NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN INTERACTIVE COMPUTER AIDED DESIGN AND ANALYSIS PACKAGE

by

Terrence L. Ewald

March 1986

Thesis Advisor: G. Thaler

Approved for public release; distribution is unlimited

URITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION aval Postgraduate School; 6b. OFFICE SYMBOL (If applicable) 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
ADDRESS (City, State, and ZIP Code) onterey, California 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

TITLE (Include Security Classification)
N INTERACTIVE COMPUTER AIDED DESIGN AND ANALYSIS PACKAGE

PERSONAL AUTHOR(S)
wald, Terrence L.

TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
ster's Thesis	FROM ____ TO ____	86 March	139

SUPPLEMENTARY NOTATION

COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Interactive Control Design Package

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

In this thesis the development of a controls system analysis package is iscussed. It is an interactive computer aide for the design and analysis f linear, single input/single output feedback control systems. The rogram is user friendly; it uses menus for option selections and rompts for data entry. The program will manipulate the transfer functions f multi-loop systems to produce the open and closed loop transfer unctions required for a variety of analysis techniques. The output from ny analysis may be either a tabulation of data points or a high resolution lot.

DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
NAME OF RESPONSIBLE INDIVIDUAL Prof G. Thaler	22b TELEPHONE (Include Area Code) (408)646-2134; 22c. OFFICE SYMBOL 62Tr

FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

An Interactive Computer Aided Design and Analysis Package

by

Terrence L. Ewald
Lieutenant, United States Navy
B.S., University of Missouri, Columbia, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1986

ABSTRACT

In this thesis the development of a controls system analysis package is discussed. It is an interactive computer aid for the design and analysis of linear, single input / single output feedback control systems. The program is user friendly; it uses menus for option selections and prompts for data entry. The program will manipulate the transfer functions of multi-loop systems to produce the open and closed loop transfer functions required for a variety of analysis techniques. The output from any analysis may be either a tabulation of data points or a high resolution plot.

TABLE OF CONTENTS

I. INTRODUCTION

The controls engineer must be concerned with devoting more of his time designing and less in time consuming manipulations of transfer functions and computer programming. Control systems analysis involves a great deal of calculation, especially when dealing with large systems. These calculations involve the manipulation of transfer functions. In the design process, where the characteristics of the system components may change many times, the repetitious recalculation of transfer functions can be very time consuming. Classical analysis techniques are, generally, graphical in nature. The drawing and redrawing of plots for analysis can also take up a considerable amount of the designer's time. The use of a computer to manipulate transfer functions and draw plots allows the engineer to spend more time designing.

A. BACKGROUND

Using computers to help solve engineering problems is certainly not a new idea. In fact, many computer aids already exist. Programs that are currently available at the Naval Postgraduate School include DSL (Dynamic Simulation Language), TUTSIM, IODE, PARAMA, PAROLE, CONTROLS, and more. CONTROLS is the only interactive package of the ones mentioned. As will be explained later, CONTROLS was used as a model because of its ease of use and user friendliness. Unfortunately, most of these programs are specialized in a particular type of analysis technique and are very hard to learn and use.

These deficiencies might require the designer to implement several different computer aid programs in the process of analyzing a simgle problem. These programs most likely are not compatible with one another and would require data in different formats.

In the case of programs that are not interactive, the user must type data into a data file using a very specific format. If the data is not formatted properly, the program will halt execution or will run to completion producing meaningless output and often will not inficate the source of error. The user is given little or no control in a non-interactive program. Too much time is wasted in running a program from start to finish each time a change is made to the data file and the user's time is spent as a computer programmer and operator.

B. PROBLEM OBJECTIVE

The objective of this project was to design a comprehensive program which would meet the needs of the designer without the problems listed in the previous section. The program should provide the most commonly used classical control system analysis techniques. It must be interactive. The user should be able to change any of the system parameters at any time without retyping all parameters. He should also be able to go from one analysis technique to another without having to re-enter any of the system parameters. The program should be easy to operate or be "user friendly." Errors input by the user should be easy to detect and correct. The user must have as much control of the program as possible without requiring him to know any special language or commands.

These requirements are fairly general in nature. There are several ways of meeting these requirements, particularly in the area of user friendly, interactive programming. The following chapter describes, in more detail, the methods used in fulfilling the objectives of this project.

II. PROGRAM DEVELOPMENT

The package developed in this thesis was designed to meet the objectives outlined in the previous chapter. It is an interactive package written in FORTRAN to run on the IBM 370 model 3033 computer at the Naval Postgraduate School. The FORTRAN programming language was chosen for the following reasons:

1) A large library of FORTRAN functions and subroutines exist to perform routine numerical calculations.

2) FORTRAN programs can interface with the state-of-the-art graphics package DISSPLA.

3) Since FORTRAN is a common programming language, it will be easy for programmers to alter and improve this package in the future.

A. APPROACH TO THE PROBLEM

A top-down programming technique was used to develop this package. First, the tasks that the program was to perform were defined. Then, each task was developed in greater detail until, finally, the very specific routines required to perform each task were implemented. Of course, a few new tasks were defined and routines created during the development of the program and were incorporated in parallel with existing routines.

The tasks which the program performs may be broken into three basic categories:

1) Those involved in manipulating the system's transfer functions.

2) Those involved in performing analysis of the system.

3) Those involved in the control of the program.

The approach used in accomplishing each of these tasks is discussed in the following subsections.

1. System Manipulation

This package uses block diagrams to depict control systems. Each block in the diagram represents a component in the system and is described by a transfer function.

Analysis is usually performed on the system's open or closed loop transfer functions. Although this package accepts nested and cascaded loops, the system will be formulated into an equivalent single feedback control loop consisting of blocks in the forward and feedback paths for which the open and closed loop transfer functions may be calculated. Each block's transfer function is defined by a numerator and denominator polynomial in either coefficient or factored form. This package makes available 20 blocks for transfer functions in the control loops. In most cases, this should allow the user to input the system exactly as it appears in a given block diagram. The system will define blocks consecutively as they are entered. These numbers do not necessarily determine the order of the blocks. The numbering of the blocks will be explained in more detail in chapter three.

It is common for a single input/single output system to consist of several nested feedback loops. These systems can be reduced to single loops by calculating the closed loop transfer function of the inner loops and including them as separate blocks in the next outer loop. This method of block diagram manipulation is performed automatically by this package and used to reduce multi-loop systems to a single loop for analysis. The user should be aware that this program does not handle the case of coupled loops.

Routines were developed to allow the user to input multi-loop systems as they appear in a block diagram by starting on the innermost loop and expanding to succeeding outer loops. This package can be made to calculate the closed loop transfer function for the inner loop and place it anywhere in the next outer loop. The coefficients for the equivalent open loop system can also be obtained; these are necessary for the root locus portion of the program due to the feature of allowing two variable parameters in calculating root loci.

It was a primary objective to be able to change the system at any time with minimum effort. To this end, routines were developed to allow the user to change a single coefficient or root, redefine an entire polynomial, redefine an entire block or add new blocks. It was required that any of these changes could be made without affecting any other part of the block diagram and that the resulting changes in the system's open and closed loop transfer functions be automatically recalculated.

2. Analysis Techniques

One of the objectives of the project was to make available, in a single package, several control system analysis techniques. The following is a list of the analysis techniques chosen for inclusion:

1) Root locus analysis
2) Closed loop Bode frequency response analysis
3) Open loop Bode frequency response analysis
4) Time response analysis

It was also a requirement that the user be able to switch from one analysis technique to another without having to reenter any system parameters. This was accomplished by using common block variables and arrays. The only exception is in the root locus portion. Here the user must decide what the two variable parameters are and enter the coefficients accordingly. These coefficients can be obtained from the transfer function manipulation before entering the root locus portion.

