based on a minimum MSE formulation, it is expected to fit peaks better than valleys.

Bit assignment based on the all-pole model is shown in Figure 4.10 (a). A total of 212 bits (at 9.6 kb/sec) are assigned to the coefficients in the block. The bit assignment remains step-like. It differs from the one based on the smoothing technique in that some bits are allocated to high frequencies in the third formant region.

The spectrum at the receiver is plotted in Figure 4.10 (b). The spectrum exhibits improved high frequency response. Large spectral gaps remain, however. Figure 4.9 (b) shows the spectrum of the quantization noise.

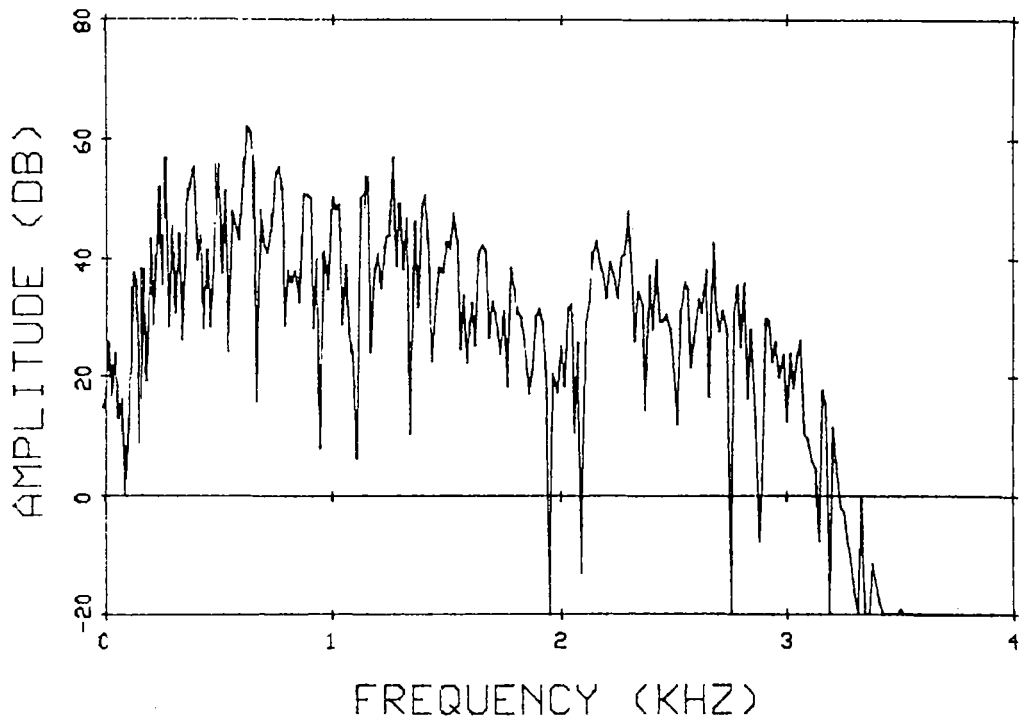


Figure 4.8(a) Input DCT spectrum.

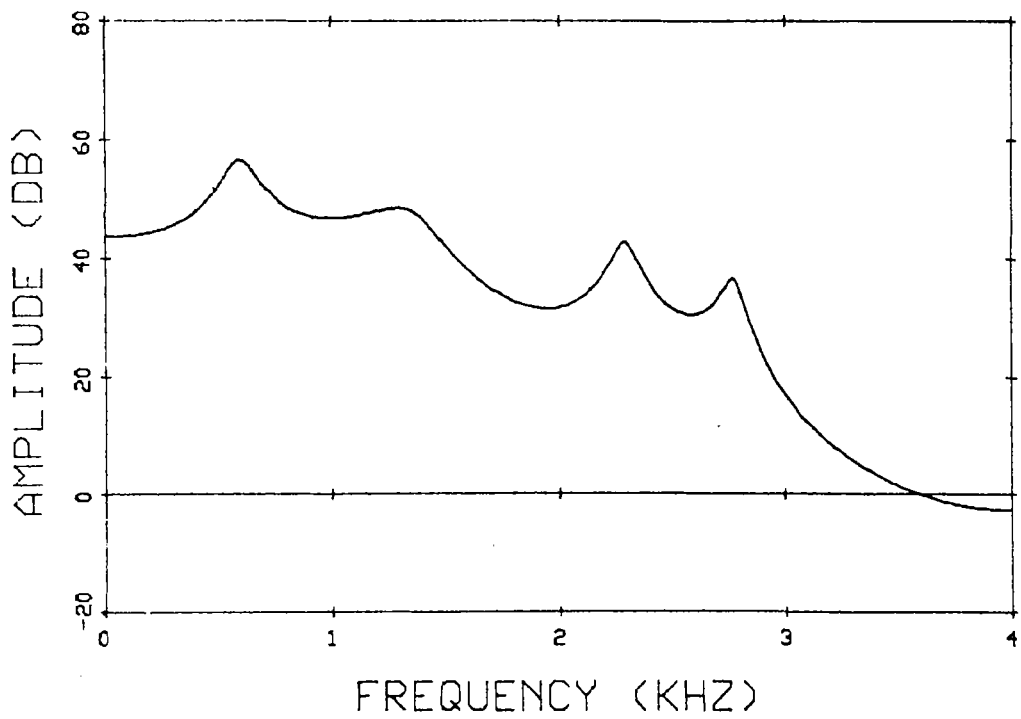


Figure 4.8(b) Basis spectrum estimate using the all-pole model of 3.1.2.

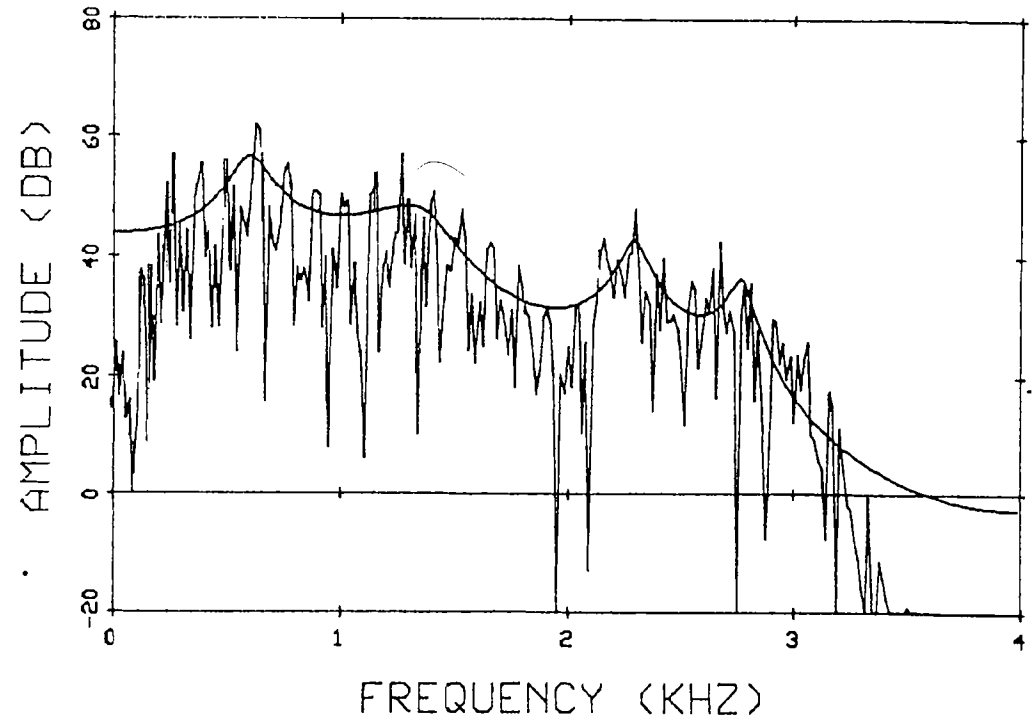


Figure 4.9(a) Superimposed plot of input DCT spectrum and estimate.

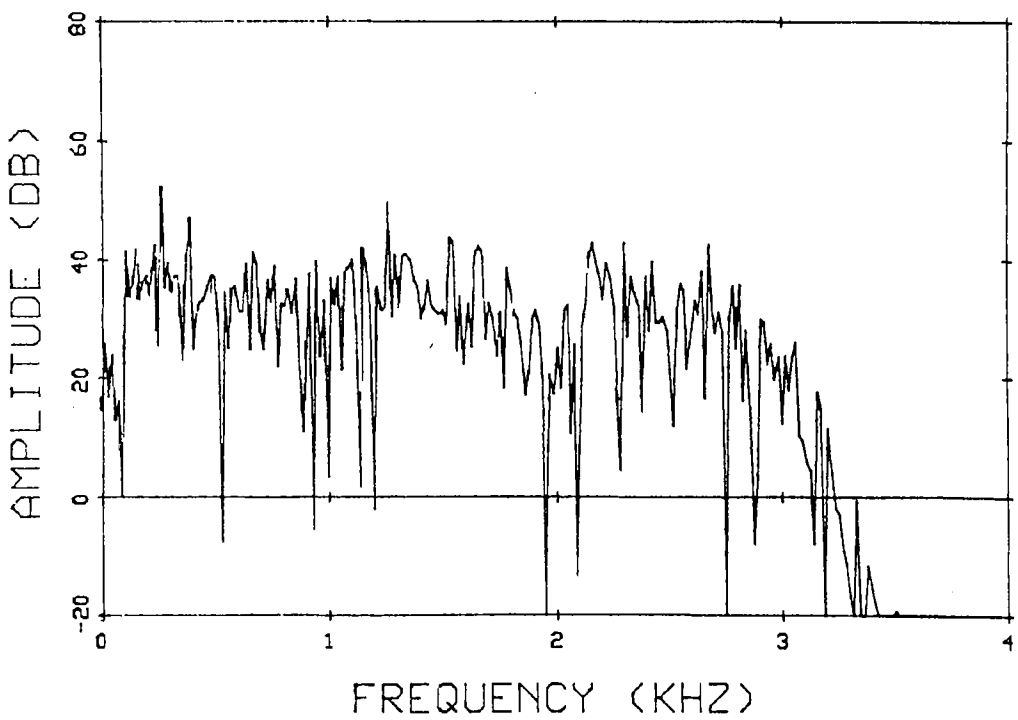


Figure 4.9(b) Quantization noise spectrum.

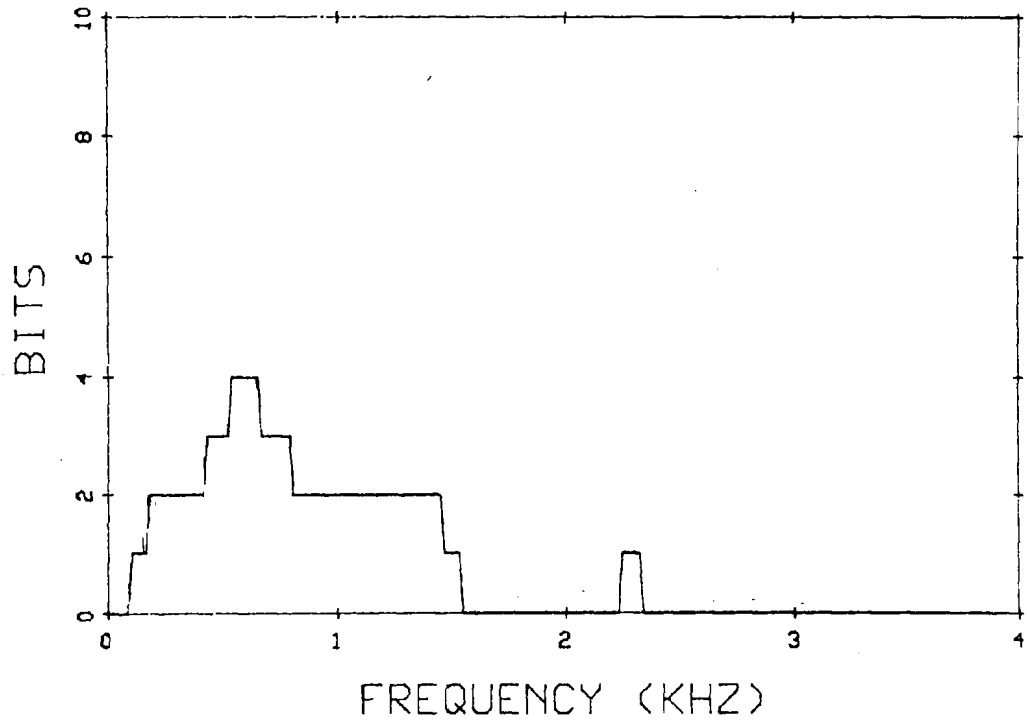


Figure 4.10(a) Bit assignment curve.

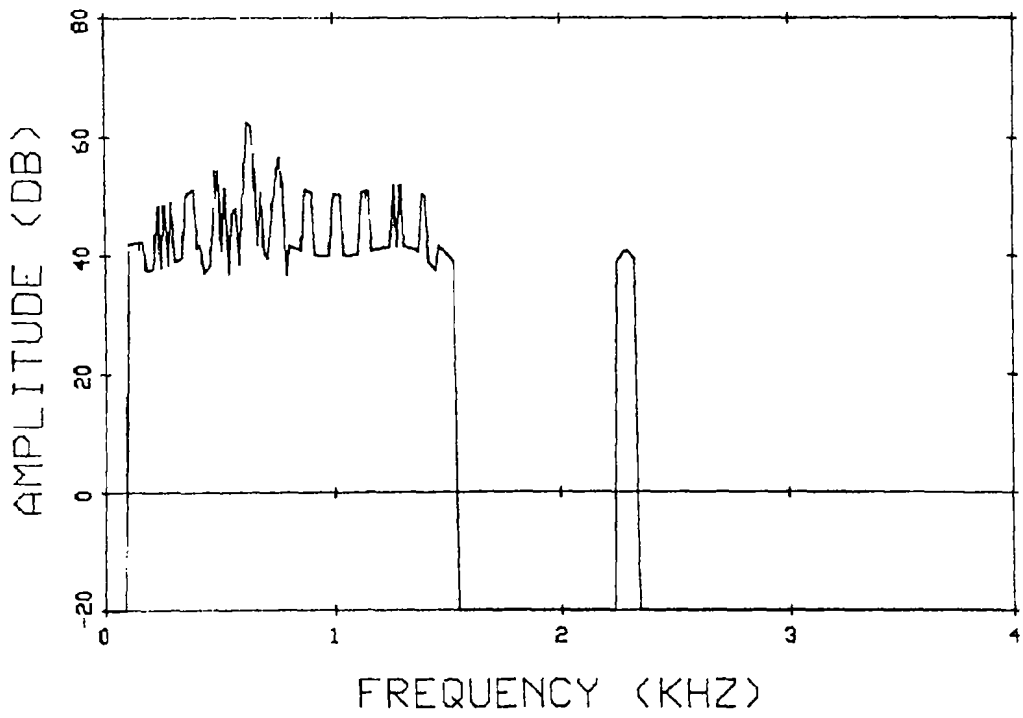


Figure 4.10(b) Receiver DCT spectrum.

We next look at the time waveforms for this scheme. In Figure 4.11 (a) we have repeated the input waveform for comparison with the receiver waveform in Figure 4.11 (b). Figure 4.12 (a) compares these waveforms on the same graph. Figure 4.12 (b) is a plot of the error waveform. Referring to Figure 4.12 (b), note the reduced error near the ends of the block (cf Figure 4.5 (b)). This is another consequence of the windowing process.

We now change our focus from a particular speech segment to the overall performance of the all-pole model scheme. SNR values are listed in Table 4.2. Figure 4.13 (a) illustrates performance as a function of the speaker. We see that the SEGSNR is relatively insensitive to the different speakers. The maximum deviation is about $\frac{1}{2}$ dB. Figure 4.13 (b) shows the performance variation as a function of sentence material. A difference of up to $2\frac{1}{2}$ dB is apparent. Thus this scheme exhibits approximately the same sensitivity to sentence variation as the smoothing technique. Overall average SEGSNR values are 11.2, 13.1, and 16.0 dB for 9.6, 12 and 16 kb/sec, about the same as the smoothing technique.

Subjectively, the all-pole model scheme exhibits different distortions when compared to the smoothing technique. The coded speech contains a background swishing or whistling sound. This distortion is closely correlated with the speech waveform and as such is not perceptually distinct from the speech. Starting at 12 kb/sec listeners notice a reverberant speech quality and some muffling. At 9.6 kb/sec a warble or burble is present. Most listeners prefer this scheme to the smoothing technique.

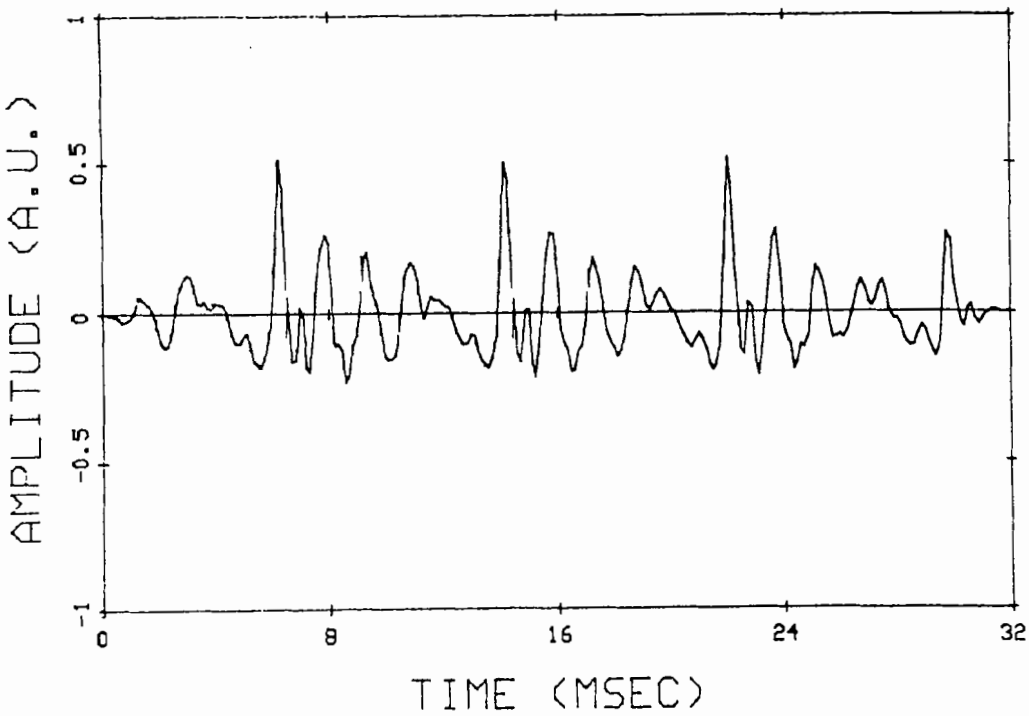


Figure 4.11(a) Input time waveform (same as Figure 4.7(a)).

