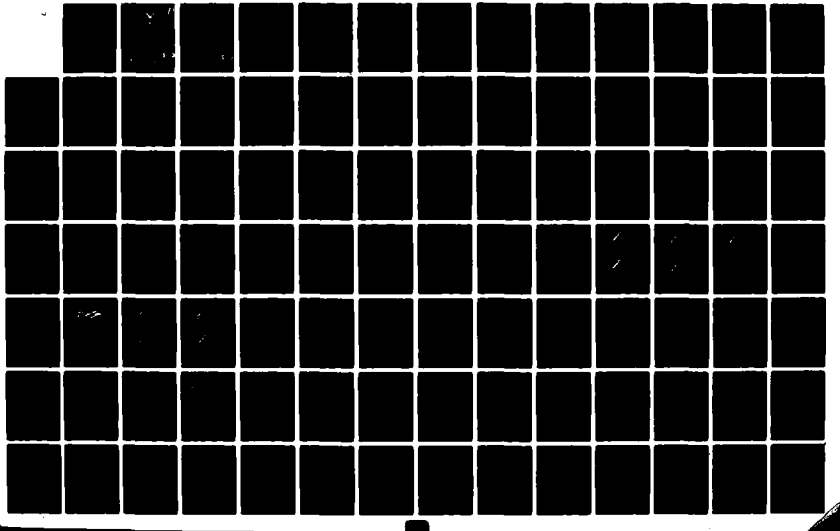
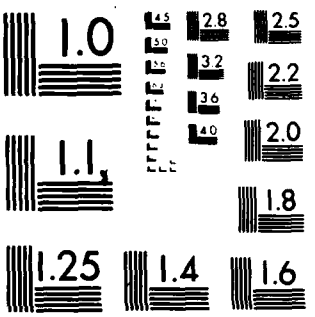
AD-A138 110 IMPLEMENTING LPC (LINEAR PREDICTIVE CODING) ALGORITHMS 1/2
IN THE STUDY OF SP..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. C E MCKOWN
UNCLASSIFIED DEC 83 AFIT/GE/EE/83D-45 F/G 9/4 NL





IMPLEMENTING LPC ALGORITHMS IN THE STUDY OF SPEECH PROCESSING

THESIS

AFIT/GE/EE/83D-45

CRAIG E. McKOWN
2LT USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/EE/83D-45

IMPLEMENTING LPC ALGORITHMS IN THE STUDY OF SPEECH PROCESSING

THESIS

AFIT/GE/EE/83D-45

CRAIG E. McKOWN
2LT USAF

Approved for public release; distribution unlimited

AFIT/GE/EE/83D-45

IMPLEMENTING LPC ALGORITHMS IN THE STUDY OF SPEECH PROCESSING

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology,
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

by

Craig E. McKown
2Lt USAF

Graduate Electrical Engineering
December 1983

PREFACE

Digital Voice Communication is a pervasive phenomenon in our society. Without even noticing it we carry on phone conversations over a digital channel. So, why is digital communication so widely used? One reason is that the digital speech signals can be made more noise immune or even secure than the analog signals. Another reason is the advanced development in integrated circuit technology, which allows easier implementation of digital processing techniques. Whatever the case, the field of digital communication is an exciting field and one which I feel is expanding. Therefore, I am glad that I could prepare my thesis under this topic.

Linear Predictive Coding is one facet of the field of digital communication. The goal of the coding is to reduce the bit rate of the signal sent over the communication channel. The system I developed does not reduce the bit rate very much, but then it is not a true communication system. It is a computer simulation of such a system, which will give the user an opportunity to explore some of the ideas, methods, and problems of LPC.

Since the system developed here is a tool, the most important product of the thesis may well be the user's guide. It presents the programs and demonstrates how a student may actually process speech through an LPC system.

But let me not forget those who made it possible for me to finish this project and receive my degree. I must thank my thesis advisor, Major Larry Kizer, who presented me with the project and let me develop it in my own way.

I wish to thank my readers, Dr. Matthew Kabrisky and Major Kenneth Castor. I also thank Dan Zambon for keeping Bertha, the computer operating most of the time.

I also need to thank Capt Willis Janssen for tolerating me those late nights in the summer when we hacked away until midnight trying to debug code.

I am also indebted to my roommate, Chuck Lutes, who let me use his computer to write this thesis.

Craig E. McKown

TABLE of CONTENTS

PAGE

Preface.....ii

Table of Contents.....iv

List of Figures.....vi

List of Tables.....vii

Abstract.....viii

I. Introduction.....I-1

Background.....I-1
Statement of the Problem.....I-4
Scope.....I-5
Overview of the System.....I-5

II. The Theory of Linear Prediction.....II-1

Autocorrelation Method.....II-7
Covariance Method.....II-8
Solution Algorithms.....II-9
Summary.....II-12

III. Development of the LPC System.....III-1

Description of the LPC Analyzer.....III-5
Description of the Synthesizer.....III-10
Synchronous Analysis.....III-12
Time Contraints.....III-15
Summary.....III-16

IV. Testing and Results.....IV-1

The System.....IV-1
Noise.....IV-5

V. Conclusions and Recommendations.....V-1

Conclusions.....V-1
Recommendations.....V-2
Further Work in Software Development.....V-3
Further Testing.....V-4

CONTENTS

Page

Bibliography.....BIB-1

Appendix A: User's Manual.....A-1

Appendix B: Program Listings.....B-1

Appendix C: Waveshapes of the Speech Signals.....C-1

Appendix D: Code for the Subroutine LATTICE.....D-1

List of Figures

Figure		Page
1-1	A model for the production of speech	I-3
2-1	Cross section of the vocal tract	II-2
2-2	Schematic representation of the vocal tract	II-2
3-1	Block diagram of the Analysis program	III-2
3-2	Block diagram of the Synthesis program	III-3
3-3	Pitch analysis frames	III-8
3-4	Predictor coefficient analysis frames	III-13
4-1	Formant Trajectory (four poles)	IV-6
4-2	Formant Trajectory (six poles)	IV-6
4-3	Formant Trajectory (eight poles)	IV-7
4-4	Formant Trajectory (ten poles)	IV-7
4-5	Formant Trajectory (twelve poles)	IV-8
4-6	Formant Trajectory (no noise)	IV-11
4-7	Formant Trajectory (no noise)	IV-11
4-8	Formant Trajectory (SNR=16dB)	IV-12
4-9	Formant Trajectory (SNR=14dB)	IV-12
4-10	Formant Trajectory (SNR=8dB)	IV-13
4-11	Formant Trajectory (SNR=4dB)	IV-13

List of Tables

Table

4-1 Ranges of the Decision Variables.....IV-3

4-2 Noise.....IV-9

ABSTRACT

This report describes a system which processes speech using linear predictive methods. The system is a software simulation of an LPC analyzer and synthesizer. The system consists of two programs, one of which processes the speech to generate the LPC parameters, and another which processes these parameters to resynthesize the speech. An important aspect of the system is that it enables the user to select from various pitch and coefficient analysis methods. It also allows the user to vary other parameters in order to simulate other changes in the processing scheme.

To test the operation of the system, a regimen of testing was performed by varying the different parameters. A separate program allows a simple method for changing all of the parameters over which the user has control. These parameters are called the decision variables and each has an allowable range of values. The system operated satisfactorally over all values of the decision variables. The flexibilty exhibited by the system in this testing indicates that the system can be a valuable tool for the study of linear predictive coding of speech in the Signal Processing Laboratory at the Air Force Institute of Technology.

CHAPTER I

Introduction

Background

Communication, and in particular, digital voice communication, is of vital concern to the U.S. Department of Defense. This concern is founded in the requirement of the military to maintain command and control over great distances. This is especially apparent in the need for aircraft to maintain contact with forces on the ground. A problem arises, though, when many aircraft need to maintain contact with the same command center. Since only a finite number of radio frequencies (channels) are available, a method of maintaining unambiguous communication is needed. One method of resolving this is time division multiple access, in which each aircraft is allocated a certain amount of time to access the communication channel. The communication system is arranged so that each communicator tranmits and receives only during its allocated time slot. Another method of sharing the channel is frequency division multiplexing, in which each aircraft is allocated a separate portion of the radio spectrum available in the communicaton channel. In either method, the number of users of each channel is limited by both the available bandwidth of the channel and the bandwidths of the users. The bandwidth of the channel is determined by the nature of the channel, the

geometry and physical realization. The bandwidth of the users is a function of the bit rate of the message to be transmitted, as the bit rate of the messages increases, the bandwidth increases. One way to allow for more users is to reduce the bit rate of each user. Linear Predictive Coding (LPC) offers a means of reducing the bit rate of each user when the message is voice communication.

The standard method of digital voice communication is pulse code modulation (PCM), in which the analog voice signal, or waveform, is sampled and quantized. Nyquist's sampling theorem assures us that the sampling rate must be twice that of the highest frequency in the original baseband signal. High quality speech requires frequency components of up to 3000 Hz [Ref 12], so after filtering, sampling is often performed at 8000 Hz. Digitization of the sampled speech is often performed by quantizing at 12 bits per sample, a rate which has proven to enable high quality speech reproduction. Such a system would require a transmission rate of 96 kb/s. Various methods of waveform coders are capable of producing high quality speech, but only at rates above about 16 kb/s [Ref 1].

Linear Predictive Coding (LPC) is a method of digital speech processing which reduces the required bandwidth of the signal by reducing the bit rate required for intelligible communication. Waveform coders, such as PCM, transmit the waveshape of the signal, whereas LPC makes no attempt to maintain the waveshape of the speech signal.

Instead, parameters which describe the speech are determined and are transmitted over the channel to be used to reconstruct the signal at the receiver. These parameters may be the prediction coefficients, which determine the digital filter used to reconstruct the speech, and information about the pitch of the speech. Proper selection of these parameters will enable fairly high quality speech at greatly reduced bit rates. A common model used to diagram the production of speech at the receiver is shown in figure 1-1.

The input to the filter is either a quasi-periodic sequence of impulses spaced at the glottal pitch period, or a random noise source. When the input to the filter is an impulse the voiced portions of the speech such as the vowel sounds are reproduced. When the input is a noise source the unvoiced portions of speech such as the fricatives

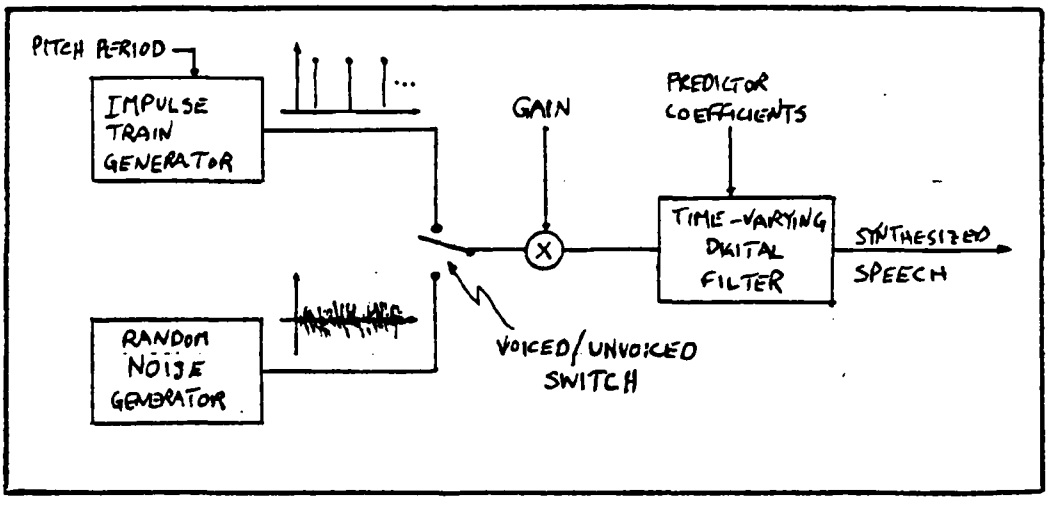


Figure 1-1 A model for the production of speech.

(s,sh,f,th) are reproduced. The gain and coefficients of the filter are determined by linear predictive analysis.

Statement of the Problem

The importance of LPC is indicated by the existence of an Air Force standard for LPC (LPC-10). As LPC becomes more prevalent a need exists for a system available at the Air Force Institute of Technology which can demonstrate some of the features and operations of LPC. Most implementatons of LPC are in hardware, with most of the system hidden within a "black box." These factors led to the need for the development of a software model which would offer a better opportunity to examine the system.

A number of algorithms exist which can be used to determine the filter coefficients. Among these are the autocorrelation and the covariance method. Also available are a number of methods of pitch detection and extraction. A software simulation is needed which would incorporate these various algorithms and methods into a single, flexible model. This model could be used as a learning device and as a tool for further study of LPC. It would allow the student or researcher an easy means to investigate the software of the system and vary the algorithms used, consider changes of the algorithm parameters and examine intermediate results. It would also provide a means of addressing some of the problems confronting LPC, in particular, the noise problem.

Scope

The system presented in this thesis is designed especially for the Signal Processing Laboratory at AFIT, where it can operate as a useful tool for the study of the general LPC method of coding speech. It is strictly software; the code is written in FORTRAN5, developed on and accesible from the Data General Eclipse S/250 computer in the laboratory. It uses existing hardware and software for the audio interface. It is meant to be easily used, easily understood, and user friendly. It should be easy to update, expand, or modify. It was designed to run as close to real time as the constraints of the laboratory would permit.

Overview of the System

The system is divided into two main programs, an analysis program and a synthesis program. This format was chosen as it best simulates the tranmitter and reciever nature of the LPC speech communication system. The inputs to the analysis program are digitized speech and the necessary decision variables (these will be explained later). The outputs of this program are the LPC parameters. The synthesizer uses these parameters to reproduce the speech.

The flexibilty of the system is afforded by the extensive use of subroutines. The system inputs the speech in time segments, called frames, and operates on these

segments sequentially. This is a recurring process, and most of the calculations are performed on each frame. The subroutine structure allows easy access to the routines which perform certain portions of the calculations, such as pitch detection, coefficient generation, and other systematically used operations. For instance, the subroutines which perform coefficient generation are grouped together, yet only one subroutine is used (although it is used on each frame) during the execution of the program. The other subroutines are retained for the case where another method of coefficient generation is required. The use of subroutines makes it easy to expand the system by adding new routines which offer different methods of performing the required calculations.

The LPC analyzer is the heart of the system. Most of the decision variables affect the operation of the analyzer, because they determine which subroutines will be used to produce the necessary parameters for transmission. It reads digitized (PCM) speech from a contiguous file of integer values. It scales the incident speech if necessary, places it in a floating point form, and writes it to an array. The parameters (pitch information, predictor coefficients, and energy) are calculated for each frame and then written to a sequential file which is the input to the synthesis program. This file acts as a communication channel between transmitter and receiver and is referred to as the channel file. Before any speech is processed, key parameters which

are needed by the synthesizer are written to the channel file. These parameters are needed by the synthesizer so that it can correctly match its decoding and synthesis scheme to a form compatible with that of the analyzer. If the forms do not match, the LPC parameters will be read incorrectly and speech will be impossible to reproduce.

The synthesizer reads the information from the channel file and processes it to create intelligible speech. It reads the pitch data and the length of the speech to be proccessed. It then generates either pseudorandom noise or a pulse train, which it writes to a array. This array is the input to the digital filter which is described by the prediction coefficients read from the channel. The output of this filter is written to a contiguous file and the system then processes the next block of information. After the entire channel file has been read, the output speech is scaled so that it may be listened to with the use of the "Audiohist" or the "Audiomod" program prepared by a previous student [Ref 3] and available as a utility program on the system in the Signal Processing Laboratory.

CHAPTER II

The Theory of Linear Prediction

Linear prediction is a method of analyzing a speech waveform so that the complete waveform need not be transmitted over a communication channel. A linear prediction system takes a digital speech signal and processes it so that only the "essence" of the signal remains; no attempt is made to maintain the waveshape of the signal. For our purpose, the essence of the signal is a parametric model of the signal, where these parameters can be used to reconstruct the signal. The linear prediction system consists of two major operations or processes. One process analyzes the incoming speech and extracts the relevant parameters and transmits these over a communication channel. Another process at the receiving end of the channel transforms these parameters into speech. This transformation is based on a time-varying digital filter with predictor coefficients which model the vocal tract of the speaker (see figures 2-1 and 2-2). Since the vocal tract changes shape slowly it can be considered fixed over a time interval on the order of 10 ms, and the digital filter can characterize the vocal tract over this short interval [Ref 12]. The input to this filter is the assumed excitation of the actual human vocal tract: glottal pulses occurring every pitch period, or random noise.

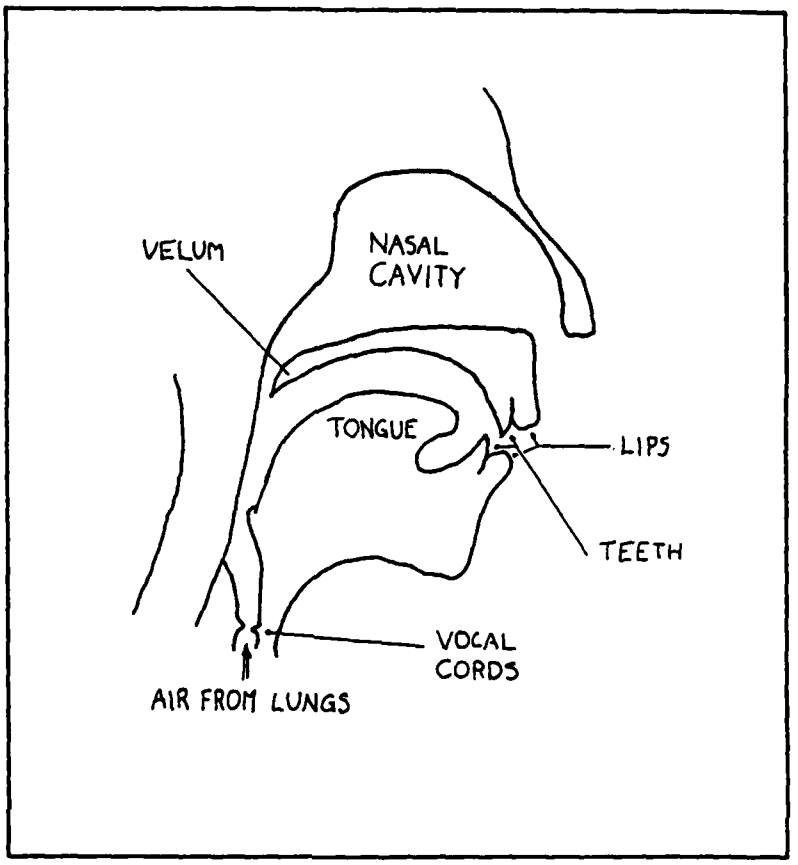


Figure 2-1 Cross section of the vocal tract showing the major anatomical structures involved in speech production.

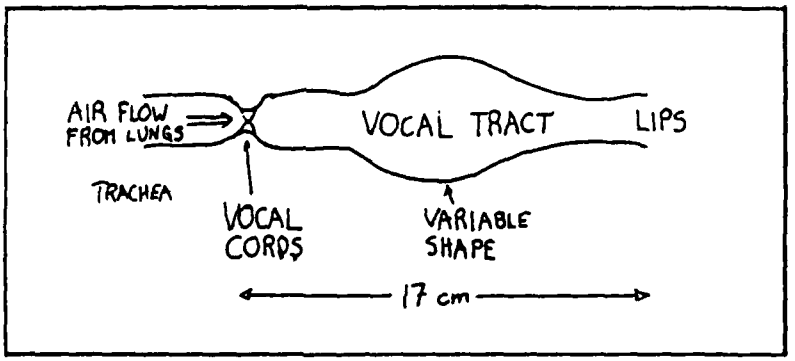


Figure 2-2 Schematic representation of the vocal tract.

The question addressed by linear prediction is: how do we find these predictor coefficients?

Speech, of course, is an analog process, so it must be digitized before it can be processed by linear prediction methods. This is usually accomplished by pulse code modulation (PCM) in which the analog signal is quantized in time (sampled) with a sampling frequency of f_s, and quantized in amplitude. For the analog waveform, s(t), the sampled waveform can be expressed as

$$s(nT) = s(t)\Big|_{t=nT} \qquad (2\text{-}1)$$

where T is the time between samples ($T=1/f_s$). Since f_s, and therefore T remain constant (in our case f_s = 8000Hz), we can write s(nT) as s(n) with no loss of generality.

If we assume that the signal, s(n), is the output of a system (our assumption above concerning speech at the receiver being the output of a filter allows this) with input u(n), then that signal can be expressed as a linear function of the past outputs and the present and past inputs. That is, the output can be predicted by a linear combination of inputs and outputs. Hence the description of this scheme as linear prediction. This relation is written as

$$s(n) = -\sum_{k=1}^{p} a_k s(n-k) + G\sum_{m=0}^{q} b_m u(n-m) \quad , b_o = 1 \qquad (2\text{-}2)$$

where a_k , $1 \leq k \leq p$, b_m , $1 \leq m \leq q$ and G are the

parameters of the system. The goal of the linear predictor is to determine the values of these parameters.

Entering the frequency domain, we can take the z-transform of both sides of (2-2) to obtain the transfer function, H(z), of the digital system. The transfer function is the ratio of the output to the input and can be expressed as

$$H(z) = \frac{S(z)}{U(z)} = \frac{1 + \sum_{m=1}^{q} b_m z^{-m}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \qquad (2\text{-}3)$$

where S(z) is the z-transform of s(n) and U(z) is the z-transform of u(n).

This equation describes a pole-zero model of the system. Variations on this model are the all-zero model, where $a_k=0 \quad ,1 \leq k \leq p$; and the all-pole model, where $b_k=0 \quad ,1 \leq k \leq q$. Historically, the all-pole method of analysis has been by far the most widely used method of linear prediction [Ref 7]. For the all-pole model the equations which must be solved form a linear set, whereas even the simplest pole-zero model gives a set of non-linear equations [Ref 10:472]. Since the all-pole model will greatly simplify the calculation of the coefficients, we will only concern ourselves with this model. Therefore, the

transfer function of interest is

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}} \qquad (2\text{-}4)$$

A(z) will be referred to as the inverse filter, and the coefficients a_k , $1 \leq k \leq p$ will be referred to as the predictor coefficients.

By taking an inverse z-transform, we return to the time domain and get the relation

$$s(n) = - \sum_{k=1}^{p} a_k s(n-k) + G\, u(n) \qquad (2\text{-}5)$$

From this equation it is clearly evident that the output sequence, s(n), can be generated with only one input, u(n) and p previous outputs.

If we assume that the input u(n) is unknown [Ref 7], we can calculate a prediction of s(n), $\tilde{s}(n)$ which is based strictly on ther past outputs. This assumption gives us a result which is independent of the input and can be written as

$$\tilde{s}(n) = - \sum_{k=1}^{p} a_k s(n-k) \qquad (2\text{-}6)$$

Now we will define the predictor error, e(n), as the difference between the original signal, s(n), and the

predicted signal, s(n). That is

$$e(n) = s(n) - \tilde{s}(n) = s(n) + \sum_{k=1}^{p} a_k s(n-k) \qquad (2\text{-}7)$$

Given s(n), we can define the total squared error, E, as

$$E = \sum_{n} e^2(n) = \sum_{n} [s(n) + \sum_{k=1}^{p} a_k s(n-k)]^2 \qquad (2\text{-}8)$$

By definition, the most accurate predictor coefficients result in the least error. To find the minimim squared error, we set the derivative to zero.

That is

$$\frac{\partial E}{\partial a_i} = 0 \qquad ,1 \leq i \leq p \qquad (2\text{-}9)$$

Equations (2-8) & (2-9) will reduce to the set of equations

$$\sum_{k=1}^{p} a_k \sum_{n} s(n-k)s(n-i) = -\sum_{n} s(n)s(n-i) \quad ,1 \leq i \leq p \quad (2\text{-}10)$$

These equations are called the normal equations [Ref 7]. Given any signal, s(n), (2-10) forms a set of p equations in p unknowns which can be solved to give the predictor coefficients which minimize E. The parameter, p, is the number of poles, and consequently the number of predictor coefficients in the transfer function. Note that the range of summation over n is the range of the signal for which the error will be minimized and remains unspecified.

Autocorrelation Method

If we let the range of summation over n be from $-\infty$ to $+\infty$, we will get a global minimization of the error and equation (2-10) reduces to

$$\sum_{k=1}^{p} a_k R(i-k) = -R(i) \qquad 1 \leq i \leq p \qquad (2\text{-}11)$$

where R(i) is the autocorrelation, and is defined as

$$R(i) = \sum_{n=-\infty}^{\infty} s(n)\, s(n-i) \qquad (2\text{-}12)$$

Since the signal is known over only a finite duration, we divide the signal into frames and assume that the signal s(n) is identically zero outside of the interval $0 \leq n \leq N-1$. A suitable way to express this is as

$$\hat{s}(n) = s(n+N)\, w(n) \qquad (2\text{-}13)$$

where w(n) is a window function which is identically zero outside of the interval $0 \leq n \leq N-1$. This windowing process produces frames which are N samples wide. Since excessive errors will be encountered at the frame boundaries because of our drastic assumption of zero outside of the frame, we need to taper the edges of the window to zero. To accomplish this we use a Hamming window. For simplicity of notation we will drop the index N and the caret and speak only of the signal s(n) which is properly only a portion (one frame) of the complete signal.

Note that R(i) is an even function, that is

$$R(i) = R(-i) \qquad (2\text{-}14)$$

The coefficients R(i-k) form an autocorrelation matrix. Because R(i) is even and is a function of only the difference of the indices, the resultant autocorrelation matrix is symmetric and all of the elements along a diagonal are equal. Such a matrix is called symmetric Toeplitz. This fact makes the linear system of equations easy to solve by recursive methods.

Covariance Method

Another technique for producing the predictor coefficients is called the covariance method. If we minimize the squared error over the finite interval $0 \leq n \leq N-1$, we get the set of equations

$$\sum_{k=1}^{p} a_k c(k,i) = - c(0,i) \qquad ,1 \leq i \leq p \qquad (2\text{-}15)$$

where c(i,k) is called the covariance, and is defined as

$$c(i,k) = \sum_{n=0}^{N-1} s(n-i)\ s(n-k) \qquad (2\text{-}16)$$

Because we define our range of summation over a finite interval, we need not window the signal, but as we will see in Chapter 4, windowing vastly improves the results of the inversion process needed to calculate the predictor coefficients.

The covariance terms are symmetric, that is

$$c(i,k) = c(k,i) \qquad (2\text{-}17)$$

and make up a symmetric covariance matrix. However, the terms of this matrix unlike the terms of the autocorrelation matrix are not equal along the diagonals.

Solution Algorithms

The solution involves inverting the matrix which describes the set of p equations to be solved. A number of algorithms exist for inverting matrices with a computer. Of main concern in developing the solution algorithms is a need for simplicity, ease of implementation in the software, and reduction of the number of calculations. The Toeplitz nature of the autocorrelation matrix makes the system of p linear equations,

$$\sum_{k=1}^{p} a_k R(i-k) = -R(i) \qquad 1 \leq i \leq p \qquad (2\text{-}11)$$

easy to solve and reduces the number of computations required. Levinson developed an elegant recursive method for solving such equations. Durbin further expanded on the recursion by exploiting the fact that when the equation is expanded into matrix form, the right side of the equation is contained on the left side [Ref 7].

The recursive solution attributed to Durbin is usually presented as [Ref 11]

1. $E(0) = R(0)$ (2-18a)

2. $k_i = -[R(i) - \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j)]/E(i-1)$,$1 \leq i \leq p$ (2-18b)

3. $a_i^{(i)} = k_i$ (2-18c)

4. $a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)}$,$1 \leq j \leq i-1$ (2-18d)

5. $E(i) = (1-k_i^2)\, E(i-1)$ (2-18e)

After the values are solved recursively for i=1,2,...,p the final solution, giving the p predictor coefficients, is

$$a_j = a_j^{(p)} \qquad 1 \leq j \leq p \quad (2\text{-}19)$$

The solution is unaffected if the autocorrelation values are scaled by a constant. Usually, the autocorrelation values are normalized by dividing them by R(0), giving normalized autocorrelations, which, except for R(0), are all less than one.

A by-product of the recursive method is E(i) which is the predictor error for a predictor of order i. If the autocorrelation function is normalized, this error value will also be normalized. This parameter is important because given the output filter described by the p predictor coefficients, the value E(p) is proportional to the gain required to reproduce the speech signal. In the prediction model, this parameter will represent the gain of the output filter.

The covariance method also uses a recursive method to

determine the coefficients [Ref 10]. The system must first be initialized by setting the following:

$$E(0) = c(0) \tag{2-20a}$$

$$B(0) = c(1) \tag{2-20b}$$

$$k_1 = -c(0)/c(1) \tag{2-20c}$$

$$a_o(1) = 1 \ , \ a_1^{(1)} = k_i \tag{2-20d}$$

$$E(1) = \left[E(0) - k_1\right]^2 B(1) \tag{2-20e}$$

The recursive equations can be written as:

1. $$b_i^{(i-1)} = 1. \tag{2-21a}$$

2. $$G = 1/B(n) \sum_{j=1}^{n+1} R(i)\ b_j^{(n)} \tag{2-21b}$$

3. $$B(i-1) = \sum_{j=1}^{i} R(i)\ \ b_j^{(i-1)} \tag{2-21c}$$

4. $$k_i = 1/B(i-1) \sum_{j=0}^{i-1} R(i)\ a_j^{(i-1)} \tag{2-21d}$$

5. $$a_j^{(i)} = a_j^{(i-1)} + k_i b_j^{(i-1)} \tag{2-21e}$$

6. $$a_i^{(i)} = k_i \tag{2-21f}$$

7. $$E(i) = (1-k_i^2)E(i-1) \tag{2-21g}$$

At this point step m is complete. After p recursions, the

final solution is

$$a_i = a_i^{(p)} \qquad ,1 \leq i \leq p \qquad (2\text{-}22)$$

which gives the predictor coefficients for the output filter of order p. The parameter E(p) is proportional to the gain of the system.

Summary

Linear Prediction is a method of parameterizing a signal. Using minimum mean square error techniques, the procedure generates the filter coefficients which describe the system producing the signal. For speech, this system is the human vocal tract. With these coefficients and a valid excitation to the output filter, the speech can be reproduced to yield intelligible results.

