THESIS

ADVANCED SIMULATION OF DIGITAL FILTERS

by

Gerald S. Doyle

September 1980

Thesis Advisor: D. E. Kirk

Approved for public release; distribution unlimited

80 12 29 169
An Advanced Simulation of Digital Filters has been implemented on the IBM 360/67 computer utilizing Tektronix hardware and software. The program package is appropriate for use by persons beginning their study of digital signal processing or for filter analysis. The ASDF programs provide the user with an interactive method by which filter pole and zero locations can be manipulated. Graphical output on both the Tektronix graphics screen and the Versatec plotter are provided to observe the effects of pole-zero movement.
Approved for public release; distribution unlimited.

Advanced Simulation of Digital Filters

by

Gerald S. Doyle
Captain, United States Army
B.S., United States Military Academy, 1973

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September, 1980

Author

Approved by: [Signature]

[Name]

Thesis Advisor

[Name]

Second Reader

[Name]

Chairman, Department of Electrical Engineering

[Name]

Dean of Science and Engineering
ABSTRACT

An Advanced Simulation of Digital Filters has been implemented on the IBM 360/67 computer utilizing Tektronix hardware and software. The program package is appropriate for use by persons beginning their study of digital signal processing or for filter analysis. The ASDF programs provide the user with an interactive method by which filter pole and zero locations can be manipulated. Graphical output on both the Tektronix graphics screen and the Versatec plotter are provided to observe the effects of pole-zero movement.
# TABLE OF CONTENTS

I. INTRODUCTION ....................................................... 7

II. SELECTION OF ASDF ALGORITHMS ................................. 8
   A. ALGORITHMS SELECTED ........................................ 9
   B. DEFINING THE DATA STRUCTURE ............................... 9

III. IMPLEMENTED SOLUTION ........................................... 11
   A. INTERFACE WITH THE Z-PLANE ................................ 11
   B. TIME RESPONSE .................................................. 19
   C. FREQUENCY RESPONSE .......................................... 27
   D. OUTPUT PLOTS .................................................... 29
   E. SUPERVISORY ROUTINES ......................................... 30

IV. STRUCTURE OF ASDF COMMANDS .................................. 34
   A. SUMMARY OF DATA INPUT MODES .............................. 37
      1. Interactive Graphics (IAG) ............................... 37
      2. Rectangular .................................................. 38
      3. Polar ......................................................... 38
   B. THE POLZRO COMMAND SUBSTRUCTURE ......................... 39
   C. SUMMARY OF POLZRO SUBCOMMANDS ........................... 43

V. LOADING THE ASDF PACKAGE ....................................... 44
   A. LOGIN PROCEDURE ............................................... 44
   B. TESTING OSX001 FOR ASDF PROGRAMS ........................ 45
      1. ASDF is on OSX001 .......................................... 46
      2. ASDF is not on OSX001 .................................... 46
VI. USING ASDF ........................................ 50
   A. EXAMPLE ONE - INTERACTIVE GRAPHICS ....... 50
   B. EXAMPLE TWO - RECTANGULAR MODE ........... 59
   C. EXAMPLE THREE - POLAR MODE ................. 80
VII. ADDITIONAL EXAMPLES .............................. 94
VIII. INTERPRETATION OF ASDF OUTPUT .......... 201
IX. SUMMARY AND CONCLUSIONS ........................ 212
APPENDIX A: PROBABLE SOLUTIONS FOR DIFFICULTIES .......................... 216
APPENDIX B: IMPORTANT SUBROUTINES AND VARIABLES ........................... 218
APPENDIX C: STRUCTURE FOR ASDF TEXT FILE ............... 224
APPENDIX D: JCL FOR EXEC HRD$CPY COMMAND .......... 226
APPENDIX E: SOURCE DECK FOR ASDF191 EXEC ............ 227
APPENDIX F: SOURCE DECK FOR ASDF192 EXEC ............ 229
APPENDIX G: SOURCE DECK FOR ASDF COMMAND: POLZRO ........ 233
APPENDIX H: SOURCE DECK FOR ASDF COMMAND: RESPONSE ... 276
APPENDIX I: SOURCE DECK FOR ASDF COMMAND: HRD$CY .......... 289
LIST OF REFERENCES ..................................... 306
INITIAL DISTRIBUTION LIST ............................ 307
ACKNOWLEDGEMENTS

The author wishes to acknowledge the people who made this effort possible. Of greatest importance is the love and guidance provided by his parents, Margaret and C. E. Doyle, and sister, Evelyn. It is to them that this thesis is dedicated. Special mention goes to Professor Richard Hamming for his insight and explanations of finite precision effects in digital filters. Thanks are due to Professors D. E. Kirk and R. D. Strum for their suggestions and comments in structuring this work. For their assistance with system problems, thanks are due to J. Foust, F. Wheeler, R. Donat, and H. Doleman. Moreover, thanks are due to E. Christian, and M. F. Bradley for their assistance in preparing the manuscript. Of no less importance are the evening computer center operators J. Kallweit, M. Anderson, and E. Donnellan for their consideration and assistance in processing the figures. However, the author alone is responsible for any shortcomings in this thesis, for the ultimate choice of what was done was his.
I. INTRODUCTION

Digital methods have recently begun to dominate the selection of transmission and filtering schemes. When the student first confronts the discrete domain, he encounters a series of unfamiliar concepts. While most of the discrete concepts have parallels in the continuous or s-domain, this dualism is not apparent initially.

The programs comprising the Advanced Simulation of Digital Filters (ASDF) provide the user with a convenient method for developing intuition in the discrete domain. This is achieved by allowing the user to generate digital filters by directly entering the poles and zeros into a graphics terminal. The ASDF programs then perform the calculations required to display the characteristics of the user generated filter.

The overriding objective of the ASDF program package is to provide the user with an interactive method for modifying the filter pole/zero locations in the z-plane, as well as to compute the filter responses. Both time and frequency responses are computed and displayed graphically on the Tektronix 4012 graphics terminal. [Ref. 9]
II. SELECTION OF ASDF ALGORITHMS

There are several parameters which are particularly important in selecting the algorithms used in the ASDF programs. Since one of the primary objectives of these programs is to provide the user with clear explanations of methods for computing digital filter characteristics, direct or straightforward methods should be used. The importance of developing a user's intuition for the discrete domain and digital filters suggests that graphical rather than numerical outputs would be most appropriate. The use of graphical outputs to characterize a filter's responses implies minimum importance of precision in these results. Since the reason for computing the filter characteristics is to develop the user's intuition for the discrete domain, there are no pressing constraints on the selected algorithms with regard to computational efficiency, speed, or memory space utilization. In fact, the final ASDF package spends the greatest portion of its execution time making the interface with the z-plane, and the form of the output easy to understand and to use. The greatest part of the memory space utilized for the ASDF programs is consumed in the interfacing processes rather than in the methods by which the data is stored, used, or computed.
A. ALGORITHMS SELECTED

Because of the above stated objectives, a simple algorithm was selected for computation of time domain responses of digital filters. Once the user enters the desired pole and zero locations, a direct-form-two difference equation representation of the filter is generated [Ref. 5]. The unit sample sequence is used for the filter input, the difference equations are solved recursively in the discrete time domain and the unit sample response is recorded as the output. The unit step response is calculated by a similar method.

It is known that the direct-form-two filter implementation is very sensitive to parameter variations, and as such cannot be utilized in situations requiring extreme accuracy. The advantages associated with simplicity of understanding the algorithms, the ease of programming the general case, and the deemphasis of accuracy in outputs for marginally stable filters outweighs this disadvantage.

B. DEFINING THE DATA STRUCTURE

A detailed description of the important variables in the ASDF programs is given in Appendix B. There is, however, an overriding data structure common to all portions of the ASDF package which is important.

The array POLZRO holds the locations of the poles and zeros of the filter under analysis. In all cases, zeros are
stored in POLZRO(I,J) for values of I from one to ten. Pole locations are stored in POLZRO(I,J) for values of I from eleven to twenty. Only the top half plane pole of each complex pair is stored. There are five sections of POLZRO(I,J), i.e. values of J from one to five. POLZRO(I,1) is the real part of the rectangular representation of the ith root. POLZRO(I,2) is the imaginary part of the rectangular representation of the ith root. POLZRO(I,3) and POLZRO(I,4) are the magnitude and angle portions of the ith root location in polar form. POLZRO(I,5) is the root type designator. Root type designators are: one for real zeros, two for complex zeros, three for real poles, and four for complex poles.

The array IROOTS(I) holds the numbers of roots of each type of root already entered into the system. IROOTS(1) and IROOTS(3) hold the number of real zeros and poles, respectively. IROOTS(2) and IROOTS(M) hold the number of complex zero and pole pairs, respectively.

The variable IERROR is the error flag, which is set whenever the user generates an error. There are numerous possible errors. These include trying to delete roots which have not been entered, or adding roots which cause the maximum system order to be exceeded.
III. IMPLEMENTED SOLUTION

Having determined the general form of the solution it remained to generate the software implementation. There were five major areas to be programmed: the interface with the z-plane, the time response, the frequency response, the plotting of output curves, and the supervisory routines.

A. INTERFACE WITH THE Z-PLANE

Since the method by which the user modifies the pole and zero locations in the z-plane is of no theoretical interest, the discussion of this implementation will be brief. Extensive coverage of the subroutines employed and the definitions of the variables used is reserved for Appendix B.

The ASDF command which allows the user to make pole-zero modifications in the z-plane is POLZRO. This command initiates execution of a load module which is structured as depicted in figure 3-1. Each of the boxes shown represents a subroutine. Lines with arrow heads on both ends indicate that program control passes to the subroutine and then directly back to the calling routine with no intermediate branches. The main program, MAIN, initializes the required arrays, and tests to ensure that the user has generated a realizable causal filter.
Figure 3-1 Flowchart Of POLZRO
The actual interaction with the z-plane is accomplished in subroutine UNTCIR. The structure of UNTCIR is diagrammed in figure 3-2. The large double ended arrow indicates that there are many subroutines of the form CRRMMM. The subroutines LETTER and BOXUC generate the unit circle and command array. PLOT-10 software [Ref. 10] statements determine the location of the user positioned cursor. This cursor position is adjusted with software tabs to one of twenty-six possible locations. Based on the X and Y location of the adjusted cursor position a specific subroutine is selected for execution. The flashing alphanumeric cursor is positioned in the selected command box using the adjusted cursor location as a reference. The subroutine then reads two alphanumeric characters typed by the user. The combination "space", "e" indicates that the user wishes the selected subroutine to be executed. The combination "space", "x" indicates that the user wishes to terminate the pole zero manipulation phase of the program.

All of the commands which change the status of the z-plane have names of the form CRRMMM where C is either "A" for add root(s) or "D" for delete root(s). The RR indicates root type: RZ - real zero, CZ - complex zero, RP - real pole, CP - complex pole. The MMM stands for the mode of data entry. IAG stands for interactive graphics i.e. the roots will be adjusted with the cursor. POL and RCT stand for polar and rectangular, respectively. The general
Figure 3-2 Structure Of Subroutine UNTCIR
structure of the CRRMMM subroutines is described in figure 3-3.

All CRRMMM subroutines follow the same general pattern. The POLZRO table is checked to determine whether or not the issued command is valid. Poles and zeros must be present before they can be deleted. There cannot be more than ten poles or ten zeros. The desired pole or zero location is next acquired. For subroutines adding roots the entered location is that of the root(s) to be added. For subroutines deleting roots the entered location is any location near the root(s) to be deleted. Using subroutine LOCATE, the POLZRO table is searched for the root of the correct type which is nearest the specified location. This nearest root is then highlighted. In all cases, except when adding roots in the IAG mode, the user is asked whether or not the correct location has been determined. Once the root location is verified the root is added and plotted with subroutines STRRTS and PLOTRT. For subroutines deleting roots the POLZRO table is adjusted with subroutine UPDATE and the unit circle is redrawn.

The user may make an error in carrying out the above steps, i.e., by placing a pole outside the unit circle, by exceeding the maximum system order, or by entering a zero outside ten units from the origin. In each of these cases an

\[1\] Hereafter, poles and zeros will be referred to as roots.
Figure 3-3a Structure of CRRMMM Subroutines
Figure 3-3b Structure Of CRRMMM Subroutines Continued
Figure 3-3c Structure Of CRRMM Subroutines Continued
error flag is set. Subroutine ERROR handles the printing of messages and subroutine LST prints a listing of the current filter status before the program resumes.

B. TIME RESPONSE

The transfer function of a digital filter is defined in equation 3.1. The \( z_i, p_i, a_k, \) and \( b_r \) will be defined in a subsequent section. The variable \( A \) is an arbitrary constant assigned by the user and is called the filter gain.

\[
H(z) = \frac{\sum_{i=1}^{M} a_k (z-z_i)}{\sum_{r=0}^{N} b_r z^{-N}} \quad (3.1)
\]

Inherent in generating the time domain response with a direct-form-two filter implementation are the determination of the coefficients \( a_k \) and \( b_r \) in equation (3-1), and the computation of the unit sample and unit step sequences. Having determined the filter coefficients, the time domain response of the filter is determined by using the appropriate filter input sequence and recording the output of the filter for the desired number of points. This is accomplished by recursively solving a difference equation which characterizes the filter for 1024 time points. The standard direct-form-two filter implementation is shown in figure 3-4. The \( z^{-1} \) terms represent unit time delays. This
Figure 3.4 Direct Form Two Filter Implementation
figure corresponds to the difference equations:

\[ y(n) = \sum_{k=1}^{N} a_k y(n-k) + \sum_{r=0}^{N} b_r x(n-N+r) \]  

(3.2)

To compute the filter's time response, three operations are required: calculate the $a_k$ and $b_r$ coefficients, generate the input sequence $x(k)$, $k=0,1,\ldots,1024$, solves the difference equations and stores the filter output values. There are four central subroutines which perform these operations: ASUBK, BSUBR, RSPNSE, and COEFF. Subroutines ABSUK, BSUBR, and COEFF generate the filter coefficients starting with the pole and zero locations specified. Subroutine RSPNSE solves the difference equations by coordinating the other three routines, generating the appropriate input sequence, and storing the output sequence values.

The ASUBK and BSUBR subroutines are identical in structure and purpose. ASUBK computes the degree of the denominator polynomial as two times the number of complex pole pairs plus the number of real poles. A complex vector is formed named POLES, which holds a listing of all the poles of the filter under analysis. The POLZRO array stores real poles in the form $X+j0$. Only the top half plane pole of each complex pole pair is stored, i.e., $(X+jY)$, where $X$ and $Y$ are non-negative. The vector POLES has entries for all real poles and both poles of each complex pole pair.
Once the POLES vector has been generated the subroutine COEFF multiplies out the factors of the form \((z-p_i)\) to generate the coefficients of the denominator polynomial.

The algorithm for multiplying out the denominator polynomial is essentially the same as would be done by hand. multiplication of a series of literal factors of the form \((z-p_i)\). The vector POLES is passed to the COEFF subroutine as the vector V. Since the denominator factors are of the form \((z-p_i)\), the V vector is changed to hold the values \(-p_i\). If the system is of order less than ten, the remaining positions of V are filled with zeros. The degree of the denominator polynomial is subsequently used to determine how many poles are actually located at the origin.

The vectors used in the subroutine COEFF hold the coefficients for the polynomial being generated. The literal powers of "z" are suppressed. The vector H1 always contains the partial product of the K factors previously multiplied together. The vector H2 represents the old partial product times the "z" of the \((z-p_i)\) factor. Since the coefficient of "z" is always one, H2 is a left shifted version of H1. The vector H3 represents the partial product times the next pole location \(p_i\). This next \(-p_i\) value is stored in V(1). The new H1 vector is formed by adding the H2 and H3 vectors. H1 now contains the product of the first \(K+1\) factors. The values in the V vector are next shifted such that the next pole location to be multiplied is always
in the V(1) position. These four steps are repeated ten times to generate the coefficients of the complete polynomial in "z". The formula for the transfer function requires coefficients of the denominator polynomial in terms of $z^{-1}$ so the order of the coefficients is reversed.

Figure 3-5 depicts several iterations of the algorithm used in the COEFF subroutine. Figure 3-5a shows the initial conditions for each of the vectors defined in the above paragraph. Figure 3-5c shows the values of the vectors after the first iteration. Figure 3-5f shows the vector values after the second iteration. The third and subsequent iterations follow this pattern.

In precisely the same way as described above, the $a_k$ coefficients are generated with the ASUBK subroutine. The zero locations are stored in POLZRO exactly as are the poles. The generation of the vector ZEROS and the use of the subroutine COEFF parallels the BSUBR subroutine exactly.

The significant variables in subroutine RSPNSE are: XSUBN, INPUT, OUTPUT, FLTRGN, A, and B. The vectors A and B hold the already computed values of $a_k$ and $b_k$. A(1) holds $a_0$; A(2) holds $a_1$ . . . A(11) holds $a_{10}$. The $b_r$ are stored similarly in the B vector. XSUBN is the current value of the input sequence. INPUT is the value of the node so labeled in figure 3-4. OUTPUT is similarly noted on figure 3-4. FLTRGN is the value of the filter gain, and is assigned by the user. The vector Z holds the delayed INPUT values.
Figure 3-5a. Initial Conditions Of COEFF Vectors.

<table>
<thead>
<tr>
<th>V</th>
<th>-p_1</th>
<th>-p_2</th>
<th>-p_3</th>
<th>-p_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>H2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>H3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3-5b. Values Of COEFF Vectors Before The End Of The First Iteration.
<table>
<thead>
<tr>
<th>H1</th>
<th>. . . 0 0 1  -p_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>. . . 0 0 1 0</td>
</tr>
<tr>
<td>H3</td>
<td>. . . 0 0 0  -p_1</td>
</tr>
</tbody>
</table>

Figure 3-5c. Values of COEFF Vectors starting the Second Iteration.

<table>
<thead>
<tr>
<th>H1</th>
<th>. . . 0 0 1  -p_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>. . . 0 1  -p_1 0</td>
</tr>
<tr>
<td>H3</td>
<td>. . . 0 0  -p_2  +p_1 p_2</td>
</tr>
</tbody>
</table>

Figure 3-5d. Values of COEFF Vectors before the End of the Second Iteration.
Figure 3-5e. Values of COEFF vectors starting the third iteration.

\[
\begin{array}{c|c|c|c}
\text{H1} & . & . & 1 - (p_1 + p_2) (p_1 p_2) \\
\hline
\text{H2} & . & . & 1 - p_1 0 \\
\hline
\text{H3} & . & . & 0 - p_2 (p_1 p_2) \\
\end{array}
\]

Figure 3-5f. Values of COEFF vectors before the end of the third iteration.

\[
\begin{array}{c|c|c|c}
\text{H1} & . & . & 1 - (p_1 + p_2) (p_1 p_2) \\
\hline
\text{H2} & . & 1 - (p_1 + p_2) (p_1 p_2) 0 \\
\hline
\text{H3} & . & 0 - p_3 + (p_1 + p_2) p_3 - (p_1 p_2 p_3) \\
\end{array}
\]
Z(1) is no delay; Z(2) is delayed one time unit; Z(3) is delayed two time units, etc.

The subroutine RSPNSE generates the appropriate input sequence. For the unit sample response the input sequence is 1, 0, 0, 0, . . . . For the unit step response the input sequence is 1, 1, 1, 1, . . . . The appropriate value of XSUBN is determined by the type of input sequence. The variable INPUT is then computed to be the sum of XSUBN and the a_k's times the delayed INPUT values. The filter output is the sum of the b_r's times the delayed INPUT values. These output values are then multiplied by the filter gain, FLTRGN, and stored in the appropriate array: YI for the unit sample response, YS for the unit step response.

C. FREQUENCY RESPONSE

For discrete systems, the frequency response is found by evaluating the system transfer function at points on the unit circle. Subroutine FRQNCY generates both the phase and the magnitude of the transfer function.

The important variables in the FRQNCY subroutine are: AA, BB, THETA, Z, NNUM, and NDEN. The variables AA and BB are the double precision values of the A and B vectors described in the section on time response. THETA is the angular displacement around the unit circle for which a value of the frequency response is calculated. Z is the
classical z-plane variable:

$$Z = e^{j \Theta} \quad (3.3)$$

NNUM is the number of numerator terms. NDEN is the number of denominator terms.

The frequency response is computed for 1024 points around the unit circle. All points computed are for THETA greater than zero and less than $\pi$. The values of $Z$ for THETA greater than $-\pi$ and less than zero are neither computed nor plotted since the magnitude of the transfer function is an even function of THETA and the phase response is an odd function of THETA.

The generation of the $a_k$ and $b_r$ coefficients has already been discussed in the section on time domain response. The evaluation of $H(z)$ is straightforward. The value of THETA is incremented through 1024 steps from zero to $\pi$. For each value of THETA generated, the value of $Z$ is computed. The value of $H(z)$ is then computed in two parts. The numerator is the sum of NNUM terms of the form $b_r$ times $z^{-r}$ for $r=0, 1, 2, \ldots$ NNUM-1. The denominator is one minus the sum of (NDEN-1) terms of the form $a_k$ times $z^{-k}$ for $k=1, 2, 3, \ldots$ NDEN-1. Dividing numerator by denominator

---

1 NNUM is equal to the degree of the numerator plus one.

2 NDEN equals the degree of the denominator plus one.
yields $H(z)$. The phase of $H(z)$ is then determined using the principal values of the arctangent function; the results are stored in the vector $YP$. The magnitude of $H(z)$ is computed, multiplied by FLTRGN, the filter gain, and subsequently stored in the vector $YM$.

D. OUTPUT PLOTS

The output plots generated by ASDF are generated by two different software programs each being different implementations of the same plotting requirements. One set of plots is generated using PLOT-10 software on the Tektronix 4012 terminal. Another set of plots utilizes the Versatec plotter and associated software. [Ref. 7] The PLOT-10 software is adequately covered in reference 10. The Versatec software is explained in Technical Note-034 in the computer center.

The plots which are generated on both devices are essentially the same. There are five plots generated on the 4012 terminal: unit sample response, unit step response, phase response, magnitude of the transfer function, and magnitude of the transfer function in decibels. Both of the time domain plots are terminated when the output value has stabilized to within one percent of the maximum generated system output. All 1024 points of the phase and magnitude plots are plotted.
The form of the Versatec output is similar to that of the 4012. One additional plot is included. It shows the unit circle with poles and zeros of the filter being analyzed. Plotting is terminated when the filter output stabilizes to within one percent of the maximum system output generated.

On both devices, the values of unit sample response are stored in a vector YI and plotted with subroutine PLTIMP. The unit step response values are stored in YS, and plotted with subroutine PLTSTP. The phase response is stored in vector YP. The magnitude of the transfer function is stored in vector YM. The magnitude of the transfer function in decibels is stored in YMDB. The three vectors YP, YM, and YMDB are all plotted with subroutine PLTTRF. The Versatec plot of the unit circle is generated with subroutine UNTCR. The 4012 plot of the unit circle and command array are generated with the subroutine UNTCIR.

E. SUPERVISORY ROUTINES

There are three supervisory routines used with the ASDF programs. These supervisory routines make the loading of ASDF transparent to the user. There is a substantial amount of file manipulation required to store and load the ASDF programs. A knowledge of the hardware and software available for these purposes is not particularly useful to the user. For these reasons, the supervisory routines are discussed only briefly. Two of these routines are based on
CP/CMS and IBM EXEC environment; the third is written in the IBM-360 Job Control Language.

The JCL program moves the ASDF program from its permanent storage location on DISK02 to the CP/CMS disks OSX001. The two EXEC routines generate an appropriate CP/CMS work space, and manipulate the ASDF text file into a usable form.

The EXEC program ASDF191 resides on the user 191-P-disk. This routine automatically requests and formats ten cylinders of temporary disk space. The ASDF TEXT file is moved from the OSX001 disk to the newly created temporary disk space 192-T. Once in place, The ASDF TEXT file name is changed to ASDF EXEC to generate the next file with the correct file type. ASDF EXEC is then edited to remove the first set of data cards which hold the ASDF192 EXEC program. The 192-T-disk is released and the user is requested to type two commands to the system. The "login 192 P" command makes the 192 disk the P-disk. The "EXEC ASDF192" command begins the execution of the second EXEC file.

The general ASDF192 EXEC structure is depicted in figure 3-6. ASDF192 EXEC is created by ASDF191 EXEC and resides on disk 192. In a manner similar to ASDF191, ASDF192 splits the ASDF TEXT file, which contains all of the component parts of the ASDF program, into a series of smaller files. Once all files have been separated, titles are altered appropriately, text files are loaded into core, and load
Figure 3-6 Structure Of ASDF192 EXEC
modules are generated. ASDF192 then reverts to an infinite
loop, reading and implementing the desired user command.
There are six user commands available to the user: POLZRO,
RESPONSE, EXEC PUNCH, EXEC HRD$CPY, OPTIONS, and INTRO. The
applicability of these commands is briefly discussed in
chapter four. This brief description is also presented by
executing the OPTIONS command once ASDF has been loaded.
IV. STRUCTURE OF ASOF COMMANDS

The ASOF program generates the time and frequency responses of realizable digital filters of order ten or less. Chapter three describes the algorithms used for these computations. The procedure for executing the ASDF program generally consists of two distinct steps. The first step is to enter the poles and zeros of the filter to be analyzed. The second step is to perform the calculations required to generate and plot the desired filter responses. There are six commands which may be executed within the ASDF environment providing the user with the ability to control the form of the output, to enter the poles and zeros of the filter, and to clarify most questions pertaining to the meaning and use of the general commands. The six fundamental commands available to the user are: OPTIONS, INTRO, POLZRO, RESPONSE, EXEC PUNCH, and EXEC HRDS CPY. Only the POLZRO command has a subordinate command structure. All of these commands are issued by typing the command on the terminal keyboard in response to: *ASDF-RE.

The OPTIONS command provides a listing of all commands available to the user. Each command is accompanied by a brief description of its intended use.

The INTRO command writes on the screen six pages of information pertaining to the POLZRO command substructure.
INTRO includes a general description of the data input modes, and a description of several input situations with the appropriate executable commands.

The POLZRO command allows the user to modify or create a filter in the z-plane. All adjustment of poles and zeros must take place with this command. By issuing this command the user may add and delete poles and/or zeros as well as adjust the filter gain constant. The commands which are issued within the POLZRO environment are described in a later section of this chapter.

The RESPONSE command displays the characteristics of the filter being created by the user. Five displays are presented: unit sample response, unit step response, phase of the transfer function, magnitude of the transfer function, and the magnitude of the transfer function in decibels.

The EXEC PUNCH command allows the user to punch a card deck of the filter pole and zero locations on the offline punch. This punched output can be picked up in room 1-140 of the computer center. The cards punched by this command may be subsequently reentered into the system so that this particular filter can be referenced or modified at a later date.

When the user desires to enter a filter in this way, the punched cards are presented to one of the computer operators over the counter in room 1-140. The cards are punched in
the correct format for direct submission into the CP/CMS card reader. The operator subsequently reads this deck into the user's virtual card reader. As the user signs onto a terminal and gains access to CP/CMS, the computer includes an additional message in the standard login typeout showing that the user has a virtual reader file. This message is of the form:

FILES:- 01 RDR, NO PRT, NO PUN

When the user wishes to access this old filter data he must execute the CP/CMS command OFFLINE READ *. The OFFLINE READ command will read the punched card filter data from the user's virtual card reader, and generate a file on the user's disk space entitled FILE FT01F001. Any files with this name present before the OFFLINE READ command is issued will be over-written by the new file. If the user does not wish to lose an older FILE FT01F001, he should use the CP/CMS ALTER command to change the name of the old file that he wishes to protect. Once the user has successfully loaded the ASDF program, the ASDF signature i.e., *ASDF-RE, is presented. The OFFLINE READ command may be issued any time that this signature appears.