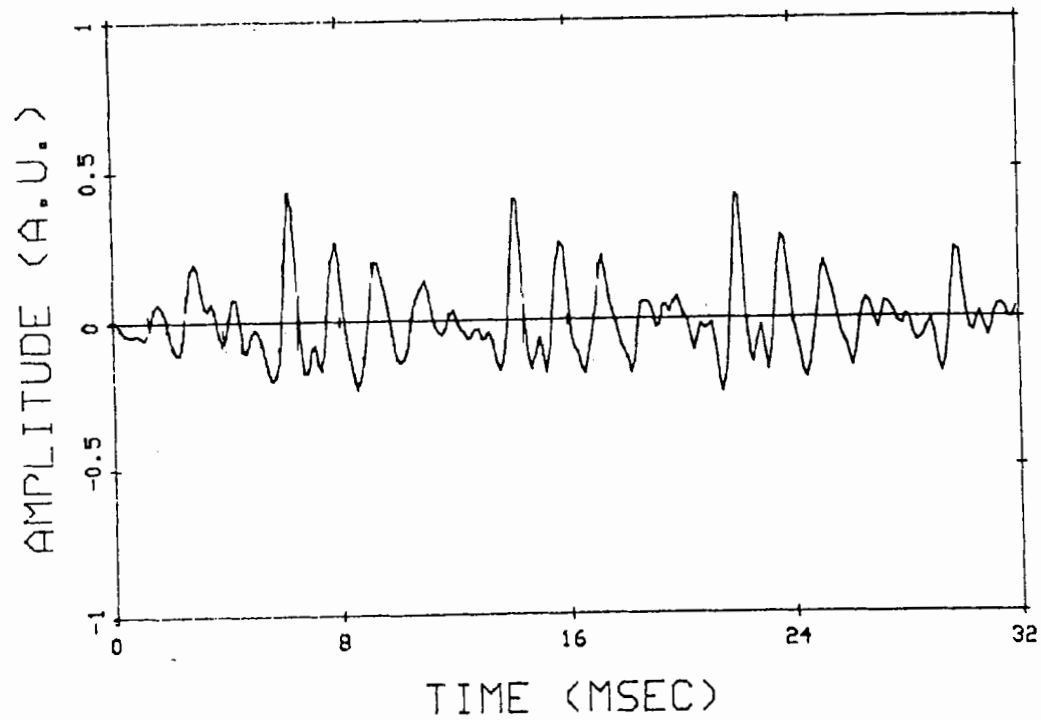


Figure 4.11(b) Receiver time waveform.

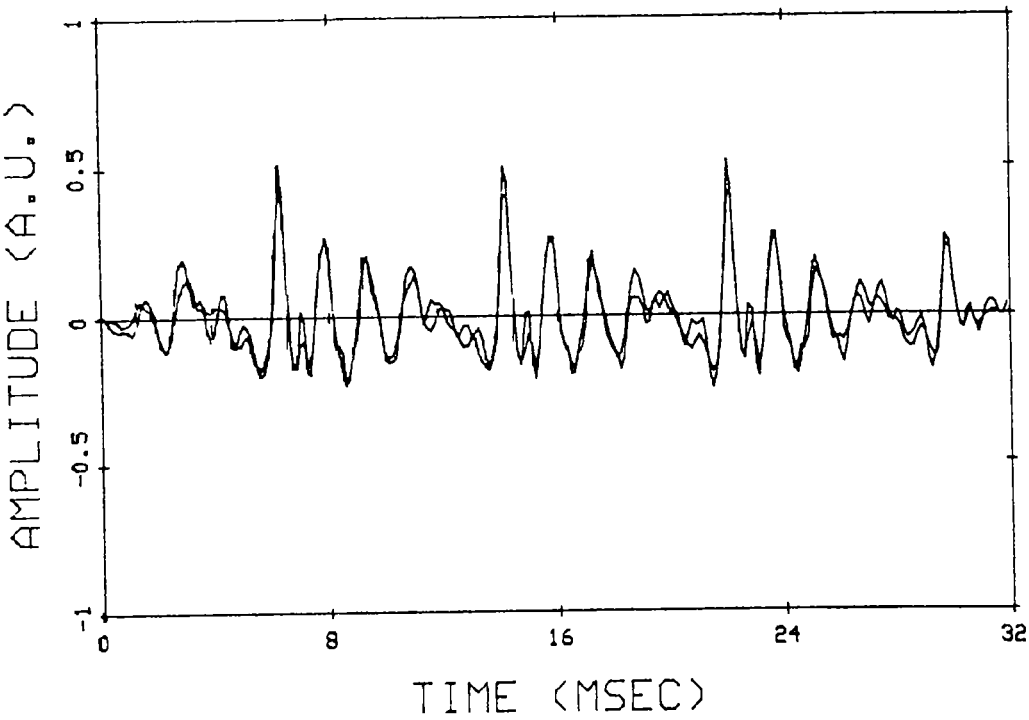


Figure 4.12(a) Superimposed plot of input and receiver time waveforms.

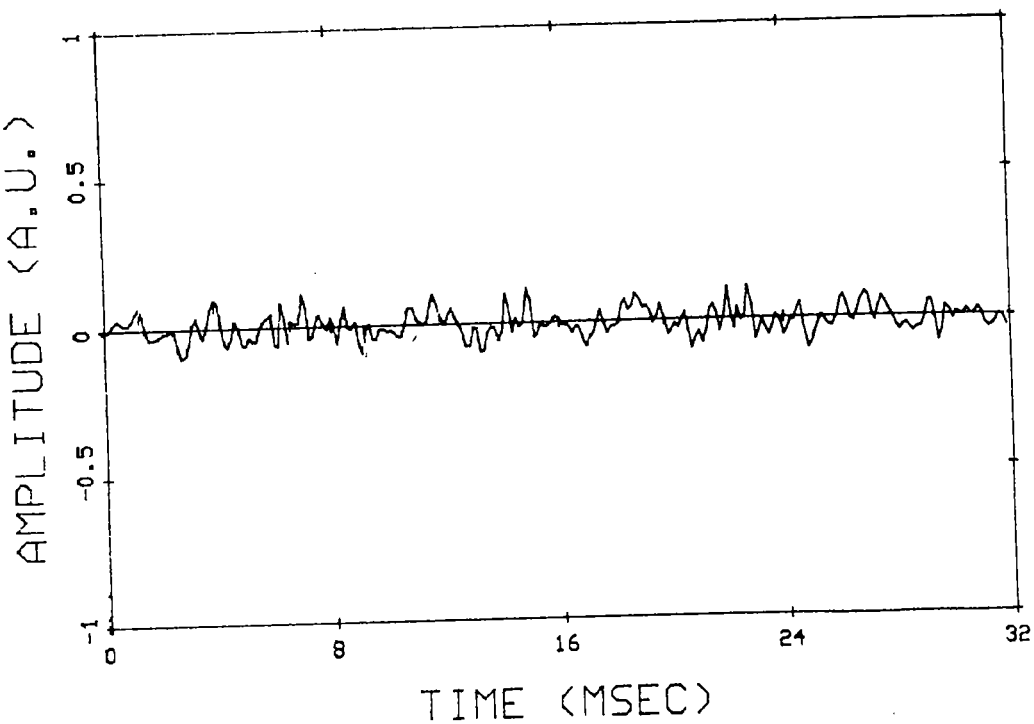


Figure 4.12(b) Quantization error time waveform.

		SNR (dB) / SEGMENTAL SNR (dB)		
Speaker	Sentence	9.6 kb/sec	12 kb/sec	16 kb/sec
1	A	12.0 / 12.1	14.4 / 14.4	17.6 / 17.6
	B	11.5 / 9.9	13.8 / 11.7	17.1 / 14.6
2	A	10.1 / 12.2	11.9 / 14.1	14.8 / 17.1
	B	11.2 / 10.9	12.9 / 12.5	15.4 / 14.9
3	A	12.7 / 12.2	15.2 / 14.2	18.1 / 17.3
	B	13.6 / 10.2	15.9 / 12.0	19.2 / 14.7
Overall Average:		11.9 / 11.2	14.0 / 13.1	17.0 / 16.0
By Speaker:				
1		11.7 / 11.0	14.1 / 13.0	17.4 / 16.1
2		10.7 / 11.6	12.4 / 13.3	15.1 / 16.0
3		13.2 / 11.2	15.5 / 13.1	18.7 / 16.0
By Sentence:				
	A	11.6 / 12.2	13.8 / 14.2	16.8 / 17.4
	B	12.1 / 10.3	14.2 / 12.0	17.3 / 14.7

Speaker 1: Male
Speaker 2: Female
Speaker 3: Male

Sentence A: It's easy to tell the depth of a well.
Sentence B: The birch canoe slid on the smooth planks.

Table 4.2 SNR Performance of the All-Pole Model Technique

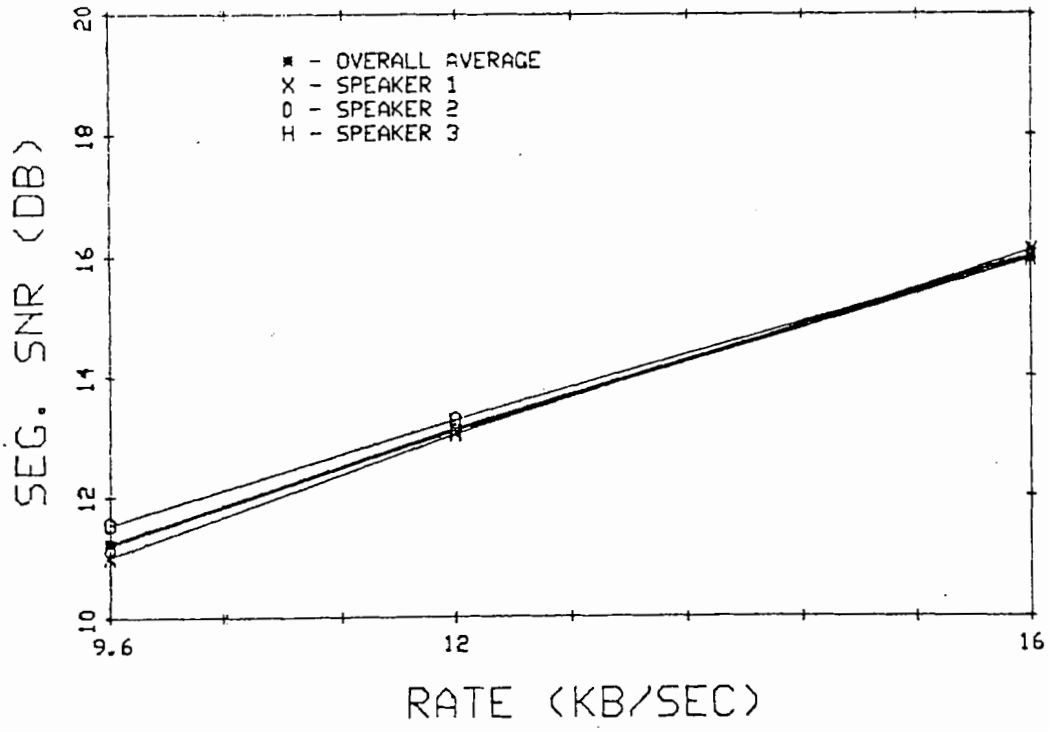


Figure 4.13(a) Sensitivity of the all-pole model technique to speaker variation.

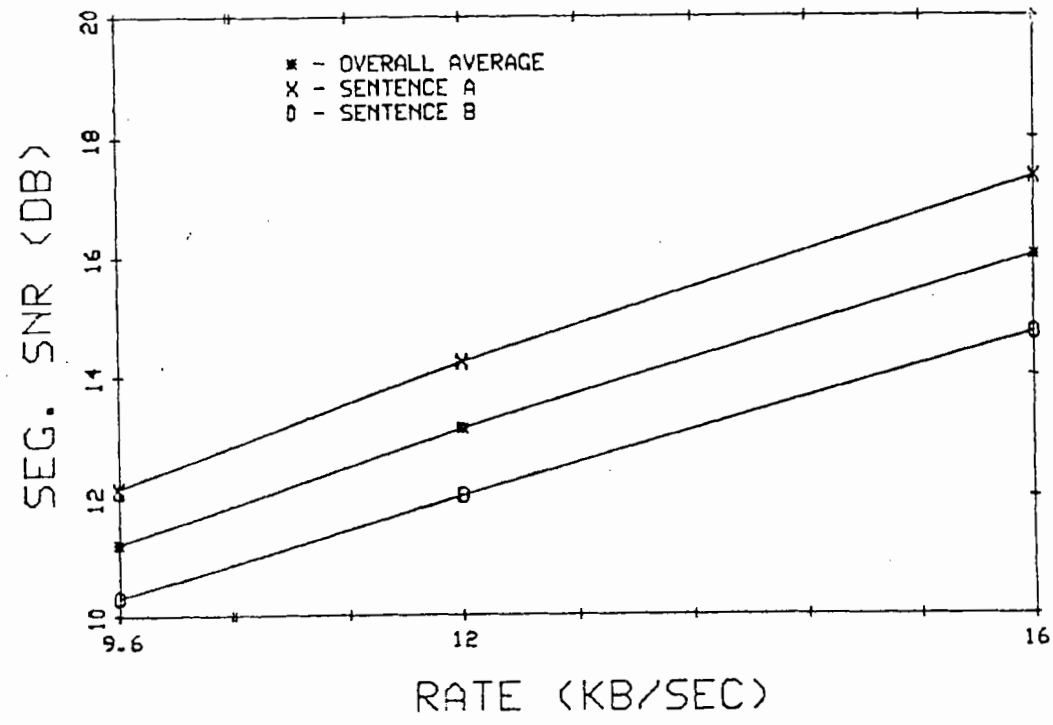


Figure 4.13(b) Sensitivity of the all-pole model technique to sentence material.

4.2.3 ATC Modifications and Refinements

We now clarify some of the modifications and refinements referred to in previous sections. To some extent several of these refinements have been incorporated into the all-pole model technique. A major modification, that of adding a pitch estimate to the basis spectrum, will be introduced and illustrated with the same example used in the previous section.

In the course of the experimental investigation it was discovered that ATC could be improved by use of pre-emphasis, de-emphasis techniques. Pre-emphasizing the input waveform boosts the energy at high frequencies relative to the energy at low frequencies. This alteration of the signal spectrum agrees in principle with the perceptual importance of high frequency formants. The result of this refinement is to cause more bits to be assigned to higher frequencies. Those high frequency components which are allocated bits are reproduced in the receiver spectrum.

The complementary operation of de-emphasis is then performed at the receiver. The filters used to accomplish pre-emphasis and de-emphasis are given by eq. (4.1) and eq. (4.2) respectively.

$$h_p(n) = \begin{cases} 1 & n = 0 \\ -\frac{1}{2} & n = 1 \\ 0 & \text{elsewhere} \end{cases} \tag{4.1a}$$

$$H_p(z) = 1 - (2z)^{-1} \tag{4.1b}$$

$$h_d(n) = 2^{-(n+1)}(1+\cos[\pi n/5]) \qquad n = 0,1,2,3,4$$
$$= 0 \qquad \text{elsewhere} \tag{4.2a}$$

$$H_d(z) \simeq \frac{1}{1-(2z)^{-1}} \tag{4.2b}$$

These filters were selected from a family of pre-emphasis characteristics with the parameter $h_p(1) = -\frac{1}{2}$ selected experimentally.

A second refinement is the use of a non-rectangular window. When a tapered window is incorporated, adjacent blocks are overlapped. The amount of overlap is determined by the explicit window shape. The raised-cosine window given by eq. (4.3) is employed. The window reduces both spectral spill-over and the discontinuities at block boundaries. An overlap of M=32 samples was found to be a good value.

$$w_M(i) = \begin{cases} \frac{1}{2}(1 - \cos[\pi(i+1)/(M+1)]) & i = 0,1,\ldots,M-1 \\ 1 & i = M,M+1,\ldots,N-M-1 \\ \frac{1}{2}(1 - \cos[\pi(N-i)/(M+1)]) & i = N-M,N-M+1,\ldots,N-1 \end{cases} \tag{4.3}$$

The window and its DFT are shown in Figures 4.14 (a) and 4.14 (b).

The above refinements operate on the time waveform. We now describe a modification to the basis spectrum estimate. We term the scheme resulting from this change the modified all-pole model technique.

In [8] Trilolet and Crochiere describe a way of adding pitch information to the basis spectrum estimate. The autocorrelation function is searched for the pitch period P (in samples) by looking for a peak away from the origin. In addition, a gain value G is determined as the ratio of the autocorrelation function at P to its value at the origin (zero sample lag). Note that G must lie in the range (0,1].

The new all-pole estimate $(\hat{\sigma}_i')^2$ is obtained from the original estimate $(\hat{\sigma}_i)^2$ by multiplying the estimate by a pitch factor $\sigma^2(P,G)$

$$\hat{\sigma}_i'^2 = \hat{\sigma}_i^2 \cdot \sigma_i^2(P,G)$$

where

$$\sigma^2(P,G) = \left| DFT\left[\{\sum_{m=0}^{\infty} G^m \delta(n-mP)\} \cdot [U_s(n)-U_s(n-N)] \right] \right|^2$$

and $U_s(n)$ is the unit step function.

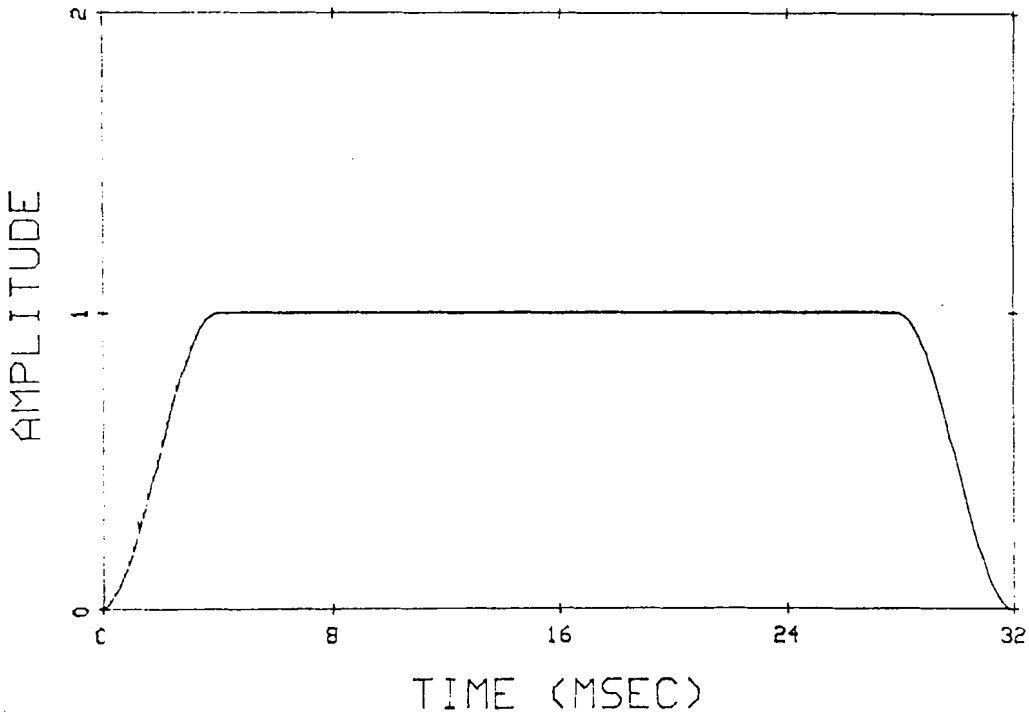


Figure 4.14(a) Time window.