CHAPTER III

Development of the LPC System

To attain the flexibility required in this linear predictive coding model, the system is divided into two separate programs. These programs are the LPC analyzer and the LPC speech synthesizer. Flow diagrams of these programs are presented in figures 3-1 and 3-2. The two programs are coupled by a file in the computer, which is called the "channel file", and which can be considered as a communication channel. The analyzer writes speech information to the channel, the vocoder reads this information and from it reproduces a synthesized version of the original speech. Both of these driving programs are composed of a number of subroutines which perform most of the calculations. This extensive use of subroutines is intended to make the system more flexible as well as easier to understand. For the most part, bookkeeping is performed by the main programs, whereas most of the LPC calcuations are performed by the subroutines. Intermediate results and parameters can be examined by looking at the relevant subroutines. The method of calculation can also be examined. This can be done by looking at the code or by placing "type" or "write" statements into the subroutines. Adding new methods simply requires the addition of the proper subroutines and their corresponding calling

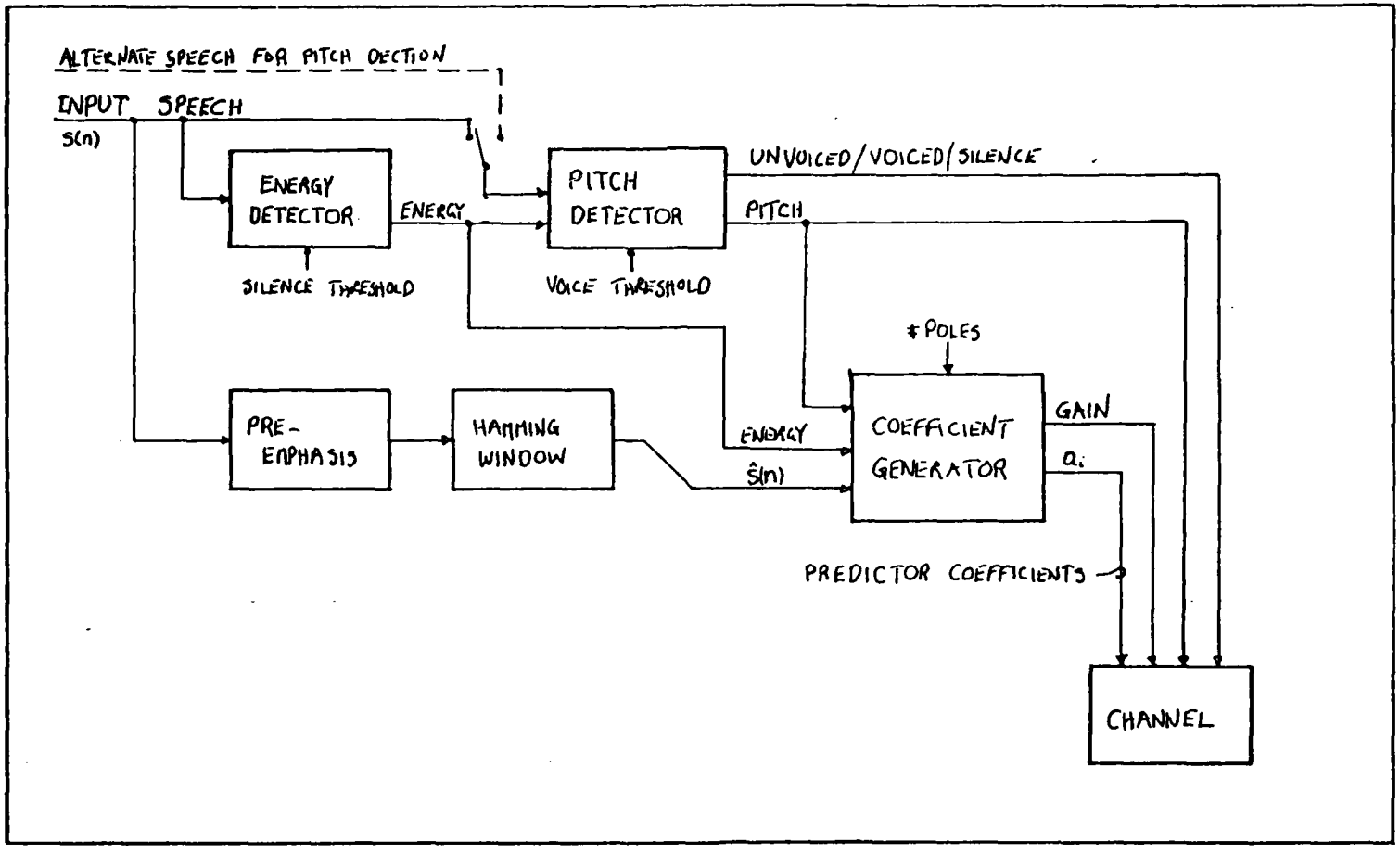


Figure 3-1 Block diagram of the Analysis program (PREDICT).

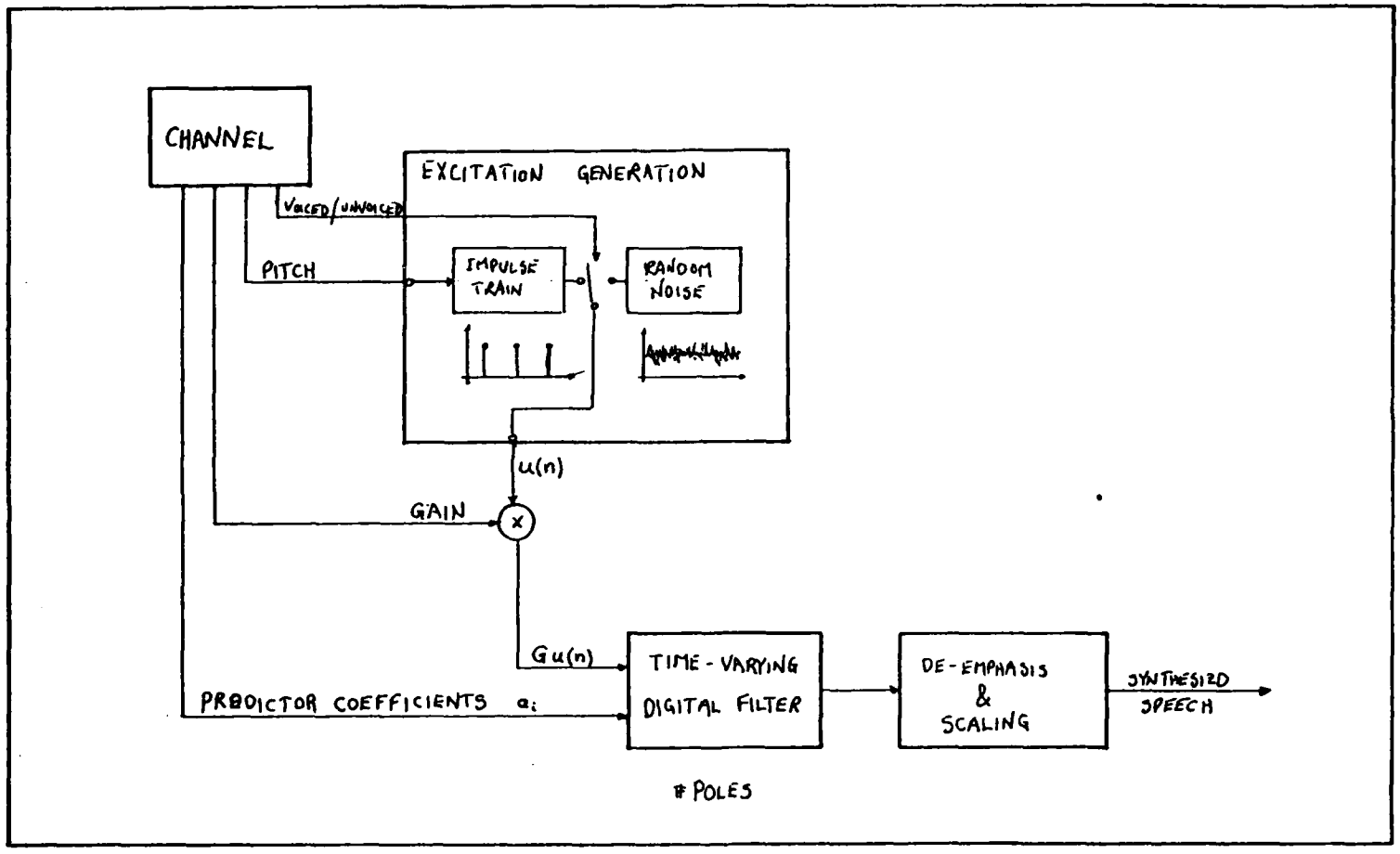


Figure 3-2 Block diagram of the Synthesis program (VOCODE).

statements in the main program.

Further flexibility is attained by the use of "decision variables."These decision variables are preset by the user and control the operation of the main programs. These variables select different prediction methods, vary the number of poles used in the analysis, and affect other scaling or test parameters. Two methods of setting the decision variables are available to the user. The easiest is to simply run the program and wait for the prompts. The other method is to write the decision variables to a file, with the aid of a program (SETUP) which is designed strictly for this purpose. This method is preferred and is useful if the user wants to hear different segments of speech without having to worry about the decision variables selected. Because the file is self-contained, the user need not be interrupted by any prompts. The decision variables will be identified and named as they are encountered in the description of the system.

The modularity of the system helped considerably in the construction and testing of the programs and subroutines. Each part of the program could be tested independently before it was consolidated into the complete program. Modularity also allowed concurrent developement of the analyzer and the vocoder because results from the analyzer could be tested by direct application of the vocoder. This also allowed various values of decision variables to be tested at each stage of the developement.

Description of the LPC Analyzer

The input to the LPC analyzer is digitized speech. The analyzer processes this signal and generates the parameters which will be written to a channel file. The first stage of the analyzer creates or opens the files to which information will be read or written. Two files are opened, one which contains the digitized speech to be coded, and another which contains the decision variables. The file which contains the incoming digitized speech must be contiguous with each block containing 256 integer valued samples. One file, the channel file, into which will be written the LPC speech information, is created. Next, the decision variables are determined, either by prompts from the terminal, or from the file containing these variables. The variables pertaining to the operation of the vocoder [number of poles in the synthesis filter (POLES), pre/de-emphasis (NEMP), unvoiced gain factor (UNGA), and the shape of the glottal pulse (NGLT)] are then written to the channel file. This completes the initialization of the system. The rest of the system is a large loop which is repeated until the input speech data is exhausted.

The first order of business within the loop is to load a large array with five blocks of speech (each block contains 256 samples) from the contiguous input file. This large array is needed because this data is written to two small arrays, one used for pitch detection, and one used for

predictor coefficient generation. Each array contains one frame of speech. The length of each frame is set by a decision variable (MAXFR). The large size is so that a wide range of frame sizes can be used by the analyzer.

Using counters to keep track of where the process is in terms of blocks and array members, a portion of the large array is written to a smaller array. This smaller array contains one frame of speech to be processed for energy and pitch. This frame is first processed by the energy subroutine. This subroutine finds the energy (sum of the squares) in the frame to determine if the data can be considered silence or speech. The test threshold is a decision variable (THRESH) which is set by the user. The energy of the frame is a functon of its length, therefore, if the length of the frame is changed, the decision variable THRESH should be changed by a proportional amount to maintain consistent results. If the threshold is not exceeded, the signal is considered silence, and no more calculations need be made on this frame. The subroutine has a memory of three previous energy calculations. The need for the memory will become apparent when the nature of the pitch detection and the synchronous nature of the analysis is discussed.

If the frame has sufficient energy to be considered speech and not background noise, the pitch is then calculated. The pitch detection routines perform two tasks: determining the voiced quality of the speech (whether the

frame is voiced or not), and then the pitch of the speech if the frame is voiced. The frame for pitch is moved at an interval (MAXPT) which is set by the user. Pitch is therefore updated every MAXPT samples and is assumed constant over the frame.

One method of pitch detection is essentially a correlation process. The frame array is correlated against itself, and the peaks which fall within an allowable range of times are tested to find a maximum. The maximum peak is then tested against a threshold which is set by the user. This threshold is the decision variable STHR. If the magnitude of the peak falls below the threshold, the speech is declared unvoiced, otherwise it is declared voiced. The pitch is a function of the location of the peak. Since the range of fundamental pitch of most human speakers falls within a fairly narrow range (70-350 Hz) [Ref 9], only a narrow range of peak positions need to be considered. Due to the harmonics and the formant structure of speech, the pitch period is a difficult calculation and is quite prone to error. Therefore interpolation is employed to smooth the curve described by the pitch values.

This interpolation delays the final value of the pitch by three frames (see figure 3-3). An estimate of the pitch in the first pitch analysis frame is computed. The frame start is shifted by the pitch shift interval (MAXPT), and the pitch of the second frame is estimated. Similarly, the pitch of the third frame is estimated. These three values

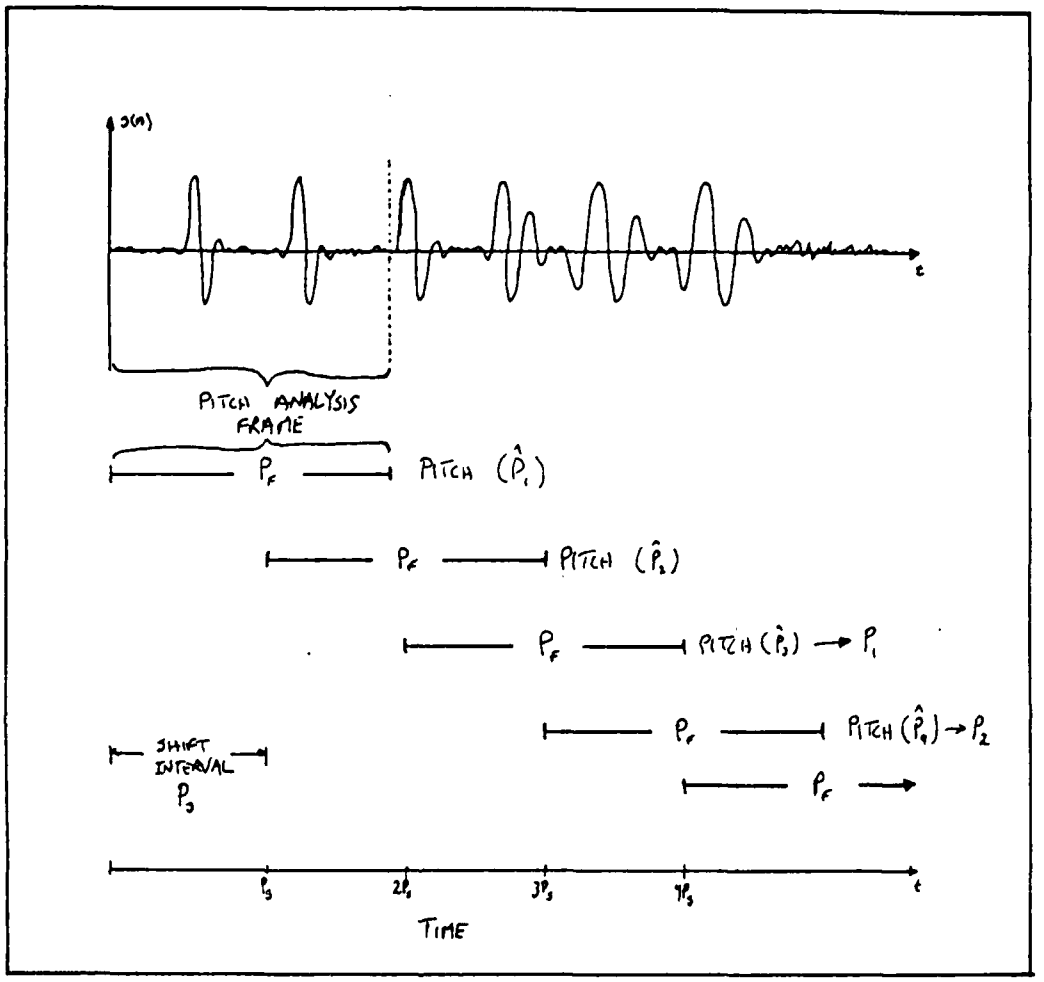


Figure 3-3 Relationship between pitch analysis frames showing a 2:1 overlap.

are used to interpolate a better estimate of the pitch in the first frame. The pitch in the following frames is determined in a similar manner. For instance, the estimate of the fourth frame is used along with the estimates of he second and third frame to interpolate a better estimate of the pitch in the second frame. This delay requires that the calculation of the predictor coefficients be equivalently delayed. The pitch detectors must therefore have a memory of three to perform the interpolation and to accomodate the delay of the coefficient analysis.

After the pitch is determined by interpolation, the LPC coefficients and the gain must be calculated. The coefficient analysis frame can be pre-processed in a number of different ways. If desired, the speech can be pre-emphasized. The pre-emphasis is accomplished wih the operation

$$y(n) = x(n) - .9x(n-1) \qquad (3\text{-}1)$$

The decision variable which controls this is NEMP. Also available to the user is the option to window the frame. A decision variable, H1, controls whether or not a Hamming window is used on the speech. After pre-processing, the LPC calculations are straight-forward, and are performed by one of the algorithms presented in Chapter II. A decision variable (MP) controls which algorithm to use, and therefore which subroutine to perform. A decision variable (POLES) also contols the number of poles used in the analysis. The system is unable to operate with more than 20 poles, because

of constraints on the size of some of the arrays. The predictor coefficients are computed synchronously, that is the start of the frame used by the coeffiecient generating routines is set by the shift introduced by the pitch period. This avoids the problem of having analysis extend over two adjacent frames.

The last task during processing of each frame is to write the relevant parameters to the channel file. These parameters are the voiced quality of the speech, the pitch, the output analysis frame size, the predictor coefficients, and the gain of the system. The output analysis frame size is the shift interval for the start of the next coefficient analysis frame.

Description of the Synthesizer

The LPC synthesizer produces speech from the parameters read from a channel file. The first stage of the synthesizer program opens the channel file and creates the file to which synthesized speech will be written. The program is initialized by reading the first four values from the channel file. These values are the number of poles used in the analysis and the synthesis (POLES), the glottal pulse shape (NGLT), the unvoiced gain factor (UNGA), and the value of the flag indicating whether pre-emphasis was employed at the input to the analyzer (NEMP). The unvoiced gain factor is used to normalize the unvoiced excitation to the output filter. The vocoder then synthesizes one variable length

frame of speech at a time. The frames have varying length because of the synchronous method of coding. To synthesize each frame, the vocoder first reads the pitch information, the length of the frame to be produced, and the gain. It then reads the predictor coefficients. The pitch information drives the synthesis process.

If the frame is voiced, an array simulating the glottal pulse drives the digital output filter. Two pulses are generated at intervals separated by the pitch period, and are written to an array of length twice the period. This array, the predictor coefficients, and the gain then drive the subroutine "THROAT" which is the digital output filter. The output of this filter is the synthesized speech and is written to the output file.

If the frame is unvoiced, a noise generation routine is called. This double-precision routine uses a uniform random number generator to produce a normal random number sequence. This routine writes the sequence to an array. The gain from the input file is scaled by the unvoiced gain factor to give a value for the gain to drive the digital output filter. This scaling is required because the excitation of the filter (the noise array) is not normalized in energy with the excitation of the filter when the speech is voiced. This array, the predictor coefficients, and the gain then drive the digital output filter. The output of this filter is the synthesized speech, which is written to the output file.

If the frame is silence, the output filter is by-passed and zeros are written directly to the output file.

If pre-emphasis was used by the analyzer, then de-emphasis must be employed before the speech is written to the output file. The inverse function of the pre-emphasis is used for this

$$y(n) = x(n) + .9y(n-1) \qquad (3-2)$$

The speech is finally scaled so that it can be listened to by using the "AUDIOHIST" or "AUDIOMOD" programs.

Synchronous Analysis

The LPC system developed in this thesis uses a synchronous method of analysis. Synchronous analysis requires an update of the predictor coefficients once every pitch period (see figure 3-4). Because of the delay in determining the correct pitch, the predictor coefficients are generated with a two frame delay with respect to the pitch detection. The shift interval of the coefficient analysis is a multiple of the pitch period (P_1 in figure 3-3). When the start of an analysis frame falls after the start of the next pitch detection frame, the value of this pitch (P_2) is used as the shift interval. The frame start is then shifted as before until a new pitch value is needed. At this point, a new frame is estimated for pitch.

In a synchronous system, analysis on each voiced segment of speech begins at the beginning of the pitch period and the analysis frame is shifted at an interval

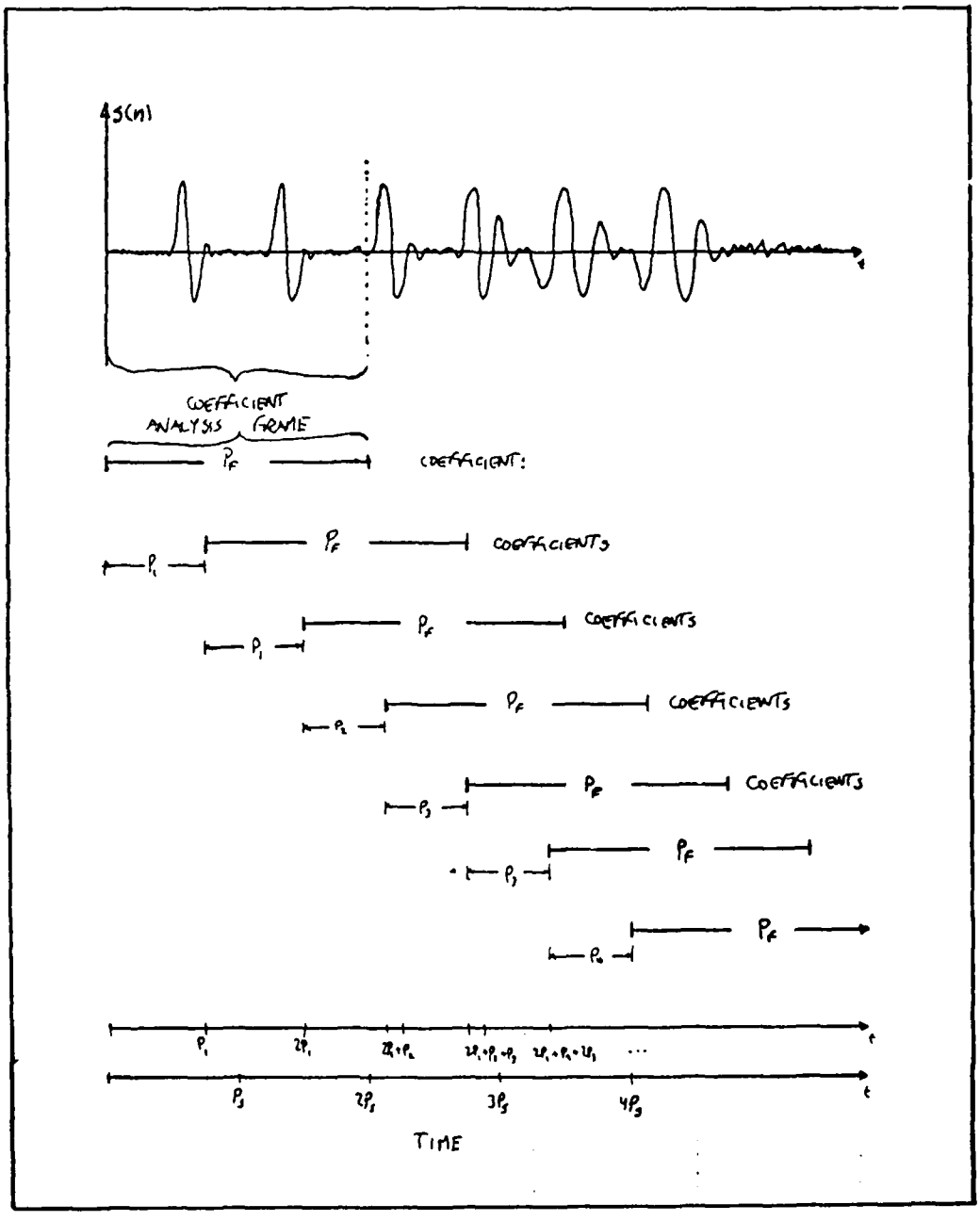


Figure 3-4 Relationship between coefficient analysis frames for a synchronous analysis system.

which is a multiple of the pitch period (see figure 3-3). The vocoder only synthesizes the speech between the start of every analysis frame. For this reason is the analysis frame size written onto the channel file. Synchronous analysis avoids the problem of having a pitch period of voiced speech straddling the boundary between two consecutive frames. When analysis straddles two frames, the coefficients must be interpolated for the portion of speech reproduced over the boundary. The interpolated values of these coefficients are not guaranteed to give stable results.

During unvoiced speech, the shift interval is constant. A constant frame rate (MAXFR) is used for shifting the coefficient analysis.

The three frame delay of the pitch detector requires a delay of the coefficient analysis by a corresponding time period. Therefore the frame starts and boundaries of the pitch analysis frame and the coefficient analysis frame are different. This is the reason for the memory of the pitch and energy detectors and the initial large array. With this array, the current pitch can be calculated with one portion of the data, and used for interpolation of past pitch. At the same time, the delayed pitch can be used to determine the correct starting location in the past for predictor coefficient generation. Therefore the system can perform coefficient calculation in step with pitch detection. This does tend to limit flexibility because it does not allow asynchronous analysis.

Some complexity is added because of the synchronous nature of the analysis. For example, the bookkeeping for each frame is doubly complex; two sets of counters must be maintained. It is also confusing that the pitch detection and predictor coefficient generating routines do not work on the same data simultaneously. This requires that the predictor coefficient generating routines operate after the pitch has been calculated for the final frame. Substantial gains are realized, though, because interpolation of predictor coefficients need not be performed. Counters would also have to be maintained in vocoder to mark the beginning and ending of boundary-overlapping sections of speech.

Time Constraints

Unfortunately, the system does not run in real-time. A number of factors affect the speed of the system, among these: the software implementation, the audio interface, and the speed of the computer.

The biggest constraint on time is the software in the system. Since the software is written in a high level language, its speed is limited by the constraints of the language. Some of these constraints are the time necessary to write to a file and the operation rate of the minicomputer. A hardware implementation would not be under these constraints. The software is also a bit cumbersome in that it must be flexible enough to operate properly with

many different possible prediction schemes. For use in the laboratory, the transmitter and the receiver programs must be run sequentially, whereas they would operate in parallel if in a normal communication configuration.

Another time constraint is that the audio interface in the lab is not prepared for real time events. The program which performs the digital to analog calculations and channel calling routines required to listen to the synthesized speech must read a file first. The size of this file is limited by the program to less than three seconds. Therefore, long utterances are impossible to process and listen to without interuption and user interaction with the system.

Summary

The two-program nature of the LPC system is used to imitate the tranmitter and receiver nature of a true communication link. These two programs are the LPC analyzer and the LPC synthesizer. The channel between them is a file written in the memory of the computer. Subroutines are used extensively to permit easy examination of internal results and provide flexibility to run or add different subroutines which perform the same analysis in different ways. Synchronous analysis is employed to simplify synthesis of the output speech.

CHAPTER IV

Testing and Results

The System

This thesis designed an LPC speech processing system which operates in the Signal Processing Laboratory at the Air Force Institute of Technology. It is an operational model and replicates a number of aspects of a true LPC speech communication system. Unfortunately, this system does not run in real time, as it is written entirely in software and must access and write files which reside in the minicomputer's memory. A real LPC processor and synthesizer would receive and transmit signals over a communication channel, and except for buffering, would require no reading or writing from files. Most of these operations would be handled by more time-efficient hardware.

A simple test of the system consists of actually processing an utterance. Each utterance is a digitized version of a sentence spoken by a human. These utterances or speech files are relatively noise-free so that the noise problem of LPC would not need to be considered. These utterances were successfully processed by the system to give intelligible results.

To demonstrate the flexibility of the system, various combinations of shift intervals, analysis window size, prediction methods, pitch detection algorithms, and

threshold values were tested. The results of these combinations indicate that the system is flexible. Table 4-1 shows the allowable ranges of the decision variables. The recommended values on this table indicate values for the decision variables which seemed the give the best results. Other testing showed that with only a very few exceptions, the system could handle all of the combinations with which it was tested. The exceptions are noted below.

On occasion, especially when a low pitched voice was processed, the vocoded speech was so unintelligible that it was impossible to determine that an utterance was present. This problem arises from the nature of the synchronous analysis. Synchronous analysis demands that two shift intervals be maintained, one for the pitch analysis window and one for the coefficient analysis window. During voiced speech, the shift interval for the coefficient analysis window is based on the pitch period, but the bookkeeping counters are based on the shift interval of the pitch detector. If the shift interval is less than the length of the longest pitch period during voiced speech, the analysis window used to calculate the prediction coefficient is incorrectly bounded. Results in this case are consistently poor, and result in unstable filters which produce unintelligible clicks, buzzes, and squeals. To eliminate this problem, the frame shift interval must be increased to accomodate the lowest pitch frequency of the file. For the file which contained the lowest pitch this interval is

Table 4-1

Ranges of the Decision Variables

Decision Variable		Lower Limit	Upper Limit	Recommended Value
POLES		6	20	16
MP	a	-	-	1
MAXFR		100	400	200
MAXPT	b	80	200	100
NEMP	a	-	-	1
NGLT	a	-	-	3
H1	a	-	-	1
MPCH	a	-	-	0
NPCS	a	-	-	1
STHR		0.0	1.0	0.35
SCAF		1.0	1000.0	1.0
THRESH		0.0	1000000.0	250.0
UNGA		0.001	100.0	0.1

a) These variables are flags which determine whether subroutines will be performed or not. They do not have upper or lower limits over a range.

b) It is recommended that the value of MAXPT be half the value of MAXFR. This gives a 2:1 overlap.

between 80 and 100 samples.

Two methods of predictor coefficient generation are included in the system and were examined extensively. Both give acceptable results and produce intelligible speech at the output. Although theory indicates that the covariance method of prediction need not be windowed, much better results are attainable when the incoming speech signal is weighted by a Hamming window. In this context, better results mean higher quality. Without the windowing, the method often produces unstable output filters. With the windowing, the resultant speech exceeded the quality of spech produced with the autocorrelation method.

The Sift routine [Ref 9,10] for pitch detection was the only satisfactory pitch detector implemented. The Sift algorithm uses an inverse filtering technique to cancel the effect of the formant structure. In a non-noise environment, this detector can consistently differentiate between voiced and unvoiced speech It can also successfully determine the pitch to produce natural sounding speech. An autocorrelation method was also examined but the algorithm did not give consistent results. The calculated pitch was monotone except for a disconcerting waver.

The number of poles in the analysis was also varied. Ten poles marked a qualitative boundary between clear speech and muffled speech. Using too few poles gave vocoded speech which was severely muffled. With such a small number of poles the filter does not have the resolution required to

describe the complete formant structure of the modeled vocal tract (see figures 4-1 to 4-5). With less than six poles, the speech becomes unintelligible. With more than ten poles, the quality of the resultant speech increases with the addition of poles, with maximum quality reached at about sixteen poles. With sixteen or more poles, the quality remains approximately the same.

Noise

The noise was introduced to each utterance by adding a random noise signal to the file containing the utterance. This was accomplished with the same random noise generating subroutine which is used to excite the vocoder during unvoiced speech. A separate program performs this noise addition. The maximum value of the noise can be varied to provide noise levels from zero to considerably in excess of the speech power. Because speech is not a stationary process, a true signal to noise level is difficult to calculate. It varies widely if the noise level is constant because the speech energy varies widely from low energy in the fricatives (s,sh,f,th...) to much higher energy in the voiced sounds such as vowels. For our concerns, a signal to noise ratio (SNR) was determined by calculating the power of the noise and comparing it to the power of the voiced portion of a clean file. Although the true SNR may vary from frame to frame and from speech file to speech file, consistent results are possible. That is, equivalent

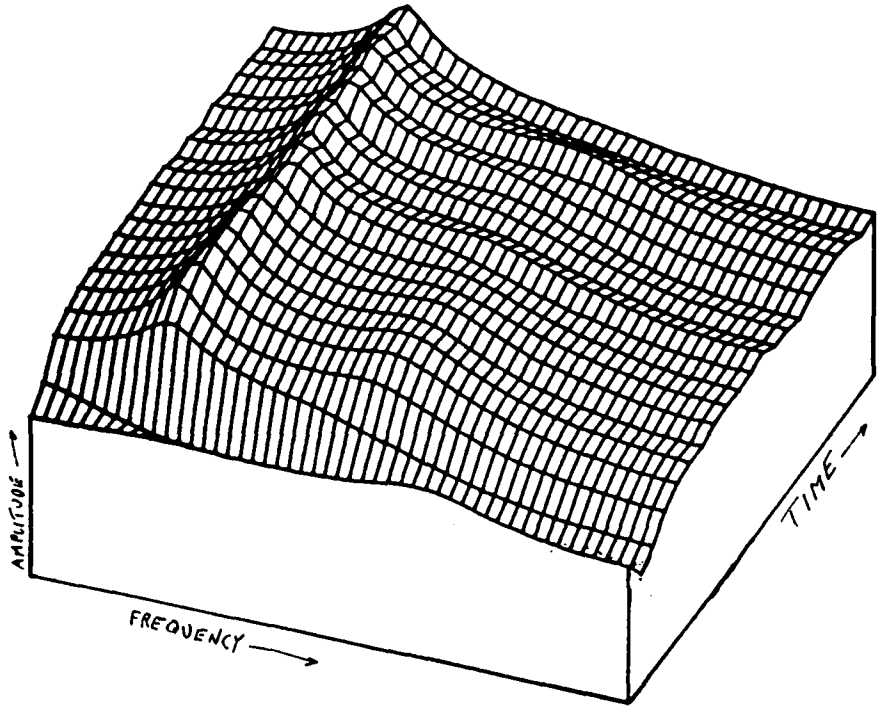


Figure 4-1 Formant Trajectory of the utterance "five." Four poles used in the analysis.