3. Program Control

The environment that the user works in is modeled after the one used in a similar interactive program called DACSAP written by Clifton M. Cooksey [Ref. 1]. Developing a flexible, user friendly program where the operator would not need to know a programming language or special commands was another objective of this project. This was accomplished using two concepts. First, the program is somewhat self-managing. This concept is applied only in those areas where total user control is not felt to be necessary. For example, some programs require that the user instruct the program to calculate the open or closed loop transfer functions after the system has been entered and prior to commencing any analysis. If the system is changed, instructions must, again, be given to recalculate the transfer functions required for further analysis.

Transfer functions are automatically calculated as they are needed to perform analysis calculations. These transfer functions are also automatically updated when any changes are made to the system by the user. Although some control of the program is taken away from the user in these cases, the chance of making an error is also reduced and the user's time is spent more efficiently.

The second concept used in program control is the use of option menus. A variety of menus are presented to the user throughout the program's execution showing the options available to him at that time. Self-management is also sometimes used in the control of the option menus. That is, the user is rarely given the opportunity to select which menu he will see next. Rather, the correct menu is automatically presented to the user based on where execution is occurring in the program and which tasks have been completed to that point.

The result of menu driven options is that the options available are presented in plain English so that the user does not have to know any special language or commands. Since the control of the menus is handled by the program, the user is always presented with a menu of options when one is required and that menu contains only options which are of interest to him at that time. These concepts allow the program execution to flow with a minimum number of inputs and far fewer errors.

B. ORGANIZATION OF THE PROGRAM

Low level routines were written based on their required output. The input requirements of these routines, then, defined the output of the next higher routine. This process of building upward was continued until the required data was that data which must be supplied by the user. For this reason, the parameter input routines form the highest levels of the program.

The basic organization of the program is shown in Figure 2.1. The lowest level routines are the analysis routines; these were written first. The calculation routines were written next and the parameter input routine last. The parameter input routines take inputs from the user and put them into forms which may be used by the calculation routines. The user controls the program primarily through these parameter input routines. Thus, these routines contain the instructions for presenting option menus and for changing and correcting parameters.

The uppermost controlling routine is the driving routine which provides the main option menu. This routine shifts program control to the parameter input routines based on the selection from the main menu. The lines with

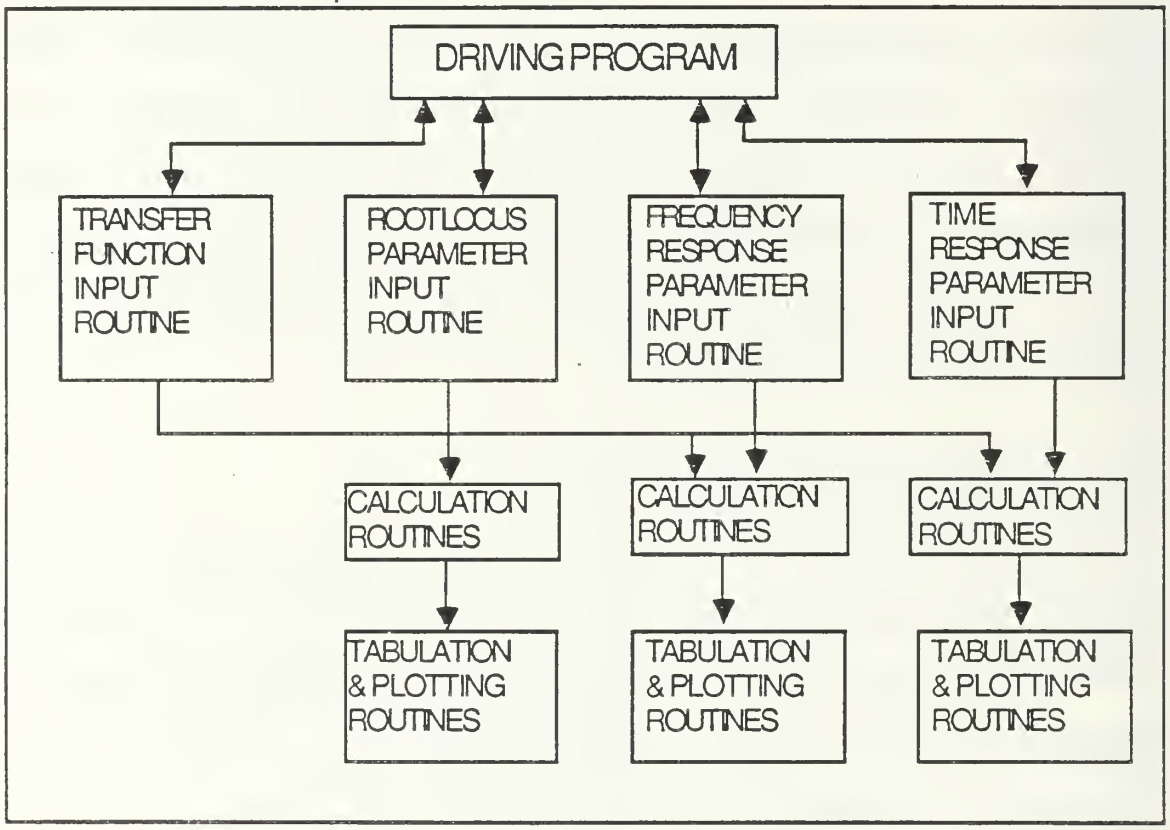

DRIVING PROGRAM
TRANSFER FUNCTION INPUT ROUTINE
ROOTLOCUS PARAMETER INPUT ROUTINE
FREQUENCY RESPONSE PARAMETER INPUT ROUTINE
TIME RESPONSE PARAMETER INPUT ROUTINE
CALCULATION ROUTINES
CALCULATION ROUTINES
CALCULATION ROUTINES
TABULATION & PLOTTING ROUTINES
TABULATION & PLOTTING ROUTINES
TABULATION & PLOTTING ROUTINES

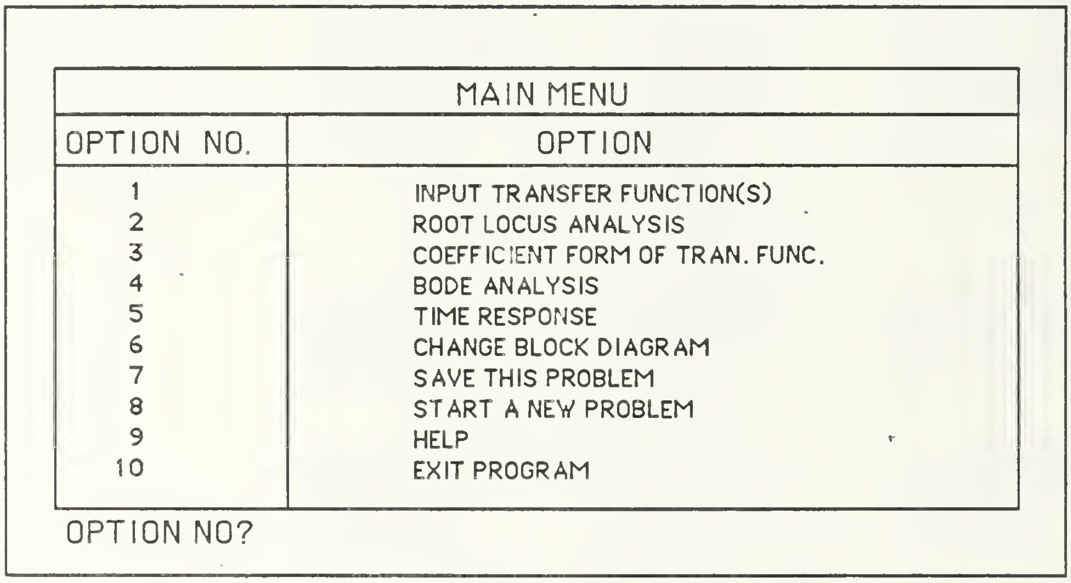
Figure 2.1 Program Organization

double arrow heads in Figure 2.1 show how control is shifted. Once the required tasks have been performed by the analysis routines, control is shifted back to the driving program. The shifting of control to the calculation and output routines is done automatically and is not controlled by the user.