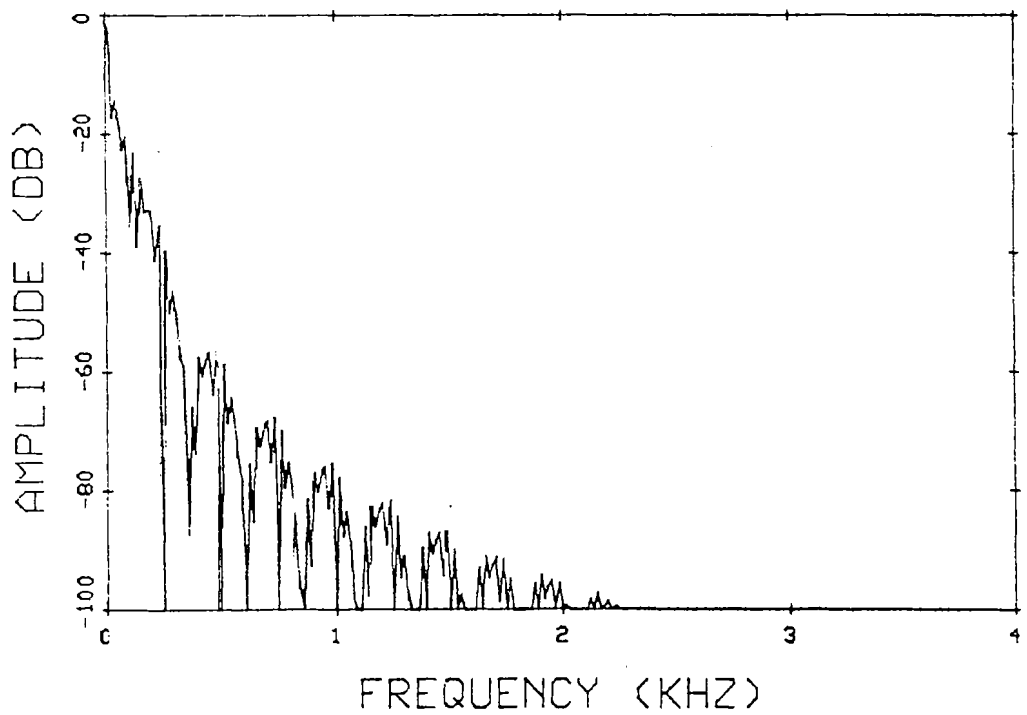


Figure 4.14(b) DFT of time window.

The pitch excitation is modelled as a decaying impulse train with period P. The amplitude of the pulses are geometrically weighted by G. Furthermore, the sequence is windowed by a rectangular window of length N.

Note that the pitch model is described compactly by only two parameters. These parameters must be included in the side information sent to the receiver.

We demonstrate the modified technique by example. Figure 4.15 (a) is a repeat of the spectrum in Figure 4.8 (a). Figure 4.15 (b) shows the basis spectrum estimate using the modified all-pole model. The pitch peaks spaced by the fundamental frequency (125 Hz) are clearly evident. The estimate captures both the gross formant structure as well as the fine pitch striations. Figure 4.16 (a) compares the estimate and spectrum directly. Note that the estimate is still poor at the spectrum extremities.

It is appropriate to mention one further refinement at this point. All the basis spectrum estimates do poorly near 0 Hz. To avoid wasting bits on a region where input signal has no energy, a change in the bit assignment is introduced. Coefficients in the range of 0 Hz to some cutoff f_c are allocated zero bits. For the simulations f_c is chosen as 125 Hz. This value is consistent with the band-pass filter and window characteristics.

Figure 4.17 (a) shows the bit assignment with the number of bits/block set at 212. The bit assignment reflects the pitch periodicity in the

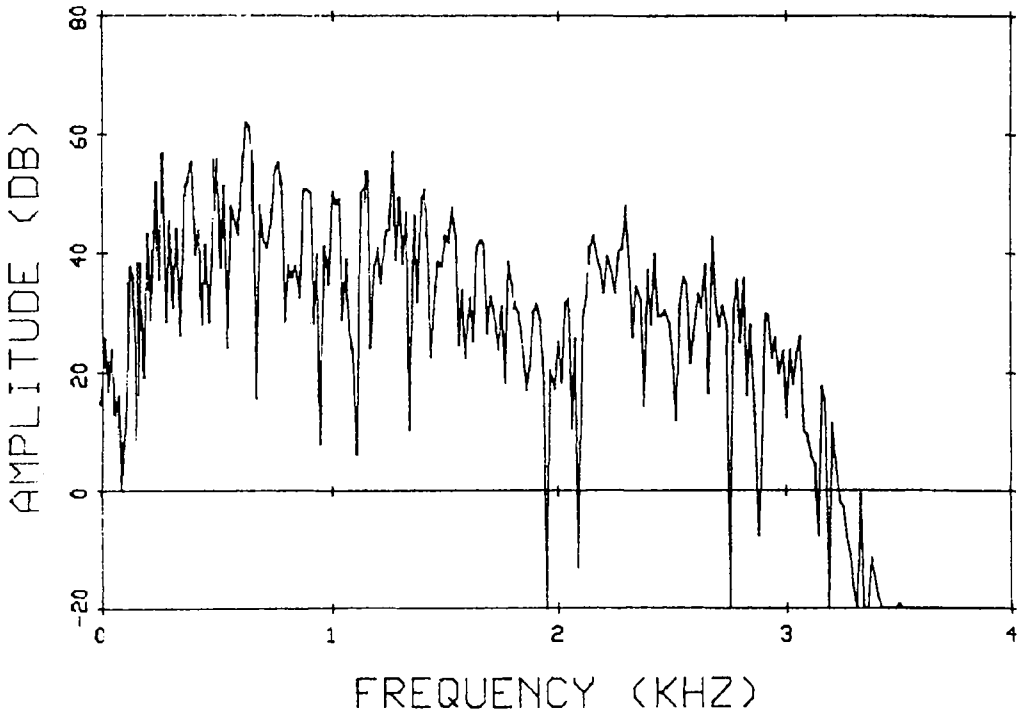


Figure 4.15(a) Input DCT spectrum.

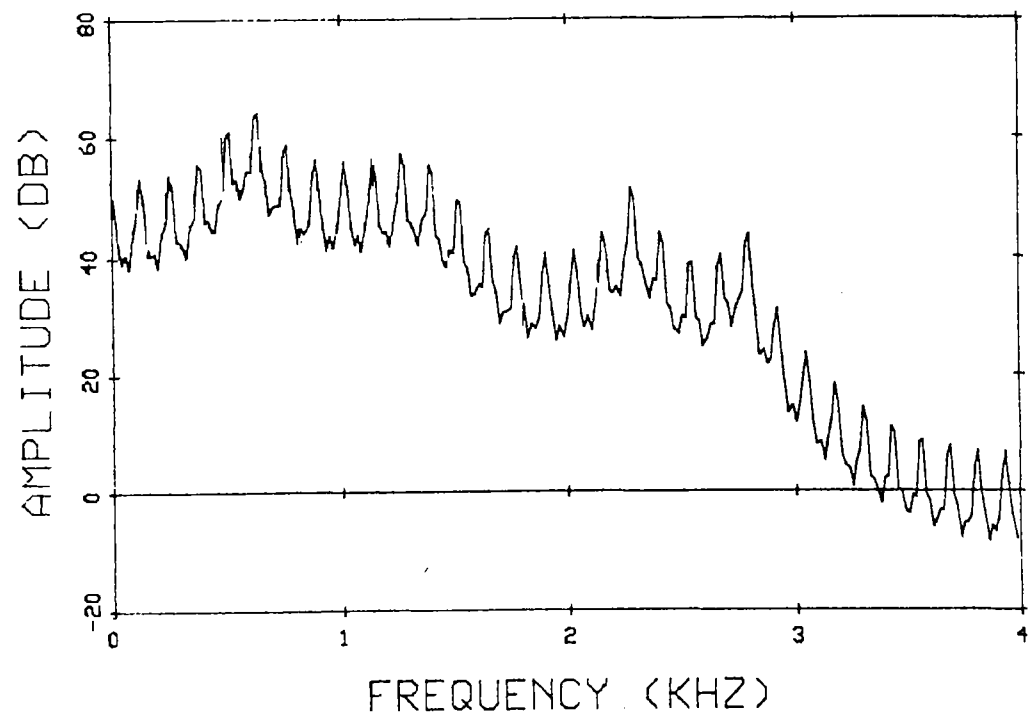


Figure 4.15(b) Basis spectrum estimate using the modified all-pole model technique.

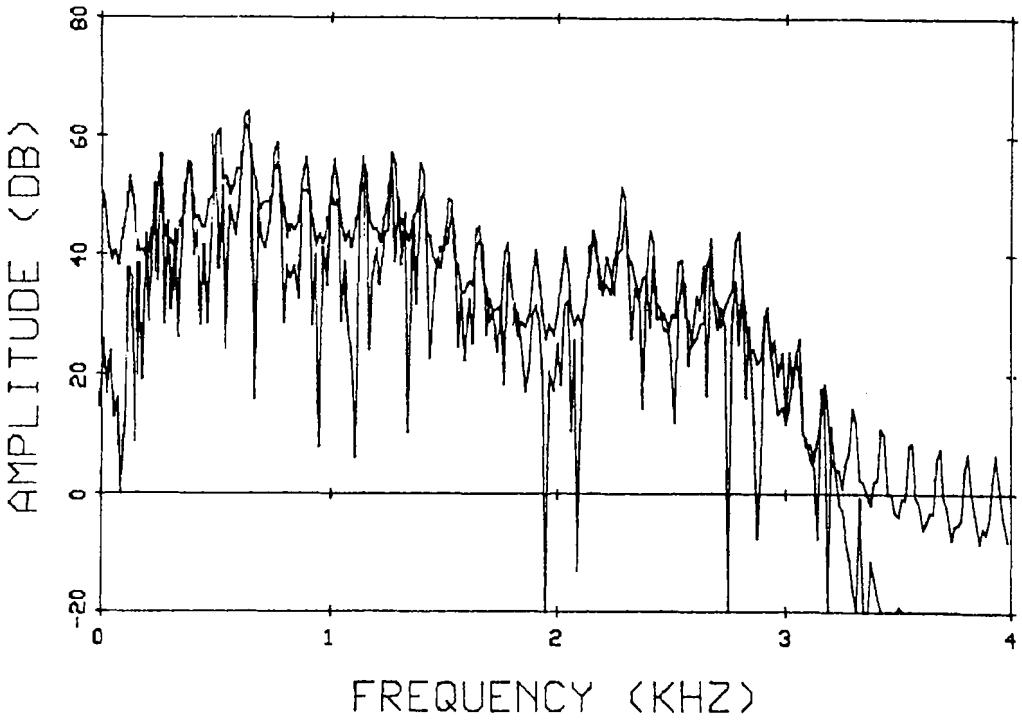


Figure 4.16(a) Superimposed plot of input DCT spectrum and estimate.

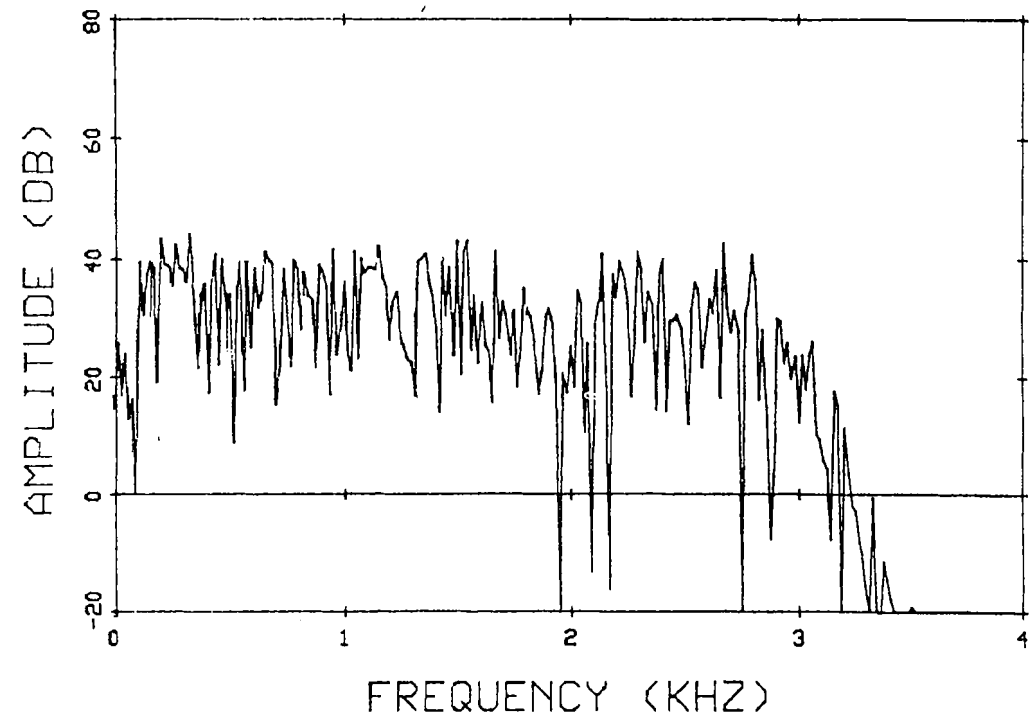


Figure 4.16(b) Quantization noise spectrum.

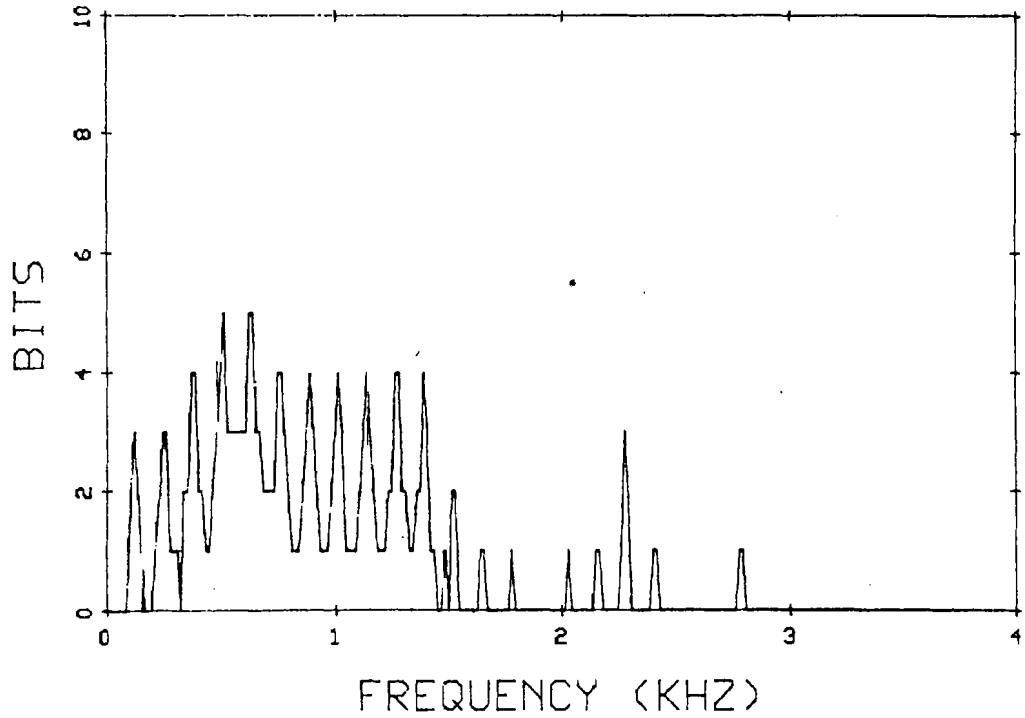


Figure 4.17(a) Bit Assignment curve.

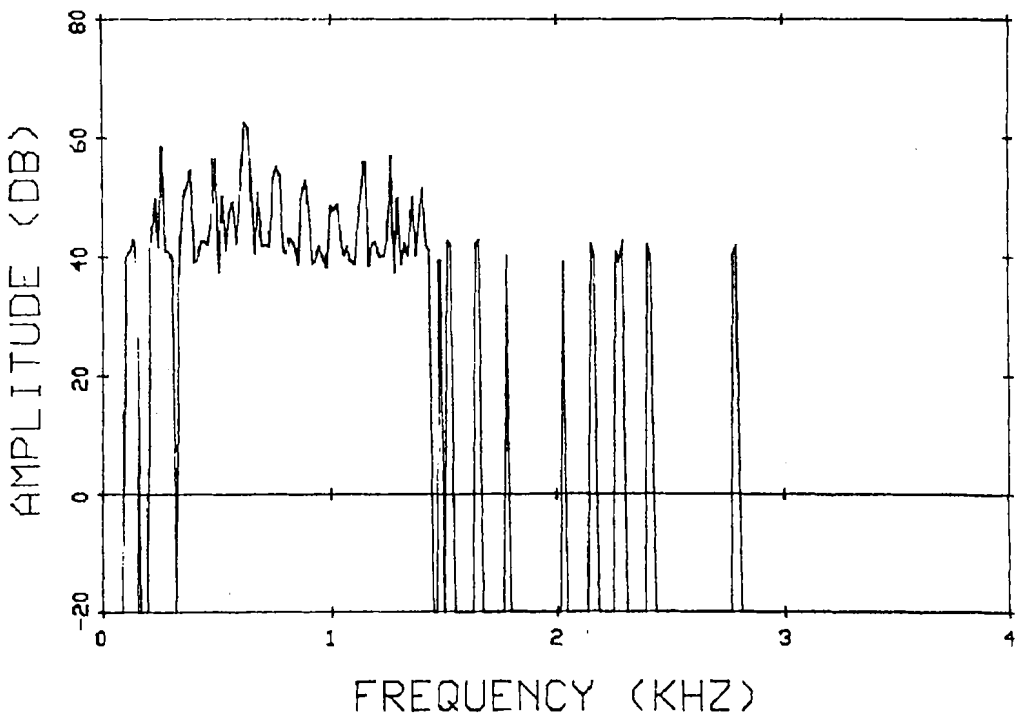


Figure 4.17(b) Receiver DCT spectrum.

basis spectrum estimate. Peaks in the spectrum are allocated bits while few bits get assigned to nulls. This frees additional bits to be assigned to higher frequency peaks.

The receiver spectrum is shown in Figure 4.17 (b). The receiver spectrum displays the fine structure of the input spectrum. Also high frequency energy is extended to the fourth formant region. The quantization noise spectrum is plotted in Figure 4.16 (b).