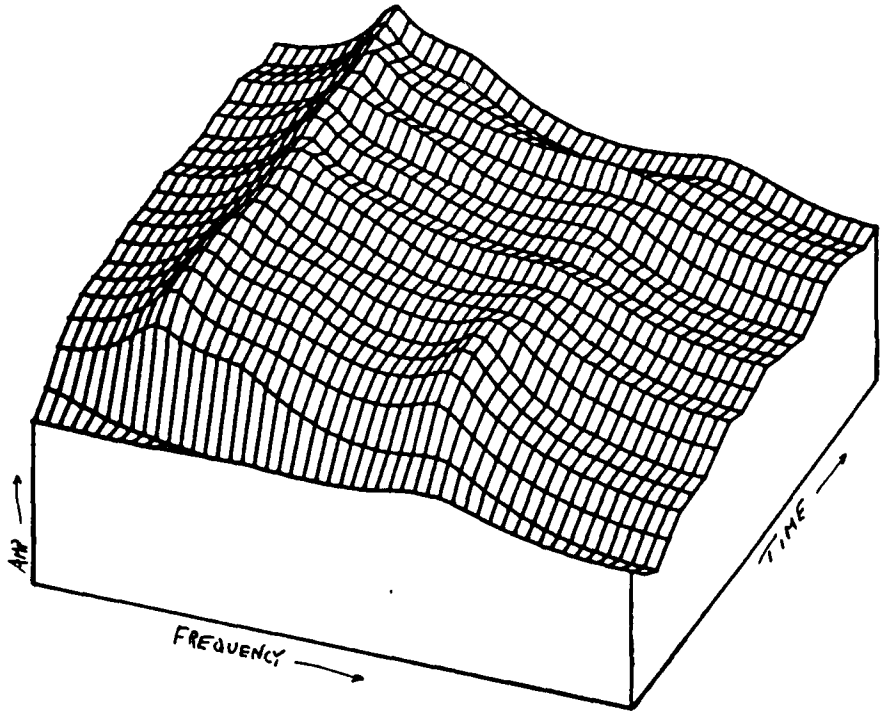


Figure 4-2 Formant Trajectory of the utterance "five." Six poles used in the analysis.

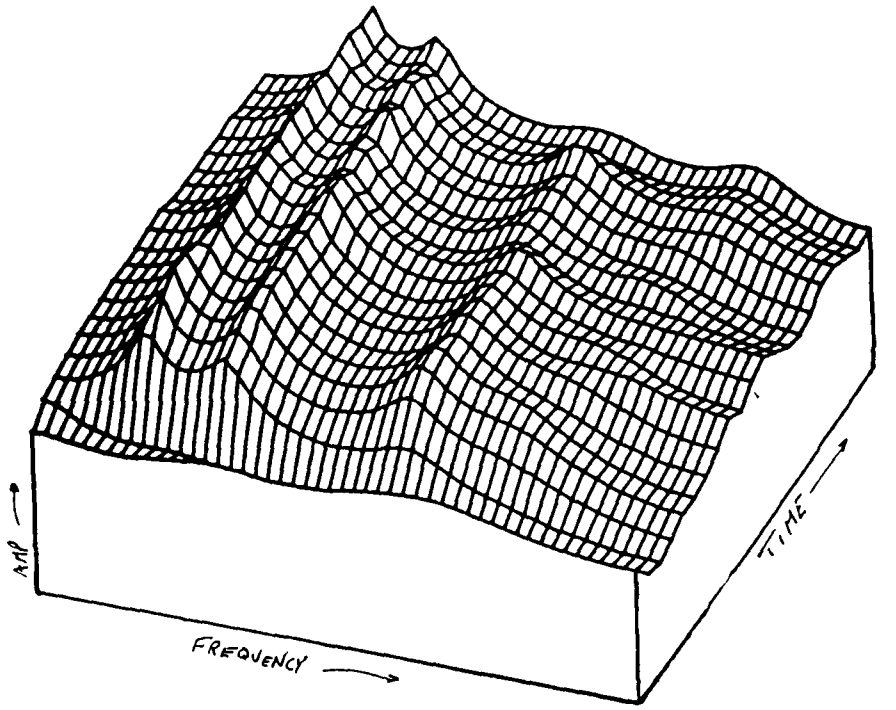


Figure 4-3 Formant Trajectory of the utterance "five." Eight poles used in the analysis.

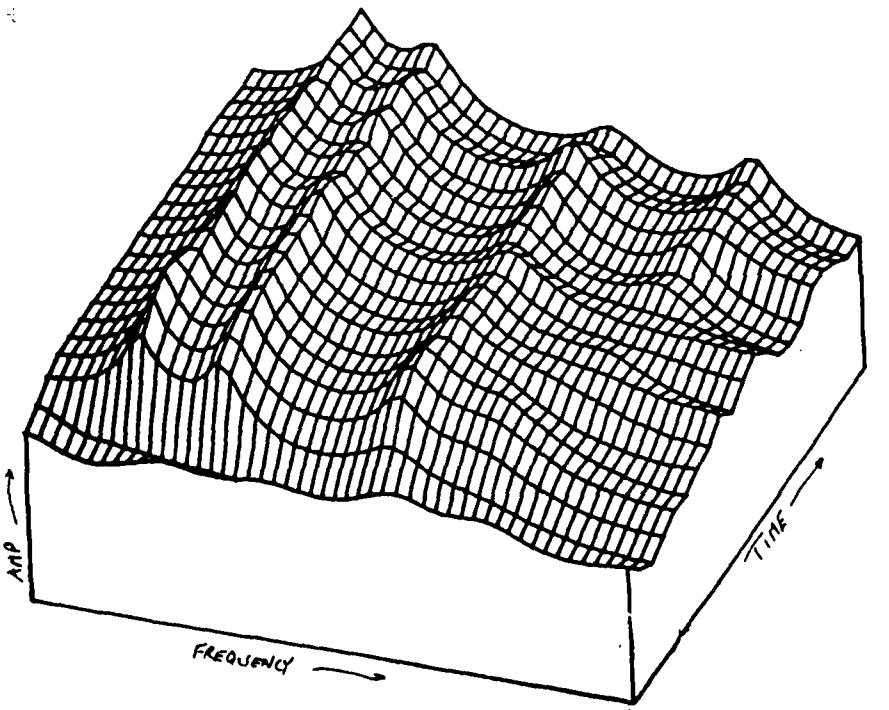


Figure 4-4 Formant Trajectory of the utterance "five." Ten poles used in the analysis.

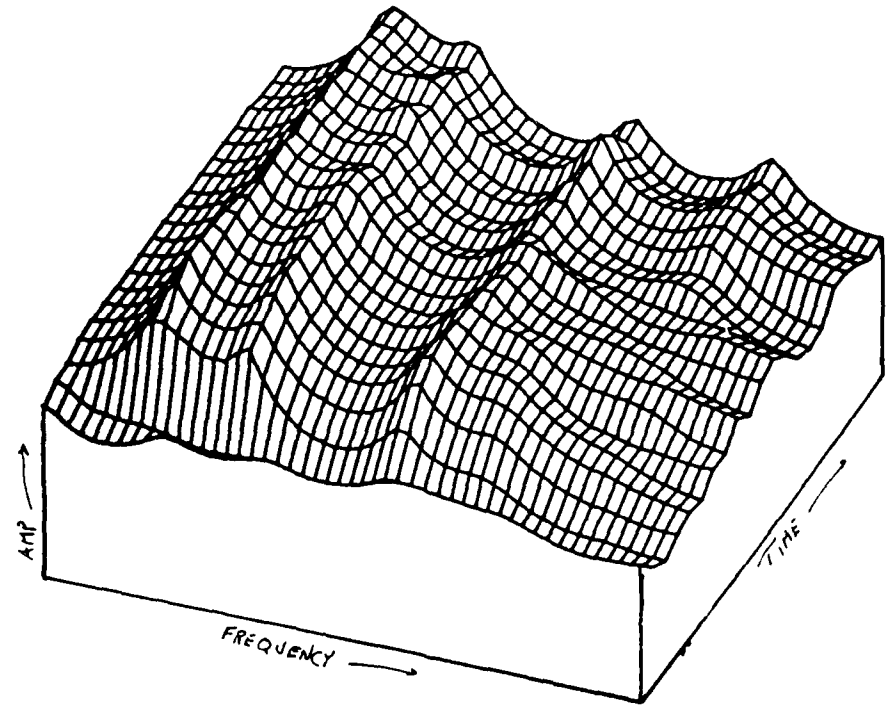


Figure 4-5 Formant Trajectory of the utterance "five." Twelve poles used in the analysis.

values of added noise gave approximately the same SNR to all cases (see table 4-2) and the synthesized speech suffered the same problems of unintelligibilty.

Table 4-2

Noise

Maximum Value of Added Noise	Noise Power in the Frame (dB)	Signal to Noise Ratio-SNR (dB)
0	58	20
100	61	16
200	64	14
500	70	8
1000	73	4

To test the effects of noise, speech files with different signal to noise ratios were tested. Degradation occured even with fairly small amount of added noise (SNR = 16dB). Severe degradation of the re-synthesized speech occured with a SNR of about 8dB for the noisy input signal. At this level, the output was unintelligible.

The analyzer consists of two parts, a pitch detector and a prediction coefficient generating routine. These two parts were examined to determine which is the more sensitive to noise corruption. This was accomplished by processing two versions of the same utterance, one which was an unaltered version of the original speech file, and one which was the original file plus an amount of random noise added. These files were processed separately wihin the same program, one for pitch and the other for the prediction coefficients. Preliminary results indicate that the

predictor coefficients are more susceptible to noise than is the pitch detection.

With this processing scheme, four permutations of noisy and clean files are possible. Two permutations use the same file for pitch detection and coefficient generation. One performs pitch detection on a clean file and coefficient generation on a noisy file, and one performs pitch detection on a noisy file and coefficient generaton on a clean file.

The clean/clean test was used as a control example and the other permutations were examined for a qualitative analysis of intelligibility. The signal to noise ratio was maintained constant over all of the files. Noisy speech files with a SNR of 8dB gave unintelligible results but were identifiable as speech, so this level was used as the noisy file. The mixed analysis (noisy/clean, clean/noisy) gave interesting results. The noisy coefficient/clean pitch gave highly degraded speech which was only slightly better than the noisy/nolisy analysis. Contrasting with this, the clean coefficient/noisy pitch gave only slightly degraded results. Three of the five tests were noticably degraded, yet still intelligible. Two were virtually indistiguishable from the clean/clean example. It was expected that the noise would degrade the pitch detection severely and render any reproducton unintelligible. In the five tests, this was not the case: the predictor coefficient generating algorithm failed. This can be heard in the output speech and seen in a plot of the frequency response of the ensemble of the

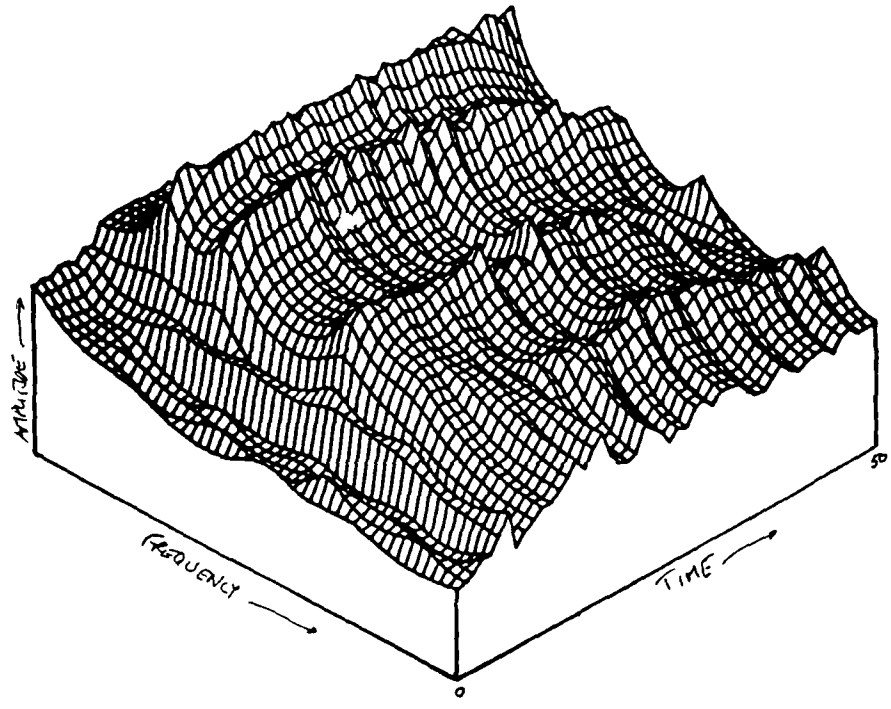


Figure 4-6 Formant Trajectory of the utterance "five." Ten poles used in the analysis. Frames 1-50 shown.

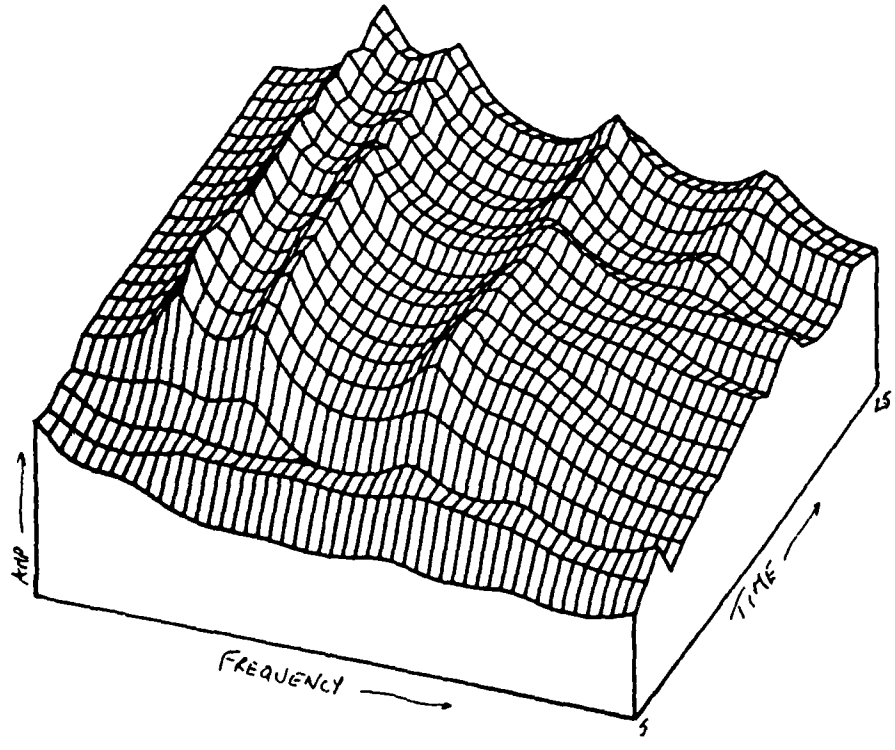


Figure 4-7 Formant Trajectory of the utterance "five." No added noise. Frames 2-25 shown. (SNR = 20 dB).

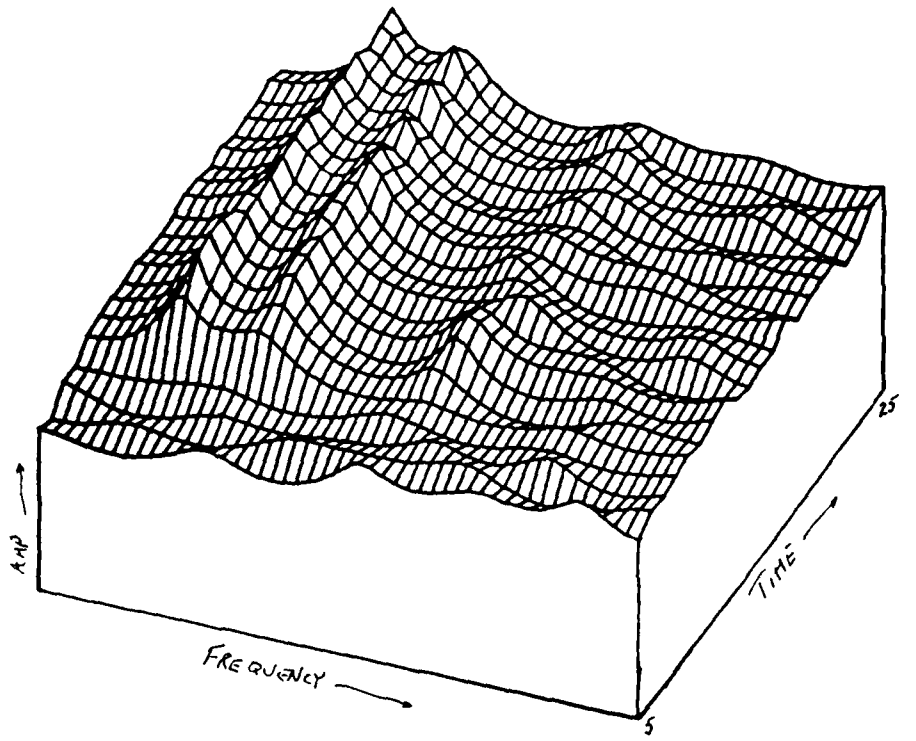


Figure 4-8 Formant Trajectory of the utterance "five" with noise. Added noise gives SNR of 16 dB.

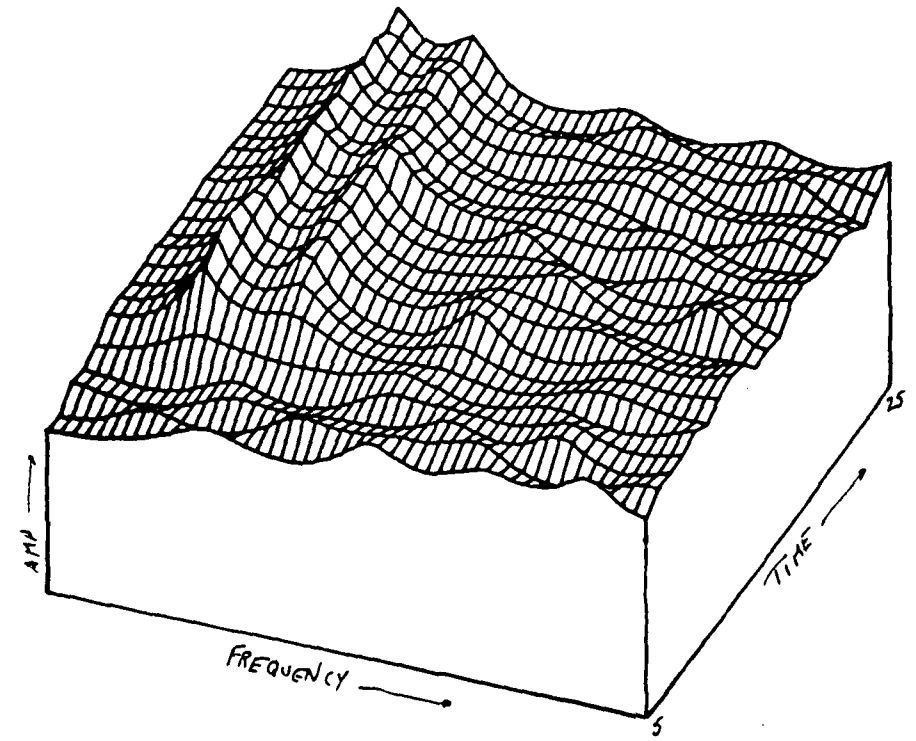


Figure 4-9 Formant Trajectory of the utterance "five" with noise. Added noise gives SNR of 14 dB.

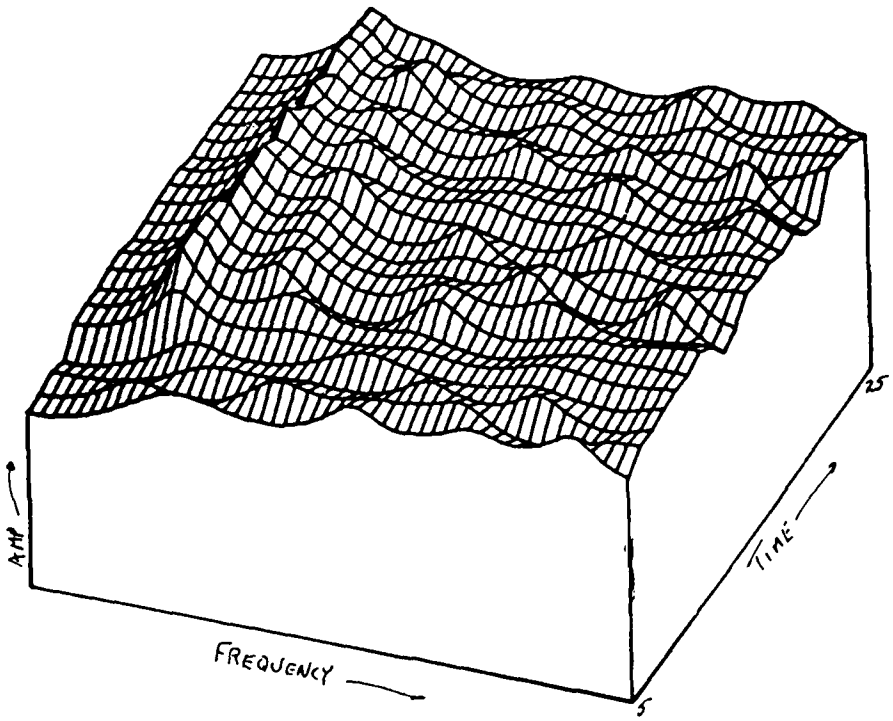


Figure 4-10 Formant Trajectory of the utterance "five" wit' noise. Added noise gives SNR of 8 dB.

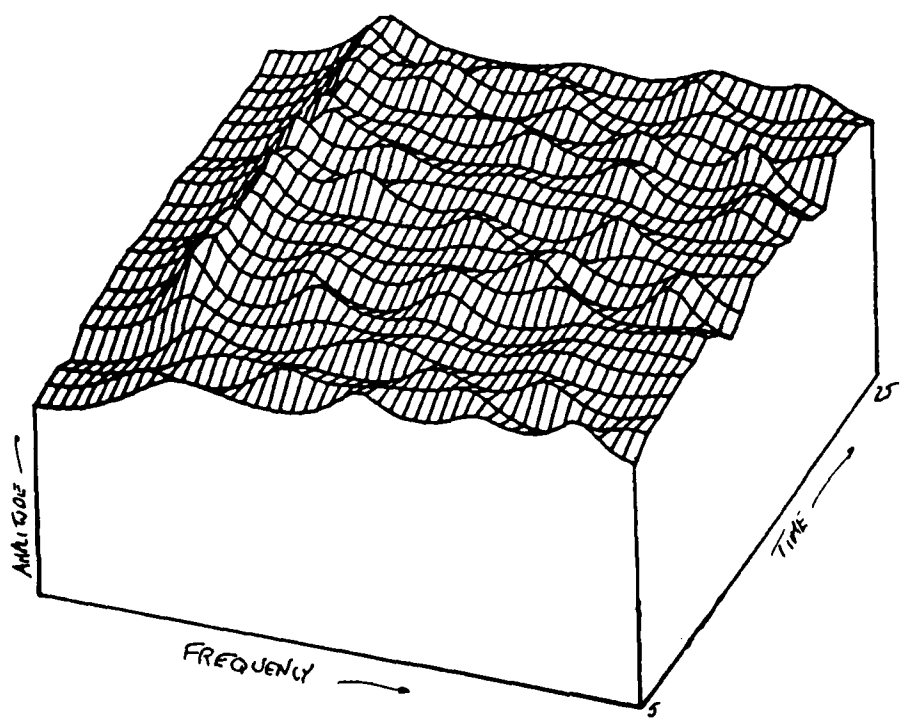


Figure 4-11 Formant Trajectory of the utterance "five" with noise. Added noise gives SNR of 4 dB.

digital filters (figures 4-6 to 4-11). The formant structure is lost in the noise; only the first formant can be located and identified on these plots.

To further examine this phenomenon, the glottal pulse excitation of the outp . filter was replaced by the random noise excitation. The results in this case sounded like a whispered utterance, but were still intelligible. In fact, preliminary results indicate that in a high noise environment, intelligibility can be gained by neglecting pitch information at the output and generate the utterance with only only a random excitation to the output filter. The waveshape of the synthesized speech was examined to possibly locate the source of the unintelligibilty. It was determined that the voiced frames in the synthesized speech are the main cause of the squeals and buzzes which make the speech unintelligible.

CHAPTER V

Conclusions and Recommendations

Conclusions

This report describes a system which processes speech using linear predictive methods. The system is a software simulation of an LPC analyzer and synthesizer. The system consists of two programs, one of which processes the speech to generate the LPC parameters, and another which processes these parameters to resynthesize the speech. An important aspect of the system is that it enables the user to select from various pitch and coefficient analysis methods. It also allows the user to vary other parameters in order to simulate other changes in the processing scheme.

To test the operation of the system, a regimen of testing was performed by varying the different parameters. A separate program allows a simple method for changing all of the parameters over which the user has control. These parameters are called the decision variables and each has an allowable range of values. The system operated satisfactorally over all values of the decision variables. The flexibilty exhibited by the system in this testing indicates that the system can be a valuable tool for the study of linear predictive coding of speech in the Signal Processing Lab at AFIT.

Some of the parameters which were tested extensively

were the number of poles in the analysis, the different methods of analysis and pitch detection. It was determined that ten poles give a reasonable representation af speech. The covariance method of detection exceeded the autocorrelation method with respect to quality of output speech. The SIFT pitch detection routine far exceeded the AUTOC method in determining pitch.

Also examined were some of the noise problems of LPC. Various noise levels were tested to determine at which level noise corruption rendered the LPC system useless. This level was found to be at a signal to noise ratio of about 8dB. Another important result was that the coefficient generation was greatly affected by noise. The effect of the predictor coefficients was much greater than the effect of the pitch detection. This result may be useful in exploring techniques to counter the effects of noise corruption.

Recommendations

The linear predictive coding system presented in this thesis can be used as a firm foundation for more study in the process of linear predictive coding of speech. Continuing effort with this project could extend in two general directions. One direction would be software oriented with further work being done to expand the system with more subroutines. The other general direction is oriented to studying more about LPC using the system as a tool.

Further Work in Software Development

Part of the flexibilty of the system stems from the extensive use of subroutines. Additional subroutines could be incorporated into the system to expand the present capabilities of the LPC anlyzer. Perhaps the first task to be attempted would be to incorporate the recursive LPC method as developed by Capt Willis Janssen [Ref 4] into this system. This would offer an opportunity to compare this method with some of the more common techniques which have been implemented in the current system. A lattice formulation of the predictor coefficients would offer another method of analysis. This method is decribed in the book by Rabiner and Schafer [Ref 10]. An undebugged version of a possible subroutine implementation is presented in Appendix D.

Other useful additions to the present system would be additional pitch detection methods. The AUTOC method might be altered slightly to give better results. The current literature describes other methods of pitch detection. The system would be greatly enhanced if it offered the availablity of more methods with which to analyze the input speech signal.

A process for simulating the bit rates actually transmited over the channel would be a useful addition. At present, all quanization is done in 2 byte (16 bit) segments. The coefficients sent over the channel are 4 byte

floating point words. The flags sent over the channel require at most 2 bits each, but each is quantized as 16 bit integers. The pitch is likewise represented as a 16 bit integer. It could easily be represented with fewer bits. The frame size information sent over the channel is redundant and could be eliminated. All of these compressions could reduce the effective bit rate of the communicated signal.

Also needed is a better interface with the audio input and output. The present means to listen to processed speech is to move the file containing the speech to another directory (on a different system even) and invoke another program. This method is time consuming and reduces the effectiveness of a synthesize-listen-compare atmosphere of testing.

Further Testing

Since this simulation was designed as a tool to study the process of linear predictive coding of speech, it seems only natural that considerable further testing can be imagined. A fine place to start further research is with the ubiquitous noise problem.

A possible technique for reducing the effect of noise was discovered in this work. The technique is to ignore the pitch detector information. If all speech is presumed to be unvoiced, the synthesized speech will resemble whispered speech. In noise, the greatest difference between the input

speech and the output speech occured during periods of the utterance declared to be voiced speech. I recommend this technique to be further explored.

Dr Kabrisky is interested in a method of compressing the formant frequencies while maintaining the ratio between them. This system models the human speech production system with the poles of a digital filter. These poles describe the formant locations. Therefore digital processing techniques could be used to shift the poles and consequently the formants.

One final recommendaton is to use this system as a means to study the speech recognition capabilities of LPC. Other research has shown the feasibilty of LPC for speech recognition tasks [Ref 2]. The feature vector as described by the predictor coefficients is easily extractable from this system. The flexibilty of the system offers the user to vary a wide range of parameters in search for a set which expedites recognition.

Bibliography

1. Atal, B. S. and Remde, J. R. "A Model of LPC Excitation for Producing Natural-Sounding Speech at Low Bit Rates," Proc. IEEE Conf. on Accoustics, Speech, and Signal Processing , pp. 614-617, 1982

2. Doddington, G. R. and Schalk, T. B. "Speech Recognition: Turning Theory to Practice," IEEE Spectrum , pp. 26-32, Sep 1981

3. Hunter, C. J. Time Axis Analysis of Gravity Distorted Speech . MS Thesis GE/EE/81D-27. Wright Patterson AFB, Ohio: School of Engineering, Air Force Institute of Technology, Dec 1981.

4. Janssen W. A. A Recursive Linear Predictive Vocoder . MS Thesis GE/EE/83D-33. Wright Patterson AFB, Ohio: School of Engineering, Air Force Institute of Technology, Dec 1983.

5. Kelton, W. D. and Law, A. M. Simulation Modeling and Analysis , New York, McGraw-Hill, 1982

6. Kinderman, A. J. and Ramage, J. G. "Computer Generation of Normal Random Variables," Journal of American Statistical Association , vol. 71, Dec 1976

7. Makhoul, J. "Linear Prediction -- A Tutorial Review," Proc. of the IEEE , vol. 63, pp. 561-580, April 1975

8. Makhoul, J. "Stable and Efficient Lattice Methods for Linear Prediction," IEEE Trans. Acoust., Speech, Signal Processing , vol. ASSP-25, pp. 423-428, Oct. 1977

9. Markel, J. D. "The SIFT Algorithm for Fundamental Frequency Estimation," IEEE Trans. Audio and Electroacoustics , vol. AU-20, no. 5, Dec. 1972

10. Markel, J. D. and Gray, A. H. Linear Prediction of Speech , New York: Springer-Verlag, 1976

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

12. Rabiner, L. R. and Schafer, R. W. "Digital Representations of Speech Signals," Proceedings of the IEEE , vol. 63, pp. 662-677, Apr 1975

13. Rosenburg, A. E. "Effects of Glottal Pulse Shape on the Quality of Natural Vowels," J. Acoust. Soc. Am. , vol. 49, pp 583-590, Feb 1971

APPENDIX A

User's Manual

USER'S MANUAL

A LINEAR PREDICTIVE CODING SYSTEM

DESIGNED AND WRITTEN BY
LT CRAIG E. MCKOWN

THIS USER'S MANUAL IS COMPOSED OF THREE PARTS, EACH CORRESPONDING TO A SEPERATE PROGRAM WHICH IS REQUIRED TO OPERATE THE COMPLETE SYSTEM. THE FIRST PART DESCRIBES THE USE OF THE PROGRAM SETUP, WHICH IS USED TO CREATE THE DECISION VARIABLE FILES. THE SECOND PART DESCRIBES THE USE OF THE PROGRAM PREDICT, WHICH IS THE LPC ANALYZER. THE LAST PART DESRIBES THE USE OF THE PROGRAM VOCODE, WHICH SYNTHESIZES THE VOCODED SPEECH.

BEFORE PREDICT OR VOCODE CAN BE RUN, A DECISION VARIABLE FILE MUST EXIST. TO CREATE A NEW DECISION VARIABLE FILE OR TO UPDATE AN OLD ONE, THE PROGRAM SETUP MUST BE USED. THE OUTPUT OF PREDICT IS THE INPUT TO VOCODE, SO PREDICT MUST BE RUN BEFORE VOCODE.

THE SPEECH INPUT TO PREDICT MUST BE IN A CONTIGUOUS FILE, IN INTEGER FORM. PREDICT AND VOCODE HAVE NO LIMITATON ON THE LENGTH OF THE SPEECH FILE, BUT THE AUDIO INTERFACE DOES. IT LIMITED TO 88 BLOCKS (2.8 SECONDS). THEREFORE IT IS RECOMMENDED THAT THE PROCESSED SPEECH BE LIMITED TO 88 BLOCKS.