The flow of information is depicted by the lines with single arrow heads in Figure 2.1. The user supplies system data via the transfer function input routine and analysis parameters via the root locus, frequency response and time response parameter input routines. Information is taken from these routines by the calculation routines which, in turn, provide information to the tabulation or plotting routines.

III. PROGRAM DESCRIPTION

This package consists of a small driving program and many subroutines. The driving program does little more than present the user with the main option menu (Fig. 3.1) and call the appropriate subroutines based on the selection made from that menu.



```
                          MAIN MENU
OPTION  NO.   |                 OPTION
      1       |     INPUT TRANSFER FUNCTION(S)
      2       |     ROOT LOCUS ANALYSIS
      3       |     COEFFICIENT FORM OF TRAN. FUNC.
      4       |     BODE ANALYSIS
      5       |     TIME RESPONSE
      6       |     CHANGE BLOCK DIAGRAM
      7       |     SAVE THIS PROBLEM
      8       |     START A NEW PROBLEM
      9       |     HELP
     10       |     EXIT PROGRAM
OPTION NO?
```

Figure 3.1 Main Option Menu

All calculations, input/output functions and remaining program control functions are performed by the subroutines. These subroutines fall into one of the following four basic categories:

1) parameter input routines
2) calculation and analysis routines
3) plotting or tabulation (output) routines
4) utility routines.

The four main branches of the program (transfer function input and three analysis sequences) all contain subroutines from each of these categories, except for the transfer function input sequence which has no output routine. Each of the four branches will be described in greater detail later in this chapter.

First, however, it is necessary to take a closer look at the control structure of the branches. Program self-management is used to a greater extent when an analysis routine is first entered. The drawing shown in Figure 3.2 depicts the

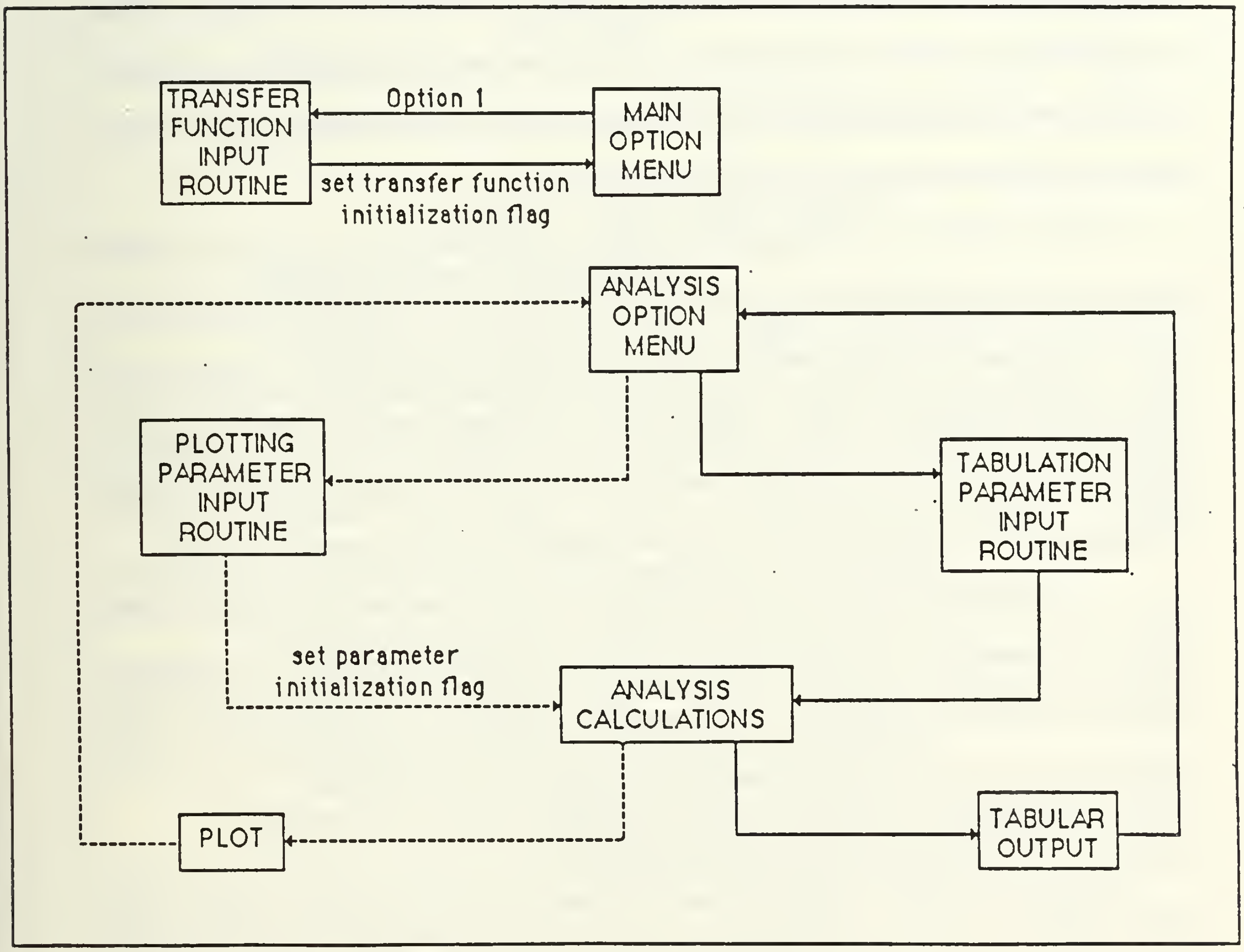

TRANSFER FUNCTION INPUT ROUTINE
Option 1
MAIN OPTION MENU
set transfer function initialization flag
ANALYSIS OPTION MENU
PLOTTING PARAMETER INPUT ROUTINE
TABULATION PARAMETER INPUT ROUTINE
set parameter initialization flag
ANALYSIS CALCULATIONS
PLOT
TABULAR OUTPUT

Figure 3.2 Control Structure Before Parameter Initialization

relationship between the main menu, the transfer function input routine and the analysis routines. Before analysis can be performed, the system must be defined by using Option 1 of the main menu. The only exception to this is when the user wants to draw only a root locus and already has the needed coefficients. Once the system has been input, the transfer function initialization flag is set. The significance of this is explained later.

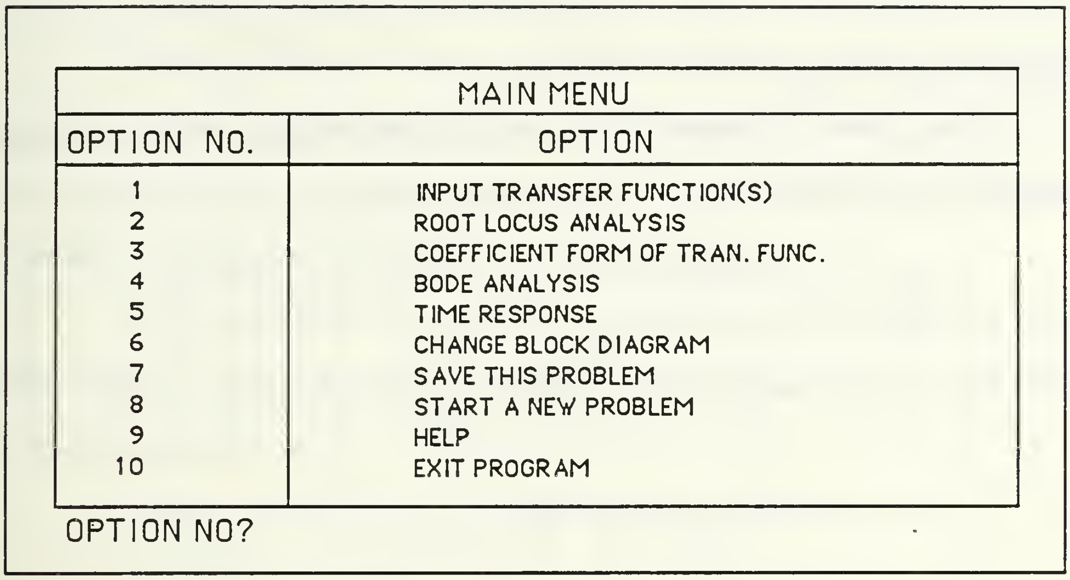
When an analysis technique is chosen from the main menu, the user is presented with the option menu for that branch. The user may select graphical or tabular output which will send him to the appropriate parameter input routine. Once all of these parameters have been entered, another flag is set. Then, analysis calculations and output automatically occur.