Time waveforms are illustrated in Figure 4.18 (a) and Figure 4.18 (b). Figure 4.19 (a) compares input and receiver waveforms directly. Figure 4.19 (b) is a plot of the error waveform. Note the amplitude of the error is reduced in comparison with all-pole model error in Figure 4.13 (b).

We next present results to indicate the performance of the overall scheme. Table 4.3 displays the SNR performance for the modified all-pole model scheme. Figure 4.20 (a) shows SEGSNR sensitivity to speaker variation. The introduction of pitch, a strongly speaker dependent phenomenon, into the basis spectrum estimation has increased the performance sensitivity to different speakers. Variations in SEGSNR of up to 2 dB are shown.

Figure 4.20 (b) shows SEGSNR variation to sentence material. SEGSNR can decrease by as much as $2\frac{1}{2}$ dB. Thus the sensitivity remains unchanged from the other schemes.

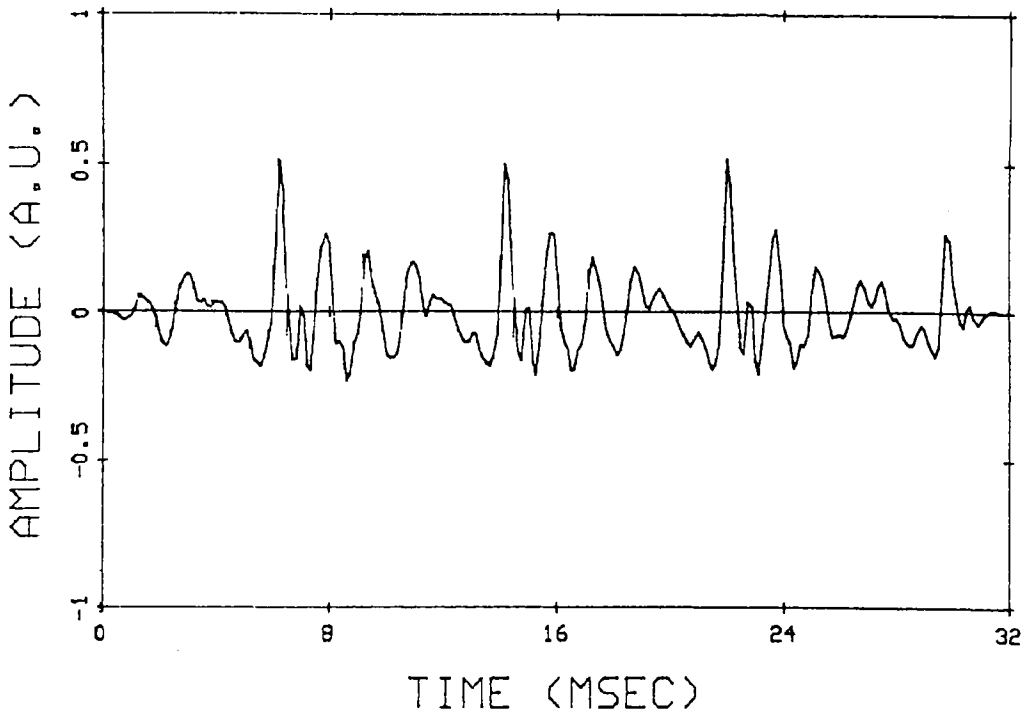


Figure 4.18(a) Input time waveform (same as Figure 4.7(a)).

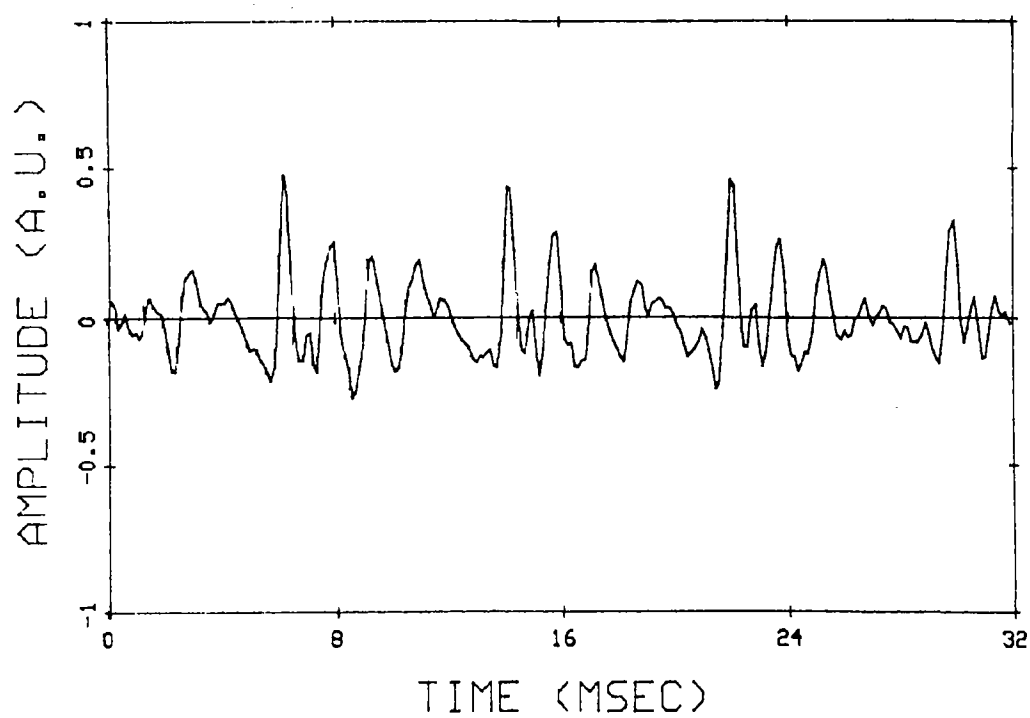


Figure 4.18(b) Receiver time waveform.

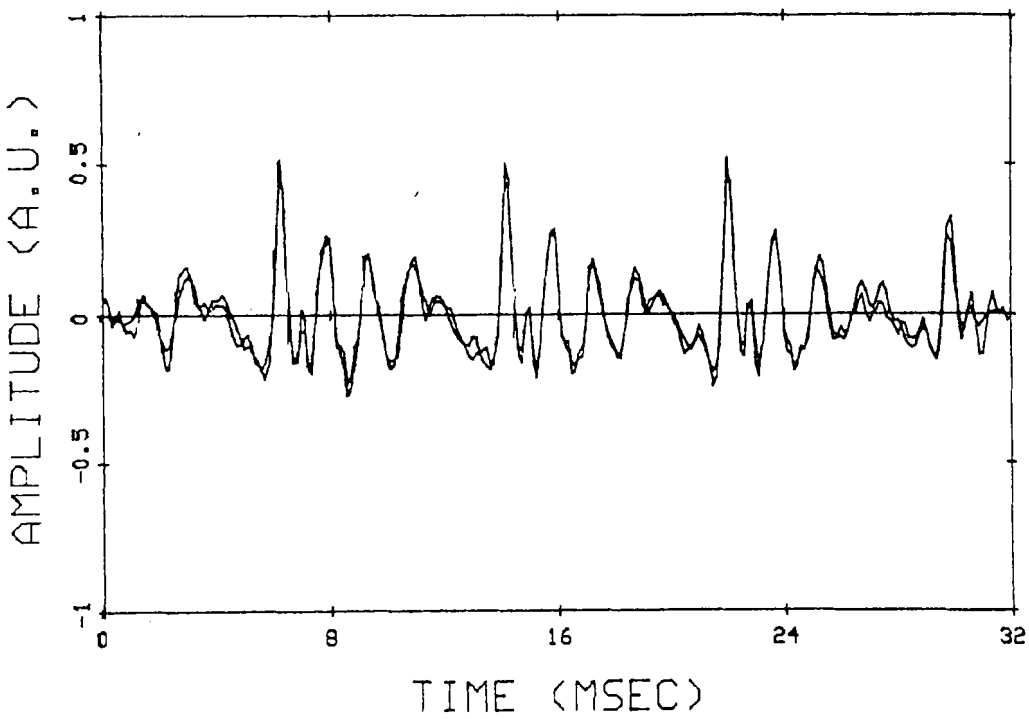


Figure 4.19(a) Superimposed plot of input and receiver time waveforms.

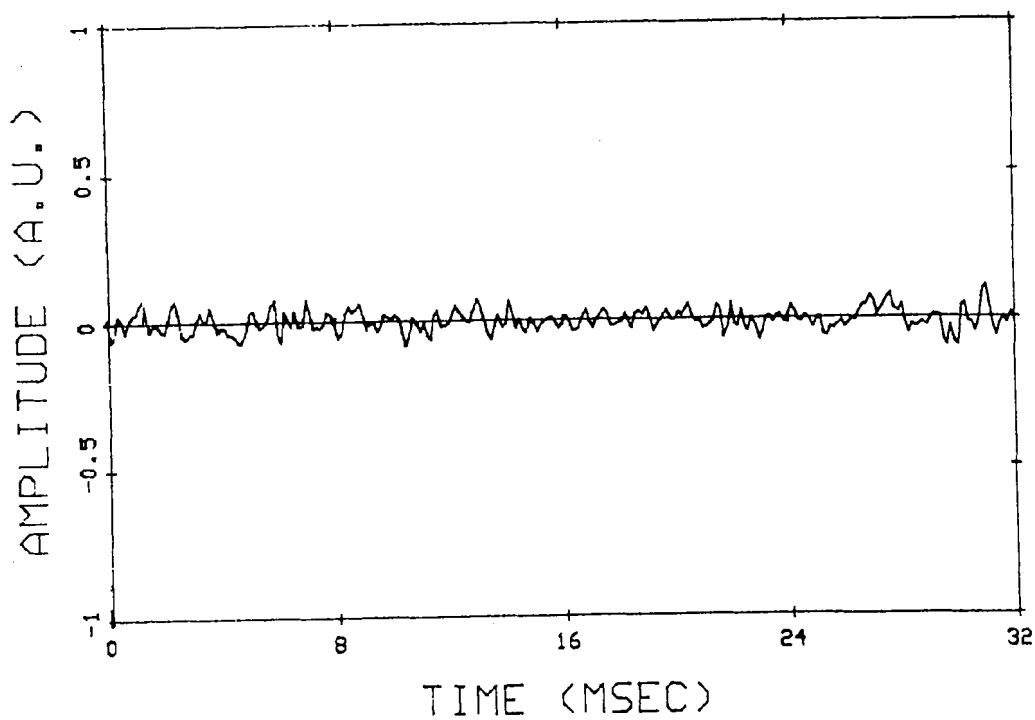


Figure 4.19(b) Quantization error time waveform.

SNR (dB) / SEGMENTAL SNR (dB)

Speaker	Sentence	9.6 kb/sec	12 kb/sec	16 kb/sec
1	A	14.4 / 14.2	16.8 / 16.4	20.1 / 19.3
	B	14.8 / 11.7	17.1 / 13.6	20.5 / 16.5
2	A	13.8 / 14.5	15.8 / 16.5	18.8 / 19.4
	B	15.0 / 12.5	17.2 / 14.3	20.4 / 16.8
3	A	14.0 / 12.8	16.2 / 14.7	19.1 / 17.4
	B	14.9 / 10.7	16.8 / 12.3	19.9 / 15.0
Overall Average:		14.5 / 12.7	16.6 / 14.6	19.8 / 17.4
By Speaker:				
1		14.6 / 12.9	16.9 / 15.0	20.3 / 17.9
2		14.4 / 13.5	16.5 / 15.4	19.6 / 18.1
3		14.4 / 11.7	16.5 / 13.5	19.5 / 16.2
By Sentence:				
	A	14.1 / 13.8	16.2 / 15.9	19.3 / 18.7
	B	14.9 / 11.7	17.0 / 13.4	20.3 / 16.1

Speaker 1: Male
Speaker 2: Female
Speaker 3: Male

Sentence A: It's easy to tell the depth of a well.
Sentence B: The birch canoe slid on the smooth planks.

Table 4.3 SNR Performance of the Modified All-Pole Model Technique

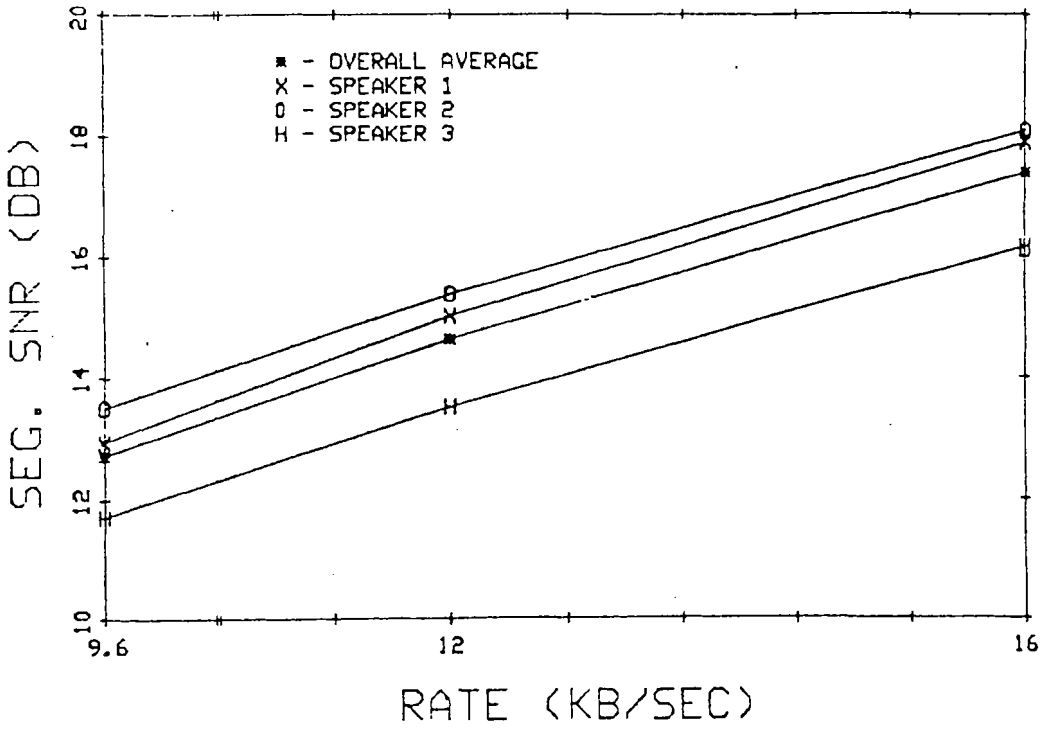


Figure 4.20(a) Sensitivity of the modified all-pole model technique to speaker variations.

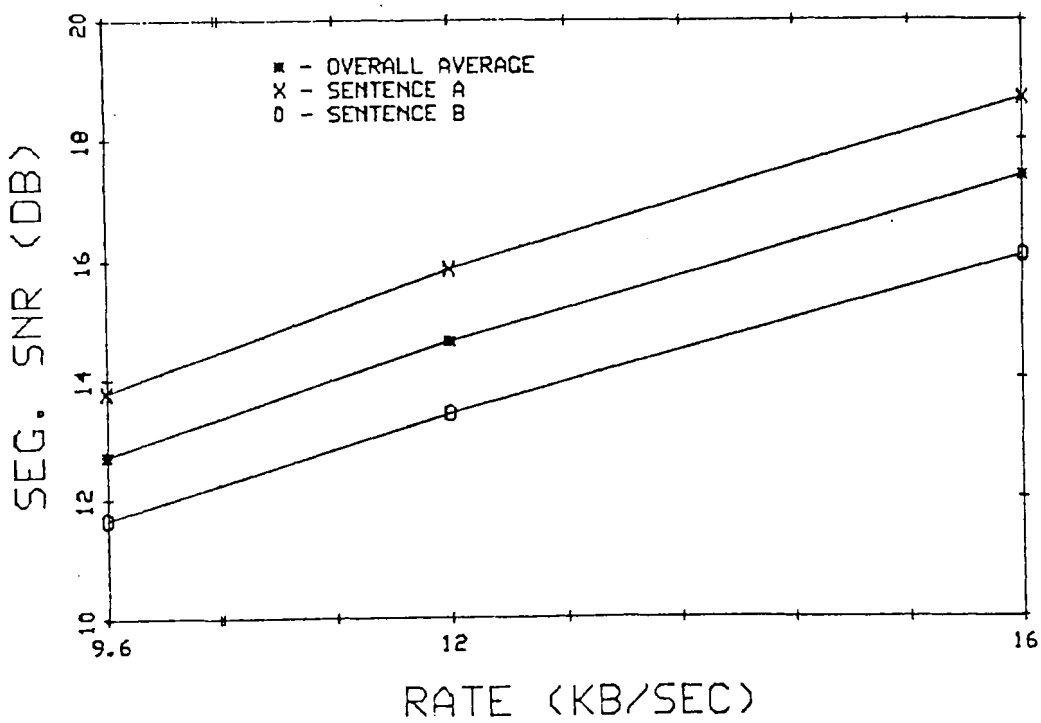


Figure 4.20(b) Sensitivity of the modified all-pole model technique to sentence material.

Fig. 4.21 compares the average performance of the three schemes examined. The modified all-pole model shows the best average performance; 12.7, 14.6 and 17.4 dB for 9.6, 12 and 16 kb/sec. This is almost 1½ dB better than the other schemes.

When speech processed with the modified all-pole model scheme is presented to listeners, they characterize the distortions as being similar in nature to those of the all-pole model scheme. Listeners generally find the distortions less objectionable and they suggest that the modified scheme has the best overall quality of the three schemes investigated. Some listeners, however, notice only a slight improvement with this technique as compared to the all-pole model scheme.

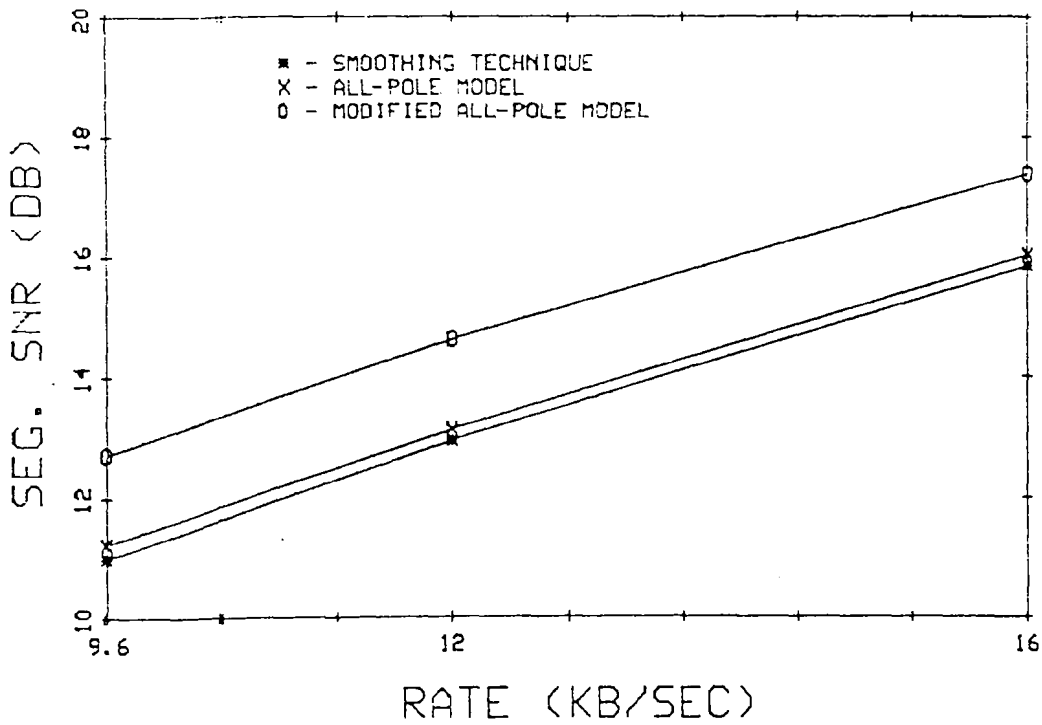


Figure 4.21 Performance comparison of the three basis spectrum estimates.

