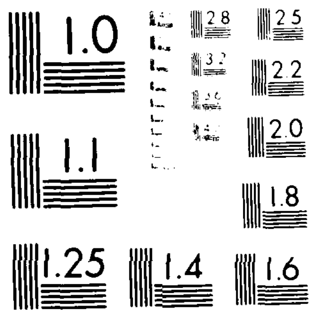
AD-A128 397 FORTRAN IMPLEMENTATION OF COMPLEX LEAST SQUARES ADAPTIVE LATTICE ALGORITH..(U) PENNSYLVANIA STATE UNIV UNIVERSITY PARK APPLIED RESEARCH LAB.. 1/1

UNCLASSIFIED B A COOPER ET AL. 12 NOV 82 F/G 12/1 NL

END
DATE
FILMED
6-83
DTIC



MICROCOPY RESOLUTION TEST CHART

(6)

FORTRAN IMPLEMENTATION OF COMPLEX LEAST SQUARES ADAPTIVE LATTICE ALGORITHM AS AN ALL ZERO INVERSE FILTER

B. A. Cooper and L. H. Sibul

Technical Memorandum
File No. TM 82-219
November 12, 1982
Contract No. N00024-79-C-6043

Copy No. 20

The Pennsylvania State University
Intercollege Research Programs and Facilities
APPLIED RESEARCH LABORATORY
Post Office Box 30
State College, PA 16801

Approved for Public Release
Distribution Unlimited

NAVY DEPARTMENT

NAVAL SEA SYSTEMS COMMAND

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER TM 82-219	2. GOVT ACCESSION NO. [illegible] 397	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FORTRAN IMPLEMENTATION OF COMPLEX LEAST SQUARES ADAPTIVE LATTICE ALGORITHM AS AN ALL ZERO INVERSE FILTER		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) B. A. Cooper and L. H. Sibul		8. CONTRACT OR GRANT NUMBER(s) N00024-79-C-6043
9. PERFORMING ORGANIZATION NAME AND ADDRESS Applied Research Laboratory P. O. Box 30 State College, PA 16801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS D. Houser, Code 63R-14 Naval Sea Systems Command Department of the Navy, Washington, DC 20362		12. REPORT DATE November 12, 1982 13. NUMBER OF PAGES 51
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

fortran, adaptive, algorithm, speech, signal processing

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

A complex least squares adaptive lattice algorithm (1) has been implemented on the ARL VAX/VMS computer as an inverse all zero filter. A FORTRAN program package is presented which also includes a recursive realization of a pole-zero prefilter and an input signal generator. A series of examples is taken from the speech field, using the inverse filter to deconvolve speech signals produced by vocal tract models of varying complexity. The input to the prefilter is shown to be accurately recovered as the output of the inverse filter.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Subject: FORTRAN Implementation of Complex Least Squares Adaptive Lattice Algorithm as an All Zero Inverse Filter

References: See Pages 14 and 15.

ABSTRACT

A complex least squares adaptive lattice algorithm (1) has been implemented on the ARL VAX/VMS computer as an inverse all zero filter. A FORTRAN program package is presented which also includes a recursive realization of a pole-zero prefilter and an input signal generator. A series of examples is taken from the speech field, using the inverse filter to deconvolve speech signals produced by vocal tract models of varying complexity. The input to the prefilter is shown to be accurately recovered as the output of the inverse filter.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

A complex least squares lattice adaptive algorithm (1) has been implemented on the ARL VAX/VMS computer as an inverse all zero filter. An interactive FORTRAN program package is presented which also includes a recursive realization of a pole-zero prefilter and an input signal generator. No knowledge of FORTRAN is required to use the package. The input signal characteristics, filter coefficients and critical lattice parameters are all user specified.

Three classes of adaptive algorithms are treated in the literature: (Widrow's) LMS, the gradient lattice, and the least squares lattice. Of the three, the least squares adaptive lattice algorithm has been shown to have the fastest convergence rate (2,3). For this reason, it has been used successfully in a wide range of applications; i.e., data deconvolution, system identification, speech processing, and spectral whitening. The program package described in this memorandum was developed with these applications in mind.

BACKGROUND

Consider a filter with transfer function

$$H(z) = \frac{Z(z)}{V(z)} = \frac{B(z)}{A(z)} \qquad (1)$$

where

$$A(z) = 1 + \sum_{k=1}^{NPOLE} A_k z^{-k} = \sum_{k=0}^{NPOLE} A_k z^{-k}, \; A_0 = 1$$

and

$$B(z) = 1 + \sum_{k=1}^{NZERO} B_k z^{-k} = \sum_{k=0}^{NZERO} B_k z^{-k}, \quad B_0 = 1$$

then $Z(z)A(z) = B(z)V(z)$ and, after taking inverse z transforms of both sides, the filter can be realized in the time domain by the following recursion:

$$z(n) + \sum_{k=1}^{NPOLE} A_k\, z(n-k) = v(n) + \sum_{k=1}^{NZERO} B_k\, v(n-k)$$

or

$$z(n) = - \sum_{k=1}^{NPOLE} A_k\, z(n-k) + v(n) + \sum_{k=1}^{NZERO} B_k\, v(n-k).$$

The roots of $A(z)$ are the poles of the system; the roots of $B(z)$ are the zeroes of the system. For stability, these roots must lie outside the unit circle because they are roots of polynomials in z^{-1}.

The resonances and antiresonances of the human vocal tract (for a particular steady state vowel sound) can be modeled by poles and zeroes, respectively, in the z plane. It is a common practice in the speech field to model these formants with an all pole model rather than with poles and zeroes; adding a sufficient number of poles has been shown to closely approximate the effect of the zeroes. The roots of the denominator polynomial are related to the vocal tract formants by the following expression (5):

$$z_k,\ z_k^* = z_{2k},\ z_{2k-1} = e^{-\sigma_k T}\, e^{\pm j 2\pi F_k T}$$

where F_k is the formant frequency and $2\sigma_k$ is the bandwidth of the k^{th} formant. There will be a pair of complex conjugate poles for each formant frequency; the order of the system will be twice the number of formants. Then

$$H(z) = \frac{1}{\prod_{k=1}^{NPOLE} (1+z_k z^{-1})} .$$

For the all-pole model, the time domain recursion becomes

$$z(n) = -\sum_{k=1}^{NPOLE} A_k \, z(n-k) + v(n).$$

An inverse filter for the system of Equation 1 would have the transfer function

$$H(z) = \frac{A(z)}{B(z)} .$$

For the speech case (all-pole prefilter), the inverse filter would be all-zero:

$$H_{pre}(z) = \frac{1}{A(z)};\; H_{inv}(z) = \frac{1}{H_{pre}(z)} = A(z).$$

Obviously, if the systems are cascaded, the input to the prefilter will be recovered as the output of the inverse filter.

A complex adaptive lattice structure has been used to develop the inverse filter. The actual lattice algorithm is taken from (1). Users are referred to (2) and (3) and the references cited therein for the derivations of the complex least squares adaptive lattice algorithm.

PROGRAM PACKAGE

The lattice program package consists of the following interactive FORTRAN programs:

GEN.FOR	input generator
FILTER.FOR	prefilter and realization of the lattice filter
LSL.FOR	complex adaptive lattice processor

the following subroutines:

SYNTH	synthesizes complex polynomial from formants and bandwidths
SYNTH1	synthesizes complex polynomial from its roots
ZCPOLY	solves polynomial with complex coefficients

and the following functions:

MTH$RANDOM	(VAX/VMS library) uniform random number generator
AMAXLOC	finds location of highest value in an array
ARRMAX	finds highest value in an array

The execution of the programs may be controlled by either one of two command procedures: LATTICE.COM when using a VT100 terminal, or LATTEK.COM when using the Tektronix terminal in Room 358, ASB. These command procedures are given in Appendix A. They may be invoked by typing @LATTICE and @LATTEK, respectively. The following command should be placed into either the command procedure or the user's LOGIN.COM file to allow access to the IMSL package. If LNK$LIBRARY has already been assigned, use LNK$LIBRARY1 or the next highest unassigned library.

```
$ASSIGN DRA0:[SYSTEM]IMSLIBS.OLB LNK$LIBRARY
```

Appendix B presents a sample terminal session for the case illustrated in Figure 3. For the user interested in modifying the programs, the rest of this memorandum deals with the structure of the individual programs.

Figure 7 illustrates the calling hierarchy of the lattice package. The inputs to program LSL are Gaussian white noise and the output time series z(n) from program FILTER.FOR. The outputs from LSL are the set of inverse filter coefficients (A_k; k=1,... NZERO) to which the lattice has adapted after the requested number of iterations, and the time series output $e_p(n)$ which is equivalent to the result of passing the inp time series z(n) through the transfer function H(z) = A(z). The outpu ilter coefficients are fed back into program FILTER to be realized rec vely with input z(n). Thus, since program FILTER operates for any H(z c is both a prefilter and the realization of the lattice filter.

A sample case, then, may be illustrated by a series of five plots (Figures 3-6). The relationships between the plots are shown by the block diagram in Figure 2:

(a) input time series
(b) prefilter transfer function
(c) output of the prefilter
(d) lattice (inverse) filter transfer function
 (realized by program FILTER)
(e) output of the lattice filter

So that the filter outputs may be most easily evaluated, the filter outputs (c) and (e) for deterministic prefilter inputs are time series plots; for a stochastic input, the outputs (c) and (e) are power spectra. The time series plots (c) and (e) (from above) for the deterministic case are presented in Figures 3-6 for one period of the waveform after the time series has stabilized (or adapted). Again, the output of the lattice filter should approximate the input to the prefilter if the lattice is a true inverse filter. It is obvious that some measure of goodness of the inverse filter should be applied to the output of the lattice filter. At present, only a graphical comparison has been attempted. However, Gray and Markel (7) suggest a spectral flatness measure which will help to smooth the discontinuities in the power spectrum caused by the occurrence of the inverse filter zeroes being offset by a few Hz. from the prefilter poles. Also, a mean square error criterion might be applied to the difference between the deterministic time series output (error signal) and the input time series. The transfer function of the lattice filter should be the inverse of the prefilter transfer function. Thus, where the prefilter has poles (peaks) the inverse filter will have zeroes (dips).

GEN.FOR

Program GEN.FOR generates both Gaussian white noise and an impulse sequence. A more accurate approximation to the glottal wave is given in (7). However, since the thrust of this work is aimed at demonstrating that the lattice filter is indeed an accurate inverse filter, the two cases used so far are sufficient models of voiced and unvoiced steady state vocal tract excitations.

Program GEN.FOR calls the uniform randor number generator, MTH$RANDOM, from the VAX/VMS library. The Gaussian noise is then developed following a technique from (9). The output time series are in complex format but the imaginary parts are zero. The sampling frequency is 10 kHz. For the complex impulse input, the frequency is variable; the examples shown in Figures 3 and 5 use F0 = 125 Hz, a reasonable fundamental frequency for an adult male voice. The interactive user inputs to program GEN are F0, fundamental frequency; M, number of desired samples; and σ, the standard deviation of the noise. The time series are each stored in an output disk file.

FILTER.FOR

The user selects the desired type of input and the source of the filter coefficients. Several options exist concerning specification of coefficients. The formant frequencies and their bandwidths may be specified and the coefficients calculated from subroutine SYNTH and then stored on disk. The stored coefficients may be used with another input option,reading in coefficients from disk. The coefficients may be typed in directly from the keyboard. For the realization of the lattice (inverse) filter, the forward predictor (outputs from program LSL) coefficients may be read in from disk as the lattice filter coefficients. The other interactive user inputs to FILTER.FOR are M, desired number of time samples; NPOLE, number of poles; and NZERO, the number of zeroes. It is possible to specify a prefilter transfer function which has both poles and zeroes (the inverse filter would still be all zero). The modeling of a pole zero process by an all-pole process is discussed by Friedlander and Maitra (6). The examples presented here involve prefilters containing only poles for the purpose of illustrating the accuracy of the all-zero inverse filter.

The filter gain is computed by the following formula:

$$G = 10 \text{ LOG } \frac{\sum_{n=1}^{M} z^2(n)}{\sum_{n=1}^{M} v^2(n)} = \frac{\sigma^2_{out}}{\sigma^2_{in}}$$

where $\mu = 0$ and $\sigma^2 = \sigma_z^2 = \sigma_x^2 + \sigma_y^2$.

The filter transfer function is realized from the filter coefficients by the following formula:

$$20 \text{ LOG } |H(\omega)| = 20 \text{ LOG } \left|\frac{1}{A(z)}\right| = 20 \text{ LOG } \left(\frac{1}{A(z)} \cdot \frac{1}{A^*(z)}\right)^{\frac{1}{2}}$$

for the prefilter and

$$20 \text{ LOG } |A(z)| = 20 \text{ LOG } (A(z) \cdot A^*(z))^{\frac{1}{2}}$$

for the lattice filter, where

$$A(z) = 1 + \sum_{k=1}^{NPOLE} A_k \, e^{-j\omega k} \; .$$

The frequency range of the plots is 0-5000 Hz.

The output power spectrum is computed (4) for the stochastic case from 10 LOG $P(\omega)$ where

$$P(\omega) = P_{in}(\omega) \cdot |H(\omega)|^2 = P_{in}(\omega) \cdot \frac{1}{A(z) \cdot A^*(z)} \; .$$

For the prefilter, $P_{in}(\omega) = \sigma^2_{in} = \sigma^2_v$. For the lattice filter, $P_{in}(\omega)$ will be $P_{out}(\omega)$ from the prefilter, and $P(\omega) = P_{in}(\omega) \cdot |H(\omega)|^2 = P_{in}(\omega) \cdot A(z) \cdot A^*(z)$. Again, the frequency range of the plots is 0-5000 Hz.

LSL.FOR

Figure 1 shows the structure of the least squares complex adaptive lattice. Table I lists the lattice parameters and their relationships to the text equations in (3) and to the FORTRAN code in (1). The algorithm from (1) has been modified (from RATFOR) for use by the ARL VAX/VMS computer. Users are referred to (2) for a detailed explanation of filter parameters. The interactive user inputs to program LSL are SNR, desired signal to noise ratio in dB; ALSL, value of α_{CLSL}; P, order of the filter; and LIM, the desired number of iterations. After LIM number of iterations, the forward predictor coefficients become the filter coefficients of the inverse filter. The convergence properties of this lattice structure have been studied by Hodgkiss (2,3) and are not considered here. The maximum allowable number of iterations is set at 5000 which allows the lattice more than enough time to adapt.

Subroutine ZCPOLY solves the filter polynomial to make sure the roots are outside the unit circle. If a root is inside the unit circle, it is inverted and a new polynomial is synthesized with subroutine SYNTH1. The coefficients are then stored on disk for use by program FILTER.

The forward and backward reflection (PARCOR) coefficients are also stored on disk for possible future use. These parameters are related to the reflection coefficients of an acoustic tube model of the vocal tract. The next step in this research effort will involve making use of these coefficients to directly identify the prefilter transfer function. Table II lists the prefilter coefficients, final lattice coefficients, and both sets of final PARCOR coefficients for the four sample cases presented in Figures 3-6.

The desired signal to noise ratio is achieved by scaling down the added noise (of user specified standard deviation from GEN.FOR) rather than scaling up the signal. This is accomplished by solving for the desired σ of the noise (8):

$$\mathrm{SNR} = 10\,\mathrm{LOG}\,\frac{\sum_{n=1}^{\mathrm{LIM}} \mathrm{sig}^2(n)}{\sum_{n=1}^{\mathrm{LIM}} \mathrm{noise}^2(n)}$$

taking the antilog:

$$10^{(\mathrm{SNR}/10)} = \frac{\sum_{n=1}^{\mathrm{LIM}} \mathrm{sig}^2(n)}{\sum_{n=1}^{\mathrm{LIM}} \mathrm{noise}^2(n)}$$

$$\sum_{n=1}^{\mathrm{LIM}} \mathrm{noise}^2(n) = \frac{\sum_{n=1}^{\mathrm{LIM}} \mathrm{sig}^2(n)}{10^{(\mathrm{SNR}/10)}}$$

$$\frac{\sum_{n=1}^{\mathrm{LIM}} \mathrm{noise}^2(n)}{\mathrm{LIM}} = \sigma^2_{\mathrm{noise}} = \frac{\sum_{n=1}^{\mathrm{LIM}} \mathrm{sig}^2(n)}{\mathrm{LIM} \cdot 10^{(\mathrm{SNR}/10)}}$$

$$\sigma_{\mathrm{noise}} = \left(\frac{\sum_{n=1}^{\mathrm{LIM}} \mathrm{sig}^2(n)}{\mathrm{LIM} \cdot 10^{(\mathrm{SNR}/10)}} \right)^{\frac{1}{2}}$$

Therefore, multiply the noise samples by the factor $\frac{\sigma_{\mathrm{noise}}}{\sigma_{\mathrm{input}}}$.

ACKNOWLEDGMENTS

The complex least squares lattice algorithm in program LSL is an adaptation of a program from the package developed by W. S. Hodgkiss, D. Alexandrou, and J. A. Presley (1-3,9).

The authors would also like to thank J. R. Sacha for the benefit of his generous help and limitless expertise.

REFERENCES

1. D. Alexandrou and W. S. Hodgkiss, "Complex adaptive one-step linear predictor algorithms," MPL TM-341, Marine Physical Laboratory, Scripps Institution of Oceanography, San Diego, CA, 23 June 1982.