After the first run through the analysis routines, the structure of control changes to that depicted in Figure 3.3. The change from the first structure (Fig. 3.2) to this one is dependent on the condition of the transfer function and parameter initialization flags. Note that more direct control of the program is given to the user via the change menus. This allows the user to make selected changes to any parameter and rerun the analysis in any sequence desired.

The following sections briefly describe the function and capabilities of the four main branches. The basic mathematical concepts used will be stated but not carried out in any detail. The reader is referred to the bibliography of this report for a list of texts which cover control system theory in detail.

A. TRANSFER FUNCTION MANIPULATIONS

Transfer function manipulations are performed by several subroutines which permit the user to interactively input, change, store or load a system's transfer functions.



MAIN MENU	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	COEFFICIENT FORM OF TRAN. FUNC.
4	BODE ANALYSIS
5	TIME RESPONSE
6	CHANGE BLOCK DIAGRAM
7	SAVE THIS PROBLEM
8	START A NEW PROBLEM
9	HELP
10	EXIT PROGRAM

```
OPTION NO?
```

Figure 3.1 Main Options Menu

1. Transfer Function Input Routines

The purpose of the input routine, TINPUT, is to permit the user to input a system's transfer functions just as they appear in a block diagram, then calculate the system's open loop transfer function for use by the analysis routines. Because of the large number of calculations required for transfer function manipulation, TINPUT is the largest of the subroutines.

The first time that the routine is entered, the user is given the opportunity to input the transfer functions from the console or from a data file. Entry from the console and from a data file will be discussed.

The analysis routines perform their calculations based on a single loop system with all blocks in the forward and feedback paths. The user may have a system of nested or cascaded loops, but these are all reduced to their equivalent closed loop transfer functions and treated as blocks in either the forward or feedback path of the final loop.

First, consider how to enter a simple single loop system. Suppose that we have the system shown in Figure 3.4 below.

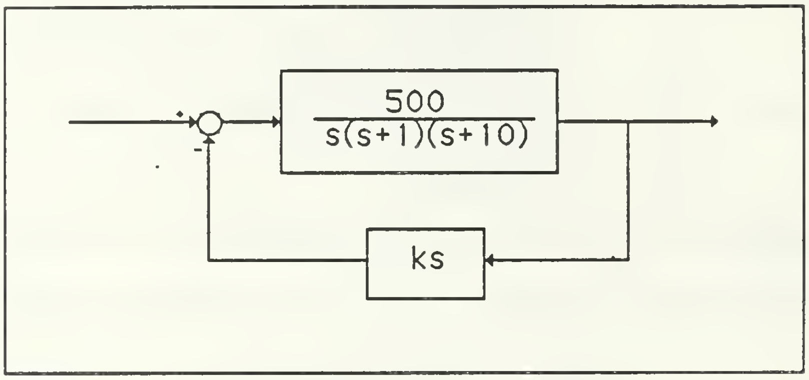
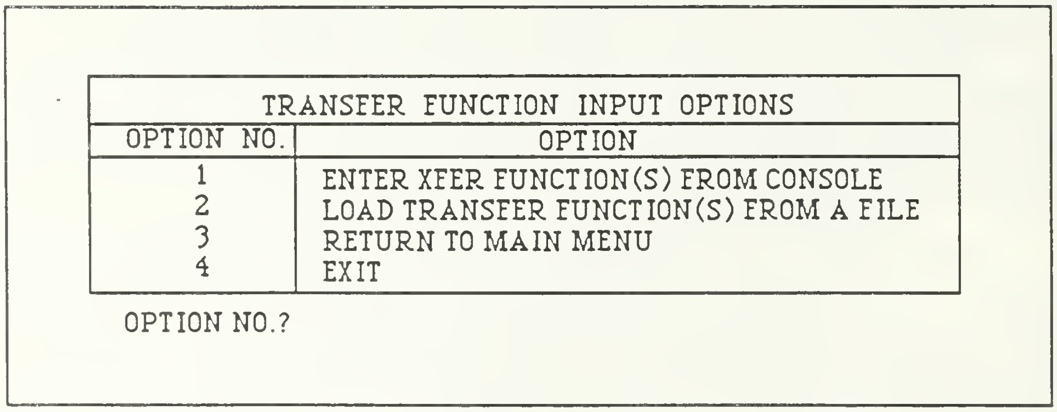


Figure 3.4 Single Loop System

For the purpose of this example, assume that k=.1. In order to define the system, select Option 1 from the main menu shown earlier in Figure 3.1. Selecting Option 1 from the main menu will produce the menu of transfer function input options shown in Figure 3.5.



TRANSFER FUNCTION INPUT OPTIONS	
OPTION NO.	OPTION
1	ENTER XFER FUNCTION(S) FROM CONSOLE
2	LOAD TRANSFER FUNCTION(S) FROM A FILE
3	RETURN TO MAIN MENU
4	EXIT

OPTION NO.?

Figure 3.5 Transfer Function Input Menu

Option 2 from this menu should be selected if the user has saved a system during a previous terminal session and would now like to continue work on that system. The user will be prompted for the filename, then the system will be loaded and transfer function input will be complete.

In the case where a transfer function is being defined for the first time, select Option 1 from the Transfer Function Input Options menu. This results in the following questions:

```
WHICH LOOP ARE YOU PREPARING TO INPUT?

HOW MANY BLOCKS ARE IN YOUR OPEN-LOOP?
```

For the system in Figure 3.4, the user would enter a 1 for the first question since there is only one loop. When working with nested loops, the user must start with the innermost loop as will be demonstrated in a following example. The second question is answered by entering a 2.

It should be noted that blocks which consist of purely a gain should not be entered as transfer functions. The program requests values for open loop gain, forward path and feedback path gain whenever they are needed for calculations. The following questions are asked for each of the blocks in the open loop:

```
TRANSFER FUNCTION NO.  1

IS THIS BLOCK IN THE FORWARD OR FEEDBACK PATH? (F OR B)

DO YOU WISH TO INPUT THE TRANSFER FUNCTION FOR THIS
BLOCK FROM A DATA FILE OR FROM THE CONSOLE?  (D OR C)
```

For this first block, the user answers F for the first question since it is in the forward path. Since the program has not been run previously, the second question is answered with a C to enter data from the console. This will result in the following questions:

WHAT IS THE ORDER OF THE NUMERATOR POLYNOMIAL?

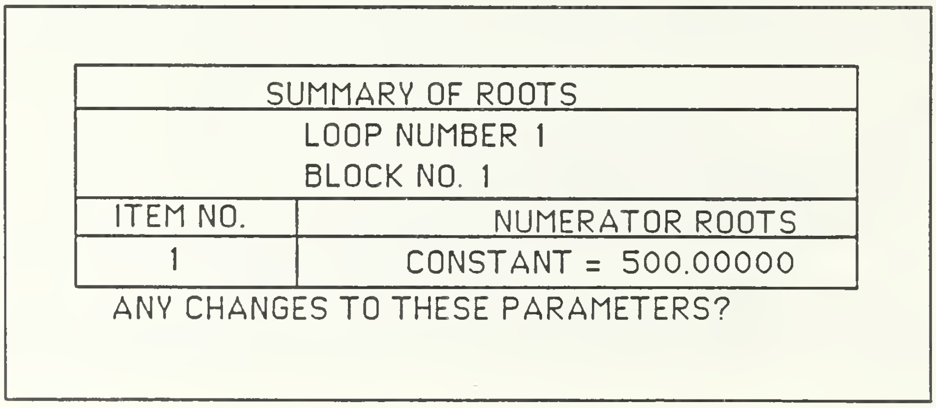
WHAT IS THE ORDER OF THE DENOMINATOR POLYNOMIAL?