The EXEC HRDSCPY command is available only when the SUBMIT function of CP/CMS is functioning. When the SUBMIT command is not functional a message is sent to the user during the login procedure which states that SUBMIT is not available. The output from this command is ultimately
available in the computer center, and consists of two parts. The first part of the output is the numerical listing of the values of the responses computed by the ASDF program. The EXEC HRD$CPY command is the only command which will allow the user access to these numerical values. The second part of the output is a series of Versatec plots, each similar to the screen presentations of the RESPONSE command, with one additional plot of the unit circle and pole/zero locations. When the EXEC HRD$CPY command is executed the ASDF program will ask several questions, using the responses to generate a JOB card. The response to the "ENTER DESIRED JOB NAME" is the title which appears on the final output. ASDF then submits the filter characteristics to the operating system for processing. The hard copy output will be found in the standard output boxes.

A. SUMMARY OF DATA INPUT MODES

There are three modes through which the poles and zeros may be entered into the z-plane: 1) interactive graphics, 2) rectangular, and 3) polar. Each of these modes has particular attributes and limitations.

1. Interactive Graphics (IAG)

This mode was designed to enter poles and zeros rapidly, in cases where accuracy is not of prime importance. The interactive graphics mode is characterized by the appearance of the graphics cursor (the intersection of two
lines which trace across the terminal screen). The position of the cursor is controlled by the two thumb wheels located to the right of the interactive ASCII keyboard on the Tektronix 4012 terminal. The coverage of the interactive graphics cursor is limited to values of X and Y greater than -1.1 and less than +1.1. Zeros outside this area must be entered with either the polar or rectangular commands. Careful adjustment of the cursor can usually locate poles or zeros within .002 units of the desired location. Poles must be located within the unit circle.

2. Rectangular

This mode was designed to enter from the terminal keyboard, poles and zeros which are known in rectangular coordinates. It is slower than the interactive graphics mode because there is a significant amount of screen rewriting required. The rectangular mode allows much greater accuracy in the location of poles and zeros. Poles may not be entered on or outside the unit circle. Zeros must be located within ten units of the z-plane origin.

3. Polar

This mode was designed to enter from the terminal keyboard, poles and zeros which are known in polar coordinates. This mode is also slower than the interactive graphics mode. It allows poles and zeros to be entered to any permissible position in the z-plane. Poles must be entered within the unit circle. Zeros must be entered within
ten units of the z-plane origin. Each polar command has a
description of the constraints on the data to be entered
automatically written on the screen. This mode is
particularly appropriate when constructing all-pass linear
phase filters.

B. THE POLZRO COMMAND SUBSTRUCTURE

In an attempt to maximize execution speed, a minimum of
information is written on the screen, thus reducing the
rewrite time after execution of each command. When program
execution begins, the user is presented with the unit
circle, and a listing of commands on the top left side of
the screen. The commands are abbreviated in the following
manner. The general command type or mode is specified
across the top of the array as INTERACTIVE, RECT, and POLAR
(see figure 4-1). Each of these modes conforms to the above
description. Down the left side of the array are the
commands of the form CRR, where C is the change to be made,
A - for add a root, D - for delete a root. RR is for the
type of root to be acted upon by the command: RZ - real
zero, CZ - complex zero, RP - real pole, and CP - complex
pole. As an example, the box in the center column aligned
with the row titled DCZ stands for the command "Delete a
Complex Zero - Rectangular Coordinates. There are two
unique commands listed as NUC, and LST. The New Unit Circle
(NUC) command simply redraws the unit circle and command

39
UNIT CIRCLE COMMANDS

KEYBOARD

RECT POLAR

CMDS

ACZ ARP DCP NUC LST

Figure 4-1 Basic Display Format
array. This is appropriate when either too many entries have been made to the terminal, and the command array is no longer clear, or when noise in the line from the computer causes a deformed array or unit circle to be drawn. The LST command provides the user with a listing of the current status of the filter being analyzed, in a self explanatory format.

In all cases when the unit circle and command array just described appear on the screen, an audible tone or bell is heard. This bell indicates to the user that the cursor is to be positioned within one of the boxes in the command array. In order to execute a command, the cursor is positioned in the box associated with the desired command. A single keyboard character (any character but the "return" key is acceptable) is then typed. The cursor then disappears. The system presents a carat pointing to the command that the computer is about to execute; the flashing cursor blinks within the box associated with the command which is about to be executed. If the box so indicated is the command that was selected, the user types a "space", an "e" (for execute) and then "carriage return". In cases where a rectangular or polar mode command is to be executed, the screen is then erased, and new information is printed. In cases where an interactive graphics command is to be executed, only the cursor reappears. This second appearance of the cursor is not accompanied by a bell or audible tone.
This indicates that the cursor is to be positioned within the unit circle. Once so positioned, a single keyboard character is typed (not "return"), and, as appropriate, the unit circle is redrawn to reflect the change in pole-zero locations. After this update of the z-plane, the circle and command array are redrawn with the cursor accompanied by the audible tone.

There are only two exceptions to this scenario. First, if the flashing cursor does not flash in the correct box, the wrong command will be executed. If an incorrect box contains the flashing cursor, the user should not type the "space", "e", sequence. Instead, he should type two other characters (e.g., "space", "space") and then "return." The cursor returns along with the audible tone, calling for a readjustment of the cursor, closer to the desired command box. The second exception occurs when the placement of all of the poles and zeros is completed. In this case, after aligning the cursor with any box, the sequence "space", "x" (for exit), "return" should be typed. This will terminate the input, or modification, stage of the program and present the user with a series of options for the next stage of filter processing. At this point a display appears which explains the options available to the user.
C. SUMMARY OF POLZRO SUBCOMMANDS

When the cursor appears along with an audible tone, the user is to position the cursor within the command array, and type a keyboard character (not "return"). When the flashing cursor appears within a box, he types ("space", "e", "return") to execute the specified command. "Space" and any other character but "x" will return the user to the command table for cursor readjustment. To terminate the input, or modification mode, the cursor is positioned over any box, and once the flashing cursor enters the box, the characters "space", "x", "return" are typed.
V. LOADING THE ASDF PACKAGE

The procedure for using the ASDF program consists of either six or seven steps depending on whether or not the ASDF package is resident on OSX001. Usually the program sections of ASDF are stored on the T2314 disk pack OSX001. Each Monday morning the OSX001 disks are scratched so that the ASDF program must be moved from the permanent storage location on DISK02, to the temporary storage location on OSX001. The user may simply sign onto a 4012 terminal, and execute three commands to load ASDF into his virtual machine. Once this loading process has been successfully completed, the commands previously discussed are at the user's disposal for filter analysis.

A. LOGIN PROCEDURE

With one minor exception, the standard login procedures (specified in Section 4.4.3 of the CP/CMS User's Manual) [Ref. 4] are applicable when entering the CP/CMS system to use ASDF. Due to the size of the ASDF programs the user must request a 400k virtual machine when logging onto CP/CMS. The virtual machine is requested as follows:

```
login XXXXTNN 400k
```

where XXXX is the user's computer center number, T is the type of user i.e. G for general user, and P for private
user, and NN is the two digit terminal number. The 400k request in the LOGIN is absolutely mandatory. Without this request the user will be able to load the ASDF program, however, when attempting to execute the RESPONSE command, an error: "INPUT COMMAND ERROR" will result. Should this error occur when the command is correctly spelled the user has no alternative but to logoff and log back on to the system requesting the correct size virtual machine. The remainder of the login procedure is identical to the examples in the CP/CMS user's manual.

B. TESTING OSX001 FOR ASDF PROGRAMS

The second step in loading the ASDF programs is to determine whether or not a copy of the programs exists on OSX001. The ASDF programs are permanently stored on the operating system DISK02. The first time that the programs are used each week, the user must move them from DISK02 to the disk OSX001. To determine whether or not OSX001 has the ASDF package, issue the command:

TOCP ASDF191 EXEC P1

The system will respond:

EXECUTION BEGINS...

237 = OSX001

ENTER DSNAME OR '/*'

The correct user response is:

F0718.asdf191

45
At this time one of two responses will appear. The particular response received is dependent on whether or not ASDF resides on OSX001.

1. **ASDF is on OSX001**
   
   When the ASDF programs are located on OSX001, the system response is:
   
   **EXTENTS EXHAUSTED, EOF SIMULATED.**
   
   **4 BLOCKS 63 RECORDS READ**
   
   This message indicates that file ASDF191 has satisfactorially been transferred to the user's virtual machine and the user is ready to continue with the subsequent loading steps.

2. **ASDF is not on OSX001**
   
   When the ASDF programs are not on OSX001, the system response will be:
   
   **DATA SET NOT FOUND (OS/360 S213)**
   
   **ENTER DSMAPAE OR '/*'**
   
   This response indicates that the user must move the ASDF programs to OSX001 before the loading procedure may be resumed. The simplest method for accomplishing this transfer is to generate a file on CP/CMS which exactly duplicates the Job Control Language program which follows.

   To generate the file which transfers ASDF, logon to CP/CMS. Once the system's responses have been entered and the system is ready, issue the command:
   
   `EDIT ASDF JCL`
Then type in, or read in 1, the program which follows.

```plaintext
// Standard JOB Card (User's Manual Sec. 3.3.2.1)
// EXEC PGM=IEBGENER
// SYSPRINT DD SYSOUT=A
// SYSIN DD DUMMY
// SYSUT1 DD DSN=F0718.ASDF,VOL=SER=DISK02,UNIT=3330,
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),DISP=(OLD,KEEP)
// SYSUT2 DD DSN=F0718.ASDF,VOL=SER=OSX001,UNIT=2314,
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),DISP=(NEW,KEEP),
//       SPACE=(TRK,(40,1),RLSE)
// EXEC PGM=IEBGENER
// SYSPRINT DD SYSOUT=A
// SYSIN DD DUMMY
// SYSUT1 DD DSN=F0718.ASDF191,VOL=SER=DISK02,UNIT=3330,
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DISP=(OLD,KEEP)
// SYSUT2 DD DSN=F0718.ASDF191,VOL=SER=OSX001,UNIT=2314,
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DISP=NEW,KEEP),
//       SPACE=(800,(3,1),RLSE)
//
```

Once this program has been correctly read in to the user's disk, it must be submitted to the operating system using the standard SUBMIT command:

```
SUBMIT ASDF JCL P1 XXXX
```

XXXX is the user's computer center number. If the SUBMIT function is working the following messages will be returned:

```
** CARDS XFERED TO 0098P **
FNAME FTYPE P1 SENT FOR OS BATCH JOB
```

After a short delay the system will send a second message:

```
FROM 0098P33 : JOB SENT TO OS WITH 21 CARDS WRITTEN (Time)
```

The user should then revert back to section B above which

1 A deck containing this program is available from Professor Kirk, Sn 302.
discusses the TOCP command. After giving the system a period of time to execute the JCL job which was just submitted, execution of the TOCP command will indicate that the ASDF files are present. If the operating system has not completed executing the transfer, the TOCP request will respond as in paragraph two above.

B. LOADING ASDF

Once ASDF resides on disk OSX001, the remaining required steps are simple. The user issues the command:

EXEC ASDF191

The system will type a series of lines onto the terminal. At the conclusion of these typewritten lines the user is prompted to issue the next two commands:

LOGIN 192 P

The system responds:

192 REPLACES P (191)

The next required entry is:

EXEC ASDF192

These two commands will cause a very long list of CP/CMS commands to be typed on the screen. Each of these CP/CMS commands is automatically executed by the operating system in the process of loading the ASDF programs. There are no further commands required from the user to complete the loading process.
When the entire ASDF program package is loaded, the user will be presented with a title page and the instruction: "HIT SPACE AND RETURN TO CONTINUE." Once the user has typed "space" and "return" the display will write out a listing of the commands at his disposal. These commands have already been discussed in chapter four.
VI. USING ASDF

Each of the following sections constitutes an example for one of the three input modes. Each example uses only one input mode. In general, these modes may be combined during the generation of any particular filter. The mode selected should be the one most convenient to the user.

A. EXAMPLE ONE - INTERACTIVE GRAPHICS

This first example uses the ASDF program in the interactive graphics mode. The objective is to observe the responses of a digital filter which has two real zeros, one at the origin and another at +0.8, two complex poles located at (0.8 +/- j0.2), and the filter gain is 1.000. The interactive graphics mode is used since the exact locations of the poles and zeros are not critical. For the purposes of this example, it is assumed that the user has already followed the login procedure described in the above paragraphs, and has loaded the ASDF program. The user has already seen the title page, and the two pages of command descriptions. The screen is clear except for the top left corner which shows:

*ASDF-RE

This group of letters is the signature for the ASDF program.
and indicates that the program is ready to execute the user's next command.

Since we must first enter the poles and zeros into the z-plane the appropriate command to type is:

POLZRO

The ASDF program will then draw the screen as shown in figure 6-1, and generate an audible tone. Initially the cursor will be located at some arbitrary position or perhaps may not appear on the screen at all. Movement of the thumb wheels moves the cursor position. Since we wish to execute in the interactive graphics mode, to add a real zero, we move the cursor (intersection of the two lines tracing across the screen) into the box in the left column labeled ARZ, (Add a Real Zero) figure 6-2. We then type any keyboard character except "return". The trace cursor will disappear, and the flashing alphanumeric cursor will appear. The program will print a carat pointing to the box selected by the user. The flashing cursor will move into the box selected.

The user then types the command "space", "e", "return". The flashing cursor disappears, and the trace cursor reappears. The absence of an audible tone at this time indicates that the trace cursor is to be positioned within the unit circle.

The 4012 terminal in the computer center requires a "control-s" after every response.
UNIT CIRCLE COMMANDS

<table>
<thead>
<tr>
<th>CMD</th>
<th>INTERACTIVE</th>
<th>KEYBOARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LST</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-2 Trace Cursor Position For Executing
The Command "Add A Real Zero"-Interactive Graphics Mode
The trace cursor is adjusted to the position of the desired real zero, (the origin in this case), as shown in figure 6-3. Any keyboard character except "return" is then typed. The ASDF program plots the zero at the origin, and returns the trace cursor with the audible tone. The audible tone reminds the user that the trace cursor is to be positioned in the command array. Since another real zero is to be entered, the cursor is positioned as in figure 6-1b again, and the same steps are followed. When the trace cursor reappears without the audible tone, position it within the unit circle as near as possible to the point (0.8 +/- j0.0) as shown in figure 6-4. (To enter the zero precisely at 0.8 the rectangular or polar mode would be used.) The steps for entering the complex pole pair exactly parallel those for the previous entries, and are depicted in figures 6-5, and 6-6.

Once all the poles and zeros have been entered, the trace cursor may be aligned with any command box, and a character typed (not "return"). Once the alphanumeric cursor enters that box, the user types "space", "x", "return". The screen is cleared and the user is asked if he wishes to change the default filter gain. Since the default value is 1.0, the user responds "n", the ASDF program clears and responds with: *ASDF-RE

The next step in the normal sequence is for the user to look at the filter characteristics. To display these
## UNIT CIRCLE COMMANDS

<table>
<thead>
<tr>
<th>CMDS</th>
<th>INTERACTIVE</th>
<th>KEYBOARD</th>
<th>RECT</th>
<th>POLAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUC.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-4 Trace Cursor Position Required To Put A Real Zero At (+0.8)
<table>
<thead>
<tr>
<th>UNIT CIRCLE COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDS</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>ACZ</td>
</tr>
<tr>
<td>ACP</td>
</tr>
<tr>
<td>ARZ</td>
</tr>
<tr>
<td>ARP</td>
</tr>
<tr>
<td>DCZ</td>
</tr>
<tr>
<td>DCP</td>
</tr>
<tr>
<td>DRZ</td>
</tr>
<tr>
<td>DRP</td>
</tr>
<tr>
<td>NUC</td>
</tr>
<tr>
<td>LST</td>
</tr>
</tbody>
</table>

Figure 6-5
Trace Cursor Position
Required To Select The Command
"Add A Complex Pole"-Interactive Graphics Mode
UNIT CIRCLE COMMANDS

<table>
<thead>
<tr>
<th>CMD</th>
<th>INTER-</th>
<th>ACTIVE</th>
<th>KEYBOARD</th>
<th>POLAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACZ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ACP</td>
<td>&gt;0e</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ARZ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ARP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DCZ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DRZ</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DRP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NUC</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LST</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 6-6 Trace Cursor Position Required To Enter A Complex Pole At (0.8+/-0.2)
characteristics the command typed is:

\[ \text{RESPONSE} \]

To get a hard copy of the filter responses the command:

\[ \text{EXEC HRDSCPY} \]

is typed. Figures 6-7a, b, c, d, e, f, and g show the pertinent portions of the output.

3. EXAMPLE TWO - RECTANGULAR MODE

This second filter example is taken from [Ref. 5], page 222. The filter is a fourth-order low pass Chebyshev. The derivation of the transfer function is described in the text. The given transfer function is factored to give the desired pole/zero locations. The filter has two complex zeros at \((-1.0 +/- j0.0)\) and two complex pole pairs, one pair at \((0.7498 +/- j0.5348)\), and one pair at \((0.7774 +/- j0.2120)\). The filter gain is given as 0.001836.

In a manner exactly paralleling the previous example, the user selects the command \text{ACZRCT} (which stands for Add a Complex Zero - Rectangular Coordinates), by positioning the trace cursor in the appropriate box, as in figure 6-8a. Once a keyboard character is typed, and the alphanumeric cursor moves into the box, the "space", "e", and "return" keys are typed. The ASDF program will erase the screen and draw the display shown in figure 6-8b. The user then types in the rectangular coordinates of either complex zero in the first complex zero pair, figure 6-8c. The system will
Figure 6-7e Phase Response For Example Two
Figure 6-7g  Magnitude Of The Transfer Function In Decibels For Example Two
<table>
<thead>
<tr>
<th>CMDS</th>
<th>INTERACTIVE</th>
<th>KEYBOARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LST</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-8a Trace Position Required For Executing The Command "Add A Complex Zero" - Rectangular Coordinates Mode
TO ADD A COMPLEX ZERO TO THE DISPLAY, ENTER THE REAL AND IMAGINARY PARTS WITHIN THE BOXES PROVIDED. EACH NUMBER REQUIRES A DECIMAL. REAL PART MINUS SIGNS MUST BE INCLUDED WITHIN THE BOX. NOTE: ALL COMPLEX ZEROS MUST BE WITHIN TEN UNITS OF THE ORIGIN.

```
REAL
>>[ ] +/- J

IMAGINARY
```

Figure 6-8b Screen Display For The Command ACZRCT
TO ADD A COMPLEX ZERO TO THE
DISPLAY, ENTER THE REAL AND
IMAGINARY PARTS WITHIN THE BOXES
PROVIDED. EACH NUMBER REQUIRES A
DECIMAL. REAL PART MINUS SIGNS
MUST BE INCLUDED WITHIN THE BOX.
NOTE: ALL COMPLEX ZEROS MUST BE
WITHIN TEN UNITS OF THE ORIGIN.

REAL

IMAGINARY

\[ \text{REAL} \quad \text{IMAGINARY} \]

\[ \begin{array}{cc}
-1.0 & \pm J0.0 \\
\end{array} \]

THE NEW COMPLEX ZERO WILL BE \(-1.0000000 \pm J0.0\)

IF CORRECT TYPE 'Y', IF NOT 'N'

\emph{Figure 6-8c} Screen Display After Entering The First Complex Zero
ask whether or not the root location has been correctly entered. The user responds 'y' for yes, 'n' for no. If the root has been incorrectly entered, an 'n' response will rewrite the screen for another attempt. Upon receiving a 'y' response ASDF rewrites the unit circle and command array, and plots the new root. The second complex zero is entered exactly as the first.

The complex poles are entered as were the zeros. The trace cursor is first placed in the box ACPRCT. (Add a Complex Pole - Rectangular Coordinates), as in figure 6-9. Figure 6-10 shows the unit circle after all poles and zeros have been entered. The POLZRO command is terminated by aligning the trace cursor with any box, typing a character, waiting for the flashing alphanumeric cursor to enter the box, and typing "space", "x". The system responds by asking about the filter gain. In this example the filter gain is to be changed to .001836, so the correct response is 'y'. The new filter gain, 0.001836, is typed to terminate the POLZRO command.

To view the filter responses, the user enters the command:

RESPONSE

when the *ASDF-RE signature appears. The hard copy responses of the filter are shown in figures 6-11a through 6-11g.
Figure 6-9 Trace Cursor Position To Execute Command ACPRCT
## UNIT CIRCLE COMMANDS

<table>
<thead>
<tr>
<th>CMD</th>
<th>INTER-</th>
<th>KEYBOARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACZ</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>ACP</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>ARZ</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>ARP</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>DCZ</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>DCZ</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>DRZ</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>DRZ</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>NUC</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>LST</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

Figure 6-10 Unit Circle After All Poles and Zeros Have Been Returned-Example Two
NUMBER OF REAL ZEROES: 0
NUMBER OF COMPLEX ZERO PAIRS: 2
NUMBER OF REAL POLES: 0
NUMBER OF COMPLEX POLE PAIRS: 2
FILTER GAIN: 0.00180000

Figure 6-11a Pole Zero Locations For Example Two
Figure 6-11d Unit Step Response For Example Two
Figure 6-11e Phase Response For Example Two
Figure 6-11f  Magnitude of $H(z)$ for Example Two
Figure 6-11g: Magnitude of H(z) in Decibels For Example Two
C. EXAMPLE THREE - POLAR MODE

This example uses the polar mode to generate an all pass filter. The filter complex pole pairs are to be located at (0.2 at +/- 45 degrees), and (0.5 at +/- 30 degrees). The filter zeros are to be at (5.0 at +/- 45 degrees), and (2.0 at +/- 30 degrees). The filter gain is to be 0.01. The first step is to change the specifications in degrees to radians. Thirty degrees equals .5235 radians. Forty five degrees is .7854 radians. The entry of poles and zeros exactly parallels the above examples, however, the desired commands are ACPPLR, (Add a Complex Pole - Polar Coordinates), and ACZPLR, (Add a Complex Zero - Polar Coordinates). Figure 6-12a shows the trace cursor position for ACPPLR. Figure 6-12b shows the trace cursor position for ACZPLR. The angular part of any pole or zero entered in the polar mode must be in radians. Figure 6-12c shows the entry of the first complex zero pair. Figure 6-12d shows the entry of the first complex pole pair. Figure 6-12e shows the change in filter gain from 1.00 to 0.01. Figures 6-13a through 6-13c show the hard copy responses for this filter. It should be noted that zeros which plot outside of the unit circle are indicated by stars. All such zeros are plotted at a radius of 1.1 along the same angular component as the actual zero location. Zeros are plotted this way for two reasons: first to make the output display consistent for all relevant filters, and second to emphasize zeros which
TO ADD A COMPLEX ZERO TO THE
DISPLAY, ENTER THE RADIAL AND
THETA COMPONENTS WITHIN PROVIDED
BOXES. THE MAGNITUDE OF RHO MUST
BE LESS THAN TEN, THETA(RADIANS)
EACH NUMBER REQUIRES A DECIMAL,
AND AS APPROPRIATE A MINUS SIGN.
BOTH ARE CONFINED TO THE BOXES.

MAGNITUDE

\[ 15.0 \quad /- \quad 78.5 \]

ANGLE

THE NEW COMPLEX ZEROS WILL BE 15.0000000 \(+/-\) 0.7854000
IF CORRECT TYPE 'Y', IF NOT 'N'.

Figure 6-12c  Screen Display After Entering The First Complex
Zero Pair In Polar Coordinates
TO ADD A COMPLEX POLE WITHIN THE UNIT CIRCLE ENTER THE RADIAL AND
THETA COMPONENTS WITHIN PROVIDED BOXES. RHO MUST BE ONE OR LESS.
THETA MUST BE ENTERED IN RADIANS EACH NUMBER requires a decimal,
AND AS APPROPRIATE A MINUS SIGN. 
BOTh ARE CONFINED TO THE BOXES.

MAGNITUDE            ANGLE
>> 0.2 +/- X .7854

THE NEW COMPLEX POLES WILL BE 0.2000000 +/- X 0.7854000
IF CORRECT TYPE 'Y', IF NOT 'N'.

Figure 6-12d  Screen Display After Entering The First Complex
Pole Pair In Polar Coordinates
THE FILTER GAIN IS CURRENTLY 1,00000000
DO YOU WISH TO CHANGE THE FILTER GAIN?
(TYPE Y OR N)
TYPE IN THE NEW FILTER GAIN 12 OR FEWER NUMBERS
INCLUDE A DECIMAL POINT
> 0.01
NEW FILTER GAIN IS: 0.01000000

Figure 6-12e Screen Representation After Changing Filter Gain
<table>
<thead>
<tr>
<th>REAL</th>
<th>IMAGINARY</th>
<th>ZHO</th>
<th>THETA</th>
<th>ZRT</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.5355272</td>
<td>Z +/- 1.5355272</td>
<td>5.0000000</td>
<td>Z +/- 0.7854000</td>
<td>Z</td>
</tr>
<tr>
<td>1.7321472</td>
<td>0.3998287</td>
<td>2.0000000</td>
<td>Z +/- 0.5235003</td>
<td>Z</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6-13a Pole Zero Locations For Example Three**
FILTER POLE ZERO LOCATIONS

UNIT CIRCLE

Figure 6-13b Pole Zero Location for Example Three
Figure 6.13c Unit Sample Response For Example Three
Figure 6-13d  Unit Step Response For Example Three
Figure 6.13f Magnitude of $H(z)$ for Example Three
Figure 6-13g  Magnitude Of H(z) In Decibels For Example Three
are close to, but outside, the unit circle. The importance of distinguishing between zeros very near the unit circle, either interior or exterior, rests on the extreme sensitivity of the phase of the frequency response as a zero moves from the interior, onto, and to the exterior of the unit circle.
The purpose of this chapter is threefold. The primary objective is to provide the user with a series of plots demonstrating the general form of those types of filters not considered in chapter six. Additionally, this chapter demonstrates the responses generated by poles located at a series of positions within the unit circle. It further provides examples for user comparison when learning to interact with the z-plane.

The filters considered in the examples of chapter six include a fourth-order low-pass filter and a second-order all-pass filter. The sequence of figures 7-1a through 7-1g shows the general pole zero placements for a high-pass filter, and associated responses. Similarly, figure 7-2a through 7-2g show the responses for a band-pass filter. Figures 7-3a through 7-3g show a band-reject or notch filter.

The secondary purpose of the chapter is to show the user the general forms of time and frequency responses as the filter poles move in the z-plane. The sequence of figures 7-4 through 7-11 shows the responses for a real pole at various locations on the real axis. Figures 7-12 through 7-19 similarly show the responses of a complex pole pair moving across the z-plane. Frequency responses are not
shown for poles on, or outside of, the unit circle since the frequency response is not defined for these cases. Figures 7-17, 7-20, and 7-21 show the change in responses for a pair of complex poles which move away from the real axis.