2. W. S. Hodgkiss and J. A. Presley, "Adaptive tracking of multiple sinusoids whose power levels are widely separated," IEEE Trans. Acoust., Speech and Signal Processing, Vol. ASSP-29, pp. 710-721, June 1981.

3. W. S. Hodgkiss and J. A. Presley, "The complex adaptive least-squares lattice," IEEE Trans. Acoust., Speech and Signal Processing, Vol. ASSP-30, pp. 330-333, April 1982.

4. L. J. Griffiths, "Rapid measurement of digital instantaneous frequency," IEEE Trans. Acoust., Speech and Signal Processing, Vol. ASSP-23, pp. 207-222, April 1975.

5. L. R. Rabiner and R. W. Schafer, <u>Digital Processing of Speech Signals</u>, Englewood Cliffs, NJ, Prentice-Hall, 1978.

6. B. Friedlander and S. Maitra, "Speech deconvolution by recursive ARMA lattice filters," Proc. 1981 Conference on Acoustics, Speech and Signal Processing, Atlanta, Georgia, March 30 - April 1, 1981, pp. 865-868.

7. A. H. Gray, Jr., and J. D. Markel, "A spectral-flatness measure for studying the autocorrelation method of linear prediction of speech analysis," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-22, pp. 207-217, June 1974.

8. B. Friedlander, "System identification techniques for adaptive signal processing," Circuits Systems Signal Process., Vol. 1, No. 1, pp. 3-41, 1982.

9. W. S. Hodgkiss, Personal communication, 8 July 1982.

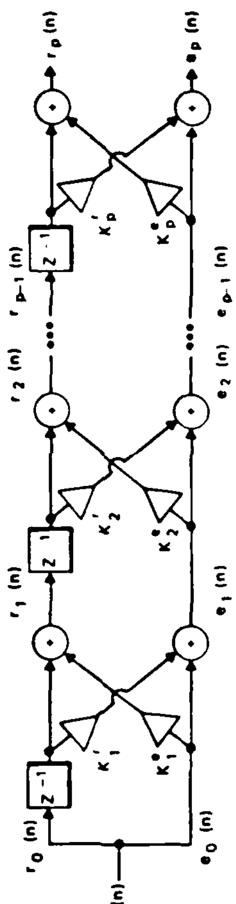
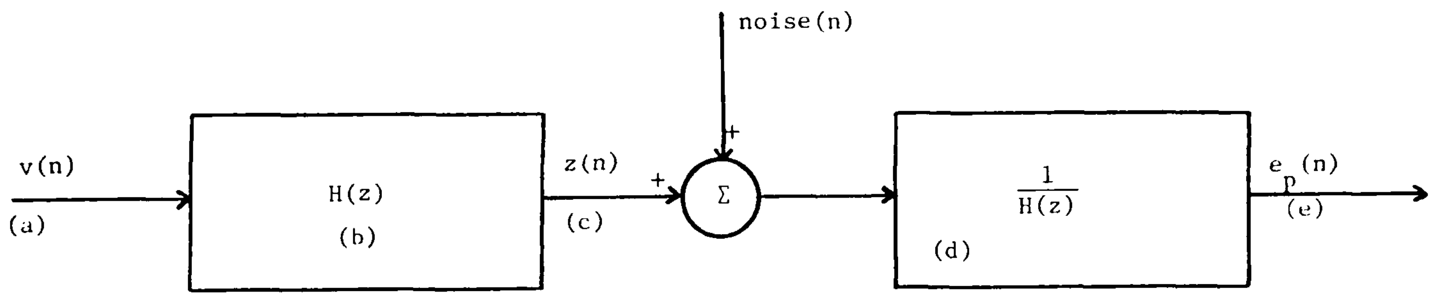


FIGURE 1. LATTICE (INVERSE) FILTER STRUCTURE



(a) Input to the prefilter, v(n) (impulse sequence or white noise sequence).

(b) Prefilter transfer function, $20 \log |H(z)|$.

(c) Output of the prefilter, z(n).

(d) Lattice (inverse) filter transfer function, $20 \log \left|\frac{1}{H(z)}\right|$.

(e) Output of the lattice filter (error signal, $e_p(n)$, for deterministic case; output power spectrum, $10 \log P(\omega)$, for stochastic case).

FIGURE 2. BLOCK FLOW DIAGRAM OF LATTICE (INVERSE) FILTER IMPLEMENTATION

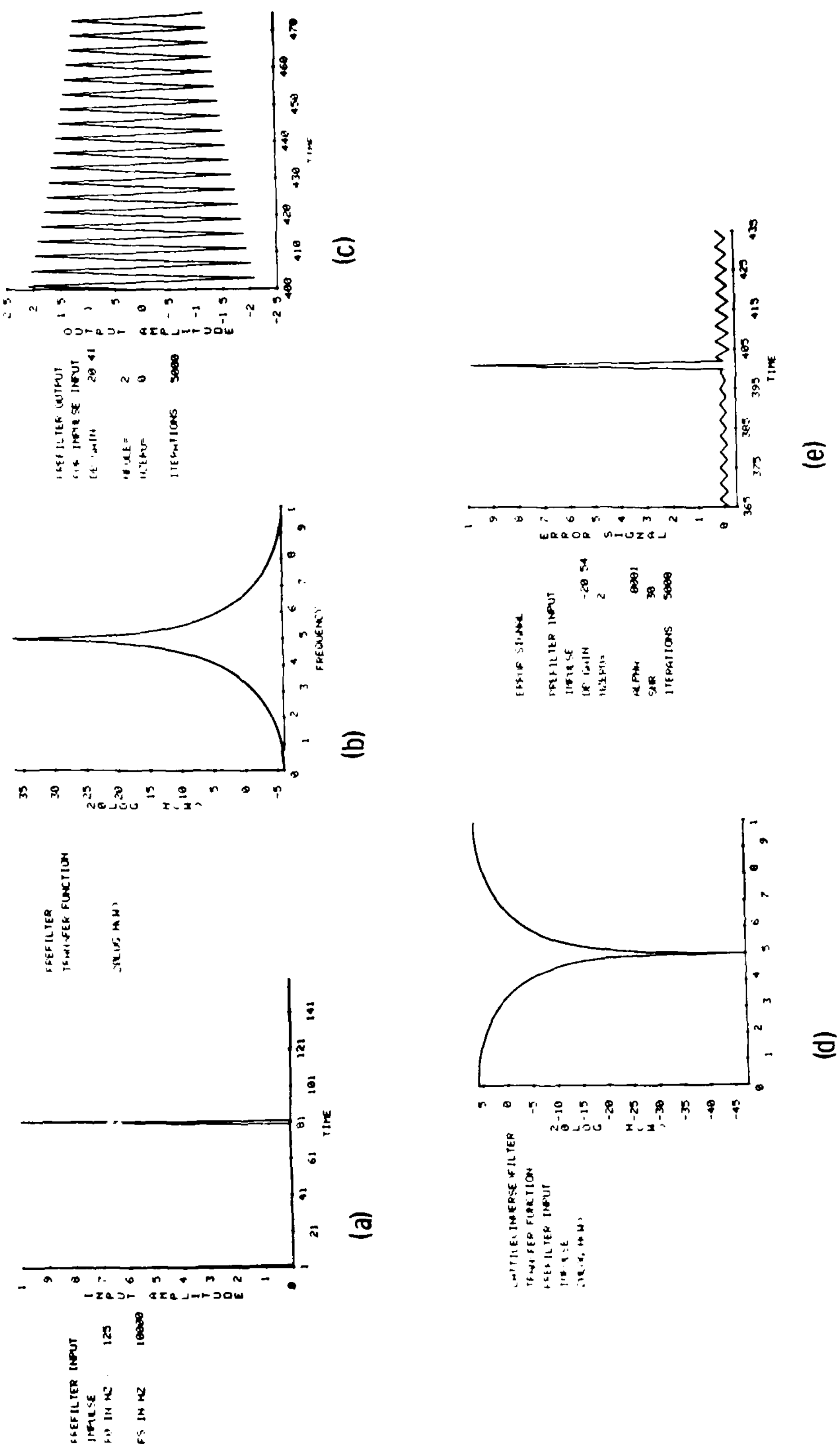


Fig. 3. Each state of the system (see Fig. 2) for an impulse function input to the one-formant case.

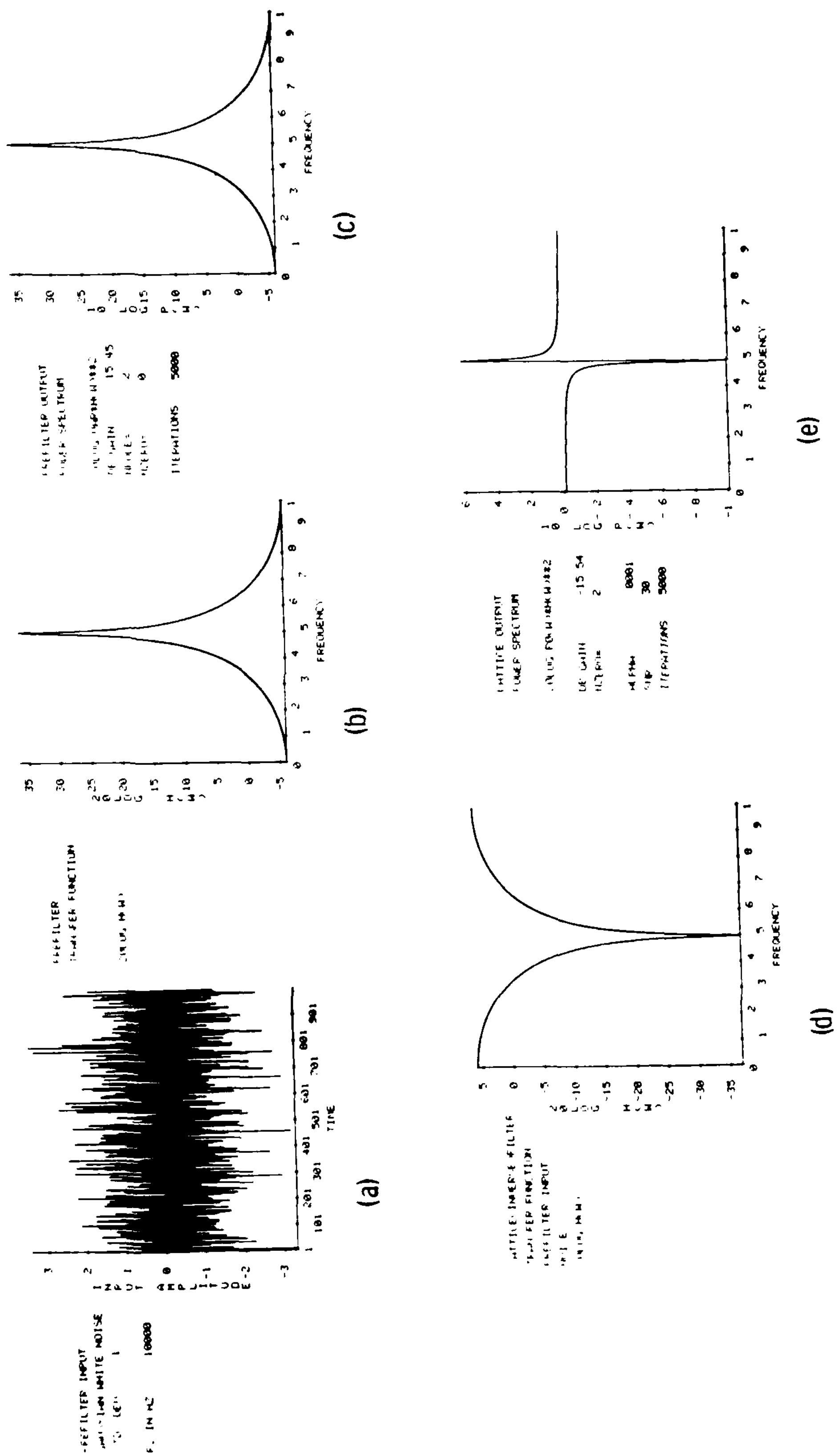


Fig. 4. Each state of the system (see Fig. 2) for a noise input to the one-formant case.

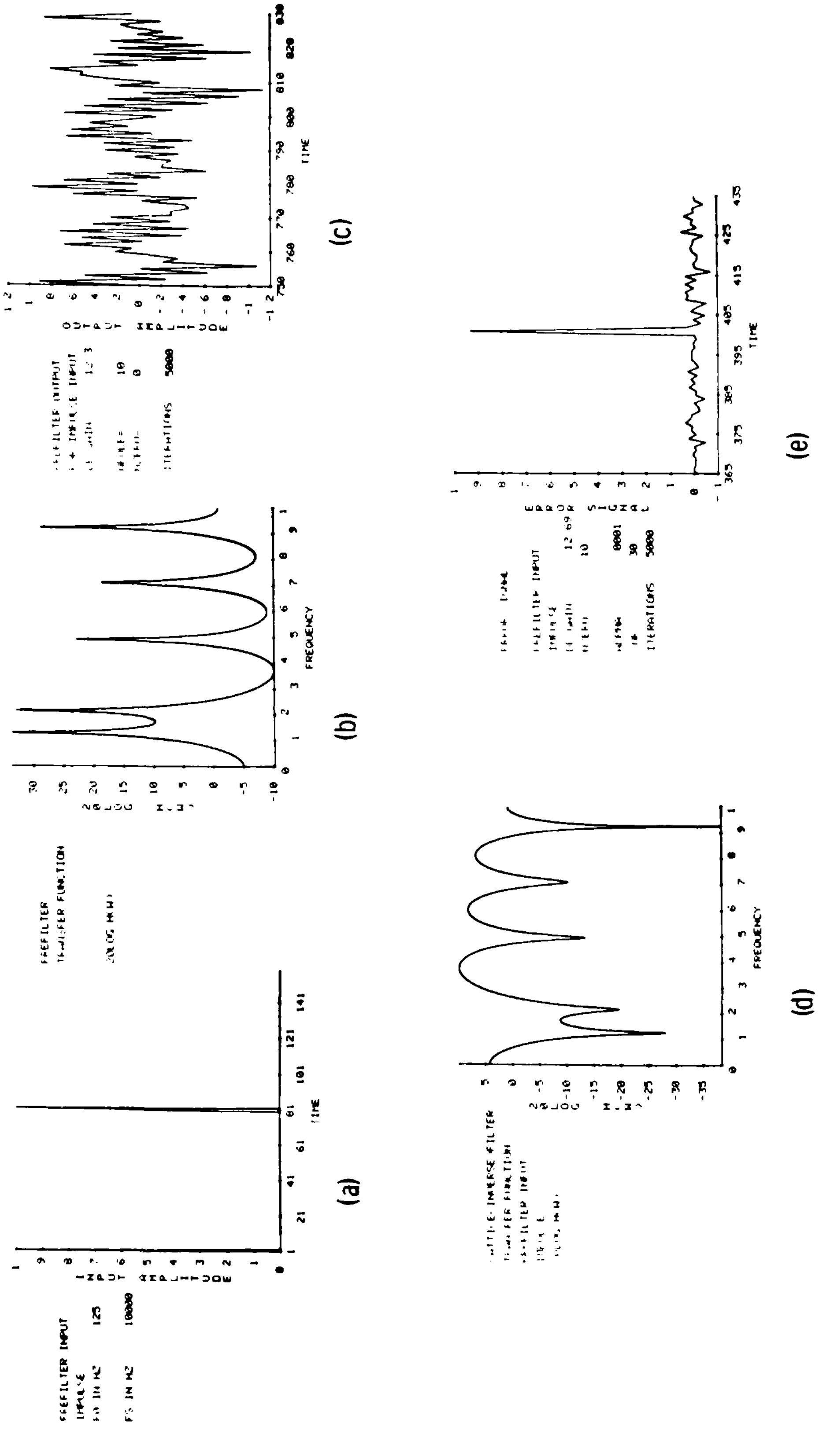


Fig. 5. Each state of the system (see Fig. 2) for an impulse function input to the five-formant case.