PROGRAM SETUP

FILE: SETUP.FR
DIRECTORY: DP4:KOWN
LANGUAGE: FORTRAN5
DATE: SEP 83
AUTHOR: W. JANSSEN / REVISED BY CRAIG MCKOWN
SUBJECT: CREATES FILE OF DECISION VARIABLES NEEDED BY THE MCKOWN LPC ANALYZER.

ARGUMENTS & VARIABLES	TYPE	PURPOSE
RELVAR	REAL ARRAY	REAL VALUED DECISION VARIABLES
INTVAR	INTEGER ARRAY	INTEGER VALUED DECISION VARIABLES
SIZER	INTEGER	NUMBER OF ELEMENTS IN RELVAR
SIZEI	INTEGER	NUMBER OF ELEMENTS IN INTVAR
OUTFILE	STRING	NAME OF DECISION VARIABLE FILE

FUNCTION:

THIS PROGRAM CREATES A FILE CONTAINING THE DECISION VARIABLES (DV) REQUIRED BY THE LPC ANALYZER DESIGNED BY C. MCKOWN. IT CAN CREATE A NEW FILE OR OVERWRITE AN OLD FILE THE PROGRAM WILL PROMPT THE USER FOR ALL NECESSARY INPUTS. THE CURRENT VALUE OF EACH DV WILL BE SHOWN AND THE USER WILL BE GIVEN AN OPTION OF CHANGING EACH ONE. THE PROGRAM WILL ALSO PRINT OUT THE DECISION VARIABLES IN A READABLE FORMAT TO THE SCREEN OR THE PRINTER OR BOTH, AS DESIRED BY THE USER.

PROGRAM USE:

THE PROGRAM IS LOADED BY THE FOLLOWING COMMAND:

```
RLDR SETUP @FLIB@
```

RUN THE PROGRAM--
"SETUP"
THE FIRST PROMPT WILL ASK IF YOU ARE UPDATING AN OLD FILE. ANSWER YES ("1") IF DV FILE CREATED PREVIOUSLY. THE NEXT PROMPT WILL ASK FOR THE FILE NAME; RESPOND WITH "FILENAME" OF THE FILE YOU WISH TO PREPARE. THE OLD FILE WILL BE OVER-WRITTEN BY ANY CHANGES MADE. THE REST OF THE PROGRAM IS EXPLAINED BY THE PROMPTS.

SEE USER'S MANUAL FOR PROGRAM "PREDICT" FOR A LIST OF THE NAMES OF THE VARIALBES TO BE CHANGED OR SET.

SUBROUTINES REQUIRED:

NONE

CHANGES:

ADDING NEW DECISION VARIABLES IS NOT DIFFICULT. ADDITIONAL SPACE REMAINS IN EACH DV ARRAY FOR AT LEAST FIVE MORE VARIABLES. THE PROGRAM MUST BE UP-DATED IN FOUR PLACES FOR EACH ADDITIONAL VARIABLE.

1) IN THE *** UPDATE ARRAYS *** SECTION. FOLLOW THE FORMAT OF THE OTHER VARIABLE UPDATES. YOU MUST CHANGE THE LINE NUMBER AFTER THE "IF()GOTO" IN THE UPDATE PRECEDING THE ADDITIONAL UPDATE.

2) IN THE *** TYPE ARRAYS *** SECTION. FOLLOW THE FORMAT OF THE OTHER TYPE STATEMENTS.

3&4) IN THE *** OUTPUT FILE *** SECTION. A NEW WRITE STATEMENT AND FORMAT STATEMENT MUST BE ADDED FOR EACH NEW VARIABLE FOLLOW THE FORMAT OF THE OTHER WRITE AND FORMAT STATEMENTS.

EXAMPLE:

```
SETUP
THIS PROGRAM CREATES OR UPDATES A DECISION VARIABLE FILE.
ARE YOU UPDATING AN OLD FILE?
(1-YES,0-NO)1
FILE NAME? DECVARR

    IF YOU CHOOSE TO CHANGE A VARIABLE ENTER : Y
    OTHERWISE ENTER ANOTHER LETTER


CURRENT VALUE OF ACCEPT/NOT ACCEPT (A-0,NA-1):          1
CHANGE VALUE?

CURRENT NUMBER OF POLES IS :         10
CHANGE VALUE?

THE METHOD OF PREDICTION IS
(AUTO-0,COVAR-1):           0
CHANGE VALUE?

CURRENT VALUE:NO. OF POINTS/SET (MAXFR):       200
CHANGE VALUE?
Y
 INPUT NEW VALUE: 160

THE CURRENT VALUE OF FILTER SPACINGS IS (MAXPT):       100
CHANGE VALUE?
Y
 INPUT NEW VALUE: 80

THE CURRENT VALUE OF PRE/DE-EMP (1-Y,0-N) IS:          1
CHANGE VALUE?

THE CURRENT VALUE OF GLOTTAL SHAPE IS
(1-POLYNOMIAL,3-IMPULSE)    :          3
CHANGE VALUE?

THE CURRENT VALUE OF HAMMING WINDOW (0-NO,1-YES):          1
CHANGE VALUE?

THE METHOD OF PITCH DETECTION IS
(SIFT-0,AUTOC-1):           0
CHANGE VALUE?

PITCH DET'N AND COEF. CAL'N FROM SAME FILE?
CURRENT VALUE (1-Y,0-N):          1
CHANGE VALUE?
```

```
THE CURRENT VALUE OF VOICED/UN THRESH IS:        .400000
CHANGE VALUE?
Y
 INPUT NEW REAL VALUE: .35

CURRENT VALUE OF SPEECH SCALE-(IN CODER):       1.00000
CHANGE VALUE?

CURRENT VALUE OF SILENCE THRESH-(IN ENER)IS:       350.000
CHANGE VALUE?

CURRENT VALUE OF UNVOICED GAIN FACTOR IS:        .100000
CHANGE VALUE?


THE ARRAYS HAVE BEEN LOADED
DO YOU WANT TO HAVE THE ARRAY TYPED(1-YES,0-NO): 1
 ACCEPT/NOT ACCEPT:                                1
 NUMBER OF POLES:                                 10
 METHOD (0-AUTO,1-COVAR,):                         0
 MAXFR:                                          160
 MAXPT:                                           80
 PRE/DE-EMP (1-Y,0-N):                             1
 GLOT (1-POLYNOMIAL,3-IMPULSE):                    3
 HAMMING WINDOW? (1-Y,0-N):                        1
 METHOD PITCH DET (0-SIFT,1-AUTOC):                0
 PITCH & COEF'S SAME FILE(1-Y,0-N):                1
VOICED/UN THRESHOLD:                          .350000
SPEECH SCALE-(IN CODER):                      1.00000
SILENCE THRESHOLD                             350.000
UNVOICED GAIN FACTOR                          .100000
WRITE DECISION VARIABLES TO SAME FILE?
(1-YES,0-NO): 1
PRINT ARRAY ON PRINTRONICS?(1-Y,0-N) 1
PROGRAM COMPLETED

STOP
R
```

DECVARR

DATE : 2/12/83

TIME : 13:56:47

ACCEPT/NOT ACCEPT	1
NUMBER OF POLES	10
METHOD (0-AUTO, 1-COVAR)	0
MAXFR	160
MAXPT	80
PRE/DE-EMP? (1-Y, 0-N)	1
GLOTTAL PULSE (1-POLY , 3-IMPULSE)	3
HAMMING WINDOW? (1-Y, 0-N)	1
METHOD PITCH DET (0-SIFT, 1-AUTOC)	0
PITCH & COEF's F'M SAME FILE(1-Y, 0-N)	1
VOICED/UNVOICED THRESHOLD	.35000
SPEECH SCALE	1.00000
SILENCE THRESHOLD	350.00000
UNVOICED GAIN FACTOR	.10000

PROGRAM PREDICT

FILE: PREDICT.FR
DIRECTORY: DP4:KOWN
LANGUAGE: FORTRAN5
DATE: SEP 83
AUTHOR: CRAIG MCKOWN
SUBJECT: DIGITAL PROCESSING OF SPEECH--
LINEAR PREDICTION ANALYZER

ARGUMENTS & VARIABLES	TYPE	PURPOSE
MAXPT	INTEGER	SAMPLES BETWEEN PITCH DETECTION
MAXFR	INTEGER	SAMPLES IN ANALYSIS WINDOW
NSET	INTEGER	COUNTER FOR PITCH FRAME NUMBER
NFRAME	INTEGER	COUNTER FOR LPC FRAME NUMBER
NPTS	INTEGER	NUMBER OF SAMPLE POINTS ANALYZED
K, S, KS, JS, JK	INTEGERS	COUNTERS (USED FOR BOOKKEEPING)
P1	INTEGER	COUNTER FOR NUMBER OF SAMPLES TO START OF NEXT LPC ANALYSIS WINDOW
JUMP	INTEGER	FLAG INDICATING NATURE OF PREVIOUS FRAME OF SPEECH (VOICED, UNVOICED OR SILENCE)
SPEEFL1	STRING	NAME OF SPEECH FILE (FOR LPC)
SPEEFL2	STRING	NAME OF SPEECH FILE (FOR PITCH)
DUMMY	STRING	NAME OF FILE HOLDING DECISION VARIABLES
PARAM	STRING	NAME OF FILE TO WHICH LPC DATA IS WRITTEN (ACTS AS TRANSMISSION CHANNEL)
DECISION VARIABLES		
POLES	INTEGER	NUMBER OF POLES IN THE OUTPUT FILTER
MP	INTEGER	METHOD OF PREDICTION (0-AUTOCOR', 1-COVARIANCE)
NGLT	INTEGER	GLOTTAL PULSE SHAPE (1-POLYNOM', 2-TRIGON', 3-IMPULSE)
MPCH	INTEGER	METHOD OF PITCH DETECTION (0-SIFT, 1-AUTOC)
NPCS	INTEGER	PITCH/LPC FILES THE SAME (0-NO, 1-YES)
NEMP	INTEGER	PRE/DE-EMPHASIS (0-NO, 1-YES)
H1	INTEGER	HAMMING WINDOW (0-NO, 1-YES)
STHR	REAL	VOICED/UNVOICED THRESHOLD (USED FOR PITCH DETECTION)
SCAF	REAL	SCALE FACTOR (INPUT SPEECH DIVIDED BY THIS TO AVOID OVERFLOW)
THRESH	REAL	SPEECH/SILENCE THRESHOLD
UNGA	REAL	UNVOICED GAIN FACTOR (OUTPUT; UNVOICED INPUT TO OUTPUT FILTER MULTIPLIED BY THIS TO PREV

VARIABLES (CONT.)

VAL	INT ARRAY	DUMMY ARRAY TO HOLD THE SAMPLED SPEECH BEFORE IT IS WRITTEN TO SPEE & SPCH
SPEE	REAL ARRAY	ARRAY HOLDING DATA FOR LPC COEFFICIENT GENERATION
SPCH	REAL ARRAY	ARRAY HOLDING DATA FOR PITCH DETECTION
AR	REAL ARRAY	ARRAY HOLDING THE LPC COEFFICIENTS AR(1)=A0, AR(2)=A1...AR(POLES)=AP. ONLY AR(2) TO AR(POLES) ARE WRITTEN TO THE CHANNEL FILE
RCOF	REAL ARRAY	REFLECTION COEFFICIENTS
AENRG	REAL ARRAY	ENERGY FROM ENERGY DETECTOR
PITCH	REAL ARRAY	PITCH FROM PITCH DETECTOR
PIT	INTEGER	INTERPOLATED PITCH (WRITTEN TO CHANNEL)
VOCD	INTEGER	FLAG INDICATING NATURE OF SPEECH (WRITTEN TO CHANNEL)
AL	REAL	ALPHA, ERROR COEFFICIENT, USED AS GAIN FOR OUTPUT CHANNEL. COMPUTED IN COEFFICIENT GENERATING ROUTINES.

FUNCTION:

THIS PROGRAM EMULATES THE ANALYSIS OF A LINEAR PREDICTIVE CODING SCHEME. IT INPUTS SAMPLED SPEECH DATA AND PRODUCES THE PARAMETERS REQUIRED BY A VOCODER TO REPRODUCE THE SPEECH. THESE PARAMETERS ARE WRITTEN TO A FILE WHICH ACTS AS THE COMMUNICATION CHANNEL. FOR MORE INFORMATION, SEE MCKOWN THESIS.
THE FORM OF THE CHANNEL FILE IS COMPATIBLE TO THE VOCODER PROGRAM BY THE SAME AUTHOR.

PROGRAM USE:

THE PROGRAM IS LOADED WITH THE FOLLOWING COMMAND:

```
RLDR PREDICT IOF SIFTB ENER DCOVAR DIRECT DAUTO AUTOC @FLIB@
```

BEFORE RUNNING THIS PROGRAM, IT IS ADVISED THAT THE USER CREATE (OR UPDATE) A FILE CONTAINING THE DECISION VARIABLES REQUIRED TO PROPERLY EXECUTE THIS PROGRAM. THIS IS EASILY ACCOMPLISHED BY USING THE PROGRAM "SETUP." SEE USER'S MANUAL FOR THE PROGRAM "SETUP."
I RECOMMEND THAT A MACRO FILE IS EMPLOYED TO RUN THIS PROGRAM AND THE VOCODER PROGRAM USED TO SYNTHESIZE THE SPEECH. THE MACRO FILE SHOULD BE OF THE FORM:

```
PREDICT SPEECHFILE1/C SPEECHFILE2/P DECVAR/I CHANNELFILE/O
```

THE FILE SPEECHFILE1 IS THE NAME OF THE INPUT SPEECH FILE USED FOR THE PREDICTOR COEFFICIENT GENERATION. THE FILE SPEECHFILE2 IS THE NAME OF THE INPUT SPEECH FILE USED TO ACCOMPLISH THE PITCH DETECTION. THE FILE DECVAR IS THE NAME

OF THE FILE WHICH CONTAINS THE DECISION VARIABLES.
THE NAME CHANNELFILE IS THE NAME OF THE FILE TO WHICH THE LPC PARAMETERS ARE WRITTEN. IT MUST HAVE THE SAME NAME AS THAT WHICH IS USED FOR VOCODE.

SUBROUTINES REQUIRED:

NAME:	LOCATION:	PURPOSE:
IOF	DP4:KOWN	READS RUN MACRO FILE
SIFTB	''	PITCH DETECTION
AUTOC	''	PITCH DETECTION
ENER	''	ENERGY DETECTION
DAUTO	''	LPC COEF GENERATION
DCOVAR	''	LPC COEF GENERATION
DIRECT	''	DIRECT FORM FILTER

NOTE:
SOME INFORMATION IS WRITTEN TO THE SCREEN TO ASSURE THE USER THAT THE PROGRAM IS INDEED RUNNING. THE RUN TIME OF THE PROGRAM IS ABOUT FOUR MINUTES, AND IS DEPENDENT UPON THE METHODS OF PITCH DETECTION AND COEFFICIENT GENERATION, AS WELL AS THE NUMBER OF POLES USED FOR ANALYSIS.

SEE USER'S MANUAL FOR "VOCODE"

PROGRAM VOCODE

FILE: VOCODE.FR
DIRECTORY: DP4:KOWN
LANGUAGE: FORTRAN 5
DATE: SEP 83
AUTHOR: CRAIG MCKOWN
SUBJECT: DIGITAL PROCESSING OF SPEECH
LINEAR PREDICTION VOCODER

ARGUMENTS & VARIABLES	TYPE	PURPOSE
SPEEFL	STRING	NAME OF A DUMMY FILE (NOT USED)
PARAM	STRING	NAME OF FILE FROM WHICH LPC DATA IS READ (ACTS AS TRANSMISSION CHANNEL)
DUMMY	STRING	NAME OF A DUMMY FILE (NOT USED)
RUMMY	STRING	NAME OF THE FILE TO WHICH DIGITIZED SPEECH IS WRITTEN (OUTPUT FILE) SPEECH HAS BEEN NORMALIZED FOR AUDIO OUTPUT WITH "AUDIOHIST"
AR	REAL ARRAY	LPC COEFFICIENTS READ FROM THE CHANNEL FILE. AR(1) IS THE AR(2) FROM THE CODER PROGRAM "PREDICT"
POLES	INTEGER	NUMBER OF POLES OF THE OUTPUT FILTER
VOCD	INTEGER	FLAG INDICATING UNVOCD/VOICED DECISION FROM CODER
PIT	INTEGER	PITCH PERIOD IN SAMPLES READ FROM CHANNEL FILE
IX	DP INT	SEED NUMBER FOR SUBROUTINE "UNVOCD"
U	REAL ARRAY	OUTPUT OF "VOICED" OR "UNVOCD" - INPUT TO "THROAT"
W	REAL ARRAY	MEMORY FOR "THROAT"
S	REAL ARRAY	OUTPUT OF THROAT - VOCODED SPEECH
INTS	INT ARRAY	INTEGER VALUES OF S - WRITTEN IN BLOCK FORM TO OUTPUT FILE
X	INT ARRAY	USED FOR WRBLK AND RDBLK
IS, IP, KS	INTEGERS	COUNTERS

FUNCTION:

THIS PROGRAM EMULATES THE VOCODER OF A LINEAR PREDICTIVE CODING SCHEME. IT TAKES AS INPUTS THE LPC PARAMETERS FROM A FILE WHICH ACTS AS THE COMMUNICATION CHANNEL, AND USES THESE TO REPRODUCE DIGITAL SPEECH. FOR MORE INFORMATION SEE THE MCKOWN THESIS.
THIS PROGRAM WAS WRITTEN TO BE USED IN CONJUNCTION WITH THE LPC CODER PROGRAM, "PREDICT" BY THE SAME AUTHOR.

PROGRAM USE:

THE PROGRAM IS LOADED BY THE FOLLOWING COMMAND:

```
RLDR VOCODE IOF UNVOCD DRAND THROAT GLOT3 GLOT2 GLOT1 @FLIB@
```

THE PROGRAM "PREDICT" MUST BE EXECUTED BEFORE THIS PROGRAM CAN BE USED; THE OUTPUT OF THAT PROGRAM IS USED AS THE INPUT FOR THIS PROGRAM. IT IS RECOMMENDED THAT A MACRO FILE BE USED TO RUN THIS PROGRAM. A SUGGESTED FORMAT IS:

```
VOCODE DUMMY/X DUMMY/Y CHANNELFILE/I OUTPUTSPEECH/O
```

THE FILE DUMMY IS THE NAME OF A DUMMY FILE. IT IS NOT USED SO IT DOES NOT HAVE TO EXIST. THE FILE CHANNELFILE MUST BE THE SAME AS IS USED FOR THE PREDICT PROGRAM. THE NAME OUTPUTSPEECH IS FOR THE FILE WHICH CONTAINS THE VOCODED SPEECH.

THE OUTPUT FILE TO THIS PROGRAM CONTAINS A DIGITAL REPRESENTATION OF THE OUTPUT SPEECH. TO LISTEN TO THE SPEECH, THE PROGRAM "AUDIOHIST" MUST BE USED. THE OUTPUT FILE MUST BE MOVED TO A DIRECTORY CONTAINING THIS PROGRAM (E.G. DP0:SPOUT). TO LISTEN TO THE OUTPUT SPEECH, GET INTO THE DIRECTORY WHICH NOW CONTAINS THE OUTPUT SPEECH FILE AND RUN THE PROGRAM "AUDIOHIST." TO THE FIRST PROMPT TYPE THE NAME OF THE OUTPUT SPEECH FILE, TO THE SECOND PROMPT TYPE "1", AND TO THE THIRD PROMPT TYPE "2."

SUBROUTINES REQUIRED:

NAME:	LOCATION:	PURPOSE:
IOF	DP4:KOWN	READS RUN MACRO FILE
UNVOCD	''	PRODUCES NORMAL RANDOM NOISE WHICH DRIVES THE OUTPUT FILTER FOR UNVOICED SPEECH
DRAND	''	PRODUCES UNIFORMLY DISTRIBUTED NOISE WHICH IS REQUIRED BY "UNVOCD"
VOICED#		PRODUCES A GLOTTAL PULSE FOR VOICED SPEECH
ACTUAL FILE NAMES ARE:		
GLOT1	''	GLOTTAL PULSE SHAPE: POLYNOM.
GLOT2	''	GLOTTAL PULSE SHAPE: TRIGONOM.
GLOT3	''	GLOTTAL PULSE SHAPE: IMPULSE
THROAT	''	THE OUTPUT FILTER

NOTE: THE INPUT FILE TO THIS PROGRAM SHOULD BE IN A FORM COMPATIBLE WITH THE CODER PROGRAM "PREDICT" WRITTEN BY C. MCKOWN. ANY OTHER FORM WILL GIVE SPURIOUS RESULTS.

```
TYPE RUN1.MC
PREDICT DP5:FIVE/C DP5:FIVE/P DECVARR/I SYNLPC/O
VOCODE S4/X LOUT/Y SYNLPC/I WORD:Y01/O
R


RUN1
 PROGRAM PREDICT      RUNNING.
       10 POLES
       25 FRAMES PROCESSED
       50 FRAMES PROCESSED
       75 FRAMES PROCESSED
      100 FRAMES PROCESSED
      125 FRAMES PROCESSED
      150 FRAMES PROCESSED
      175 FRAMES PROCESSED
      200 FRAMES PROCESSED
      225 FRAMES PROCESSED
      250 FRAMES PROCESSED
      275 FRAMES PROCESSED
      300 FRAMES PROCESSED
      325 FRAMES PROCESSED
 NPTS =    22222 MSET =      276

STOP
 PROGRAM VOCODE       RUNNING.
 NDONE =      336
 SPEECH VOCODED
 THE MAX VALUE FOUND WAS     2531

STOP
R
```