The final intent of the chanter is to provide the user with additional figures for comparison should he need further practice in manipulating poles and zeros in the \( z \)-plane. It is important to note that the ASDF program will not allow the user to position poles on or outside the unit circle. This limitation results in several plots which cannot be reproduced by the user.
Figure 7-1a Pole Zero Locations For A Second-Order High-Pass Filter
Figure 7-1d  Unit Step Response For A Second-Order High-Pass Filter
Figure 7-1e  Phase Response For A Second-Order High-Pass Filter
Figure 7-1f  Magnitude Of $H(z)$ For A Second-Order High-Pass Filter
Figure 7-1g  Magnitude Of $H(z)$ In Decibels For A Second-Order High-Pass Filter
Figure 7-2a Pole Zero Locations For A Fourth-Order Band-Pass Filter
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 7-2b Plot of Pole Zero Locations For A Fourth-Order Band-Pass Filter

104
Figure 7-2c  Unit Sample Response For A Fourth-Order Band-Pass Filter
Figure 7-2d Unit Step Response for a Fourth-Order Band-Pass Filter.
Figure 7-2e Phase Response For A Fourth-Order Band-Pass Filter
Figure 7-2f Magnitude of H(z) For a Fourth-Order Band-Pass Filter
Figure 7-28 Magnitude of $H(z)$ for a Fourth-Order Band-Pass Filter
Figure 7-3a Pole Zero Locations For A Fourth-Order Notch Filter
Figure 7-3d Unit Step Response For A Fourth-Order Notch Filter
Figure 7-3e Phase Response For a Fourth-Order Notch Filter
Figure 7-3f  Magnitude Of $H(z)$ For A Fourth-Order Notch Filter
Figure 7-3g Magnitude In Decibels For A Fourth-Order Notch Filter
Figure 7.4b Unit Sample Response for a Single Real Pole at (-1.05)
Figure 7-10 Unit Step Response For A Single Real Pole At (-1.05)
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 7-5a Example of a Single Real Pole on the Unit Circle at (-1.0)
Figure 7-5b Unit Sample Response For A Single Real Pole At (-1.0)
Figure 7-6a Example of A Single Real Pole At (-0.8)
Figure 7-6b: Unit Sample Response For A Single Real Pole at (-1.0)
Figure 7-7d  Phase Response For A Single Real Pole At (-0.8)
Figure 7.6f Magnitude of H(z) in Decibels for a Single Real Pole at (-0.8)
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 7-7a. Example of a Single Real Pole At (-0.3)
Figure 7-7c  Unit Step Response for A Single Real Pole At (-0.3)
Figure 7-7f Magnitude Of H(z) For A Single Real Pole At (-0.3)
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 7-8a Example Of A Single Real Pole At (0.3)
Figure 7-8c  Unit Step Response For A Single Real Pole At (0.3)
Figure 7-8d Phase Response For A Single Real Pole At (0.3)
Figure 7-8f  Magnitude of H(z) in Decibels For A Single Real Pole At (0.3)
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 7-9a Example of a Single Real Pole at (0, 0.8)
Figure 7-9b  Unit Sample Response For A Single Real Pole At (0.8)
Figure 7-9d Phase Response For A Single Real Pole At (0.8)
Figure 7-9f  Magnitude Of H(z) For A Single Real Pole At (0.8)
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 7-10a Example of a Single Real Pole on the Unit Circle at (1.0)
Figure 7-10b  Unit Sample Response For A Single Real Pole At (1.0)
Figure 7-10c: Unit Step Response for a Single Real Pole at (1.0)
Figure 7-11a Example Of A Single Real Pole Outside The Unit Circle At (1.05)
Figure 7-11b  Unit Sample Response For A Single Real Pole At (1.05)
Figure 7-12a  Example Of A Complex Pole Pair At (-0.8+j0.5)
Figure 7-12b  Unit Sample Response For A Complex Pole Pair At (-0.8 ± j0.5)
Figure 7-12c  Unit Step Response For A Complex Pole Pair At (0.8 +j0.5)
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 7.13a Example of a Complex Pole Pair on the Unit Circle At
\((-0.866025, \pm j0.5)\)
Figure 7-13b Unit Sample Response Of A Complex Pole Pair At (-.866025 +j0.5)
Figure 7-13c  Unit Step Response Of A Complex Pole Pair At \((-0.866025 + j0.5)\)
Figure 7.14c  Unit Step Response For a Complex Pole Pair At \((-0.8 + j0.5)\)
Figure 7-14e Magnitude of $H(z)$ for a complex pole pair at $(-0.8+j0.5)$
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 7-15a Example Of A Complex Pole Pair At (-0.3 +j0.5)
Figure 7-15f  Magnitude of $H(z)$ in Decibels For A Complex Pole Pair At $(-0.3 + j0.5)$
Figure 7.16a Example of a Complex Pole Pair At (0.3, j0.5)
Figure 7-16b  Unit Sample Response For A Complex Pole Pair At (0.3 + j0.5)
Figure 7-16d  Phase Response For A Complex Pole Pair At (0.3 +j0.5)
Figure 7-16f  Magnitude Of H(z) In Decibels For A Complex Pole Pair At (0.3 +j0.5)
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 7-17a  Example Of A Complex Pole Pair At (0.8 +j0.5)
Figure 7-17e  Magnitude Of $H(z)$ For A Complex Pole Pair At $(0.8 + j0.5)$
Figure 7-17f  Magnitude Of H(z) In Decibels For A Complex Pole Pair At (0.8 +j0.5)
Figure 7.18a Example Of A Complex Pole Pair On The Unit Circle At (0.866025+j0.5)
Figure 7-18b  Unit Sample Response For A Complex Pole Pair At (0.866025 ±j0.5)
Figure 7-19a  Example of a Complex Pole Pair Outside the Unit Circle At (0.9, j0.5)
Figure 7.19b  Unit Sample Response For A Complex Pole Pair At (0.9 + j0.5)
Figure 7-19c  Unit step response for a complex pole pair at (0.9 + j0.5)
Figure 7-20b  Unit Sample Response Of A Complex Pole Pair At (0.8 +j0.1)
Figure 7-20d Phase Response For A Complex Pole Pair At (0.8 + j0.1)
Figure 7-20e  Magnitude Of H(z) For A Complex Pole Pair At \((0.8 + j0.1)\)
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Example of a Complex Pole Pair At (0.8 + j0.3)

Figure 7-21a
Figure 7.21d  Phase Response For A Complex Pole Pair At (0.8 ±j0.3)
Figure 7-21f Magnitude Of H(z) In Decibels For a Complex Pole Pair At (0.8 ±j0.3)
VIII. INTERPRETATION OF ASDF OUTPUT

When the user initially confronts the ASDF program it is expected that he will attempt to duplicate some of the examples presented earlier in this thesis or similar examples found in the literature. These types of applications of ASDF are straightforward and the results generated can be compared with published results. Interpretation of the output plots is seldom confusing.

As the user becomes more familiar with the use of the ASDF program and the positioning of the poles and zeros in the z-plane he will undoubtedly attempt to generate filters with sharper transition regions, and greater attenuation in the stop band(s). In general these filter improvements are accomplished by increasing the order of the system and positioning the poles closer to the unit circle. As the order of the system increases, and as the poles of the filter become closer to the unit circle, the filter becomes increasingly subject to the difficulties associated with the finite precision arithmetic used in the filter implementation.

A tacit assumption made very early in most digital signal processing courses (and often quickly forgotten) is that infinite precision arithmetic is available in performing the calculations. This assumption breaks down
when digital filters are realized in hardware or simulated. In general there are three sources of error which may affect the performance of digital filters. Due to the finite precision representation of the coefficients, the values of the implemented filter coefficients are only approximately equal to the desired values. Second, errors may be generated in the arithmetic operations. In general, the product of two n-bit numbers results in a 2n-bit result. An error occurs when this 2n-bit number is converted to n-bits for succeeding computations. The third type of error encountered is the analog to digital conversion required for processing analog signals with digital filters. Only the first two of these errors is of concern in the ASDF program.

While an in depth analysis of finite precision effects is beyond the scope of this work, let us consider the filter which has eight zeros located at the origin, and eight poles located at -0.9. Entering these locations into the z-plane using the ASDF program generates the filter responses in figures 6-1a, b, c. A superficial glance at these responses indicates that this filter is unstable. The filter must be stable, however, since all of the poles lie within the unit circle.

There are several phenomena involved in this example. The first is the dynamic range limitation of the system on which the filter is implemented. The IBM-360 is limited to computing with numbers between $16^{64}$ and $-16^{-63}$. Once the
UNIT CIRCLE

FILTER POLE ZERO LOCATIONS

Figure 8-1a Example Filter With Eight Poles At -0.9 And Eight Zeros At The Origin
Figure 8-1c  Step Response For A Filter Exceeding System Dynamic Range Limitations
larger of these limits is exceeded, the computations are terminated and it is not clear whether the designed filter is actually unstable or has merely exceeded the dynamic range limitations of the machine. The dedicated user may decide to modify the programs presented in the appendices to investigate the time responses of such systems. Clearly the form of the time responses will not be significantly altered if a modified unit sample response

$$1 \times 10^{-10}, 0, 0, 0, \ldots$$

and a modified step response

$$1 \times 10^{-10}, 1 \times 10^{-10}, 1 \times 10^{-10}, \ldots$$

is used.

If these modifications are performed, two other phenomena become apparent, each relating to the finite precision implementation of the digital filter. The first is that, in general, digital filters have deadbands, the second that the filter output may degenerate to a limit cycle. The original works by Blackman [Ref. 1] demonstrate that digital filters exhibit both of these effects. Extensive coverage of these phenomena is not appropriate here, however, a dead-band in a digital filter can be simplistically described as a span of filter input values for which the filter output does not change. The limit cycle effect is evident when a digital filter responds to a steady state input. For constant inputs, the filter output may run through a cycle of values. The proneness for
experiencing limit cycles is obvious in the calculation of the step response. A close look at the unit sample response shows that after a sufficiently large number of input values, this unit sample sequence becomes in reaching steady state, a constant.

When observing the output response of the example filter, there is still another possibility for explaining this output. This explanation revolves around the fact that the poles in the implemented filter are not at the locations expected. In fact, the poles may actually move outside the unit circle even though they were entered within. In general, there are only a finite number of pole locations possible with finite word length coefficient representations. The effect of this limitation is best explained using another example. Consider a filter with three zeros located at 0.5, and 0.2 +/- 0.5, and four poles located at -0.999, -0.6, and -0.69 +/- 0.69. The infinite precision transfer function for this filter is expressed in equation 8.1.

\[
\frac{z^3 - 0.9 z^2 + 0.49 z - 0.145}{z^4 + 2.979 z^3 + 3.75822 z^2 + 2.3497398 z + 0.57074868}
\]

Now consider the finite precision case where each of the coefficients must be represented by a six bit number. The pole at -0.6 is expressed in binary (exclusive of sign) as
.10011001100... and is truncated to six bits as .100110.

When all of the coefficients are so represented, the transfer function is expressed by equation 8.2. Clearly the poles and zeros of this second transfer function are not the same in equation 8.1. In general the differences will change all of the filter responses. In fact for this particular case the new poles of the system are at 

\[ \pm 0.70522 \pm \jmath 0.16677, \quad \text{and} \quad \pm 0.76353 \pm \jmath 0.698616; \] this filter is unstable!

The reason for the shift in pole locations is clearly portrayed by the filter whose characteristic equation is equation 8.3.

\[ (Z + 0.995)^8 = 0 \quad (8.3) \]

This filter has eight poles at -0.995. Note what happens to the filter when an error is introduced by truncating the infinite precision coefficient values to thirty-two bits. For simplicity restrict the discussion to the ramifications of an error generated in the value of the constant \((0.995)^8\). This error can be as large as \(2 \times 10^{-10}\). With this error explicitly included, the new characteristic equation will be:

\[ (Z + 0.995)^8 + 2 \times 10^{-10} = 0 \quad (8.4) \]
The roots of equation (8.4) are complex, and lie on a circle of radius 0.06 around the point \(-0.995 + J 0.0\). Figure 8-2 shows that three of these roots lie outside the unit circle. Thus with a finite precision implementation of the filter (even with 32 bits of precision), it is possible for the pole locations to move so far that the filter actually becomes unstable.

The results of this section as well as four types of ASDF outputs which have been observed, are summarized in figure 8-3. Trace A shows the unit sample response of a stable filter. Trace B shows the response of a filter which exceeds the dynamic range limitations of the implementation system. Trace B1 is for an unstable filter and B2 is for a stable filter caught in a limit cycle. These two traces can be observed or explored only by modifying the unit sample sequence. Trace C shows a stable filter which falls into a limit cycle below the dynamic range limit of the machine upon which the implementation has been performed. Note that the performance of a filter above the dynamic range limitation of the IBM 360 (or any other machine) cannot be observed, unless the ASDF program is modified.
Figure 8-2 Three Of These Complex Poles Are Outside The Unit Circle
Figure 8-3 Summary of Possible Observed Unit Sample Responses
IX. SUMMARY AND CONCLUSIONS

The Advanced Simulation of Digital Filters has been implemented on the IBM 360, and is intended for the use by persons beginning their study of digital signal processing. The programs included within the package have been designed to make the user's interaction with the z-plane as simple as possible. While the programs themselves are available for study, all of the computations are invisible to the user, in an attempt to develop the user's intuition of the discrete domain without overwhelming him with arithmetic computations.

The ASDF package provides the user with an interactive method by which the filter pole and zero locations can be manipulated and the effects observed of such movement. ASDF can simulate filters of up to tenth order, and is punctuated with numerous comments and instructions which are to assist the user. The interface with the z-plane is available in a rapid interactive graphics mode, as well as in the slower polar and rectangular modes, each providing greater position location accuracy. The filter responses generated by ASDF are presented as a series of displays directly on the Tektronix 4012 terminal. Versatec plot capability has been integrated into the system to provide high quality copies of the system responses.

212
A. CONCLUSIONS

There are several points which are worth considering for persons intending to pursue this or other areas of interactive systems development with the graphics terminals. While comments may appear obvious they are stated to minimize the difficulties which can be encountered. First, a serious attempt should be made to define the entire data structure before beginning the programming stage of the work. All phases of the effort must be planned before commencing with the actual programming to insure that the data structure selected is sufficiently general, and sufficiently flexible to endure the additions and changes which inevitably occur when initial objectives are overlooked or changed. Second, it is imperative to master all of the software tools available and applicable to the project. There are features of the PLOT 10 software which exist and were not used in the ASDF program. This shortcoming is due to insufficient a priori knowledge. For example, the writing of text portions of the screen presentations should have been much simpler. Finally, it is important to research all of the attributes of the devices used for the final problem solution, and direct the solution to take advantage of all the system capabilities.
B. AREAS FOR FUTURE WORK

There are several areas in which this program could be improved or extended. The possibility exists for transferring the ASDF programs to the Tektronix 4082 computing system. This move would avoid many of the shortcomings associated with the current IBM 360 system. The user would be relatively immune to system crashes, and the constraints placed upon him by concerning the IBM system's being operational and/or available. There are several disadvantages associated with this move. With the advent of the new computer system the majority of the current problems concerning system reliability, and constraints on available access hours should be eliminated. Furthermore, transferring these programs to the 4082 system will require significant rewriting to make execution possible on a system which has a much smaller virtual machine. Furthermore to fully benefit from the 4082 system, the interfacing routines should be rewritten to take advantage of the 4082 refresh capability. It seems that all of the effort involved for this change is not worth the expected improvements.

The ASDF program is now useful for analyzing digital filters. It seems likely that this capability could be expanded into a method for computer-aided design of digital filters. There are standard transformations which allow the filter designer to map analog filter requirements into the
appropriate digital requirements, and as well there are transformations which make possible the computation of digital filter transfer functions from known analog transfer functions. There are also a variety of optimization techniques referenced in the literature which could be implemented.

The other area of obvious expansion for the ASDF program is to include the capability for performing spectral analysis. It would be worthwhile (though perhaps not particularly involved) to include in the system the ability to calculate Discrete Fourier Transforms, and graphically display the results. Additionally, it would be possible to demonstrate the correspondence between performing convolution in the time domain directly and performing the convolution in the frequency domain by multiplying the DFT's. The relation between the circular and linear convolution could also be graphically portrayed.

As a final precautionary note, it is not possible to overestimate the length of time required to generate well documented, correct, useful programs on an interactive basis. Changes in system parameters, and the limitations in system software can make the process of simplifying the user's interactive responsibilities a very difficult process for anyone not thoroughly acquainted with all available resources. The myriad of possible incorrect user responses which must be anticipated make the problem extensive.
APPENDIX A: PROBABLE SOLUTIONS FOR DIFFICULTIES

SYMPTOM: RESPONSE command returns an error message upon execution.
SOLUTION: Check the virtual machine size with the command "CP Q F" to insure a virtual machine size of 400K. If the user has only been allocated 256K, logoff and log back on as directed in chapter five. Load ASDF as described in chapter six.

SYMPTOM: User has inadvertently entered the control program, i.e., CP.
SOLUTION: Type in "b" and "return." Normal program execution should resume.

SYMPTOM: The carat appears with the blinking cursor, but it is not located in the command array.
SOLUTION: Move the trace cursor into the command array, and type a "space." Normal operation should resume.

SYMPTOM: No terminal activity.
SOLUTION: First, check to see whether or not the system has crashed. If the system has crashed, logon as in chapter five, and load ASDF as in chapter six. If the system has not crashed, move the cursor to the center of the screen and
type "space" and "return." If normal operation has not resumed hit the "break" key. When the CP prompt appears, type "b." Normal operation should resume.

SYMPTOM: An error is detected in the execution of ASDF191 EXEC indicating that no temporary disk space is available.
SOLUTION: Check with the consultant or a systems programmer.

SYMPTOM: The user cannot escape from a delete pole or delete zero command in the interactive graphics mode (IAG).
SOLUTION: Implicit in each request to delete from the z-plane is a request, validating the change to be made. In the delete IAG mode the trace cursor returns after illustrating the root to be deleted. Type "y" if the correct root was identified. Type "n" if the wrong root was identified, and attempt the procedure again.

SYMPTOM: The user did not enter the correct amount of data and the program terminated on a 217 error.
SOLUTION: Type "b" and "return." The program should restart with the ASDF signature. At this point reenter the command desired. If executing the POLZRO command it is advisable to execute the list (LST) command to check on the current filter status.
APPENDIX B: IMPORTANT SUBROUTINES AND VARIABLES

The subroutines and variables described in this appendix are applicable to all programs within the ASDF package.

A. SUBROUTINE NAMES

The subroutines within the ASDF package are divided into three major categories: 1) Plot-10 software, 2) Versatec software, and 3) routines generated explicitly for the ASDF programs.

1. Plot-10 Software Subroutines

The Plot-10 subroutines are all adequately described in reference 10, and are listed below.

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERASE</td>
<td>MOVARS</td>
</tr>
<tr>
<td>RECOVR</td>
<td>HOME</td>
</tr>
<tr>
<td>NEULIN</td>
<td>SWINDO</td>
</tr>
<tr>
<td>POLTRN</td>
<td>MOVEA</td>
</tr>
<tr>
<td>LINTRN</td>
<td>DRAWR</td>
</tr>
<tr>
<td>MOVREL</td>
<td>VCURSR</td>
</tr>
<tr>
<td>POINTA</td>
<td>SCURSR</td>
</tr>
<tr>
<td>TABHOR</td>
<td>TABVER</td>
</tr>
<tr>
<td>FIN</td>
<td>FINITT</td>
</tr>
<tr>
<td>DRWARS</td>
<td></td>
</tr>
</tbody>
</table>
2. **Versatec Subroutines**

The Versatec subroutines are listed below, and are explained in detail in reference 7.

- **PLOTS**  PLOT  WINDOW
- **SCALE**  AXIS  SYMBOL
- **LINE**

3. **ASDF Subroutines**

Each of the following subroutines have been generated for the ASDF programs.

- **CRRMMM** - Add and delete poles and zeros from the z-plane. The change "C" can be either "A" for add or "D" for delete. The roots "RR" can be "RZ" for real zero, "CZ" for complex zero, "RP" for real pole, or "CP" for complex pole. Mode "MMM" can be "IAG" for interactive graphics, "POL" for poles, or "RCT" for rectangular.
- **FLTR2** - Prints a listing of the pole and zero locations of the filter being considered.
- **ASUBK** - Generates the $a_k$ coefficients required to evaluate the transfer function.
- **BSUBR** - Generates the $b_p$ coefficients required to evaluate the transfer function.
- **ADJUST** - Insures that the correct number of zeros located at the origin are included in the coefficients $a_k$ and $b_p$.
- **RSPNSE** - Computes the time responses of the filter.
- **FRNACY** - Computes the phase and magnitude of the transfer function for the prescribed filter.
PRPPLT - Scales the ordinate data prior to plotting the responses.
PLTIMP - Plots the unit sample response.
PLTSTP - Plots the unit step response.
PLTTRF - Plots the phase of the transfer function as well as the magnitude in linear and decibel format.
UNTCR - Plots the unit circle with the appropriate poles and zeros.
START - Initializes all arrays and required variables.
FINISH - Checks to insure that the constructed filter is causal, and generates a file to save the pole and zero locations.
ANGLE - Draws the angle sign for polar commands.
ERROR - Controls program flow and error messages for user generated errors.
LST - Prints a listing of the current pole zero location.
PLTTBL - Plots the poles and zeros on the unit circle.
CHCKSZ - Insures that roots are available to be deleted and that the maximum system order will not be exceeded.
UPDATE - Makes changes in the POLZRO array to correspond to user changes in the z-plane.
LOCATE - Searches the POLZRO table for the root closest to that which the user wishes to delete.
STRRTS - Puts poles and zeros into the POLZRO array.
PLOTTRT - Plots a single pole or zero on the unit circle.
B. IMPORTANT VARIABLES

The variables described below pertain to all of the subroutines and programs within the ASDF package.

IROOTS(I) - Keeps track of the number of poles and zeros in the system under consideration. IROOTS(1) is the number of real zeros. IROOTS(2) is the number of complex zero pairs. IROOTS(3) is the number of real poles. IROOTS(4) is the number of complex pole pairs.

POLZRO(I,J) - The locations of the zeros are stored for values of I from one to ten. The pole locations are stored for values of I from 11 through 20. The real part of the rectangular representation of a root location is stored with J equal to one. The imaginary part of the rectangular representation for a root location is stored with J equal to two. The magnitude and angle of the root’s polar representation are stored for J values of three and four, respectively. POLZRO(I,5) holds the root type.

IERROR - Keeps track of the generation of an error. Once the error has been identified the value is set to one. When the error has been explained to the user the value is reset to zero.

A(I) and B(I) - Store the values of the ak and br coefficients.

AA(I) and BB(I) - Store the double precision values of the ak and br coefficients.
YI(I) - Stores the values of the unit sample response to be plotted.
YS(I) - Stores the values of the unit step response to be plotted.
YP(I) - Stores the values of the phase of the transfer function to be plotted.
YM(I) - Stores the magnitudes of the transfer function to be plotted.
YMDB(I) - Stores the transfer function magnitudes in decibels to be plotted.
EN(I) - Stores the time values against which the dependent variables are plotted.
IYIPTS - Holds the number of unit sample points to be plotted.
IYSPTS - Holds the number of unit step response points to be plotted.
NDEN - Is the order of the denominator plus one.
NNUM - Is the order of the numerator plus one.
NOORDER - Is the order of the system under consideration. This can be either of the numerator or denominator.
LAST - Holds the number of points to be plotted.
FLTRGN - Is the value that the user assigns to the filter gain.
ZEROS(I) - Holds the complex coefficients which will be made real to generate the b coefficients.
POLES(I) - Holds the complex coefficients which will be made real to generate the $a_k$ coefficients.

The remainder of the variables are either obvious, or used merely for control of the program flow, i.e. do loop indicies, etc.
APPENDIX C. STRUCTURE FOR ASDF TEXT FILE

The file ASDF TEXT is a composite file made up of all of the required programs and EXEC files for running the ASDF program package. These files have been consolidated to minimize the time required for transfer from DISK02 to OSX001. The structure of the composite file is listed below:

BREAK001
EXEC file ASDF192
BREAK002
TEXT file for POLZRO command
BREAK003
TEXT file for RESPONSE command
BREAK004
Information printed out by the INTRO command
BREAK005
VTEC1 JCL - used for EXEC HRD$CPY
BREAK006
VTEC2 JCL - used for EXEC HRD$CPY
BREAK007
Labels for RESPONSE plots
BREAK008
EXEC file to execute the EXEC PUNCH command
BREAK009
Initial file FT01F001 with no nules or zeros

BREAK010

TEXT file for generating JOB cards

BREAK011

TEXT file for the INTRO command

BREAK012

Information printed by OPTIONS command

BREAK013

Information printed on the title page

BREAK014

EXEC file to execute HRD$CPY command

BREAK015

TEXT file for the OPTIONS command

BREAK016

TEXT file for the software page clear

BREAK017
APPENDIX D. JCL FOR EXEC HRD$CPY COMMAND

The Job Control Language required for the execution of the EXEC HRD$CPY command is provided to give insight into the file manipulation utilized by the ASDF package. Should the permanent locations of any of the portions of the ASDF package change this section of JCL must be changed also.

JCL for VTEC1

//GO EXEC PGM=THEPLT,REGION=180K
//STEPLIB DD UNIT=3330,VOL=SER=DISK03,DISP=SHR,DSN=SI502.ASDF
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT15F001 DD DDNAME=PLOTPARM
//PLOTLOG DD SYSOUT=A
//VECTR2 DD DSN=+%VECTR2,DISP=(,PASS),SPACE=(CYL,(5,5)),UNIT=SYSDA
//VECTR1 DD DSN=+%VECTR1,DISP=(,PASS),SPACE=(TRK,(1,1)),UNIT=SYSDA
//GO SYSIN DD *

JCL for VTEC2

//PLOT EXEC PGM=IEVMAPP,COND=(4,LT)
//FT15F001 DD DDNAME=PLOTPARM
//STEPLIB DD DSN=SYS1.VTECPLOT,DISP=SHR
//PLOTLOG DD SYSOUT=A
//VECTR1 DD DISP=(OLD,DELETE),DSN=%&VECTR1
//VECTR2 DD DISP=(OLD,DELETE),DSN=%&VECTR2
//SYSVECTR DD SYSOUT=(A,,5555)
//VECTTAPE DD DUMMY
//
&COMMENT****************************************************************************** ASDF1001
&COMMENT* SOURCE DECK FOR ASDF COMMAND: EXEC ASDF191 ASDF1002
&COMMENT* ASDF1003
&COMMENT* ASDF1004
&COMMENT* ASDF1005
&COMMENT* ASDF1006
&COMMENT* ASDF1007
&COMMENT GENERATE A STACK TO EDIT THE ASDF92 FILE ASDF1008
&COMMENT OFF OF THE BEGINNING OF THE ASDF TEXT FILE ASDF1009
&COMMENT WHICH IS STORED ON DISK02 ASDF1010
&COMMENT ASDF1011
&BEGINSTACK ASDF1012
FO718*ASDF# ASDF1013
N ASDF1014
D ASDF1015
F BREAK002 ASDF1016
D 50000 ASDF1017
# FILE ASDF192 ASDF1018
&ENDSTACK ASDF1019
&COMMENT ASDF1020
&COMMENT RESERVE AND FORMAT THE TEMPORARY DISK SPACE ASDF1021
&COMMENT CP DEFINE T2314 192 10 ASDF1022
&COMMENT FORMAT T ALL (NOTYPE) ASDF1023
&COMMENT &COMMENT GIVE THE PROGRAMS ACCESS TO ALL REQUIRED LIBRARIES ASDF1024
&COMMENT GLOBAL I TEKLIB SYSLIB ASDF1025
&COMMENT CP SET LINELN 72 ASDF1026
&COMMENT &COMMENT MODIFY THE TOP CP EXEC ON THE SYSTEM DISK ASDF1027
&COMMENT SO THAT IT WILL ACCEPT THE STACK OF ASDF1028
&COMMENT EDIT COMMANDS AND GET ASDF TEXT FROM DISK ASDF1029
&COMMENT LOADMOD OSOSDISK (CLEAR ASDF1030
&COMMENT CP ST 138E4 18111811 ASDF1031
&COMMENT CP ST 1392C 4780CF34 ASDF1032
&COMMENT START * ASDF TEXT T1 237 ASDF1033
&COMMENT &COMMENT GENERATE THE CORRECT FILETYPE FOR ASDF ASDF1034
&COMMENT &COMMENT ALTER ASDF TEXT T1 ASDF EXEC T1 ASDF1035
&COMMENT &COMMENT PERFORM THE EDITING OF THE ASDF TEXT FILE ASDF1036
&COMMENT &COMMENT ASDF1037
&COMMENT CEDIT ASDF EXEC ASDF1038
&COMMENT ALTER ASDF EXEC T1 ASDF TEXT T1 ASDF1039
&COMMENT &COMMENT PREPARE TO MOVE THE USER TO THE
&COMMENT TEMPORARY DISK - INFORM THE USER OF
&COMMENT THE APPROPRIATE COMMANDS TO ISSUE
&PRINT
&PRINT
&PRINT
&PRINT ISSUE THE COMMAND: LOGIN 192 P
&PRINT
&PRINT
&PRINT ISSUE THE COMMAND: EXEC ASDF192
&PRINT
&PRINT
### SOURCE DECK FOR ASDF192 EXEC

**COMMENT**
**COMMENT** GENERATE A STACK OF EDIT COMMANDS TO
**COMMENT** PROCESS THE COMBINED ASDF TEXT FILE.

**BEGSTACK**

<table>
<thead>
<tr>
<th>File</th>
<th>File</th>
<th>File</th>
<th>File</th>
<th>File</th>
<th>File</th>
<th>File</th>
<th>File</th>
<th>File</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASDF2021</td>
<td>ASDF2022</td>
<td>ASDF2023</td>
<td>ASDF2024</td>
<td>ASDF2025</td>
<td>ASDF2026</td>
<td>ASDF2027</td>
<td>ASDF2028</td>
<td>ASDF2029</td>
<td>ASDF2030</td>
</tr>
<tr>
<td>ASDF2031</td>
<td>ASDF2032</td>
<td>ASDF2033</td>
<td>ASDF2034</td>
<td>ASDF2035</td>
<td>ASDF2036</td>
<td>ASDF2037</td>
<td>ASDF2038</td>
<td>ASDF2039</td>
<td>ASDF2040</td>
</tr>
<tr>
<td>ASDF2041</td>
<td>ASDF2042</td>
<td>ASDF2043</td>
<td>ASDF2044</td>
<td>ASDF2045</td>
<td>ASDF2046</td>
<td>ASDF2047</td>
<td>ASDF2048</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FILE

ENDSTACK

COMMENT SPLIT EACH SUCCESSIVE FILE OFF OF THE COMPOSITE ASDF TEXT FILE; THEN CHANGE THE FILE TYPE TO P1.