IS THIS TRANSFER FUNCTION IN FACTORED OR COEFFICIENT FORM? (F OR C)

The order of the numerator is 1. The order of the denominator is 3. The transfer function is in factored (F) form. Note that this program assumes the following form for each term in factored form (S - root). The numerator and denominator of each block must be of the same form. For transfer functions in factored form, input is prompted as follows:

WHAT IS THE NUMERATOR GAIN CONSTANT?

The answer to this question for the above system is 500. Since there are no zeros in the numerator, the system will not prompt for roots. After the numerator gain is entered, the system will present the parameters just entered and give the user the chance to change them as shown in Figure 3.6.



```
                    SUMMARY OF ROOTS
                    LOOP NUMBER 1
                    BLOCK NO. 1
 ITEM NO.   |           NUMERATOR ROOTS
     1      |       CONSTANT =  500.00000
 ANY CHANGES TO THESE PARAMETERS?
```

Figure 3.6 Summary of Numerator Roots

If there are zeros, the system will prompt for roots in the same manner as shown below for the denominator.

```
WHAT IS THE DENOMINATOR GAIN CONSTANT?

ROOT NO.  1
     REAL PART =
     IMAGINARY PART =


ROOT NO.  2
     REAL PART =
     IMAGINARY PART =


ROOT NO.  3
     REAL PART =
     IMAGINARY PART =
```

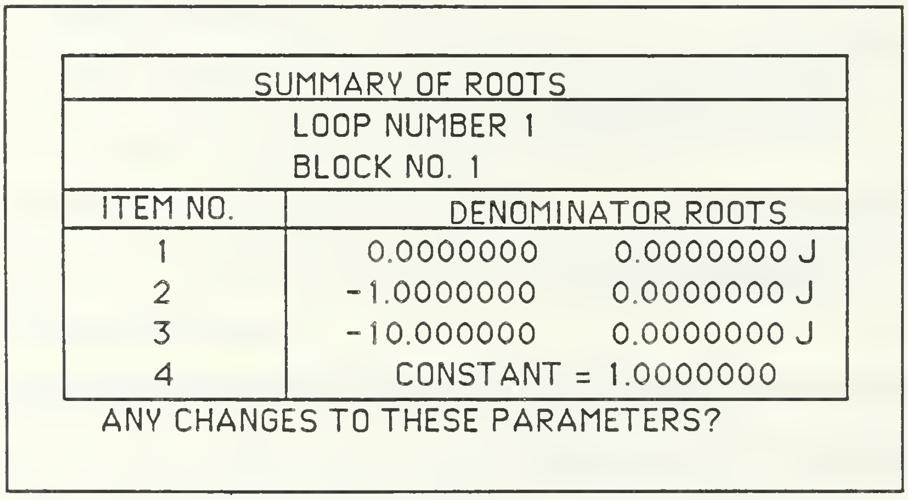
The denominator gain constant is 1. There are three poles in the denominator and they would be entered as follows:

N	REAL PART	IMAGINARY PART
1	0	0
2	-1	0
3	-10	0

After the denominator parameters have been entered, the system will give the user a chance to change them as shown in Figure 3.7. If the user had chosen to calculate the numerator and denominator polynomials in coefficient form, he would be prompted as follows:

```
NUMERATOR COEFFICIENTS

     COEFFICIENT  OF  S  **  0  =
```

In the above example, the user would respond by entering 500. The system then presents the coefficients just entered and gives the user the chance to change them.



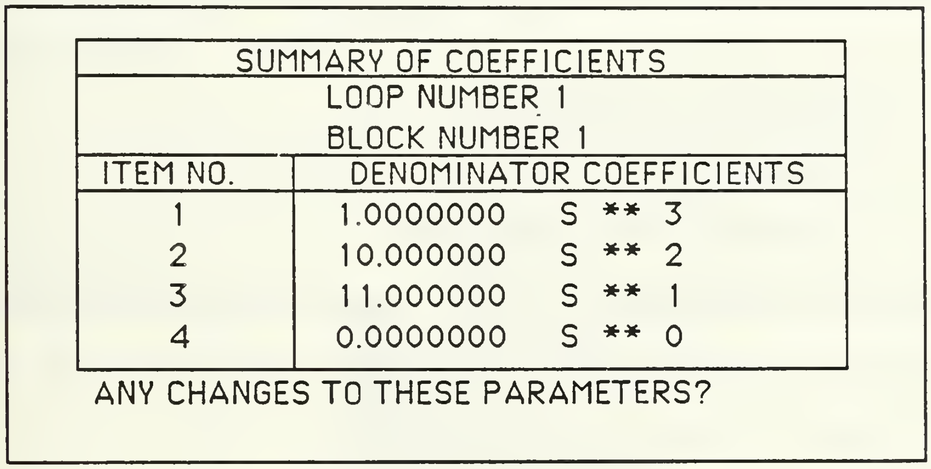
```
                 SUMMARY OF ROOTS
                   LOOP NUMBER 1
                   BLOCK NO. 1
   ITEM NO.          DENOMINATOR ROOTS
      1          0.0000000       0.0000000 J
      2         -1.0000000       0.0000000 J
      3         -10.000000       0.0000000 J
      4            CONSTANT = 1.0000000
  ANY CHANGES TO THESE PARAMETERS?
```

Figure 3.7 Summary of Denominator Roots

The denominator is entered in the same manner. The program will prompt the user as follows:

```
DENOMINATOR COEFFICIENTS
     COEFFICIENT OF S ** 3 =
     COEFFICIENT OF S ** 2 =
     COEFFICIENT OF S ** 1 =
     COEFFICIENT OF S ** 0 =
```

For the above system, the user would enter 1, 11, 10 and 0 in that order. After they have been input, the coefficients are displayed as shown in Figure 3.8 and may be corrected as necessary.



```
                SUMMARY OF COEFFICIENTS
                    LOOP NUMBER 1
                    BLOCK NUMBER 1
ITEM NO.       DENOMINATOR COEFFICIENTS
    1          1.0000000     S ** 3
    2          10.000000     S ** 2
    3          11.000000     S ** 1
    4          0.0000000     S ** 0
ANY CHANGES TO THESE PARAMETERS?
```

Figure 3.8 Summary of Denominator Coefficients

At this point, the program will prompt the user for information concerning block number 2 as follows:

TRANSFER FUNCTION NO. 2

IS THIS BLOCK IN THE FORWARD OR FEEDBACK PATH? (F OR B)

DO YOU WISH TO INPUT THE TRANSFER FUNCTION FOR THIS BLOCK FROM A DATA FILE OR FROM THE CONSOLE? (D OR C)

For the system above, the user would answer B to indicate this block is in the feedback path and C to enter the block parameters from the console. The program would then prompt the user for the following information exactly as for the first block.

WHAT IS THE ORDER OF THE NUMERATOR POLYNOMIAL?

WHAT IS THE ORDER OF THE DENOMINATOR?

IS THIS TRANSFER FUNCTION IN FACTORED OR COEFFICIENT
FORM? (F OR C)

For the system above, the user would enter 1, 0 and C. The user could alternatively enter F to indicate factored form if desired. These answers are followed by prompts for the numerator and denominator coefficients as shown below.

NUMERATOR COEFFICIENTS

COEFFICIENT OF S ** 1

COEFFICIENT OF S ** 0

In this case, the answers are .1 and 0. Again the program will present the information just entered and ask if any changes need to be made. Then the program prompts for the denominator coefficients.