APPENDIX B

Program Listings

```
C*************************************************************************
C
C       PROGRAM:          PREDICT
C       AUTHOR:           CRAIG MCKOWN
C       DATE:             SEP 83
C       LANGUAGE:         FORTRAN5
C
C       FUNCTION:         THIS PROGRAM EMULATES THE INPUT TO A LINEAR
C                         PREDICTION ENCODER.  IT DETERMINES THE PITCH
C                         PERIOD OF THE INCOMING SPEECH AND THE PARAMETERS
C       OF THE OUTPUT FILTER REQUIRED TO GENERATE THE SPEECH AT THE
C       VOCODER.  THESE PARAMETERS ARE WRITTEN TO A FILE (PARAM) WHICH
C       ACTS AS THE TRANSMISSION CHANNEL.
C       THIS PROGRAM HAS BEEN WRITTEN SO THAT MANY OF THE DECISION
C       VARIABLES CAN BE SET BY THE USER.  THIS ALLOWS THE USER TO VARY
C       THE METHODS OF PREDICTION OR OTHER PARAMETERS WHICH MAY AFFECT
C       THE QUALITY OF THE VOCODED SPEECH.
C
C       LOAD LINE:        RLDR PREDICT IOF ENER SIFTB DIRECT DCOVAR DAUTO
C                         AUTOC @FLIB@
C
C*************************************************************************
C
C    VARIABLES, PARAMETERS, AND ARGUMENTS
C
C       MAXPT:   SAMPLES BETWEEN PITCH DETECTION
C       MAXFR:   SAMPLES IN ANALYSIS WINDOW (PITCH & COEFS)
C       NFRAME:  KEEPS TRACK OF THE LPC FRAME NUMBER
C       NSET:    KEEPS TRACK OF THE PITCH FRAME NUMBER
C       K,S,KS,JS,JK     :COUNTERS
C       NPTS:    NUMBER OF POINTS ANALYZED
C       SPEEFL1:  NAME OF SPEECH FILE (FOR LPC)
C       SPEEFL2:  NAME OF SPEECH FILE (FOR PITCH)
C       PARAM:   FILE TO WHICH LPC DATA IS WRITTEN
C                (ACTS AS THE TRANMISSION CHANNEL)
C       DUMMY:   HOLDS THE DECISION VARIABLES
C       POLES:   NUMBER OF POLES IN THE LPC ANALYSIS
C       VAL:     DUMMY ARRAY TO HOLD THE SAMPLE SPEECH FROM THE RDBLK
C                BEFORE IT IS WRITTEN TO SPCH & SPEE
C       SPCH:    ARRAY TO HOLD DATA NEEDED FOR PITCH DETECTION
C       SPEE:    ARRAY TO HOLD DATA NEEDED FOR LPC COEFFICIENT PREDICTON
C       JUMP:    FLAG DENOTING THE MOST PREVIOUS TYPE OF SPEECH (VOICED,
C                UNVOICED, OR SILENCE)
C       P1 :     NUMBER OF SAMPLES TO START OF NEXT LPC ANALYSIS WINDOW
C       AR :     LPC COEFFICIENTS :AR(1)=A0,AR(2)=A1,...,AR(N)=AP
C                ONLY AR(2) THROUGH AR(NPOLES) ARE SENT THROUGH THE CHANNEL
C       PIT:     PITCH (DELAYED BY TWO PITCH DETECTION FRAMES)
C       UNGA:    UNVOICED GAIN FACTOR
C       THRESH:  SILENCE THRESHOLD
C       SCAF:    SCALE FACTOR
C       STHR:    VOICED/UNVOCD THRESHOLD FOR THE PITCH DETECTOR.
C       NGLT:    DECISION VARIABLE IDENTIFYING THE GLOTTAL PULSE SHAPE.
C       MP:      DECISION VARIABLE IDENTIFYING THE METHOD OF PREDICTION.
C       H1:      DECISION VARIABLE FOR PRESENCE OF HAMMING WINDOW.
C       MPCH:    DECISION VARIABLE IDENTIFYING METHOD OF PITCH DETECTON.
C
C*************************************************************************

        INTEGER MAIN(7),SPEEFL1(7),PARAM(7),DUMMY(7),SPEEFL2(7),H1
        INTEGER VAL(1280),POLES,DELAY,S,F,P1,INTVAR(10),PIT,VOCD
```

```
      INTEGER NAL(1280)
      DIMENSION SPCH(400),PBUF(100),PITCH(3),AR(20),RCOF(20)
      DIMENSION SPEE(400),RELVAR(10),AENRG(3)

      DATA NPTS,NFRAME,NSET,K,JK,JEND/0,0,0,0,0,0/
      DATA S,KS,JUMP,JS/1,1,1,1/
      DATA PITCH/3*0.0/,AENRG/3*0.0/
      DATA YMEMN/0.0/
      MAXPT = 160              ;  DEFAULT VALUES
      MAXFR = 320              ;
      NFILES = 4

C*** CALL IOF AND OPEN ALL REQUIRED FILES.
      CALL IOF(NFILES,MAIN,SPEEFL1,SPEEFL2,DUMMY,PARAM,MS,S1,S2,S3,S4)

      CALL OPEN(1,SPEEFL1,1,IER)        ; SPEECH FILE (LPC)
      CALL OPEN(4,SPEEFL2,1,JER)        ; SPEECH FILE (PITCH)
      IF((IER.NE.1).OR.(JER.NE.1)) TYPE "OPEN FILE ERROR ",IER,JER

      CALL OPEN(2,DUMMY,3,JER)          ; DECISION VARIABLES
      IF(JER.NE.1) TYPE "OPEN FILE ERROR ",JER

      CALL DFILW(PARAM,JER)             ; LPC PARAMETERS
      IF (JER.EQ.13) GO TO 40
      IF (JER.NE. 1) TYPE "DELETE FILE ERROR ",JER
 40   CALL CFILW(PARAM,2,JER)
      IF (JER.NE.1) TYPE "CREATE FILE ERROR ",JER
      CALL OPEN(3,PARAM,3,JER)
      IF(JER.NE.1) TYPE "OPEN FILE ERROR ",JER

C***  GET DECISION VARIABLES
      READ (2,42) (RELVAR(I),I=1,10)
      READ (2,43) (INTVAR(I),I=1,15)
 42   FORMAT(3X,F12.5)
 43   FORMAT(3X,I10)
      IF(INTVAR(1).EQ.1) GO TO 45

      ACCEPT "NUMBER OF POLES IN THE LPC FILTER: ",POLES
      ACCEPT "METHOD OF PREDICTION: 0-AUTOCORR, 1-COVARIANCE",MP
      ACCEPT "METHOD OF PITCH DETECTION: 0-SIFT, 1-AUTOC",MPCH
      ACCEPT "THRESHOLD (SILENCE/SPEECH): ",THRESH
      ACCEPT "PRE/DE-EMPHASIZE? (YES-1,NO-0): ",NEMP
      ACCEPT "UNVOICED GAIN FACTOR (UNGA): ",UNGA
      ACCEPT "SCALE FACTOR (SCAF): ",SCAF
      ACCEPT "VOICED/UNVOCD THRESHOLD: ",STHR
      ACCEPT "GLOTTAL PULSE SHAPE(1-POLY,3-IMPULSE): ",NGLT
      ACCEPT "HAMMING WINDOW? (1-Y,0-N):  ",H1
      ACCEPT "PITCH AND COEFFICIENT FILES THE SAME?(0-NO,1-YES)",NPCS
      GO TO 46

 45   POLES = INTVAR(2)
      MP = INTVAR(3)
      MAXFR = INTVAR(4)
      MAXPT = INTVAR(5)
      NEMP = INTVAR(6)
      NGLT = INTVAR(7)
      H1 = INTVAR(8)
      MPCH = INTVAR(9)
      NPCS = INTVAR(10)
      STHR = RELVAR(1)
```

```
        SCAF = RELVAR(2)
        THRESH = RELVAR(3)
        UNGA = RELVAR(4)

  46    TYPE POLES," POLES "
        NPOLES = POLES + 1
        MAXFR1 = MAXFR - 1                    ;FOR HAMMING WINDOW
        WRITE BINARY(3) POLES,NEMP,UNGA,NGLT        ; CHANNEL WRITE

C*************************************************************************
C*************************************************************************
C***  THE OPERATIONAL PROGRAM...

C***  START OF LOOP
  50    CONTINUE

C***  READ IN A NEW BLOCK OF DATA
        CALL RDBLK(1,K,VAL,5,IER)     ; READ THE SPEECH INTO AN ARRAY
        IF (IER.EQ.9) GO TO 260
        IF (IER.NE.1) TYPE "READ FILE ERROR ", IER

C***  START A NEW FRAME OF PITCH DETECTION
  52    NSET = NSET + 1
        NPOINT = 0
        F = S + MAXFR - 1
        IF(NPCS.EQ.0) GO TO 60
        DO 55 J=S,F
            NPOINT=NPOINT+1
            SPCH(NPOINT) = FLOAT(VAL(J))/SCAF    ; SPCH FOR SIFT & ENER
 55     CONTINUE
        GO TO 61

C***  USED IF PITCH AND COEFFICIENT FILES ARE DIFFERENT
 60     CALL RDBLK(4,K,NAL,5,IER)     ; READ THE SPEECH INTO AN ARRAY
        IF (IER.EQ.9) GO TO 260
        IF (IER.NE.1) TYPE "READ FILE ERROR ", IER
        DO 61 J=S,F
            NPOINT=NPOINT+1
            SPCH(NPOINT) = FLOAT(NAL(J))/SCAF    ; SPCH FOR SIFT & ENER
 61     CONTINUE

C***  CALCULATE ENERGY IN A FRAME
 62     CALL ENER(SPCH,THRESH,NEN,AENRG,MAXFR)
        IF (NEN.EQ.0) GO TO 65        ; NO NEED TO GET PITCH

C***  CALL TO THE SUBROUTINES WHICH PERFORM PITCH ANALYSIS
 63     IF(MPCH.EQ.0) CALL SIFTA(SPCH,PITCH,STHR,MAXFR)
        IF(MPCH.EQ.1) CALL AUTOC(SPCH,PITCH,STHR,MAXFR)

 65     DELAY = NSET - 2
        IF (DELAY.LE.0) GO TO 225     ; TRUE PITCH IS DELAYED


C***  GET PREDICTION COEFFICIENTS
 70     CONTINUE
        NFRAME = NFRAME + 1
        IF((MOD(NFRAME,25)).EQ.0) TYPE NFRAME," FRAMES PROCESSED"
        IF(AENRG(3).LT.THRESH) GO TO 204     ; NO NEED TO FIND COEFFICIENTS

        DO 75 I = 1,20              ;
```

```
            RCOF(I) = 0.0          ; INITIALIZE ARRAYS
            AR(I)   = 0.0          ;   FOR PREDICTION
 75       CONTINUE                 ;   COEFFICIENTS
          AL = 0.0                 ;
          NPOINT = 0
          F = KS + MAXFR - 1       ; A COUNTER

C***   LOAD ARRAY AND PREEMPHASIZE FOR COEFFICIENT GENERATION
          IF(H1.EQ.0) GO TO 179    ; NO NEED FOR HAMMING WINDOW
          IF(NEMP.EQ.0) GO TO 81   ; NO PRE-EMPHASIS
          DO 80 J = KS,F
               YMEMD = FLOAT(VAL(J))*(.54-.46*COS(NPOINT*6.28318/MAXFR1))
               SP1 = YMEMD - .9*YMEMN
               NPOINT = NPOINT + 1
               SPEE(NPOINT) = SP1/SCAF
               YMEMN = YMEMD
  80       CONTINUE
          GO TO 82
C***   NO PRE-EMPHASIS
  81      DO 82 J = KS,F
               SP1 = FLOAT(VAL(J))*(.54-.46*COS(NPOINT*6.28318/MAXFR1))
               NPOINT = NPOINT + 1
               SPEE(NPOINT) = SP1/SCAF
  82       CONTINUE
          GO TO 189

C*** NO HAMMING WINDOW
 179      IF(NEMP.EQ.0) GO TO 181     ; NO PRE-EMPHASIS
          DO 180 J = KS,F
               YMEMD = FLOAT(VAL(J))
               SP1 = YMEMD - .9*YMEMN
               NPOINT = NPOINT + 1
               SPEE(NPOINT) = SP1/SCAF
               YMEMN = YMEMD
 180      CONTINUE
          GO TO 189
C***   NO PRE-EMPHASIS
 181      DO 182 J = KS,F
               NPOINT = NPOINT + 1
               SPEE(NPOINT) = FLOAT(VAL(J))/SCAF
 182      CONTINUE
 189      CONTINUE


C***   CALL TO SUBROUTINE TO DETERMINE THE FILTER COEFFICIENTS
 190      IF (MP.EQ.0) CALL AUTO(MAXFR,SPEE,POLES,AR,AL,RCOF)
          IF (MP.EQ.1) CALL COVAR(MAXFR,SPEE,POLES,AR,AL,RCOF)

C***   CALCULATE VALUES TO BE WRITTEN TO THE CHANNEL
          IF (PITCH(3).EQ.0.0) GO TO 200   ;UNVOICED SPEECH
C***   VOICED SPEECH
          PIT = INT(PITCH(3))
          VOCD = 1
          P1 = 2*PIT
          IF(JUMP.NE.0) P1 = P1/2      ; IF PREVIOUS SET NOT VOICED,
          IF((P1.GT.MAXPT).AND.(JUMP.EQ.0)) P1 = P1/2
          JS = JS + P1                 ;     MORE FREQUENT ANALYSIS
          KS = KS + P1
          JUMP = 0
          GO TO 210
```

```
C***  UNVOICED SPEECH
 200    PIT = 0
        VOCD = 0
        P1 = MAXPT
        IF (JUMP.NE.1) P1 = P1/2      ; IF PREVIOUS SET NOT UNVOICED,
        JS = JS + P1                  ;    MORE FREQUENT ANALYSIS
        KS = KS + P1
        JUMP = 1
        GO TO 210
C***  SILENCE
 204    PIT = 0
        AL = 0.0
        VOCD = 2
        P1 = MAXPT
        JS = JS + P1
        KS = KS + P1
        JUMP = 2
        DO 209  I = 2,20
            AR(I) = 0.0
  209   CONTINUE

C***  WRITE COEFFICIENTS TO CHANNEL FILE
  210   CONTINUE
        NPTS = NPTS + P1
        WRITE BINARY(3) VOCD,P1,PIT,AL              ;CHANNEL WRITE
        WRITE BINARY(3) (AR(J),J=2,NPOLES)          ;CHANNEL WRITE
X211    TYPE VOCD,P1,PIT,AL,AENRG(3)
X       WRITE(12,212) AL
X212    FORMAT(1X,F12.3)
X       WRITE(12,213) (AR(J),J=1,NPOLES)
X213    FORMAT(9(1X,F12.6))
X       ACCEPT "CONTINUE?(1-YES,0-NO): ",ICK
X       IF(ICK.EQ.0) GO TO 290

C***  BOOKKEEPING ROUTINE
 220    IF(JS.LE.MAXPT) GO TO 70  ; GET PREDICTION COEFFICIENTS
        JS = JS - MAXPT
 225    CONTINUE
        S = S + MAXPT
        IF (S.LT.768) GO TO 50    ; GO TO START OF LOOP
        S = S - 256
        KS= KS- 256
        K = K+1
        IF (JEND.GT.1) GO TO 270  ; AFTER DELAY OF TWO, FINALLY EXIT
        IF (JEND.EQ.1) GO TO 70
        GO TO 50                  ; GO TO START OF LOOP

 260    CONTINUE
        JEND = JEND + 1
        GO TO 70          ; CALCULATE PREDICTOR COEFFICIENTS AGAIN

C*************************************************************************
C*************************************************************************

C***  EXIT PROCESS
 270    MSET=NSET                ;TOTAL # OF SETS
        ICK = 1                   ;   AN END-OF-FILE INDICATOR
        WRITE BINARY(3) ICK       ;   CHANNEL WRITE

C*** CLOSE THE FILES
```

```
290     CALL CLOSE(1,IER)
        CALL CLOSE(4,JER)
        IF ((IER.NE.1).OR.(JER.NE.1)) TYPE " CLOSE FILE ERROR ",IER,JER
        CALL CLOSE(2,IER)
        CALL CLOSE(3,JER)
        IF ((IER.NE.1).OR.(JER.NE.1)) TYPE " CLOSE FILE ERROR ",IER,JER

        TYPE " NPTS = ",NPTS," MSET = ",MSET
        STOP
        END
```

```
      SUBROUTINE IOF(N,MAIN,F1,F2,F3,F4,MS,S1,S2,S3,S4)
C*********************************************************************
C       ADAPTED FROM SUBROUTINE WRITTEN BY LT. SIMMONS 10 SEPT 81
C
C       THIS FORTRAN 5 SUBROUTINE WILL READ FROM THE FILE COM.CM
C       (FCOM.CM IN THE FORE GROUND) THE PROGRAM NAME, ANY GLOBAL
C       SWITCHES, AND UP TO FOUR LOCAL FILE NAMES AND CORRESPONDING
C       LOCAL SWITCHES.
C
C       ARGUMENTS:
C
C       N IS THE NUMBER OF LOCAL FILES AND SWITCHES TO BE READ FROM
C       (F)COM.CM.   N MUST BE 1, 2, 3, OR 4.
C
C       MAIN IS AN ASCII ARRAY FOR THE MAIN PROGRAM FILE NAME.
C
C       F1, F2, F3, AND F4 ARE THE FOUR ASCII ARRAYS TO RETURN THE
C       LOCAL FILE NAMES.
C
C       MS IS A TWO-WORD INTEGER ARRAY THAT HOLDS ANY GLOBAL SWITCHES
C
C       S1, S2, S3, AND S4 ARE TWO-WORD INTEGER ARRAYS THAT HOLD THE
C       LOCAL SWITCHES CORRESPONDING TO F1 THROUGH F4 RESPECTIVELY.
C
C*********************************************************************

      DIMENSION MAIN(7),MS(2)
      INTEGER F1(7),F2(7),F3(7),F4(7),S1(2),S2(2),S3(2),S4(2)

C     CHECK BOUNDS ON N
      IF((N.LT.1).OR.(N.GT.4)) STOP    ; N OUT OF BOUNDS

C     PROCESS THE DATA IN (F)COM.CM
      CALL GROUND(I)   ;FIND OUT WHICH GROUND PROGRAM IS IN
      IF(I.EQ.0)OPEN 0,"COM.CM"        ;OPEN CH. 0 TO COM.CM
      IF(I.EQ.1)OPEN 0,"FCOM.CM"       ;OPEN CH. 0 TO FCOM.CM
      CALL COMARG(0,MAIN,MS,IER)       ;READ FROM (F)COM.CM
      IF(IER.NE.1) TYPE" COMARG ERROR:",IER
      WRITE(10,1) MAIN(1)              ;TYPE PROGRAM NAME
1     FORMAT(' PROGRAM ',S13,'RUNNING.')
      CALL COMARG(0,F1,S1,JER)         ;READ FROM (F)COM.CM
      IF(JER.NE.1) TYPE" COMARG ERROR (F1):",JER
      IF(N.EQ.1) GO TO 2               ;TEST N
      CALL COMARG(0,F2,S2,KER)         ;READ FROM (F)COM.CM
      IF(KER.NE.1) TYPE" COMARG ERROR (F2):",KER
      IF(N.EQ.2) GO TO 2               ;TEST N
      CALL COMARG(0,F3,S3,LER)         ;READ FROM (F)COM.CM
      IF(LER.NE.1) TYPE" COMARG ERROR (F3):",LER
      IF(N.EQ.3) GO TO 2               ;TEST N
      CALL COMARG(0,F4,S4,LER)         ;READ FROM (F)COM.CM
      IF(LER.NE.1) TYPE" COMARG ERROR (F4):",LER

2     CLOSE 0
      RETURN
      END
```

```
C*****************************************************************
C
C       THIS SUBROUTINE DETERMINES WHETHER A FRAMES ENERGY
C       EXCEEDS A SILENCE THRESHOLD  (NEN=0:SILENCE; NEN=1:SPEECH)
C
C       AENRG IS A THREE-MEMBER ARRAY WHICH HOLDS A MEMORY OF THE
C             PREVIOUS VALUES OF THE COMPUTED ENERGY.
C
C*****************************************************************
        SUBROUTINE ENER(SPCH,THRESH,NEN,AENRG,MAXFR)

        DIMENSION SPCH(1),AENRG(1)
        NEN = 1                          ; PRESET DECISION TO SPEECH
        SUM = 0.0                        ; INITIALIZE SUM
        DO 100 J=1,MAXFR
          S1 = SPCH(J)
          SUM = SUM + S1 * S1            ; ENERGY = SUM OF SQUARES
100     CONTINUE
        AENRG(3) = AENRG(2)
        AENRG(2) = AENRG(1)
        AENRG(1) = SUM
        IF(SUM.LT.THRESH) NEN = 0
        RETURN
        END
```

```
C*************************************************************************
C
C       SIFT ALGORITHM PROCESSING - STEP1
C
C       INPUT PARAMETERS:  SPCH(J)  (J=1,2,...,MAXFR)
C                THE SPEECH SIGNAL TO BE PROCESSED FOR PITCH
C
C       OUTPUT PARAMETER:  PITCH(J)  (J=1,2,3)
C                                (UNITS IN SAMPLES)
C
C       NOTE: PARAMETERS FIXED FOR FS=8 KHZ
C
C*************************************************************************

        SUBROUTINE SIFTA(SPCH,PITCH,STHR,MAXFR)
        DIMENSION SPCH(1),PBUF(100),AF(4),PF(4),DF(5),D(5),ABUF(33)
        DIMENSION U(100),A(5),P(5),RC(5),PITCH(1)
        DATA AF/1.,-2.340366,2.011900,-.614109/
        DATA PF/.0357082,-.0069956,-.0069956,.0357082/
        DATA P/1.,4*0./

        MAX4 = INT(MAXFR/4)          ;MAXFR = 320  ;MAX4 = 80
        MAX80 = MAX4
        AX4 = FLOAT(MAX4) - 1.                     ;AX4 = 79.
        AX5 = AX4 - 4.                          ;AX5 = 75.
        MAX6 = MAX4 - 4
C***    INITIALIZE MEMORY OF DIRECT TO ZERO
        DO 10 J=1,5
            DF(J)=0.0
            D(J)=0.0
  10    CONTINUE
C***    PRE-FILTER, DOWN-SAMPLER, DIFFERENCER AND HAMMING WINDOWER.
        UPREV=0.0
        DO 20 J=1,MAXFR
            CALL DIRECT(AF,PF,3,DF,SPCH(J),SOUT)
            IF (MOD(J,4).NE.0) GO TO 20
            K=J/4
            PBUF(K)=SOUT
            U(K)=(SOUT-UPREV)*(.54-.46*COS((K-1.)*6.28318/AX4))
            UPREV=SOUT
  20    CONTINUE
C***    COMPUTE INVERSE FILTER COEFFICIENTS
        CALL AUTO(MAX4,U,4,A,ALP,RC)
C***    PERFORM INVERSE FILTERING AND HAMMING WINDOW
        DO 30 J=1,MAX80
            CALL DIRECT(P,A,4,D,PBUF(J),FOUT)
            IF (J.LE.4) GO TO 30
            PBUF(J-4)=FOUT*(.54-.46*COS((J-5)*6.28318/AX5))
  30    CONTINUE
C***    PERFORM AUTOCORRELATION ON PITCH BUFFER
        DO 25 JJ=1,33
            J=JJ-1
            NMJ=MAX6 - J
            SUM=0.
            DO 15 I=1,NMJ
                IPJ=I+J
                SUM=SUM+PBUF(I)*PBUF(IPJ)
  15        CONTINUE
            ABUF(JJ)=SUM
  25    CONTINUE
```

```
C***    OBTAIN PITCH VALUES FROM LAST THREE FRAMES
        P1=PITCH(1)/4.  + 1.
        P2=PITCH(2)/4.  + 1.
        P3=PITCH(3)/4.  + 1.
        IF(PITCH(1).EQ.0.0) P1 = 0.0
        IF(PITCH(2).EQ.0.0) P2 = 0.0
        IF(PITCH(3).EQ.0.0) P3 = 0.0
C***    GET PEAK WITHIN RANGE[6,32]
        L=6
        AMAX=ABUF(L)
        DO 35 J=6,32
            IF(ABUF(J).LE.AMAX) GO TO 35
            AMAX=ABUF(J)
            L=J
  35    CONTINUE
C***    TEST FOR MAX EQUAL ZERO
        IF (AMAX.EQ.0.) GO TO 60
C***    TEST FOR LEFT HAND EDGE.  IF ABUF(L) IS NOT A PEAK SET UNVOICED
        IF (ABUF(L).LT.ABUF(L-1)) GO TO 60
C***    PERFORM PARABOLIC INTERPOLATION ABOUT LOCATION L
        AA=ABUF(L-1)-ABUF(L)
        AA=(AA+ABUF(L+1)-ABUF(L))/2.
        BB=(ABUF(L+1)-ABUF(L-1))/4.
        AP=ABUF(L)-BB*BB/AA
        AL=L-BB/AA
        V=AP/ABUF(1)
C***    TEST WITH VARIABLE THRESHOLD
        IF (L.GE.19) GO TO 40
        DD =-1.*(L-6.)/13.+2.
        GO TO 50
  40    CONTINUE
        DD =-1.*(L-19.)/13.+1
  50    CONTINUE
        V=V/DD
C***    DECISIONS
        IF(V.GE.STHR) GO TO 70
        IF(P1.EQ.0.) GO TO 60
        STHQ = .9*STHR
        IF(V.GE.STHQ) GO TO 70
  60    P0=0.
        GO TO 80
  70    P0=AL

  80    IF(ABS(P1-P3).LE..375*P3) P2=(P1+P3)/2.

C***    IF(P0 AND P1 ARE CLOSE) AND (P2 NOT 0) BUT P3 = 0, THEN
C***    USE LINEAR EXTRAPOLATION FOR P2 (COMING OUT OF VOICED).
        IF (P3.NE.0.) GO TO 90
        IF(P2.EQ.0.)GO TO 90
        IF (ABS(P0-P1).GT.0.2*P1) GO TO 90
        P2=(2.*P1)-P0
C***    TEST FOR ISOLATED "VOICED" AND INCORRECT END OF "VOICED"
  90    IF (P1.NE.0.) GO TO 100
        IF (ABS(P2-P3).GT.(.375*P3)) P2=0.
C***    UPDATE FRAMES
 100    PITCH(3)=(P2 - 1.)*4.
        PITCH(2)=(P1 - 1.)*4.
        PITCH(1)=(P0 - 1.)*4.
        IF(P2.EQ.0.0) PITCH(3) = 0.0
```

```
      IF(P1.EQ.0.0) PITCH(2) = 0.0
      IF(P0.EQ.0.0) PITCH(1) = 0.0

      RETURN
      END
```

```
C*************************************************************
C
C       THIS SUBROUTINE CALCULATES THE PITCH PERIOD
C
C       INPUT PARAMETERS:  SPCH(J)   J=1,2,...,MAXFR
C             THE SPEECH SIGNAL TO BE PROCESSED FOR PITCH
C
C       OUTPUT PARAMETERS:   PITCH(J)   J=1,2,3
C                THE PITCH IN NUMBER OF SAMPLES
C
C       NOTE:  PARAMETERS SET FOR FS = 8KHZ
C
C*************************************************************

        SUBROUTINE AUTOC(SPCH,PITCH,STHR,MAXFR)
        DIMENSION SPCH(1),AF(4),PF(4),DF(5),ABUF(33),PBUF(400)
        DIMENSION PITCH(1)
        INTEGER MXFTH,MNFTH,MXLTH,MNLTH
        DATA PF/.0357082,-.0069956,-.0069956,.0357082/
        DATA AF/1.,-2.340366,2.011900,-.614109/

        AXFR = MAXFR - 1
C***  INITIALIZE MEMORY OF DIRECT TO ZERO
        DO 10 I = 1,5
            DF(I) =0.0
  10    CONTINUE

C***  PREFILTER AND FIND PEAKS IN FIRST & LAST THIRD OF FRAME
C***  MINIMUM OF THESE IS CLIPPING THRESHOLD
        NFIRTH = INT(MAXFR/3)
        NLASTH = INT(MAXFR*2/3)
        MXFTH = 0.0                    ; SET COMPARATORS TO ZERO
        MXLTH = 0.0
        DO 20 I = 1,MAXFR
            CALL DIRECT(AF,PF,3,DF,SPCH(I),SOUT)
            PBUF(I) = SOUT*(.54-.46*COS((I-1.)*6.2318/AXFR))
X           PBUF(I) = SOUT
            IF ((I.LE.NFIRTH).AND.(PBUF(I).GT.MXFTH))MXFTH = PBUF(I)
            IF ((I.GE.NLASTH).AND.(PBUF(I).GT.MXLTH))MXLTH = PBUF(I)
  20    CONTINUE
        IF(MXFTH.LE.MXLTH) MXLTH = MXFTH   ;MIN PEAK IS MXLTH
        MXFTH = .75*MXFTH
        MNFTH = .50*MXFTH
        MXLTH = -(MXFTH)
        MNLTH = -(MNFTH)
C*** CLIP SPEECH
        DO 40 I = 1,MAXFR
            IF(PBUF(I).LT.MXFTH) GO TO 25 .
            PBUF(I) = 1.0
            GO TO 40
  25        IF(PBUF(I).LT.MNFTH) GO TO 26
            PBUF(I) = .5
            GO TO 40
  26        IF(PBUF(I).LT.MXLTH) GO TO 30
            IF(PBUF(I).LT.MNLTH) GO TO 29
            PBUF(I) = 0.0
            GO TO 40
  29        PBUF(I) = -.5
            GO TO 40
  30        PBUF(I) = -1.0
```

```
 40     CONTINUE

C***   COMPUTE AUTOCORRELATIONS
       DO 60 JJ = 1,151
            J = JJ-1
            NMJ = MAXFR - J
            SUM = 0.0
            DO 50 I = 1,NMJ
                 IPJ = I + J
                 SUM = SUM + PBUF(I)*PBUF(IPJ)
 50         CONTINUE
            ABUF(JJ) = SUM
 60     CONTINUE

C***   OBTAIN PITCH VALUES FROM LAST THREE FRAMES
       P1 = PITCH(1)*2.5
       P2 = PITCH(2)*2.5
       P3 = PITCH(3)*2.5
       L = 16
       AMAX = ABUF(L)
       DO 70 J = 16,150
            IF(ABUF(J).LE.AMAX) GO TO 70
            AMAX = ABUF(J)
            L = J
 70     CONTINUE
       IF(AMAX.EQ.0.0) GO TO 100        ;TEST FOR MAX EQUAL ZERO
       IF(ABUF(L).LT.ABUF(L-1)) GO TO 100   ;TEST FOR L.H. EDGE
       V = ABUF(L)/ABUF(1)
       AL = L
C***   TEST V WITH THE THRESHOLD
       IF(V.GE.STHR) GO TO 110
       IF(P1.EQ.0.0) GO TO 100
       STHQ = .9*STHR
       IF(V.GE.STHQ) GO TO 110
 100   P0 = 0.0
       GO TO 120
 110   P0 = AL
 120   IF(ABS(P1-P3).LE..375*P3) P2=(P1+P3)/2.
       IF(P3.NE.0.) GO TO 130
       IF(P2.EQ.0.) GO TO 130
       IF(ABS(P0-P1).GT.0.2*P1) GO TO 130
       P2=(2.*P1)-P0
C***  TEST FOR ISOLATED "VOICED" & INCORRECT END OF "VOICED"
 130   IF(P1.NE.0.0) GO TO 140
       IF(ABS(P2-P3).GT.(.375*P3)) P2 = 0.0
C***   UPDATE PITCH
 140   PITCH(3) = P2/2.5
       PITCH(2) = P1/2.5
       PITCH(1) = P0/2.5

       RETURN
       END
```

```
C*****************************************************************
C
C       THIS ROUTINE IMPLEMENTS THE DIRECT FORM FILTER.
C
C*****************************************************************
        SUBROUTINE DIRECT(A,P,M,D,XIN,XOUT)

        DIMENSION A(1),P(1),D(1)

        XOUT = 0.0
        D(1) = XIN
        DO 10 J = 1, M
            I = M + 1 - J
            XOUT = XOUT + D(I+1)*P(I+1)
            D(1) = D(1) - A(I+1)*D(I+1)
            D(I+1) = D(I)
  10    CONTINUE
        XOUT = XOUT + D(1)*P(1)
        RETURN
        END
```

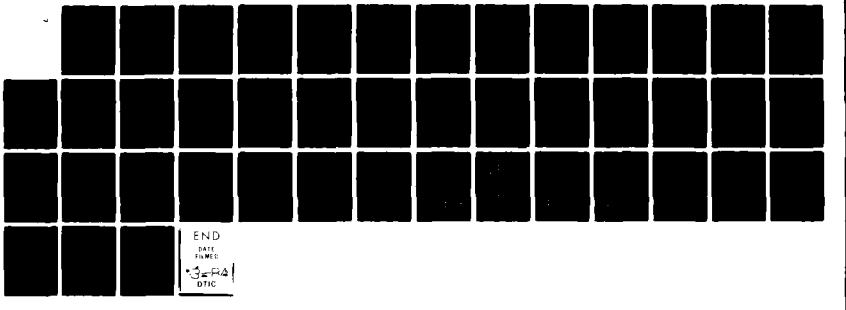
```
C*********************************************************************
C
C       SUBROUTINE AUTO AS PRESENTED ON PAGE 219 OF MARKEL & GRAY
C
C       THE ARITHMETIC IN THIS SUBROUTINE IS PERFORMED IN DOUBLE
C       PRECISION TO REDUCE THE EFFECTS OF ILL-CONDITIONING OF THE
C       AUTOCORRELATION MATRIX.
C
C*********************************************************************

        SUBROUTINE AUTO(N,X,M,A,ALPHA,RC)

        DIMENSION X(1),A(1),RC(1)
        DOUBLE PRECISION DA(20),DRC(20),DR(21),DAL,S,AT

C***  SET THE INITIAL VALUES TO ZERO
        DO 5 I = 1,20
            DA(I) = DBLE(0.0)
            DRC(I) = DBLE(0.0)
  5     CONTINUE

        MP=M+1
C***  COMPUTE THE AUTOCORRELATION TERMS
        DO 15 K=1,MP
            DR(K)=DBLE(0.0)
            NK=N-K+1
            DO 10 NP=1,NK
               DR(K)=DR(K)+DBLE(X(NP)*X(NP+K-1))
 10         CONTINUE
 15     CONTINUE
        DO 17 I = 2,21
            DR(I) = DR(I)/DR(1)
 17     CONTINUE
        ALPHA = SNGL(DR(1))
        DR(1) = DBLE(1.0)
        DRC(1)=-DR(2)/DR(1)
        DA(1)=DBLE(1.0)
        DA(2)=DRC(1)
        DAL=DR(1)+DR(2)*DRC(1)
        DO 40 MINC=2,M
            S=DBLE(0.0)
            DO 20 IP=1,MINC
                S=S+DR(MINC-IP+2)*DA(IP)
 20         CONTINUE
            DRC(MINC)=-S/DAL
            MH=MINC/2+1
            DO 30 IP=2,MH
                IB=MINC-IP+2
                AT=DA(IP)+DRC(MINC)*DA(IB)
                DA(IB)=DA(IB)+DRC(MINC)*DA(IP)
                DA(IP)=AT
 30         CONTINUE
            DA(MINC+1)=DRC(MINC)
            DAL=DAL+DRC(MINC)*S
            IF(DAL) 50,50,40
 40     CONTINUE
        DO 45 I = 1,20
            A(I) = SNGL(DA(I))
            RC(I) = SNGL(DRC(I))
 45     CONTINUE
```

```
      ALPHA = SNGL(DAL)
      ALPHA = SQRT(ALPHA*RO)
50    RETURN
      END
```

AD-A138 110 IMPLEMENTING LPC (LINEAR PREDICTIVE CODING) ALGORITHMS IN THE STUDY OF SP..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. C E MCKOWN