COMMENT SPLIT ASDF TEXT POLZRO TEXT BREAK002 BREAK003
SPLIT ASDF TEXT POLZRO TEXT BREAK002 BREAK003
SPLIT ASDF TEXT RESPONSE TEXT BREAK003 BREAK004
SPLIT ASDF TEXT INTRO FT04F001 BREAK004 BREAK005
SPLIT ASDF TEXT VTEC1 JCL BREAK005 BREAK006
SPLIT ASDF TEXT VTEC2 JCL BREAK006 BREAK007
SPLIT ASDF TEXT VTEC2 JCL BREAK007 BREAK008
SPLIT ASDF TEXT VTEC2 JCL BREAK008 BREAK009
SPLIT ASDF TEXT VTEC2 JCL BREAK009 BREAK010
SPLIT ASDF TEXT VTEC2 JCL BREAK010 BREAK011
SPLIT ASDF TEXT INTRO TEXT BREAK011 BREAK012
SPLIT ASDF TEXT INTRO TEXT BREAK012 BREAK013
SPLIT ASDF TEXT OPTIONS FT03F001 BREAK012 BREAK013
SPLIT ASDF TEXT OPTIONS FT03F001 BREAK013 BREAK014
SPLIT ASDF TEXT OPTIONS FT03F001 BREAK014 BREAK015
SPLIT ASDF TEXT OPTIONS FT03F001 BREAK015 BREAK016
SPLIT ASDF TEXT CLEAR TEXT BREAK016 BREAK017
SPLIT ASDF TEXT CLEAR TEXT BREAK017

COMMENT EDIT EACH OF THE FILES JUST CONSTRUCTED TO ELIMINATE THE LOGICAL SEPARATORS.

COMMENT THESE SEPARATORS ARE ALL OF THE FORM BREAKXX.
&COMMENT
EDIT POLZRO TEXT P1
EDIT RESPONSE TEXT P1
EDIT INTRO FT04F001 P1
EDIT VTEC1 JCL P1
EDIT VTEC2 JCL P1
EDIT FILE FT02F001 P1
EDIT PUNCH EXEC P1
EDIT FILE FT01F001 P1
EDIT JOB$CRD TEXT P1
EDIT INTRO TEXT P1
EDIT FILE FT03F001 P1
EDIT COVER FT04F001 P1
EDIT HRD$CPY EXEC P1
EDIT OPTIONS TEXT P1
EDIT CLEAR TEXT P1
&COMMENT LOAD EACH OF THE CREATED FILES
&COMMENT AND GENERATE A CORE IMAGE
&COMMENT SO THAT SUBSEQUENT LOADING AND
&COMMENT LINKING IS NOT REQUIRED.
&COMMENT
LOAD POLZRO
GENMOD POLZRO (NOMAP P2)
LOAD RESPONSE
GENMOD RESPONSE (NOMAP P2)
LOAD INTRO
GENMOD INTRO
LOAD JOB$CRD
GENMOD JOB$CRD
LOAD OPTIONS
GENMOD OPTIONS (NOMAP P2)
LOAD CLEAR
GENMOD CLEAR (NOMAP P2)
&TYPEDOUT OFF
ALTER COVER FT04F001 P1 FILE FT04F001 P1
INTRO
ALTER FILE FT04F001 P1 COVER FT04F001 P1
ALTER INTRO FT04F001 P1 FILE FT04F001 P1
OPTIONS
CLEAR
&COMMENT
&COMMENT THESE NEXT STATEMENTS FORM AN
&COMMENT INFINITE LOOP WHICH PROCESSES
&COMMENT SUCCESSIVE ASDF AND CP/CMS COMMANDS.
&COMMENT
-LOOP &PRINT *ASDF-READY
&READ
&ERROR &GOTO -LOOP2
*************************************************
C* SOURCE DECK FOR ASDF COMMAND: POLZRO *
C*
*************************************************
C* THIS IS THE MAIN PROGRAM FROM WHICH ALL SUBROUTINES
C* WILL BE CALLED
C*
COMMON IROOTS(4),POLZRO(20,5),IERROR,IJLD
CALL START
10 CALL UNICIR
IF(IROOTS(1)+2*IROOTS(2) .LE. IROOTS(3)+2*IROOTS(4)) CALL FINISH
IF(IROOTS(1)+2*IROOTS(2) .LE. IROOTS(3)+2*IROOTS(4)) STOP
CALL ERASE
CALL MOVABS(126,425)
CALL ANMODE
WRITE(6,100)
100 FORMAT(1X,'FILTER IS NOT CAUSAL, TAKE APPROPRIATE CORRECTIVE*,
X' ACTION.*)
CALL RECOVR
CALL MOVABS(301,381)
CALL ANMODE
WRITE(6,102)
102 FORMAT('HIT SPACE AND RETURN TO CONTINUE*')
CALL RECOVR
CALL MOVABS(525,337)
CALL ANMODE
READ(5,101) IN
101 FORMAT(A1)
C* CALL RECOVR
IOLD=1
GO TO 10
END

SUBROUTINE LETTER
C*
C* THIS SUBROUTINE GENERATES THE ALPHANUMERIC INFORMATION FOR
C* THE UNIT CIRCLE SUBROUTINE (32)
C*
DIMENSION ICM(13)
DATA ICM/3,HAC2,3,HACR,3,HARZ,3,HARP,3,HDC2,3,HDCR,3,HDRZ,3,HDRP,3,HNUC,
1 3,HLS,T,3,HLEY,3,HRDY,3,HEXY/
C* MOVE THE CURSOR TO THE SCREEN TOP LEFT CORNER (1)
C*
CALL HOME
C* GENERATE A BLANK LINE (1)
C* CALL LINEF
C* ANMODE IS REQUIRED FOR FORTRAN I/O IN GRAPHIC MODE (1)
C* CALL ANMODE
C* PRINT THE TITLES (2)
C* WRITE (6,100)
  100 FORMAT (6X,20HUNIT CIRCLE COMMANDS)
C* GET THE TERMINAL BACK TO THE GRAPHIC MODE (1)
C* CALL RECOVR
C* CALL NEWLIN
C* CALL LINEF
C* CALL ANMODE
C* WRITE (6,101)
  101 FORMAT (9X,18HINTER- KEYBOARD)
C* CALL RECOVR
C* GENERATE A CARRIAGE RETURN AND A LINE FEED (1)
C* CALL NEWLIN
C* CALL ANMODE
C* WRITE (6,102)
  102 FORMAT (3X,25HCMDS ACTIVE RECT POLAR)
C* CALL RECOVR
C* CALL NEWLIN
C* CALL LINEF
C* PRINT THE COMMANDS (6)
C* DO 40 I=1,10
C* CALL ANMODE
C* WRITE (6,103) ICMDF(M)
C* CALL RECOVR
C* CALL LINEF
  103 FORMAT (4X,A3)
C* RETURN
C* END

SUBROUTINE PLOTUC
C* THIS SUBROUTINE DRAWS THE UNIT CIRCLE WITH THE AXES
C* CONFINE THE UNIT CIRCLE TO A 600 X 600 RASTER UNIT SQUARE
C* POSITION THE SCREEN WINDOW
C* XMIN = (1023-600) = 423
C* YMIN = 0
C* CALL SWINDO (393,630,0,630)
C* GENERATE THE UNIT CIRCLE (4)
C* CALL VWINDO (0,1.1,0,360)
C* CALL POLTRN (0,360,0)
C* CALL MOVEA (1,0)
C* CALL DRAWA (1,360)
C* RETURN TO RECTANGULAR COORDINATES
C* VIRTUAL WINDOW: -1.1 LE X,Y LE +1.1 (2)
C* CALL LINTRN
C* CALL VWINDO (-1.1,1.2,-1.1,1.2)
C* GENERATE THE X-AXIS (2)
C* CALL MOVEA (1,0)
C* CALL DRAWA (-2,0)
C* GENERATE THE Y-AXIS (2)
C* CALL MOVEA (0,1)
C* CALL DRAWA (0,-2)
C* GENERATE THE X-AXIS TICK MARKS (5)
C* B IS THE TICK MARK LENGTH IN USER UNITS
C* DO 10 I=1,19
B=.05
IF(I.EQ.5.OR.I.EQ.15)B=.1
CALL MOVEA (-1.+FLOAT(I)*.1,-B/2)
10 CALL DRAWA (0,B)
C* GENERATE THE Y-AXIS TICK MARKS (5)
C* B IS THE TICK MARK LENGTH IN USER UNITS
C* DO 20 I=1,19
B=.05
IF(I.EQ.5.OR.I.EQ.15)B=.1
CALL MOVEA (-B/2,1.+FLOAT(I)*.1)
20 CALL DRAWA (B,0)
RETURN
END

SUBROUTINE BOXUC
C* THE PURPOSE OF THIS SUBROUTINE IS TO GENERATE THE "CHOICE"
C* BOXES FOR THE UNIT CIRCLE SUBROUTINE
C* DIMENSION IBCORD (3,3)
C* DATA DESCRIBING THE LOWER LEFT CORNER (LLC) OF
C* EACH BOX AND THE NUMBER OF BOXES IN EACH COLUMN
C* IBCORD (J,1) IBCORD (J,2) IBCORD (J,3)
C* LLCX LLCY NO. BOXES
C* DATA IBCORD/140,238,336,439,483,483,10,8,8/
C* GENERATE THE "CHOICE" BOXES (10)
C* DO 10 J=1,3
C* MOVE TO THE LOWER LEFT CORNER OF THE BOX (1)
C* CALL MOVABS (IBCORD(J,1),IBCORD(J,2))
C* "IBCORD(J,3)" CONTROLS THE NUMBER OF BLOCKS TO BE DRAWN (10)
C* K=IBCORD(J,3)
      DO 20 I=1,K
C* DRAW THE ACTUAL BOX (4)
C* CALL DRWREL (0,16)
CALL DRWREL (14,0)
      CALL DRWREL (-14,0)
C* PROCEED TO THE NEXT HIGHER BOX IN THE COLUMN (1)
C* 20 CALL MOVREL (0,22)
10 CONTINUE
RETURN
END

SUBROUTINE ACZIAK
COMMON IROOTS(4),POLZRO(20,5), IERROR
CALL CHKKSZ (2,2)
IF (IERROR.NE.0) RETURN
CALL VCURSR (ICHAR, X, Y)
CALL STRRTS (2, 1, X, Y)
RETURN
END

SUBROUTINE APNIAG
COMMON IXXX(4), PXXXXX(20, 5), IERROR
CALL CHKXSZ (2, 4)
IF (IERROR .NE. 0) RETURN
CALL VCURSR (ICHAR, X, Y)

C* INSURE THAT THE POLES ARE WITHIN THE UNIT CIRCLE

C* IF (SORT (X**2 + Y**2) .GT. 1) IERROR = 9
IF (IERROR .NE. 0) RETURN
CALL STRRTS (4, 1, X, Y)
RETURN
END

SUBROUTINE ARNIAG
COMMON IXSSX(4), PXXXXX(20, 5), IERROR
CALL CHKXSZ (1, 1)
IF (IERROR .NE. 0) RETURN
CALL VCURSR (ICHAR, X, Y)

C* INSURE THAT THE POLE IS WITHIN THE UNIT CIRCLE

C* Y = 0
IF (SORT (X**2 + Y**2) .GT. 1) IERROR = 9
IF (IERROR .NE. 0) RETURN
CALL STRRTS (1, 1, X, Y)
RETURN
END

SUBROUTINE DNIAG
COMMON IXXX(4), POLI(20, 5), IERROR
CALL CHKXSZ (-1, 2)
IF (IERROR .NE. 0) RETURN
10 CALL VCURSR(ICHAR,X,Y)
    CALL LOCATE(2,X,Y,LINEPZ)
C*   INDICATE THE COMPLEX ZERO NEAREST THE USER'S CURSOR
C*   WITH A BRIGHT DOT FIVE TIMES (4)
C*
    XX=POLZRO(LINEPZ,1)
    YY=POLZRO(LINEPZ,2)
    IF(POLZRO(LINEPZ,3).LE.1.0) GO TO 15
    ANGLE=ATAN2(YY,XX)
    XX=1.05*COS(ANGLE)
    YY=1.05*SIN(ANGLE)
15    DO 20 NREPS=1,5
    CALL BELL
    IF(Y.LE.0.)CALL POINTA(XX,-YY)
20    IF(Y.GT.0.)CALL POINTA(XX,YY)
    CALL SCURSR(ICHAR,IX,IV)
    IF(ICHAR.NE.121).AND.(ICHAR.NE.89)GO TO 10
    CALL UPDATE(2,LINEPZ)
    CALL NUC
    RETURN
    END

SUBROUTINE DCPIAG
COMMON IXXXX(4),POLZRO(20,5),IERROR
    CALL CHKSSZ(-1,1)
    IF(IERROR.NE.0)RETURN
10   CALL VCURSR(ICHAR,X,Y)
    CALL LOCATE(4,X,Y,LINEPZ)
C*   INDICATE THE COMPLEX POLE NEAREST THE USER'S CURSOR
C*   FIVE TIMES WITH A BRIGHT DOT (4)
C*
    DO 20 NREPS=1,5
    CALL BELL
    IF(Y.LE.0.)CALL POINTA(POLZRO(LINEPZ,1),-POLZRO(LINEPZ,2))
20    IF(Y.GT.0.)CALL POINTA(POLZRO(LINEPZ,1),POLZRO(LINEPZ,2))
    CALL SCURSR(ICHAR,IX,IV)
    IF(ICHAR.NE.121).AND.(ICHAR.NE.89)GO TO 10
    CALL UPDATE(4,LINEPZ)
    CALL NUC
    RETURN
    END

SUBROUTINE DZRIAG
COMMON IXXXX(4),POLZRO(20,5),IERROR
    CALL CHKSSZ(-1,1)
    IF(IERROR.NE.0)RETURN
10 CALL VCURSR(ICHAR,X,Y)
   CALL LOCATE(1,X,0,LINEPZ)
   DO 20 NREPS=1,5
   CALL BELL
20 CALL POINTA(POIZRO(LINEPZ,1),0,)
   CALL SCURSR(ICHAR,IX,IV)
   IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
   CALL UPDATE(1,LINEPZ)
   CALL NUC
   RETURN
END

SUBROUTINE DRPIAG
COMMON IXXXX(4),POLLZRO(20,5),IERRORE
CALL CHKKSZ(-1,1,3)
   IF(IERRORE.NE.0)RETURN
10 CALL VCURSR(ICHAR,X,Y)
   CALL LOCATE(3,X,0,LINEPZ)
   DO 20 NREPS=1,5
   CALL BELL
20 CALL POINTA(POIZRO(LINEPZ,1),0,)
   CALL SCURSR(ICHAR,IX,IV)
   IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
   CALL UPDATE(3,LINEPZ)
   CALL NUC
   RETURN
END

SUBROUTINE UNTCIR
C* THIS SUBROUTINE CONTROLS THE IMPLEMENTATION OF
C* COMMANDS ASSOCIATED WITH THE UNIT CIRCLE
C* DATA I5E/2H.E/
   DATA I5X/I2H X/
   DIMENSION IBCORD(4,3),IHTBL(10),IVTBL(10)
   COMMON IXXXX(4),PXXXX(20,5),IERRORE,IOLO
C* SET THE TABLE SIZE TO TEN
C* CALL TTBLSZ(10)
C* LOCATE THE HORIZONTAL AND VERTICAL TABS (2)
C* DATA IHTBL/128,226,324,0,0,0,0,0,0,0/,
   DATA IVTBL/439,461,483,505,527,549,571,593,615,637/,
C* INITIATE THE GRAPHICS ROUTINES (1)
C* IF(IOLD.EQ.0) CALL INIT
C* IF WORKING ON AN OLD FILTER - CONTINUE WITH IT
C* IF(IOLD.EQ.1) CALL NUC
IIF(IOLD.EQ.1) GO TO 30
IIF(IOLD.EQ.0) CALL OLFLTR
IIF(IOLD.EQ.0) CALL ERASE
IIF(IOLD.EQ.1) CALL NUC
IIF(IOLD.EQ.1) GO TO 30
C* GENERATE THE ALPHANUMERIC I/O FOR THE COMMAND TABLE
C* OF THE UNIT CIRCLE SUBROUTINE (1)
C* CALL LETTER
C* GENERATE THE "CHOICE" BOXES (1)
C* CALL BOXUC
C* GENERATE THE UNIT CIRCLE (1)
C* CALL PLOTUC
C* ALLOW THE USER TO SELECT THE DESIRED COMMAND (1)
C* 30 CALL BELL
CALL SCURSR (II5,IIX,IIY)
C* ADJUST CURSOR FOR INACCURACY ON USER'S PART (1)
C* CALL MOVABS (IIX-63,IIY+3)
C* PUT THE ALPHANUMERIC CURSOR IN THE NEAREST BOX (2)
C* CALL TABHOR(IIHTBL)
CALL TAVVER(IIVBL)
C* RECORD THE LOCATION OF THE CURSOR AFTER
C* THE TAB COMMAND HAS BEEN EXECUTED (1)
C* CALL SEELOC (IA,18)
C* HIGHLIGHT THE CHOOSEN BLOCK / COMMAND WITH THE FLASHING CURSOR (4)
C* CALL ANMODE
READ (5,104) IZ
104 FORMAT(A2)
    CALL RECOVR
C*
C* IF HE HAS RESPONDED WITH "SPACE" AND "X"
C* TERMINATE PROGRAM EXECUTION (2)
C*
    IF(IZ.EQ.IX)RETURN
C*
C* IF THE USER HAS NOT RESPONDED WITH A "SPACE"
C* AND AN "E" THEN DO NOT EXECUTE THIS COMMAND
C* RETURN HIM TO THE CURSOR (1)
C*
    IF(IZ.NE.ISE)GO TO 30
C*
C* ROUTE THE PROGRAM CONTROL TO THE APPROPRIATE (34)
C* SUBROUTINE BASED ON THE TABBED CURSOR LOCATION
C*
40    IF(I.A.EQ.128.AND.IB.EQ.637)CALL ACZIAG
    IF(I.A.EQ.128.AND.IB.EQ.615)CALL ACPIAG
    IF(I.A.EQ.128.AND.IB.EQ.593)CALL ARZIAG
    IF(I.A.EQ.128.AND.IB.EQ.571)CALL ARPCT
    IF(I.A.EQ.128.AND.IB.EQ.549)CALL ARZIAG
    IF(I.A.EQ.128.AND.IB.EQ.505)CALL DRZIAG
    IF(I.A.EQ.128.AND.IB.EQ.483)CALL DRPIAG
    IF(I.A.EQ.128.AND.IB.EQ.461)CALL NUG
    IF(I.A.EQ.128.AND.IB.EQ.439)CALL LST
    IF(I.A.EQ.226.AND.IB.EQ.637)CALL ACZRCI
    IF(I.A.EQ.226.AND.IB.EQ.615)CALL ACPRCT
    IF(I.A.EQ.226.AND.IB.EQ.593)CALL ARZRCI
    IF(I.A.EQ.226.AND.IB.EQ.571)CALL ARPCT
    IF(I.A.EQ.226.AND.IB.EQ.549)CALL ARZRCI
    IF(I.A.EQ.226.AND.IB.EQ.505)CALL DRZRCI
    IF(I.A.EQ.226.AND.IB.EQ.483)CALL DRPRCT
    IF(I.A.EQ.226.AND.IB.EQ.461)CALL LST
    IF(I.A.EQ.324.AND.IB.EQ.637)CALL ACZPLR
    IF(I.A.EQ.324.AND.IB.EQ.615)CALL ACPPPLR
    IF(I.A.EQ.324.AND.IB.EQ.593)CALL ARPPLR
    IF(I.A.EQ.324.AND.IB.EQ.571)CALL ARPPPLR
    IF(I.A.EQ.324.AND.IB.EQ.549)CALL ARZPLR
    IF(I.A.EQ.324.AND.IB.EQ.505)CALL DRZPLR
    IF(I.A.EQ.324.AND.IB.EQ.483)CALL DRPPPLR
    IF(I.A.EQ.324.AND.IB.EQ.461)CALL LST
    IF(IZERROR.NE.3)CALL ERROR
    GO TO 30
END

SUBROUTINE OGLFLTR
COMMON IROOTS(4),POLZRO(20,5),IERROR,IOLD,FLTRGN
CALL MOVABS(126,374)
CALL ANMODE
WRITE(6,100)
100 FORMAT('DO YOU HAVE AN OLD FILTER THAT YOU WISH TO MODIFY?')
CALL RECOVR
CALL MOVABS(399,330)
CALL ANMODE
WRITE(6,101)
101 FORMAT('TYPE Y OR N')
CALL RECOVR
CALL SCURSR(IN,IX,IY)
IF(IN.NE.121.AND.IN.NE.89)RETURN
DO 10 I=1,20
10 READ(1,104) (POLZRO(I,J),J=1,5)
FORMAT(1X,5F12.8)
READ(1,105) (IROOTS(I),I=1,4)
FORMAT(1X,4I12)
READ(1,106) FLTRGN
106 FORMAT(1X,F16.8)
IOLD=1
RETURN
END

SUBROUTINE FINISH
COMMON IROOTS(4),POLZRO(20,5),IERROR,IOLD,FLTRGN
CALL ERASE
CALL ANMODE
REWIND 1
DO 10 I=1,20
WRITE(1,100) (POLZRO(I,J), J=1,5)
10 FORMAT(1X,5F12.8)
10 CONTINUE
WRITE(1,101) (IROOTS(I),I=1,4)
101 FORMAT(1X,4I12)
IF(IOLD.EQ.0)FLTRGN=1.
CALL RECOVR
CALL MOVABS(217,528)
CALL ANMODE
WRITE(6,102) FLTRGN
102 FORMAT('THE FILTER GAIN IS CURRENTLY: ',F12.8)
CALL RECOVR
CALL MOVABS(252,484)
CALL ANMODE
WRITE(6,103)
103 FORMAT('DO YOU WISH TO CHANGE THE FILTER GAIN?')
CALL RECOVR
CALL MOVABS(427,440)
CALL ANMODE  
WRITE(6,104)  
104 FORMAT(' TYPE Y OR N')  
CALL RECOVR  
CALL SCURSR(NEWGN,IX,IY)  
CALL ANMODE  
IF(NEWGN .NE. 121 .AND. NEWGN .NE. 89) GO TO 30  
CALL RECOVR  
CALL MOVABS(182,396)  
CALL ANMODE  
WRITE(6,106)  
106 FORMAT(' TYPE IN THE NEW FILTER GAIN, 12 OR FEWER NUMBERS')  
CALL RECOVR  
CALL MOVABS(357,374)  
CALL ANMODE  
WRITE(6,109)  
109 FORMAT(' INCLUDE A DECIMAL POINT')  
CALL RECOVR  
CALL MOVABS(421,330)  
CALL ANMODE  
READ(5,107) FLTRGN  
107 FORMAT(F12.8)  
CALL RECOVR  
CALL MOVABS(259,286)  
CALL ANMODE  
WRITE(6,108) FLTRGN  
108 FORMAT(' NEW FILTER GAIN IS: ',F16.8)  
110 FORMAT(1X,F16.8)  
30 CALL ANMODE  
WRITE(1,110) FLTRGN  
CALL FIN  
40 RETURN  
END  

SUBROUTINE NUC  
CALL ERASE  
CALL LETTER  
CALL BOXUC  
CALL PLOTUC  
CALL PLTTLB  
RETURN  
END  

SUBROUTINE ACIRCT  
COMMON IXXXXX(4),POLZRO(20,5),IERROR  
DIMENSION LTRACZ(64)  
DATA LTRACZ/4,HTO A,4HDD A,4H CMM,4HPLEX,4HZER,4HD TO,4HTHE  
X,4H ,4HDISP,4HLAY,4H ENT,4HER T,4HHE R,4HEAL ,4HHA NDS/  
ASDF3481  
ASDF3482  
ASDF3483  
ASDF3484  
ASDF3485  
ASDF3486  
ASDF3487  
ASDF3488  
ASDF3489  
ASDF3490  
ASDF3491  
ASDF3492  
ASDF3493  
ASDF3494  
ASDF3495  
ASDF3496  
ASDF3497  
ASDF3498  
ASDF3499  
ASDF3500  
ASDF3501  
ASDF3502  
ASDF3503  
ASDF3504  
ASDF3505  
ASDF3506  
ASDF3507  
ASDF3508  
ASDF3509  
ASDF3510  
ASDF3511  
ASDF3512  
ASDF3513  
ASDF3514  
ASDF3515  
ASDF3516  
ASDF3517  
ASDF3518  
ASDF3519  
ASDF3520  
ASDF3521  
ASDF3522  
ASDF3523  
ASDF3524  
ASDF3525  
ASDF3526  
ASDF3527  
ASDF3528
X 4HIMAG,4HINAR,4HY PA,4HRTS,4HWITH,4HINT,4HNE B
X 4HDES,4HPROC,4HIDEO,4H EA,4HCH N,4HUMBE,4HR RE,4HQUR
X 4HDES,4HPROC,4HIDEO,4H EA,4HCH N,4HUMBE,4HR RE,4HQUR
X 4HDES,4HPROC,4HIDEO,4H EA,4HCH N,4HUMBE,4HR RE,4HQUR
X 4HDES,4HPROC,4HIDEO,4H EA,4HCH N,4HUMBE,4HR RE,4HQUR
X 4HDES,4HPROC,4HIDEO,4H EA,4HCH N,4HUMBE,4HR RE,4HQUR
DATA LTRACZ(64)/HIN. /
CALL CHKSZ(2,2)
IF(IERROR.NE.0)RETURN
10 CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,3
CALL ANMODE
WRITE(6,100)(LTRACZ(I+J-1),J=1,8)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE (6,101)
100 FORMAT (21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE (6,102)
101 FORMAT (27X,4HREAL,9X,9HIMAGINARY)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE (6,103)
102 FORMAT (34X,5H+/- J)
CALL RECOVR
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(10,-22)
CALL DRWREL(-138,0)
CALL MOVREL(212,0)
CALL DRWREL(0,22)
CALL DRWREL(140,0)
CALL DRWREL(0,-22)
CALL DRWREL(-140,0)
CALL MOVREL(0,3)
CALL CARTN
CALL ANMODE
WRITE (6,104)
103 FORMAT (22X,***)
READ (5,104) X,Y
104 FORMAT (5X,7F10.7)
CALL RECOVR
IF(SQRT(X**2+Y**2).GE.10.)IERROR=10
IF(IERROR.NE.0)RETURN
CALL NEWLIN
CALL ANMODE
WRITE (6,105)
CALL NEWLIN
CALL ANMODE
i=ABS(y)
WRITE(6,105) X,Z
105 FORMAT(9X,2SHIFT THE NEW COMPLEX ZERO WILL BE ,F10.7,6H +/- J,F10.7)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE(6,106)
106 FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
CALL SCURSR(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL NUC
CALL STRRTS(2,1,X,Y)
RETURN
END