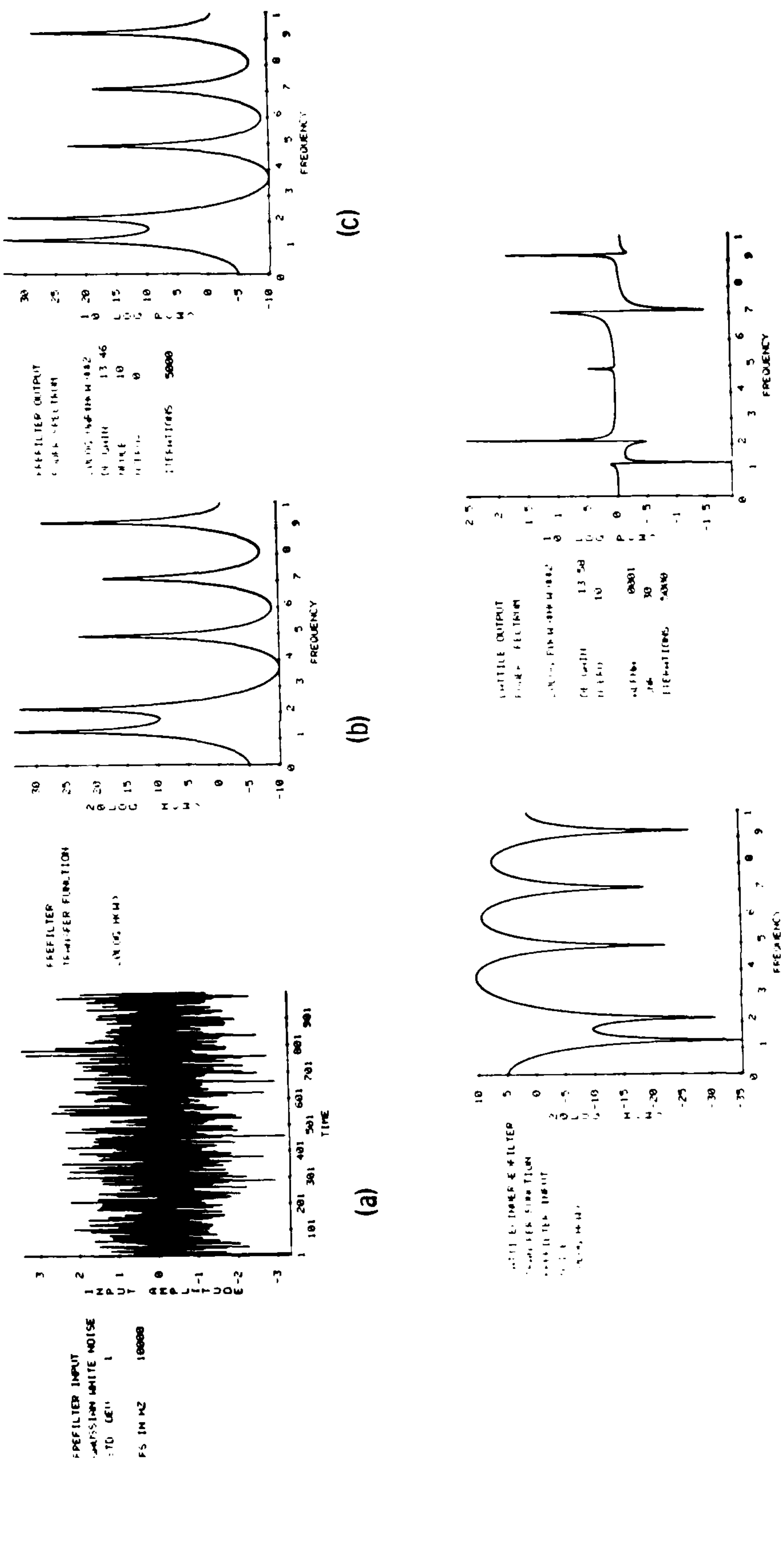


Fig. 6. Each state of the system (see Fig. 2) for a noise input to the five-formant case.

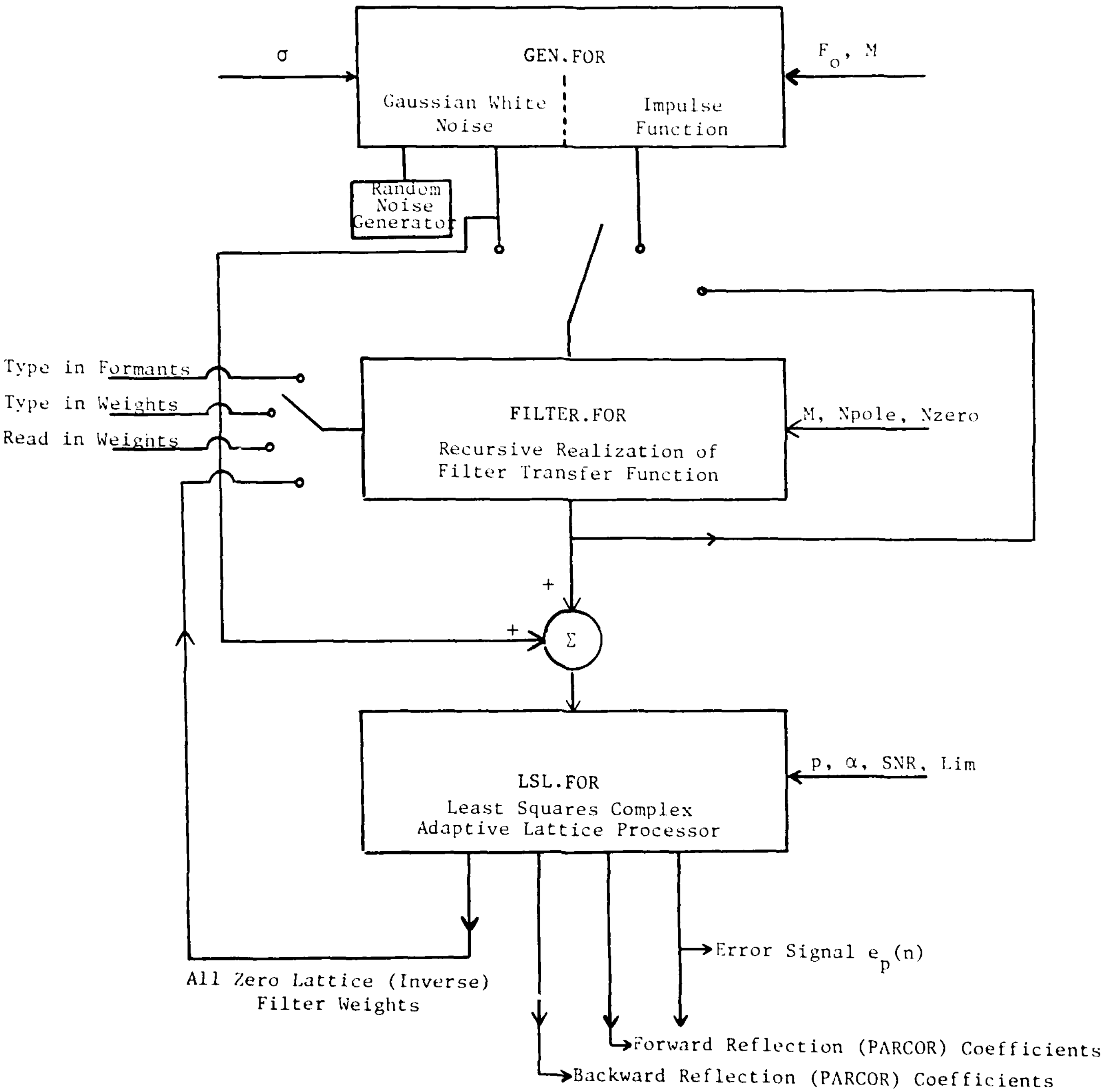


FIGURE 7. CALLING HIERARCHY AND STRUCTURE OF COMMAND PROCEDURE

TABLE I

RELATIONSHIPS BETWEEN LATTICE PARAMETERS, REF. 1 SYMBOLS, AND FORTRAN VARIABLES

Lattice (Inverse) Filter Parameter	REF. 1 SYMBOL	FORTRAN VARIABLE
Vector of forward predictor coefficients at stage i	$a_k^{(i)}(n)$ $1 \le k \le i-1$	A(M) $1 \le M \le J-1$
Highest (ith) forward predictor at state i	$a_i^{(i)}(n)$	A(J)
Vector of backward predictor coefficients at stage i	$b_k^{(i)}(n)$ $1 \le k \le i-1$	B(M,J) $1 \le M \le J-1$
Vector of backward predictors at previous stage, previous time	$b_{k-1}^{(i-1)}(n-1)$	B1(M-1,J-1)
Initialize backward predictor at stage i	$b_o^{(i)}(n)$	B(2,J)
Zero out vector of backward predictors for lower stages at time = -1	$b_k^{(i)}(-1)$ $0 \le k \le i-1,\ i \ne 0,\ i \ne p$	B1(I,J)
Forward power at stage i	$E_i^e(n)$	EE(J)
Initialize forward power at time = n	$E_o^e(n)$	EE(2)
Backward power at stage i	$E_i^r(n)$	ER(J)
Initialize backward power for lower stages at time = -1	$E_i^r(-1)$ $i \ne p$	ER1(I)
Initialize backward power at time = n	$E_o^r(n)$	ER(2)
Forward reflection coefficient at stage i	$K_i^e(n)$	KE(J)
Backward reflection coefficient at stage i	$K_i^r(n)$	KR(J)
Forward error at stage i	$e_i(n)$	SE(J)
Initialize forward error at time = n	$e_o(n)$	SE(2)
Backward error at stage i	$r_i(n)$	SR(J)
Initialize backward error at time = n	$r_o(n)$	SR(2)
Initialize backward error for lower stages at time = -1	$r_i(-1)$ $i \ne p$	SR1(I)

TABLE 1 (Continued)

RELATIONSHIPS BETWEEN LATTICE PARAMETERS, REF. 1 SYMBOLS, AND FORTRAN VARIABLES

Lattice (Inverse) Filter Parameter	REF. 1 SYMBOL	FORTRAN VARIABLE
Stepsize parameter at stage i	$\Delta_i(n)$	DEL(J)
Initialize stepsize for each stage i at time = -1	$\Delta_i(-1)$ $i \neq 0$	DEL(I)
Input time series	$x(n)$	X(I)
Gain related parameter $(\text{gain} = (1-\gamma_{i-2}(n-1))^{-1})$	$\gamma_{i-2}(n-1)$	GAM(J-2)
Initialize gamma for time = n - 1	$\gamma_{-1}(n-1)$	GAM(1)
Stage variable, current stage	i	J $2 \leq J \leq P+2$
Stage variable, lower stages	k	M $3 \leq M \leq J-1$
Time variable	(n)	Implicit
Time = -1, initial time	(-1)	Denoted by 1(I)
Previous time	(n-1)	Denoted by 1( )
Related to fade factor, $(1-\alpha_{CLSL})$	α_{CLSL}	ALSL
Filter order	P	P(in LSL), NZERO(in filter)
Number of time samples (iterations)	N	LIM(in LSL), M(in filter)

Case Figure No.	Prefilter Formant, Hz	Prefilter Bandwidth, Hz	Prefilter A_k, k=1,...Npole	Lattice Filter A_k, k=1,...Nzero
3	2500.00	150.00	0.000000 0.985112	0.000000 0.995760
4	2500.00	150.00	0.000000 0.985112	-0.002642 0.985765
5	650.30 1075.70 2463.10 3558.30 4631.30	94.10 91.40 107.40 198.70 89.80	-0.266366 -0.522137 0.174152 0.247624 0.513600 0.260513 0.141718 -0.496971 -0.241914 0.943518	-0.232250 -0.543238 0.145636 0.211085 0.515390 0.260130 0.067095 -0.420906 -0.221051 0.860902
6	650.30 1075.70 2463.10 3558.30 4631.30	94.10 91.40 107.40 198.70 89.80	-0.266366 -0.522137 0.174152 0.247624 0.513600 0.260513 0.141718 -0.496971 -0.241914 0.943518	-0.263710 -0.529765 0.172208 0.254260 0.508698 0.264767 0.146636 -0.505169 -0.236822 0.941123

TABLE II

FORMANT FREQUENCIES, THEIR BANDWIDTHS, AND CORRESPONDING FILTER COEFFICIENTS FOR THE FOUR INPUT CASES DEPICTED IN FIGS. 3-6

APPENDIX A

FORTRAN SOURCE CODE

Command Procedure LATTICE.COM
Controls Sequence of Program Execution for VT100 Terminals

```
**** TO STOP SCROLL, PUSH 'NO SCROLL'. TO START AGAIN, PUSH 'NO SCROLL'.****

************************************************************************

SINGLE CHANNEL LEAST SQUARES ADAPTIVE ALGORITHM PROGRAM AND PLOTTING PACKAGE

BETH A. COOPER
**
****************************************************************************
THIS PROGRAM PACKAGE HAS BEEN REVISED AS OF 12/6/82. THE LISTINGS SUPPLIED
WITH BBN INTERNAL MEMORANDUM 82-219, DATED 11/12/82 ARE NO LONGER CORRECT.
PLEASE OBTAIN NEW LISTINGS FOR THE FOLLOWING PROGRAMS: FILTER.FOR, GEN.FOR
LSL.FOR, AND SYNTH.FOR.
*****************************************************************************

A COMMAND PROCEDURE CONTROLS THE COMPILING, LINKING AND EXECUTION OF EACH
PROGRAM. THE COMMAND PROCEDURE WILL PROMPT FOR EACH PROGRAM TO BE RUN.


YOU MAY EXIT FROM THE PROCEDURE AT ANY TIME BY TYPING NO AFTER THE PROMPT TO
CONTINUE. YOU WILL BE ABLE TO OBTAIN HARD COPIES OF DATA FILES WITHIN THE
COMMAND PROCEDURE, BUT AFTER YOU EXIT FROM THE PROCEDURE, ALL YOUR DATA FILES
WILL BE DELETED TO SAVE DISK SPACE.  BE SURE TO REMEMBER BETWEEN PROGRAMS WHICH
DISKFILES YOUR DATA IS STORED IN.


TO BEGIN, ENTER NAME OF FIRST PROGRAM YOU WOULD LIKE TO RUN.
ENTER PROGRAM NAMES WITHOUT THE FILE TYPE, FOR EXAMPLE, FILTER, NOT FILTER.FOR.

AFTER YOU FINISH LOOKING AT A PLOT ON THE SCREEN, PRESS THE SPACE BAR TO
CAUSE THE PLOT TO ERASE AND THE NEXT PLOT TO APPEAR.

WHEN USING THE TEKTRONIX PLOTTER, YOU WILL NEED TO DEPRESS THE RETURN BUTTON
TWICE TO GET THE PLOTS TO ERASE.


PLEASE WAIT FOR THE COMPUTER TO COMPILE AND LINK THE PROGRAMS. YOU WILL
RECEIVE A PROMPT WHEN THE COMPUTER IS READY.


TYPE LATTICE.TXT
FORTRAN POLYROOT
LINK POLYROOT
FORTRAN POLY
LINK POLY
FORTRAN GEN
LINK GEN
FORTRAN FILTER
LINK FILTER,LATTLIB/LIBRARY,SYS$LIBRARY:GCSLIB/LIB,SYS$LIBRARY:GCSVT100/LIB
FORTRAN LSL
LINK LSL,LATTLIB/LIBRARY
PURGE
L1:
INQUIRE F1 "WHICH PROGRAM?"
ASSIGN/USER_MODE SYS$COMMAND: SYS$INPUT:
RUN 'F1'
INQUIRE CHECK "ENTER Y[ES] TO CONTINUE"
IF .NOT.CHECK THEN GOTO LABEL
GOTO L1

$LABEL:
$INQUIRE CHECK "ENTER Y[ES] FOR HARD COPY OF ALL FILTER COEFFICIENTS."
$IF .NOT.CHECK THEN GOTO LABEL4
$INQUIRE F2 "ENTER NUMBER OF FILE IN WHICH YOU HAVE STORED PREFILTER INFO."
$LIST 'F2'
$INQUIRE F3 "ENTER NUMBER OF FILE IN WHICH YOU HAVE STORED LATTICE INFO."
$LIST 'F3'
$LABEL4:
$INQUIRE CHECK "ENTER Y[ES] TO DELETE ALL DATA FILES."
$IF .NOT.CHECK THEN GOTO LABEL5
$DELETE *.DAT;*
$LABEL5:
$EXIT
```

Command Procedure LATTEK.COM
Controls Sequence of Program Execution for Tektronix Terminal

```
**** TO STOP SCREEN, PUSH NO SCROLL. TO START AGAIN, PUSH NO SCROLL.****

*************************************************************************

SINGLE CHANNEL LEAST SQUARES ADAPTIVE ALGORITHM PROGRAM AND PLOTTING PACKAGE

BETH A. COOPER

*************************************************************************
THIS PROGRAM PACKAGE HAS BEEN REVISED AS OF 12/6/82. THE LISTINGS SUPPLIED
WITH ARL INTERNAL MEMORANDUM 82-219, DATED 11/12/82 ARE NO LONGER CORRECT.
PLEASE OBTAIN NEW LISTINGS FOR THE FOLLOWING PROGRAMS: FILTER.FOR, GEN.FOR,
LSL.FOR, AND SYNTH.FOR.
*************************************************************************


A COMMAND PROCEDURE CONTROLS THE COMPILING, LINKING AND EXECUTION OF EACH
PROGRAM. THE COMMAND PROCEDURE WILL PROMPT FOR EACH PROGRAM TO BE RUN.


YOU MAY EXIT FROM THE PROCEDURE AT ANY TIME BY TYPING NO AFTER THE PROMPT TO
CONTINUE.

TO BEGIN, ENTER NAME OF FIRST PROGRAM YOU WOULD LIKE TO RUN.
ENTER PROGRAM NAMES WITHOUT THE FILE TYPE, FOR EXAMPLE, FILTER, NOT FILTER.FOR.

AFTER YOU FINISH LOOKING AT A PLOT ON THE SCREEN, PRESS THE SPACE BAR TO
CAUSE THE PLOT TO ERASE AND THE NEXT PLOT TO APPEAR.

WHEN USING THE TEKTRONIX PLOTTER, YOU WILL NEED TO DEPRESS THE RETURN BUTTON
TWICE TO GET THE PLOTS TO ERASE.


PLEASE WAIT FOR THE COMPUTER TO COMPILE AND LINK THE PROGRAMS. YOU WILL
RECEIVE A PROMPT WHEN THE COMPUTER IS READY.


FORTRAN POLYROOT
LINK POLYROOT
FORTRAN POLY
LINK POLY
FORTRAN GEN
LINK GEN
FORTRAN FILTER
LINK FILTER,GCSFRA,LATLIB/LIBRARY,SYS$LIBRARY:GCSLIB/LIB,SYS$LIBRARY:GCSTEK/-
LB
FORTRAN LSL
LINK LSL,LATLIB/LIBRARY
PURGE
L1:
$INQUIRE P1 "WHICH PROGRAM TO USE?"
$ASSIGN/USER_MODE SYS$COMMAND: SYS$INPUT:
$RUN 'P1'
$INQUIRE CHECK "ENTER Y(ES) TO CONTINUE"
$IF .NOT.CHECK THEN GOTO LABEL
$GOTO L1
$LABEL:
$INQUIRE CHECK "ENTER Y(ES) FOR HARD COPY OF ALL FILTER COEFFICIENTS."
$IF .NOT.CHECK THEN GOTO LABEL4
$INQUIRE P2 "ENTER NUMBER OF FILE IN WHICH YOU HAVE STORED PREFILTER INFO."
$LIST 'P2'
$INQUIRE P3 "ENTER NUMBER OF FILE IN WHICH YOU HAVE STORED LATTICE INFO."
$LIST 'P3'
$LABEL4:
$INQUIRE CHECK "ENTER Y(ES) TO DELETE ALL DATA FILES."
$IF .NOT.CHECK THEN GOTO LABEL5
$DELETE *.DAT;*
$LABEL5:
$EXIT
```