UNCLASSIFIED DEC 83 AFIT/GE/EE/83D-45 F/G 9/4 NL

2/2

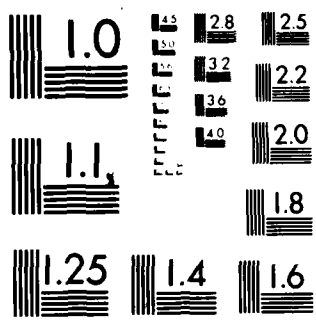


END

DATE FILMED

3-84

DTIC



```
C*************************************************************************
C
C       SUBROUTINE COVAR AS PRESENTED ON PG 221 OF MARKEL & GRAY
C
C       THE NUMERICAL MANIPULATIONS REQUIRED IN THIS ALGORITHM ARE
C       PERFORMED IN DOUBLE PRECISION ARITHMETIC TO COMBAT POSSIBLE
C       ILL-CONDITIONING OF THE COVARIANCE MATRIX.
C
C*************************************************************************

        SUBROUTINE COVAR(N, X, M, A, ALPHA, GRC)

        DIMENSION X(1), A(1), GRC(1)
        DOUBLE PRECISION B(210), BETA(20), CC(21)
        DOUBLE PRECISION DA(20), DGRC(21), DALPHA, S, GAM

        MP = M + 1
C***  SET THE INITIAL VALUES TO ZERO.
        DO 299 I = 1,210
            B(I) = DBLE(0.0)
 299    CONTINUE
        DALPHA = DBLE(0.0)
        CC(1) = DBLE(0.0)
        CC(2) = DBLE(0.0)

C***  CALCULATE THE COVARIANCE TERMS
        DO 10 NP = MP,N
            NP1 = NP - 1
            DALPHA = DALPHA + DBLE(X(NP)*X(NP))
            CC(1) = CC(1) + DBLE(X(NP)*X(NP1))
            CC(2) = CC(2) + DBLE(X(NP1)*X(NP1))
  10    CONTINUE

        B(1) = DBLE(1.0)
        BETA(1) = CC(2)
        DGRC(1) = -CC(1)/CC(2)
        DA(1) = DBLE(1.0)
        DA(2) = DGRC(1)
        DALPHA = DALPHA + DGRC(1)*CC(1)
        MF = M
        DO 130 MINC = 2,MF
C*** CALCULATE THE COVARIANCE TERMS
            DO 20 J = 1,MINC
                JP = MINC + 2 - J
                N1 = MP + 1 - JP
                N2 = N + 1 - MINC
                N3 = N + 2 - JP
                CC(JP) = CC(JP-1)+DBLE(X(MP-MINC)*X(N1))
     X          -DBLE(X(N2)*X(N3))
  20        CONTINUE

            CC(1) = DBLE(0.0)
            DO 30 NP = MP,N
                CC(1) = CC(1) + DBLE(X(NP-MINC)*X(NP))
  30        CONTINUE
            MSUB = (MINC*MINC-MINC)/2
            MM1 = MINC - 1
            B(MSUB+MINC) = DBLE(1.0)
            DO 70 IP = 1,MM1
                ISUB = (IP*IP-IP)/2
```

```
            IF (BETA(IP)) 140,70,40
 40         GAM = DBLE(0.0)
            DO 50 J = 1,IP
               GAM = GAM+CC(J+1)*B(ISUB+J)
 50         CONTINUE
            GAM = GAM/BETA(IP)
            DO 60 JP = 1,IP
               B(MSUB+JP)=B(MSUB+JP)-GAM*B(ISUB+JP)
 60         CONTINUE
 70      CONTINUE
         BETA(MINC) = DBLE(0.0)
         DO 80 J = 1,MINC
            BETA(MINC) = BETA(MINC)+CC(J+1)*B(MSUB+J)
 80      CONTINUE
         IF (BETA(MINC)) 140,120,90
 90      S = DBLE(0.0)
         DO 100 IP = 1,MINC
            S = S + CC(IP)*DA(IP)
100      CONTINUE
         DGRC(MINC) = -S/BETA(MINC)
         DO 110 IP = 2,MINC
            M2 = MSUB + IP - 1
            DA(IP) = DA(IP) + DGRC(MINC)*B(M2)
110      CONTINUE
         DA(MINC+1) = DGRC(MINC)
120      CONTINUE
         S = DGRC(MINC)*DGRC(MINC)*BETA(MINC)
         DALPHA = DALPHA - S
         IF (DALPHA) 140,140,130
130   CONTINUE

140   CONTINUE
      DO 150 I = 1,MP
         A(I) = SNGL(DA(I))
         GRC(I) = SNGL(DGRC(I))
150   CONTINUE
      ALPHA = SNGL(SQRT(DALPHA))
      RETURN
      END
```

```
C*****************************************************************
C
C       PROGRAM:        VOCODE.
C       AUTHOR:         CRAIG MCKOWN
C       DATE:           24 AUG 83 - 30 SEP 83
C
C       FUNCTION:       USES THE OUTPUT OF "PREDICT" (THE LPC CODER) TO
C                       PRODUCE OUTPUT SPEECH.  THIS IS A VOCODER.
C
C       LOAD LINE:      RLDR VOCODE IOF UNVOCD DRAND GLOT1 GLOT2 GLOT3
C                       THROAT @FLIB@
C
C*****************************************************************
C
C       PARAM:  FILE FROM WHICH LPC DATA IS READ
C       DUMMY:  A DUMMY FILE, NOT USED
C       RUMMY:  FILE TO WHICH NORMALIZED VOCODED SPEECH IS WRITTEN
C       AR:     LPC COEFFICIENTS - AR(1) IN THIS PROGRAM IS THE SAME
C               AS AR(2) IN THE CODER PROGRAM.
C       U:      OUTPUT OF "UNVOCD" OR "VOICED" - AN INPUT TO "THROAT"
C       W:      MEMORY FOR "THROAT"
C       S:      OUTPUT OF "THROAT" - VOCODED SPEECH
C       INTS:   ARRAY WITH INTEGER VALUES OF S
C       X:      INTEGER ARRAY USED FOR WRBLK
C       POLES:  THE NUMBER OF POLES OF THE OUTPUT FILTER.
C       VOCD:   FLAG WHICH DENOTES VOICED/UNVOICED DECISION FROM CODER
C       PIT:    PITCH INFORMATION (PITCH PERIOD IN SAMPLES)
C       IX:     DOUBLE PRECISION SEED NUMBER FOR SUBROUTINE "UNVOCD"
C
C*****************************************************************

        INTEGER SPEEFL(7),PARAM(7),DUMMY(7),RUMMY(7),MAIN(7)
        INTEGER POLES,P1,X(256),VOCD,PIT,INTS(200)
        DIMENSION U(250),AR(20),W(0:20),S(250)
        DOUBLE PRECISION IX

        DATA VALF,NPTS,N6,N5/0.0,0,0,0/
        DATA IS,IP,KS,KEND/1,0,0,0/
        IX = DBLE(203)
        NDONE = 0
        NFILES = 4
        CALL IOF(NFILES,MAIN,SPEEFL,DUMMY,PARAM,RUMMY,MS,S1,S2,S3,S4)
        CALL DFILW(RUMMY,IER)
        IF(IER.EQ.13) GO TO 40
        IF(IER.NE. 1) TYPE " DELETE FILE ERROR ",IER
  40    CALL CFILW(RUMMY,3,88,IER)
        IF (IER.NE.1) TYPE " CREATE FILE ERROR ", IER
        CALL OPEN(4,RUMMY,3,IER)
        IF(IER.NE. 1) TYPE " OPEN FILE ERROR ",IER
        CALL OPEN(3,PARAM,3,IER)
        IF(IER.NE.1) TYPE " OPEN FILE ERROR ",IER

C***  READ OVERALL PAREMETERS FOR VOCODING OF SPEECH
        READ BINARY(3) POLES,NEMP,UNGA,NGLT

C*****************************************************************
C***  SYNTHESIZE ONE [VARIABLE LENGTH] FRAME OF SPEECH
 100    CONTINUE
C***  READ FRAME PARAMETERS
        READ BINARY(3,END=1001) VOCD,P1,PIT,AL
```

```
        DO 110 J=1,POLES
            READ BINARY(3,END=1001) AR(J)
 110    CONTINUE

C***  SET MEMORY OF OUTPUT FILTER TO ZERO
        DO 120 I = 0,20
            W(I) = 0.0
 120    CONTINUE
C***  VOCD/UNVOCD/SILENCE DECISION
        IF(VOCD.EQ.1) GO TO 300    ;VOICED SPEECH
        IF(VOCD.EQ.2) GO TO 400    ;SILENCE
C***  UNVOICED SPEECH
        CALL UNVOCD(U,P1,IX)
        AL = AL*UNGA                              ; UNVOICED GAIN FACTOR
        CALL THROAT(U,P1,AR,POLES,AL,S,W)
        GO TO 500
C***  VOICED SPEECH
 300    CONTINUE
        IF(NGLT.EQ.1) CALL VOICED1(U,PIT,P1)          ; POLY GLOT SHAPE
        IF(NGLT.EQ.2) CALL VOICED2(U,PIT,P1)          ; TRIG GLOT SHAPE
        IF(NGLT.EQ.3) CALL VOICED3(U,PIT,P1)          ; IMPULSE GLOT SHAPE
        CALL THROAT(U,P1,AR,POLES,AL,S,W)
        GO TO 500
C***  SILENCE
 400    CONTINUE
        DO 450 I = 1,P1
            S(I) = 0.0                ; AUTOMATICALLY SET S TO ZERO
 450    CONTINUE

C***  DE-EMPHASIZE AND WRITE SPEECH
 500    IF(NEMP.EQ.0) GO TO 555       ;NO PRE/DE-EMPHASIS
        CONTINUE
        DO 550 J = 1,P1
            IF(VALF.GT.2500.) VALF = 2500.            ;
            IF(VALF.LT.-2500.) VALF = -2500.
            IF((VALF.GT. -0.01).AND.(VALF.LT. 0.01)) VALF = 0.0
            VALD = S(J)
            IF(VALD.GT.2000.) VALD = 2000.
            IF(VALD.LT.-2000.) VALD = -2000.
            VALE = VALD + .9*VALF  ;Y(Z) = X(Z) + .9(Z**-1)Y(Z)
            VALF = VALE
            INTS(J) = INT(VALE)
 550    CONTINUE
        GO TO 560
C***  NO DE-EMPHASIS
 555    CONTINUE
        DO 560 J=1,P1
            INTS(J) = INT(S(J))
 560    CONTINUE

C***  COUNTER & WRITE ROUTINE
        IP = IP + P1
        L = 1
        IF(IP.GE.256) GO TO 210   ;SPLIT S(I) & WRBLK(...,X,...)
        DO 200 I = IS,IP
            X(I) = INTS(L)        ;LOAD UP X(I) AS REQUIRED
            L = L + 1
 200    CONTINUE
        GO TO 240                 ;SKIP WRBLK
 210    CONTINUE
```

```
        IP = IP - 256                ;RESET IP
        DO 220 I = IS,256
            X(I) = INTS(L)           ;LOAD UP X(I)
            L = L + 1
 220    CONTINUE
        CALL WRBLK(4,KS,X,1,IER)
        IF (IER.EQ.9) GO TO 1001     ;END OF FILE
        IF(IER.NE.1) TYPE " WRBLK ERROR ON FILE #2 ",IER
        IF(IP.EQ.0) GO TO 230
        DO 230 I = 1,IP
            X(I) = INTS(L)           ;RESTART LOAD UP OF X(I)
            L = L + 1
 230    CONTINUE
        KS = KS + 1                  ;INCREMENT BLOCK COUNT
 240    IS = IP+1
        NDONE = NDONE + 1
        GO TO 100

C***  SPEECH VOCODED
 1001   CONTINUE
        TYPE " NDONE = ",NDONE
        TYPE " SPEECH VOCODED "
C***  NORMALIZATION ROUTINE
        S1 = 0.0
        DO 700 J = 0,87
            CALL RDBLK(4,J,X,1,IER)
            IF(IER.EQ.9) GO TO 701
            IF(IER.NE.1) TYPE " RDBLK ERROR ON FILE #2 ",IER
            DO 600 I = 1,256
                N2 = IABS(X(I))
                IF(N2.GT.N5) N5 = N2      ; CHECK FOR MAXIMUM VALUE
 600        CONTINUE
            KEND = J
 700    CONTINUE
 701    CONTINUE
        TYPE" THE MAX VALUE FOUND WAS " ,N5
        S2 = 2000.0/FLOAT(N5)
        DO 800 J = 0,KEND
            CALL RDBLK(4,J,X,1,IER)
            IF(IER.NE.1) TYPE " READ BLOCK ERROR ",IER
            DO 750 I = 1,256
                S1 = FLOAT(X(I))*S2      ; NORMALIZE TO A MAX OF 2000
                X(I) = INT(S1)
 750        CONTINUE
            CALL WRBLK(4,J,X,1,IER)
            IF (IER.NE.1) TYPE " WRITE BLOCK ERROR ",IER
 800    CONTINUE
        IF (KEND.GE.87) GO TO 900    ;PRECAUTIONARY STEP TO AVOID OVER-
        DO 840 I = 1,256             ;      LOADING FILE #4
            X(I) = 0
 840    CONTINUE
        DO 850 J = KEND,87
            CALL WRBLK(4,J,X,1,IER)      ;SET ALL UNSET BLOCKS TO ZERO
 850    CONTINUE
C***  CLOSE FILES AND CHECK FOR ARITHMETIC ERRORS
 900    CALL CLOSE(4,IER)
        CALL CLOSE(3,JER)
        IF((IER.NE.1).OR.(JER.NE.1)) TYPE " CLOSE FILE ERROR ",IER,JER
        CALL DVDCHK(IDIV1)
        IF(IDIV1.EQ.1) TYPE " DIVIDE BY ZERO OCCURRED "
```

```
CALL OVERFL(IFLO1)
IF (IFLO1.EQ.1) TYPE " OVERFLOW OCCURRED "
IF (IFLO1.EQ.3) TYPE " UNDERFLOW OCCURRED "

STOP
END
```

```
C*********************************************************************
C
C    THIS SUBROUTINE CREATES A NORMAL RANDOM SEQUENCE WHICH WILL
C    BE A RANDOM NOISE INPUT TO THROAT.  THE OUTPUT OF THIS
C    PROGRAM IS AN ARRAY, U(I), OF LENGTH AS DETERMINED IN THE
C    CALLING ROUTINE.
C
C        NOTE: THIS MUST BE LINKED TOGETHER WITH DRAND.
C
C        PARAMETERS:  U(I)      : OUTPUT SEQUENCE
C                     FRMSIZ    : LENGTH OF ARRAY
C                     IX        : DOUBLE PRECISION SEED TO THIS ROUTINE
C                                 A NEW IX IS GENERATED BY THE PROGRAM
C                                 TO FEED THE NEXT ITERATION.
C                     DRAND(IX): A DOUBLE PRECISION FUNCTION -
C                                 GENERATES UNIFORM PDF.
C
C*********************************************************************
        SUBROUTINE UNVOCD(U,FRMSIZ,IX)

        DOUBLE PRECISION    U1,V,W,T,X,E,E2,E10,E3,Z,P,P1,F,P2,P3,P4,P5,PI
     X  ,PI2,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A12,A13,A14,A15,A16,A17,A18
        DOUBLE PRECISION INTEGER IX
        INTEGER FRMSIZ
        DIMENSION U(1)
        DATA A1/.884070402298758D0/,
     X       A2/1.131131635444180D0/,
     X       A3/.986655477086949D0/,
     X       A4/.958720824790463D0/,
     X       A5/.630834801921960D0/,
     X       A6/.755591531667601D0/,
     X       A7/.034240503750111D0/,
     X       A8/.911312780288703D0/,
     X       A9/.479727404222441D0/,
     X       A10/1.105473661022070D0/,
     X       A12/.872834976671790D0/,
     X       A13/.049264496373128D0/,
     X       A14/.595507138015940 1D0/,
     X       A15/.805577924423817D0/,
     X       A16/.053377549506886D0/,
     X       A17/.973310954173898D0/,
     X         E/2.216035867166471D0/,
     X       A18/.180025191068563D0/,

C*** CALCULATE THE NORMAL FUNCTION
        PI = 3.141592653 6D0
        PI2 = (PI*2.0)**(-.5)
        E2 = E**2.0
        E3 = E2/2.0
        DO 2200 N = 1,FRMSIZ
            U1 = DRAND(IX)
            IF(U1 .GT. A1)GO TO 1000
            V = DRAND(IX)
            X = E * (A2 * U1 + V - 1)
            GO TO 9000

1000        IF(U1 .LT. A17)GO TO 1200
1005        V = DRAND(IX)
            W = DRAND(IX)
            T = E3 - DLOG(W)
```

```
            E10 = T * V**2
            IF(E10 .GT. E3)GO TO 1005
            IF(U1 .LT. A3)GO TO 1010
            X = -(2.0 * T)**.5
            GO TO 9000

1010        X = (2.0 * T)**.5
            GO TO 9000

1200        IF(U1 .LT. A4)GO TO 1500
1300        V = DRAND(IX)
            W = DRAND(IX)
            Z = V - W
            T = E - A5 * DMIN1(V,W)
            P = DMAX1(V,W)
            IF(P .LT. A6)GO TO 1800
            P1 = A7 * DABS(Z)
            F = PI2*DEXP(-T*T/2.0)-A18*(E-DABS(T))
            IF(P1 .LE. F)GO TO 1800
            GO TO 1300
1500        IF(U1 .LE. A8)GO TO 1700
1600        V = DRAND(IX)
            W = DRAND(IX)
            Z = V - W
            T = A9 + A10 * DMIN1(V,W)
            P2 = DMAX1(V,W)
            IF(P2 .LE. A12)GO TO 1800
            P3 = A13 * DABS(Z)
            F = PI2 * DEXP(-T*T/2.0)-A18*(E-DABS(T))
            IF(P3 .LE. F)GO TO 1800
            GO TO 1600
1700        V = DRAND(IX)
            W = DRAND(IX)
            Z = V - W
            T = A9 - A14 * DMIN1(V,W)
            P4 = DMAX1(V,W)
            IF(P4 .LE. A15)GO TO 1800
            P5 = A16 * DABS(Z)
            F = PI2 * DEXP(-T*T/2.0)-A18*(E-DABS(T))
            IF(P5 .LE. F)GO TO 1800
            GO TO 1700
1800        IF(Z .LT. 0.0)GO TO 1900
            X = -T
            GO TO 9000

1900        X = T
9000        CONTINUE
            U(N) = SNGL(X)
2200    CONTINUE
        IF(FRMSIZ.GE.200) GO TO 2400
        DO 2300 I = FRMSIZ+1,200
            U(I) = 0.0
2300    CONTINUE
2400    RETURN
        END
```

```
C*********************************************************************
C
C       THIS FUNCTION IS A UNIFORM RANDOM NUMBER GENERATOR.
C
C*********************************************************************

      DOUBLE PRECISION FUNCTION DRAND(IX)
      DOUBLE PRECISION INTEGER IX,A,P,B15,B16,XHI,XALO,LEFTLO,FHI,K
      DATA A/16807D0/,B15/32768D0/,B16/65536D0/,P/2147483647D0/
      XHI = IX/B16
      XHI = XHI - DMOD(XHI,1D0)
      XALO = (IX - XHI * B16) * A
      LEFTLO = XALO/B16
      LEFTLO = LEFTLO - DMOD(LEFTLO,1D0)
      FHI = XHI * A + LEFTLO
      K = FHI/B15
      K = K - DMOD(K,1D0)
      IX = (((XALO-LEFTLO*B16)-P) + (FHI-K*B15)*B16)+K
      IF(IX.LT.0.D0)  IX = IX + P
      DRAND = IX * 4.656612875D-5 / 1.52304D0
      RETURN
      END
```

```
C*************************************************************************
C
C         THIS SUBROUTINE PRODUCES AN INPUT TO THE SYNTHESIS FILTER
C
C   GLOTTAL PULSE SHAPE  -      POLYNOMIAL FUNCTION
C
C
C                .                                   .
C             .     .                             .     .
C          .           .                       .           .
C       .                 .                 .                 .
C    .                       .             .                       .
C ---.-----------------.-----------------------.-----------------.-----------------
C    ^        ^       ^                       ^                 ^
C    ^       NP      NN                       ^                 ^
C    ^        ^       ^                       ^                 ^
C    ^<-TP->^<-TN->^                          ^                 ^
C    ^        ^       ^                       ^                 ^
C    ^                                        ^                 ^
C    ^                                        ^                 ^
C    ^<-----1 PITCH PERIOD------------------->^                 ^
C    ^                                        ^                 ^
C
C     INPUTS:
C        PPF:     THE PITCH PERIOD
C        SIZE:    THE FRAME SIZE
C
C     OUTPUTS:
C        U(I):    THE OUTPUT SEQUENCE NEEDED AS INPUT TO "THROAT."
C
C*************************************************************************
      SUBROUTINE VOICED1(U,PPF,SIZE)

      DIMENSION U(200)
      INTEGER PPF,SIZE

      NPOS =1
      TP = .030 * FLOAT(PPF)
      TN = .012 * FLOAT(PPF)
      NP = INT(TP)
      NN = INT(TP + TN)
      M=SIZE/PPF
      K = 0
      DO 60 J = 1,M
C***CALCULATE ONE FRAMES WORTH OF U
            TIME = 1.0
            DO 50 I=1,PPF
                K = K + 1
                IF(I .GT. NP)GO TO 20
                U(K) = (3.*(TIME/TP)**2) - (2.*(TIME/TP)**3)
                GO TO 40
 20             IF(I .GT. NN)GO TO 30
                U(K) = (1.-((TIME-TP)/TN)**2)
                GO TO 40
 30             U(K) = 0.0
 40             TIME = TIME + 1.0
 50         CONTINUE
 60   CONTINUE
      DO  70 I = SIZE+1,200
          U(I) = 0.                ; ZERO FILL THE REST OF THE ARRAY
 70   CONTINUE
```

```
RETURN
END
```

```
C*************************************************************************
C
C       THIS SUBROUTINE PRODUCES AN INPUT TO THE SYNTHESIS FILTER
C
C  GLOTTAL PULSE SHAPE - TRIGONOMETRIC
C
C          .                                   .
C        .   .                               .   .
C      .       .                           .       .
C    .           .                       .           .
C  .               .                   .               .
C  |               |                   |               |
C  |       NP      |       NN          |               |
C  |       |       |       |           |               |
C  |<-TP->|<-TN->|                     |               |
C  |       |       |       |           |               |
C  |                                   |               |
C  |                                   |               |
C  |<-----1 PITCH PERIOD-------------->|
C  |                                   |
C    INPUTS:
C       PPF:     THE PITCH PERIOD
C       SIZE:    THE FRAME SIZE
C
C    OUTPUTS:
C       U(I):     THE OUTPUT SEQUENCE NEEDED AS INPUT TO "THROAT."
C
C*************************************************************************
      SUBROUTINE VOICED2(U,PPF,SIZE)

      DIMENSION U(200)
      INTEGER PPF,SIZE

      PI = 3.14159
      PI2 = PI/2.0
      NPOS =1
      TP = .030 * FLOAT(PPF)
      TN = .012 * FLOAT(PPF)
      NP = INT(TP)
      NN = INT(TP + TN)
      M = SIZE/PPF              ; NUMBER OF PITCH PERIODS PER FRAME
      K = 0
      DO 60 J = 1,M
C***  CALCULATE ONE FRAME OF U(I)
          TIME = 1.0
          DO 50 I = 1,PPF
              K = K + 1
              IF(I .GT. NP)GO TO 20
              U(K) = (.5)*(1.0-COS(TIME*PI/TP))
              GO TO 40
 20           IF(I .GT. NN)GO TO 30
              U(K) = (.5)*COS(((TIME-TP)/TN)*PI2)
              GO TO 40
 30           U(K) = 0.0
 40           TIME = TIME + 1.0
 50       CONTINUE
 60   CONTINUE
      DO 70 I = SIZE+1,200
          U(I) = 0.0
 70   CONTINUE
```

```
RETURN
END
```

```
C*************************************************************************
C
C       THIS SUBROUTINE PRODUCES AN INPUT TO THE SYNTHESIS FILTER
C
C  GLOTTAL PULSE SHAPE  -   IMPULSE (TP=1,TN=0)
C
C
C             .                                    .
C          .     .                              .     .
C        .         .                          .         .
C       .           .                        .           .
C  ____.____________.___________________.____________.  ______________
C     ^      ^      ^                   ^
C     ^      NP     NN                  ^
C     ^      ^      ^                   ^
C     ^<-TP->^<-TN->^                   ^
C     ^      ^      ^                   ^
C     ^                                 ^
C     ^                                 ^
C     ^<-----1 PITCH PERIOD----------->^
C     ^                                 ^
C
C     INPUTS:
C        PPF:   THE PITCH PERIOD
C        SIZE:  THE FRAME SIZE
C
C     OUTPUTS:
C        U(I):    THE OUTPUT SEQUENCE NEEDED AS INPUT TO "THROAT."
C
C*************************************************************
        SUBROUTINE VOICED3(U,PPF,SIZE)

        DIMENSION U(200)
        INTEGER PPF,SIZE

        TIME = 1.0
        G = 1.0
        U(1) = G
        NPF2 = PPF+2
        DO 400 K = 2,PPF
            U(K) = 0.0
 400    CONTINUE
        U(PPF+1) = G
        DO 500 K = NPF2,SIZE
            U(K) = 0.0
 500    CONTINUE
        RETURN
        END
```

```
C*****************************************************************
C
C    THIS SUBROUTINE INPUTS U (A SEQUENCE OF VOICED/UNVOICED
C    INPUTS) AND PASSES IT THROUGH A TIME VARYING DIGITAL FILTER
C    TO PRODUCE AN OUTPUT SPEECH SEQUENCE.
C
C    INPUTS:
C        U(I):    SEQUENCE GENERATED BY EITHER "VOICED" OR "UNVOCD."
C                 EITHER A PULSE AT THE PITCH PERIOD OR RANDOM NOISE.
C        ICOUNT:  THE FRAME LENGTH
C        FILTER:  THE FILTER COEFFICIENTS
C        NORDER:  THE ORDER OF THE FILTER
C        GAIN1:   THE GAIN OF THE FILTER, AL IN THE "VOCODE."
C        W(I):    THE MEMORY OF THE FILTER
C
C    OUTPUTS:
C        W(I):    THE MEMORY OF THE FILTER
C        S(I):    THE OUTPUT SPEECH SEQUENCE
C
C*****************************************************************

        SUBROUTINE THROAT(U,ICOUNT,FILTER,NORDER,GAIN1,S,W)

        DIMENSION U(1),FILTER(1),W(0:20),S(1)

        DO 500 N=1,ICOUNT
          TOTAL = 0.0
          DO 400 K=1,NORDER
            TOTAL = TOTAL - W(K) * FILTER(K)
 400      CONTINUE
          W(0) = TOTAL + GAIN1 * U(N)
          S(N) = W(0)
          DO 450 I=1,NORDER
            J = NORDER + 1 - I
            W(J) = W(J-1)
 450      CONTINUE
          W(0) = 0.0
 500    CONTINUE
        IF(ICOUNT.GE.200) GO TO 1000
        DO 600 I = ICOUNT+1,200
            S(I) = 0.0
 600    CONTINUE
1000    RETURN
        END
```

```
C*********************************************************************
C
C       PROGRAM:          SETUP
C       AUTHOR:           WILL JANSSEN / REVISED BY C MCKOWN
C       DATE:             17 APRIL 83  / ON 2 SEPT 83
C       LANGUAGE:         FORTRAN5
C       FUNCTION:         THIS PROGRAM ALLOWS THE USER TO SETUP A FILE THAT
C                         CONTAINS   INFORMATION REQUIRED TO RUN THE
C                         LINEAR PREDICTIVE CODER WRITTEN BY CRAIG MCKOWN.
C                         THE PROGRAM WILL ALLOW THE USER THE FOLLOWING
C                         OPTIONS.
C                         1) CREATE A NEW FILE
C                         2) UPDATE AN OLD FILE
C                         3) PRINT PARAMETERS
C
C       LOAD COMMAND LINE: RLDR SETUP @FLIB@
C
C       NOTE:             1) THE ARRAYS ARE SET TO MAX OF 10 VARIABLES
C                            EACH.
C                         2) THE REAL ARRAY IS CALLED RELVAR AND THE
C                            INTEGER ARRAY IS CALLED INTVAR
C
C*********************************************************************
C
C       SETUP
C
C*********************************************************************

        DIMENSION RELVAR(10),INTVAR(15),OUTFILE(7)
        INTEGER YES,YES2,SIZER,SIZEI,YES5
C***  SIZER=REAL ARRAY SIZE,    SIZEI=INTEGER ARRAY SIZE
        SIZER = 10
        SIZEI = 15
C*** NEW OR OLD FILE ***
        TYPE "THIS PROGRAM CREATES OR UPDATES A DECISION VARIABLE FILE."
        TYPE "ARE YOU UPDATING AN OLD FILE?"
        ACCEPT"(1-YES,0-NO)",YES

C*** GET FILE NAME ***
 20     ACCEPT"FILE NAME? "
        READ(11,39)OUTFILE(1)
 39     FORMAT(S13)
        IF (YES.EQ.1)GO TO 30
        CALL DFILW(OUTFILE,JER)
        IF(JER.NE.13) TYPE "YOU DELETED A CURRENT FILE!"
        IF((JER.NE.1).AND.(JER.NE.13)) TYPE "DELETE FILE ERROR",JER
        CALL CFILW(OUTFILE,2,JER)
        IF (JER.NE.1) TYPE "CREATE FILE ERROR!",JER
 30     CALL OPEN(1,OUTFILE,3,IER)
        IF(IER .NE. 1)TYPE"OPEN ERROR ",IER

C*** INITIALIZE THE ARRAYS,NEW FILES-SET = TO 0,OLD FILES-READ IN
C    OLD FILES***
        IF (YES.EQ.1)GO TO 50
        DO 45 I=1,SIZER
 45     RELVAR(I) = 0.0
        DO 47 I=1,SIZEI
 47     INTVAR(I) = 0
        GOTO 60
 50     READ (1,901)(RELVAR(I),I=1,SIZER)
```

```
      READ (1,902)(INTVAR(I),I=1,SIZEI)

C*** UPDATE ARRAYS ***
 60   CONTINUE
      TYPE" <CR>
     X    IF YOU CHOOSE TO CHANGE A VARIABLE ENTER : Y <CR>
     X    OTHERWISE ENTER ANOTHER LETTER   <CR> "

      TYPE " "
      TYPE"CURRENT VALUE OF ACCEPT/NOT ACCEPT (A-0,NA-1): ",INTVAR(1)
      TYPE"CHANGE VALUE?   "
      CALL RCHAR(ICHAR,IER)
      IF(ICHAR.NE.89)GO TO 5000
      ACCEPT" <CR> INPUT NEW VALUE : ",INTVAR(1)

5000  TYPE " "
      TYPE"CURRENT NUMBER OF POLES IS : ",INTVAR(2)
      TYPE"CHANGE VALUE?   "
      CALL RCHAR(ICHAR,IER)
      IF(ICHAR.NE.89)GO TO 5001
      ACCEPT" <CR> INPUT NEW VALUE: ",INTVAR(2)

5001  TYPE " "
      TYPE"THE METHOD OF PREDICTION IS "
      TYPE"(AUTO-0,COVAR-1): ",INTVAR(3)
      TYPE"CHANGE VALUE?   "
      CALL RCHAR(ICHAR,IER)
      IF(ICHAR.NE.89)GO TO 5002
      ACCEPT"<CR> INPUT NEW VALUE: ",INTVAR(3)

5002  TYPE " "
      TYPE"CURRENT VALUE:NO. OF POINTS/SET (MAXFR): ",INTVAR(4)
      TYPE"CHANGE VALUE?   "
      CALL RCHAR(ICHAR,IER)
      IF(ICHAR.NE.89)GO TO 5003
      ACCEPT"<CR> INPUT NEW VALUE: ",INTVAR(4)

5003  TYPE " "
      TYPE"THE CURRENT VALUE OF FILTER SPACINGS IS (MAXPT): ",INTVAR(5)
      TYPE"CHANGE VALUE?   "
      CALL RCHAR(ICHAR,IER)
      IF(ICHAR.NE.89)GO TO 5004
      ACCEPT"<CR> INPUT NEW VALUE: ",INTVAR(5)

5004  TYPE " "
      TYPE"THE CURRENT VALUE OF PRE/DE-EMP (1-Y,0-N) IS: ",INTVAR(6)
      TYPE"CHANGE VALUE?   "
      CALL RCHAR(ICHAR,IER)
      IF(ICHAR.NE.89)GO TO 5005
      ACCEPT"<CR> INPUT NEW VALUE: ",INTVAR(6)

5005  TYPE " "
      TYPE"THE CURRENT VALUE OF GLOTTAL SHAPE IS  "
      TYPE"(1-POLYNOMIAL,3-IMPULSE)   : ",INTVAR(7)
      TYPE"CHANGE VALUE?   "
      CALL RCHAR(ICHAR,IER)
      IF(ICHAR.NE.89)GO TO 5006
      ACCEPT"<CR> INPUT NEW VALUE: ",INTVAR(7)

5006  TYPE " "
```

```
      TYPE"THE CURRENT VALUE OF HAMMING WINDOW (0-NO, 1-YES): ", INTVAR(8)
      TYPE"CHANGE VALUE?    "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR. NE. 89)GO TO 5007
      ACCEPT"<CR> INPUT NEW VALUE: ", INTVAR(8)

5007  TYPE " "
      TYPE"THE METHOD OF PITCH DETECTION IS "
      TYPE"(SIFT-0, AUTOC-1): ", INTVAR(9)
      TYPE"CHANGE VALUE?    "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR. NE. 89)GO TO 5008
      ACCEPT"<CR> INPUT NEW VALUE: ", INTVAR(9)

5008  TYPE " "
      TYPE"PITCH DET'N AND COEF. CAL'N FROM SAME FILE?"
      TYPE"CURRENT VALUE (1-Y, 0-N): ", INTVAR(10)
      TYPE"CHANGE VALUE?    "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR. NE. 89)GO TO 5010
      ACCEPT"<CR> INPUT NEW VALUE: ", INTVAR(10)

5010  TYPE " "
      TYPE"THE CURRENT VALUE OF VOICED/UN THRESH IS: ", RELVAR(1)
      TYPE"CHANGE VALUE?    "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR. NE. 89)GO TO 5011
      ACCEPT"<CR> INPUT NEW REAL VALUE: ", RELVAR(1)

5011  TYPE " "
      TYPE"CURRENT VALUE OF SPEECH SCALE-(IN CODER): ", RELVAR(2)
      TYPE"CHANGE VALUE?    "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR. NE. 89)GO TO 5015
      ACCEPT"<CR> INPUT NEW REAL VALUE: ", RELVAR(2)

5015  TYPE " "
      TYPE"CURRENT VALUE OF SILENCE THRESH-(IN ENER)IS: ", RELVAR(3)
      TYPE"CHANGE VALUE?    "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR. NE. 89)GO TO 5016
      ACCEPT"<CR> INPUT NEW REAL VALUE: ", RELVAR(3)

5016  TYPE " "
      TYPE"CURRENT VALUE OF UNVOICED GAIN FACTOR IS: ", RELVAR(4)
      TYPE"CHANGE VALUE?    "
      CALL RCHAR(ICHAR, IER)
      IF(ICHAR. NE. 89)GO TO 5020
      ACCEPT"<CR> INPUT NEW REAL VALUE: ", RELVAR(4)

5020  CONTINUE

C*** TYPE ARRAY ***
      TYPE"THE ARRAYS HAVE BEEN LOADED"
      ACCEPT"DO YOU WANT TO HAVE THE ARRAY TYPED(1-YES, 0-NO): ", YES
      IF(YES .EQ. 0)GO TO 200
      TYPE" ACCEPT/NOT ACCEPT:                    ", INTVAR(1)
      TYPE" NUMBER OF POLES:                      ", INTVAR(2)
      TYPE" METHOD (0-AUTO, 1-COVAR, ):           ", INTVAR(3)
      TYPE" MAXFR:                                ", INTVAR(4)
```

```
      TYPE" MAXPT:                                    ",INTVAR(5)
      TYPE" PRE/DE-EMP (1-Y,0-N):                     ",INTVAR(6)
      TYPE" GLOT (1-POLYNOMIAL,3-IMPULSE):            ",INTVAR(7)
      TYPE" HAMMING WINDOW? (1-Y,0-N):                ",INTVAR(8)
      TYPE" METHOD PITCH DET (0-SIFT,1-AUTOC):        ",INTVAR(9)
      TYPE" PITCH & COEF'S SAME FILE(1-Y,0-N):        ",INTVAR(10)
      TYPE"VOICED/UN THRESHOLD:                       ",RELVAR(1)
      TYPE"SPEECH SCALE-(IN CODER):                   ",RELVAR(2)
      TYPE"SILENCE THRESHOLD                          ",RELVAR(3)
      TYPE"UNVOICED GAIN FACTOR                       ",RELVAR(4)

C*** OUTPUT FILE ***

200   TYPE "WRITE DECISION VARIABLES TO SAME FILE?"
      ACCEPT "(1-YES,0-NO): ",YES2
      IF (YES2 .EQ. 1)GO TO 75
      CALL CLOSE(1,IER)
      IF (IER .NE. 1)TYPE"CLOSE FILE ERROR1 ",IER
      ACCEPT"FILE NAME? "
      READ(11,69)OUTFILE(1)
 69   FORMAT(S13)
      CALL DFILW(OUTFILE,JER)
      IF(JER.EQ.13) TYPE "YOU DELETED A CURRENT FILE!"
      IF((JER.NE.1).AND.(JER.NE.13)) TYPE "DELETE FILE ERROR",JER
      CALL CFILW(OUTFILE,2,JER)
      IF (JER.NE.1) TYPE "CREATE FILE ERROR!",JER
 70   CALL OPEN(1,OUTFILE,3,IER)
      IF(IER .NE. 1)TYPE"OPEN ERROR ",IER
 75   CALL REWIND(1)
      WRITE (1,901)(RELVAR(I),I=1,SIZER)
      WRITE (1,902)(INTVAR(I),I=1,SIZEI)
      CALL CLOSE(1,IER)
      IF (IER .NE. 1)TYPE"CLOSE FILE ERROR2 ",IER
      ACCEPT"PRINT ARRAY ON PRINTRONICS?(1-Y,0-N) ",YES
      IF(YES .EQ. 0)GO TO 1001
      WRITE(12,1499)OUTFILE(1)
      CALL FGDAY(IMON,IDAY,IYEAR)
      CALL FGTIME(IHOUR,IMIN,ISEC)
      WRITE (12,1311)IDAY,IMON,IYEAR
      WRITE (12,1312)IHOUR,IMIN,ISEC
 1311 FORMAT("0","DATE :  ",1X,I2,"/",I2,"/",I2)
 1312 FORMAT("0","TIME :  ",1X,I2,":",I2,":",I2)
      WRITE(12,1500)INTVAR(1)
      WRITE(12,1501)INTVAR(2)
      WRITE(12,1502)INTVAR(3)
      WRITE(12,1503)INTVAR(4)
      WRITE(12,1504)INTVAR(5)
      WRITE(12,1505)INTVAR(6)
      WRITE(12,1506)INTVAR(7)
      WRITE(12,1507)INTVAR(8)
      WRITE(12,1508)INTVAR(9)
      WRITE(12,1509)INTVAR(10)
      WRITE(12,1600)RELVAR(1)
      WRITE(12,1601)RELVAR(2)
      WRITE(12,1602)RELVAR(3)
      WRITE(12,1603)RELVAR(4)
1499  FORMAT(1X,S13)
1500  FORMAT("0","  ACCEPT/NOT ACCEPT                 ",I6)
1501  FORMAT("0","  NUMBER OF POLES                   ",I6)
1502  FORMAT("0","  METHOD (0-AUTO,1-COVAR)           ",I6)
```

```
1503    FORMAT("0","  MAXFR                                          ",I6)
1504    FORMAT("0","  MAXPT                                          ",I6)
1505    FORMAT("0","  PRE/DE-EMP? (1-Y,0-N)                          ",I6)
1506    FORMAT("0","  GLOTTAL PULSE (1-POLY ,3-IMPULSE)              ",I6)
1507    FORMAT("0","  HAMMING WINDOW? (1-Y,0-N)                      ",I6)
1508    FORMAT("0","  METHOD PITCH DET (0-SIFT,1-AUTOC)              ",I6)
1509    FORMAT("0","  PITCH & COEF'S F'M SAME FILE(1-Y,0-N)          ",I6)
1600    FORMAT("0","  VOICED/UNVOICED THRESHOLD                    ",F12.5)
1601    FORMAT("0","  SPEECH SCALE                                 ",F12.5)
1602    FORMAT("0","  SILENCE THRESHOLD                            ",F12.5)
1603    FORMAT("0","  UNVOICED GAIN FACTOR                         ",F12.5)
900     FORMAT(3X,'TEST1 : ',F10.5)
901     FORMAT(3X,F12.5)
902     FORMAT(3X,I10)
1000    TYPE"PROGRAM COMPLETED"
1001    STOP
        END
```

```
C*************************************************************************
C
C       PROGRAM SCALE.FR
C
C       THIS PROGRAM SCALES SPEECH FILES SO THAT THERE IS A MAX VALUE
C       OF 1900 AND CAN DE-EMPHASIZE SPEECH
C
C       INPUT: MUST BE A BLOCKED FILE
C
C*************************************************************************

        DIMENSION S1(256),U(256)
        DOUBLE PRECISION IX
        INTEGER OUTFILE(7),INFILE(7),FILUFD(18),SPEECH(256)
        IX = DBLE(203)
        FLIP = 1.0
        NNEWS = 0
        ACCEPT"WARNING: THE INPUT FILE MUST BE AN INTEGER FILE <CR>
     X                  AND BE IN BLOCKED FORM. <CR> <CR>
     X          DO YOU WISH TO CONTINUE?(1-Y,0-N) ",NYZ
        IF(NYZ .EQ. 0)GO TO 60
        ACCEPT"INPUT FILENAME :"
        READ(11,39)INFILE(1)
39      FORMAT(S13)
        OPEN 1,INFILE,ATT="CI",ERR=40
        FLIP = 1.0
        ACCEPT"OUTPUT FILENAME :"
        READ(11,39)OUTFILE(1)
        OPEN 2,OUTFILE,ERR=50
        NDE = 0
        ACCEPT"OUTPUT FILE SIZE? ",ISIZE
        ACCEPT"PERFORM NOISE ADDITION?(1-Y,0-N)",NNOIS
        IF(NNOIS.EQ.0)GO TO 53
        ACCEPT"SIZE OF MAX NOISE?(REAL) ",VNOSIZ
53      CONTINUE
        MBLOCK = 1
        N15 = 0
        NV = 0
70      N6 = 0
        DO 80 I=1,256
        S1(I) = 0.0
        U(I) = 0.0
80      CONTINUE
        N5 = 0
100     CONTINUE
        CALL RDBLK(1,NV,SPEECH,MBLOCK,IENDS)
        IF(NNOIS.EQ.0) GO TO 110
        CALL UNVOCD(U,256,IX)
 110    DO 200 J=1,256
          IF(NNOIS.EQ.0)GO TO 120
          NNEWS = INT(U(J)*VNOSIZ/2)
            IF((J.EQ.1).AND.(NV.EQ.1)) TYPE " NOISE ADDED "
120       SPEECH(J) = SPEECH(J) + NNEWS
180       N2 = IABS(SPEECH(J))
          IF(N2 .GT. N5) N5 = N2
          N6 = N6 + 1
200     CONTINUE
        CALL WRBLK(2,NV,SPEECH,MBLOCK,IENDS)
        NV = NV + 1
        IF(NV .LT. ISIZE)GO TO 100
```