SUBROUTINE ACPRCT
COMMON IXXXXX(4),POLZRO(20,5),IERROR
DIMENSION LTRACP(64)
DATA LTRACP/4HTD A,4HOD A,4H COM,4HPLEX,4H POL,4HE WI,4HTHIN
X ,4H THE,4HUNIT,4H CIR,4HCLE,4HEN,4HER T,4HEHE R,4HEAL
X ,4HAND,4HIMAG,4HINAR,4H PA,4HRTS,4HW ITH,4HIN T,4HHE B
X ,4HOXES,4HPROV,4HIDED,4HE EA,4HCH N,4HUMBE,4HR RE,4HQUIR
X ,4HE A,4HDECI,4HMAL,4H REA,4HPA,4HR T,4HMUS,4H SIG
X ,4HHS,4HMUST,4H BE,4HNICE,4HME,4HUNER 4HUE T,4HHE B
X ,4HBUX,4HNOTE,4H: AL,4HCL 0,4HMPLE,4H PO,4HLES,4HMUST
X ,4HBE,4HLOC A,4HTED,4HINSI,4HDE T,4HHE U,4HNIT,4HCIRC/
DATA LTRACP(64)/4HLE /
CALL CHKSZ(2,4)
IF(IERROR.NE.0)RETURN
10 CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,579
30 CALL ANMODE
WRITE(6,101)(LTRACP(I+J-1),J=1,8)
100 FORMAT (21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE(6,101)
101 FORMAT (27X,4HREAL,9X,9HIMAGINARY)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE (6,102)
102 FORMAT (34X,5H/- - J)  ASDF3625
CALL RECOVR   ASDF3626
CALL MOVAB$(320,302) ASDF3627
CALL DRWREL(O,22) ASDF3628
CALL DRWREL(138,0) ASDF3629
CALL DRWREL(10,22) ASDF3630
CALL DRWREL(-138,0) ASDF3631
CALL MOVREL(212,0) ASDF3632
CALL DRWREL(0,22) ASDF3633
CALL DRWREL(140,0) ASDF3634
CALL DRWREL(-140,0) ASDF3635
CALL MOVREL(0,3) ASDF3636
CALL CARTN ASDF3637
CALL ANMODE ASDF3638
WRITE (6,103) ASDF3639
103 FORMAT (22X,'*') ASDF3640
READ (5,104) X,Y ASDF3641
104 FORMAT(F10.7,5X,F10.7) ASDF3642
CALL RECOVR ASDF3643
IF(SQRT(X**2+Y**2).GT.1.0)ERROR=10 ASDF3644
IF( ERROR.NE.0) RETURN ASDF3645
CALL NEWLIN ASDF3646
CALL NEWLIN ASDF3647
CALL ANMODE ASDF3648
Z=ABS(Y) ASDF3649
WRITE(6,105) X,Z ASDF3650
105 FORMAT(9X,29H THE NEW COMPLEX POLE WILL BE ,F10.7,6H +/- J,F10.7) ASDF3651
CALL RECOVR ASDF3652
CALL NEWLIN ASDF3653
CALL ANMODE ASDF3654
WRITE(6,106) ASDF3655
106 FORMAT(21X,32H IF CORRECT TYPE 'Y', IF NOT 'N'.) ASDF3656
CALL RECOVR ASDF3657
CALL SCURSR(ICHAR,IX,IY) ASDF3658
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10 ASDF3659
CALL NUC ASDF3660
CALL STRRTS(4,1,X,Y) ASDF3661
RETURN ASDF3662
END ASDF3663

SUBROJINE AR2RCT
COMMON IXXXXX(4),POLZRO(20,5),ERROR
DIMENSION LTRARZ(64)
DATA LIRARZ/4HTO A,SHDD A,4H READ,4HL XE,4HRD V,4HD TH,4HE DI
X 4HS 4HPLY,4H ERT,4HER T,4HRE R,4HEAL,4HPAR,4H WIT
X 4HHIN,4HTHE ,4HDOX,4HPROV,4HIDE,4H TH,4HE NU,4HMBER
X 4HIS,4HTO I,4HNCUL,4HDE A,4H DEC,4HIMAL,4H AND,4H MIN ASDF3669
ASDF3670
ASDF3671
ASDF3672
DATA LTRARZ(64)/4HEN,
CALL CHKSFZ(1,1)
IF(TERROR,NE.0)RETURN
10 CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTRARZ(I+J-1),J=1,8)
CALL RECOVR
30 CALL NEWLIN
100 FORMAT (21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE (6,101)
101 FORMAT (27X,4HREAL)
CALL RECOVR
CALL NEWLIN
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(-138,0)
CALL MOVREL(0,3)
CALL CARTN
CALL ANMODE
WRITE (6,103)
103 FORMAT (22X,'>')
READ (5,104) X
104 FORMAT(F10.7)
CALL RECOVR
IF(ABS(X)GE.10.)TERROR=12
IF(TERROR,NE.0)RETURN
CALL NEWLIN
CALL ANMODE
Z=0
WRITE(6,105) X,Z
105 FORMAT(5X,26H THE NEW REAL ZERO WILL BE ,F10.7,3H +J,F10.7)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
ASDF3676
WRITE(6,106)
106 FORMAT(21X,32H IF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
ASDF3672
IF(ABS(X).GT.1.0) IERROR=13
IF(IERROR.NE.0) CALL ERROR
IF(IERROR.NE.0) RETURN
CALL NEWLIN
CALL NEWLIN
CALL ANMODE
Z=0.
WRITE(6,105) X,Z
105 FORMAT(1X,26H THE NEW REAL POLE WILL BE ,F10.7,3H+J,F10.7)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
106 FORMAT(1X,32H IF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
CALL SCURSI(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89) GO TO 10
CALL NUC
CALL STRRIS(3,1,X,0.)
RETURN
END

SUBROUTINE DCRCT
COMMON IX XXX(5), POLZRO(20,5), IERROR
DIMENSION LTDCZ(64)
DATA LTDCZ(4)=-0.4HELET,4HE A,4HCOMP,4HLEX,4HZERO,4HR FRO
X 4HM 4THE 4HDISP,4HLAY,4H EN T,4H ERT,4H E,4H X4H A
X 4HND 4HY 4HCOOR,4HDINFO,4H TES,4HOF,4HPDI,4HTH N
X 4HEAR 4HONE,4HOF T,4H H E,4H R,4H OOTS,4HTO,4HRE 0,4HELET
X 4HEX,4HENTE,4HR TH,4H ESE,4HVALU,4HES M,4H THI,4H N IM
X 4HE 4HBOX,4HS,4HCRO,4HIDE A,4H DEC,4HIMAL,4H AN
X 4HD 4HMINU,4HS SL,4HGN A,4HS AP,4HPRO P,4HR IAT,4HE
X 4HNO 4HSIGN,4H IND,4H ICAT,4HES P,4HOSIT,4HIV E,4HVALU/
DATA LTDCZ(64)=4HES /
CALL CHKSZ(-1,2)
IF(IERROR.NE.0) RETURN
10 CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100) LTDCZ(I+J-1), J=1,8
30 CALL NEWLIN
100 FORMAT(21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE (6,101)
101 FORMAT(29X,'X',12X,'Y')
CALL RECOVR
CALL NEWLIN
CALL MOVABS(320, 302)
CALL DRWREL(0, 22)
CALL DRWREL(138, 0)
CALL DRWREL(0, -22)
CALL DRWREL(-138, 0)
CALL MOVREL(184, 0)
CALL DRWREL(0, 22)
CALL DRWREL(140, 0)
CALL DRWREL(0, -22)
CALL DRWREL(-140, 0)
CALL MOVREL(0, 3)
CALL CARTN
CALL ANMODE
WRITE(6,103)
103 FORMAT(22X,'**')
READ(5,104)X,Y
104 FORMAT(F10.7,5X,F10.7)
CALL RECOVR
CALL LOCATE(2,X,Y,LINEPZ)
CALL NEWLIN
CALL NEWLIN
CALL ANMODE
WRITE(6,105)POLZRO(LINEPZ,1),POLZRO(LINEPZ,2)
105 FORMAT(1X,2SHDELETED ZERO PAIR WILL BE ,F10.7,6H +/- J,F10.7)
CALL RECOVR
CALL NEWLIN
CALL NEWLIN
CALL ANMODE
WRITE(*,106)
106 FORMAT(2IX,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
CALL SCURS((ICHAR,IX,IV)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL UPDATE(2,LINEPZ)
CALL NUC
RETURN
END

SUBROUTINE DCPRCT
COMMON IXXXXX(4),POLZRO(20,5),IERRO
DIMENSION LTRDCP(64)
DATA LTRDCP/100000,0,HELET,1,HE A,4HCOMP,4HLEX,4HPOLE,4H FRO
X,4HM
X,4HX,4HHE,4HUNIT,4H CIR,4HCLIE,4H ENT,4HER T,4HHE T
X,4HHE U,4HHE P,4HCOOR,4HDINA,4HTES,4HOF T,4HHEP
X,4HINT,4HNEAR,4H ONE,4H OF,4HTHE,4HROOT,4HS TO,4H BE
X,4HDEL,4HELET,4H EN,4HTER,4HTHE,4HVALU,4HES I,4HN TH
X,4HE,4HBOXE,4HSE 1,4HNCLU,4HDE A,4H DEC,4HIMAL,4H, AN
ASDF3817
ASDF3818
ASDF3819
ASDF3820
ASDF3821
ASDF3822
ASDF3823
ASDF3824
ASDF3825
ASDF3826
ASDF3827
ASDF3828
ASDF3829
ASDF3830
ASDF3831
ASDF3832
ASDF3833
ASDF3834
ASDF3835
ASDF3836
ASDF3837
ASDF3838
ASDF3839
ASDF3840
ASDF3841
ASDF3842
ASDF3843
ASDF3844
ASDF3845
ASDF3846
ASDF3847
ASDF3848
ASDF3849
ASDF3850
ASDF3851
ASDF3852
ASDF3853
ASDF3854
ASDF3855
ASDF3856
ASDF3857
ASDF3858
ASDF3859
ASDF3860
ASDF3861
ASDF3862
ASDF3863
ASDF3864
X $HDS $HMLU $HMSI $HGSN A $HSP $HPROP $HRAT $HEZ
X $HNO $HSPM $HIND $HICAT $HES Z $HOSITE $HIVE $HVALU/ASDF3865
DATA LTRDCL(64)/4HES ./ASDF3866
CALL CHCKSZ(-1,4) RETURN ASDF3867
IF (ERROR.NE.0) RETURN ASDF3868
10 CALL ERASE ASDF3869
CALL MOVABS(0,528) ASDF3870
DO 30 I=1,57,8 ASDF3871
CALL ANMODE ASDF3872
WRITE(*,100)(LTRDCL(I+J-1),J=1,8) ASDF3873
CALL RECOVR ASDF3874
30 CALL NEWLIN ASDF3875
CALL NEWLIN ASDF3876
CALL ANMODE ASDF3877
WRITE(*,101) ASDF3878
100 FORMAT (21X,8A4) ASDF3879
101 FORMAT (29X,'X',12X,'Y') ASDF3880
CALL RECOVR ASDF3881
CALL NEWLIN ASDF3882
CALL MOVABS(320,302) ASDF3883
CALL DRWREL(0,22) ASDF3884
CALL DRWREL(138,0) ASDF3885
CALL DRWREL(-138,0) ASDF3886
CALL MVREL(184,0) ASDF3887
CALL DRWREL(0,22) ASDF3888
CALL DRWREL(140,0) ASDF3889
CALL DRWREL(0,-22) ASDF3890
CALL DRWREL(-140,0) ASDF3891
CALL MVREL(0,3) ASDF3892
CALL CTRN ASDF3893
CALL ANMODE ASDF3894
WRITE(*,103) ASDF3895
103 FORMAT (22X,'>') ASDF3896
READ (5,104) X,Y ASDF3897
104 FORMAT (F10.7,5X,F10.7) ASDF3898
CALL RECOVR ASDF3899
CALL LOCATE(4,X,Y,LINPZ) ASDF3900
CALL NEWLIN ASDF3901
CALL ANMODE ASDF3902
WRITE(6,105)POLZRLINEPZ,1,POLZRLINEPZ,2 ASDF3903
105 FORMAT (11X,26HDELETED POLE PAIR WILL BE ,F10.7,6H +/- J,F10.7) ASDF3904
CALL RECOVR ASDF3905
CALL LOCATE(4,X,Y,LINPZ) ASDF3906
CALL NEWLIN ASDF3907
CALL ANMODE ASDF3908
WRITE(6,106) ASDF3909
106 FORMAT(2IX,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
CALL SCURSR(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL UPDATE(4,LINEPZ)
CALL NUC
RETURN
END

SUBROUTINE DRZRCT
COMMON IX XXXX(4),POLZRO(20,5),IERROR
DIMENSION LTRDRZ(64)
DATA LTRDRZ/4H0 D,4HELET,4HE A,4HREAL,4H ZER,4HO FR,4HOM T
X 4HHE 4HDISP,4HLAY,4H ENT,4HER T,4HHE ,4HX C,4HOORD
X 4HIN- 4H ATE,4HOFR A,4HPOL,4HNT N,4HENT,4HTHE,4HRoot
X 4H TO 4HBE D,4HELET,4HED, 4H ENT,4HER T,4HHE V,4Halue
X 4H 4HWITH,4HIN T,4HHE B,4HOX P,4HROI,4HOED, 4H INC
X 4HLUDE,4HA DE,4HCIMA,4HL A,4HNO M,4HINUS,4H SIG,4HNS AS
X 4H 4HAPPR,4HOPRI,4HATE, 4H NO,4H SIG,4HNS IN,4HDICA
X 4H TES,4HAPO,4HSITI,4HVE V,4Halue,4H. ,4H ,4H /
DATA LTRDRZ/64/4H/
CALL CHKSQL(-1,11)
IF(IERROR.NE.0)RETURN
10 CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTRDRZ(I+J-11),J=1,8)
CALL RECOVR
CALL NEWLIN
30 FORMAT((21X,8A4))
CALL NEWLIN
CALL ANMODE
WRITE(6,101)
100 FORMAT(2IX,2*)
CALL RECOVR
CALL NEWLIN
CALL MOVABS(320,302)
CALL DAWREL(0,22)
CALL DAWREL(138,01)
CALL DAWREL(0,-22)
CALL DAWREL(-138,0)
CALL MOVREL(0,3)
CALL CARTN
CALL ANMODE
WRITE(6,103)
103 FORMAT(22X,2*)
READ (5,104) X
FORMAT(F10.7)
CALL RECOVR
CALL LOCATE((1,X,0.,LINEPZ)
CALL NEWLIN
CALL NEWLIN
CALL AMODE
Z=O.
WRITE(6,105)POLZRO(LINEPZ,1),POLZRO(LINEPZ,2)
FORMAT(15X,26HEDELETED REAL ZER0 WILL BE ,F10.7,3H +J,F10.7)
CALL RECOVR
CALL NEWLIN
CALL AMODE
WRITE(6,106)
FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
CALL SCURSR((CHAR,IX,ITY)
IF(CHAR,NE.121.AND. CHAR,NE.89) GO TO 10
CALL UPDATE(1,LINEPZ)
CALL NUC
RETURN
END

SUBROUTINE ORPRCT
COMMON [XXXXYX(4), POLZRO(20,5), IERROR
DIMENSION LTRDRP(64)
DATA LTRDRP(64)/4H /
CALL CHECK(1, 3)
IF(IERROR,NE.0) RETURN
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100) LTRDRP(I+J-1),J=1,8
CALL RECOVR
30 CALL NEWLIN
100 FORMAT (21X,0A4)
CALL NEWLIN
CALL AMODE
WRITE (6,101)
101 FORMAT(28X,'X')
    CALL RECOVR
    CALL NEWLIN
    CALL MOVABS(320,302)
    CALL DRWREL(0,22)
    CALL DRWREL(138,0)
    CALL DRWREL(0,-22)
    CALL DRWREL(-138,0)
    CALL MOVREL(0,3)
    CALL CARIN
    CALL ANMODE
    WRITE(6,103)
103 FORMAT(22X,'*')
    READ(5,104) X
104 FORMAT(F10.7)
    CALL RECOVR
    CALL LOCATE(3,X,0.,LINEPZ)
    CALL NEWLIN
    CALL NEWLIN
    CALL ANMODE
    WRITE(6,105)POLZRO(LINEPZ,1),POLZRO(LINEPZ,2)
105 FORMAT(15X,264)DELETED REAL POLE WILL BE ,F10.7,3H +J,F10.7
    CALL RECOVR
    CALL NEWLIN
    CALL ANMODE
    WRITE(6,106)
106 FORMAT(32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
    CALL RECOVR
    CALL SCUSRK([CHAR,IX, IY])
    IF(CHAR.NE.121. AND. CHAR.NE.89)GO TO 10
    CALL UPDATE(3,LINPZ)
    CALL NUC
    RETURN
END

SUBROUTINE ACIPLR
COMMON IXXXX(4), POLZRO(20,5), IERROR
DIMENSION LTPACZ(164)
DATA LTPACZ/4H0D A,4HDD A,4H COM,4HPLEX,4HZER,4H TO,4H THE X
    4H DISP,4HLAY,4H ENT,4HER T,4H ME R,4HADIA,4HL AN
    4H HD,4HTET,4HA CO,4HMPO,4HTEN,4H WIT,4H HIN,4HPROV
    4H RIDE,4HBOXE,4H T,4H M,4H MIGN,4HTUDE,4H OF,4HRHO
    4H MUST,4HBE L,4HESS,4HTHAN,4H TEN,4H TH,4HETA,4HRADI
    4HAN,4HEACH,4H NUM,4HBER,4HREQU,4HRES,4H A D,4HECIM
    4H HAL,4HAMD,4HAS A,4HPRO,4HPRIA,4HTE A,4H MIN,4HUS S
    4H HDON,4HBOTH,4H ARE,4H COV,4HFINE,4H TO,4H THE,4H BOX/
    DATA LTPACZ(164)/4HES. /
    CALL CHKKSZ2(2,2)

ASDF4009
ASDF4010
ASDF4011
ASDF4012
ASDF4013
ASDF4014
ASDF4015
ASDF4016
ASDF4017
ASDF4018
ASDF4019
ASDF4020
ASDF4021
ASDF4022
ASDF4023
ASDF4024
ASDF4025
ASDF4026
ASDF4027
ASDF4028
ASDF4029
ASDF4030
ASDF4031
ASDF4032
ASDF4033
ASDF4034
ASDF4035
ASDF4036
ASDF4037
ASDF4038
ASDF4039
ASDF4040
ASDF4041
ASDF4042
ASDF4043
ASDF4044
ASDF4045
ASDF4046
ASDF4047
ASDF4048
ASDF4049
ASDF4050
ASDF4051
ASDF4052
ASDF4053
ASDF4054
ASDF4055
ASDF4056
10 IF (ERROR NE. 1) RETURN
    CALL ERASE
    CALL MOVE$(10,528)
    DD 30 I=1,57,8
    CALL ANMODE
    WRITE (6,100) LTPACZ(I+J-1), J=1,8
    CALL RECOVR
    CALL NEWLIN
    CALL RECOVR
    CALL ANMODE
    WRITE (6,101)
100 FORMAT (21X,8A4)
    CALL NEWLIN
    CALL ANMODE
    WRITE (6,102)
101 FORMAT (34X, 'MAGNITUDE', 8X, 'ANGLE')
    CALL RECOVR
    CALL NEWLIN
    CALL ANMODE
    WRITE (6,102)
102 FORMAT (34X,3H+/−)
    CALL RECOVR
    CALL ANMODE
    WRITE (6,103)
103 FORMAT (22X,'+++')
    READ (5,104) RHO, THETA
104 FORMAT (F10.7,5X,F10.7)
    CALL RECOVR
    IF (RHO .GE. 10) I ERROR=14
    IF (ERROR NE.0) CALL ERROR
    IF (ERROR NE.0) RETURN
    CALL NEWLIN
    CALL ANMODE
    THETA=ABS(THETA)
    WRITE (6,105) RHO, THETA
105 FORMAT (4X,30H THE NEW COMPLEX ZEROS WILL BE ,F10.7,6H+/− ,F10.7)
    CALL RECOVR
CALL DWRNL(-45.0)
CALL DWRNL(-25.0)
CALL DWRNL(-15.0)
CALL DWRNL(15.0)
CALL DWRNL(25.0)
CALL DWRNL(45.0)
CALL DWRNL(-90.0)
CALL DWRNL(-135.0)
CALL DWRNL(-180.0)
CALL DWRNL(-225.0)
CALL DWRNL(-270.0)
CALL DWRNL(-315.0)
CALL DWRNL(-359.0)
CALL DWRNL(359.0)
CALL DWRNL(315.0)
CALL DWRNL(270.0)
CALL DWRNL(225.0)
CALL DWRNL(180.0)
CALL DWRNL(135.0)
CALL DWRNL(90.0)
CALL DWRNL(45.0)
CALL DWRNL(-45.0)

CALL ANMODE
WRITE(6,103)
103 FORMAT(22X,'*4')
READ(5,104) RHO, THETA
104 FORMAT(10.7,5X,10.7)
CALL RECOVER
IF(IIHRD.GT.1.0) THEN ERROR=15
IF(IIERROR) RETURN
CALL NEWLIN
CALL NEWLIN
CALL ANMODE
THETA=ABS(THETA)
WRITE(6,105) RHO, THETA
105 FORMAT(9X,3OH'THE NEW COMPLEX POLES WILL BE ,F10.7,6H +/- ,F10.7)
CALL RECOVER
CALL ANGLE(727,260)
CAL NEWLIN
CALL ANMODE
WRITE(6,106)
106 FORMAT(21X,'32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL ANMODE
CALL SCURR(ICHAR, IX, IV)
IF(ICHAR .NE. 121 .AND. ICHAR .NE. 89) GO TO 10
CALL NUC
CALL STARTS(4,2, RHO, THETA)
RETURN
END

SUBROUTINE ARZPLR
COMMON IXXXX(6), PLOZRO(20,5), IERROR
DIMENSION LTPARZ(64)
DATA LTPARZ'/4HDO A,4HDO A,4HREA,4HLE,4HRE,4HDT,4HDO TH,4HDE DI
X,4HSP-4HTHE,4HBOX,4HPROV,4HIDED,4H TH,4HE MA,4HGNIT
X,4HDOE,4HIF R,4HMO M,4HUST 4HBE L,4HESS,4HITAN,4H TEN
X,4H T,4HNUMB,4HER,4HINC,4HODIN,4HDE,4HCIMA,4H AN
X,4HDE,4HIGN,4H AS,4HAPPR,4HOPRI,4HATE,4HMIN,4H BE
X,4HCON,4HHTA1N,4HD W,4HITHI,4HNI TH,4HE 80,4HX,4HNO S
ASDF4153
ASDF4154
ASDF4155
ASDF4156
ASDF4157
ASDF4158
ASDF4159
ASDF4160
ASDF4161
ASDF4162
ASDF4163
ASDF4164
ASDF4165
ASDF4166
ASDF4167
ASDF4168
ASDF4169
ASDF4170
ASDF4171
ASDF4172
ASDF4173
ASDF4174
ASDF4175
ASDF4176
ASDF4177
ASDF4178
ASDF4179
ASDF4180
ASDF4181
ASDF4182
ASDF4183
ASDF4184
ASDF4185
ASDF4186
ASDF4187
ASDF4188
ASDF4189
ASDF4190
ASDF4191
ASDF4192
ASDF4193
ASDF4194
ASDF4195
ASDF4196
ASDF4197
ASDF4198
ASDF4199
ASDF4200
DATA 9TPARZ(60)/4H
CALL CHKSIZE(1,1)
IF(IERROR.NE.0)RETURN
10 CALL ERASE
   CALL MOVABS(0,528)
   DO 30 I=1,57,8
   CALL ANMODE
   WRITE(6,100)(9TPARZ(I+J-1),J=1,8)
   CALL RECOVR
   CALL NEWLIN
30 FORMAT(21X,8A4)
   CALL NEWLIN
   CALL ANMODE
   WRITE(6,101)
101 FORMAT(25x,'MAGNITUDE')
   CALL RECOVR
   CALL NEWLIN
   CALL MOVABS(320,302)
   CALL DRWREL(-22)
   CALL DRWREL(138,0)
   CALL DRWREL(-22)
   CALL DRWREL(-138,0)
   CALL MOVREL(0,3)
   CALL CARTH
   CALL ANMODE
   WRITE(6,103)
103 FORMAT(22x,'1')
   READ(5,104)RHO
104 FORMAT(F10.7)
   CALL RECOVR
   IF(RH0.GT.10)IERROR=16
   IF(IERROR.NE.0)RETURN
   CALL NEWLIN
   CALL ANMODE
   CALL ANMODE
   Theta=0.
   WRITE(6,105)RHO,Theta
105 FORMAT(12X,26HTHE NEW REAL ZERO WILL BE ,F10.7,3X,F10.7)
   CALL RECOVR
   CALL ANGELE(677,260)
   CALL NEWLIN
   CALL ANMODE
   WRITE(6,106)
106 FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
   CALL RECOVR
   CALL SCURSF1CHAR,IX,1Y)
   IF(ICHAR.NE.121.AND.1CHAR.NE.89)GO TO 10
CALL NUC
CALL STRRTS(1,2,RHO,0.)
RETURN
END

SUBROUTINE ARPLR
COMMON 1XXX(4),POLZRO(20,5),IERRO
DIMENSION LTPAR(68)
DATA LTPAR/HTO A,4HDD A,4H REA,4HL PO,4HLE T,4HO TH,4HE DI
X,4HS- ,4HPLAY,4H ENT,4HER A,4H VAL,4HUE O,4HF RH,4H O,
X,4HONE 4HOR L,4HESS),4H IN,4HTHE,4HBOX,4HPRV,4HIED
X,4H ,4HTHE ,4HNUMB,4HER, ,4HINCL,4HUDIN,4HG DE,4HCIMA
X,4HL ,4HAND,4HSIGN,4H AS,4HAPP,4HOPR,4HATE,4HMUST
X,4H BE ,4HCONT,4HAIN,4HOD W,4HTHIN,4H THE,4H BOX,4H
X,4H ,4HOTE,4H TH,4H MEA,4HIGNIT,4H UDE,4HOF R,4HDD M
X,4HMUST ,4HE 0,4HNE 0,4HR LE,4HSS, ,4H ,4H ,4H /
DATA LTPAR(64)/4H /
CALL CHKSSZ(1,3)
IF (ERROR,NE.0) RETURN
10
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTPAR(I+J-1),J=1,8)
CALL RECOV
30 CALL NEWH
100 FORMAT (21X,8A4)
CALL NEWH
CALL ANMODE
WRITE (6,101)
101 FORMAT (25X,'MAGNITUDE')
CALL RECOV
CALL NEWH
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(130,0)
CALL DRWREL(0,22)
CALL DRWREL(-138,0)
CALL MOVREL(0,3)
CALL CARIN
CALL ANMODE
WRITE (6,103)
103 FORMAT (22X,'>')
READ (5,104) RHO
104 FORMAT (F10.7)
CALL RECOV
IF (RHO .GT. 1.0) IERROR=17
IF (ERROR,NE.0) RETURN

CALL NEWLIN
CALL NEWLIN
CALL ANMODE
THETA=0.
WRITE(6,105) RHO,THETA
105 FORMAT(12X,26TH'HE NEW REAL POLE WILL BE ,F10.7,3X,F10.7)
CALL REC0VR
CALL ANGLE(677,260)
CALL NEWLIN
CALL ANMODE
WRITE(6,106)
106 FORMAT(12X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL REC0VR
CALL SCURS(ICHAR,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL NUC
CALL SIRRTS(3,2,RHO,0.1)
RETURN
END