Program GEN.FOR

```
C*****************************************************************
C THIS PROGRAM HAS BEEN REVISED AS OF 12/6/82. THE LISTINGS SUPPLIED
C WITH ARL INTERNAL MEMORANDUM 82-219, DATED 11/12/82 ARE NO LONGER
C CORRECT. PLEASE OBTAIN NEW LISTINGS.
C *****************************************************************


C INPUT GENERATOR FOR PREFILTER.


C THIS PROGRAM GENERATES COMPLEX GAUSSIAN NOISE WITH VARIABLE STANDARD
C DEVIATION AND COMPLEX IMPULSE SEQUENCE OF VARIABLE FREQUENCY.


        REAL AR(5000), AI(5000), MTHSRANDOM,ANFREQ


        COMPLEX X(5000),APULSE(5000)


        PI = 4.0*ATAN(1.0)
        TWOPI=PI*2

1000    TYPE*,'$ENTER NUMBER OF SAMPLES TO GENERATE.'
        TYPE*,'$THIS NUMBER SHOULD CORRESPOND TO DESIRED NUMBER OF
        1 ITERATIONS OF LATTICE.'
        TYPE*,'$((1<=M<=5000).'
        ACCEPT*,M

        IF(M.GT.5000)THEN
        TYPE*,'$INCORRECT INPUT, TRY AGAIN.'
        GOTO 1000
        END IF

        TYPE*,'$ENTER DESIRED STANDARD DEVIATION OF GAUSSIAN WHITE NOISE.'
        TYPE*,'$ (MUST BE FLOATING POINT).:'

        ACCEPT*,STD



        TYPE*,'$ENTER DESIRED FREQUENCY OF IMPULSE FUNCTION IN HZ.'
        ACCEPT*,NFREQ
        ANFREQ=(FLOAT(NFREQ))
        N=JNINT(10000.0/NFREQ)

 ASSIGN OUTPUT FILE NUMBER.
        NFILE=2


        WRITE(NFILE,12)
2       FORMAT(5X,20HGAUSSIAN NOISE INPUT,13X)


        WRITE(NFILE,600) NFILE
00      FORMAT(5X,41HTHIS INFORMATION IS STORED IN DISK FILE: ,I5)

C INITIALIZE RANDOM NUMBER GENERATOR WITH ARBITRARY LARGE INTEGER.

        K=958749273


C THE NEXT LOOP PREVENTS TAKING ALOG10 OR SQRT OF ZERO.
        DO 100 I=1,M

C MTHSRANDOM IS A UNIFORM RANDOM NOISE GENERATOR.
        AR(I)=MTHSRANDOM(K)
        AI(I)=MTHSRANDOM(K)
        IF(AI(I).EQ.0.0) AI(I)=.001
        IF(AR(I).EQ.0.0) AR(I)=.001
        IF(AR(I).EQ.1.0) AR(I)=.999
        IF(AI(I).EQ.1.0) AI(I)=.999

100     CONTINUE


        NUM1=M/2

C GENERATE GAUSSIAN NOISE.

C THIS LOOP COMPUTES VALUES FOR TWO ADJACENT ARRAY POINTS AT ONE TIME.
        DO 70 J=1,NUM1


        X1=AR(2*J-1)
        X2=AR(2*J)
        Y1=AI(2*J-1)
        Y2=AI(2*J)
        AR(2*J-1)=STD*SQRT(-2.*ALOG(Y1))*COS(TWOPI*Y2)
        AR(2*J)=STD*SQRT(-2.*ALOG(Y1))*SIN(TWOPI*Y2)
        AI(2*J-1)=STD*SQRT(-2.*ALOG(X1))*COS(TWOPI*X2)
        AI(2*J)=STD*SQRT(-2.*ALOG(X1))*SIN(TWOPI*X2)
70      CONTINUE



        DO 95 I=1,M
        AI(I)=0.0
        X(I)=CMPLX(AR(I),AI(I))
95      CONTINUE


C WRITE GAUSSIAN NOISE TO DISK.

        WRITE(7,607)STD,(X(I),I=1,M)
607     FORMAT(5X,5HSTD: ,F12.6/(5X,4F12.6))
        WRITE(NFILE,607)STD,(X(I),I=1,M)
        ENDFILE NFILE

C THE NEXT PORTION OF THE PROGRAM GENERATES IMPULSE SEQUENCE.


C ASSIGN OUTPUT FILE NUMBER.
        NFILE=3


C SET ARRAY TO ZEROES.
```

```
      DO 97 I=1,M
      APULSE(I)=CMPLX(0.0,0.0)
97    CONTINUE

C SET IMPULSES AT IMPULSE POINTS.
      DO 96 I=1,M,N
      APULSE(I)=CMPLX(1.0,0.0)
96    CONTINUE

      WRITE(NFILE,697)AMPDB
697   FORMAT(5X,13HIMPULSE INPUT,1X,4HDB: ,F7.1)

      WRITE(NFILE,600) NFILE

C WRITE IMPULSE FUNCTION TO DISK.

      WRITE(NFILE,603)(APULSE(I),I=1,M)
603   FORMAT(5X,1E12.6)

      ENDFILE NFILE

      TYPE*,'%WHITE NOISE IN FILE 1.'
      TYPE*,'%GAUSSIAN NOISE IN FILE 2.'
      TYPE*,'%IMPULSE INPUT IN FILE 3.'
      TYPE*,'%MAKE SURE TO RECORD THESE NUMBERS.'
      ENDFILE 7
      STOP
      END
```