V Conclusions

In this chapter we summarize the contents of preceeding chapters and state the conclusions of the study. In a final section we outline areas for future investigations.

We have presented a basic transform coding framework. The flexibility in TC lies in the choice of the transformation and in the choice of the quantization strategy. We have selected the DCT as a computationally efficient approximation to the optimal Karhunen-Loève transformation. As well, a general quantization strategy has been formulated, with several specific schemes implemented in a computer simulation. The simulation affirms than ATC is an excellent alternative for speech coding. It offers a range of qualities from very good at 16 kb/sec, to noticeably distorted (but completely intelligible) at 9.6 kb/sec. In fact, at 9.6 kb/sec, ATC (using the modified all-pole scheme) gives better quality than any other scheme presently available.

The performance of ATC depends to a large degree on the accuracy of the basis spectrum estimate. In this regard, the smoothing technique is too crude an estimate to model speech spectra accurately. The estimate yields a step-like bit assignment curve which does not capture the fine structure (pitch striations) in the original spectrum. The smoothing technique is not speech specific. This apparent drawback may be an advantage in the coding of inputs other than speech, or in cases where the coder must handle several types of inputs (as in the telephone network which carries both speech and voice-band data). For example, we

have shown that a 24 kb/sec ATC coder using the smoothing technique, when applied to voice-band data at 4800 b/sec, yields an error rate of 10^{-4}; this compares favourably with other coding schemes, and can certainly be improved.

Basis spectrum estimates using the all-pole model yield better quality coded speech. The all-pole model provides an accurate, concise representation of the overall basis spectrum shape. The addition of pitch information into the modified all-pole model scheme results in the best overall performance of the three schemes investigated. The pitch model provides the necessary fine structure for the basis spectrum estimate. The improvement obtained using the pitch model seems somewhat speaker dependent.

The quality of speech coded by ATC is improved by the use of a non-rectangular window. Windowing reduces clicks by reducing energy near the block boundaries. An additional benefit of windowing is reduced spectral spill-over.

A pre-emphasis filter at the transmitter and a de-emphasis filter at the receiver can further improve ATC performance. The pre-emphasis filter causes better high frequency reproduction by boosting the energy in the higher frequencies before coding. The de-emphasis filter helps to reduce click amplitude since the filter attenuates the high frequency components of the clicks. A disadvantage of too much pre-emphasis is that the processed speech has an audible swishing background noise.

At low rates such as 9.6 kb/sec, speech processed by ATC exhibits noticeable distortion. One type of distortion is a kind of warbling or burbling noise which is perceptually integrated with the speech material and sounds quite unnatural. We speculate that the warbling may be caused by a lack of continuity in the formant trajectories. Since the quantization of the transform coefficients is independent from block to block, it is possible that formant trajectories (especially those of the lower energy second and third formants) can become discontinuous.

ATC appears complex when compared to conventional waveform coders. The main sources of complexity, however, are two types of signal processing. First, the computation of a DFT is an integral part of the DCT calculation and of the all-pole basis spectrum estimate. Second, a recursive matrix inversion (Levinson's algorithm) is required to generate the all-pole filter. In a practical ATC coder these two functions must run in real-time. The general usefulness of the DFT as a signal processing element may well mean that a DFT module (perhaps a self-contained integrated circuit) might be available in the near future. The all-pole filter calculation can presently be done in real-time with fast microprocessors. Hence, a real-time adaptive transform coder is a feasible device.

A situation where ATC may prove useful, independently of whether or not the required circuitry becomes sufficiently inexpensive in the short term to permit its widespread use in private terminal equipment, is in message store-and-forward or message retrieval systems. The point of the coding is to ensure efficient use of digital storage. Expense is limited

because a single coder serves a large group of subscribers. ATC may also be a viable alternative for secure communications. In this case, digital transmission facilitates the encryption of high security messages. A constraint in some secure systems is that existing communication links with limited channel capacity must be utilized. For example, a telephone channel with special compensation (i.e. adaptive equalization) can support transmission rates of up to 9.6 kb/sec, a rate at which ATC yields speech of communication grade quality.

Last, we present several suggestions for future work in ATC. One area which needs further investigation is the determination of those waveform characteristics which correlate with the preceived distortions. Such a study is fundamental to the extension of ATC to even lower bit rates. Below 12 kb/sec the coding scheme has few bits to assign and therefore must allocate bits only to perceptually important spectral components. A second area of interest is the transmission of voice-band data using ATC. The spectrum of voice-band data differs markedly from that of speech. The ability of ATC, a frequency domain technique, to adapt to varying spectral energy distributions makes ATC a logical choice for systems which transmit both speech and data. Other specific issues in ATC that need investigation are the effects of channel errors on coder performance, pole-zero modelling of the basis spectrum, and side information quantization strategies. The solution of these problems will help push ATC towards practicality.

APPENDIX A

Efficient, In-Place Computation of the Discrete Cosine Transform (DCT)

Let $x(n)$, $n = 0,1,\ldots N-1$ be an N point data sequence and $X(k)$ be its corresponding DCT. Then

$$X(k) = \frac{2c(k)}{N} \sum_{n=0}^{N-1} x(n)\cos[\pi(2n+1)k/2N] \qquad k = 0,1,\ldots,N-1 \quad . \qquad (A.1)$$

where $c(k) = 1/\sqrt{2}$ $\quad k = 0$

$= 1$ $\quad k = 1,2,\ldots,N-1$

Reference [20] describes a method for using an N point DFT to calculate an N point DCT. For convenience, we now rederive this result.

Let the summation in eq. (A.1) be denoted by

$$F(k) = \sum_{n=0}^{N-1} x(n)\cos[\pi(2n+1)k/2N] \qquad (A.2)$$

Assuming N is even, define auxiliary sequence $y(n)$ by

$$y(n) = x(2n)$$

$$y(N-1-n) = x(2n+1) \qquad n = 0,1,\ldots,(N/2)-1 \qquad (A.3)$$

Substituting $y(n)$ for $x(n)$ in eq. (A.2) yields

$$F(k) = \sum_{n=0}^{N/2-1} y(n)\cos[(4n+1)k/2N] + \sum_{n=0}^{N/2-1} y(N-1-n)\cos[(4n+3)k/2N]$$

If we let n' = N-1-n in the second summation, simplify, and combine the sums, we get

$$F(k) = \sum_{n=0}^{N-1} y(n)\cos[\pi(4n+1)k/2N]$$

We may evaluate F(k) as the real part of the sequence

$$H(k) = \sum_{n=0}^{N-1} y(n)\exp[-j\pi(4n+1)k/2N]$$

$$= \exp[-j\pi k/2N] \sum_{n=0}^{N-1} y(n)\exp[-j2\pi nk/N]$$

Identifying the summation as the DFT of y(n), namely Y(k),

$$F(k) = \mathrm{Re}[H(k)] = \mathrm{Re}\{\exp[-j\pi k/2N]Y(k)\} \qquad (A.4)$$

Furthermore, it can easily be shown

$$H(N-k) = -j\ H(k)^*$$

(* denotes complex conjugation). Thus

$$F(k) = \mathrm{Re}\ [H(k)] \qquad k = 0,1,2,\ldots,(N/2)-1$$

$$F(N-k) = -\mathrm{Im}\ [H(k)] \qquad k = 1,2,\ldots,(N/2) \qquad (A.5)$$

and X(k) can be evaluated from F(k).

The above derivation shows that an N point DCT may be calculated using an N point DFT. We next consider the problem of in-place calculation.

The rearrangement indicated in eq. (A.3) is simplified when one considers x(n) as a 2 by (N/2) matrix stored in a linear array (as in the FORTRAN convention).

For example, if N = 8

$$\begin{bmatrix} x(0) & x(2) & x(4) & x(6) \\ x(1) & x(3) & x(5) & x(7) \end{bmatrix}$$

Consider the transpose of this matrix,

$$\begin{bmatrix} x(0) & x(1) \\ x(2) & x(3) \\ x(4) & x(5) \\ x(6) & x(7) \end{bmatrix}$$

Comparing storage order,

Original	Transpose	y(n)
x(0)	x(0)	x(0)
x(1)	x(2)	x(2)
x(2)	x(4)	x(4)
x(3)	x(6)	x(6)
x(4)	x(1)	x(7)
x(5)	x(3)	x(5)
x(6)	x(5)	x(3)
x(7)	x(7)	x(1)

Thus rearranging the transpose order to get y(n) is a simple matter of reversing the order of the elements in the second half of the array. Algorithms for the in-place transposition of rectangular matrices may be found in [21], [22].

To summarize, the steps in computing the DCT are as follows:

1) following eq. (A.3) rearrange the data in-place (using matrix transposition)

2) use an in-place FFT algorithm (for real data) to calculate Y(k) (note that an FFT of an N point real sequence needs only N storage locations)

3) perform the complex multiplication in eq. (A.4)

4) calculate F(k) from eq. (A.5)

5) multiply F(k) by 2 c(k)/N

The inverse DCT is computed in a similar manner. The IDCT is defined as:

$$x(n) = \sum_{k=0}^{N-1} c(k)X(k)\cos[\pi(2n+1)k/2N]$$

$$n = 0,1,\ldots, N-1$$

The IDCT may be computed

as

$$w(n) = \mathrm{Re} \sum_{k=0}^{N-1} \{c(k)\ X(k)\exp[j\pi k/2N]\}\exp[j2\pi nk/N]$$

and

$$x(2n) = w(n)$$

$$n = 0,1,\ldots, (N/2)-1$$

$$x(2n+1) = w(N-1-n)$$

Note that the real part operation may be performed in the frequency domain. If

$$R(k) = c(k)X(k)\exp[j\pi k/2N]$$

we can take the IDFT of $\frac{1}{2}[R(k) + R^*(N-k)]$ and avoid taking real parts in the sample domain.

The IDCT can be calculated in-place using the following steps,

1) replace the X(k) sequence by $\frac{1}{2}[R(k) + R^*(N - k)]$

2) use an in-place IFFT algorithm (for transforms of real data) to calculate a permuted version of x(n)

3) use matrix transposition to unscramble x(n)

Two subroutines (listed in Appendix B), DCT and IDCT, implement these procedures for N a power of 2 ranging from 4 to 2048. RFFT performs the real forward and inverse FFT operations. MTIP2 is used for in-place matrix transposition. Note that the complex exponentials are calculated recursively for increased efficiency.

APPENDIX B

This appendix lists the FORTRAN modules used in the computer simulations. Only routines which are directly relevant to the ATC simulation are presented. Auxiliary programs used for filtering and windowing, as well as all system dependent routines, are excluded.

A sequence of operations is required to simulate the complete coder action. A file containing the speech samples is first passed through a pre-emphasis filter. The filtered speech is partitioned into coding blocks by windowing the speech using overlapping windows. Thus adjacent blocks have common speech samples. These blocks are placed in a temporary file. This file is coded via the ATC program. After coding, the overlapping portions of adjacent blocks are added together to undo the effects of windowing. A last step de-emphasizes the coded speech to produce the desired output file.

Functional Description of Program Modules

ATC - Adaptive transform coding main program

ASSIGN - Bit assignment for ATC

AUTO - Autocorrelation method of determining best all-pole model filter

DCT - Forward (in-place) discrete cosine transform

DECMCR - Gets input and output file names

DPCLOS - Closes a file opened by DPOPEN

DPOPEN - Opens a file for reads (READW) or writes (WRITW)

EXIT - Stops program execution

FLTYPE - Adds a default file extension to a file name

HDRIN - Inputs an audio file header (header contains sampling frequency, creation date, and file length)

HDROUT - Outputs an audio file header

IDCT - Inverse (in-place) discrete cosine transform

MESSAG - Interface to system message utility (used to print I/O errors)

MT21 - Nucleus called by MTIP2

MTIP2 - Transposes a 2 by N/2 or a N/2 by 2 matrix in-place

QUANTIZE - Quantizes a variable given a quantizer characteristic

RAD50 - ASCII to an internal character format

READW - Reads from a file

RENEW - Replenishes a buffer with a new set of samples

RENINT - Initialization for RENEW

REQUES - Requests execution of a specified task (used to invoke Master Console Routine MCR)

RFFT - In-place FFT (and inverse FFT) for real data

RLTRFM - Nucleus called by RFFT

SCHED - Schedules task for execution at a given time

SINFO - Side information using the smoothing technique

TCSINFO - Side information using all-pole model or modified all-pole model

TIME - System time routine

VFFT - Nucleus called by RFFT

VSORTP - Sorts an array (in increasing order / keeping track of sort permutation).