DENOMINATOR COEFFICIENTS

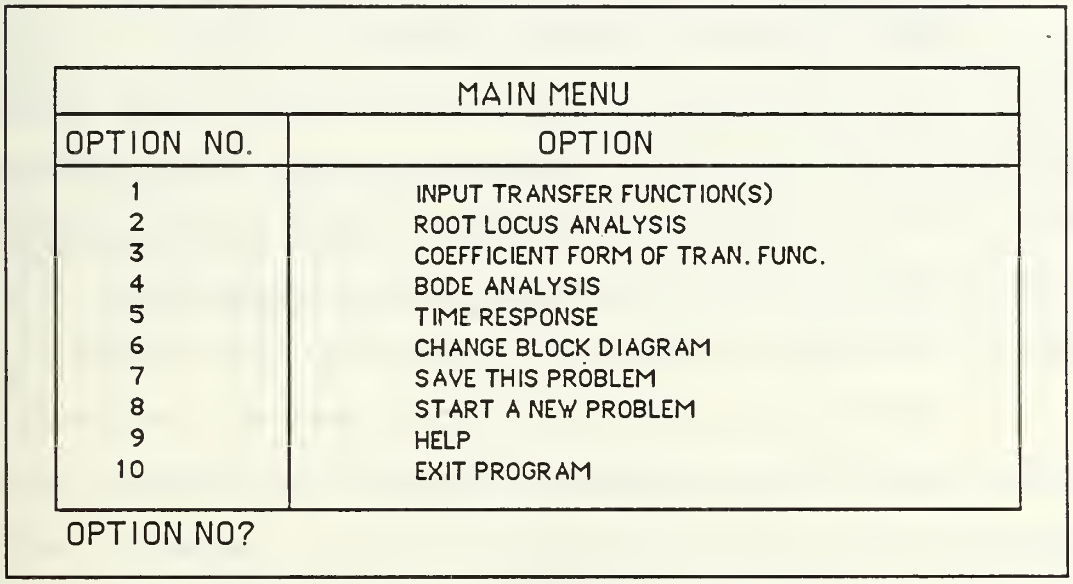
COEFFICIENT OF S ** 0

The response for this example is 1. The system recognizes that there is essentially no denominator in this case and therefore will not present the denominator coefficients to the user.

After the data for both transfer functions has been entered, the program will again ask if the user wants to change the system just entered. At this point, the user can change any information that has just been entered or expand to an outer loop and enter more information about the system. This will be explained in more detail later in this section. For now, just assume that the entire system is the one presented above. In this case, the user would answer with N to the following question.

WANT TO MAKE ANY CHANGES TO YOUR TRANSFER
FUNCTION(S)? (Y OR N)

The program then returns to the main menu shown in Figure 3.1 and repeated below. The user could choose to perform some sort of analysis at this point, but first suppose he wishes to save the system just entered for later use. In order to save the system, choose Option 7 from the main menu. The program then asks the user to specify



```
                        MAIN MENU
OPTION  NO.  |                 OPTION
     1       |         INPUT TRANSFER FUNCTION(S)
     2       |         ROOT LOCUS ANALYSIS
     3       |         COEFFICIENT FORM OF TRAN. FUNC.
     4       |         BODE ANALYSIS
     5       |         TIME RESPONSE
     6       |         CHANGE BLOCK DIAGRAM
     7       |         SAVE THIS PROBLEM
     8       |         START A NEW PROBLEM
     9       |         HELP
    10       |         EXIT PROGRAM

OPTION NO?
```

Figure 3.1 Main Options Menu

a name for the file where the problem will be saved.

```
WHAT NAME DO YOU WISH TO GIVE YOUR DATA FILE?
(8 CHARACTERS MAX)
```

The user must enter only a filename. The program automatically assigns DATA as the file type and A1 as the file mode. The program also assumes that this is a new file. If the user names a file that already exists, that information will be lost. After the user names a file, the program presents the name to the user and gives him the chance to change his mind.Assume the user wants to name his file LOOP1.

```
FN FT FM = LOOP1 DATA A1
IS THIS CORRECT?  (Y OR N)
```

An affirmative response to this question results in the following question.

```
DO YOU WISH TO HAVE THE CLOSED LOOP TRANSFER
FUNCTION CALCULATED AND SAVED?  (Y OR N)
```

If the user wants to subsequently analyze the system being saved, he should answer with N since the analysis routines automatically form the closed loop system. If the system being saved corresponds to a block in a larger system, the user should respond with Y. This case will be demonstrated below. After the system has been saved, the user is returned to the main menu of Figure 3.1.

The system may consist of up to 20 components in the forward and feedback paths of a single feedback control loop. Each component must be defined by a transfer function and by the path (forward or feedback) in which it is located. The numerator and denominator polynomials for each transfer function are entered separately and may be described by either polynomial coefficients or polynomial roots. After each polynomial has been entered, the user is presented with the coefficients or roots just entered and given the opportunity to change any which may have been entered incorrectly.

Instead of typing in the transfer function polynomials for a given block, the user may choose to load the block's transfer function from a data file. If this option is chosen, the numerator and denominator polynomial coefficients are pulled from the disk file, via the subroutine TLOAD, and placed in the appropriate arrays for the current block. The purpose of this feature is to allow one to design subsystems separately and store their transfer functions to disk.

Then, when designing the outer loops of the system, the subsystem may be loaded directly into its proper block.

As previously mentioned, when the transfer function input routine is first entered, the user may choose to load the entire system from a data file. If this option is chosen, all data pertaining to a stored control loop will be loaded from a user specified data file. This feature allows the user to continue the analysis or design of a system that had been worked on during a previous terminal session without having to retype any system parameters. When the load is completed, the program will continue its operation as if the user had just finished entering the transfer functions from the console.

Now suppose that the system of interest is the one shown in Figure 3.9.

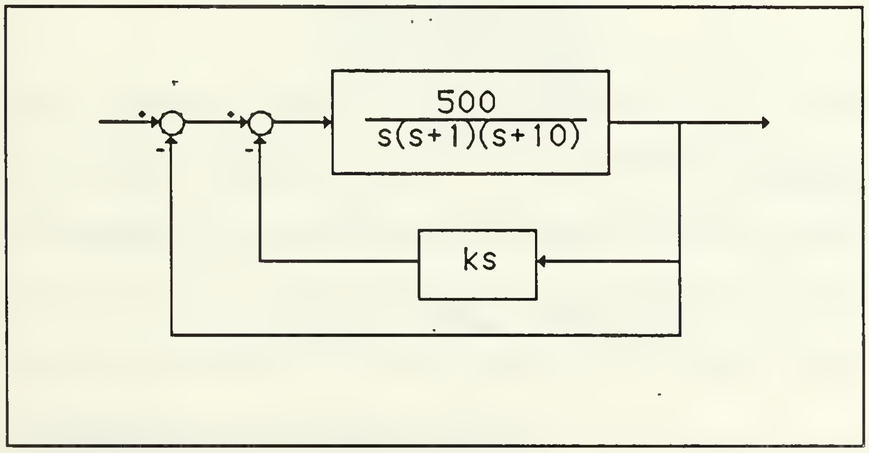
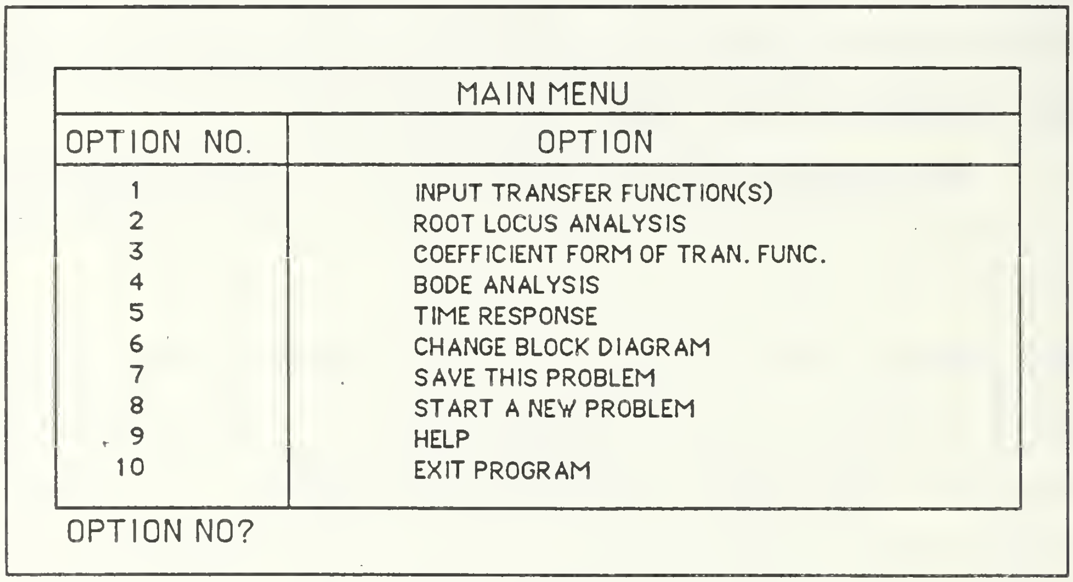