Program FILTER.FOR

Recursive Realization of Prefilter and Lattice (Inverse) Filter Transfer Functions

```
C*************************************************************************
C THIS PROGRAM HAS BEEN REVISED AS OF 12/6/82. THE LISTINGS SUPPLIED WITH
C ARL INTERNAL MEMORANDUM 82-219, DATED 11/12/82 ARE NO LONGER CORRECT.
C PLEASE OBTAIN A NEW LISTING.
C*************************************************************************

C THIS PREFILTER IS A RECURSIVE REALIZATION OF THE FOLLOWING TRANSFER
C FUNCTION:

C        H(Z)=B(Z)/A(Z)

C WHERE   A(Z)=1+A1*Z**-1+A2*Z**-2+. . .+ AP*Z**-P

C AND     B(Z)=1+B1*Z**-1+B2*Z**-2+. . . BP*Z**-P

C THEN, IN THE TIME DOMAIN:

C   Z(N)=-A1*Z(N-1)-A2*Z(N-2)-. . .-AP*Z(N-P)+V(N)+B1*V(N-1)+. . .+BP*V(N-P)

C P IS THE FILTER ORDER.
C THE MAXIMUM VALUE OF P IS 16.
C FOR THE TRANSFER FUNCTION AND DEVELOPMENT OF
C THE RECURSIVE REALIZATION, SEE TEXT MATERIAL.
C*************************************************************************

        REAL F(5000),PTS(1),OPTS(1),VV(5000),ZZZ(5000)
        1,R(5000),SSR,S(5000),SSP,FF(8),BBW(8)
        1,T(5000),E(5000),SD,FF(5000),ZREG(5000)
        1,RREG(5000),VREG(5000),STDPA,RA,VVREG(5000)
        1,II(5000),VVV(5000),ZZZZ(5000),FFF(5000)
        1,RREG(5000),SREG(5000),SS(5000),ZZREG(5000)
        1,FFREG(5000),FORM(8),BW(8),BBB(17),CCC(17)

        COMPLEX V(5000),Z(5000),A(16),B(16),
        1ZI(16),VI(16),AA(16),BB(16),AH(5000),HH(5000)
        1,P,W,J,WT,SI(5000),WJ,HN(5000),FC(5000)

        DATA OPTS/'FILTER'/

        PI=4.0*ATAN(1.0)

        FS=10000.0
        IFS=INT(FS/2.0)

100     TYPE*,'ENTER 1 IF YOU WOULD LIKE TO READ CALCULATED INVERSE
        1FILTER COEFFICIENTS FROM DISK.'

        TYPE*,'ENTER 0 IF YOU WOULD LIKE TO TYPE IN FILTER COEFFICIENTS.'
        TYPE*,'ENTER 2 IF YOU WOULD LIKE TO SPECIFY FORMANTS AND
        1 BANDWIDTHS.'
        TYPE*,'ENTER 3 IF YOU WOULD LIKE TO READ IN PREFILTER COEFFICIENTS
        1 FROM DISK.'
        ACCEPT*,NN

        IF(NN.NE.0.AND.NN.NE.1.AND.NN.NE.2.AND.NN.NE.3)THEN

        TYPE*,'INCORRECT INPUT, TRY AGAIN.'
        GOTO 100
        END IF

        IF(NN.EQ.1) GOTO 79

1500    TYPE*,'ENTER NUMBER OF SAMPLES TO ANALYZE.'

        TYPE*,'THIS NUMBER SHOULD CORRESPOND TO DESIRED NUMBER OF
        1 ITERATIONS OF THE LATTICE (M<=5000).'

        ACCEPT*,M

        IF(M.GT.5000)THEN
        TYPE*,'INCORRECT INPUT, TRY AGAIN.'
        GOTO 1500
        END IF

1300    TYPE*,'IF YOU WISH TO SPECIFY FORMANTS AND BANDWIDTHS, THERE WILL
        1 BE TWO (COMPLEX CONJUGATE) POLES FOR EACH FORMANT.'
        TYPE *,'ENTER THE NUMBER OF POLES (1<=NPOLE<=16).'

        ACCEPT*,NPOLE

        IF(NPOLE.GT.16.OR.NPOLE.EQ.0) THEN
        TYPE*,'INCORRECT INPUT, TRY AGAIN.'
        GOTO 1300
        END IF

        IF (NPOLE.EQ.0) GOTO 297
        IF(NN.EQ.0) GOTO 407
        IF(NN.EQ.3)GOTO 445

C ENTER FORMANT FREQUENCIES AND THEIR BANDWIDTHS.

        DO 111 I=1,(NPOLE/2)
        WRITE(6,119) I
119     FORMAT(5X,3HI= ,I5)
        TYPE*,'ENTER FORMANT FREQUENCY IN HZ.'
        ACCEPT*,FORM(I)
        FF(I)=FORM(I)
        TYPE*,'ENTER BANDWIDTH OF THIS FORMANT FREQUENCY IN HZ.'
        ACCEPT*,BW(I)
        BBW(I)=BW(I)
111     CONTINUE

C COMPUTE COMPLEX PREFILTER COEFFICIENTS.
C N IS THE ORDER OF THE POLYNOMIAL IN Z**-1.
C FF IS AN ARRAY OF FORMANT FREQUENCIES.
C BBW IS AN ARRAY OF THEIR CORRESPONDING BANDWIDTHS.
C BBB (OUTPUT) IS AN ARRAY OF THE REAL PARTS OF THE FILTER COEFFICIENTS.
C CCC (OUTPUT) IS AN ARRAY OF THE IMAGINARY PARTS OF THE FILTER COEFFICIENTS.
```

```
      CALL SYNTHE(NPOLE,FF,BW,RRR,CCC,FS)
      TYPE*,'FILTER COEFFICIENTS IN ORDER OF INCREASING POWER.'
      TYPE*,'A(0)=1.0'
      DO 230 II=1,NPOLE
      A(II)=C REAL(CMPLX(RRR(II)),CCC(II)))
      WRITE(6,130)A(II)
230   CONTINUE
130   FORMAT(5X,2E12.6)

C OPTION TO SAVE THESE PREFILTER FORMANTS, BANDWIDTHS, AND CORRESPONDING
C FILTER COEFFICIENTS ON DISK FOR FUTURE USE.

      TYPE*,'ENTER 1 IF YOU WOULD LIKE TO SAVE THIS INFORMATION ON DISK.'
      TYPE*,'OTHERWISE, ENTER 0.'
      ACCEPT*,NSAVE
      IF(NSAVE.EQ.1)THEN
      TYPE*,'ENTER NUMBER OF DISK FILE TO SAVE THIS IN.'
      TYPE*,'USE A FILE NUMBER BETWEEN 40 AND 60, INCLUSIVE.'
      ACCEPT*,NFORM
      WRITE(NFORM,898)NFORM
      WRITE(NFORM,321)
      WRITE(NFORM,301)
301   FORMAT(5X,7HFORMANT,10X,9HBANDWIDTH)
      DO 220 K=1,(NPOLE/2)
      WRITE(NFORM,302)FORM(K),BW(K)
302   FORMAT(5X,F7.2,10X,F6.2)
220   CONTINUE
      WRITE(NFORM,304)
304   FORMAT(5X,45HFILTER COEFFICIENTS (INCREASING POWER, A0=1).)
      DO 221 K=1,NPOLE
      WRITE(NFORM,303) A(K)
303   FORMAT(5X,2E12.6)
221   CONTINUE
      ENDFILE NFORM
      GOTO 231
      END IF
      GOTO 146
45    CONTINUE

 OPTION TO READ IN STORED PREFILTER COEFFICIENTS FROM DISK.

      TYPE*,'FROM WHAT DISK FILE WOULD YOU LIKE TO READ
     1 FILTER COEFFICIENTS ?'
      TYPE*,'THIS WILL BE A FILE NUMBER BETWEEN 40 AND 60, INCLUSIVE.'
      ACCEPT*,NDISK
      READ(NDISK,203)
03    FORMAT(51X)
      NFORM=NDISK
      READ(NDISK,323)
23    FORMAT(14X)
      READ(NDISK,324)
24    FORMAT(31X)
      DO 322 K=1,(NPOLE/2)
      READ(NDISK,320)FORM(K),BW(K)
20    FORMAT(5X,F7.2,10X,F6.2)
22    CONTINUE
      READ(NDISK,145)
15    FORMAT(50X)
      DO 823 K=1,NPOLE
      READ(NDISK,433)A(K)
433   FORMAT(5X,2E12.6)
823   CONTINUE

C DISPLAY FORMANTS, BANDWIDTHS, AND COEFFICIENTS TO SCREEN.
C IF YOU'RE USING A TEKTRONIX TERMINAL, MAKE A HARD COPY OF THE SCREEN.

231   CALL INITT
      CALL ERASE
      CALL HOME
      WRITE(6,898)NFORM
898   FORMAT(5X,36HTHIS INFORMATION STORED IN DISK FILE,5X,I5)
      WRITE(6,321)
321   FORMAT(5X,9HPREFILTER)
      WRITE(6,301)
      DO 420 I=1,(NPOLE/2)
      WRITE(6,302)FORM(I),BW(I)
420   CONTINUE
      WRITE(6,304)
      DO 421 K=1,NPOLE
      WRITE(6,303) A(K)
421   CONTINUE


146   NZERO=0
      GOTO 11

C TYPE IN THE FILTER COEFFICIENTS FOR THE PREFILTER.

907   CONTINUE

      TYPE *,'SELECT DENOMINATOR COEFFICIENTS, (AS REAL NUMBERS) A1 THROUGH
     1 AP.'

      TYPE*,'MAKE SURE YOU HAVE READ THE COMMENT STATEMENTS CAREFULLY BEFORE
     1 YOU TYPE IN THE FILTER COEFFICIENTS.'

C TYPE IN DENOMINATOR COEFFICIENTS.
      DO 219 I=1,NPOLE
      WRITE(6,17)I
      TYPE*,'ENTER A.'
      ACCEPT*,AI
      A(I)=CMPLX(AI,0.0)
219   CONTINUE
17    FORMAT(5X,4HFOR ,I5)
987   CONTINUE

4200  TYPE*,'ENTER NUMBER OF ZEROS (1<=NZERO<=16).'
      ACCEPT*,NZERO

      IF(NZERO.GT.16) THEN
      TYPE*,'INCORRECT INPUT, TRY AGAIN.'
      GOTO 4200
      END IF
```

```
      IF(NZERO.EQ.0) GOTO 41
C IF THE PREFILTER HAS ANY ZEROES, TYPE THE NUMERATOR COEFFICIENTS IN.

      TYPE*,'ENTER NUMERATOR COEFFICIENTS, (AS REAL NUMBERS) B1 THROUGH
     1 BP.'

      DO 43 I=1,NZERO
      WRITE(6,17)I
      TYPE*,'ENTER B.'
      ACCEPT*,BI
      B(I)=CMPLX(BI,0.0)
43    CONTINUE
41    CONTINUE

C ENTER TYPE OF INPUT FOR PREFILTER.

600   TYPE*,'ENTER DESIRED INPUT FILE NUMBER.'
      TYPE*,'NOISE INPUT=FILE 2'
      TYPE*,'SINUSOIDAL INPUT=FILE 3'

      ACCEPT*,NFILE

      IF(NFILE.NE.2.AND.NFILE.NE.3)THEN
      TYPE*,'INCORRECT INPUT, TRY AGAIN.'
      GOTO 600
      END IF

C ASSIGN OUTPUT FILE NUMBER.
      LOUT=21
      WRITE(6,11)LOUT
11    FORMAT(5X,'OUTPUT IN FILE: ',I5)

      TYPE*,'BE SURE TO RECORD THIS NUMBER.'

      IF(NFILE.EQ.0.OR.NFILE.EQ.2.OR.NFILE.EQ.3) GOTO 62

C START HERE TO READ IN INVERSE FILTER COEFFICIENTS FROM DISK (THESE
C WILL BE THE OUTPUTS OF THE LATTICE FILTER).

      CONTINUE
      NFILE=21
      NCOEF=11
      NF=16

      READ(NF,251)
1     FORMAT(5I6)
      READ(NF,112)
2     FORMAT(2A6)

      READ (NCOEF,186)
186   FORMAT(5I6)

      READ(NCOEF,189)NZERO,N
189   FORMAT(8X,I5,9X,I5)

C READ IN ERROR SIGNAL FROM DISK.

      READ(NF,113)ALPHA,SNR,(EC(I),I=1,N)
113   FORMAT(10X,F9.6,10X,F12.6,17X/(5X,2F15.6))

C COMPUTE RANGE OF ERROR SIGNAL FOR PLOTTING PURPOSES.

      DO 236 I=1,N
      EE(I)=REAL(EC(I))
      ENEG(I)=-1.0*EE(I)
      T(I)=FLOAT(I)
236   CONTINUE

      EMAX=ARRMAX(EE,N)
      EMIN=-1.0*(ARRMAX(ENEG,N))

      IF (NZERO.EQ.0) GOTO 61

C READ INVERSE FILTER PREDICTOR COEFFICIENTS FROM DISK.
      READ(NCOEF,187)(B(J),J=1,NZERO)
187   FORMAT(11X/(5X,3E12.6))

61    IF (NPOLE.EQ.0) GOTO 623

C IF THE INVERSE FILTER HAS ANY POLES, READ IN THE DENOMINATOR COEFFICIENTS.
      READ(NCOEF,187)(A(J),J=1,NPOLE)
623   CONTINUE

C DETERMINE WHICH FILE WAS THE INPUT TO THE PREFILTER.
      READ(NFILE,622)NFILE
622   FORMAT(17X,I5)

62    CONTINUE

      IF(NFILE.EQ.3) THEN
      READ(NFILE,663)F0
663   FORMAT(2X,F7.3)
      GOTO 666
      END IF
      READ(NFILE,69)
69    FORMAT(3I1)

666   CONTINUE
      READ(NFILE,63)
63    FORMAT(5I6)

      IF(NFILE.EQ.4.OR.NFILE.EQ.3) GOTO 787
      IF(NFILE.EQ.2.OR.NFILE.EQ.2) THEN

C READ IN INPUT SIGNAL FROM DISK.
C THIS IS FOR EITHER PREFILTER OR LATTICE FILTER.
```

```
C FOR STOCHASTIC SIGNAL....
        READ(NFILE,87)STD,(V(I),I=1,M)
87      FORMAT(10X,F12.6/(5X,4F12.6))

        IF(NFILE.EQ.21)THEN
        READ(NFILE,77)(SS(I),I=1,IFS)
77      FORMAT(5X,F12.6)
        VAR=1.0
        GOTO 271
        END IF
        VAR=STD**2
        GOTO 271
        END IF
797     CONTINUE

C FOR DETERMINISTIC SIGNAL, READ THIS INSTEAD.
        READ(NFILE,7) (V(I),I=1,M)
7       FORMAT (5X,4F12.6)
        VAR=ARRVAR(V,M)
        GOTO 271

271     CONTINUE


        IF (NZERO.GT.NPOLE) NN=NZERO
        IF (NPOLE.GE.NZERO) NN=NPOLE

        ANPOLE=(FLOAT(NPOLE))
        ANZERO=(FLOAT(NZERO))
        AM=(FLOAT(M))

C INITIALIZE THE FIRST FEW OUTPUTS TO ZERO.
        DO 30 N=1,NN
        Z(N)=CMPLX(0.0,0.0)
30      CONTINUE

C START THE RECURSIVE REALIZATION.

        DO 20 I=NN+1,M
        Z(I)=V(I)

C FOR DENOMINATOR COEFFICIENTS..
        IF(NPOLE.EQ.0) GOTO 18
        DO 19 J=1,NPOLE
        ZI(J)=Z(I-J)
        AI(J)=A(J)*ZI(J)
        Z(I)=Z(I)-AI(J)
19      CONTINUE
18      CONTINUE

C FOR NUMERATOR COEFFICIENTS..
        IF(NZERO.EQ.0) GOTO 41
        DO 42 J=1,NZERO
        VI(J)=V(I-J)
        BB(I)=B(J)*VI(J)
        Z(I)=Z(I)+BB(I)
42      CONTINUE
41      CONTINUE
20      CONTINUE

        IF(NFILE.EQ.21) GOTO 799

        WRITE(NOUT,109) NFILE
109     FORMAT(5X,'20HINPUT FILE: ',I5)

        WRITE(NOUT,116)
116     FORMAT(5X,'29HFILTER TIME DOMAIN OUTPUT')

        WRITE(NOUT,13)NOUT
13      FORMAT(5X,'41HTHIS INFORMATION IS STORED IN DISK FILE: ',I5)
799     CONTINUE


        IF(NFILE.EQ.1.OR.NFILE.EQ.3) GOTO 206
803     IF(NFILE.GE.2.OR.NFILE.LE.21) THEN

C COMPUTE THE STANDARD DEVIATION OF THE OUTPUT SIGNAL.
        SMI=0.0
        SMR=0.0
        DO 38 I=1,M
        SMR=SMR+(REAL(Z(I)))**2
        SMI=SMI+(AIMAG(Z(I)))**2
38      CONTINUE
        STDR=SMR/M
        STDI=SMI/M
        AVAR=STDR+STDI
        STDZ=SQRT(AVAR)

C COMPUTE GAIN FOR A STOCHASTIC SIGNAL.
        GAIN=10*ALOG10(AVAR/(STD**2))
        IF(NFILE.EQ.21) GOTO 211

C WRITE Z(N) TO DISK.
        WRITE(NOUT,16)STDZ,(Z(I),I=1,M)
16      FORMAT(5X,'5HSTD: ',F12.6/(5X,4F12.6))
        GOTO 211
        END IF
206     CONTINUE

211     CONTINUE


C COMPUTE MAXIMUM AND MINIMUM POINTS IN V(N) AND Z(N).
        DO 25 I=1,M
        ZZZ(I)=REAL(Z(I))
        VV(I)=REAL(V(I))
        ZNEG(I)=-1.0*ZZZ(I)
        VNEG(I)=-1.0*VV(I)
        T(I)=(FLOAT(I))
25      CONTINUE
        VMIN=-1.0*(ARRMAX(VNEG,M))
        VMAX=ARRMAX(VV,M)
        ZMAX=ARRMAX(ZZZ,M)
        ZMIN=-1.0*(ARRMAX(ZNEG,M))


        IF(NFILE.EQ.2)GOTO 333

        IF(NFILE.GE.3.OR.NFILE.GE.3) THEN
C COMPUTE GAIN FOR A DETERMINISTIC SIGNAL.
```

```
      DMI=0.0
      DMR=0.0
      VMR=0.0
      VMI=0.0

      DO 226 I=1,N
      DMI=DMI+(AIMAG(Z(I)))**2
      DMR=DMR+(REAL(Z(I)))**2
      VMR=VMR+(REAL(V(I)))**2
      VMI=VMI+(AIMAG(V(I)))**2
226   CONTINUE
      DMR=DMR/N
      DMI=DMI/N
      VMR=VMR/N
      VMI=VMI/N
      CNR=10*ALOG10((DMR+DMI)/(VMR+VMI))

      IF(NFILE.EQ.2)THEN

C FOR A DETERMINISTIC SIGNAL, WRITE Z(N) TO DISK.
      WRITE(UNIT,216)(Z(I),I=1,N)
216   FORMAT(5X,2E12.6)
      END IF
      END IF




C PLOT INPUT, V(N).

13    CONTINUE

      IF(NFILE.EQ.2)) GOTO 676

      TYPE*,'ENTER 1 IF YOU WOULD LIKE TO PLOT INPUT TIME SERIES.'
      TYPE*,'ENTER 0 OTHERWISE.'
      ACCEPT*,NIN
      IF(NIN.NE.1) GOTO 676

C ASSIGN NUMBER OF POINTS TO PLOT.
14    CONTINUE
      TYPE*,'ENTER FIRST AND LAST TIME POINTS OF INPUT TO PLOT.'
      WRITE(6,702)
702   FORMAT(5X,6H(1<N<2),I5)
      ACCEPT*,N1,N2
      AN1=(FLOAT(N1))
      AN2=(FLOAT(N2))
      TYPE*,'ENTER INTERVAL BETWEEN POINTS ON PLOT.'
      ACCEPT*,NSKIP
      PTS(1)=(FLOAT((N2-N1+NSKIP)/NSKIP))
      NLENGTH=PTS(1)
      K=1
      DO 102 I=N1,N2,NSKIP
      VVV(K)=V(I)
      TT(K)=T(I)
      VVNEG(K)=-1.0*VVV(K)
      K=K+1
102   CONTINUE

      IF(NFLAG.EQ.2)GOTO 292

      VMAX=ARRMAX(VVV,NLENGTH)
      VMIN=-1.0*ARRMAX(VVNEG,NLENGTH)

266   CONTINUE
      IF(NFILE.EQ.2)GOTO 949
      IF(NFILE.EQ.3)GOTO 910

C THIS PLOT FOR DIGITAL INPUT
      CALL [illegible]
      CALL [illegible](0.0,7.0,0.0,5.0)
      CALL [illegible]('TEXT')
      CALL [illegible](0.0,100.0,0.0,100.0)
      CALL [illegible](0.0,80.0,'PREFILTER INPUT:\')
      CALL [illegible](0.0,75.0,'DIGITAL EXCITATION\')
      CALL [illegible](0.0,70.0,'FO:\')
      CALL [illegible](0.0,60.0,'FS IN HZ.:\')
      CALL RESET('[illegible]')
      CALL [illegible](20.0,70.0,FO)
      CALL [illegible](20.0,60.0,FS)
      CALL [illegible](2.0,7.0,0.0,5.0)
      CALL [illegible]('XBOTH')
      CALL [illegible]('YBOTH')
      CALL [illegible]('XLABEL','TIME\')
      CALL [illegible]('YLABEL',' INPUT AMPLITUDE\')
      CALL [illegible](AN1,AN2,VMIN,VMAX)
      CALL [illegible]('SCALE')
      CALL [illegible](TT,VVV,1.0,PTS,PTS)
      CALL [illegible](DUMMY)
      READ(5,501) DUMMY
      CALL [illegible]
      GOTO 500


C THIS PLOT FOR STOCHASTIC INPUT
910   CONTINUE
      CALL [illegible]
      CALL [illegible](0.0,7.0,0.0,5.0)
      CALL [illegible]('TEXT')
      CALL [illegible](0.0,100.0,0.0,100.0)
      CALL [illegible](0.0,80.0,'PREFILTER INPUT:\')
      CALL [illegible](0.0,75.0,'GAUSSIAN WHITE NOISE\')
      CALL [illegible](0.0,70.0,'STD. DEV.: \')
      CALL [illegible](0.0,60.0,'FS IN HZ.:\')
      CALL [illegible]('[illegible]')
      CALL [illegible](20.0,60.0,FS)
      CALL [illegible]('[illegible]')
      CALL [illegible](20.0,70.0,STD)
      CALL [illegible](2.0,7.0,0.0,5.0)
      CALL [illegible]('XBOTH')
      CALL [illegible]('YBOTH')
      CALL [illegible]('XLABEL','TIME\')
      CALL [illegible]('YLABEL',' INPUT AMPLITUDE\')
      CALL [illegible](AN1,AN2,VMIN,VMAX)
      CALL [illegible]('SCALE')
      CALL [illegible](TT,VVV,1.0,PTS,PTS)
      CALL [illegible](DUMMY)
      READ(5,501) DUMMY
      CALL [illegible]
      GOTO 500
```

```
C THIS PLOT FOR IMPULSE INPUT
404     CONTINUE
        CALL USTART
        CALL UDAREA(0.0,7.0,0.0,5.0)
        CALL USET('TEXT')
        CALL UWINDO(0.0,100.0,0.0,100.0)
        CALL UPRINT(0.0,80.0,'PREFILTER INPUT:\')