SUBROUTINE DCZPLR
COMMON IXXX(4),POLZRO(20,5),IERRO1R
DIMENSION LTPOCZ(64)
DATA LTPOCZ/4HTO 0.4HELET,4HE A,4HCOMP,4HLEX,4HZERO,4H FRO
X 4HM,4HTHE,4HDI5P,4HLAY,4H ENT,4HER T,4HHE T,4HRHO1
X 4H AND,4HTHE,4HTA1,4HCOOR,4HDINA,4HTES,4HOF A,4H P01
X 4HTHE,4HT,4NA,4HNEAR,4H ONE,4H O F,4HTHE,4HRROOT,4HS TO,4H BE
X 4HBOXE,4HS,4HTES,4HE VA,4HUES,4H IN
X 4HBOXE,4H MUNU,4HS SI,4HSN,4HAP,4HPROP,4HRHAT,4HE
X 4H MUNU,4HSN,4HIND,4HICAT,4HES P,4HOSIT,4HIVE,4HVALU/
DATA LTPOCZ/4HTO 4HELET,4HE A,4HCOMP,4HLEX,4HZERO,4H FRO
CALL CHCKSZ(-1,-2)
IF(IERROR.NE.0)RETURN
10 CALL ERASE
CALL MOVB$0(5,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100)(LTPOCZ(I+J-1),J=1,8)
30 CALL NEWLIN
100 FORMAT(21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE(6,101)
101 FORMAT(21X,'RHO',12X,'THETA')
CALL REC0VR
CALL NEWLIN
CALL REC0VR
CALL NEWLIN
CALL ANMODE
WRITE (6,102)
102 FORMAT (34X,3H+/-)
CALL ANGLE(504,304)
CALL RECOVR
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(0,-22)
CALL DRWREL(-138,0)
CALL MOVREL(212,0)
CALL DRWREL(0,22)
CALL DRWREL(140,0)
CALL DRWREL(0,-22)
CALL DRWREL(-140,0)
CALL MOVREL(0,3)
CALL CARTN
CALL ANMODE
WRITE (6,103)
103 FORMAT (22X,3H0,THETA)
READ (5,104) 3H0,THETA
104 FORMAT(F10.7,5X,F10.7)
CALL RECOVR
X=RH0*COS(THETA)
Y=RH0*SIN(THETA)
CALL LOCATE(2,X,Y,LINEPZ)
CALL NEWLIN
CALL NEWLIN
CALL ANMODE
WRITE(6,105)POLZRO(LINEPZ,3),POLZRO(LINEPZ,4)
105 FORMAT(9X,30HDELETED COMPLEX ZEROS WILL BE ,F10.7,6H+/-,F10.7)
CALL RECOVR
CALL ANGLE(727,260)
CALL NEWLIN
CALL ANMODE
WRITE(6,106)
106 FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
CALL SCURSR(1CHAR,IX,IY)
IF(1CHAR.NE.-121.AND.1CHAR.NE.89)GO TO 10
CALL UPDATE(2,LINEPZ)
CALL NUC
RETURN
END

SUBROUTINE DCPLLR
COMMON 1KXXXX(4), POLZRO(20,5), IERROR
DIMENSION LTOCP(64)
DATA LTPDCP,HTO,D,HELETA,HECOMP,HLEX,HPOLE,HFRO  
X 4M,4THE,5HUNIT,HCLCR,HERT,4HHE  
X 4HRHO,4HSTG,4HTA,4HCODA,4HDINA,4TIES,4HOFAN  
X 4M,4HPOIN,4HTNE,4HARO,4HNEU,4HFT,4HETRO,4HOTS  
X 4HTO,4HSTG,4HED,4HERT,4HHEV,4HVVALUE  
X 4HSIN,4HBX,4HS,4HINC,4HUE,4HDE,4HCIMA  
X 4HL,4HSTG,4HMINU,4HSIL,4HON,4HSAP,4HPRO,4HRAT  
X 4HE,4HNO,S,4HIGN,4HIMPL,4HIES,4HPST,4HTIVG,4HVAL  
DATA LTPDCP(64)/4HUES /*  
CALL CHKKSZ(101)  
IF (ERRNO .NE. 0) RETURN  
10 CALL ERASE  
CALL MOVABS(O,528)  
DO 30 I=1,57,8  
CALL ANMODE  
WRITE(6,100)LTPDCP(I+1),J+1,8  
CALL RECOVR  
CALLNEWLIN  
30 CALLNEWLIN  
100 FORMAT(2IX,8A4)  
CALL NEWLIN  
CALL ANMODE  
WRITE(6,101)  
101 FORMAT(27X,'RHO',12X,'THETA')  
CALL RECOVR  
CALL NEWLIN  
CALL ANMODE  
WRITE(6,102)  
102 FORMAT(34X,'H+/H')  
CALL RECOVR  
CALL ANODE  
WRITE(6,103)  
103 FORMAT(22X,'E')  
READ(5,104)RHOTHETA  
104 FORMAT(F10.7,9X,F10.7)  
CALL RECOVR
X=RHO*COS(THETA)
Y=RHO*SIN(THETA)
CALL LOCATE(4,X,Y,LINEPZ)
CALL NEWLIN
CALL NEWLIN
CALL ANMODE
WRITE(6,105)POLZRO(LINEPZ,3),POLZRO(LINEPZ,4)
105 FORMAT(9X,29DELETED COMPLEX POLE WILL BE ,F10.7,6H +/- ,F10.7)
CALL RECOVR
CALL ANGLE(727,260)
CALL NEWLIN
CALL ANMODE
WRITE(6,106)
106 FORMAT(21X,32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
CALL SCURSR(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL UPDATE(4,LINEPZ)
CALL NUC
RETURN
END

SUBROUTINE DRZPLR
COMMON (XXXXX141, POLZRO(20,5), IERROR
DIMENSION LTPDRZ(64)
DATA LTPDRZ/4H0 0,4HELET,4HE A,4HREAL,4HZER,4HOF 4HOM T
X 4HHE,4HDISP,4HLAY,4H ENT,4HFR,4HFR,4HHR,4HCO
K 4HRDIN,4HRDIN,4HRASE,4HCLAT,4HEDE W,4HTH,4H TH,4HEMA
X 4HFO,4HFZERO,4H, OF,4H A P,4HFINT,4H NEA,4HR,4HE RE
X 4HOT,4HFO,4HDE,4HELET,4HELOPE,4HELOPE,4H THE,4H VAL
X 4HUE,4HUE,4HIE,4HIN,4HNE,4H DX P,4HROV,4HDOM,4H INC
X 4HLODE,4HDE,4HIMA,4HHL,4HHD,4HIN,4H SIG,4H AS
X 4HAI,4HAD,4HAP,4HAP,4HATE,4H NO,4HSIGN,4H IS,4HPSTI/
DATA LTPDRZ/64!/4HIVE/
CALL CHKSTZ(-1,1)
10 IF(ICHAR.NE.0)RETURN
CALL ERASE
CALL MOVABS(0,528)
DO 30 I=1,57,8
CALL ANMODE
WRITE(6,100) (LTPDRZ(I+J-1), J=1,8)
30 CALL NEWLIN
100 FORMAT(21X,8A4)
CALL NEWLIN
CALL ANMODE
WRITE(6,101)
101 FORMAT(27X,'RHO')
CALL RECOVR
CALL NEWLIN
CALL MOVABS(320, 302)
CALL DRWREL(0, 22)
CALL DRWREL(138, 0)
CALL DRWREL(0, -22)
CALL MOVREL(-138, 0)
CALL CARTN
CALL ANMODE
WRITE (6, 103)
103 FORMAT(22X, '***')
READ (5, 104) RHO
104 FORMAT(F10.7, 5X, F10.7)
CALL RECOVR
CALL LOCATE(1, RHO, 0, LINEPZ)
CALL NEWLIN
CALL ANMODE
THETA=0.
WRITE(6, 105) POLZRO(LINEPZ), POLZRO(LINEPZ)
105 FORMAT(2X, 26HDELETED REAL ZERO WILL BE ,F10.7, 3X, F10.7)
CALL RECOVR
CALL ANGLE(677, 260)
CALL NEWLIN
CALL ANMODE
WRITE(6, 106)
106 FORMAT(21X, 32HIF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
CALL SCURS(ICHAR, IX, IY)
IF(ICHAR NE. 121) AND. ICHAR. NE. 89 GO TO 10
CALL UPDATE(1, LINEPZ)
CALL NUC
RETURN
END

SUBROUTINE DRPLPL
COMMON IXXX(4), POLZRO(20, 5), IERROR
DIMENSION LTDPDRP(64)
DATA LTDPDRP/HTO D, 4HELET, 4HE A, 4HREAL, 4H POL, 4HE FR, 4HOM T
X, 4HE, 4HUNIT, 4H CIR, 4H ELE, 4H ENY, 4HER T, 4HHNE, 4HRHO
X, 4tm, 4HCOOR, 4HDINA, 4HTE F, 4HOR A, 4H 'TH, 4META', 4H VAL
X, 4HUE, 4SHOP Z, 4HHERO, 4H OF, 4SHA PO, 4HINT, 4HNEAR, 4H THE
X, 4HROOT, 4H TO, 4HEBE D, 4HELET, 4HED, 4HENTE, 4HR TH
X, 4HE, 4HWALU, 4HE WI, 4HTIN, 4H THE, 4HR BOX, 4HT PRO, 4HVIDE
X, 4HD, 4HINCL, 4HUNE, 4HA DC, 4HCIMA, 4HL, 4HND M, 4HMINUS
X, 4H, 4HSIGN, 4H AS, 4HAPPR, 4HDPRI, 4HATE, 4H, 4H /
DATA LTDPDRP(64)/4H /
CALL CHCKSZ(-1,3)
10 CALL ERASE
10 CALL MOVABS(0,528)
DO 30 I=1,57,8
30 CALL ANMODE
WRITE(6,100)(LTPDRP(I+J-1),I,J=1,8)
CALL RECOVR
CALL NEWLIN
100 FORMAT (21X,'RHO')
CALL RECOVR
CALL NEWLIN
CALL MOVABS(320,302)
CALL DRWREL(0,22)
CALL DRWREL(138,0)
CALL DRWREL(-138,0)
CALL MOVREL(0,3)
CALL CARTN
CALL ANMODE
WRITE (6,101)
101 FORMAT (22X,'RHO')
CALL RECOVR
CALL NEWLIN
CALL LOCATE(3,RHO,0.,LINEPZ)
CALL NEWLIN
CALL ANMODE
WRITE(6,102)
102 FORMAT (12X,26DELETED REAL POLE WILL BE 'F10.7,3X,F10.7)
CALL RECOVR
CALL ANGLE(677,260)
CALL NEWLIN
CALL ANMODE
WRITE(6,103)
103 FORMAT (21X,32IF CORRECT TYPE 'Y', IF NOT 'N'.)
CALL RECOVR
CALL SCURSR(ICHAR,IX,IY)
IF(ICHAR.NE.121.AND.ICHAR.NE.89)GO TO 10
CALL UPDATE(3,LINEPZ)
CALL NUCl
RETURN
SUBROUTINE ANGLE(IX, IY)
CALL MOVABS(IX, IY)
CALL MOVREL(14, -20)
CALL DRWREL(-9, -18)
CALL DRWREL(-3, -1)
CALL DRWREL(-1, 7)
CALL DRWREL(-3, 3)
CALL DRWREL(-9, 2)
CALL MOVREL(0, -20)
RETURN
END

SUBROUTINE ERROR
COMMON ROOTS(4), POLZRO(20, 5), IERROR
CALL LSIT
CALL MOVABS(0, 154)
CALL ANMODE
GO TO (2, 2, 2, 4, 4, 4, 6, 8, 8, 8, 6, 6, 8, 8, 6, 6), IERROR
2 WRITE(6, 100)
100 FORMAT(2IX, 'MAXIMUM SYSTEM ORDER EXCEEDED.' )
GO TO 10
4 WRITE(6, 101)
101 FORMAT(9IX, 'ROOTS NOT AVAILABLE TO BE DELETED.' )
GO TO 10
6 WRITE(6, 102)
102 FORMAT(1IX, 'POLES MAY NOT BE ENTERED OUTSIDE THE UNIT CIRCLE.' )
GO TO 10
8 WRITE(6, 103)
103 FORMAT(10X, 'ZEROS MUST BE WITHIN TEN UNITS OF THE UNIT CIRCLE.' )
10 CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE(6, 104)
104 FORMAT(25X, 'COMMAND NOT PROCESSED.' )
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE(6, 105)
105 FORMAT(1IX, 'WHEN YOU ARE READY TO RETURN HIT', 
1* ANY KEYBOARD KEY*)
CALL RECOVR
CALL SCURSR(ICHAR, IX, IY)
IERROR=0
CALL NUC
END
WRITE(6,106) (POLZRO(I,J), J=1,4), INTYP(ITYPEP)
IF(POLZRO(I,5).EQ.2) OR POLZRO(I,5).EQ.4)
WRITE(6,105) (POLZRO(I,J), J=1,4), INTYP(ITYPEP)
105 FORMAT(1P, 'F10.7, P +/- F10.7, P +/- F10.7,
13X,' P +/- F10.7, ' P +/- A2', ' P')
106 FORMAT(1P, 'F10.7, ' P 'F10.7, ' P 'F10.7,
1P ' F10.7, ' P ' F10.7, ' P')
CALL RECOVR
CALL NEWLIN
20 CONTINUE
CALL ANMODE
WRITE(6,102) (IP, I =1,70)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE (6,107)
107 FORMAT(/)
CALL RECOVR
IF(IERRO. NE.0) RETURN
CALL NEWLIN
CALL ANMODE
WRITE(6,110) IROOTS(1)
110 FORMAT(20X, 'NUMBER OF REAL ZEROS:', I3)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE(6,111) IROOTS(2)
111 FORMAT(20X, 'NUMBER OF COMPLEX ZERO PAIRS:', I3)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE(6,113) IROOTS(3)
113 FORMAT (20X, 'NUMBER OF REAL POLES:', I3)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE (6,114) IROOTS (4)
114 FORMAT (20X, 'NUMBER OF COMPLEX POLE PAIRS:', I3)
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE (6,115)
115 FORMAT (1HIT SPACE THEN RETURN')
CALL RECOVR
CALL NEWLIN
CALL ANMODE
WRITE (6,116)
READ(5,116) IN
116 FORMAT(A1)
CALL RECOVR
CALL NUC
RETURN
END

SUBROUTINE PLTBL
C*
C* SUBROUTINE PLOTS THE POLZRO TABLE OF THE UNIT CIRCLE (12)
C*
COMMON IXXX(4),POLZRO(20,5)
DO 10 I=1,20
C*
C* IF POLZRO(I,5)=0 NO ROOT HAS BEEN STORED
C*
IF(POLZRO(I,5).EQ.0)GO TO 10
CALL PLOT(1,IFIX(POLZRO(I,5)),POLZRO(I,1),POLZRO(I,2))
10 CONTINUE
RETURN
END

SUBROUTINE CHKKSZ (IDELTA, ITYPE)
C*
C* THIS SUBROUTINE CHECKS THE TABLE OF ROOTS TO SEE
C* IF IT IS POSSIBLE TO EXECUTE THE DESIRED SUBROUTINE
C*
CIDELTA: IS THE CHANGE IN THE NUMBER OF ROOTS EXPECTED
C* BY THE CALLING SUBROUTINE
C* ITYPE: IS THE TYPE OF ROOTS TO BE ALTERED, I.E.,
C* COMPLEX, REAL, ZERO, OR POLE
C* IERROR: RETURNS THE ERROR MESSAGE NUMBER IF IT IS
C* NOT POSSIBLE TO EXECUTE THE SELECTED COMMAND;
C* THE MAXIMUM SYSTEM ORDER IS LIMITED TO TEN AND
C* ROOTS MAY NOT BE DELETED IF THEY HAVE NOT BEEN INPUT
C*
COMMON IROOTS(4),PXXXX(20,5),IERROR
C*
IROOTS(1) = FLAG COUNTING REAL ZEROS
C* IROOTS(2) = FLAG COUNTING COMPLEX ZERO PAIRS
C* IROOTS(3) = FLAG COUNTING REAL POLES
C* IROOTS(4) = FLAG COUNTING COMPLEX POLE PAIRS
C*
Determine whether root(s) are being added or deleted (1)
C*
IF(IDELTA.LT.0)GO TO 10
C*
C* ROOT(S) ARE BEING ADDED (3)
C* CALCULATE THE EXPECTED ORDER OF THE SYSTEM (1)
C*
IF (ITYPE.EQ.1 .OR. ITYPE.EQ.2) NORDER=IROOTS(1)+2*IROOTS(2)+IDELTA
IF (ITYPE.EQ.3 .OR. ITYPE.EQ.4) NORDER=IROOTS(3)+2*IROOTS(4)+IDELTA
C*
C* IF "NORDER" IS GREATER THAN 10, MAXIMUM SYSTEM ORDER HAS BEEN
C* EXCEEDED AND AN ERROR RESULTS (1)
C*
20 IF Y.GT.0 CALL POINTA(xx,yy)
RETURN
C*
C* ROOTS ARE BEING DELETED (2)
C* CALCULATE THE EXPECTED ROOT TOTAL IF THE CALLING
C* SUBROUTINE EXECUTES (1)
C* PLUS SIGN IS DUE TO "IDELTA" BEING LESS THAN ZERO
C*
10 IXPTOT=IROOTS(ITYPE)+IDELTA
C*
C* IF THE EXPECTED ROOTS TOTAL "IXPTOT" IS NEGATIVE
C* AN ERROR IS GENERATED (1)
C*
999 IF (IXPTOT.LT.0) ERROR=ITYPE+4
RETURN
END
SUBROUTINE UPDATE(ITYPE,LINEPZ)
C*
C* THIS SUBROUTINE WILL UPDATE THE POLZRO TABLE WHEN A ROOT IS REMOVED
C*
COMMON IROOTS(4),POLZRO(20,5)
C*
"LINEPZ" IS THE LINE OF POLZRO ARRAY TO BE DELETED
C*
IF (LINEPZ.EQ.20) GO TO 45
IF (LINEPZ.GT.10) GO TO 30
IF (LINEPZ.EQ.10) GO TO 15
DO 10 J=LINEPZ,9
DO 10 K=1,5
10 POLZRO(J,K)=POLZRO(J+1,K)
15 DO 20 M=1,4
20 POLZRO(10+M)=-10.
POLZRO(10+5)=0.
GO TO 60
30 DO 40 J=LINEPZ,19
DO 40 K=1,5
40 POLZRO(J,K)=POLZRO(J+1,K)
45 DO 50 M=1,4
50 POLZRO(20+M)=-10.
POLZRO(20+5)=0.
C* ADJUST THE ROOT COUNTER SINCE A ROOT HAS BEEN DELETED
C*
60 IROOTS(IATYPE)=IROOTS(IATYPE)-1
RETURN
END

SUBROUTINE LOCATE(IATYPE,X,Y,LINEPZ)
C*
C* THIS SUBROUTINE LOCATES THE STORED ROOT NEAREST THE CURSOR POSITION
C*
C* COMMON IXXXXX(4), POLZRO(20,5)
C*
C* I"TYPE" IS THE TYPE OF ROOT SEARCHED
C* "LINEPZ" IS THE LINE OF POLZRO WHICH CONTAINS THE ROOT
C* OF THE DESIRED TYPE WHICH IS CLOSEST TO THE
C* USER-POSITIONED CURSOR
C* "DIST" IS THE SHORTEST DISTANCE FOUND FROM STORED ROOT
C* TO THE USER-POSITIONED CURSOR
C*
C* STRRTS STORES ONLY THE COMPLEX POLE IN THE TOP HALF PLANE
C*
SY=ABS(Y)
DIST=20.
DO 10 I=1,20
IF(IFIX*POLZRO(I,5).NE.ITYPE)GO TO 10
CKDIST=SQRT((X-POLZRO(I,1))**2+(SY-POLZRO(I,2))**2)
IF(CKDIST.LT.DIST)LINEPZ=I
IF(CKDIST.LT.DIST)DIST=CKDIST
10 CONTINUE
RETURN
END

SUBROUTINE START
C*
C* THIS SUBROUTINE Initializes ALL THE VARIABLES STORED IN
C* COMMON BEFORE EXECUTION OF THE PROGRAM (10)
C*
C* COMMON IROOTS(4), POLZRO(20,5), IERROR, IOLD
IOLD=0
DO 10 J=1,4
10 IROOTS(J)=0
IERROR=0
DO 20 J=1,20
POLZRO(J,5)=0.
DO 20 K=1,4
20 POLZRO(J,K)=-10.
RETURN
END

SUBROUTINE STRTS(IYPE,MODE,ARGMT1,ARGMT2)
C* THIS SUBROUTINE IS DESIGNED TO STORE ROOTS PREVIOUSLY
C* GENERATED IN THE ROOT TABLE "POLZRO"
C* COMMON IROOTS(4), POLZRO(20,5), IERROR
C* PI=3.14159265
C* IF(ARGMT1.EQ.0..AND.ARGMT2.EQ.0.) GO TO 45
C* GIVEN "MODE", "ARGMT1", AND "ARGMT2", GENERATE THE REAL,
C* IMAGINARY, RADIUS, AND THETA VALUES OF THE UPPER HALF PLANE
C* ROOT TO BE STORED (THIS INCLUDES THE REAL AXIS).
C* GO TO (10,20), MODE
C* "MODE" DICTATES THAT "ARGMT1" AND "ARGMT2" ARE RECTANGULAR
C*
10 RREAL=ARGMT1
     RIMAG=ABS(ARGMT2)
     RADIUS=SQR(T(ARGMT1**2+ARGMT2**2))
     THETA=ATAN2(ARGMT1,ARGMT2)
     GO TO 30
C* "MODE" DICTATES THAT "ARGMT1" AND "ARGMT2" ARE POLAR
C*
50 IF(ARGMT2.LE.PI..AND..ARGMT2.GT.-PI.) GO TO 40
     IF(ARGMT2.LE.-PI.) ARGMT2=ARGMT2+2.*PI
     IF(ARGMT2.GT.PI.) ARGMT2=ARGMT2-2.*PI
     GO TO 50
40 THETA=ARGMT2
     RREAL=RADIUS*CO(S(THETA))
     RIMAG=RADIUS*SIN(THETA)
     GO TO 30
C* IF THE ROOT IS TO BE AT THE ORIGIN,
C* ASSIGN VALUES SUCH THAT ILLEGAL LIBRARY FUNCTION
C* ARGUMENTS WILL NOT BE ASSIGNED
C*
45 RREAL=0.
     RIMAG=0.
     RADIUS=0.
     THETA=0.
C* STORAGE PROCEDURE IS A FUNCTION OF THE TYPE OF ROOTS TO BE STORED
C*
C* 30 GO TO (110,210,120,220), ITYPE
C* A REAL ZERO IS BEING STORED
C* IROOTS(1) IS THE NUMBER OF REAL ZEROS ALREADY STORED
C* 110 POLZRO(IROOTS(1)+IROOTS(2)+1,1)=REAL
   POLZRO(IROOTS(1)+IROOTS(2)+1,2)=0.
   POLZRO(IROOTS(1)+IROOTS(2)+1,3)=ABS(RREAL)
   IF(ARGMT1.GE.0.)THETA=0.
   IF(ARGMT1.LT.0.)THETA=PI
   POLZRO(IROOTS(1)+IROOTS(2)+1,4)=THETA
   IROOTS(1)=IROOTS(1)+1
   GO TO 999
C* A REAL POLE IS TO BE STORED
C* IROOTS(3) IS THE NUMBER OF REAL POLES ALREADY STORED
C* 120 POLZRO(IROOTS(3)+IROOTS(2)+1,1)=REAL
   POLZRO(IROOTS(3)+IROOTS(2)+1,2)=0.
   POLZRO(IROOTS(3)+IROOTS(2)+1,3)=ABS(RREAL)
   IF(ARGMT1.GE.0.)THETA=0.
   IF(ARGMT1.LT.0.)THETA=PI
   POLZRO(IROOTS(3)+IROOTS(2)+1,4)=THETA
   IROOTS(3)=IROOTS(3)+1
   GO TO 999
C* A COMPLEX ZERO IS TO BE STORED
C* IROOTS(2) IS THE NUMBER OF COMPLEX ZERO PAIRS ALREADY STORED
C* 210 POLZRO(IROOTS(2)+IROOTS(2)+1,1)=REAL
   POLZRO(IROOTS(2)+IROOTS(2)+1,2)=ABS(RIMAG)
   POLZRO(IROOTS(2)+IROOTS(2)+1,3)=ABS(RADIUS)
   IF(ARGMT1.EQ.0.)AND.ARGMT2.EQ.0.1 RADIUS=1.
   POLZRO(IROOTS(2)+IROOTS(2)+1,4)=ARCS(RREAL/ABS(RADIUS))
   POLZRO(IROOTS(2)+IROOTS(2)+1,5)=2.
   IROOTS(2)=IROOTS(2)+1
   GO TO 999
C* A COMPLEX POLE IS TO BE STORED
C* IROOTS(4) IS THE NUMBER OF COMPLEX POLE PAIRS ALREADY STORED
C* 220 POLZRO(IROOTS(4)+IROOTS(3)+1,1)=REAL
   POLZRO(IROOTS(4)+IROOTS(3)+1,2)=RIMAG
   POLZRO(IROOTS(4)+IROOTS(3)+1,3)=ABS(RADIUS)
   IF(ARGMT1.EQ.0.)AND.ARGMT2.EQ.0.1 RADIUS=1.
POLZRO/IROOTS(4)+IROOTS(3)+11,4) = ARCOS(RREAL/ABS(RADIUS))
POLZRO/IROOTS(4)+IROOTS(3)+11,5) = 4.