Figure 3.9 Nested Loops

As in the previous system, assume that k=.1. To enter this system, the user must choose Option 1 from the main menu shown in Figure 3.1 and repeated below. Choosing Option 1 from the main menu results in the menu shown if Figure 3.5 and repeated below. The user could enter all the information from the console if so desired. Assume that in the previous system example the user opted to save the closed loop transfer function in the file LOOP1. The user may then enter the inner loop of the system in Figure 3.9

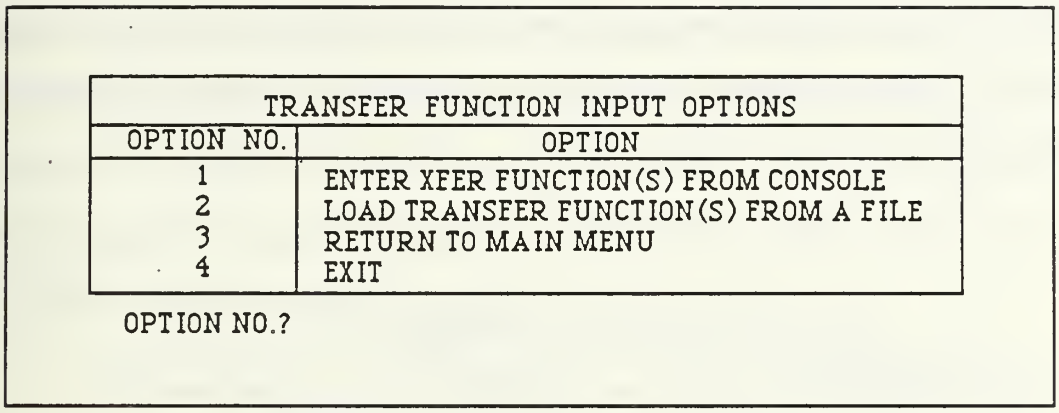


```
                         MAIN MENU
OPTION NO.                    OPTION
    1              INPUT TRANSFER FUNCTION(S)
    2              ROOT LOCUS ANALYSIS
    3              COEFFICIENT FORM OF TRAN. FUNC.
    4              BODE ANALYSIS
    5              TIME RESPONSE
    6              CHANGE BLOCK DIAGRAM
    7              SAVE THIS PROBLEM
    8              START A NEW PROBLEM
    9              HELP
   10              EXIT PROGRAM
OPTION NO?
```

Figure 3.1 Main Options Menu

from a datafile by choosing Option 2. The program will then ask the user for the name of the file where the information is stored.

WHAT IS THE NAME OF YOUR DATA FILE? (FILENAME ONLY)



TRANSFER FUNCTION INPUT OPTIONS	
OPTION NO.	OPTION
1	ENTER XFER FUNCTION(S) FROM CONSOLE
2	LOAD TRANSFER FUNCTION(S) FROM A FILE
3	RETURN TO MAIN MENU
4	EXIT

```
OPTION NO.?
```

Figure 3.5 Transfer Function Input Menu

In this case, the user would type in the name LOOP1. The program then asks the following:

```
FN FT FM = LOOP1  DATA  A1
IS THIS CORRECT?  (Y OR N)
```

An affirmative response results in the program loading the information from the file LOOP1 and then asking the user if he would like to change the transfer function. A response of N would return the user to the main menu. A response of Y would send the user to the Transfer Function Change Options menu. The rest of the system in Figure 3.9 will be entered in the following section.

2. Transfer Function Change Routine

The instructions required to change or expand the block diagram are also contained in TINPUT. When any changes are made to the system, the routine automatically recalculates the parameters of the open loop transfer function. The option to change transfer functions may be selected from the main menu or from any of the analysis routines' change menus (or by answering affirmatively when program asks the user if he would like to change the transfer function). Those changes which may be performed on the block diagram are shown in Figure 3.10.

TRANSFER FUNCTION (T.F.) CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE A T.F. IN THE CURRENT LOOP
2	ADD A NEW BLOCK TO THE CURRENT LOOP
3	CHANGE T.F. IN AN INNER LOOP
4	EXPAND TO AN OUTER LOOP
5	NO CHANGES
6	HELP

Figure 3.10 Transfer Function Change Menu

Now, finish entering the system started at the end of the previous section. Assume the user answered with Y when the program asked the user if there were any changes to the transfer function just loaded from data file LOOP1. This would result in the above menuAlso, assume that the user did not save the closed loop transfer function. The transfer function entered from LOOP1 is actually the inner loop of the entire system. To enter the rest of the system, the user must expand to an outer loop. This is done by selecting Option 4 from the Transfer Function Change Options menu above in Figure 3.9. This results in the following:

```
IF YOU THINK YOU MIGHT WANT TO GO BACK AND MAKE
CHANGES TO THE CURRENT LOOP, YOU SHOULD SAVE IT.

DO YOU WISH TO SAVE THE TRANSFER FUNCTION(S) IN
THIS LOOP? (Y OR N)
```

Since the user has already saved this information, he would respond with N to this question. Even if the system had not been saved, if the user feels there is no reason to change the inner loop at some later time he could decide not to save the present transfer function. The program then asks the user for information concerning the expansion.

```
DO YOU WISH TO INPUT THE OUTER LOOP TRANSFER
FUNCTION FROM A DATA FILE OR FROM CONSOLE? (D OR C)

INTO WHICH BLOCK OF THE NEW LOOP DO YOU WISH TO PLACE
THE CLOSED LOOP TRANSFER FUNCTION OF THE INNER LOOP?

IS THIS BLOCK IN THE FORWARD OR FEEDBACK PATH? (F OR B)
```

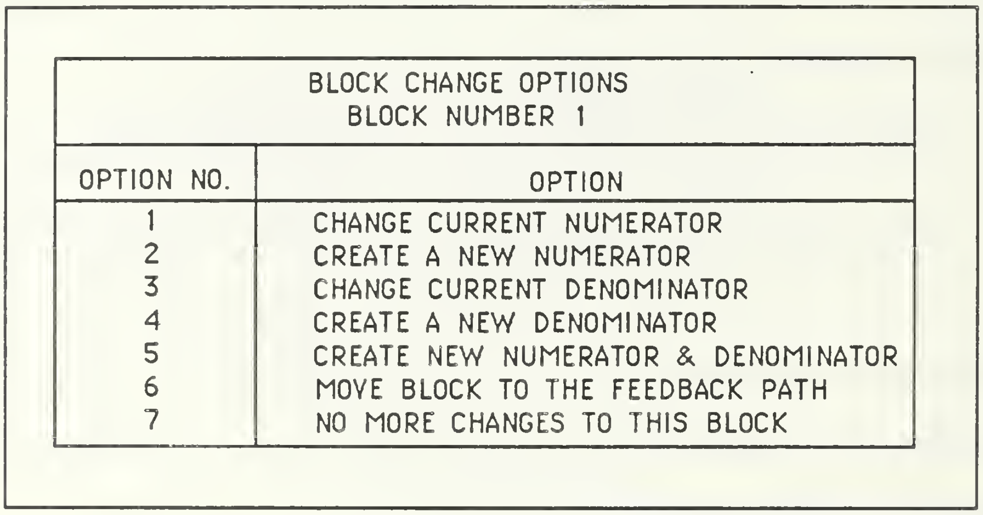
For this case the user would answer these questions with C, 1 and F. At this point the program calculates the closed loop transfer function of the inner loop and places it in the equivalent block of the outer loop.

```
THE CLOSED LOOP TRANSFER FUNCTION FROM THE INNER
LOOP HAS BEEN PLACED IN BLOCK NO. 1 OF THIS LOOP.

HOW MANY BLOCKS (INCLUDING BLOCK NO. 1) ARE IN
THIS LOOP?
```

For the case being considered there is only one block. Once again, the program will ask the user if he wants to make any changes to the transfer function. The user would answer negatively; he is then returned to the main menu. This completes entry of the nested loop system presented in the Figure 3.8.