        CALL UPRINT(0.0,75.0,'IMPULSE\')
        CALL UPRINT(0.0,70.0,'FO IN HZ.:\')
        CALL UPRINT(0.0,60.0,'FS IN HZ.:\')
        CALL USET('INTEGER')
        CALL UPRINT(20.0,70.0,F0)
        CALL UPRINT(20.0,60.0,FS)
        CALL UDAREA(2.0,7.0,0.0,5.0)
        CALL USET('XBOTH')
        CALL USET('YBOTH')
        CALL USET('XLABEL','TIME\')
        CALL USET('YLABEL',' INPUT AMPLITUDE\')
        CALL UWINDO(AN1,AN2,VMIN,VMAX)
        CALL USET('OWNSCALE')
        CALL UPLOT(TT,VMV,1.0,PTS,PLTS)
        CALL LAINE(IUNIT)
        READ(5,501) DUMMY
        CALL UEND

500     CONTINUE
501     FORMAT(A1)

        CALL USTART
        CALL UERASE
        CALL UEND

C OPTION TO REPLOT WITH DIFFERENT SCALES OR DIFFERENT TIME SEGMENT.

        TYPE*,'ENTER 1 IF YOU WOULD LIKE TO PLOT THE INPUT AGAIN ON A
     1 DIFFERENT SCALE,'
        TYPE*,'OTHERWISE, TYPE IN 0.'

        ACCEPT*,NINPUT

        IF(NINPUT.EQ.1) THEN
        NFLAG=2
        TYPE*,'ENTER MAXIMUM VERTICAL SCALE LIMIT.'
        ACCEPT*,VMAX
        TYPE*,'ENTER MINIMUM VERTICAL SCALE LIMIT.'
        ACCEPT*,VMIN
        GOTO 334
        END IF
676     CONTINUE

C PLOT OUTPUT,Z(N).

        TYPE*,'ENTER 1 IF YOU WOULD LIKE TO PLOT OUTPUT TIME SERIES.'
        TYPE*,'OTHERWISE ENTER 0.'
        ACCEPT*,NOP
        IF(NOP.EQ.1.AND.NFILE.EQ.2) GOTO 348
        IF(NOP.NE.1.AND.NFILE.NE.2) GOTO 599

677     CONTINUE

C COMPUTE NUMBER OF POINTS, AND RANGES OF PLOT.

        TYPE*,'ENTER FIRST AND LAST TIME POINTS OF OUTPUT TO PLOT.'
        WRITE(6,722)
        ACCEPT*,N1,N2
        AN1=(FLOAT(N1))
        AN2=(FLOAT(N2))
        TYPE*,'ENTER INTERVAL BETWEEN POINTS ON PLOT.'
        ACCEPT*,NSKIP
        PTS(1)=(FLOAT((N2-N1+NSKIP)/NSKIP))
        NLENGTH=PTS(1)
        K=1
        DO 702 I=N1,N2,NSKIP
        ZZZZ(K)=ZZZ(I)
        TT(K)=T(I)
        ZZNEG(K)=-1.0*ZZZZ(K)
        K=K+1
702     CONTINUE

        IF(NFILE.EQ.2) GOTO 339

        ZMAX=ARRMAX(ZZZZ,NLENGTH)
        ZMIN=-1.0*ARRMAX(ZZNEG,NLENGTH)

339     CONTINUE
        IF(NFILE.EQ.2) GOTO 565

C THIS PLOT FOR PREFILTER
        CALL USTART
        CALL UDAREA(0.0,7.0,0.0,5.0)
        CALL USET('TEXT')
        CALL UWINDO(0.0,100.0,0.0,100.0)
        CALL UPRINT(0.0,80.0,'PREFILTER OUTPUT\')
        IF(NFILE.EQ.2)THEN
        CALL UPRINT(0.0,75.0,'FOR NOISE INPUT\')
        END IF
        IF(NFILE.EQ.3) THEN
        CALL UPRINT(0.0,75.0,'FOR IMPULSE INPUT\')
        END IF
        IF(NFILE.EQ.4) THEN
        CALL UPRINT(0.0,75.0,'FOR GLOTTAL EXCITATION INPUT\')
        END IF
        CALL UPRINT(0.0,70.0,'PP GAIN:\')
        CALL UPRINT(0.0,60.0,'NPOLE=\')
        CALL UPRINT(0.0,55.0,'NZERO=\')
        CALL UPRINT(0.0,45.5,'ITERATIONS:\')
        CALL USET('REAL NUMBER')
        CALL UPRINT(20.0,70.0,GAIN)
        CALL USET('INTEGER')
        CALL UPRINT(20.0,60.0,ANPOLE)
        CALL UPRINT(20.0,55.0,ANZERO)
        CALL UPRINT(20.0,45.0,AN)
        CALL UDAREA(2.0,7.0,0.0,5.0)
        CALL USET('XBOTH')
        CALL USET('YBOTH')
```

```
        CALL UDSKEY('YLABEL','TIME\')
        CALL UDSKEY('YLABEL',' OUTPUT AMPLITUDE\')
        CALL USET('NO*SCALE')
        CALL USTPC(AT1,AT2,ZMIN,ZMAX)
        CALL UPLOT(TT,ZZZZ,1,0,PTS,LPTS)
        CALL UAI(DUMMY)
        READ(5,501) DUMMY
        CALL UEND
        GOTO 566

 THIS PLOT FOR LATTICE FILTER
65      CONTINUE
        CALL USTART
        CALL UDAREA(0.0,7.0,0.0,5.0)
        CALL USET('TEXT')
        CALL UWINDO(0.0,100.0,0.0,100.0)
        CALL UPRINT(0.0,95.0,'LATTICE(INVERSE)\')
        CALL UPRINT(0.0,80.0,'FILTER OUTPUT\')
        CALL UPRINT(0.0,75.0,'PREFILTER INPUT:\')
        IF(NFILT.EQ.2)THEN
        CALL UPRINT(0.0,70.0,'NOISE\')
        END IF
        IF(NFILT.EQ.3)THEN
        CALL UPRINT(0.0,70.0,'IMPULSE\')
        END IF
        IF(NFILT.EQ.4) THEN
        CALL UPRINT(0.0,70.0,'GLOTTAL\')
        END IF
        CALL UPRINT(0.0,65.0,'DB GAIN:\')
        CALL UPRINT(0.0,55.0,'IZERO=\')
        CALL UPRINT(0.0,45.0,'ALPHA:\')
        CALL UPRINT(0.0,40.0,'SNR:\')
        CALL UPRINT(0.0,35.0,'ITERATIONS:\')
        CALL USET('REAL NUMBER')
        CALL UPRINT(20.0,65.0,GAIN)
        CALL UPRINT(20.0,45.0,ALPHA)
        CALL UPRINT(20.0,40.0,SNR)
        CALL USET('INTEGER')
        CALL UPRINT(20.0,55.0,IZERO)
        CALL UPRINT(20.0,35.0,NP)
        CALL UDAREA(2.0,7.0,0.0,5.0)
        CALL USET('XBOTH')
        CALL USET('YBOTH')
        CALL UDSKEY('YLABEL','TIME\')
        CALL UDSKEY('YLABEL',' OUTPUT AMPLITUDE\')
        CALL USET('NO*SCALE')
        CALL USTPC(AT1,AT2,ZMIN,ZMAX)
        CALL UPLOT(TT,ZZZZ,1,0,PTS,LPTS)
        CALL UAI(DUMMY)
        READ(5,501) DUMMY
        CALL UEND

 6      CONTINUE

        CALL USTART
        CALL UERASE
        CALL UEXIT

 OPTION TO REPLOT ON DIFFERENT SCALE OR DIFFERENT TIME SEGMENT.

        TYPE*,'ENTER 1 IF YOU WOULD LIKE TO PLOT THE OUTPUT AGAIN
        1 ON A DIFFERENT SCALE.'
        TYPE*,'OTHERWISE, TYPE IN 0.'

        ACCEPT*,OUTPUT

        IF(OUTPUT.EQ.1) THEN
        FLAG=2
        TYPE*,'ENTER MAXIMUM VERTICAL SCALE POINT.'
        ACCEPT*,ZMAX
        TYPE*,'ENTER MINIMUM VERTICAL SCALE POINT.'
        ACCEPT*,ZMIN
        GOTO 677
        END IF

600     CONTINUE
        IF(NFILT.NE.2)GOTO 599

C COMPUTE GAIN FOR THE LATTICE FILTER.
        EMI=0.0
        EMI=0.0
        VMR=0.0
        VMI=0.0

        DO 229 I=1,N
        EMI=EMI+(AIMAG(EC(I)))**2
        EMR=EMR+(REAL(EC(I)))**2
        VMR=VMR+(REAL(V(I)))**2
        VMI=VMI+(AIMAG(V(I)))**2
229     CONTINUE
        EMR=EMR/N
        EMI=EMI/N
        VMR=VMR/N
        VMI=VMI/N
        GAIN=10*ALOG10((EMR+EMI)/(VMR+VMI))

C PLOT ERROR SIGNAL.

338     CONTINUE

        TYPE*,'ENTER 1 IF YOU WOULD LIKE TO PLOT ERROR SIGNAL.'
        TYPE*,'OTHERWISE ENTER 0.'
        ACCEPT*,ERR
        IF (ERR.NE.1) GOTO 599

339     CONTINUE

C COMPUTE NUMBER OF POINTS AND RANGES OF PLOT.

        TYPE*,'ENTER FIRST AND LAST TIME POINTS OF ERROR TO PLOT.'
        WRITE(6,7021)N
        ACCEPT*,T1,T2
        AT1=FLOAT(T1)
```

```
      AN2=(FLOAT(N2))
      TYPE*,'ENTER INTERVAL BETWEEN POINTS ON PLOT.'
      ACCEPT*,NSKIP
      PTS(1)=(FLOAT((N2-N1+NSKIP)/NSKIP))
      [illegible]
      K=1
      DO 102 I=N1,N2,NSKIP
      EEE(K)=E(I)
      TT(K)=T(I)
      ENEG(K)=-1.0*EEE(K)
      K=K+1
102   CONTINUE

      IF(IFLAT.EQ.2) GOTO 150

      EMAX=[illegible](EEE, [illegible])
      EMIN=-1.0*[illegible]

150   CALL HSTART
      CALL [illegible](0.0,2.0,0.0,0.5,0)
      CALL [illegible]('[illegible]')
      CALL [illegible](0.0,100.0,[illegible],0,100.0)
      CALL [illegible](0.0,40.0,'[illegible]\')
      CALL [illegible](0.0,[illegible],'[illegible]\')
      IF([illegible].EQ.2)[illegible]
      CALL [illegible](0.0,65.0,'[illegible]\')
      END IF
      IF([illegible].EQ.1)[illegible]
      CALL [illegible](0.0,65.0,'[illegible]\')
      END IF
      IF([illegible].EQ.1)[illegible]
      CALL [illegible](0.0,65.0,'[illegible]\')
      END IF
      CALL [illegible](0.0,60.0,'[illegible]\')
      CALL [illegible](0.0,55.0,'[illegible]\')
      CALL [illegible](0.0,45.0,'ALPHA:\')
      CALL [illegible](0.0,40.0,'[illegible]\')
      CALL [illegible](0.0,35.0,'ITERATIONS:\')
      CALL [illegible]
      CALL [illegible](20.0,60.0,[illegible])
      CALL [illegible](20.0,45.0,ALPHA)
      CALL [illegible](20.0,40.0,[illegible])
      CALL [illegible]('INTEGER')
      CALL [illegible](20.0,55.0,[illegible])
      CALL [illegible](20.0,35.0,[illegible])
      CALL [illegible](2.0,7.0,0.0,5.0)
      CALL [illegible]('[illegible]')
      CALL [illegible]('[illegible]')
      CALL [illegible]('XLABEL','TIME\')
      CALL [illegible]('YLABEL','ERROR SIGNAL\')
      CALL [illegible]('[illegible]SCALE')
      CALL HWINDO(AN1,AN2,EMIN,EMAX)
      CALL [illegible](TT,EEE,1,0,PTS,[illegible])
      CALL [illegible](DUMMY)
      READ(5,501) DUMMY
      CALL [illegible]

      CALL HSTART
      CALL HERASE
      CALL [illegible]

C [illegible] TO [illegible] ANOTHER SCALE OR DIFFERENT TIME SEGMENT.

      TYPE*,'[illegible] 1 IF YOU WOULD LIKE TO PLOT ERROR AGAIN ON ANOTHER
     1 SCALE.'
      TYPE*,'[illegible], TYPE IN 0.'

      ACCEPT*,[illegible]
      IF([illegible].EQ.1) THEN
      IFLAG=2
      TYPE*,'ENTER THE MAXIMUM VERTICAL SCALE POINT.'
      ACCEPT*,EMAX
      TYPE*,'ENTER THE MINIMUM VERTICAL SCALE POINT.'
      ACCEPT*,EMIN
      GOTO [illegible]
      END IF

599   CONTINUE

C COMPUTE THEORETICAL TRANSFER FUNCTION AND OUTPUT POWER SPECTRUM.

C FOR POLES IN THE TRANSFER FUNCTION...
      DO 227 I=1,IFS
      IF([illegible].EQ.0)GOTO 143
      F(I)=(FLOAT(I))/IFS
      W(I)=2.0*[illegible]*F(I)
      AN(I)=CMPLX(1.0,0.0)
      AD(I)=CMPLX(1.0,0.0)
      DO 201 J=1,[illegible]
      [illegible]=-1.0*[illegible](I)*(FLOAT(J))
      [illegible]=W(I)*(FLOAT(J))
      [illegible]=CMPLX(0.0,[illegible])
      [illegible]=CMPLX(0.0,[illegible])
      [illegible]=CEXP([illegible])
      [illegible]=CEXP([illegible])
      [illegible](I)=[illegible](I)+A(J)*[illegible]
      [illegible](I)=[illegible](I)+A(J)*[illegible]
201   CONTINUE

C     FOR POLES ONLY, NO ZEROES IN THE TRANSFER FUNCTION....
      IF([illegible].NE.0) GOTO 143
      SSA=REAL(AD(I)*AD(I))
      [illegible]=[illegible](SSA)
      IF ([illegible].EQ.0.0.OR.SSA.EQ.0.0) THEN
      H(I)=0.0
      S(I)=0.0
      GOTO 227
      END IF
      H(I)=20.0*ALOG10(1.0/SQ)
      S(I)=10.0*ALOG10(VAR/SSA)
227   CONTINUE
143   CONTINUE

C FOR ZEROES IN THE TRANSFER FUNCTION...
      IF([illegible].EQ.0) GOTO 157
```

```
      F(I)=(FLOAT(I))/IFS
      W(I)=2.0*PI*F(I)
      HN(I)=CMPLX(1.0,0.0)
      HD(I)=CMPLX(1.0,0.0)
      DO 226 L=1,NZERO
      WI=-1.0*W(I)*(FLOAT(L))
      WR=W(I)*(FLOAT(L))
      WI=CMPLX(0.0,WI)
      WR=CMPLX(0.0,WR)
      H=CEXP(WI)
      HN=CEXP(WR)
      HN(I)=HN(I)+B(L)*H
      HD(I)=HD(I)+A(L)*H
26    CONTINUE

 FOR ZEROES ONLY, NO POLES IN THE TRANSFER FUNCTION...
      IF(NPOLE.NE.0) GOTO 14
      SSR=REAL(HN(I)*CONJG(HN(I)))
      SQ=(SQRT(SSR))
      IF(SQ.EQ.0.0.OR.SSR.EQ.0.0)THEN
      R(I)=0.0
      S(I)=0.0
      GOTO 225
      END IF
      R(I)=20.0*ALOG10(SQ)
      S(I)=10.0*ALOG10(SSR*VAR)
25    CONTINUE
14    CONTINUE

 FOR POLES AND ZEROES IN THE TRANSFER FUNCTION...
      IF(NPOLE.EQ.0.OR.NZERO.EQ.0) GOTO 157
      HA=REAL((HN(I)*CONJG(HN(I)))/(HD(I)*CONJG(HD(I))))
      SQ=SQRT(HA)
      SIGHA=HA*VAR
      IF(SQ.EQ.0.0.OR.HA.EQ.0.0) THEN
      R(I)=0.0
      S(I)=0.0
      GOTO 157
      END IF
      R(I)=20.0*ALOG10(SQ)
      S(I)=10.0*ALOG10(SIGHA)
57    CONTINUE
27    CONTINUE

 WRITE OUTPUT POWER SPECTRUM TO DISK.


      IF(NFILE.LE.2)THEN
      WRITE(NOUT,273)(S(I),I=1,IFS)
73    FORMAT(5X,F12.6)
      GOTO 237
      END IF
      IF(NFILE.GT.2)) GOTO 237

 FOR LATTICE FILTER, ADD (IN DB) INPUT POWER SPECTRUM TO 20LOG \H(W)\**2

      IF(NFILE.EQ.2).AND.INFILE.EQ.2) THEN
      DO 273 I=1,IFS
      S(I)=S(I)+SS(I)
73    CONTINUE
      END IF

C COMPUTE NUMBER OF POINTS AND RANGES OF PLOT.

237       AFREQ=1.0
          DF(1)=FLOAT(IFS)

          DO 291 I=1,IFS
          FREQ(I)=-1.0*R(I)
          SREQ(I)=-1.0*S(I)
          F(I)=F(I)*2.0
291       CONTINUE
          RMIN=-1.0*AMAX(FREQ,IFS)
          RMAX=AMAX(R,IFS)
          SMIN=-1.0*AMAX(SREQ,IFS)
          SMAX=AMAX(S,IFS)

C PLOT TRANSFER FUNCTION (FREQUENCY RESPONSE).


          TYPE*,'ENTER 1 IF YOU WOULD LIKE TO PLOT TRANSFER FUNCTION.'
          TYPE*,'OTHERWISE ENTER 0.'
          ACCEPT*,ITERS
          IF(ITERS.NE.1)THEN
          IF(NFILE.EQ.2.OR.INFILE.EQ.2)GOTO 493
          GOTO 294
          END IF



          CALL USTART
          CALL USAREA(0.0,7.0,0.0,5.0)
          CALL USET('TEXT')
          CALL USTPOS(0.0,100.0,0.0,100.0)
          IF(NFILE.LE.2))GOTO 193
          CALL UPRINT(10.0,85.0,'DIRECT FILTER\')
          GOTO 194
193       CONTINUE
          CALL UPRINT(10.0,85.0,'LATTICE(INVERSE)FILTER\')
          CALL UPRINT(10.0,75.0,'NOISE FILTER INPUT:\')
          IF(INFILE.EQ.2)THEN
          CALL UPRINT(10.0,70.0,'NOISE\')
          END IF
          IF(INFILE.NE.1)THEN
          CALL UPRINT(10.0,70.0,'IMPULSE\')
          END IF
          IF(INFILE.NE.1)THEN
          CALL UPRINT(10.0,70.0,'CONTAIN\')
          END IF
194       CONTINUE
          CALL UPRINT(10.0,80.0,'TRANSFER FUNCTION\')
          CALL UPRINT(10.0,65.0,'20LOG \H(W)\\')
          CALL USET('SAME SCALE')
          CALL USAREA(2.0,7.0,0.5,5.0)
          CALL USET('XLOG')
          CALL USET('YLOG')
          CALL ULABEL('XLABEL','FREQUENCY\')
          CALL ULABEL('YLABEL','20LOG \H(W)\\')
          CALL USTPOS(0.0,AFREQ,RMIN,RMAX)
```

```
      CALL USET('[illegible]SCALE')
      CALL UPLOT1(F,S,1,0,PTS,OPTS)
      CALL [illegible](DUMMY)
      READ(5,501) DUMMY
      CALL UEND

      CALL USTART
      CALL UERASE
      CALL UEND

      IF(MFILT.EQ.3.OR.MFILT.EQ.4) GOTO 293
      IF([illegible].LE.2.OR.[illegible].LE.2) THEN

C PLOT OUTPUT POWER SPECTRUM
193   CONTINUE

      TYPE*,'ENTER 1 IF YOU WOULD LIKE TO PLOT OUTPUT POWER SPECTRUM,'
      TYPE*,'ENTER 0 OTHERWISE.'
      ACCEPT*,ISPEC
      IF(ISPEC.NE.1) GOTO 293

      CALL USTART
      CALL UDAREA(0.0,7.0,0.0,5.0)
      CALL USET('TEXT')
      CALL UWINDO(0.0,100.0,0.0,100.0)
      IF(MFILT.EQ.21)GOTO 176
      CALL UPRINT(0.0,85.0,'PREFILTER OUTPUT\')
      CALL UPRINT(0.0,70.0,'20LOG VAR*\H(W)\**2\')
      CALL UPRINT(0.0,65.0,'DB GAIN:\')
      CALL UPRINT(0.0,60.0,'NPOLE=\')
      CALL UPRINT(0.0,55.0,'NZERO=\')
      CALL UPRINT(0.0,45.5,'ITERATIONS:\')
      CALL USET('REAL NUMBER')
      CALL UPRINT(20.0,65.0,GAIN)
      CALL USET('INTEGER')
      CALL UPRINT(20.0,[illegible].0,ANPOLE)
      CALL UPRINT(20.0,55.0,ANZERO)
      CALL UPRINT(20.0,45.0,A[illegible])
      GOTO 177
176   CONTINUE
      CALL UPRINT(0.0,85.,'LATTICE OUTPUT\')
      CALL UPRINT(0.0,70.0,'20LOG PO(K)*\H(W)\**2\')
      CALL UPRINT(0.0,60.0,'DB GAIN:\')
      CALL UPRINT(0.0,55.0,'NZERO=\')
      CALL UPRINT(0.0,45.0,'ALPHA:\')
      CALL UPRINT(0.0,40.0,'SNR:\')
      CALL UPRINT(0.0,35.0,'ITERATIONS:\')
      CALL USET('REAL NUMBER')
      CALL UPRINT(20.0,60.0,GAIN)
      CALL UPRINT(20.0,45.0,ALPHA)
      CALL UPRINT(20.0,40.0,SNR)
      CALL USET('INTEGER')
      CALL UPRINT(20.0,55.0,ANZERO)
      CALL UPRINT(20.0,35.0,A[illegible])
177   CONTINUE
      CALL USET('[illegible]')

      CALL UPRINT(0.0,80.0,'POWER SPECTRUM\')
      CALL USET('[illegible]')
      CALL UDAREA(2.0,7.0,0.0,5.0)
      CALL USET('XBOTH')
      CALL USET('YBOTH')
      CALL UPSET('XLABEL','FREQUENCY\')
      CALL UPSET('YLABEL','10 LOG F(W)\')
      CALL UWINDO(0.0,[illegible],YMIN,YMAX)
      CALL USET('[illegible]SCALE')
      CALL UPLOT1(F,S,1,0,PTS,OPTS)
      CALL [illegible](DUMMY)
      READ(5,501) DUMMY
      CALL UEND

      END IF

293   CALL USTART
      CALL UERASE
      CALL UEND

      [illegible]

      STOP
      END
```