WRITW - Writes to a file

```
C                                                       10/04/79 D. SLOAN
C TRANSFORM CODING PROGRAM
C
      PROGRAM ATC
C
C DATA DEFINITIONS
C
      PARAMETER BUFSIZE=512                    ! 2 BLOCK SPEECH BUFFER
      PARAMETER MAXPARA=256                    ! MAX TRANSFORM SIZE
      PARAMETER MAXBITS=8                      ! MAX QUANTIZER BITS
      PARAMETER QSIZE=510                      ! 2**(MAXBITS+1)-2
      PARAMETER UPBND=16383.,LOWBND=-16384.    ! D/A RANGE
      PARAMETER YES='Y', NO='N', SLASH='/'
C LUNS
      PARAMETER FILEIN=1,FILEOUT=2,FILEQNT=3,INDIR=4,KBDI=5,KBDO=5,
     1          LPR=6,FILEPLT=7
C
      BYTE INFILE(38),OUTFILE(38),QNTNAMES(20,6),TIM(8),
     1     IOSTB,GMCR,BDUMMY,LAPLC,UNIF,MULAW,QTYPE,
     2     GAUSS,GAMM,ALAW,DTYPE
C
      LOGICAL PLOT,ZEROFRM,PITCH
C
      INTEGER*2 IBUFFER(BUFSIZE),BITS(MAXPARA),IOST(2),SAMPLES,
     1          RATE,R,BUFCNT,FRMCNT,OFFSET,HRS,MIN,SEC
C
      INTEGER*4 NFRAME,I4DUMMY
C
      REAL BUFFER(BUFSIZE),Q(QSIZE),SIGMA(MAXPARA),FGAIN(MAXPARA),
     1     SIGMAOLD(MAXPARA),QSCALE(MAXBITS)
C
      EQUIVALENCE (IOST,IOSTB), (BUFFER,IBUFFER)
C
      DATA LAPLC,UNIF,MULAW,GAUSS,GAMM,ALAW/'L','U','M','N','G','A'/,
     1     QNTNAMES/'D','P','1',':','L','A','P','L','A','C','E','.',
     2     'Q','N','I',5*0, 'D','P','1',':','U','N','I','F','O','R',
     3     'M','.','Q','N','T',5*0, 'D','P','1',':','M','U','L','A',
     4     'W','.','Q','N','T',7*0, 'D','P','1',':','G','A','U','S',
     5     'S','.','Q','N','T',7*0, 'D','P','1',':','G','A','M','M',
     6     'A','.','Q','N','T',7*0, 'D','P','1',':','A','L','A','W',
     7         '.','Q','N','T',8*0/
C
      CLIP(X)=AMAX1( AMIN1(X,UPBND), LOWBND)
C
      LUNIN=KBDI
      LUNOUT=KBDO
C
C GET THE INPUT AND OUTPUT FILE NAMES
C
10    CALL DECMCR('$AUDIO FILES: ',OUTFILE,'AUD',INFILE,'AUD',
     1            LUNIN,LUNOUT,GMCR)
      IF(OUTFILE(1).EQ.0 .OR. INFILE(1).EQ.0)GO TO 10
C
C OPEN THE INPUT AUDIO FILE
C
      CALL DPOPEN(FILEIN,IOST,INFILE, ,'READ',NBLK)
      CALL MESSAG(,IOST)
      IF(IOSTB.LE.0)GO TO 10
      NBLK=IABS(IOST(2))-1
      IF(NBLK.LE.1)GO TO 850
C
C READ THE INPUT FILE HEADER (ALSO PRINTED ON LPR)
C
      CALL HDRIN(FILEIN,IOST,SFREQ,NDBLK,LPR)
      IF(IOSTB.LE.0)GO TO 850
C
C OPEN THE OUTPUT AUDIO FILE
C
      NBLK=-NBLK                          ! ALLOCATE A NON-CONTIGUOUS FILE
      CALL DPOPEN(FILEOUT,IOST,OUTFILE, ,'CREATE',NBLK)
      CALL MESSAG(,IOST)
      IF(IOSTB.GT.0)GO TO 40
C ERROR IN OPENING OUTPUT FILE
      CALL DPCLOS(FILEIN,IOST,'SAVE')
      CALL MESSAG(,IOST)
      GO TO 10
C
C WRITE THE OUTPUT AUDIO FILE HEADER (ALSO PRINTED ON LPR)
C
40    CALL HDROUT(FILEOUT,IOST,SFREQ,NBLK-1,LPR)
      IF(IOSTB.LE.0)GO TO 750
C
C FIND OUT THE FRAME LENGTH IN SAMPLES
C
50    IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,1000)
      READ(LUNIN,1100,ERR=50,END=750)SAMPLES
      NPARA=SAMPLES
      IF(NPARA.LT.4 .OR. NPARA.GT.MAXPARA)GO TO 50
      I4DUMMY=NBLK-1
      NFRAME=I4DUMMY*256/SAMPLES
```

```
      I4DUMMY=0
C
C FIND OUT THE QUANTIZER TYPE
C
60    IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,1200)
      READ(LUNIN,1300,END=750)NCHR,QTYPE
      IF(NCHR.NE.1)GO TO 60
      IF(QTYPE.EQ.LAPLC .OR. QTYPE.EQ.GAUSS .OR.
     1  QTYPE.EQ.GAMM)GO TO 70
      IF(QTYPE.NE.UNIF .AND. QTYPE.NE.MULAW .AND.
     1  QTYPE.NE.ALAW)GO TO 60
C
C OPEN QUANTIZER FILE
C
70    IF(QTYPE.EQ.LAPLC)J=1
      IF(QTYPE.EQ.UNIF )J=2
      IF(QTYPE.EQ.MULAW)J=3
      IF(QTYPE.EQ.GAUSS)J=4
      IF(QTYPE.EQ.GAMM)J=5
      IF(QTYPE.EQ.ALAW )J=6
C
      OPEN(UNIT=FILEQNT,NAME=QNTNAMES(1,J),TYPE='OLD',READONLY,
     1    ERR=750,FORM='UNFORMATTED',BUFFERCOUNT=2)
C
C FILL IN QUANTIZER ARRAY "Q"
C
      NLEV=2
      INDX=1
C
      DO 90 I=1,MAXBITS
      READ(FILEQNT,ERR=750)(Q(J),J=INDX,INDX+NLEV-1)
      INDX=INDX+NLEV
90    NLEV=NLEV*2
C
C CLOSE QUANTIZER FILE
C
      CLOSE(UNIT=FILEQNT,DISPOSE='SAVE')
C
C READ MAXIMUM NUMBER OF BITS TO BE ALLOCATED TO AN INDIVIDUAL COEFFICIENT
C
100   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,1700)
      READ(LUNIN,1100,ERR=100,END=750)MAXASN
      IF(MAXASN.LT.0 .OR. MAXASN.GT.MAXBITS)GO TO 100
      IF(MAXASN.EQ.0)MAXASN=MAXBITS
C
C READ QUANTIZER LOADING FACTORS
C ONE FOR EACH NUMBER OF BITS THAT COULD BE ASSIGNED
C
110   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,1400)MAXASN
      READ(LUNIN,1500,ERR=110,END=750)(QSCALE(I),I=1,MAXASN)
C
C DEFAULT VALUE FOR QSCALE IS 1.0
C
      DO 111 I=1,MAXASN
      IF(QSCALE(I).LE.0.0)QSCALE(I)=1.0
111   CONTINUE
C
C GET NOISE SHAPING PARAMETER
C
115   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,3800)
      READ(LUNIN,1500,ERR=115,END=750)GAMMA
      IF(GAMMA.LT.-1.0 .OR. GAMMA.GT.0.0)GO TO 115
C
C GET DISTORTION FUNCTION TYPE
C
117   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,4000)
      READ(LUNIN,1300,END=750)NCHR,DTYPE
      IF(NCHR.EQ.0)DTYPE=QTYPE          ! DEFAULT IS QTYPE
      IF(DTYPE.NE.LAPLC .AND. DTYPE.NE.GAUSS .AND.
     1  DTYPE.NE.UNIF)GO TO 117
C
C GET BIT RATE
C
120   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,1800)
      READ(LUNIN,1500,ERR=120,END=750)BITRATE
      IF(BITRATE.LE.0.0)GO TO 120
      RATE=NPARA*BITRATE/SFREQ
      TEMP=RATE*SFREQ/NPARA
      NFRBUF=BUFSIZE/SAMPLES            ! NO. OF FRAMES PER SAMPLE BUFFER
C
C PARAMETER FOR FRAME TO FRAME MEMORY IN VARIANCE ESTIMATE
C
125   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,3400)
      READ(LUNIN,1500,ERR=125,END=750)ALPHA
      IF(ALPHA.LT.0.0 .OR. ALPHA.GE.1.0)GO TO 125
      BETA=1.0-ALPHA
C
C GET PREDICTOR ORDER
C
130   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,3600)
```

```
      READ(LUNIN,2800,ERR=130,END=750)NCHR,M
      IF(NCHR.EQ.0 .OR. M.LT.0 .OR. M.GT.24)GO TO 130
      MP=M+1
      PITCH=.FALSE.
      IF(M.EQ.0)GO TO 145
C
C GET PITCH ESTIMATION PARAMETER
C
140   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,3900)
      READ(LUNIN,1300,END=750)NCHR,BDUMMY
      IF(BDUMMY.EQ.NO)GO TO 145
      IF(BDUMMY.NE.YES .AND. NCHR.GT.0)GO TO 140
      PITCH=.TRUE.
C
C OUTPUT CODING SCHEME PARTICULARS TO THE LINE PRINTER
C
145   WRITE(LPR,2300)NPARA,QTYPE,MAXASN
      WRITE(LPR,2350)(QSCALE(I),I=1,MAXASN)
      WRITE(LPR,2400)RATE,TEMP
      WRITE(LPR,2375)GAMMA,DTYPE
      IF(M.GT.0)WRITE(LPR,2500)M,PITCH
      IF(M.EQ.0)WRITE(LPR,2600)
      IF(ALPHA.GT.0.0)WRITE(LPR,3500)ALPHA
C
C PREEMPHASIS - DEEMPHASIS OPTION
C ( DEFAULT IS NO PREEMPHASIS - DEEEMPHASIS )
C
150   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,2900)
      READ(LUNIN,3000,ERR=150,END=750)NCHR,FBRK1,FBRK2
C
      DELTAF=SFREQ/(2.*NPARA)
      F=0.0
C
      DO 160 I=1,NPARA
      FGAIN(I)=1.0
      IF(FBRK2.GT.FBRK1)FGAIN(I)=(1.+(F/FBRK1)**2) / (1.+(F/FBRK2)**2)
C INITIALIZE SIGMAOLD TO ZERO
      SIGMAOLD(I)=0.0
160   F=F+DELTAF
C
      IF(FBRK2.GT.FBRK1)WRITE(LPR,3100)FBRK1,FBRK2
C
C PLOT BASIS SPECTRUM OPTION
C
      PLOT=.FALSE.
180   IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,3200)
      READ(LUNIN,1300,END=750)NCHR,BDUMMY
      IF(BDUMMY.EQ.NO .OR. NCHR.EQ.0)GO TO 185
      IF(BDUMMY.NE.YES)GO TO 180
      PLOT=.TRUE.
C
C OPEN PLOT FILE
C
      NBLKP=-NFRAME*((NPARA*4+511)/512)
      CALL DPOPEN(FILEPLT,IOST,'DP1:PLT.DAT',,'CREATE',NBLKP)
      IF(IOSTB.GT.0)GO TO 185
      CALL MESSAG(,IOST)
      WRITE(LUNOUT,3300)
      PLOT=.FALSE.
C
C ASK FOR STARTING TIME
C
185   CALL TIME(TIM)
      IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,4100)TIM
      IF(LUNOUT.EQ.KBDO)WRITE(LUNOUT,4200)
      READ(LUNIN,4300,ERR=185,END=750)HRS,MIN,SEC
C
C SCHEDULE TASK
C
      CALL SCHED(HRS,MIN,SEC)
C
C INITIALIZATION
C
187   NBSTP=1
      NBLOCKS=(NPARA+127)/128
      NBYTEP=NPARA*4
      NMEM=0
      NBST=2                              ! START AT BLOCK 2 (SKIP HEADER)
      NBB=BUFSIZE/256                     ! NO. OF BLOCKS PER BUFFER
      NRENEWS=(NFRAME+NFRBUF-1)/NFRBUF
      X2=0.0
      E2=0.0
C
C INITIATE FIRST TRANSFER
C
      CALL RENINT(FILEIN,BUFSIZE,NMEM,NBLK,NBST)
C
      DO 300 BUFCNT=1,NRENEWS
C
C PROCESS A NEW BUFFER
```

```
C
      CALL RENEW(BUFFER,N,IOST)
      IF(IOSTB.LE.0)GO TO 750
      LAST=N/SAMPLES
C
C TRANSFORM THE BUFFER
C
      DO 190 FRMCNT=1,LAST
      OFFSET=(FRMCNT-1)*SAMPLES+1
190   CALL DCT(BUFFER(OFFSET),SAMPLES)
C
C QUANTIZE THE FRAMES
C
      DO 250 FRMCNT=1,LAST
      OFFSET=(FRMCNT-1)*SAMPLES
      I4DUMMY=I4DUMMY+1                ! INCREMENT OVERALL FRAME COUNT
C
      ZEROFRM=.TRUE.
C
      DO 220 I=1,NPARA
      TEMP=(BUFFER(I+OFFSET)**2)*FGAIN(I)
      ZEROFRM=ZEROFRM .AND. TEMP.EQ.0.0
      IF(ALPHA.EQ.0.0)GO TO 220
      TEMP=ALPHA*SIGMAOLD(I)+BETA*TEMP
      SIGMAOLD(I)=TEMP
220   SIGMA(I)=TEMP
C
C IF AN ENTIRE FRAME IS ZERO SKIP QUANTIZATION
C
      IF(ZEROFRM)GO TO 250
C
C PREPARE SIDE INFORMATION
C
      IF(M.GT.0)CALL TCSINFO(SIGMA,NPARA,MP,SFREQ,PITCH)
C
      IF(.NOT. PLOT)GO TO 232
C
C WRITE BASIS SPECTRUM TO PLOT FILE
C
      CALL WRITW(FILEPLT,IOST,NBSTP,NBYTEP,SIGMA)
      NBSTP=NBSTP+NBLOCKS
      IF(IOSTB.GT.0)GO TO 232
      CALL MESSAG(,IOST)
      CALL DPCLOS(FILEPLT,IOST,'DELETE')
      CALL MESSAG(,IOST)
      WRITE(LUNOUT,3300)
      PLOT=.FALSE.
C
C BIT ASSIGNMENT
C
232   R=RATE
      CALL ASSIGN(SIGMA,BITS,R,NPARA,MAXASN,GAMMA,SFREQ,DTYPE)
C
      DO 250 I=1,NPARA
C
      X1=BUFFER(I+OFFSET)
      X2=X2+X1*X1
      Y1=0.0
C
      IF(SIGMA(I).GT.0.0)GO TO 240
      WRITE(LPR,3700)I4DUMMY,I,SIGMA(I)
      IF(SIGMA(I).LT.0.0)GO TO 750
      GO TO 248
C
240   IF(BITS(I).EQ.0)GO TO 248
      S=QSCALE(BITS(I))
      NLEV=2**BITS(I)
      LEVEL1=NLEV-1
      FS=S*SQRT(SIGMA(I)/FGAIN(I))
      Y1=X1/FS
      CALL QUANTIZE(Y1,Q(LEVEL1),NLEV)
      Y1=Y1*FS
248   E2=E2+(X1-Y1)**2
      BUFFER(I+OFFSET)=Y1
250   CONTINUE
C
C INVERSE TRANSFORM THE BUFFER
C
      DO 260 FRMCNT=1,LAST
      OFFSET=(FRMCNT-1)*SAMPLES+1
260   CALL IDCT(BUFFER(OFFSET),SAMPLES)
C
C PLACE SAMPLES IN INTEGER OUTPUT BUFFER
C
      DO 270 I=1,N
270   IBUFFER(I)=CLIP(BUFFER(I))
C
C OUTPUT SAMPLES TO DISK FILE
C
      NBYTES=N*2
```

```
      CALL WRITW(FILEOUT,IOST,NBST,NBYTES,IBUFFER)
      IF(IOSTB.LE.0)GO TO 750
300   NBST=NBST+NBB                    ! UPDATE STARTING BLOCK
C SIGNAL TO NOISE CALCULATION
      SNR=10.*ALOG10(X2/E2)
      WRITE(LPR,2700)SNR
      WRITE(KBDO,2700)SNR
C
C EXIT TO CLOSE INPUT, OUTPUT FILE ( AND PLOT FILE IF NECESSARY)
C
      CALL MESSAG(,IOST)
      CALL DPCLOS(FILEOUT,IOST,'SAVE')
      IF(.NOT. PLOT)GO TO 850
      CALL MESSAG(,IOST)
      CALL DPCLOS(FILEPLT,IOST,'SAVE')
      GO TO 850
C
C EXIT TO CLOSE INPUT, DELETE OUTPUT FILE (AND DELETE PLOT FILE
C                                          IF NECESSARY)
C
750   CALL MESSAG(,IOST)
      CALL DPCLOS(FILEOUT,IOST,'DELETE')
      IF(.NOT. PLOT)GO TO 850
      CALL MESSAG(,IOST)
      CALL DPCLOS(FILEPLT,IOST,'DELETE')
C
C CLOSE INPUT FILE
C
850   CALL MESSAG(,IOST)
      CALL DPCLOS(FILEIN,IOST,'SAVE')
      CALL MESSAG(,IOST)
C
C OPTION TO PROCESS ANOTHER SET OF DATA IF AN INDIRECT FILE
C IS SPECIFIED
C
      IF(LUNIN.NE.INDIR)GO TO 900
      READ(LUNIN,1300,END=900)NCHR,BDUMMY
      CLOSE(UNIT=LPR)
      IF(BDUMMY.EQ.SLASH)GO TO 10
900   IF(NCHR.EQ.NO .AND. LUNIN.NE.INDIR)CALL REQUES(RAD50('...MCR'))
      CALL EXIT
C
C FORMATS
C
1000  FORMAT(/'$FRAME SIZE: ')
1100  FORMAT(I10)
1200  FORMAT(/'$QUANTIZER TYPE: ')
1300  FORMAT(Q,80A1)
1400  FORMAT(/' ENTER',I2,' QUANTIZER LOADING PARAMETERS'
     1       /'$(STARTING FROM 1 BIT QUANTIZER): ')
1500  FORMAT(<MAXBITS>E10.0)
1700  FORMAT(/'$ENTER MAXIMUM BIT ALLOCATION: ')
1800  FORMAT(/'$BIT RATE (BITS/SECOND): ')
2300  FORMAT(' NUMBER OF PARAMETERS PER FRAME:',I4,5X,
     1       'QUANTIZER TYPE: ',A1,5X,'MAXIMUM BIT ALLOCATION:',I2)
2350  FORMAT('0QUANTIZER LOADING FACTORS: ',<MAXBITS>F10.3)
2375  FORMAT('0NOISE SHAPING PARAMETER:',F8.3,5X,
     1       'DISTORTION FUNCTION TYPE: ',A1)
2400  FORMAT('0TOTAL NUMBER OF BITS PER FRAME:',I7,10X,
     1        'TOTAL BIT RATE:',F8.0)
2500  FORMAT('0PREDICTOR ORDER:',I3,5X,'PITCH ESTIMATION:',L2)
2600  FORMAT('0NO BASIS SPECTRUM ESTIMATION')
2700  FORMAT(/'0SIGNAL-TO-NOISE RATIO =',G13.6/)
2800  FORMAT(Q,I10)
2900  FORMAT(/'$ENTER FREQUENCY BREAK-POINTS FOR',
     1       ' PREEMPHASIS-DEEMPHASIS: ')
3000  FORMAT(Q,2E40.0)
3100  FORMAT('0BREAK-POINTS FOR PREEMPHASIS-DEEMPHASIS ARE',
     1       F6.0,' AND',F7.0,' HZ')
3200  FORMAT(/'$PLOT FILE? ')
3300  FORMAT(' *** PLOT FILE DELETED ***')
3400  FORMAT(/'$FRAME TO FRAME MEMORY PARAMETER: ')
3500  FORMAT('0FRAME TO FRAME MEMORY PARAMETER:',F7.4)
3600  FORMAT(/'$PREDICTOR ORDER: ')
3700  FORMAT(/'0*** NON-POSITIVE VARIANCE DETECTED ***'/
     1        '  FRAME:',I6,'  PARAMETER:',I6,'  VARIANCE:',G14.4)
3800  FORMAT(/'$NOISE SHAPING PARAMETER: ')
3900  FORMAT(/'$PITCH ESTIMATION? ')
4000  FORMAT(/'$DISTORTION FUNCTION FOR BIT ASSIGNMENT: ')
4100  FORMAT('0PRESENT TIME IS ',8A1)
4200  FORMAT(/'$ENTER STARTING TIME: ')
4300  FORMAT(J(12,1X))
      END
```

```
      SUBROUTINE ASSIGN(SIGMA,BITS,BITSPF,N,MAXBITS,GAMMA,SFREQ,DTYPE)
C
C                                                  12/03/79 D. SLOAN
C
C ROUTINE TO ASSIGN A PORTION OF BITSPF BITS PER FRAME TO EACH
C OF THE N PARAMETERS. THE ASSIGNMENT IS DONE IN AN OPTIMAL
C (IE. MINIMUM MSE) WAY IF THE PARAMETERS ARE INDEPENDENT RANDOM
C VARIABLES.
C
C SIGMA  - VARIANCE VECTOR
C BITS   - NUMBER OF BITS FOR EACH PARAMETER
C          ( CAN BE 0 FOR LOW VARIANCE PARAMETERS )
C BITSPF - TOTAL NUMBER OF BITS PER FRAME
C N      - NUMBER OF PARAMETERS
C MAXBITS- MAXIMUM BIT ALLOCATION TO A PARAMETER
C          ( NOTE: MAXBITS MUST BE LESS THAN OR EQUAL TO 8 )
C GAMMA  - NOISE SHAPING PARAMETER
C SFREQ  - SAMPLING FREQUENCY
C DTYPE  - DENSITY TYPE FOR DISTORTION FUNCTION
C                    DTYPE= 'L' FOR LAPLACIAN DENSITY
C                    DTYPE= 'N' FOR GAUSSIAN DENSITY
C                    DTYPE= 'U' FOR UNIFORM QUANTIZER
C                               DESIGNED FOR GAUSSIAN DENSITY
C
      PARAMETER LOWFREQ=125.0 ! FREQUENCY TO START BIT ASSIGNMENT
      PARAMETER MAXPARA=256
      BYTE DTYPE
      INTEGER*2 BITS(N),INDEX(MAXPARA*8),BITSPF
      REAL SIGMA(N),P(MAXPARA*8),DK(8,3)
C
      COMMON/WKAREA/P,INDEX
      COMMON/CORRECT/GCOR(MAXPARA)    !CORRECTION ARRAY FROM TCSINFO
C
C THIS DATA STATEMENT INITIALIZES THE DISTORTION DIFFERENCE ARRAY
C
      DATA DK/6.366E-1,  2.459E-1,  8.295E-2,  2.505E-2,  6.996E-3,
     1        1.861E-3,  4.806E-4,  1.221E-4,
     2        5.000E-1,  3.238E-1,  1.217E-1,  3.911E-2,  1.127E-3,
     3        3.042E-3,  7.914E-4,  1.943E-4,
     4        6.366E-1,  2.446E-1,  8.136E-2,  2.590E-2,  8.050E-3,
     5        3*0.0/
C
      IF(N.GT.MAXPARA.OR.MAXBITS.GT.8)STOP '  BIT ASSIGNMENT ERROR'
C
C SELECT DISTORTION FUNCTION
C
      IF(DTYPE.EQ.'N')I=1
      IF(DTYPE.EQ.'L')I=2
      IF(DTYPE.EQ.'U')I=3
C
      IF(DTYPE.EQ.'U' .AND. MAXBITS.GT.5)STOP '  MAXBITS>5 FOR UDF'
C
C ZERO BITS ARRAY
C
      DO 25 J=1,N
25    BITS(J)=0
C
C CALCULATE STARTING INDEX FOR BIT ASSIGNMENT
C
      LOWJ=IFIX(2.*LOWFREQ*N/SFREQ)
C
      DO 30 J=1,N
      S=0.0
      IF(J.GE.LOWJ)S=SIGMA(J)
      IF(GAMMA.LT.0.0)S=S*(GCOR(J)**GAMMA)
C
C CALCULATE MARGINAL RETURNS
C
      DO 30 L=1,MAXBITS
      ISUB=J+(L-1)*N
      INDEX(ISUB)=J
30    P(ISUB)= -S*DK(L,I)
C
C SORT LINEAR ARRAY P IN DECREASING ORDER KEEPING
C TRACK OF PERMUTATIONS IN INDEX
C
      CALL VSORTP(P,N*MAXBITS,INDEX)
C
C ASSIGN BITS ONE BY ONE IN ORDER OF SORTED P ARRAY
C
      DO 40 J=1,BITSPF
      ISUB=INDEX(J)
40    BITS(ISUB)=BITS(ISUB)+1
      RETURN
      END
```

```
      SUBROUTINE AUTO(N,R,MP,A,ALPHA,RC)
C                                                      21/09/78 D.SLOAN
C  THIS SUBROUTINE CALCULATES THE PREDICTOR COEFFICIENTS AND THE
C  REFLECTION COEFFICIENTS USING THE AUTOCORRELATION TECHNIQUE.
C
C  N     - DIMENSION OF AUTOCORRELATION VECTOR
C  R     - AUTOCORRELATION VECTOR
C  MP    - NUMBER OF PREDICTOR AND REFLECTION COEFFICIENTS
C  A     - VECTOR OF PREDICTOR COEFFICIENTS
C  ALPHA - NORMALIZATION FACTOR
C  RC    - REFLECTION COEFICIENTS
C
      DIMENSION A(MP),RC(MP)
      DIMENSION R(N)
C
      RC(1)=-R(2)/R(1)
      A(1)=1.0
      A(2)=RC(1)
      ALPHA=R(1)+R(2)*RC(1)
C
      MPM1=MP-1
      DO 50 MINC=2,MPM1
        S=0.0
        MINO=MINC+2
C
        DO 30 IP=1,MINC
          S=S+R(MINO-IP)*A(IP)
30      CONTINUE
C
        RCM=-S/ALPHA
        RC(MINC)=RCM
        MH=(MINC/2)+1
C
        DO 40 IP=2,MH
          IB=MINO-IP
          AIB=A(IB)
          AIP=A(IP)
          AT=AIP+RCM*AIB
          A(IB)=AIB+RCM*AIP
          A(IP)=AT
40      CONTINUE
C
        A(MINC+1)=RCM
        ALPHA=ALPHA+RCM*S
        IF(ALPHA.LE.0.0) GO TO 60
50    CONTINUE
C
C
60    RETURN
      END
```

```
      SUBROUTINE DCT(X,N)
C                                                    14/07/78 D. SLOAN
C ROUTINE TO DO A DISCRETE COSINE TRANSFORM IN PLACE
C USING A COMPLEX N/2 POINT FFT
C REFERENCE: M.J. NARASIMHA AND A.M. PETETSON
C          IEEE TRANS. ON COMM. VOL. 26 NO. 6 JUNE 1978
C  X - IN THE ARRAY OF DATA SAMPLES ON INPUT AND THE
C      ARRAY OF DCT SAMPLES ON OUTPUT
C  N - IS THE SIZE OF THE TRANSFORM
C
C N.B. N MUST BE A POWER OF 2 NOT GREATER THAN 2048
C
      PARAMETER PIBY2=1.570796327,HROOT2=0.7071067810
      REAL X(N)
C
      ANG=PIBY2/N
      CC=COS(ANG)
      SS= -SIN(ANG)
      FACTOR=2.0/N
      NFFT=N/2
      NFFT2=NFFT/2
C
C PERMUTE INPUT ARRAY
C
      CALL MIP2(X,N,1)
      DO 10 I=1,NFFT2
      TMP=X(I+NFFT)
      X(I+NFFT)=X(N-I+1)
10    X(N-I+1)=TMP
C
C TAKE REAL FFT
C
      CALL RFFT(X,N,1)
C
C COMPLEX MULTIPLY AND SCALING
C
      C1=CC
      S1=SS
      X(1)=X(1)*FACTOR*HROOT2
      X(NFFT+1)=X(NFFT+1)*FACTOR*HROOT2
      DO 20 I=2,NFFT
      XI=X(I)
      YI=X(I+NFFT)
      X(I)=(C1*XI-S1*YI)*FACTOR
      X(I+NFFT)=-(C1*YI+S1*XI)*FACTOR
      TMP=C1*CC-S1*SS
      S1=C1*SS+S1*CC
20    C1=TMP
C
C PERMUTE OUTPUT ARRAY
C
      DO 30 I=2,NFFT2
      TMP=X(I+NFFT)
      X(I+NFFT)=X(N-I+2)
30    X(N-I+2)=TMP
      RETURN
      END
```

```
      SUBROUTINE IDCT(X,N)
C
C ROUTINE TO DO A INVERSE DISCRETE COSINE TRANSFORM IN PLACE
C USING A COMPLEX N/2 POINT FFT
C REFERENCE: M.J NARASIMHA AND A.M PETERSON
C            IEEE TRANS. ON COMM. VOL. 26 NO. 6 JUNE 1978
C
C  X - THE ARRAY OF DCT SAMPLES ON INPUT