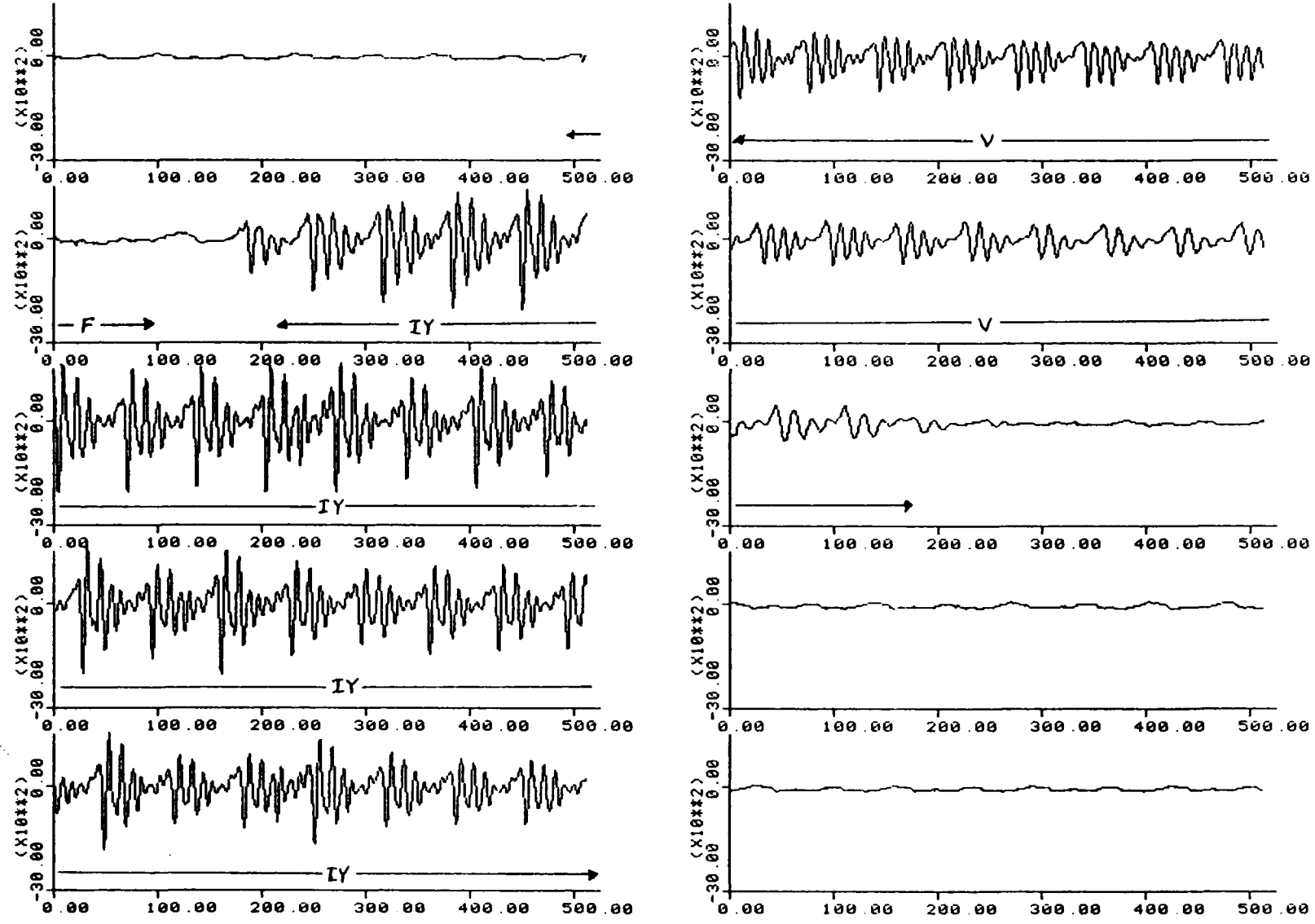
```
500     TYPE"THE FOLLOWING NO. OF POINTS WHERE CHECKED ",N6
        TYPE"AND THE MAX. VALUE FOUND WAS ",N5
        S2 = 1900.0 / FLOAT(N5) * FLIP
        N6 = 0
        NV = 0
600     CALL RDBLK(2,NV,SPEECH,MBLOCK,IENDS)
        DO 700 J=1,256
          S1(J) = FLOAT(SPEECH(J)) * S2
          SPEECH(J) = INT(S1(J))
700     CONTINUE
        CALL WRBLK(2,NV,SPEECH,MBLOCK,IER)
        N6 = N6 + 1
        NV = NV + 1
        IF(NV .LT. ISIZE)GO TO 600
900     CONTINUE
        N15 = N6 * 256
        TYPE"THE FOLLOWING NO. OF POINTS WERE OUTPUT ",N15
        CALL CLOSE(1,IER)
        IF(IER .NE. 1)TYPE"CLOSE ERROR ON INPUT ",IER
        CALL CLOSE(2,IER)
        IF(IER .NE. 1)TYPE"CLOSE ERROR ON OUTPUT ",IER
        TYPE"BLOCKS PROCESSED: ",N6
        GO TO 60
50      TYPE"OPEN ERROR ON OUTPUT "
        GO TO 60
40      TYPE"OPEN ERROR ON INPUT "
60      STOP
        END
```

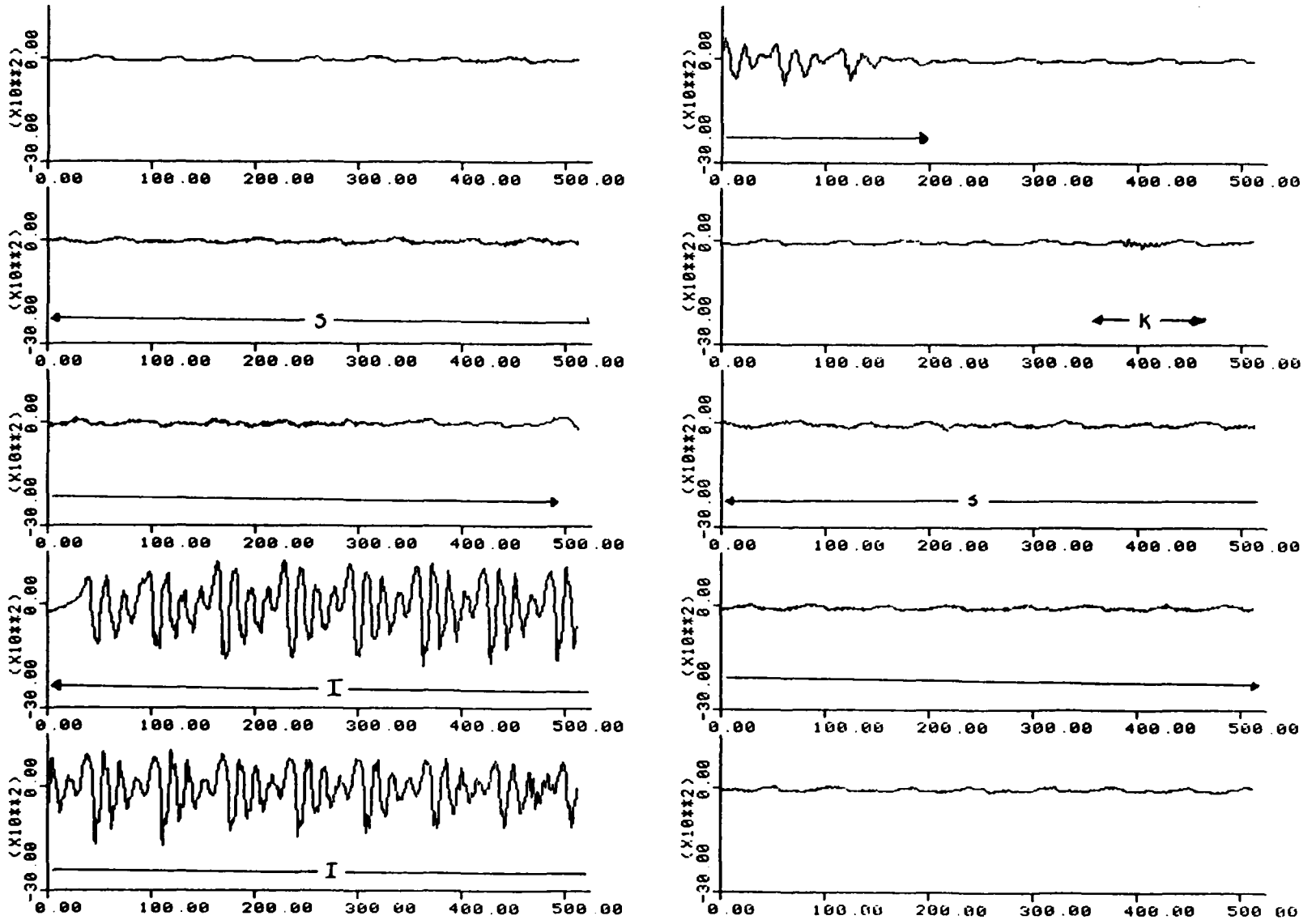
```
C*****************************************************************
C
C       LOAD LINE:RLDR TSTRND DRAND UNVOCD PLOT10 PLOT5.LB
C                 GRPH.LB @FLIB@
C
C       THIS PROGRAM RUNS EITHER THE UNIFORM OR NORMAL GENERATOR
C       AND PROVIDES A PLOT(PRINTRONIX OR TEKTRONIX) AND/OR THE
C       MEAN AND VARIANCE.
C
C*****************************************************************

        DIMENSION IT(500),U(256),T(500),XHOR(128),YVER(128),W(256)
        DOUBLE PRECISION INTEGER IX
        INTEGER FRMSIZ,NAME1(7),NAME2(7)
        ACCEPT"HOW MANY 256 POINT SETS? ",NUM

        NUFRM = NUM * 256
        IX = DBLE(203)
        DO 50 I = 1,256
            U(I) = DBLE(0.0)
            W(I) = DBLE(0.0)
 50     CONTINUE
        DO 100 I=1,500
          IT(I) = 0
 100    CONTINUE
        ICOUNT = 0
        SUM1 = 0.0
        SUM2 = 0.0
        K = 0
        ACCEPT"CHOOSE RANDOM GENERATOR(1-NORMAL,0-UNIFORM) ",NORM
        IF(NORM .EQ. 1)GO TO 1200
        DO 1000 NTIM=1,NUM
          DO 900 MTIM = 1,128
            ICOUNT = 2 + ICOUNT
            PEMP = SNGL(DRAND(IX))
            TEMP = PEMP * 500.
            SUM1 = SUM1 + TEMP
            SUM2 = SUM2 + (TEMP)**2
            ITEMP = INT(TEMP)
            IF((ITEMP .GT. 500).OR.(ITEMP .LT. 0))GO TO 600
            XHOR(MTIM) =  TEMP
            IT(ITEMP) = IT(ITEMP) + 1
            GO TO 800
 600        TYPE"DATA EXCEEDS BOUNDARY AT ",ITEMP
 800        PEMP = SNGL(DRAND(IX))
            TEMP = PEMP * 500.
            SUM1 = SUM1 + TEMP
            SUM2 = SUM2 + (TEMP)**2
            ITEMP = INT(TEMP)
            IF((ITEMP.GT.500).OR.(ITEMP.LT.0)) GO TO 850
            YVER(MTIM) =  TEMP
            IT(ITEMP) = IT(ITEMP) + 1
            GO TO 900
 850        TYPE " DATA EXCEEDS BOUNDARY AT ",ITEMP
 900      CONTINUE
          IF ((IER.NE.1).OR.(JER.NE.1)) TYPE " WRBLK ERROR ",IER,JER
          K = K + 1
1000    CONTINUE
        TYPE "PRODUCED UNIFORM DISTRIBUTION "
        GO TO 5000
```

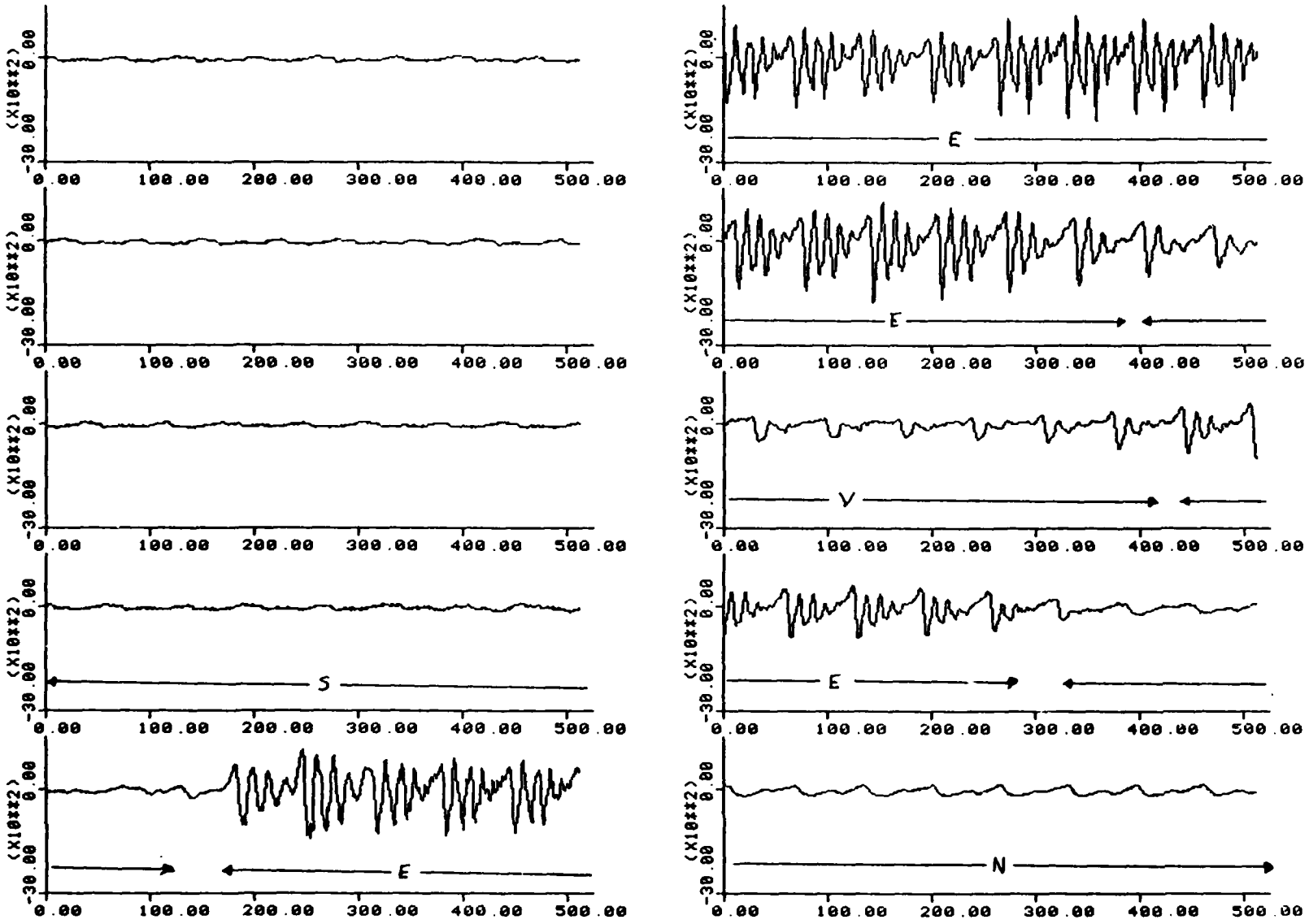
```
1200     FRMSIZ = 256
         DO 3000 NTIM=1,NUM
           CALL UNVOCD(U,FRMSIZ,IX)
           CALL UNVOCD(W,FRMSIZ,IX)
           DO 2500 NR=1,256
             ITEMP = INT(U(NR) *  80.0)
             ITEMP = ITEMP + 250          ;CENTERING FOR PLOTS
             IF((ITEMP .GT. 500).OR.(ITEMP .LE. 0))GO TO 1400
             SUM1 = SUM1 + FLOAT(ITEMP)
             SUM2 = SUM2 + FLOAT(ITEMP)**2
             XHOR(NR) = ITEMP
             IT(ITEMP) = IT(ITEMP) + 1
             ICOUNT = ICOUNT + 1
             GO TO 1600
1400         CONTINUE
1600         ITEMP = INT(W(NR) * 80.0)
             ITEMP = ITEMP + 250          ;CENTERING FOR PLOTS
             IF((ITEMP .GT. 500).OR.(ITEMP.LT.0)) GO TO 2000
             SUM1 = SUM1 + FLOAT(ITEMP)
             SUM2 = SUM2 + FLOAT(ITEMP)**2
             IT(ITEMP) = IT(ITEMP) + 1
             ICOUNT = ICOUNT + 1
             GO TO 2500
2000         CONTINUE
2500       CONTINUE
           K = K + 1
3000     CONTINUE
         TYPE "PRODUCED NORMAL DISTRIBUTION "
5000     CONTINUE
         DO 4000 K = 1,500
             T(K) = FLOAT(IT(K))
4000     CONTINUE
         XMEAN = SUM1/ICOUNT
         XMEAN2 = XMEAN**2
         VAR = SUM2/ICOUNT - XMEAN2
         TYPE "VAR = ",VAR
         STDEV = SQRT(VAR)

C*******PLOTS******
         ACCEPT"DO YOU WANT A PLOT?(1-Y,0-N) ",NYES
         IF(NYES.NE.1) GO TO 5600
         ACCEPT"USE PRINTRONICS PLOTTER?(1-Y,0-N) ",NO
         IF(NO.EQ.0) GO TO 5500
         NP = 1
         SF = 1.0
         NPTS = 500
         CALL PLOT10(T,NPTS,NP,XO,YO,SF)
         NP = 10
         CALL PLOT10(T,NPTS,NP,XO,YO,SF)
         GO TO 5600
5500     IFSCL = 0
         MODE = 0
         NG = 1
         N = 500
         CALL GRPH2("DENSITY",NG,T,U,N,MODE,YM,YA,IFSCL)
5600     CONTINUE
         TYPE " MEAN = ",XMEAN," STDEV = ",STDEV

         STOP
         END
```