IF(IROOTS(4)+IROOTS(4)+1
   94 IF(IYPE.EQ.1.OR.IYPE.EQ.3) CALL PLOTRT(IYPE,RREAL,0.)
   94 IF(IYPE.EQ.2.OR.IYPE.EQ.4) CALL PLOTRT(IYPE,RREAL,RIMAG)
   94 RETURN
   94 END

SUBROUTINE PLOTRT(IYPE, RREAL, RIMAG)
C* THE PURPOSE OF THIS SUBROUTINE IS TO PLOT
C* THE ROOT JUST ENTERED INTO THE SYSTEM
C* DIMENSION IZERO(16), ISTAR(16)
C* DATA POINTS REQUIRED FOR DRAWING ZEROS
C* DATA IZERO/2,-2,0,-2,-2,-2,-2,2,2,2,2,2,0/
C* DATA ISTAR/-5,5,10,-10,-5,10,0,-10,5,10,-10,10,5,-10,0/
C* IF ZERO IS OUTSIDE THE UNIT CIRCLE, PLOT A STAR.
C* RADIUS = SQRT(RREAL**2 + RIMAG**2)
C* IF(RADIUS.GT.1.) GO TO 300
C* SINCE ROOT IS INSIDE THE UNIT CIRCLE
C* IF THE ROOT IS A ZERO GO TO 100
C* IF THE ROOT IS A POLE GO TO 200
C* RZ CZ RP CP
C* GO TO (100,100,200,200),IYPE
C* CENTER THE ZERO ON THE LOCATION STORED IN POLZRO
C* 100 CALL MOVEA(RREAL,RIMAG)
C* GENERATE THE ZERO (3)
C* CALL MOVREL(1,3)
   120 DO I = 1,15,2
   120 CALL DWRREL(IZERO(I),IZERO(I+1))
C* IF THE ZERO IS REAL, RETURN
C* IF(IYPE.EQ.1.) GO TO 999
C* PLAT THE CORRESPONDING COMPLEX ZERO IN THE LOWER HALF PLANE (4)
CALL MOVEA(RREAL,-RIMAG)
CALL MOVREL(1,3)
DO 130 I=1,15,2
130 CALL DRWREL(IZER0(I),化的(2))
  GO TO 999
C* PLOT THE POLE (REAL, OR UPPER HALF PLANE) (5)
C*
200 CALL MOVEA(RREAL,RIMAG)
CALL MOVREL(3,3)
CALL DRWREL(-6,-6)
CALL MOVREL(0,5)
CALL DRWREL(-6,-6)
C*
C* IF THE POLE IS REAL, RETURN
C*
  IF(ITYPE.EQ.3)GO TO 999
C*
C* IF THE POLE IS COMPLEX, PLOT THE CORRESPONDING
C* LOWER HALF PLANE POLE (5)
C*
CALL MOVEA(RREAL,-RIMAG)
CALL MOVREL(3,3)
CALL DRWREL(-6,-6)
CALL MOVREL(0,5)
CALL DRWREL(-6,-6)
GO TO 999
300 ANGLE=ATAN2(RIMAG,RREAL)
  PREAL=1.05*COS(ANGLE)
  PRIMAG=1.05*SIN(ANGLE)
  CALL MOVEA(PREAL,PRIMAG)
  DO 320 M=1,2
  DO 310 I=1,16,4
    CALL MOVREL(ISTAR(I),ISTAR(I+1))
    CALL DRWREL(ISTAR(I+2),ISTAR(I+3))
  310 CONTINUE
  IF(M.EQ.2 OR ITYPE.EQ.1)GO TO 999
  CALL MOVEA(PREAL,-PRIMAG)
  320 CALL MOVEA(PREAL,-PRIMAG)
  999 RETURN
END
APPENDIX H: SOURCE DECK FOR ASDF COMMAND: RESPONSE

C**************************************************************
C* SOURCE DECK FOR ASDF COMMAND: RESPONSE *
C**************************************************************
C* COMMON IROOTS(4), POLZRO(20,5), IERROR
C* COMMON /ABVCTR/ A(I1), B(I1)
C* COMMON /IMPULS/ YI(1025), YS(1025), EN(1025)
C* COMMON /FREQ/ YP(1026), YM(1026), PI(1026)
C* LINEAR=1
C* LOG=2
C*
C* INITIALIZE A AND B TO ZERO
C*
C* DO 5 K=1,20
C* A(K)=0.
C* B(K)=0.
C*
C* READ THE POLZRO TABLE OFF OF THE FILE
C*
C* DO 20 I=1,20
C* 100 FORMAT(1X,5F12.8)
C* 20 READ(I,100) (POLZRO(I,J),J=1,5)
C*
C* READ THE IROOTS VALUES FROM THE FILE
C*
C* READ(1,101) (IROOTS(I),I=1,4)
C* 101 FORMAT(1X,4I2)
C* 102 FORMAT(1X,F12.8)
C* FLTRGN=ABS(FLTRGN)
C* ICHAR=0
C* PRINT OUT A TABLE OF THE POLES AND ZEROS
C* CALL FLTR2(FLTRGN)
C* GENERATE THE A(K) COEFFICIENTS
C* CALL ASUBK(A,NORDER)
C* NDEN=NORDER+1
C* GENERATE THE B(K) COEFFICIENTS
C* CALL ASUBR(B,NORDER)
C* NNUM=NORDER+1
C*
C* CHECK TO SEE IF THE FILTER IS CAUSAL
C*  IF(NNUM.GT.NDEN)WRITE(6,103)
103 FORMAT(1I1,'* FILTER IS NOT CAUSAL - PROGRAM TERMINATES*')
C*  IF(NNUM.GT.NDEN)STOP
C* ADJUST THE COEFFICIENTS IF NNUM DOES NOT EQUAL NDEN
C*  IF(NNUM.LT.NDEN)CALL ADJUST(NNUM,NDEN)
C* COMPUTE THE UNIT SAMPLE RESPONSE
C*  CALL RSPNSE(0,FLTRGN)
C* PLOT THE UNIT SAMPLE RESPONSE
C*  CALL PLTIMG(ICHAR)
C* IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
C* CALCULATE THE STEP RESPONSE
C*  CALL RSPNSE(1,FLTRGN)
C* PLOT THE STEP RESPONSE
C*  CALL PLTITP(ICHAR)
C* IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
C* COMPUTE THE MAGNITUDE AND PHASE OF THE TRANSFER FUNCTION
C*  CALL FRQCNY(NNUM,NDEN,FLTRGN)
C* PLOT THE PHASE OF THE TRANSFER FUNCTION
C*  CALL PLTTRF3(LINEAR,ICHAR)
C* IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
C* PLOT THE MAGNITUDE OF THE TRANSFER FUNCTION
C* FIRST WITH LINEAR AXIS, THEN IN DECIBELS
C*  CALL PLTTRF4(LINEAR,ICHAR)
C* IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
CALL PLTTRF4(LOG,ICHAR)
IF(ICHAR.EQ.88.OR.ICHAR.EQ.120)GO TO 30
STOP
END
SUBROUTINE ASUBK (POLE,NORDER)
C* GENERATE THE AK COEFFICIENTS FROM THE POLES
C*
COMMON IROOTS(4), POLZRO(20,5), IERROR
COMPLEX POLES(11), AODEF(11)
REAL POLE(11)
C* INITIALIZE ALL THE VECTORS AND INDICES TO ZERO
C*
DO 5 K=1,11
POLES(K)=0,0.
POLE(K)=0.
AODEF(K)=0.
5
C* "J" INDEXES THE TABLE "POLES"
C* "K" INDEXES THE TABLE "POLZRO"
C*
J=1
K=1
C* COMPUTE THE ORDER OF THE SYSTEM OF POLES
C*
NORDER=IROOTS(3)+2*IROOTS(4)
IF(NORDER.NE.0)GO TO 10
POLE(1)=1.0
RETURN
C* THE VECTOR "POLES" WILL CONTAIN A LIST OF ALL
C* THE POLES, IE., X+J0, X+JY, X-JY, . . .
C* POLES(J)=CMPLX(POLZRO(10+K,1),POLZRO(10+K,2))
C* TEST TO SEE IF ALL ROOTS HAVE BEEN
C* STORED IN THE "POLES" VECTOR
C*
IF(J.EQ.NORDER)GO TO 20
J=J+1
C* IF THE ROOT IS REAL CONTINUE PROCESSING THE NEXT ROOT
C*
IF(POLZRO(K+10,5).NE.3)GO TO 15
K=K+1
GO TO 10
C* SINCE THE ROOT IS COMPLEX PUT THE CONJUGATE
C* OF THE ROOT IN THE "POLES" TABLE.
C*
POLES(J)=CMPLX(POLZRO(10+K,1),-POLZRO(10+K,2))
IF(J.EQ.NORDER) GO TO 20
J=J+1
K=K+1
GO TO 10

C* GENERATE THE COEFFICIENTS FOR THE D(z) POLYNOMIAL
C* OF THE TRANSFER FUNCTION H(z).
C* 20 CALL COEFF(POLES,NORDER,ACOEF)
C* CHANGE THE COEFFICIENTS FROM COMPLEX ASUBK TO REAL ASUBK
C* DO 40 J=1,11
   40 POLE(J)=REAL(ACOEF(J))
RETURN
END

SUBROUTINE BSJBR(ZERO,NORDER)
C* GENERATE THE BIR COEFFICIENTS FROM THE ZEROS
C* COMMON IROOTS(4),POLZRO(20,5),IERROR
   COMPLEX ZEROS(11), BCOEF(11)
   REAL ZERO(11)
C* INITIALIZE ALL VECTORS AND INDICES
C* DO 5 K=1,11
   5 ZEROS(K)=(0.,0.)
   ZERO(K)=0.
   BCOEF(K)=(0.,0.)
C* "J" INDEXES THE "ZEROS" TABLE
C* "K" INDEXES THE "POLZRO" TABLE
C* J=1
   K=1
C* COMPUTE THE ORDER OF THE SYSTEM OF ZEROS.
C* NORDER=IROOTS(1)+2*IROOTS(2)
   IF(NORDER.NE.0)GO TO 10
   ZERO(1)=1.0
RETURN
C* THE VECTOR "ZEROS" CONTAINS A LIST OF ALL
C* OF THE ZEROS, IE., X+J0, X+JY, X-JY, . . .
10  ZEROS(J)=CMPLX(POLZRO(K,1),POLZ.O(K,2))
C*  CHECK TO SEE IF ALL THE ROOTS HAVE BEEN PROCESSED
C*  IF(J.EQ.NORDER)GO TO 20
     J=J+1
C*  IF THE ROOT IS REAL CONTINUE PROCESSING THE NEXT ROOT
C*  IF(POLZRO(K,5).NE.1)GO TO 15
     K=K+1
     GO TO 10
C*  SINCE THE ROOT IS COMPLEX, ENTER THE CONJUGATE
C*  INTO THE TABLE "ZEROS".
C*  ZEROS(J)=CMPLX(POLZRO(K,1),-POLZRO(K,2))
C*  IF(J.EQ.NORDER)GO TO 20
     J=J+1
     K=K+1
     GO TO 10
C*  GENERATE THE BSUBR COEFFICIENTS
C*  CALL COEFF(ZEROS, NORDER, BCOEF)
C*  MAKE THE COEFFICIENTS BSUBR REAL.
C*  DO 40 J=1,11
     ZERO(J)=REAL(BCOEF(J))
     RETURN
   40  END

SUBROUTINE COEFF(V,N,H1)
C*  "V" - VECTOR OF ROOTS; "N" - THE ORDER OF THE SYSTEM
C*  "H1" - VECTOR WITH THE FINAL COEFFICIENTS
C*  H(1) = Z**(0), H(2) = Z**(-1), ..., H(11) = Z**(-10)
C*  COMPLEX H(11), H2(11), H3(11), V(11)
C*  INITIALIZE VECTORS
     IZERO=0
     DO 10 I=1,11
        H1(I)=(0.,0.)
        H2(I)=(0.,0.)
        H3(I)=(0.,0.)
        V(I)=(1.,0.)
   10  IZERO=I
   20  RETURN
END
C* THE ROOTS COME FROM THE FORMER SUBROUTINE
C* THESE ARE ROOTS OF THE POLYNOMIALS N(Z), AND D(Z)
C* WHERE H(Z)=N(Z)/D(Z)
C* THE FACTORS OF THE POLYNOMIAL ARE OF THE FORM Z-ROOT
C*
DO 15 K=1,11
   15 V(K)=-V(K)
C* WHEN K=N, WE HAVE GENERATED THE H1 COEFFICIENTS
C* OF N(Z), OR D(Z) OF H(Z)=N(Z)/D(Z)
C* THE REMAINING ITERATIONS ARE PERFORMED TO SIMPLIFY
C* OBTAINING COEFFICIENTS FOR N(Z**-1), AND D(Z**-1) LATER
C*
DO 30 K=1,10
C* "H2" REPRESENTS THE PREVIOUS RESULT TIMES Z**(-1)
C*
   40 H2(I+1)=H1(I)
   50 H2(I)=0.*0.*
   60 H2(I)=H1(I)*V(I)
C* GENERATE THE NEW FINAL PRODUCT WITH K ROOTS MULTIPLIED.
C*
   70 H1(I)=H2(I)+H3(I)
C* GET THE NEXT ROOT TO BE MULTIPLIED AND PUT IT INTO "V(I)"
C*
   80 V(I)=V(I+1)
   90 CONTINUE
C* TO GET THE COEFFICIENTS FOR H(Z**-1)=N(Z**-1)/D(Z**-1)
C* THE ORDER OF THE COEFFICIENTS MUST BE REVERSED.
C*
DO 100 I=1,11
   100 H2(I)=H1(I2-1)
   100 H1(I)=H2(I)
99 RETURN
END

SUBROUTINE ADJUST(NNUM,NDEN)
COMMON /ABVCTR/A(11),B(11)
IDELTA=NDEN-NNUM
ISTOP=11-IDELTA

ASDF6241
ASDF6242
ASDF6243
ASDF6244
ASDF6245
ASDF6246
ASDF6247
ASDF6248
ASDF6249
ASDF6250
ASDF6251
ASDF6252
ASDF6253
ASDF6254
ASDF6255
ASDF6256
ASDF6257
ASDF6258
ASDF6259
ASDF6260
ASDF6261
ASDF6262
ASDF6263
ASDF6264
ASDF6265
ASDF6266
ASDF6267
ASDF6268
ASDF6269
ASDF6270
ASDF6271
ASDF6272
ASDF6273
ASDF6274
ASDF6275
ASDF6276
ASDF6277
ASDF6278
ASDF6279
ASDF6280
ASDF6281
ASDF6282
ASDF6283
ASDF6284
ASDF6285
ASDF6286
ASDF6287
ASDF6288
DO 10 I=1, STOP
B(I+1)=B(I)+I+1-DELTA)
DO 20 K=1, I+DELTA
20 B(K)=0.
RETURN
END

SUBROUTINE PLTIMP(ICHAR)
C* THIS SUBROUTINE PLOTS THE UNIT SAMPLE RESPONSE
C*
DIMENSION PICT(1025)
COMMON /IMPULS/ YI(1025), YS(1025), EN(1025)
DO 5 K=1, 1024
5 EN(K+1)=K-1
EN(1)=YI(1025)
NFIN=YI(1025)
DO 13 K=1, NFIN
PICT(K+1)=YI(K)
CONTINUE
PICT(1)=YI(1025)
CALL INIT
CALL XFRM(2)
CALL YFRM(2)
IF(NFIN.LT.70) CALL LINE(-1)
IF(NFIN.GE.70) CALL LINE(0)
IF(NFIN.LT.70) CALL SYMBL(1)
IF(NFIN.GE.70) CALL SYMBL(0)
CALL SIZE(, 4)
CALL PLOT(EN, PICT)
JFIN=YI(1025)+1
IF(NFIN.GE.70) GO TO 40
DO 30 K=2, JFIN
CALL MOVEA(EN(K), PICT(K))
30 CALL DRAWA(EN(K), 0.0)
CONTINUE
DO 40 K=1, JFIN
CALL MOVEA(0.0, 0.0)
40 CALL DRAWA(EN(JFIN), 0.0)
RETURN
END

SUBROUTINE PLSSTP(ICHAR)
C* THIS SUBROUTINE PLOTS THE UNIT STEP RESPONSE
C*
DIMENSION PICT(1025)
COMMON /IMPSUS/ YI(1025), YS(1025), EN(1025)
KLAST=-500.
DO 5 K=1,1024
5 CONTINUE
EN(K+1)=K-1
EN(1)=YS(1025)
NF(N)=YS(1025)
DO 13 K=1,NF(N)
PICT(K+1)=YS(K)
13 CONTINUE
PICT(1)=NF(N)
CALL INIT
CALL XFRM(2)
CALL YFRM(2)
IF(NF(N).LE.70)GO TO 40
IF(NF(N).GE.70) GO TO 190
IF(NF(N).LE.70)GOSUB 10
IF(NF(N).GE.70) GOSUB 110
CALL PLOT(EN(N), PICT)
JFIN=NFIN+1
IF(NF(N).GE.70)GO TO 40
DO 30 K=2,JFIN
CALL MOVEA(EN(K), PICT(K))
CALL DRAWA(EN(K), 0.)
30 CONTINUE
40 CALL MOVEA(0.0, 0.0)
CALL DRAWA(EN(JFIN), 0.0)
CALL LABEL
CALL VCURS((ICHAR, X, Y)
CALL FIN
RETURN
END

SUBROUTINE PLITRF(IFNCTN, ISCALE, ICHAR)
C* THIS SUBROUTINE PLOTS BOTH THE PHASE AND MAGNITUDE OF THE TRANSFER FUNCTION (LINEAR AND LOG)
C*
COMMON /FREQ/ YP(1026), YM(1026), PI(1026)
DO 10 K=1,1024
YP(K+2)=FLOAT(K-1)*3.14/1024.
10 PI(1)=1.025.
CALL INIT
CALL XFRM(2)
CALL YFRM(2)
IF(IFNCTN.EQ.3)CALL PLOT(PI, YP)
IF(ISCALE.EQ.1)GO TO 20
15 YM1)=20.*ALOG10(YM(I))
YM(2)=.01
20 IF(IFNCTN.EQ.4)CALL PLOT(P1,YM)
CALL LABLE
CALL TSEND
CALL VCURSR(ICHAR,X,Y)
CALL FIN
RETURN
END

SUBROUTINE RSPNSE (IFNCTN,FLTRGN)

C* THIS SUBROUTINE COMPUTES THE UNIT SAMPLE RESPONSE OF THE SYSTEM
C* THE "Z" VECTOR HOLDS THE DELAYED VALUES
C* FOR THE RECURSIVE CALCULATION.
C*
C* DIMENSION Z(I1)
C* REAL INPUT
C*
C* "AVBCTR" IS THE PAIR OF VECTORS HOLDING
C* THE VALUES OF A(K), AND B(K)
C*
C* COMMON /AVBCTR/ A(I1), B(I1)
C*
C* "IMPULS" IS THE LABELED COMMON HOLDING THE UNIT SAMPLE RESPONSE
C*
C* COMMON /IMPULS/ Y1(1025), YS(1025), EN(1025)
C*
C* INITIALIZE THE Z VECTOR
C*
C* XLAST=0.
C* XMIN=0.
C* XMAX=0.
C* DELTA=0.
C* DO 5 K=1,11
C* Z(K)=0.
C* DO 30 N=1,1024
C* COMPUTE THE 1024 POINT UNIT SAMPLE OR STEP RESPONSE
C*
C* EACH VALUE OF XSUBN WILL BE ONE EXCEPT FOR THE CASE
C* OF THE UNIT SAMPLE RESPONSE WHEN N>1, AND THEN XSUBN=0.
C*
C* XSUBN = 1.
IF(N.GT.1.AND.IFNC.1.EQ.0)ISUBN=0.
INPUT=SUBN
DO 10 K=2,11
   INPUT=INPUT-A(K)*Z(K)
  Z(I)=INPUT
   OUTPUT=0.
   DO 20 K=1,11
  20 OUTPUT=OUTPUT+B(K)*Z(K)
   DO 25 K=1,10
  25 Z(12-K)=Z(11-K)
  1EN=N
C* TEST TO SEE IF A STEADY STATE HAS BEEN ACCEIVED.
  27 IF(ABS(OUTPUT).GT.XMAX)XMAX=ABS(OUTPUT)
   REPORT=.01*XMAX
   IF(ABS(OUTPUT-XLAST).GT.DIFFER)DIFFER=DELTA=0.
   XLAST=OUTPUT
   DELTA=DELTA+1.
   IF(DIFFER.GT.25)GO TO 36
   IF(IFNCN.EQ.01)Y1(N)=OUTPUT*X1RNG
   IF(IFNCN.EQ.11)Y1(N)=OUTPUT*X1RNG
   IF(IFNCN.EQ.01)Y1(1025)=YEND-15
   IF(IFNCN.EQ.01)Y1(1025)=YEND-15
  999 RETURN
END
SUBROUTINE FRNCY(NNUM,NDEN,FLTRGM)
C* FRNCY COMPUTES THE FREQUENCY RESPONSE FOR
C* THE FILTER BEING ANALYZED.
C* COMMON /AVBCTR/A(11),B(11)
C* COMMON /FREQ/VP(1026),YM(1026)
C* REAL*8 X,Y,THETA,ZAO,ONE,AA(11),BB(11),XX,G(2)
C* COMPLEX*8 Z,FRCTOP,FRGBOT,HOFZ,ZEXP,TERM,TERMD
C* EQUIVALENCE (G,HOFZ)
C* ZR=0.0D+00
C* MAKE THE COEFFICIENTS DOUBLE PRECISION
C* DO 20 K=1,11
   AA(K)=A(K)
   BB(K)=B(K)
  20 CONTINUE
C* COMPUTE PHASE AND MAGNITUDE OF H(Z)
C* DO 60 K=1,1025
THETA=FLOAT(K-1)*3.14159265/1024.
Z=CDEXP(DCMPLX(ZRO,THETA))
25 FRCTOP=0.
FRCBOT=0.
C* COMPUTE THE VALUE OF THE NUMERATOR OF H(Z)
C* DO 40 N=1,NUM
ZEXP=Z**(N-1)
TERM=DCMPLX(AA(N),ZRO)/ZEXP
FRCTOP=FRCTOP+TERM
40 CONTINUE
C* COMPUTE THE VALUE OF THE DENOMINATOR FACTORS OF H(Z)
C* DO 50 N=2,NDEN
TERM=DCMPLX(AA(N),ZRO)/Z**(N-1)
FRCBOT=FRCBOT+TERM
50 CONTINUE
C* GET H(Z) IN FINAL FORM
C* HOFZ=FRCTOP/DCMPLX(ONE,ZRO)+FRCBOT
C* COMPUTE THE MAGNITUDE OF H(Z)
C* YM(K+2)=CDABS(HOFZ)
C* COMPUTE THE PHASE OF H(Z)
C* X=DATAN2(Y,X)
10 IF(X.EQ.ZRO.AND.Y.EQ.ZRO)YP(K+2)=YP(K+1)
20 IF(X.EQ.ZRO.AND.Y.EQ.ZRO)GO TO 60
30 X=DATAN2(Y,X)
40 CONUTINUE
50 YP(1)=1025.
YP(2)=0.
YM(1)=1025.
YM(2)=0.
RETURN
END
SUBROUTINE FLTR2 (FLTRN)
C*  FLTR2 WRITES A LISTING OF ALL
C*  POLE AND ZERO LOCATIONS FOR REVIEW BEFORE
C*  PROCEEDING WITH THE PLOTTING OF RESPONSES
C*
COMMON IROOTS(4), POLZRO(20, 5)
DIMENSION IRTTP(5)
DATA L'/1HZ', 'IP'/1HP', 'IZP'/2HZP/
DATA IRTTP/2'H, 2'HZ, 2'HP, 2'HCP/
CALL INIT
CALL FIN
WRITE (6, 102) (I, I=1, 70)
102 FORMAT (1X, 70A1)
WRITE (6, 100)
100 FORMAT (1X, 'Z', 5X, 'REAL', 6X, 'Z', 3X, 'IMAGINARY', 3X, 'Z'
1, '6X, 'RHC', 6X, 'Z', 5X, 'THEMA', 5X, 'Z RT Z')
WRITE (6, 102) (I, I=1, 70)
DO 10 I=1, 10
10 IYPEP = POLZRO(I, 5) + 1.
IF (POLZRO(I, 5), NE, 2, AND, POLZRO(I, 5), NE, 4,)
WRITE (6, 104) (POLZRO(I, J), J=1, 4), IRTTP(IYPEP)
104 FORMAT (1X, 4X, 4X, 20X, 'Z ', F10.7, Z '+/- ', F10.7, Z ', F10.7,
111 FORMAT(20X, 'NUMBER OF COMPLEX ZERO PAIRS:', I3) WRITE(6, 113) IROOTS(3)
113 FORMAT(20X, 'NUMBER OF REAL POLES:', I3) WRITE(6, 114) IROOTS(4)
114 FORMAT(20X, 'NUMBER OF COMPLEX POLE PAIRS:', I3) WRITE(6, 117) FLTRGN
117 FORMAT(25X, 'FILTER GAIN:', F12.8) WRITE(6, 115) READ(5, 116)
115 FORMAT(' HSP SPACE THEN RETURN') READ(5, 116)
116 FORMAT(A1) RETURN

SUBROUTINE LABEL
C* SUBROUTINE LABEL WRITES THE LABELS ON EACH C* OF THE FIVE PLOTS DRAWN
C*
DIMENSION ITITLE(40)
CALL ANMODE
READ(2, 100) IX, IY, NUM, (ITITLE(J), J=1, 40)
CALL RECOVR
CALL MOVABS(IX, IY)
CALL ANMODE
WRITE(6, 101) (ITITLE(J), J=1, NUM)
101 FORMAT(20X, ITITLE(J))
READ(2, 100) IX, IY, NUM, (ITITLE(J), J=1, 40)
DO 10 I=1, NUM
CALL RECOVR
CALL MOVABS(IX, IY)
CALL ANMODE
WRITE(6, 102) ITITLE(I)
10 FORMAT(I1, A1)
10 IY=IY+22
READ(2, 100) IX, IY, NUM, (ITITLE(J), J=1, 40)
CALL RECOVR
CALL MOVABS(IX, IY)
CALL ANMODE
WRITE(6, 101) (ITITLE(J), J=1, NUM)
READ(2, 100) IX, IY, NUM, (ITITLE(J), J=1, 40)
CALL RECOVR
CALL MOVABS(IX, IY)
CALL ANMODE
WRITE(6, 101) (ITITLE(J), J=1, NUM)
CALL RECOVR
CALL TSEND
RETURN
END
C*** SOURCE DECK FOR ASDF COMMAND: HRD$CPY
C*
C*** SOURCE DECK FOR ASDF COMMAND: HRD$CPY
C*
C COMMON IROOTS(4), POLZRO(20, 5), IERROR
C COMMON /ABVCTR/ A(11), B(11)
C COMMON /IMPULS/ Y1(1026), YS(1026), EN(1026)
C COMMON /FREQ/ YP(1026), YM(1026), PI(1026), YMDB(1026)
C CALL ERRSET(208, 256, -1, 1, 1)
C LINEAR=1
C LOG=2
C*
C* INITIALIZE A AND B TO ZERO
C*
DO 5 K=1, 20
A(K)=0.
5 B(K)=0.
C*
C* READ THE POLZRO TABLE OFF OF THE FILE
C*
DO 20 I=1, 20
20 READ(5, 100) (POLZRO(I, J), J=1, 5)
100 FORMAT(1X, 5F12.8)
C*
C* READ THE IROOTS VALUES, AND FILTER GAIN FROM THE FILE
C*
READ(5, 101) (IROOTS(I), I=1, 4)
101 FORMAT(1X, 4I2)
READ(5, 102) FLTRGN
102 FORMAT(1X, F12.8)
FLTRGN=ABS(FLTRGN)
C*
C* PRINT OUT A TABLE OF THE POLES AND ZEROS
C*
CALL FLTRZ(FLTRGN)
C*
C* GENERATE THE A(K) COEFFICIENTS
C*
CALL ASUBK(A, NORDER)
NDEN=NORDER+1
C*
C* GENERATE THE B(R) COEFFICIENTS
C*
CALL BSUBK(B, NORDER)
NNUM=NORDER+1
C*
C* CHECK TO SEE IF THE FILTER IS CAUSAL
C* IF(INUM GT NDEN)WRITE(6,103)
   103 FORMAT(10(I1,1,Es FILTER IS NOT CAUSAL - PROGRAM TERMINATES!))
C* IF(INUM GT NDEN)STOP
C* ADJUST THE COEFFICIENTS IF NNUM DOES NOT EQUAL NDEN
C* IF(INUM NE NDEN)CALL ADJUST(INUM,NDEN)
C* COMPUTE THE UNIT SAMPLE RESPONSE
C* CALL RSPNSE(O,FLTRGN)
C* CALCULATE THE STEP RESPONSE
C* CALL RSPNSE(1,FLTRGN)
C* CALCULATE THE FREQUENCY RESPONSE
C* CALL FRQNCY(INUM,NDEN,FLTRGN)
    CALL PRPPLT(IYIPTS,IYSPTS,NUMPLT)
C* PLOT THE UNIT SAMPLE RESPONSE
C* CALL PLTIMP(IYIPTS,FLTRGN,NUMPLT)
C* PLOT THE STEP RESPONSE
C* CALL PLTSTP(IYSPTS,FLTRGN,NUMPLT)
C* PLOT THE PHASE OF THE TRANSFER FUNCTION
C* CALL PLTTRF(3,LINEAR,FLTRGN,NUMPLT)
C* PLOT THE MAGNITUDE OF THE TRANSFER FUNCTION
C* CALL PLTTRF(4,LINEAR,FLTRGN,NUMPLT)
    CALL PLTTRF(4,LOG,FLTRGN,NUMPLT)
C* PLOT THE UNIT CIRCLE WITH POLES AND ZEROS
C* CALL UNTCRU(NUMPLT)
C* OUTPUT THE DATA
   CALL OUTNUM
   STOP
   END
SUBROUTINE ADJUST(NNUM, NDEN)
COMM COV /ABVCTR/A(II), B(II)
I DELTA=NDEN-NNUM
ISTOP=1-I-IDELTA
DO 10 I=1,ISTOP
10 B(I2-I)=B(I2-I-IDELTA)
DO 20 K=1,IDELTA
20 B(K)=0,
NNUM=NDEN
RETURN
END

SUBROUTINE OUTNUM
C*
C* OUTNUM PRINTS OUT THE NUMERICAL VALUES
C* OF THE FILTER'S RESPONSES
C*
COMM COV /IMPULS/ YI(1026), YS(1026), E N(1026)
COMM COV /FREQ/ VP(1026), YM(1026), PI(1026), YMDB(1026)
DIMENSION M(1026)
DO 20 L=1,1026
20 M(L)=L-1
DO 40 I=1,18
CALL HEADING
DO 40 J=1,6
IBEG=((I-1)*60+(J-1)*10+1
IEND=IBEG+9
IF(I.EQ.18,AND.J.EQ.1)IEND=1024
WRITE(6,104)(M(K),YI(K),YS(K),VP(K),YM(K),PI(K),K=IBEG,IEND)
104 FORMAT(102X,16,2E20.8,3E21.8,/)!
IF(I.EQ.18,AND.J.EQ.1)GO TO 41
40 CONTINUE
41 WRITE(6,105)
105 FORMAT(1H1)
RETURN
END