Program LSL.FOR
Least Squares Complex Lattice Processor

```
*************************************************************************
THIS PROGRAM HAS BEEN REVISED AS OF 12/6/82. THE LISTINGS SUPPLIED WITH
AEL INTERNAL MEMORANDUM 82-219, DATED 11/12/82 ARE NO LONGER CORRECT.
PLEASE OBTAIN A NEW LISTING.
*************************************************************************

COMPLEX SINGLE CHANNEL LEAST SQUARES LATTICE ALGORITHM

PROGRAM PACKAGE AND USER INTERFACE BY B.A. COOPER.
LSL ALGORITHM BY R. HODGKISS (REF.1).

      COMPLEX EF(20), EB(20), EB1(20), GAM(20), AKB(16)
     1, Y(5000), DEL(20), SF(20), SB1(20), SB(20),XXX(5000)
     1, KF(20), KB(20), E(20), DELTMP, CONJG, H(20,20), H1(20,20)
     1, XNOISE(5000),EEE(5000),AEEV(20),ZZ(20),AMLT(20),AKF(16)

      REAL AMP,XREAL(5000),ALSL

      INTEGER P

      INTEGER*2 NEWPAG

      DATA NEWPAG/Z0C1B/

ASSIGN INPUT FILE NUMBER.
      NFILE=21

      READ(NFILE,66) IOFILE
      FORMAT(17X,I5)

      READ(I[illegible],88)
      FORMAT([illegible])

      READ(NFILE,99)
      FORMAT(51X)

ASK THE USER FOR SOME PARAMETERS.

00    TYPE *,'$ ENTER FILTER ORDER (1<=P<=16):'
      ACCEPT *, P

      IF(P.GT.16) THEN
      TYPE*,'$INCORRECT INPUT, TRY AGAIN.'
      GOTO 4000
      END IF

U0    TYPE *,'$ ENTER ALPHA(ALSL):'
      TYPE*,'$(0.0<ALSL<1.0).'
      ACCEPT *, ALSL

      IF(ALSL.LE.0.0.OR.ALSL.GE.1.0) THEN
      TYPE*,'$INCORRECT INPUT, TRY AGAIN.'
      GOTO 4100
      END IF

      TYPE*,'$SPECIFY DESIRED SNR IN DB AS A REAL NUMBER.'
      ACCEPT*,SNR

1200  TYPE*,'$ENTER NUMBER OF ITERATIONS DESIRED.'
      TYPE*,'$THIS MUST NOT EXCEED NUMBER OF SAMPLES GENERATED!'
      ACCEPT*,ITN

      IF(ITN.GT.5000)THEN
      TYPE*,'$INCORRECT INPUT, TRY AGAIN.'
      GOTO 1200
      END IF

C ASSIGN OUTPUT FILE NUMBERS.
      NCOEF=11
      KFOUT=12
      KBOUT=13
      NKF=14
      NKB=15
      NE=16
      NFB=17
      NBA=18
      NOISE=7

      WRITE(6,12)KFOUT
12    FORMAT(5X,40HFORWARD PREDICTOR COEFFICIENTS IN FILE: ,I5)

      WRITE(6,48)KBOUT
48    FORMAT(5X,41HBACKWARD PREDICTOR COEFFICIENTS IN FILE: ,I5)

      WRITE(6,11)NKF
11    FORMAT(5X,37HFORWARD PARCOR COEFFICIENTS IN FILE: ,I5)

      WRITE(6,47)NKB
47    FORMAT(5X,38HBACKWARD PARCOR COEFFICIENTS IN FILE: ,I5)

      WRITE(6,49)NCOEF
49    FORMAT(5X,46HFINAL FORWARD PREDICTOR COEFFICIENTS IN FILE: ,I5)

      WRITE(6,55)NE
55    FORMAT(5X,22HERROR SIGNAL IN FILE: ,I5)

      WRITE(6,56)NFB
56    FORMAT(5X,43HFINAL FORWARD PARCOR COEFFICIENTS IN FILE: ,I5)

      WRITE(6,57)NBA
57    FORMAT(5X,44HFINAL BACKWARD PARCOR COEFFICIENTS IN FILE: ,I5)

      TYPE*,'$MAKE SURE TO RECORD THESE NUMBERS.'

      TYPE*,'$IF USING THE TEKTRONIX TERMINAL, DEPRESS PAGE BUTTON'
      TYPE*,'$TO CLEAR SCREEN FOR HARD COPY.'

      WRITE(NF,600)NE
```

```
      WRITE(NKF,600)NKF
      WRITE(NKB,600)NKB
      WRITE(NPOUT,600)NPOUT
      WRITE(NCOEF,600)NCOEF
      WRITE(NOUT,600)NOUT
00    FORMAT(5X,30HTHIS INFORMATION IS STORED IN DISK FILE: ,I5)

      WRITE(NE,5)P,LIM
      WRITE(NOUT,5) P,LIM
      WRITE(NPOUT,5) P,LIM
      WRITE(NKF,5) P,LIM
      WRITE(NKB,5) P,LIM
      WRITE(NCOEF,5) P,LIM
      FORMAT(5X,3HP: ,I5,5X,3HM: ,I5)

      SEXP=10.0**(SNR/10.0)

 READ IN NOISE.
      READ(NOISE,225)STD,(XNOISE(I),I=1,LIM)
25    FORMAT(10X,F12.6/(5X,4F12.6))

      IF(INFILE.GE.3)GOTO 107

 READ IN STOCHASTIC OUTPUT OF PREDICTOR.
      READ (XFILE,25)SIGSTC,(XXX(I),I=1,LIM)
5     FORMAT(10X,F12.6/(5X,4F12.6))
      IF(INFILE.LE.2)GOTO 108
7     CONTINUE

 IF PREDICTOR OUTPUT IS DETERMINISTIC, READ IT IN HERE.
      READ(XFILE,208) (XXX(I),I=1,LIM)
08    FORMAT(5X,4F12.6)

 COMPUTE VARIANCE OF SIGNAL

8     STUM=0.0
      SSUM=0.0
      DO 962 I=1,M
      STUM=STUM+(AIMAG(XXX(I)))**2
      SSUM=SSUM+(REAL(XXX(I)))**2
2     CONTINUE
      SSUM=SSUM/LIM
      STUM=STUM/LIM
      SSUM=SSUM+STUM

 COMPUTE SCALING FOR ADDED NOISE IN ORDER TO ACHIEVE DESIRED SNR.

      CONTINUE
      SCALE=SQRT(SSUM/SEXP)
      AMP=SCALE/STD

C ADD SCALED NOISE TO SIGNAL.
      DO 73 I=1,LIM
      X(I)=XXX(I)+(AMP*XNOISE(I))
73    CONTINUE

C ***************************************************************
C THE FOLLOWING IS TAKEN FROM REF. 1 (COMMENTS MINE).
C****************************************************************

C THE NEXT SECTION INITIALIZES THE PARAMETERS AT TIME ZERO FOR ALL STAGES OF
C THE LATTICE.  FOR THIS PROGRAM, THE STAGES WILL BE COUNTED 2<=I<=P+2 RATHER
C THAN 0<=I<=P.  THESE STATEMENTS CORRESPOND TO EQUATIONS 4A-4D IN REF. 7.

      DO 20 I=1,P+1

      DO 10 J=1,P+2
      F(I,J)=(0.,0.)
      RI(I,J)=(0.,0.)
10    CONTINUE

20    CONTINUE

C INITIALIZE ERI(I) FOR ALL STAGES = EPSILON = .001  TO AVOID DIVIDING
C BY ZERO IN EQUATION 4I.

      DO 30 I=1,P+2
      ERI(I)=(0.001,0.)
      DEI(I)=(0.,0.)
      SPI(I)=(0.,0.)
30    CONTINUE

C THE INNER LOOP IS EXECUTED FOR EACH TIME SAMPLE.
C INITIALIZE PARAMETERS AT STAGE ZERO OF THE LATTICE.  THE NEXT FOUR STATEMENTS
C CORRESPOND TO EQUATIONS 4E-4G IN REF.7.

      DO 190 I=1,LIM
      SF(2)=X(I)
      SR(2)=X(I)
      ER(2)=(1.0-ALSQ)*ERI(2)+CABS(X(I))**2
      EF(2)=ER(2)
      GAM(1)=(0.,0.)

C THE NEXT LOOP IS EXECUTED FOR EACH STAGE OF THE LATTICE.
C THE NEXT GROUP OF STATEMENTS CORRESPOND TO EQUATIONS 4H-4O.

      DO 60 J=3,P+2
      DELTMP=(1.0-ALSQ)*DEL(J)-SF(J-1)*CONJG(SRI(J-1))/(1.0-GAM(J-2))
      DEL(J)=DELTMP
```

```
      KF(J)=CONJG(EST(J))/EF(J-1)
      KB(J)=DEL(J)/EB1(J-1)
      SF(J)=SF(J-1)+KB(J)*SB1(J-1)
      SB(J)=SB1(J-1)+KF(J)*SF(J-1)
      EF(J)=EF(J-1)-CABS(DEL(J))**2/EB1(J-1)
      EB(J)=EB1(J-1)-CABS(DEL(J))**2/EF(J-1)
      GAM(J-1)=GAM(J-2)+CABS(SB1(J-1))**2/EB1(J-1)
      A(J)=KB(J)
      B(2,J)=KF(J)

COMPUTE PREDICTOR COEFFICIENTS (EQ. 4B-15).

      DO 50 M=3,J-1
      B(M,J)=B1(M-1,J-1)+KF(J)*A(M)
      A(M)=A(M)+KB(J)*B1(M-1,J-1)
50    CONTINUE

80    CONTINUE


THE NEXT GROUP OF STATEMENTS SAVE VALUES AT TIME N WHICH WILL BE NEEDED
(AS FUNCTIONS OF TIME N-1) AFTER N IS UPDATED AND THE VALUES ARE
RECOMPUTED.

      DO 80 J=2,P+1

      DO 70 JJ=3,P+2
      B1(J,JJ)=B(J,JJ)
70    CONTINUE

80    CONTINUE

      DO 90 J=2,P+2
      EB1(J)=EB(J)
      SB1(J)=SB(J)
90    CONTINUE

****************************************************
END OF PROGRAM FROM REF. 1
****************************************************



SAVE FORWARD PREDICTOR COEFFICIENTS:

      WRITE(NOUT,181)I,(A(J+2),J=1,P)
81    FORMAT(5X,30HFORWARD PREDICTOR COEFFICIENTS,5X,
     13HN: ,I5/(5X,4F12.6))

SAVE BACKWARD PREDICTOR COEFFICIENTS:

      WRITE(NOUT,184)I,(B(J+1,P+2),J=1,P)
84    FORMAT(5X,29HBACKWARD PARCOR COEFFICIENTS,5X,
     13HN: ,I5/(5X,4F12.6))

SAVE FORWARD PARCOR COEFFICIENTS:

C     WRITE(NKF,182)I,(KF(J+2),J=1,P)
C182  FORMAT(5X,31HBACKWARD PREDICTOR COEFFICIENTS,5X,
C    13HN: ,I5/(5X,4F12.6))

C SAVE BACKWARD PARCOR COEFFICIENTS:

C     WRITE(NKB,183)I,(KB(J+2),J=1,P)
C183  FORMAT(5X,27HBACKWARD PARCOR COEFFICIENTS,5X,
C    13HN: ,I5/(5X,4F12.6))

      EEE(I)=(SF(P+2))

100   CONTINUE



C SAVE ERROR SIGNAL
      WRITE(NE,124)ALSL,SNR,(EEE(I),I=1,LIM)
124   FORMAT(5X,5HALF: ,F9.6,5X,5HSNR: ,F12.6,
     15X,12HERROR SIGNAL/(5X,2F15.6))

C SAVE FINAL FORWARD AND BACKWARD PARCOR COEFFICIENTS
      DO 188 J=1,P
      AKF(J)=KF(J+2)
      AKB(J)=KB(J+2)
188   CONTINUE

      WRITE(NFO,191)(AKF(J),J=1,P)
      WRITE(NBA,192)(AKB(J),J=1,P)
191   FORMAT(5X,33HFINAL FORWARD PARCOR COEFFICIENTS/(5X,2F12.6))
192   FORMAT(5X,34HFINAL BACKWARD PARCOR COEFFICIENTS/(5X,2F12.6))

      TYPE*,'ENTER NUMBER OF FILE TO SAVE COEFFICIENTS IN.'
      TYPE*,'PLEASE USE A FILE NUMBER BETWEEN 70 AND 99, INCLUSIVE.'
      ACCEPT*,NSAVE

      WRITE(5,262)NFRAG
262   FORMAT(A2)

      IF(INFILE.EQ.2) THEN
      WRITE(6,211)
      WRITE(NSAVE,211)
      END IF
      IF(INFILE.EQ.3) THEN
      WRITE(6,112)
      WRITE(NSAVE,112)
      END IF
      IF(INFILE.EQ.4) THEN
      WRITE(6,113)
      WRITE(NSAVE,113)
      END IF
211   FORMAT(5X,50HLATTICE (INVERSE) FILTER FOR PREFILTER INPUT=NOISE)
112   FORMAT(5X,52HLATTICE (INVERSE) FILTER FOR PREFILTER INPUT=IMPULSE)
113   FORMAT(5X,52HLATTICE (INVERSE) FILTER FOR PREFILTER INPUT=DIGITAL)


      WRITE(6,192)(AKB(J),J=1,P)
      WRITE(6,191)(AKF(J),J=1,P)
      WRITE(NSAVE,191)(AKF(J),J=1,P)
```

```
      WRITE(NSAVE,192)(AKR(J),J=1,P)

SAVE FINAL FORWARD PREDICTOR COEFFICIENTS
      DO 146 J=1,P
      AMLT(J)=A(J+2)
6     CONTINUE

      II=1
      DO 345 I=P,1,-1
      APEV(II)=AMLT(I)
      II=II+1
5     CONTINUE
      WRITE(6,908)(AMLT(LL),LL=1,P)
      WRITE(NSAVE,908)(AMLT(LL),LL=1,P)
8     FORMAT(5X,32HFORWARD PREDICTOR COEFFICIENTS: /(5X,2E12.6))

SOLVE FOR ROOTS OF THIS EQUATION IN Z**-1 TO MAKE SURE THEY ARE OUTSIDE
THE UNIT CIRCLE (SO THAT Z-PLANE ZEROES ARE INSIDE UNIT CIRCLE).
ZCPOLY IS PART OF THE IMSL PACKAGE.

      APEV(P+1)=(1.0,0.0)
      CALL ZCPOLY(APEV,P,ZZ,IER)

APEV IS AN ARRAY OF THE FORWARD PREDICTOR COEFFICIENTS.
P IS THE ORDER OF THE POLYNOMIAL.
ZZ (OUTPUT) IS AN ARRAY OF THE ROOTS OF THE POLYNOMIAL IN Z**-1.
IER IS AN ERROR CODE.

      WRITE(6,796)(CABS(ZZ(I)),I=1,P)
      WRITE(NSAVE,796)(CABS(ZZ(I)),I=1,P)
6     FORMAT(5X,30HZ-PLANE ZEROES (ABS. VALUE): /(5X,F12.6))
      DO 227 I=1,P

IF ROOT IS INSIDE UNIT CIRCLE, TAKE RECIPROCAL.

      IF((CABS(ZZ(I))).LT.1.0)THEN
      ZZ(I)=1.0/ZZ(I)
      NFLAG=1
      END IF
7     CONTINUE
      IF(NFLAG.EQ.1)THEN
      WRITE(6,999)(CABS(ZZ(I)),I=1,P)
      WRITE(NSAVE,999)(CABS(ZZ(I)),I=1,P)
9     FORMAT(5X,43HNEW Z-PLANE ZEROES (ABS. VALUE): /(5X,F12.6))

      TYPE*,'IF ON THE TEKTRONIX TERMINAL, MAKE HARD COPY.'


THEN RECOMPUTE THE FILTER COEFFICIENTS.
P IS THE ORDER OF THE POLYNOMIAL.
ZZ IS AN ARRAY OF THE ROOTS OF THE POLYNOMIAL IN Z**-1.
AMLT (OUTPUT) IS AN ARRAY OF THE NEW PREDICTOR COEFFICIENTS.

      CALL SYNTH(P,ZZ,AMLT)

SAVE THE Z**-1 PLANE ZEROES AND COEFFICIENTS, AND WRITE THEM TO SCREEN.


      WRITE(6,610)(AMLT(MM),MM=1,P)
      WRITE(NSAVE,610)(AMLT(MM),MM=1,P)
610   FORMAT(5X,36HNEW FORWARD PREDICTOR COEFFICIENTS: /(5X,2E12.6))
      END IF
      WRITE(NCOEF,197)(AMLT(J),J=1,P)
197   FORMAT(5X,36HFINAL FORWARD PREDICTOR COEFFICIENTS/
     1(5X,1E12.6))
101   CONTINUE

      ENDFILE NSAVE
      ENDFILE IK
      ENDFILE NOUT
      ENDFILE NROUT
      ENDFILE IXE
      ENDFILE IXW
      ENDFILE NCOEF
      ENDFILE NOISE
      ENDFILE NFD
      ENDFILE NPA
      STOP
      END
```

```
      SUBROUTINE SYNTH1(N,A,B)

THIS SUBROUTINE SYNTHESIZES A COMPLEX POLYNOMIAL IN Z**-1 FROM AN
ARRAY OF THE ROOTS OF THE POLYNOMIAL.

N IS THE ORDER OF THE POLYNOMIAL.
B (OUTPUT) IS AN ARRAY OF THE  FILTER COEFFICIENTS OF THE POLYNOMIAL
IN Z**-1.

      COMPLEX A(16),B(17),ANEG(16)
      PI=4.0*ATAN(1.0)
      DO 555 I=1,N
55    CONTINUE
      DO 30 K=1,N
      B(K)=(0.0,0.0)
0     CONTINUE
      B(2)=-1.0*A(1)
      B(1)=(1.0,0.0)
      DO 20 I=2,N
      DO 10 J=(I+1),2,-1
      B(J)=B(J)-A(I)*B(J-1)
0     CONTINUE
0     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SYNTH(N,FF,BBW,BB,CC,FSS)

C*********************************************************************
C THIS PROGRAM HAS BEEN REVISED AS OF 1/24/83. THE LISTINGS SUPPLIED
C WITH ARL INTERNAL MEMORANDUM 82-219, DATED 11/12/82 ARE NO LONGER
C CORRECT. PLEASE OPTAIN NEW LISTINGS.
C*********************************************************************


C THIS SUBROUTINE SYNTHESIZES A COMPLEX POLYNOMIAL  IN Z**-1FROM AN
C ARRAY OF FORMANT FREQUENCIES AND AN ARRAY OF THEIR RESPECTIVE BANDWIDTHS.

C N IS THE NUMBER OF FORMANTS AND THE DEGREE OF THE POLYNOMIAL.
C FF IS AN ARRAY OF THE FORMANT FREQUENCIES IN HZ.
C BBW IS AN ARRAY OF THEIR BANDWIDTHS IN HZ.
C BB (OUTPUT) IS AN ARRAY OF THE REAL PARTS OF THE FILTER COEFFICIENTS.
C CC (OUTPUT) IS AN ARRAY OF THE IMAGINARY PARTS OF THE FILTER COEFFICIENTS.

C SEE TEXT FOR RELATIONSHIPS BETWEEN FORMANTS, THEIR BANDWIDTHS,
C AND THE FILTER COEFFICIENTS.


      COMPLEX A(16),B(17),ANEG(16),PFEXP,ANFEXP,E2P,E2N
      REAL BB(17),CC(17),FF(8),BBW(8)
      PI=4.0*ATAN(1.0)
      DO 107 I=1,(N/2)
      FF(I)=FF(I)/FSS
      BBW(I)=BBW(I)/FSS
      E1=EXP(-BBW(I)*PI)
      PFEXP=CMPLX(0.0,(2.0*PI*FF(I)))
      ANFEXP=CMPLX(0.0,(-2.0*PI*FF(I)))
      E2P=CEXP(PFEXP)
      E2N=CEXP(ANFEXP)
C THESE ARE THE ZPLANE POLES.
      A(I*2)=E1*E2P
      A((I*2)-1)=E1*E2N
107   CONTINUE
      DO 30 K=1,17
      B(K)=(0.0,0.0)
30    CONTINUE
      B(2)=-1.0*A(1)
      B(1)=(1.0,0.0)
      DO 20 I=2,N
      DO 10 J=(I+1),2,-1
      B(J)=B(J)-A(I)*B(J-1)
10    CONTINUE
20    CONTINUE

      DO 92 MM=1,N
      BB(MM)=REAL(B(MM+1))
      CC(MM)=AIMAG(B(MM+1))
92    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RANDNORM(UNIFORM,SEED,D1,D2)
C  NORMALLY DISTRIBUTED RANDOM NUMBER GENERATOR BASED ON CACM 334
C
C  RETURNS DEVIATES D1 AND D2, BOTH RANDOM BOTH FROM A NORMAL
C   DISTRIBUTION OF SIGMA = 1.
C
C  UNIFORM IS THE NAME OF A UNIFORM RANDOM NUMBER GENERATOR, AND
C   SEED IS A SEED VALUE.  SEED SHOULD PROBABLY BE A VARIABLE
C   IN THE CALLING ROUTINE, AS MANY RNGS MODIFY THE SEED VALUE.
C   UNIFORM MUST BE DECLARED EXTERNAL IN THE CALLING ROUTINE.
C
100   X = 2.0*UNIFORM(SEED) - 1.0
      Y = 2.0*UNIFORM(SEED) - 1.0
      S = X**2 + Y**2
      IF (S .EQ. 0.0 .OR. S .GT. 1.0) GO TO 100
      U = SQRT(-2.0*ALOG(S)/S)
      D1 = X*U
      D2 = Y*U
      RETURN
      END
```

```
      FUNCTION MAXINDC(ARRAY,LENARAY)

C THIS FUNCTION FINDS THE ARRAY COORDINATE OF THE ELEMENT WITH THE MAXIMUM
C VALUE.
C ARRAY IS AN ARRAY OF POINTS.
C LENARAY IS THE LENGTH OF THE ARRAY.

      DIMENSION ARRAY(LENARAY)

      AMAXAY=ARRAY(1)
      AMAXIC=FLOAT(1)
      DO 10 I=2,LENARAY
      IF(ARRAY(I).GT.AMAXAY) THEN
      AMAXAY=ARRAY(I)
      AMAXIC=FLOAT(I)
      END IF
10    CONTINUE
      RETURN
      END
```

```
      FUNCTION ARRMAX(ARRAY,LENARAY)

C THIS FUNCTION RETURNS THE MAXIMUM VALUE IN THE ARRAY.
C ARRAY IS AN ARRAY OF POINTS.
C LENARAY IS THE LENGTH OF THE ARRAY.

      DIMENSION ARRAY(LENARAY)
      ARRMAX=ARRAY(1)
      DO 10 I=2,LENARAY
      IF(ARRAY(I).GT.ARRMAX) ARRMAX=ARRAY(I)
10    CONTINUE
      RETURN
      END
```