C      AND THE DATA SAMPLES ON OUTPUT
C  N - THE SIZE OF THE TRANSFORM
C
C N.B. N MUST BE A POWER OF 2 NOT GREATER THAN 2048
C
      PARAMETER PIBY2=1.570796327,HROOT2=0.7071067810
      REAL X(N)
C
      ANG=PIBY2/N
      CC=COS(ANG)
      SS=SIN(ANG)
      NFFT=N/2
      NFFT2=NFFT/2
C
C PERMUTE INPUT ARRAY
C
      DO 10 I=2,NFFT2
      TMP=X(I+NFFT)
      X(I+NFFT)=X(N-I+2)
10    X(N-I+2)=TMP
C
C PREPARE DATA
C
      C1=CC
      S1=SS
      X(1)=X(1)*HROOT2*N
      X(NFFT+1)=X(NFFT+1)*HROOT2*N
      DO 20 I=2,NFFT
      XI=X(I)*N
      YI=X(I+NFFT)*N
      X(I)=(XI*C1+YI*S1)*0.5
      X(I+NFFT)=(XI*S1-YI*C1)*0.5
      TMP=C1*CC-S1*SS
      S1=C1*SS+S1*CC
20    C1=TMP
C
C TAKE REAL INVERSE FFT
C
      CALL RFFT(X,N,-1)
C
C PERMUTE OUTPUT ARRAY
C
      DO 30 I=1,NFFT2
      TMP=X(I+NFFT)
      X(I+NFFT)=X(N-I+1)
30    X(N-I+1)=TMP
      CALL MTIP2(X,N,-1)
      RETURN
      END
```

```
      SUBROUTINE MTIP2(A,N,IFN)
C                                                    14/07/78 J. COSTA
C
C IFN>0 TRANSPOSES THE MATRIX A(2,N/2) IN PLACE
C IFN<0 TRANSPOSES THE MATRIX A(N/2,2) IN PLACE
C
      REAL A(1)
      IF(IFN.EQ.0)RETURN
      N2=1
1     N2=N2+N2
      IF(N2.GE.N)RETURN
      N1=N/N2
      IF(IFN.GT.0)CALL MT21(A,N1,N2)
      IF(IFN.LT.0)CALL MT21(A,N2,N1)
      GO TO 1
      END
C
      SUBROUTINE MT21(A,N1,N2)
C
C TRANSPOSES THE 2*2 SUBMATRICES OF A N1*N2 MATRIX
C
      REAL A(N1,N2)
      DO 1 J=1,N2,2
      DO 1 I=1,N1,2
      T=A(I+1,J)
      A(I+1,J)=A(I,J+1)
1     A(I,J+1)=T
      RETURN
      END
```

```
      SUBROUTINE QUANTIZE(X,Q,LEVELS)
C                                                    10/04/79 D. SLOAN
C
C ROUTINE TO QUANTIZE X USING QUANTIZER Q
C     X - ON INPUT: VARIABLE TO BE QUANTIZED
C       - ON OUTPUT: QUANTIZED VARIABLE
C     Q - QUANTIZER ARRAY OF OUTPUT LEVELS (BREAK POINTS ARE
C         HALF WAY BETWEEN OUTPUT LEVELS)
C LEVELS- NUMBER OF LEVELS IN QUANTIZER
C
C
      REAL Q(LEVELS)
      INTEGER*2 L,U
C
      L=1
      U=LEVELS
20    I=(L+U)/2
      IF(X-Q(I))30,80,40
30    U=I-1
      IF(U.GE.L)GO TO 20
      I=I-1
      GO TO 50
C
40    L=I+1
      IF(U.GE.L)GO TO 20
C
50    IF(I.GT.0)GO TO 60
      X=Q(1)
      RETURN
C
60    IF(I.LT.LEVELS)GO TO 70
      X=Q(LEVELS)
      RETURN
C
70    T=(Q(I)+Q(I+1))*0.5
      IF(X.LE.T)X=Q(I)
      IF(X.GT.T)X=Q(I+1)
80    RETURN
      END
```

```
      SUBROUTINE RFFT(X,N,IFN)
C                                                          13/08/78 D. SLOAN
C
C ROUTINE TO DO A REAL FFT IN PLACE
C SEE RLTRFM DESCRIPTION FOR DATA ORDERING
C
C  X       - DATA ARRAY
C            NOTE: N SHOULD BE A POWER OF 2 RANGING FROM 2 TO 2048
C  N       - DIMENSION OF X
C  IFN     - INPUT PARAMETER TO DETERMINE DIRECTION OF THE TRANSFORM
C            IF IFN > 0, X IS CONSIDERED AS THE TIME SERIES ON INPUT
C               AND THE FREQUENCY POINTS ON OUTPUT
C            IF IFN < 0, X IS CONSIDER AS THE FREQUENCY POINTS ON INPUT
C               AND THE TIME SERIES ON OUTPUT
C
      REAL X(N)
      IF(N.LT.2 .OR. N.GT.2048)RETURN
C
      NBY2=N/2
      IF(IFN.LT.0)GO TO 10
C
C FORWARD TRANSFORM
C
      CALL MIP2(X,N,+1)
      CALL VFFT(X,X(NBY2+1),NBY2,+1)
      CALL RLTRFM(X,X(NBY2+1),NBY2,+1)
C
      RETURN
C
C INVERSE TRANSFORM
C
10    CALL RLTRFM(X,X(NBY2+1),NBY2,-1)
      CALL VFFT(X,X(NBY2+1),NBY2,-1)
      CALL MIP2(X,N,-1)
C
      RETURN
      END
```

```
      SUBROUTINE RLTRFM(X,Y,NEL,IFN)
C                                                      13/10/78 P. KABAL
C  FORWARD DISCRETE FOURIER TRANSFORM -
C
C    IF IFN IS GREATER THAN ZERO, THIS SUBROUTINE COMPLETES THE DISCRETE
C    FOURIER TRANSFORM OF 2*N REAL DATA POINTS. THE FIRST STEP IN THIS
C    PROCESS IS TO REARRANGE THE DATA SO THAT IT IS STORED ALTERNATELY
C    IN THE ARRAYS X AND Y. LET THE INPUT REAL SEQUENCE BE
C    A(1),A(2),...,A(2*N). THESE VALUES ARE STORED IN THE ARRAYS X AND
C    Y IN THE FOLLOWING ORDER.
C       A(1)      - X(1)
C       A(2)      - Y(1)
C       A(3)      - X(INC+1)
C       A(4)      - Y(INC+1)
C       ...         ...
C       A(2*N-1)  - X(INC*(N-1)+1)
C       A(2*N)    - Y(INC*(N-1)+1)
C    THE NEXT STEP IS TO TRANSFORM THE DATA USING A COMPLEX DISCRETE
C    FOURIER TRANSFORM OF LENGTH N. THE FINAL STEP IS TO CALL THIS
C    SUBROUTINE TO COMPLETE THE TRANSFORM. LET THE N+1 COMPLEX
C    OUTPUT VALUES BE (AR(1),AI(1)),...,(AR(N+1),AI(N+1)). THESE
C    OUTPUT VALUES ARE STORED AS FOLLOWS.
C       AR(1)    - X(1)
C       AR(2)    - X(INC+1)
C       ...        ...
C       AR(N)    - X(INC*(N-1)+1)
C       AR(N+1)  - Y(1)
C       AI(2)    - Y(INC+1)
C       AI(3)    - Y(INC*2+1)
C       ...        ...
C       AI(N)    - Y(INC*(N-1)+1)
C    NOTE THAT AR(N+1) IS STORED IN THE PLACE THAT WOULD NORMALLY BE
C    RESERVED FOR AI(1). THIS IS POSSIBLE SINCE AI(1) (AND AI(N+1)) ARE
C    KNOWN A PRIORI TO BE ZERO. A TYPICAL CALLING SEQUENCE IS AS FOLLOWS.
C        NEL=INC*(N-1)+1
C      CALL VFFT(X,Y,NEL,+INC)
C      CALL RLTRFM(X,Y,NEL,+INC)
C
C  INVERSE DISCRETE FOURIER TRANSFORM -
C
C    IF IFN IS LESS THAN ZERO, THIS SUBROUTINE PREPARES THE DATA FOR
C    EVALUATING AN INVERSE DISCRETE FOURIER TRANSFORM THAT RESULTS IN
C    2*N REAL DATA VALUES. LET THE N+1 COMPLEX INPUT VALUES BE
C    (AR(1),AI(1)),...,(AR(N+1),AI(N+1)). IF THESE VALUES ARE TO RESULT
C    IN A REAL SEQUENCE AFTER THE INVERSE DISCRETE FOURIER TRANSFORM,
C    AI(1) AND AI(N+1) MUST BOTH BE ZERO. THE REMAINING 2*N DATA
C    VALUES (N+1 REAL COMPONENTS AND N-1 IMAGINARY COMPONENTS) ARE
C    STORED IN THE ARRAYS X AND Y AS FOLLOWS.
C       AR(1)    - X(1)
C       AR(2)    - X(INC+1)
C       ...        ...
C       AR(N)    - X(INC*(N-1)+1)
C       AR(N+1)  - Y(1)
C       AI(2)    - Y(INC+1)
C       AI(3)    - Y(INC*2+1)
C       ...        ...
C       AI(N)    - Y(INC*(N-1)+1)
C    NOTE THAT AR(N+1) IS STORED IN THE PLACE THAT WOULD NORMALLY BE
C    RESERVED FOR AI(1). THIS SUBROUTINE IS THEN CALLED TO PREPARE
C    THE DATA FOR THE INVERSE DISCRETE FOURIER TRANSFORM. THE OUTPUT
C    OF THE INVERSE TRANSFORM CONTAINS THE 2*N DESIRED REAL DATA POINTS
C    STORED ALTERNATELY IN THE ARRAYS X AND Y. LET THE 2*N REAL DATA
C    POINTS BE A(1),A(2),...,A(2*N). THESE ARE STORED AS FOLLOWS.
C       A(1)      - X(1)
C       A(2)      - Y(1)
C       A(3)      - X(INC+1)
C       A(4)      - Y(INC+1)
C       ...         ...
C       A(2*N-1)  - X(INC*(N-1)+1)
C       A(2*N)    - Y(INC*(N-1)+1)
C    A TYPICAL CALLING SEQUENCE IS AS FOLLOWS.
C        NEL=INC*(N-1)+1
C      CALL RLTRFM(X,Y,NEL,-INC)
C      CALL VFFT(X,Y,NEL,-INC)
C
C  SUBROUTINE PARAMETERS -
C
C    X    - N VALUES, THE ODD NUMBERED ELEMENTS OF AN ARRAY OF REAL DATA
C           OR THE REAL PART OF COMPLEX DATA
C    Y    - N VALUES, THE EVEN NUMBERED ELEMENTS OF AN ARRAY OF REAL DATA
C           OR THE IMAGINARY PART OF COMPLEX DATA
C    NEL  - THE ARRAYS X AND Y ARE EACH DIMENSIONED NEL, WHERE
C           NEL=INC*(N-1)+1, AND N IS THE NUMBER OF POINTS IN THE
C           TRANSFORM. FOR INC=1, NEL=N.
C    IFN  - INPUT PARAMETER, EQUAL TO +INC FOR THE TRANSFORM,
C                            EQUAL TO -INC FOR THE INVERSE TRANSFORM,
C           WHERE INC REPRESENTS THE INCREMENT BETWEEN ELEMENTS OF THE
C           ARRAYS (NORMALLY 1). IF INC IS EQUAL TO 1, ALL ELEMENTS OF THE
C           ARRAYS X AND Y ARE USED IN PROCESSING THE DATA. IF INC IS
C           GREATER THAN 1 EVERY INC-TH ELEMENT OF EACH ARRAY IS USED IN
```

```
C          PROCESSING THE DATA.
C    N.B. THE FAST FOURIER TRANSFORM ALGORITHM REQUIRES THAT THE LENGTH
C         OF THE TRANSFORM, N, BE A POWER OF TWO.
C
      DIMENSION X(NEL),Y(NEL)
      DATA PI/3.141592654/
C
C  INITIALIZATION
      INC=IABS(IFN)
      N=(NEL+(INC-1))/INC
      ARG=PI/FLOAT(N)
      CC=COS(ARG)
      SS=-SIN(ARG)
      C1=CC
      S1=SS
      TM=X(1)
      X(1)=TM+Y(1)
      Y(1)=TM-Y(1)
      IF(IFN.GT.0) GO TO 20
      C1=-C1
      SS=-SS
      X(1)=0.5*X(1)
      Y(1)=0.5*Y(1)
C
C  COMPLETE THE TRANFORM OR PREPARE FOR THE INVERSE TRANSFORM
20    NH=(N/2)*INC+1
      JST=INC+1
      IF(JST.GT.NH) GO TO 50
      K=N*INC-(INC-1)
      DO 40 J=JST,NH,INC
        AR0=X(J)+X(K)
        AR1=Y(J)+Y(K)
        AI0=Y(J)-Y(K)
        AI1=X(J)-X(K)
        XX=C1*AR1+S1*AI1
        YY=S1*AR1-C1*AI1
        Y(K)=0.5*(YY-AI0)
        Y(J)=0.5*(YY+AI0)
        X(K)=0.5*(AR0-XX)
        X(J)=0.5*(AR0+XX)
        TM=C1*CC-S1*SS
        S1=C1*SS+S1*CC
        C1=TM
        K=K-INC
40    CONTINUE
C
50    RETURN
C
      END
```

```
      SUBROUTINE SINFO(SIGMA,NPARA,M)
C                                                        15/09/78 D. SLOAN
C ROUTINE TO AVERAGE, DECIMATE AND LINEARLY INTERPOLATE LOG SIGMA VALUES
C SEE THE ZELINSKI AND NOLL PAPER ON TRANSFORM CODING
C
C SIGMA - VARIANCE VECTOR
C NPARA - NUMBER OF PARAMETERS
C M     - AVERAGE OVER M PARAMETERS (IE. SUBSAMPLE VARIANCE BY M)
C
      REAL SIGMA(NPARA)
C
      IF(M.GT.NPARA)RETURN
      DELTA=FLOAT(M)
C
C AVERAGE OVER M ELEMENTS
C
      DO 200 I=1,NPARA,M
      SUM=0.0
C
      DO 100 J=I,I+M-1
100   SUM=SUM+SIGMA(J)
      TEMP=-80.0
      IF(SUM.GT. 1.0E-35)TEMP=ALOG(SUM/DELTA)
200   SIGMA(I)=TEMP
      IF(M.EQ.NPARA)GO TO 350
C
C LINEAR INTERPOLATION OF LOG SIGMA VALUES
C
      DO 300 I=1,NPARA-M,M
      A=(SIGMA(I+M)-SIGMA(I))/DELTA
      B=((I+M)*SIGMA(I)-I*SIGMA(I+M))/DELTA
C
      DO 300 J=I,I+M-1
300   SIGMA(J)=EXP(A*J+B)
C
350   S=EXP(SIGMA(NPARA-M+1))
C
      DO 400 J=NPARA-M+1,NPARA
400   SIGMA(J)=S
      RETURN
      END
```

```
      SUBROUTINE TCSINFO(SIGMA,NPARA,M,SFREQ,PITCH)
C                                                        28/02/79 D. SLOAN
C SIDE INFORMATION FOR TRANSFORM CODING
C REFERENCE: A VOCODER-DRIVEN ADAPTATION STRATEGY
C            FOR LOW-BIT RATE ADAPTIVE TRANSFORM
C            CODING OF SPEECH
C
C SIGMA - VARIANCE VECTOR
C NPARA - NUMBER OF PARAMETERS
C M     - NUMBER OF PREDICTOR COEFFICIENTS + 1
C SFREQ - SAMPLING FREQUENCY
C PITCH - LOGICAL VARIABLE DETERMINING PITCH ESTIMATION
C
      PARAMETER MAXPARA=256
      PARAMETER MPITCH=60.    ! MINIMUM PITCH FREQUENCY IN HERTZ
C
      LOGICAL PITCH
C
      REAL SIGMA(NPARA)
C
      COMMON /WKAREA/A(25),RC(25),X(1)
      COMMON/CORRECT/GCOR(MAXPARA)
C
      IF(M.GT.25)STOP '  TOO MANY PREDICTOR COEFFICIENTS'
C
      NP1=NPARA+1
      NP2=NPARA*2
      I2=NP1
C
      DO 10 I=1,NPARA
      X(I)=SIGMA(I)
      X(I2)=0.0
10    I2=I2+1
C
C TAKE REAL INVERSE FFT IN PLACE
C
      CALL RFFT(X,NP2,-1)
      IF(.NOT. PITCH)GO TO 16
C
C LOCATE SECOND MAXIMUM OF AUTOCORRELATION FUNCTION
C
      XMAX=-1.E30
      LIMIT=MINO(IFIX(SFREQ/MPITCH),NPARA)
C
      DO 15 I=M+1,LIMIT
      IF(X(I).LE.XMAX)GO TO 15
      XMAX=X(I)
      MAX=I
15    CONTINUE
C
      GAIN=XMAX/X(1)
C
C GENERATE OPTIMAL (MINIMUM SQUARE ERROR) PREDICTOR
C
C NOTE: THE A(I) ARE THE ANALYSIS FILTER COEFFICIENTS
C
16    M1=M
17    CALL AUTO(NPARA,X,M1,A,ALPHA,RC)
C
C CHECK FOR AN UNSTABLE FILTER
C
      IF(ALPHA.GT.0 .OR. M1.LE.3)GO TO 18
C
C FILTER IS UNSTABLE, DECREASE ORDER BY 1 AND TRY AGAIN
C
      M1=M1-1
      GO TO 17
C
18    I2=NP1
      DO 20 I=1,NPARA
      X(I)=0.0
      IF(I.LE.M)X(I)=A(I)
      X(I2)=0.0
20    I2=I2+1
C
C TAKE REAL FFT IN PLACE
C
      CALL RFFT(X,NP2,+1)
C
C STORE THE "LINEAR PREDICTED SPECTRUM" IN SIGMA
C
      I2=NP1
      X(I2)=0.0
C
      DO 30 I=1,NPARA
      SIGMA(I)=ALPHA/(X(I)**2+X(I2)**2)
      GCOR(I)=SIGMA(I)
30    I2=I2+1
      IF(.NOT. PITCH)GO TO 60
C
```

```
C CALCULATE FINE STRUCTURE PART OF SPECTRUM
C
      I2=NP1
      MAXM1=MAX-1
C
      DO 40 I=1,NPARA
      IM1=I-1
      K=IM1/MAXM1
      X(I)=0.0
      IF(K*MAXM1.EQ.IM1)X(I)=GAIN**K
      X(I2)=0.0
40    I2=I2+1
C
C TAKE REAL FFT INPLACE
C
      CALL RFFT(X,NP2,+1)
C
C STORE COMPLETED "BASIS SPECTRUM"
C
      I2=NP1
      X(I2)=0.0
C
      DO 50 I=1,NPARA
      SIGMA(I)=SIGMA(I)*(X(I)**2+X(I2)**2)
50    I2=I2+1
C
60    RETURN
      END
```

```
      SUBROUTINE VFFT(X, Y, NEL, IFN)
C                                                    VERIC 13/10/78 P. KABAL
C   THIS SUBROUTINE CALCULATES THE DISCRETE FOURIER TRANSFORM OR THE
C   INVERSE DISCRETE FOURIER TRANSFORM OF A SET OF COMPLEX NUMBERS USING
C   THE FAST FOURIER TRANSFORM ALGORITHM DEVELOPED BY G. SANDE (MIXED
C   RADIX 4 AND RADIX 2, SEE W.M. GENTLEMAN AND G. SANDE, FAST FOURIER
C   TRANSFORMS - FOR FUN AND PROFIT, 1966 FALL JOINT COMPUTER
C   CONFERENCE). THE CALCULATION IS DONE IN PLACE, THAT IS THE OUTPUT
C   DATA REPLACES THE INPUT DATA. THE M-TH COMPLEX OUTPUT DATA POINT
C   OF THE N POINT DISCRETE FOURIER TRANSFORM IS
C
C                N            -(K-1)
C      CO(M) = SUM  CI(K) * W         ,  WHERE W = EXP(J*2*PI*(M-1)/N),
C               K=1
C
C   J IS THE IMAGINARY OPERATOR, AND CI(K) IS THE K-TH COMPLEX INPUT
C   DATA POINT. SIMILARLY, THE K-TH COMPLEX OUTPUT DATA POINT OF THE N
C   POINT INVERSE DISCRETE FOURIER TRANSFORM IS
C
C                      N            (K-1)
C      CO(K) = (1/N) SUM  CI(M) * W       .
C                     M=1
C
C   THIS IMPLEMENTATION OF THE FAST FOURIER TRANSFORM CALCULATES
C   THE EXPONENTIAL FUNCTION RECURSIVELY IN ORDER TO GET GREATER
C   SPEED AT THE EXPENSE OF SOME LOSS OF PRECISION. THE SUBROUTINE
C   ARGUMENTS ARE AS FOLLOWS.
C
C   X       - ARRAY OF LENGTH N CONTAINING THE REAL PART OF THE DATA.
C   Y       - ARRAY OF LENGTH N CONTAINING THE IMAGINARY PART OF THE DATA.
C   NEL     - NUMBER OF ELEMENTS IN EACH OF THE ARRAYS X AND Y.
C             NEL=INC*(N-1)+1, WHERE N IS THE NUMBER OF POINTS IN THE
C             TRANSFORM. N MUST BE A POWER OF TWO NOT GREATER THAN 1024.
C             FOR INC=1, NEL=N.
C   IFN     - INPUT PARAMETER, EQUAL TO +INC FOR THE TRANSFORM,
C                              EQUAL TO -INC FOR THE INVERSE TRANSFORM.
C             INC REPRESENTS THE INCREMENT BETWEEN DATA ELEMENTS
C             (NORMALLY 1). IF INC IS EQUAL TO 1, EVERY ELEMENT OF THE
C             ARRAYS X AND Y IS USED. WHEN INC IS GREATER THAN 1, ONLY
C             EVERY INC-TH ELEMENT OF EACH ARRAY IS USED.
C
      DIMENSION X(NEL), Y(NEL), KE(10), KI(10), CA(5), SA(5)
      EQUIVALENCE (KE( 1),KE01), (KE( 2),KE02), (KE( 3),KE03),
     *            (KE( 4),KE04), (KE( 5),KE05), (KE( 6),KE06),
     *            (KE( 7),KE07), (KE( 8),KE08), (KE( 9),KE09),
     *            (KE(10),KE10)
      EQUIVALENCE (KI( 1),KI01), (KI( 2),KI02), (KI( 3),KI03),
     *            (KI( 4),KI04), (KI( 5),KI05), (KI( 6),KI06),
     *            (KI( 7),KI07), (KI( 8),KI08), (KI( 9),KI09),
     *            (KI(10),KI10)
      DATA NP/-999/,INCP/-999/,LG2NMX/10/,PI2/6.283185307/
C
C  CHECK IF THE INITIALIZATION HAS ALREADY BEEN DONE
      INC=IABS(IFN)
      N=(NEL+(INC-1))/INC
      IF (N.EQ.NP .AND. INC.EQ.INCP) GO TO 250
C
C  FIND THE LOGARITHM (BASE 2) OF THE NUMBER OF POINTS
      NP=2
      DO 100 LOG2N=1,LG2NMX
        IF (NP.EQ.N) GO TO 120
        NP=NP+NP
  100 CONTINUE
      NP=-999
      GO TO 1000
C
  120 INCP=INC
      INC2=INC+INC
      NI=INC*NP
C
C  SET UP THE UNSCRAMBLING LOOP PARAMETERS (NOTE THAT THE VARIABLES
C  KE=J AND KI=J ARE EQUIVALENT TO KE(J) AND KI(J) RESPECTIVELY)
      KE01=NI
      KI01=NI/2
      DO 200 J=2,LG2NMX
        KE(J)=KI(J-1)
        KI(J)=MAX0(KE(J)/2,INC)
  200 CONTINUE
C
C  SET UP THE INITIAL VALUES FOR THE RECURSIVE CALCULATION OF
C  THE TWIDDLE FACTORS
      FN=N
      IF (LOG2N.LE.1) GO TO 250
      FM4=PI2/FN
      LOG4N=LOG2N/2
      DO 230 K=1,LOG4N
        CA(K)=COS(FM4)
        SA(K)=SIN(FM4)
        FM4=4.0*FM4
  230 CONTINUE
```

```
C
  250 IF (IFN.GT.0) GO TO 320
C
C  FOR THE INVERSE TRANSFORM ONLY,
C  FORM THE COMPLEX CONJUGATE OF THE INPUT DATA
      DO 300 I=1,N1,INC
        Y(I)=-Y(I)
  300 CONTINUE
C
  320 IF (LOG2N.LE.1) GO TO 650
C
C
C  CALCULATE THE RADIX 4 FAST FOURIER TRANSFORM
      M4=N1
      DO 600 K=1,LOG4N
        M=M4/4
        CC=CA(K)
        SS=SA(K)
        C1=1.0
        S1=0.0
C
        DO 500 J=1,M,INC
          IF(J.EQ.1) GO TO 340
C
C  CALCULATE THE TWIDDLE FACTORS
          C2=C1*C1-S1*S1
          S2=C1*S1+C1*S1
          C3=C2*C1-S2*S1
          S3=C2*S1+S2*C1
  340     JMM4=J-M4
C
          DO 400 I=M4,N1,M4
            J0=I+JMM4
            J1=J0+M
            J2=J1+M
            J3=J2+M
            AR0=X(J0)+X(J2)
            AR1=X(J0)-X(J2)
            AI0=Y(J0)+Y(J2)
            AI1=Y(J0)-Y(J2)
            AR2=X(J1)+X(J3)
            AR3=X(J1)-X(J3)
            AI2=Y(J1)+Y(J3)
            AI3=Y(J1)-Y(J3)
            X(J0)=AR0+AR2
            Y(J0)=AI0+AI2
            IF (J.EQ.1) GO TO 360
C
C  MULTIPLY BY THE TWIDDLE FACTORS
            X(J2)=C1*(AR1+AI3)+S1*(AI1-AR3)
            Y(J2)=C1*(AI1-AR3)-S1*(AR1+AI3)
            X(J1)=C2*(AR0-AR2)+S2*(AI0-AI2)
            Y(J1)=C2*(AI0-AI2)-S2*(AR0-AR2)
            X(J3)=C3*(AR1-AI3)+S3*(AI1+AR3)
            Y(J3)=C3*(AI1+AR3)-S3*(AR1-AI3)
            GO TO 400
C
C  TWIDDLE FACTORS ARE ONE AND ZERO
  360       X(J2)=AR1+AI3
            Y(J2)=AI1-AR3
            X(J1)=AR0-AR2
            Y(J1)=AI0-AI2
            X(J3)=AR1-AI3
            Y(J3)=AI1+AR3
  400     CONTINUE
C
C  CALCULATE THE NEXT SET OF TWIDDLE FACTORS RECURSIVELY
          TMP=C1*CC-S1*SS
          S1=C1*SS+S1*CC
          C1=TMP
  500   CONTINUE
        M4=M
  600 CONTINUE
C
C
C  PICK UP ANY EXTRA FACTORS OF TWO TO COMPLETE THE TRANSFORM
      IF (LOG2N.EQ.2*LOG4N) GO TO 750
  650 DO 700 I=1,N1,INC2
        J=I+INC
        AR0=X(I)+X(J)
        AI0=Y(I)+Y(J)
        X(J)=X(I)-X(J)
        Y(J)=Y(I)-Y(J)
        X(I)=AR0
        Y(I)=AI0
  700 CONTINUE
C
C  UNSCRAMBLE THE OUTPUT ARRAYS
  750 JJ=1
      DO 800 K10=  1,KE10,K110
```

```
      DO 800 K09=K10,KE09,KI09
      DO 800 K08=K09,KE08,KI08
      DO 800 K07=K08,KE07,KI07
      DO 800 K06=K07,KE06,KI06
      DO 800 K05=K06,KE05,KI05
      DO 800 K04=K05,KE04,KI04
      DO 800 K03=K04,KE03,KI03
      DO 800 K02=K03,KE02,KI02
      DO 800 K01=K02,KE01,KI01
        IF (JJ.LE.K01) GO TO 790
        TMP=X(JJ)
        X(JJ)=X(K01)
        X(K01)=TMP
        TMP=Y(JJ)
        Y(JJ)=Y(K01)
        Y(K01)=TMP
  790   JJ=JJ+INC
  800 CONTINUE
C
C  FOR THE INVERSE TRANSFORM ONLY,
C  FORM THE COMPLEX CONJUGATE OF THE SCALED OUTPUT ARRAY
      IF (IFW.GT.0) GO TO 1000
      DO 900 I=1,N1,INC
        X(I)=X(I)/FN
        Y(I)=-Y(I)/FN
  900 CONTINUE
C
 1000 RETURN
C
      END
```