SUBROUTINE HEADING
C*
C* HEADING PRINTS THE PAGE HEADINGS
C*
DATA IX/1HX/
WRITE(6,100)
100 FORMAT(1H1)
WRITE(6,101)(IX,1=1,114)
101 FORMAT(IX,114A1)
WRITE(6,1081)
108 FORMAT('X',112X,'X')
WRITE(6,103)  
103 FORMAT('X',N*5X,'UNIT SAMPLE RESPONSE',3X,'STEP RESPONSE',3X
X'PHASE OF H(Z) (RADIANS)',2X,'MAGNITUDE OF H(Z)',4X,
X'OMEGA (RADIANS) X')
WRITE(6,108)
WRITE(6,101)(IX,IX=1,114)
WRITE(6,102)
102 FORMAT(/)  
RETURN
END

SUBROUTINE UNTCR(NUMPLT)
C* UNTCR PLOTS THE UNIT CIRCLE AND
C* ALL POLES AND ZEROS
C*
DIMENSION UCX(362),UCY(362)
COMMON IROOTS(14),PZROOT(20,5),IERROR
DO 10 I=1,361
THETA=FLOAT(1-I)*2*3.14159/360.
UCX(I)=2.5*COS(THETA)
UCY(I)=2.5*SIN(THETA)
CONTINUE
CALL BOX(0,FLTRGN,NUMPLT)
CALL PLOT(-7.117,-5381,-3)
CALL PLOT(-2.75,4.25,+3)
CALL PLOT(0.25,4.25,+2)
CALL PLOT(-2.25,4.25,+3)
CALL PLOT(-2.5,7.0,+2)
IUP=+3
IDOWN=-2
HDLTA=.25
VOLTA=.2
X=2.75
DO 70 I=1,123
IF(1.NE.7.AND.1.NE.17.AND.1.NE.2.AND.1.NE.22)GO TO 71
Y=4.45
CALL PLOT(X,Y,IUP)
Y=4.05
CALL PLOT(X,Y,IDOWN)
GO TO 70
71  Y=4.35
CALL PLOT(X,Y,IUP)
Y=Y-VOLTA
CALL PLOT(X,Y,IDOWN)
70  X=X+HDLT A
VOLTA=.25
C* GENERATE THE AK COEFFICIENTS FROM THE POLES
C* COMMON IROOTS(4), POLZR0(20+5), IERROR
C* REAL POLE[11]
C* INITIALIZE ALL THE VECTORS AND INDICES TO ZERO
C* DO 5 K=1,11
      POLES(K)=(0.,0.)
      POLE(K)=0.
      ACOEF(K)=0.
5 S
C* "J" INDEXES THE TABLE "POLES"
C* "K" INDEXES THE TABLE "POLZR0"
C* J=1
      K=1
C* COMPUTE THE ORDER OF THE SYSTEM OF POLES
C* NORDER=IROOTS(3)+2*IROOTS(4)
      IF(NORDER.NE.0)GO TO 10
      POLE(1)=1.0
      RETURN
C* THE VECTOR "POLES" WILL CONTAIN A LIST OF ALL
C* THE POLES, IE., X+JO, X+JY, X-JY, . . .
C* 10 POLES(J)=CMPLX(POLZR0(10+K,1),POLZR0(10+K,2))
C* TEST TO SEE IF ALL ROOTS HAVE BEEN
      IF(J.EQ.NORDER)GO TO 20
      J=J+1
C* C* IF THE ROOT IS REAL CONTINUE PROCESSING THE NEXT ROOT
C* IF(POLZR0(K+10,5).NE.3)GO TO 15
      K=K+1
      GO TO 10
C* SINCE THE ROOT IS COMPLEX PUT THE CONJUGATE
C* OF THE ROOT IN THE "POLES" TABLE.
C* 15 POLES(J)=CMPLX(POLZR0(10+K,1),-POLZR0(10+K,2))
      IF(J.EQ.NORDER) GO TO 20
J=J+1
K=K+1
GO TO 10

C* GENERATE THE COEFFICIENTS FOR THE D(z) POLYNOMIAL
C* OF THE TRANSFER FUNCTION H(z).
C* CALL COEFF(POLES,NORDER,ACOEFL
C* CHANGE THE COEFFICIENTS FROM COMPLEX ASUBK TO REAL ASUBK
C* DO 40 J=1,11
  40 POLE(J)=REAL(ACOEFL(J))
  RETURN
END

SUBROUTINE BSUBR(ZERO,NORDER)
C* GENERATE THE B(R) COEFFICIENTS FROM THE ZEROS
C* COMMON IROOTS(4),POLZRD(20,5),IERRO
C* COMPLEX ZEROS(11),BCOEFL1
C* REAL ZERO(11)
C* INITIALIZE ALL VECTORS AND INDICES
C* DO 5 K=1,11
  5 ZEROS(K)=(0.,0.)
  ZERO(K)=0.
  BCOEF(K)=(0.,0.)
C* "J" INDEXES THE "ZEROS" TABLE
C* "K" INDEXES THE "POLZRO" TABLE
C* J=1
C* K=1
C* COMPUTE THE ORDER OF THE SYSTEM OF ZEROS.
C* NORDER=IROOTS(1)+2*IROOTS(2)
  IF(NORDER.NE.0)GO TO 10
  ZERO(I1)=1.0
  RETURN
C* THE VECTOR "ZEROS" CONTAINS A LIST OF ALL
C* OF THE ZEROS, IE., X+J0, X+JY, X-JY, ...
C* 10 ZEROS(J)=CMPLX(POLZRO(K,1),POLZRO(K,2))
C* CHECK TO SEE IF ALL THE ROOTS HAVE BEEN PROCESSED
C* IF(J.EQ.NORDER)GO TO 20
  J=J+1
C* IF THE ROOT IS REAL CONTINUE PROCESSING THE NEXT ROOT
C* IF(POLZRO(K,5).NE.1)GO TO 15
  K=K+1
  GO TO 10
C* SINCE THE ROOT IS COMPLEX, ENTER THE CONJUGATE
C* INTO THE TABLE *ZEROS*.
  ZEROS(J)=CMPLX(POLZRO(K,1),-POLZRO(K,2))
  IF(J.EQ.NORDER)GO TO 20
  J=J+1
  K=K+1
  GO TO 10
C* GENERATE THE BSUBR COEFFICIENTS
C* 20 CALL COEFF(ZEROS, NORDER, BCOEF)
C* MAKE THE COEFFICIENTS BSUBR REAL.
C* DO 40 J=1,11
  ZERO(J)=REAL(BCOEF(J))
  RETURN
40 END

SUBROUTINE COEFF(V,N,H1)
C* "V" - VECTOR OF ROOTS; "N" - THE ORDER OF THE SYSTEM
C* "H1" - VECTOR WITH THE FINAL COEFFICIENTS
C* H (1)=Z**0, H(2)=Z**(-1), . . . ,H(11)=Z**(-10)
C* COMPLEX H1(11),H2(11),H3(11),V(11)
C* INITIALIZE VECTORS
  IZERO=0
  DO 10 I=1,11
    HI(I)=(0.,0.)
    H2(I)=(0.,0.)
  10 H3(I)=(0.,0.)
  HI(I)=(1.,0.)
C* THE ROOTS COME FROM THE FORMER SUBROUTINE
C* THESE ARE ROOTS OF THE POLYNOMIALS N(Z), AND D(Z)
C* WHERE H(Z) = N(Z)/D(Z).
C* THE FACTORS OF THE POLYNOMIAL ARE OF THE FORM Z-ROOT
C* DO 15 K=1,11
   V(K)=-V(K)
C* WHEN K=N, WE HAVE GENERATED THE H1 COEFFICIENTS
C* OF N(Z), OR D(Z) OF H(Z)=N(Z)/D(Z)
C* THE REMAINING ITERATIONS ARE PERFORMED TO SIMPLIFY
C* OBTAINING COEFFICIENTS FOR N(Z**-1), AND D(Z**-1) LATER
C* DO 30 K=1,10
C* "H2" REPRESENTS THE PREVIOUS RESULT TIMES Z**(1)
C* DO 40 I=1,10
   H2(I+1)=H1(I)
   H2(I)=0.,0.,
   DO 50 I=1,11
   H3(I)=H1(I)*V(I)
C* GENERATE THE NEW FINAL PRODUCT WITH K ROOTS MULTIPLIED.
C* DO 60 I=1,11
   H1(I)=H2(I)+H3(I)
C* GET THE NEXT ROOT TO BE MULTIPLIED AND PUT IT INTO "V(I)"
C* DO 70 I=1,10
   V(I)=V(I+1)
   30 CONTINUE
C* TO GET THE COEFFICIENTS FOR H(Z**-1)=N(Z**-1)/D(Z**-1)
C* THE ORDER OF THE COEFFICIENTS MUST BE REVERSED.
C* DO 80 I=1,11
   H2(I)=H1(I-1)
   DO 90 I=1,11
   H1(I)=H2(I)
   95 RETURN
END

SUBROUTINE RSPNSE (IFNCTN,FLTRGN)
C* THIS SUBROUTINE COMPUTES THE UNIT SAMPLE RESPONSE OF THE SYSTEM
C* THE "Z" VECTOR HOLDS THE DELAYED VALUES
C* FOR THE RECURSIVE CALCULATION.
C* REAL*8 XMIN, XMAX, XLAST, XSUBN, INPUT, OUTPUT, AA(11), BB(11),
  XZ(11), DIFFER, DFTRGN, ZRO, ONE, PTOONE
C* "ABVCTR" IS THE PAIR OF VECTORS HOLDING
C* THE VALUES OF A(K), AND B(K)
C* COMMON /ABVCTR/ A(11), B(11)
C* "IMPSL" IS THE Labeled COMMON HOLDING THE UNIT SAMPLE RESPONSE
C* COMMON /IMPSL/ YI(1026), YS(1026), EN(1026)
C* INITIALIZE THE CONSTANTS AND Z VECTOR
C*
C* IFLAG=0
ZRO=0.0D+00
ONE=1.0D+00
PTOONE=1.0D-02
XMIN=ZRO
XMAX=ZRO
XLAST=ZRO
DELTA=0.
DFTRGN=FLTRGN
DO 4 K=1,11
AA(K)=A(K)
4 BB(K)=B(K)
DO 5 K=1,11
Z(K)=0.
C* COMPUTE THE 1024 POINT UNIT SAMPLE OR STEP RESPONSE
C* DO 30 N=1,1024
C* EACH VALUE OF XSUBN WILL BE ONE EXCEPT FOR THE CASE
C* OF THE UNIT SAMPLE RESPONSE WHEN N>1, AND THEN XSUBN=0.
C*
XSUBN=ONE
IF(N.GT.1.AND.INCN.T.EQ.0)XSUBN=ZRO
INPUT=XSUBN
DO 10 K=2,11
10 INPUT=INPUT-A(K)*Z(K)
Z(11)=INPUT
OUTPUT=ZRO
DO 20 K=1,11
OUTPUT=OUTPUT+B(K)*Z(K)
20 DO 25 K=1,10
25   Z(12-K)=Z(11-K)
   IEND=N
C* TEST TO SEE IF A STEADY STATE HAS BEEN ACHIEVED.
C*
  IF(DABS(OUTPUT).GT.XMAX)XMAX=DABS(OUTPUT)
  DIFFER=P'TOONE*XMAX
  IF(DABS(OUTPUT-XLAST).GT.DIFFER)DELTAS=0.
  XLAST=OUTPUT
  DELTA=DELTA+1
  IF(IIFNCN.EQ.0)YI(N)=OUTPUT*DFRGN
  IF(IIFNCN.EQ.1)YS(N)=OUTPUT*DFRGN
  IF(DELTA.LE.20.)GO TO 30
  IFLAG=1
  DELTA=-2000.
38   CONTINUE
30   IF(IFLAG.EQ.0.AND.IFNCN.EQ.0)YI(1025)=IEND-15
   IF(IIFNCN.EQ.1)YS(1025)=IEND-15
999   RETURN
END

SUBROUTINE FRNCGY(NNUM,NDEN,FLTRGN)
COMMON /ABVCTR/A(11),B(11)
COMMON /FREQ/YP(1026),YM(1026),PI(1026),YMDB(1026)
REAL*8 X,Y,THETA,ZRO,ONE,AA(11),BB(11),XX,G(2)
COMPLEX*16 Z,FRCTOP,FRCBOT,HOFZ,ZEXP,TERM,TERMD
EQUIVALENCE (G,HOFZ)
ZRO=0.0D+00
ONE=1.0D+00

C* MAKE THE COEFFICIENTS DOUBLE PRECISION
C*
DO 456 KKK=1,11
   AA(KKK)=A(KKK)
   BB(KKK)=B(KKK)
456   CONTINUE
DO 60 K=1,1025
   THETA=FLOAT(K-11)*3.14159265/1024.
   Z=CEXP(DCMPLX(ZRO,THETA))
25   FRCTOP=0.
   FRCB0T=0.
   DO 40 N=1,NNUM
      ZEXP=Z**N-1
      TERM=DCMPLX(BB(N),ZRO)/ZEXP
      FRCTOP=FRCTOP+TERM
40   CONTINUE
DO 50 N=2,NOEN
TERM=DCMPLX(AA(N),ZRO)/Z**(N-1)
FRCCBOT=FRCCBOT+TERM
50 CONTINUE

CALCULATE H(Z)

HOFZ=FRCTOP/(DCMPLX(TONE,ZRO)+FRCCBOT)
VM(K)=CDABS(HOFZ)
55 VM(K)=VM(K)*FLTRGN
K=G11
Y=G(2)
IF(X.EQ.ZRO.AND.Y.EQ.ZRO)YP(K+1)=YP(K)
IF(X.EQ.ZRO.AND.Y.EQ.ZRO)GO TO 60
XX=DATANZ(Y,X)
YP(K)=XX
60 CONTINUE
RETURN
END

SUBROUTINE FLTRGN
COMMON IROOTS(4),POLZR0(20,5)
DIMENSION IRTP(5)
DATA 12,1HZ,1P,1HP,IZP/2HP/
DATA IRTP/2H,2HRZ,2HCZ,2HRP,2HCP/
WRITE(6,100)
100 FORMAT(1H1)
WRITE(6,101)
101 FORMAT(1D1)
WRITE(6,102)(12,1=1,70)
102 FORMAT(13X,70A1)
WRITE(6,103)
103 FORMAT(31X,'Z',5X,'REAL',6X,'Z',3X,'IMAGINARY',3X,'Z',
1,6X,'RHO',6X,'Z',5X,'THETA',5X,'Z RT Z')
WRITE(6,104)(12,1=1,70)
104 FORMAT(6,102)
DI 0 10=1,10
FORMAT=POZR0(I,5)+1
IF(POLZR0(I,5),NE.2,AND.POLZR0(I,5),NE.+1)
WRITE(6,105)(POLZR0(J,1),J=1,4),IRTP(ITYPEP)
IF(POLZR0(I,5),EQ.-2.,OR.POLZR0(I,5),EQ.+1)
WRITE(6,106)(POLZR0(J,1),J=1,4),IRTP(ITYPEP)
106 FORMAT(6,103)
103 FORMAT(30X,'Z',F10.7,' Z +/-I',F10.7,' Z',F10.7,  
13X,'Z +/-I',F10.7,' Z',F10.7,' Z',F10.7,  
13X,'Z +/-I',F10.7,' Z',F10.7,' Z')
104 FORMAT(30X,'Z',F10.7,' Z',F10.7,  
13X,'Z +/-I',F10.7,' Z',F10.7,' Z',F10.7,  
13X,'Z +/-I',F10.7,' Z',F10.7,' Z')
10 CONTINUE
WRITE(6,112)(IZP,1=1,35)
112 FORMAT(31X,35A2)
DO 20 I=11,120
   TYPEP=POLZRO(I,5)+1.
   IF (POLZRO(I,5),NE.2.,AND.POLZRO(I,5),NE.4.)
      WRITE(6,106) (POLZRO(I,J), J=1,4), INTP(TYPEP)
      IF (POLZRO(I,5),EQ.2.,OR.POLZRO(I,5),EQ.4.)
         WRITE(6,105) (POLZRO(I,J), J=1,4), INTP(TYPEP)
 105 FORMAT (30X*,P ,F10.7,*,P +/ ,F10.7,*,P ,F10.7,*,P ,F10.7,*,P ,F10.7,*,P )
 106 FORMAT (30X*,P ,F10.7,*,P +/ ,F10.7,*,P ,F10.7,*,P ,F10.7,*,P )
 20 CONTINUE
   WRITE(6,107) (IP, I =1,70)
   WRITE (6,107)
   107 FORMAT (/)
   WRITE(6,110) IROOTS(1)
   WRITE(6,111) IROOTS(2)
   WRITE(6,113) IROOTS(3)
   WRITE(6,114) IROOTS(4)
   WRITE(6,115) FTRGN
   113 FORMAT (50X, 'NUMBER OF REAL POLES:*',I3)
   114 FORMAT (50X, 'NUMBER OF COMPLEX POLE PAIRS:*',I3)
   115 FORMAT (50X, 'FILTER GAIN:*',F12.8)
   WRITE (6,100)
   RETURN
SUBROUTINE PRPPLT(IYIPTS, IYSPTS, NUMPLT)
C* THIS SUBROUTINE SCALES THE ORGDATE
C* DATA BEFORE PLOTTING
C*
COMMON /IMPSL/Y1(1026), YS(1026), EN(1026),
COMMON /FREQ/YP(1026), YN(1026), PI(1026), YM8(1026)
NPLT=1
IYPTS=Y1(1025)
IYSPTS=YS(1025)
CALL WINDOW(0.45,0.,21.11)
CALL PLTS(0.,0.,0.,)
ISUB=1
IF (ABS(Y1(1)) .LE. ABS(Y1(2))) ISUB=1
TEMP=Y1(ISUB)
Y1(ISUB)=0.
CALL SCALE(Y1,5.,1024.,+1)
Y1(ISUB)=TEMP
ISUB=2
IF (ABS(YS(1)) .LE. ABS(YS(2))) ISUB=1
ASDF7578
ASDF7579
ASDF7580
ASDF7581
ASDF7582
ASDF7583
ASDF7584
ASDF7585
ASDF7586
ASDF7587
ASDF7588
ASDF7589
ASDF7590
ASDF7591
ASDF7592
ASDF7593
ASDF7594
ASDF7595
ASDF7596
ASDF7597
ASDF7598
ASDF7599
ASDF7600
ASDF7601
ASDF7602
ASDF7603
ASDF7604
ASDF7605
ASDF7606
ASDF7607
ASDF7608
ASDF7609
ASDF7610
ASDF7611
ASDF7612
ASDF7613
ASDF7614
ASDF7615
ASDF7616
ASDF7617
ASDF7618
ASDF7619
ASDF7620
ASDF7621
ASDF7622
ASDF7623
ASDF7624
ASDF7625
TEMP=YS(ISUB)
YS(ISUB)=0.
CALL SCALE(YS,5.,1024,+1)
YS(ISUB)=TEMP
TEMP=YM(I)
YM(I)=0.
CALL SCALE(YM,5.,1024,+1)
YM(I)=TEMP
DO 10 I=1,1024
IF(YM(I).EQ.0.)YM(I)=1E-70
10 YMDB(I)=20.*ALOG10(YM(I))
ISUB=2
IF(ABS(YMDB(1)) .LE. ABS(YMDB(2))) ISUB=1
TEMP=YMDB(ISUB)
*C* THIS STATEMENT PREVENTS THE STAIRCASE FOR
*C* ALL PASS FILTERS
*C*
YMDB(ISUB)=-10.
CALL SCALE(YMDB,5.,1024,+1)
YMDB(ISUB)=TEMP
20 CALL SCALE(YP,5.,1024,+1)
RETURN
END
SUBROUTINE PLTNP(LAST,FLTRGN,NUMPNT)
*C* THIS SUBROUTINE PLOTS THE UNIT SAMPLE RESPONSE
*C*
COMMON /IMPULS/Y1(1025),YS(1026),EN(1026)
CALL BOX1(1,FLTRGN,NUMPNT)
DO 5 I=1,11024
5 EN(I)=1-1
CALL SCALE(EN,8.,LAST+1)
CALL AXIS(2.179,2.2012,1,NN,-1,8.,0.,EN(LAST+1),EN(LAST+2))
CALL AXIS(2.179,2.2012,20HUNIT SAMPLE RESPONSE,+20,5.,90.,Y1(1025))
CALL PLOT(2.179,2.2012,3)
YZERO=-Y1(1025)/Y1(1026)
ISTOP=8.+EN(LAST+2)+1.5
IF(ISTOP.GT.1024)ISTOP=1024
FVAL=EN(LAST+1)
DELTA=EN(LAST+2)
DO 7 I=ISTOP
7 EN(I)=1-1
DO 10 K=1,ISTOP
X=(EN(K)-FVAL)/DELTA
Y=(Y1(K)-Y1(1025))/Y1(1026)
IF(Y.GE.ZERO.AND.ISTOP.LE.80)CALL SYMBOL(X;Y;1,1;180.;-1)
IF(Y.LT.ZERO.AND.ISTOP.LE.80)CALL SYMBOL(X;Y;1,1;0.;-1)
CALL PLOT(X;Y;+3)
10 IF(YZERO.LT.0.)CALL PLOT(X;0.;+2)
IF(YZERO.GE.0.)CALL PLOT(X;YZERO;+2)
IF(YZERO.LT.0.)GO TO 999
CALL PLOT(0.;YZERO;+3)
CALL PLOT(X;YZERO;+2)
999 CALL PLOT(-2.179,-2.2012,-3)
NUMPLT=NUMPLT+1
RETURN
END

SUBROUTINE PLTSTP(LAST,FLTRGN,NUMPLT)
C* THIS SUBROUTINE PLTS THE UNIT STEP RESPONSE
C*
COMMON /IMPULS/YI(1026),YS(1026),EN(1026)
CALL BOX(1,FLTRGN,NUMPLT)
DO 5 I=1,1024
EN(I)=1.0
5 CALL SCALE(EN,8.,LAST+1)
CALL AXIS(2.179,2.2012,13,STEP RESPONSE,+13,5,,90.,YS(1025),
XY(1026))
CALL AXIS(2.179,2.2012,1HN,-1,8.,0.,EN(LAST+1),EN(LAST+2))
CALL PLOT(2.179,2.2012,-3)
YZERO=-(YS(1025)/YS(1026))
ISTOP=8.*EN(LAST+2)+1.5
IF(ISTOP.GT.1024)ISTOP=1024
FVAL=EN(LAST+1)
5 EN(LAST+2)
DO 7 I=1,ISTOP
EN(I)=1.0
7 EN(I)=1.0
DO 10 K=1,ISTOP
X=(EN(K)-FVAL)/DELT
Y=(YS(K)-YS(1025))/YS(1026)
IF(ISSTOP.LE.80)CALL SYMBOL(X,Y;1,1;180.,-1)
CALL PLOT(X;Y;+3)
10 IF(YZERO.GE.0.)CALL PLOT(X;YZERO;+2)
IF(YZERO.LT.0.)GO TO 999
CALL PLOT(0.;YZERO;+3)
CALL PLOT(X;YZERO;+2)
999 CALL PLOT(-2.179,-2.2012,-3)
NUMPLT=NUMPLT+1
RETURN
END
SUBROUTINE PLTTRF(IFNCTN,ITYPE,FLTRGN,NUMPLT)
C*  THIS SUBROUTINE PLOTS THE PHASE AND MAGNITUDE OF THE TRANSFER FUNCTION
C* COMMON /FREQ,YPI(1026),YMI(1026),PI(1026),YMDB(1026)
DO 10 K=1,1024
10 PI(K)=FLOAT(K-1)*3.14159/1024.
CALL BOX1(FLTRGN,NUMPLT)
CALL SCALE(PI,8.*1024.,+1)
CALL AXIS(2.179,2.2012,7HRAD I A N S , -7.8,.0.,PI(1025),PI(1026))
IF(IFNCTN.EQ.4)GO TO 30
CALL SCALE(YPI,5.*1024.,+1)
CALL AXIS(2.179,2.2012,24PHASE OF H(Z) (RADIANS),+24,.5,.90.,)
XYP(1025),YP(1026))
CALL PLOT(2.179,2.2012,-3)
CALL LINE(PI,YPI,1024.,1,0,0)
GO TO 40
30 IF(IFNCTN.EQ.2)GO TO 50
CALL AXIS(2.179,2.2012,17MAGNITUDE OF H(Z),+17,.5,.90.,YN(1025),
XYM(1026))
CALL PLOT(2.179,2.2012,-3)
CALL LINE(PI,YM,1024.,1,0,0)
GO TO 40
50 CALL AXIS(2.179,2.2012,22MAGNITUDE OF H(Z) (DB),
X*22,.5,.90.,YMDB(1025),YMDB(1026))
CALL PLOT(2.179,2.2012,-3)
CALL LINE(PI,YMDB,1024.,1,0,0)
40 CALL PLOT(-2.179,-2.2012,-3)
NUMPLT=NUMPLT+1
RETURN
END

SUBROUTINE BOX1(I,FLTRGN,NUMPLT)
C*  THIS SUBROUTINE DRAMAS THE BOUNDARIES OF THE PLOTTING AREAS
C* DIMENSION CORNER(6,2)
DATA CORNER /1.0,.1,.13,.0,.1,.1,10.,-10.,10.,-10.,10./
CALL FACTOR(.491)
CALL PLOT(CORNER(NUMPLT,1),CORNER(NUMPLT,2),-3)
CALL PLOT(-12.375,0.,+2)
CALL PLOT(12.375,-9.5625,+2)
CALL PLOT(0.9,-56.25,+2)
CALL PLOT(0.,0.,+2)
IF(IF.0)RETURN
CALL SYMBOL(7.4922,1.83,.139,14HFILTER GAIN = .0 , +14)
ASDF7722
ASDF7723
ASDF7724
ASDF7725
ASDF7726
ASDF7727
ASDF7728
ASDF7729
ASDF7730
ASDF7731
ASDF7732
ASDF7733
ASDF7734
ASDF7735
ASDF7736
ASDF7737
ASDF7738
ASDF7739
ASDF7740
ASDF7741
ASDF7742
ASDF7743
ASDF7744
ASDF7745
ASDF7746
ASDF7747
ASDF7748
ASDF7749
ASDF7750
ASDF7751
ASDF7752
ASDF7753
ASDF7754
ASDF7755
ASDF7756
ASDF7757
ASDF7758
ASDF7759
ASDF7760
ASDF7761
ASDF7762
ASDF7763
ASDF7764
ASDF7765
ASDF7766
ASDF7767
ASDF7768
ASDF7769
REFERENCES


<table>
<thead>
<tr>
<th>No.</th>
<th>Name and Contact Information</th>
<th>No. Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Defense Technical Information Center</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Cameron Station</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alexandria, Virginia</td>
<td></td>
</tr>
<tr>
<td></td>
<td>22314</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Library, Code 0142</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Naval Postgraduate School</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monterey, California</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93940</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Department Chairman, Code 62</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Department Of Electrical Engineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Naval Postgraduate School</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monterey, California</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93940</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Professor D. C. Kirk, Code 62Ki</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Department Of Electrical Engineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Naval Postgraduate School</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monterey, California</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93940</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Professor S. Parker, Code 62Px</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Department Of Electrical Engineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Naval Postgraduate School</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monterey, California</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93940</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Professor R. D. Strum, Code 62St</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Department of Electrical Engineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Naval Postgraduate School</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monterey, California</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93940</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Professor Robert H. Nunn, Code 69Nm</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Department of Mechanical Engineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Naval Postgraduate School</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monterey, California</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93940</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Ms. Jane Foust, Code 0141</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Naval Postgraduate School</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Monterey, California</td>
<td></td>
</tr>
<tr>
<td></td>
<td>93940</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Captain Gerald S. Doyle, USA</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Orchard Drive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Chester, New Jersey</td>
<td></td>
</tr>
<tr>
<td></td>
<td>07930</td>
<td></td>
</tr>
</tbody>
</table>