APPENDIX B

SAMPLE TERMINAL SESSION FOR THE CASE ILLUSTRATED IN FIG. 3

```
@LATTEK
WHICH PROGRAM TO USE?: GEN
$ENTER NUMBER OF SAMPLES TO GENERATE.
$THIS NUMBER SHOULD CORRESPOND TO DESIRED NUMBER OF ITERATIONS OF LATTICE.

$((1<=MK=5000).
5000
$ENTER DESIRED STANDARD DEVIATION OF GAUSSIAN WHITE NOISE.
$ (MUST BE FLOATING  POINT).:
1.0
$ENTER DESIRED FREQUENCY OF IMPULSE FUNCTION IN HZ.
125.0
$WHITE NOISE IN FILE 1.
$GAUSSIAN NOISE IN FILE 2.
$IMPULSE INPUT IN FILE 3.
$MAKE SURE TO RECORD THESE NUMBERS.
FORTRAN STOP
ENTER Y[ES] TO CONTINUE: Y
WHICH PROGRAM TO USE?: FILTER
$ENTER 1 IF YOU WOULD LIKE TO READ CALCULATED INVERSE FILTER COEFFICIENTS
FROM
 DISK.
$ENTER 0 IF YOU WOULD LIKE TO TYPE IN FILTER COEFFICIENTS.
$ENTER 2 IF YOU WOULD LIKE TO SPECIFY FORMANTS AND  BANDWIDTHS.
$ENTER 3 IF YOU WOULD LIKE TO READ IN PREFILTER COEFFICIENTS FROM DISK.
2
$ENTER NUMBER OF SAMPLES TO ANALYZE.
$THIS NUMBER SHOULD CORRESPOND TO DESIRED NUMBER OF ITERATIONS OF THE LATT
ICE (M
<=5000).
5000
```

```
$IF YOU WISH TO SPECIFY FORMANTS AND BANDWIDTHS, THERE WILL BE TWO (COMPLE
% CONJ
UGATE) POLES FOR EACH FORMANT.
$ENTER NUMBER OF POLES (1<=NPOLE<=16).
2
     I=        1
$ENTER FORMANT FREQUENCY IN HZ.
2500.0
$ENTER BANDWIDTH OF THIS FORMANT FREQUENCY IN HZ.
150.0
$FILTER COEFFICIENTS IN ORDER OF INCREASING POWER.
$(A(0)=1.0).
          0.000000     0.000000
          0.985112     0.000000
$ENTER 1 IF YOU WOULD LIKE TO SAVE THIS INFORMATION ON DISK.
$OTHERWISE, ENTER 0.
1
$ENTER NUMBER OF DISK FILE TO SAVE THIS IN.
$USE A FILE NUMBER BETWEEN 40 AND 60, INCLUSIVE.
40
```

(SCREEN WILL ERASE)

```
    THIS INFORMATION STORED IN DISK FILE        40
    PREFILTER
    FORMANT            BANDWIDTH
    2500.00            150.00
    FILTER COEFFICIENTS (INCREASING POWER, A0=1).
         0.000000     0.000000
         0.985112     0.000000
$ENTER DESIRED INPUT FILE NUMBER.
$NOISE INPUT:FILE 2
$IMPULSE INPUT:FILE 3
3
    OUTPUT IN FILE:     21
$BE SURE TO RECORD THIS NUMBER.
$ENTER 1 IF YOU WOULD LIKE TO PLOT INPUT TIME SERIES.
$ENTER 0 OTHERWISE.
1
$ENTER FIRST AND LAST TIME POINTS OF INPUT TO PLOT.
    (1<=M<=) 5000
1
149
$ENTER INTERVAL BETWEEN POINTS ON PLOT.
1
```

(PLOT (a) APPEARS ON SCREEN)

(DEPRESS CR OR SPACE BAR ONCE OR TWICE TO ERASE PLOT)

```
$ENTER 1 IF YOU WOULD LIKE TO PLOT THE INPUT AGAIN ON A  DIFFERENT SCALE.
$OTHERWISE, TYPE IN 0.
0
$ENTER 1 IF YOU WOULD LIKE TO PLOT OUTPUT TIME SERIES.
$OTHERWISE ENTER 0.
1
$ENTER FIRST AND LAST TIME POINTS OF OUTPUT TO PLOT.
    (1<=M<=) 5000
400
480
$ENTER INTERVAL BETWEEN POINTS ON PLOT.
1
```

(PLOT (c) APPERAS ON SCREEN)

```
#ENTER 1 IF YOU WOULD LIKE TO PLOT THE OUTPUT AGAIN ON A DIFFERENT SCALE.
#OTHERWISE, TYPE IN 0.
0
#ENTER 1 IF YOU WOULD LIKE TO PLOT TRANSFER FUNCTION.
#OTHERWISE ENTER 0.
1
```

(PLOT (b) APPEARS ON SCREEN)

```
FORTRAN STOP
ENTER Y[ES] TO CONTINUE: Y
WHICH PROGRAM TO USE?: LSL
# ENTER FILTER ORDER (1<=P<=16):
2
# ENTER ALPHA(CLSL):
#(0.0<ALSL<1.0).
.0001
#SPECIFY DESIRED SNR IN DB AS A REAL NUMBER.
30.0
#ENTER NUMBER OF ITERATIONS DESIRED.
#(1<=LIM<=5000).
5000
    FORWARD PREDICTOR COEFFICIENTS IN FILE:     12
    BACKWARD PREDICTOR COEFFICIENTS IN FILE:     13
    FORWARD PARCOR COEFFICIENTS IN FILE:     14
    BACKWARD PARCOR COEFFICIENTS IN FILE:     15
    FINAL FORWARD PREDICTOR COEFFICIENTS IN FILE:     11
    ERROR SIGNAL IN FILE:     16
    FINAL FORWARD PARCOR COEFFICIENTS IN FILE:     17
    FINAL BACKWARD PARCOR COEFFICIENTS IN FILE:     18
#MAKE SURE TO RECORD THESE NUMBERS.
#IF USING THE TEKTRONIX TERMINAL, DEPRESS PAGE BUTTON
#TO CLEAR SCREEN FOR HARD COPY. (DON'T DO THIS YET)
#ENTER NUMBER OF FILE TO SAVE COEFFICIENTS IN.
#PLEASE USE A FILE NUMBER BETWEEN 70 AND 99, INCLUSIVE.
70
```

(PRESS PAGE BUTTON <u>NOW</u> TO CLEAR TEKTRONIX SCREEN)

```
    FILTER FOR PREFILTER INPUT=IMPULSE
    FINAL BACKWARD PARCOR COEFFICIENTS
        0.000000    0.000000
        0.995760    0.000000
    FINAL FORWARD PARCOR COEFFICIENTS
        0 000000    0.000000
        0.995458    0.000000
    FORWARD PREDICTOR COEFFICIENTS:
        0.000000    0.000000
        0.995760    0.000000
    Z-1PLANE ZEROES (ABS. VALUE):
        1 002127
        1.002127
FORTRAN STOP
ENTER Y[ES] TO CONTINUE: Y
WHICH PROGRAM TO USE?: FILTER
$ENTER 1 IF YOU WOULD LIKE TO READ CALCULATED INVERSE FILTER COEFFICIENTS
FROM
 DISK.
$ENTER 0 IF YOU WOULD LIKE TO TYPE IN FILTER COEFFICIENTS.
$ENTER 2 IF YOU WOULD LIKE TO SPECIFY FORMANTS AND  BANDWIDTHS.
$ENTER 3 IF YOU WOULD LIKE TO READ IN PREFILTER COEFFICIENTS FROM DISK.
1
$ENTER 1 IF YOU WOULD LIKE TO PLOT OUTPUT TIME SERIES.
$OTHERWISE ENTER 0.
1
$ENTER FIRST AND LAST TIME POINTS OF OUTPUT TO PLOT.
    (1<=M<=) 5000
365
435
$ENTER INTERVAL BETWEEN POINTS ON PLOT.
1
```

(THIS PLOT NOT SHOWN IN FIG. 3; IT IS EQUIVALENT TO PLOT (e))

```
$ENTER 1 IF YOU WOULD LIKE TO PLOT THE OUTPUT AGAIN ON A DIFFERENT SCALE.
$OTHERWISE, TYPE IN 0.
0
$ENTER 1 IF YOU WOULD LIKE TO PLOT ERROR SIGNAL.
$OTHERWISE ENTER 0.
1
$ENTER FIRST AND LAST TIME POINTS OF ERROR TO PLOT.
    (1<=M<=) 5000
365
435
$ENTER INTERVAL BETWEEN POINTS ON PLOT.
1
```

(PLOT (e) APPEARS ON SCREEN)

```
ENTER 1 IF YOU WOULD LIKE TO PLOT ERROR AGAIN ON ANOTHER SCALE.
$OTHERWISE, TYPE IN 0.
0
$ENTER 1 IF YOU WOULD LIKE TO PLOT TRANSFER FUNCTION.
$OTHERWISE ENTER 0.
1
```

(PLOT (d) APPEARS ON SCREEN)

```
FORTRAN STOP
ENTER Y[ES] TO CONTINUE: N
ENTER Y[ES] FOR HARD COPY OF ALL FILTER COEFFICIENTS : N
ENTER Y[ES] TO DELETE ALL DATA FILES : N
$
```

(END OF COMMAND PROCEDURE)

DISTRIBUTION LIST FOR ARL TM 82-219, by B. A. Cooper and L. H. Sibul, dated November 12, 1982

Commander
Naval Sea Systems Command
Department of the Navy
Washington, DC 20362

Attention: Dr. E. G. Liszka, PMS 406B
Copy No. 1

Attention: Mr. D. Porter, SEA 63R1
Copy No. 2

Attention: Mr. D. C. Houser, SEA 63R-14
Copy No. 3

Attention: Code SEA 9961 (Library)
Copies 4 and 5

Commander
Naval Air Systems Command
Department of the Navy
Washington, DC 20362

Attention: Mr. J. C. Smith
Copy No. 6

Director
Applied Research Laboratory
University of Texas
Austin, TX 78712

Attention: Dr. J. F. Willman
Copy No. 7

Marine Physical Laboratory
Scripps Institute of Oceanography
San Diego, CA 92152

Attention: W. S. Hodgkiss
Copy No. 8

Attention: K. M. Watson
Copy No. 9

Director
Applied Physics Laboratory
University of Washington
Seattle, WA 98105

Attention: Mr. C. Sienkiewicz
Copy No. 10

Commanding Officer
Naval Coastal Systems Center
Panama City, FL 32401

Attention: Dr. David Skinner
Copy No. 11

Bolt Beranek and Newman, Inc.
1300 North 17th Street
Arlington, VA 22209

Attention: Dr. Henry Cox
Copy No. 12

Systems Control Technology, Inc
1801 Page Mill Road
P. O. Box 10180
Palo Alto, CA 94303

Attention: Dr. Benjamin Friedlander
Copy No. 13

Department of Electrical Engineering
College of Engineering and Applied Science
The University of Rochester
Rochester, NY 14627

Attention: Prof. E. L. Titlebaum
Copy No. 14

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314

Copies 15 through 20

DAT
FILM