References

1) N.S. Jayant, "Digital Coding of Speech Waveforms: PCM, DPCM, and DM Quantizers", Proc. IEEE, Vol. 62, pp. 611-632, May 1974.

2) R.E. Crochiere, S.A. Weber, and J.L. Flanagan, "Digital Coding of Speech in Sub-bands", B.S.T.J., Vol. 55, pp. 1069-1085, October 1976.

3) J. Makoul, "Linear prediction: A Tutorial Review", Proc. IEEE, Vol. 63, pp. 561-580, April 1975.

4) R. Zelinski and P. Noll, "Adaptive Transform Coding of Speech Signals", IEEE Trans. Acoust. Speech and Sig. Proc., Vol.ASSP-25, pp. 299-309, August 1977.

5) S.J. Campanella and G.S. Robinson, "A Comparison of Orthogonal Transformations for Digital Speech Processing", IEEE Trans. Comm. Tech., Vol. COM-19, pp. 1045-1049, December 1971.

6) P.A. Wintz, "Transform Picture Coding", Proc. IEEE, Vol. 60, pp. 809-820, July 1972.

7) J.M. Tribolet and R.E. Crochiere, "Frequency Domain Coding of Speech", (to be published in IEEE Trans. Acoust. Speech and Sig. Proc.).

8) J.M. Tribolet and R.E. Crochiere, "A Vocoder-Driven Adaptation Strategy for Low-Bit Rate Adaptive Transform Coding of Speech", Proc. 1978 Int. Conf. on Digital Signal Processing, Florence, Italy, September 1978.

9) J.L. Flanagan, M.R. Schroeder, B.S. Atal, R.E. Crochiere, N.S. Gayant and J.M. Tribolet, "Speech Coding", IEEE Trans. Comm., Vol. COM-27, pp. 710-736, April 1976.

10) N. Amed and K.R. Rao, Orthogonal Transforms for Digital Signal Processing, Springer-Verlag, N.Y., 1975.

11) B. McDermott, C. Scagliola, and D. Goodman, "Perceptual and Objective Evaluation of Speech Processed by Adaptive Differential PCM", B.S.T.J., 57. (May-June 1978) pp. 1597-1618.

12) P. Mermelstein, "Evaluation of a Segmental SNR Measure as an Indicator of the Quality of ADPCM Coded Speech", presented at Joint Meeting of the Acoustical Society of America and Acoustical Society of Japan, Honolulu, Hawaii, Nov. 1978.

13) K.S. Shanmugan, "Comments on Discrete Cosine Transform", IEEE Trans. Comput., Vol. C-24, p. 759, July 1975.

14) J. Huang and P. Schultheiss, "Block Quantization of Correlated Gaussian Random Variables", IEEE Trans. Comm. Syst., Vol. CS-11, pp 289-296, 1963.

15) A. Segall, "Bit Allocation and Encoding for Vector Sources", IEEE Trans. Info. Theory, Vol. IT-22, pp. 162-169, March 1976.

16) G. Max, "Quantizing for Minimum Distortion", IRE Trans. Info. Theory, Vol. IT-6, pp. 7-12, 1960.

17) S.P. Lloyd, "Least-Squares Quantization in PCM", unpublished memorandum, Bell Laboratories, 1957.

18) J.D. Markel and A.H. Gray, Jr., Linear Prediction of Speech, Springer-Verlag, N.Y., 1976.

19) A.H. Gray, Jr., R.M. Gray, J.D. Markel, "Comparison of Optimal Quantizations of Speech Reflection Coefficients", IEEE Trans. Acoust. Speech and Sig. Proc., Vol. ASSP-25, pp 9-23, February 1977.

20) M.J. Narashimha and A.M. Peterson, "On the Computation of the Discrete Cosine Transform", IEEE Trans. on Comm., Vol. 26., June 1978.

21) N. Brenner, "Algorithm 467; Matrix Transposition in Place [F1]", Comm. ACM. 16(11), pp. 693-694, 1973.

22) Knuth D., The Art of Computer Programming, Vol. I, Addison-Wesley, Reading, Mass., 1967, p. 180, prob. 12, and p. 517, Solution to prob. 12.