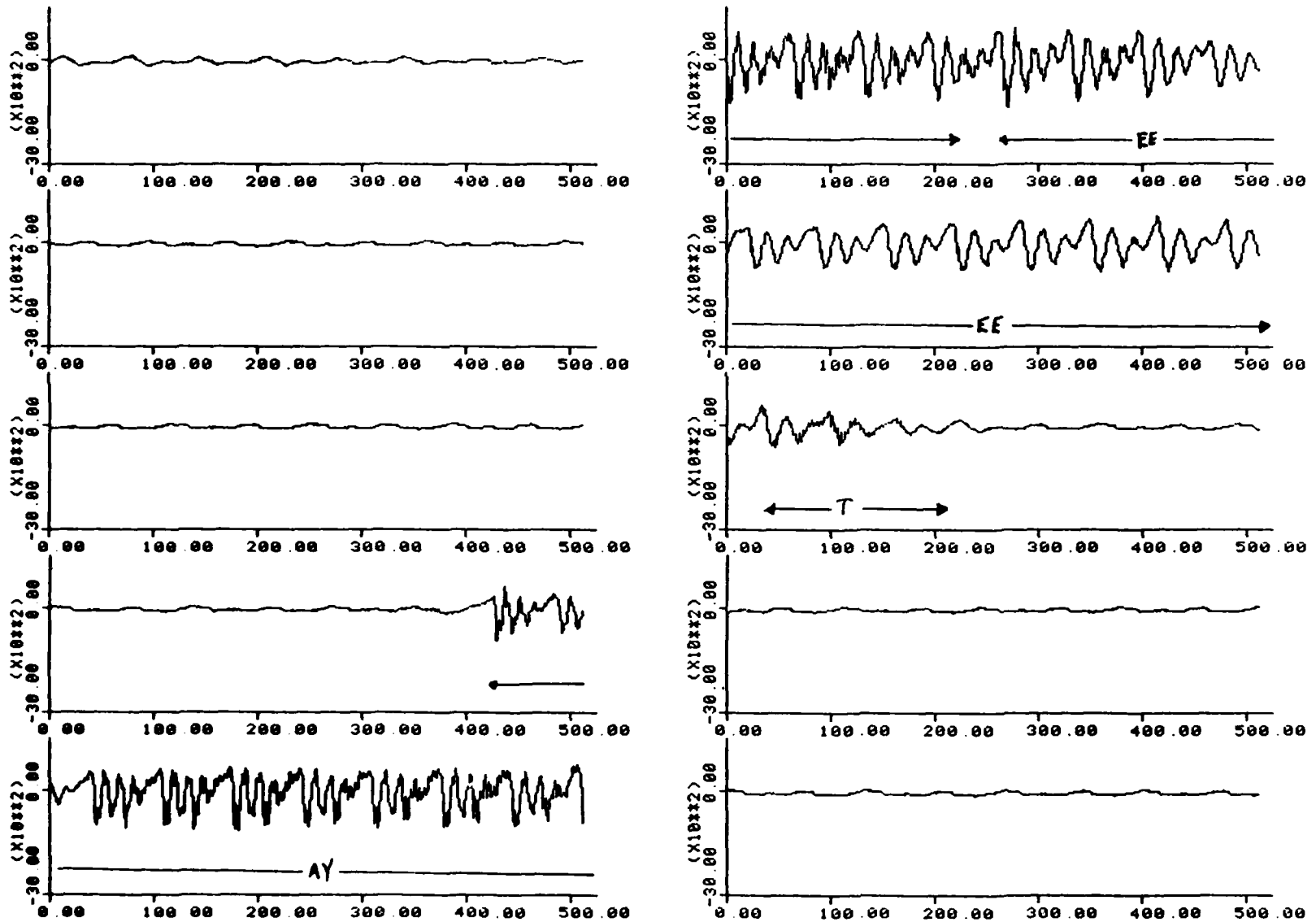
APPENDIX C

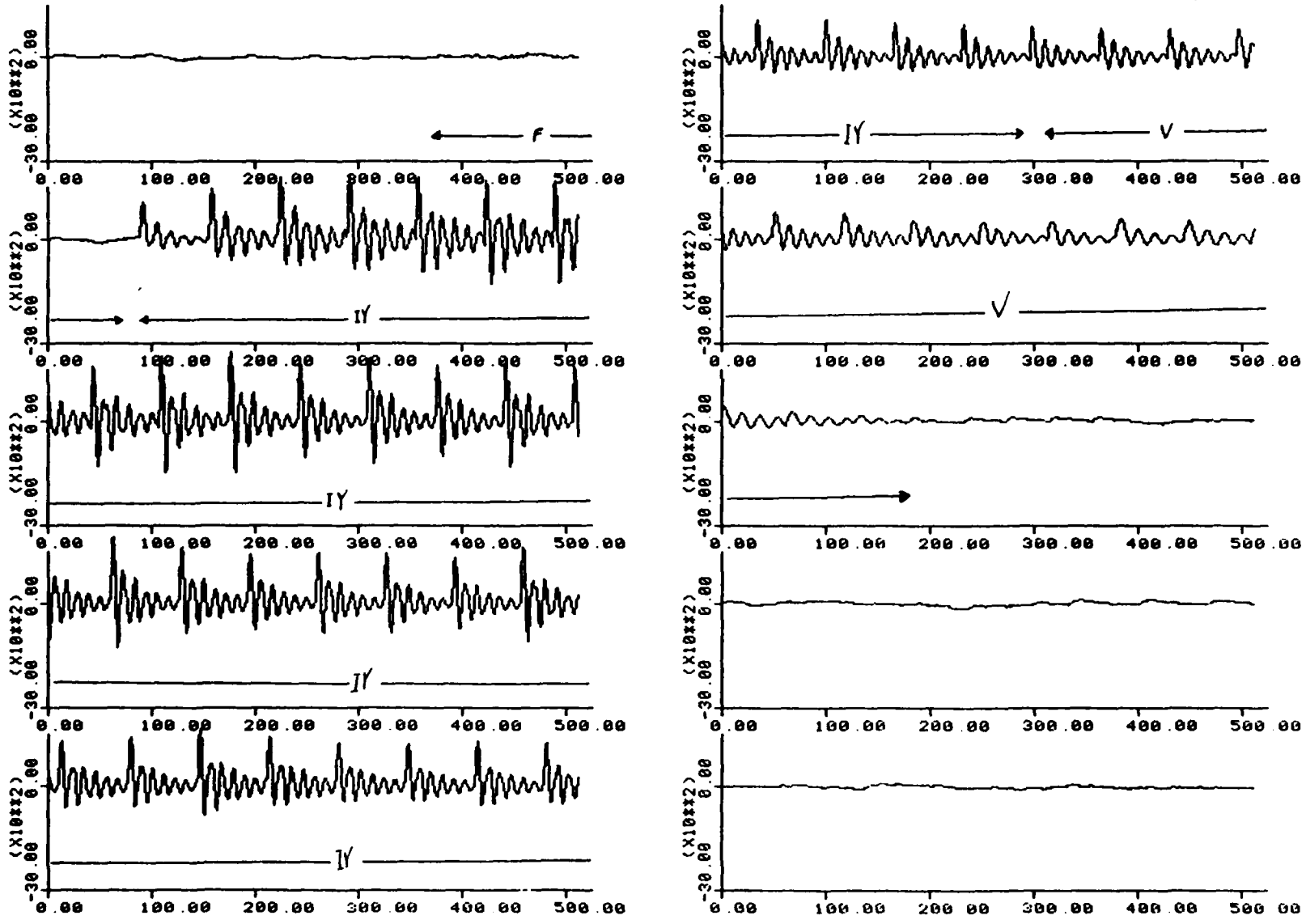
Waveshapes of the Speech Signals

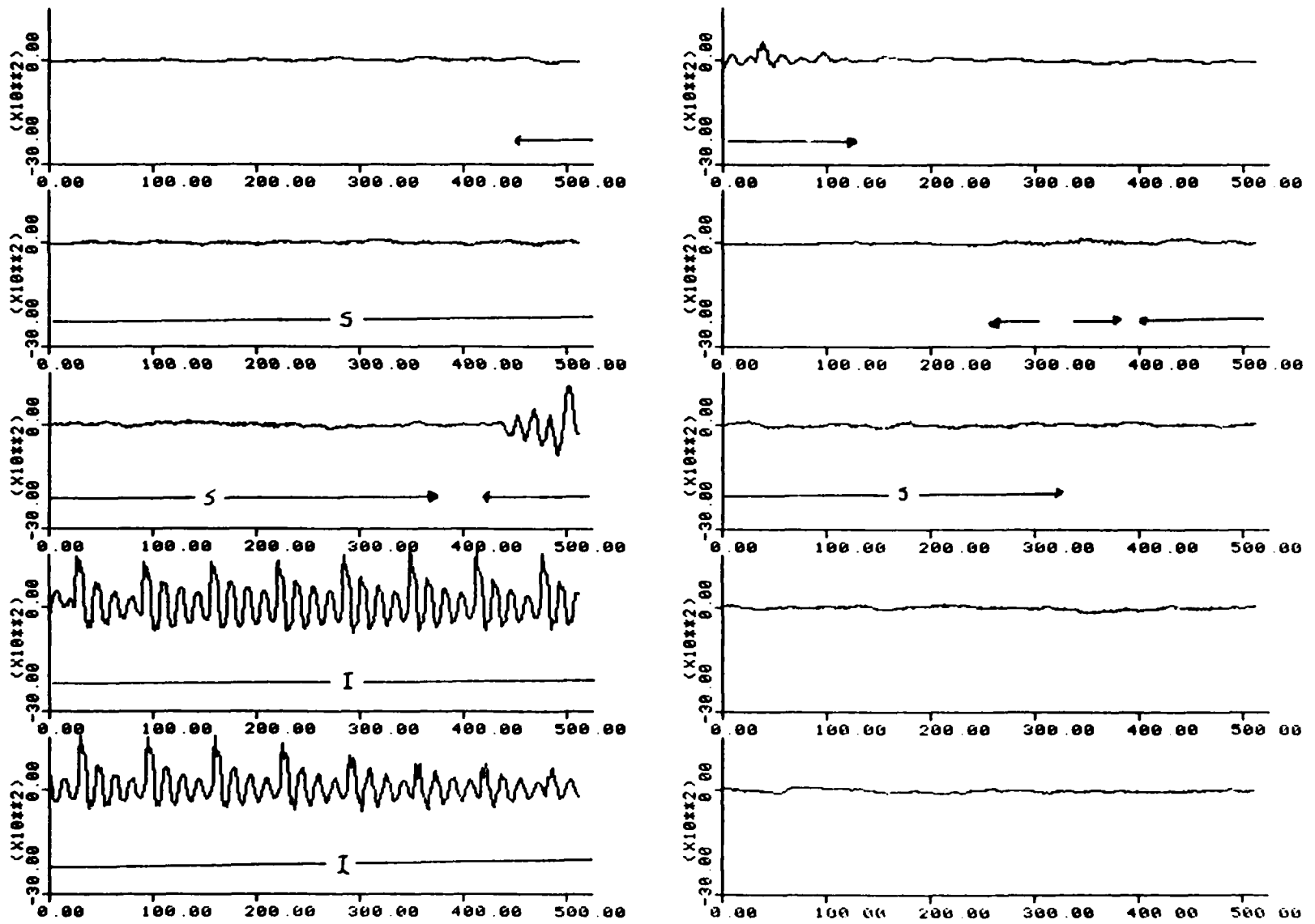

(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
F
IY
V

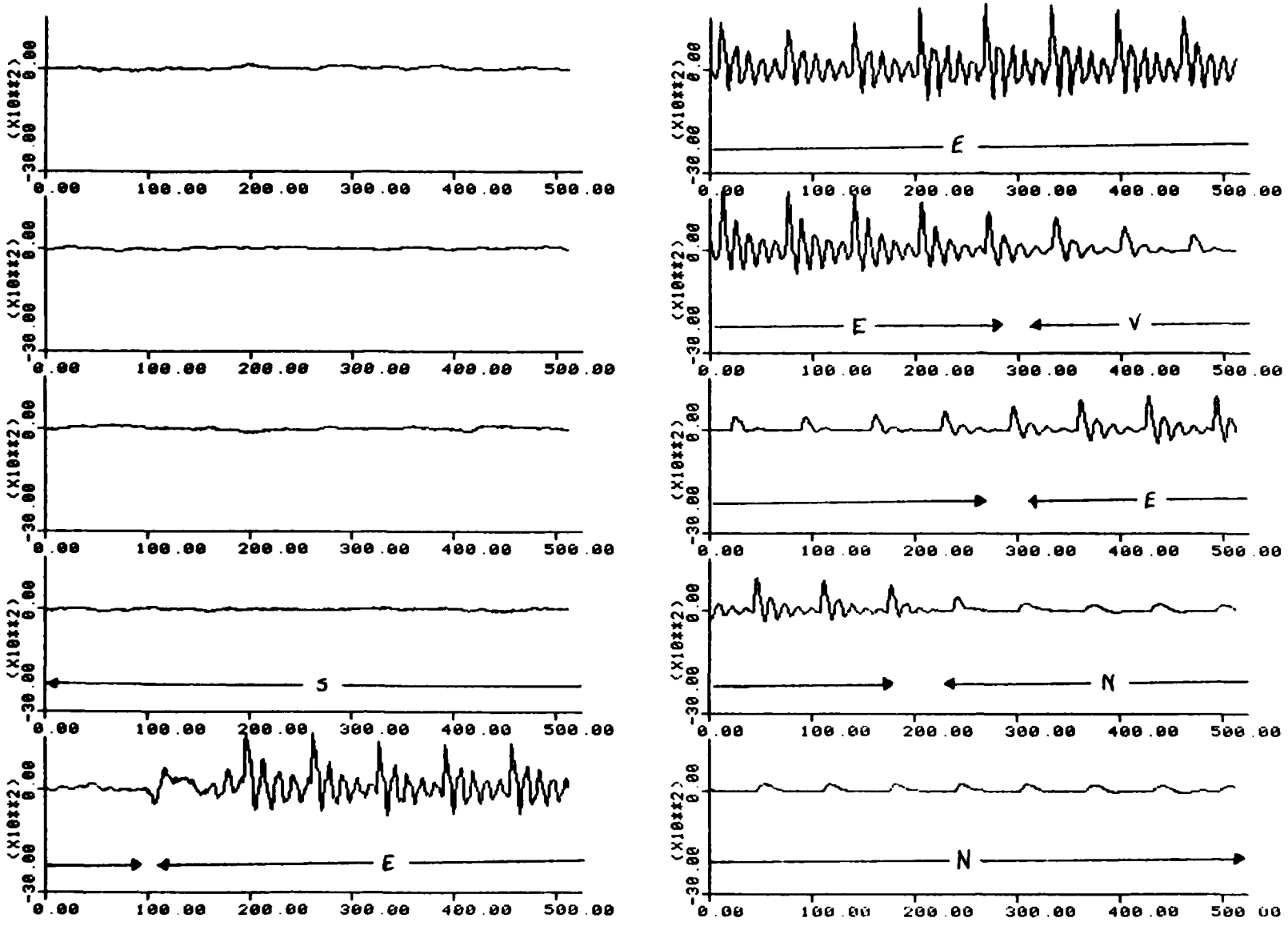

(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
S
I
K

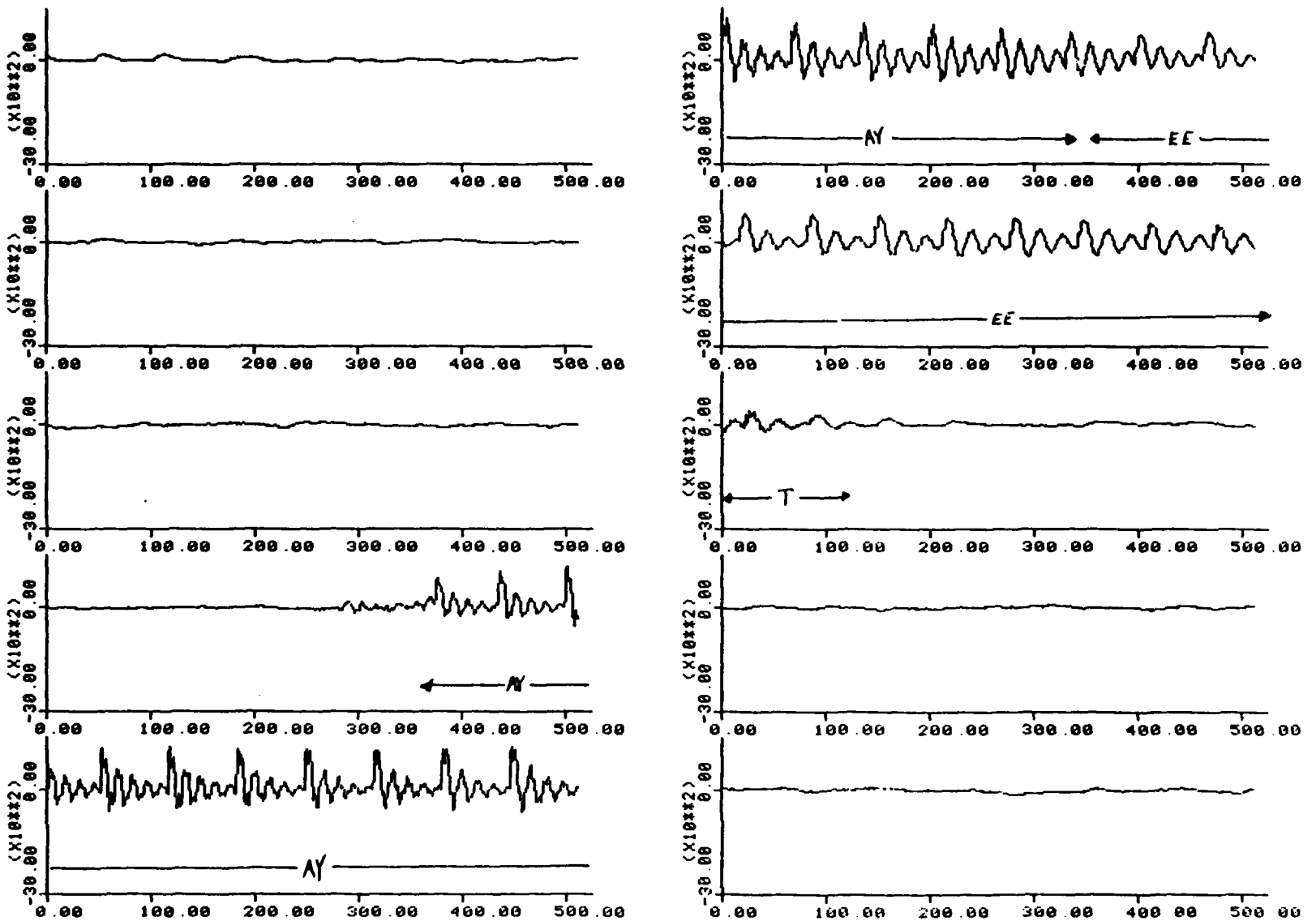

(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
S
E
E
V
E
N

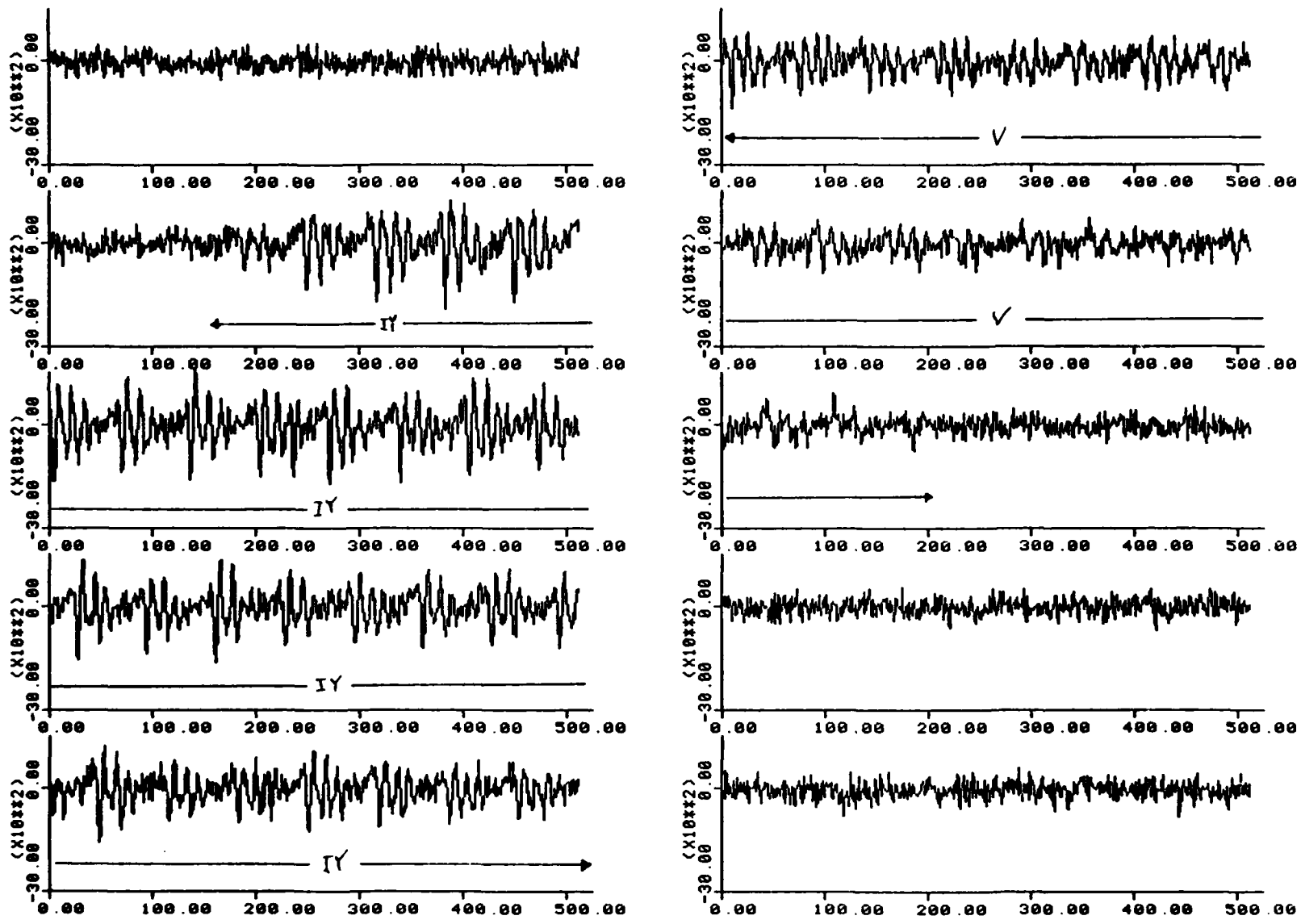

(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
EE
EE
T
AY

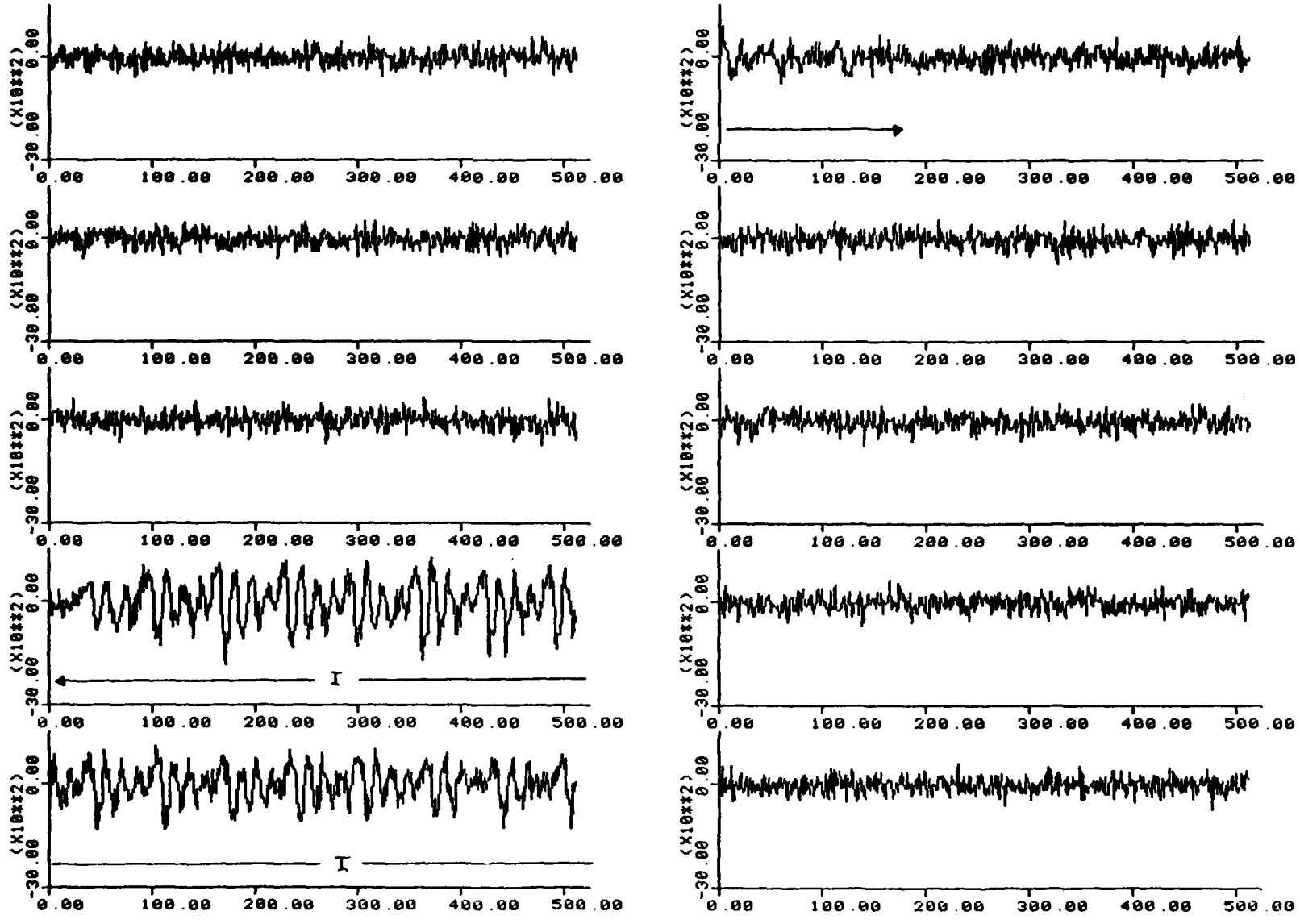

(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
F
IV
V

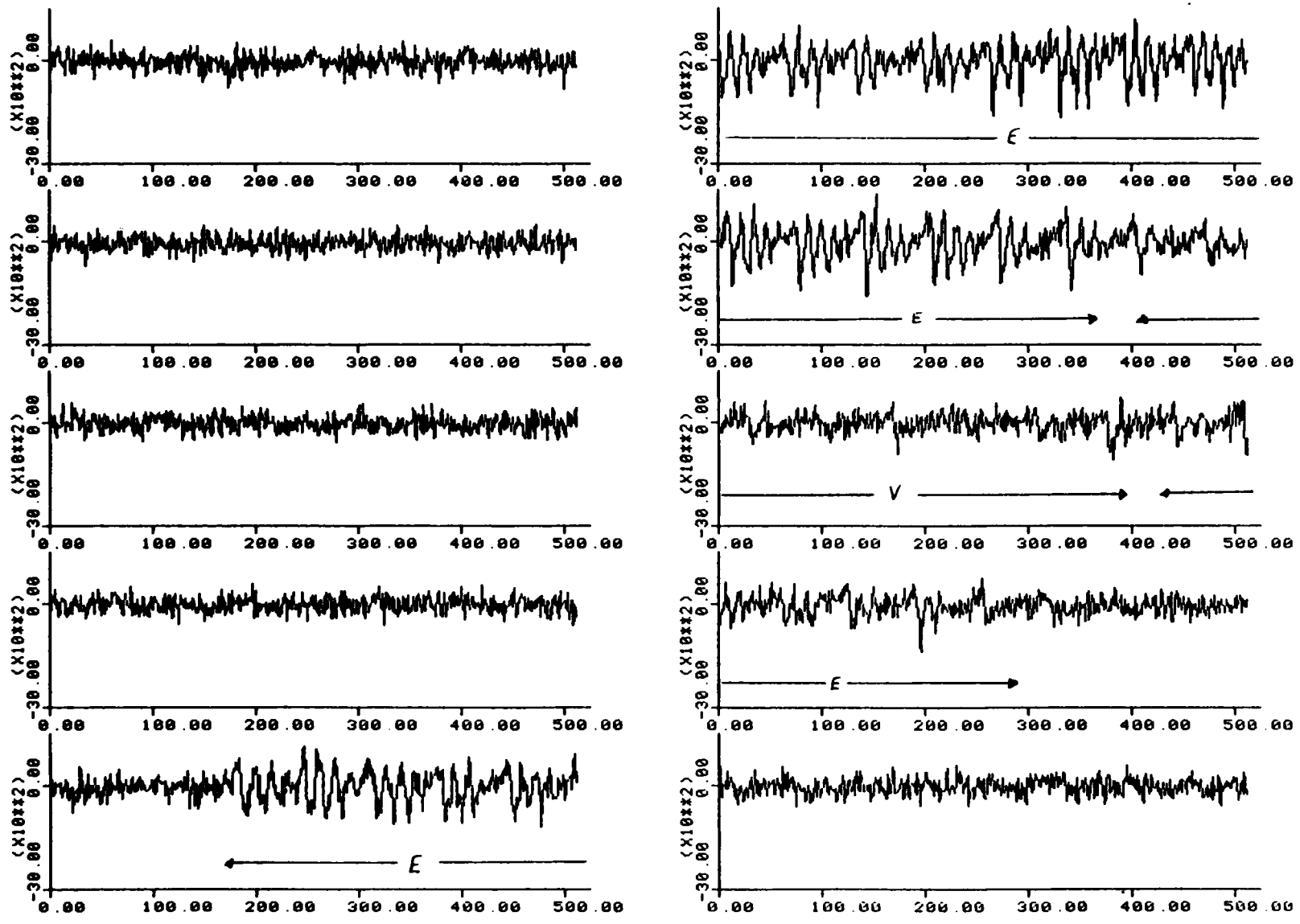

(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
S
S
I
I

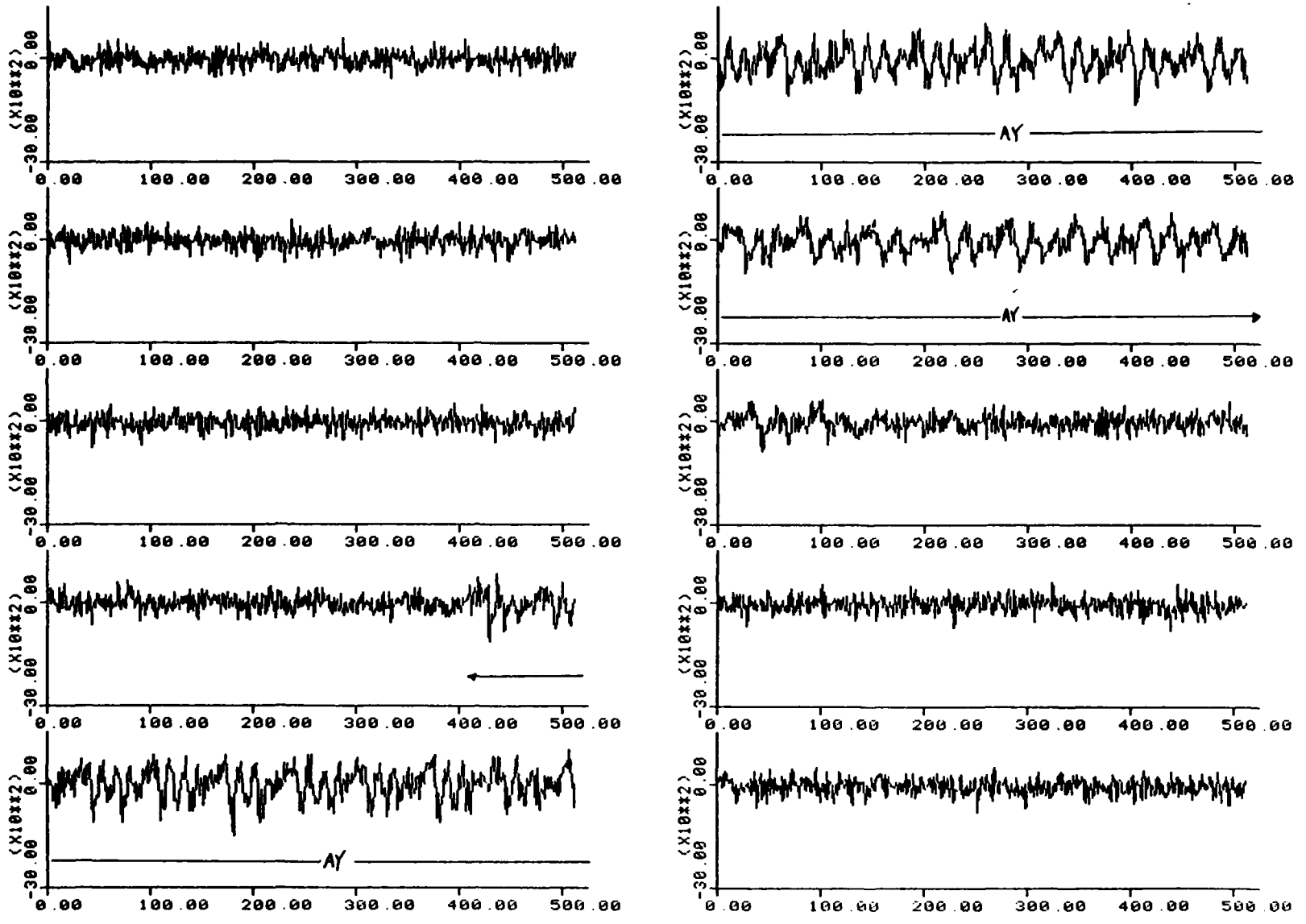

(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
S
E
E
V
E
N
N


(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
AY
EE
EE
T
AY
AY


(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
IY
IX
V


(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00


(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
E
E
V
E
E


(X10**2)
0.00
-30.00
0.00
100.00
200.00
300.00
400.00
500.00
AY

APPENDIX D

Code for Subroutine LATTICE

```
C*********************************************************************
C
C       THIS SUBROUTINE CALCULATES THE PREDICTOR COEEFICIENTS
C       BY THE LATTICE METHOD AS PRESENTED ON PP 411-416 OF
C       RABINER & SCHAFER.
C
C*********************************************************************

        SUBROUTINE LATTICE(N,X,POLES,A,ALPHA,K)

        DIMENSION X(1),A(1)
        DOUBLE PRECISION B(0:400),E(400),DA(20),DK(20),DAL,RO
        DOUBLE PRECISION TEMP1,TEMP2,SUM1,SUM2,EM,D2
        REAL K(1)
        INTEGER POLES

X       CALL OVERFL(IFLO2)
X       IF(IFLO2.EQ.1) TYPE " OVERFLOW IN PREDICT "
X       IF(IFLO2.EQ.3) TYPE " UNDERFLOW IN PREDICT "
        DO 10 I = 1,POLES
           DA(I) = DBLE(0.0)
           DK(I) = DBLE(0.0)
  10    CONTINUE
        KNE = 1
        D2 = DBLE(2.0)
        DAL = DBLE(0.0)
        B(0) = DBLE(0.0)
        DO 20 I = 1,N
            DAL = DAL + DBLE(X(I)*X(I))
  20    CONTINUE
X       DAL = 1D0
        DAL = DAL/1D04
X       RO = DAL
        DO 30 M = 1,N
           E(M) = DBLE(X(M))
           B(M) = DBLE(X(M))
  30    CONTINUE
        SUM1 = DBLE(0.0)
        SUM2 = DBLE(0.0)
        DO 40 M = 1,N
            M1 = M - 1
            TEMP1 = E(M)*B(M1)
            TEMP2 = (E(M)*E(M)) + (B(M1)*B(M1))
            SUM1 = SUM1 + TEMP1
            SUM2 = SUM2 + TEMP2
  40    CONTINUE
        DK(1) =  D2*SUM1/SUM2
        IF(DABS(DK(1)).GT.DBLE(1.))TYPE " ERROR "
X       TYPE " DK(",KNE,") = ",DK(1)
        DA(1) =  DK(1)
        DO 200 I = 2,POLES
          I1 = I - 1
          DO 50 M = 1,N
            M1 = M - 1
            EM = E(M)
            E(M) = EM - DK(I1)*B(M1)
            B(M) = B(M1) - DK(I1)*EM
            IF(DABS(E(M)).LE.1D-15) E(M) = DBLE(0.0)
            IF(DABS(B(M)).LE.1D-15) B(M) = DBLE(0.0)
  50        CONTINUE
```

```
            SUM1 = DBLE(0.0)
            SUM2 = DBLE(0.0)
            DO 60 M = 1,N
              M1 = M - 1
              TEMP1 = E(M)*B(M1)
              TEMP2 = (E(M)*E(M)) + (B(M1)*B(M1))
              SUM1 = SUM1 + TEMP1
              SUM2 = SUM2 + TEMP2
   60       CONTINUE
X           CALL OVERFL(IFLO2)
X           IF(IFLO2.EQ.1) TYPE " OVERFLOW IN SUM      "
X           IF(IFLO2.EQ.3) TYPE " UNDERFLOW IN SUM      "
            DK(I) =  D2*SUM1/SUM2
            IF(DABS(DK(I)).GT.DBLE(1.))TYPE " ERROR "
X           TYPE " DK(",I,") = ",DK(I)
            DA(I) =  DK(I)
            DO 80 J = 1,I1
              DA(J) = DA(J) - DK(I)*DA(I-J)
   80       CONTINUE
X           DAL = DAL -  DK(I)*DK(I)*DAL
  200     CONTINUE
X           CALL OVERFL(IFLO)
X           IF(IFLO.EQ.1) TYPE " OVERFLOW IN LATTICE "
X           IF(IFLO.EQ.3) TYPE " UNDERFLOW IN LATTICE "
            DO 250 M = 1,N
              M1 = M - 1
              E(M) = E(M) - DK(I1)*B(M1)
              IF(DABS(E(M)).LE.1D-20) E(M) = DBLE(0.0)
  250     CONTINUE
X           DO 300 M = 1,N
X               DAL = DAL + E(M)*E(M)
X300      CONTINUE
X           ALPHA = SNGL(SQRT(R0*DAL))
            ALPHA = SNGL(SQRT(DAL))
X           TYPE " ALPHA = ",ALPHA
            DO 100 I = 1,19
              IMJ = 21 - I
              A(IMJ) = -SNGL(DA(IMJ-1))
              K(I) = SNGL(DK(I))
  100     CONTINUE
            A(1) = 1.0
            K(20) = SNGL(DK(20))
X           CALL OVERFL(IFLO1)
X           IF(IFLO1.EQ.1) TYPE " OVERFLOW IN ALPHA    "
X           IF(IFLO1.EQ.3) TYPE " UNDERFLOW IN ALPHA    "
X           ACCEPT"CONTINUE ON A NUMBER",IJKL
            RETURN
            END
```

VITA

Craig Eugene McKown was born on 23 May 1960 in Merced, California. He is the son of Thomas E. and Evelyn L. McKown. He attended high school in Bellevue, Nebraska and graduated from Bellevue West High School in June of 1978. He subsequently attended Washington University in St. Louis with an Air Force ROTC scholarship and graduated with the degree Bachelor of Science in Electrical Engineering in May of 1982. One week after graduation and commissioning, Lt McKown entered the Air Force Institute of Technology. He is a member of Tau Beta Pi and Eta Kappa Nu engineering honor societies. He is also a student member (81) of IEEE.

Permanent Address: 1600 Alcove Ct
Midland, TX 79703

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/EE/83D-45	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENY	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Craig E. McKown, BSEE, 2LT, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1983 December 15. PAGE COUNT 136
16. SUPPLEMENTARY NOTATION 7 Feb 84	Approved for public release: IAW AFR 190-17 Lynn Wolaver Dean for Research and Professional Development Air Force Institute of Technology (AU) Wright-Patterson AFB OH 45433

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
17	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Linear Predictive Coding; Digital Speech Processing;

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: IMPLEMENTING LPC ALGORITHMS IN THE STUDY OF SPEECH PROCESSING

Thesis Chairman: Larry Kizer, Major, USAF

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Larry Kizer, Major, USAF	22b. TELEPHONE NUMBER (Include Area Code) (513)-255-3517 22c. OFFICE SYMBOL AFIT/ENY

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

This report describes a system which processes speech using linear predictive methods. The system is a software simulation of an LPC analyzer and synthesizer. The system consists of two programs, one of which processes the speech to generate the LPC parameters, and another which processes these parameters to resynthesize the speech. An important aspect of the system is that it enables the user to select from various pitch and coefficient analysis methods. It also allows the user to vary other parameters in order to simulate other changes in the processing scheme.

To test the operation of the system, a regimen of testing was performed by varying the different parameters. A separate program allows a simple method for changing all of the parameters over which the user has control. These parameters are called the decision variables and each has an allowable range of values. The system operated satisfactorily over all of the decision variables. The flexibility exhibited by the system in this testing indicates that the system can be a valuable tool for the study of linear predictive coding of speech in the Signal Processing Laboratory at the Air Force Institute of Technology.