If the user answers affirmatively when asked if he wants to change the transfer function, he will be presented with the menu in Figure 3.9. If the first option is chosen, the user will be asked which block (if more than one block exists) he wishes to change and is presented with the block change option shown in Figure 3.11.

Changing the block's numerator or denominator (options 1 or 3) means that the user simply wishes to alter one or more of the polynomial's coefficients or roots leaving all other block parameters unchanged.



BLOCK CHANGE OPTIONS
BLOCK NUMBER 1

OPTION NO.	OPTION
1	CHANGE CURRENT NUMERATOR
2	CREATE A NEW NUMERATOR
3	CHANGE CURRENT DENOMINATOR
4	CREATE A NEW DENOMINATOR
5	CREATE NEW NUMERATOR & DENOMINATOR
6	MOVE BLOCK TO THE FEEDBACK PATH
7	NO MORE CHANGES TO THIS BLOCK

Figure 3.11 Block Change Menu

By creating a new numerator or denominator (options 2 or 4), the user will be able to change the order of the polynomial and must enter all of the roots or coefficients for the polynomial. The form of the polynomial (factored or coefficient) must, however, remain the same as previously selected.

By selecting the option to create a new numerator and denominator (Option 5), all data concerning the current block is, essentially, erased and the user is sent to the transfer function input routine to re-enter a transfer function for that block. This permits one to enter transfer function polynomials of different form and order into any block of the control loop. Just as one may initially enter transfer functions from a file, the user may choose to create the new numerator and denominator by loading them from a data file.

One may move a block from the forward path to the feedback path and vice versa (Option 6). All other features about the block, including the block number, remain unchanged. Only its location within the control loop is changed.

If the second option from Figure 3.10 is chosen, the user is simply sent to the transfer function input routine to enter data for a new block. The block may be added to the forward or feedback path and may be entered from the console or from a data file.

This package was designed to operate on a single feedback control loop. However, since control loop transfer functions may be saved to and loaded from data files, multiple loop systems may be designed and analyzed in the following manner.

The user must design the inner loop(s) of the system first, such as Loop 1 in Figure 3.12 or as in the case presented above. Once the designer is satisfied that Loop 1 meets his specifications, he saves the system parameters and starts a new problem by loading the inner loop's closed loop transfer function into an outer loop block and entering the remaining blocks as he wishes. Option 4 of Figure 3.10 simplifies this process. As mentioned previously, the user does not have to save the inner loop's closed loop transfer function and then start a new problem. He can merely expand to an outer loop as demonstrated in the previous example. At this time, the program will ask the user if he wants to save the inner loop transfer function. The program will continue from this point as in the previous example whether the user saves the inner loop or not.

The user may choose to place the inner loop transfer function into any block in the next loop. In the case shown in Figure 3.12, block 2 is selected and the outer loop appears as shown in 3.13. It should be noted that the closed loop transfer function does not have to go into block number 1. It could go into any block number since all blocks

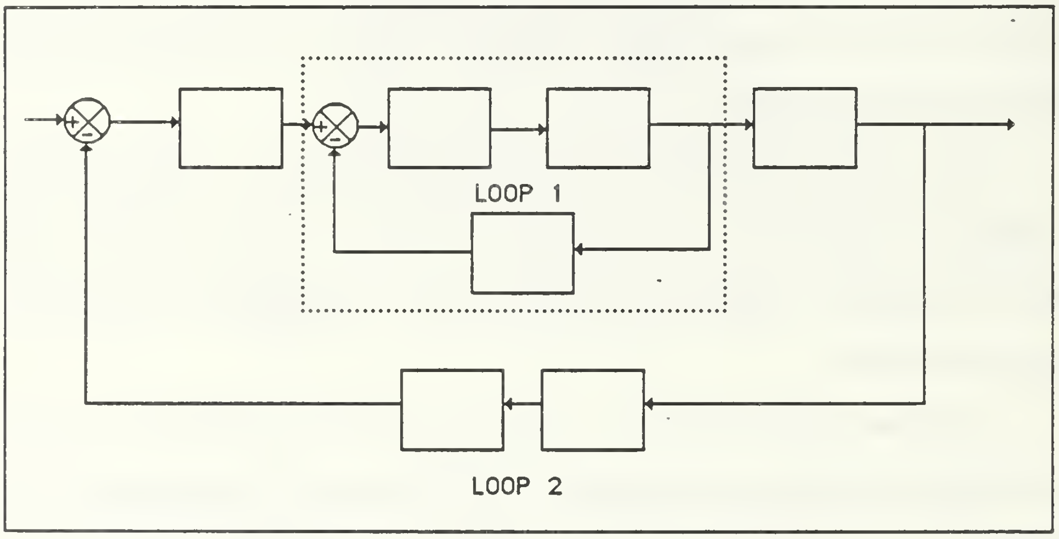


Figure 3.12 Sample Multiloop System

are in cascade. As long as the user tells the program it is in the forward path, there is no difference. The block number is merely a convenience for the user. It allows him to keep track of where blocks appear in his diagram. An important feature of this option is that the program keeps track of the block in the outer loop containing the inner loop transfer function. This becomes important when this option (Option 4) is used in conjunction with Option 3 (CHANGE A TRANSFER FUNCTION IN AN INNER LOOP).

In order to make changes in an inner loop, the user saves his outer loop, starts a new problem, loads the inner loop and makes the desired changes. Like Option 4, Option 3 (Fig 3.9) automatically controls this sequence of events, prompting the user each step of the way. After this option has been chosen and all changes have been made to the inner loop, the

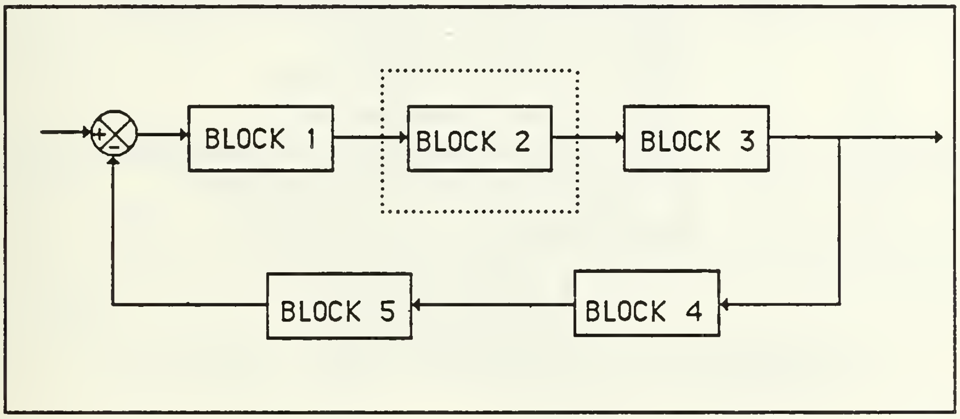
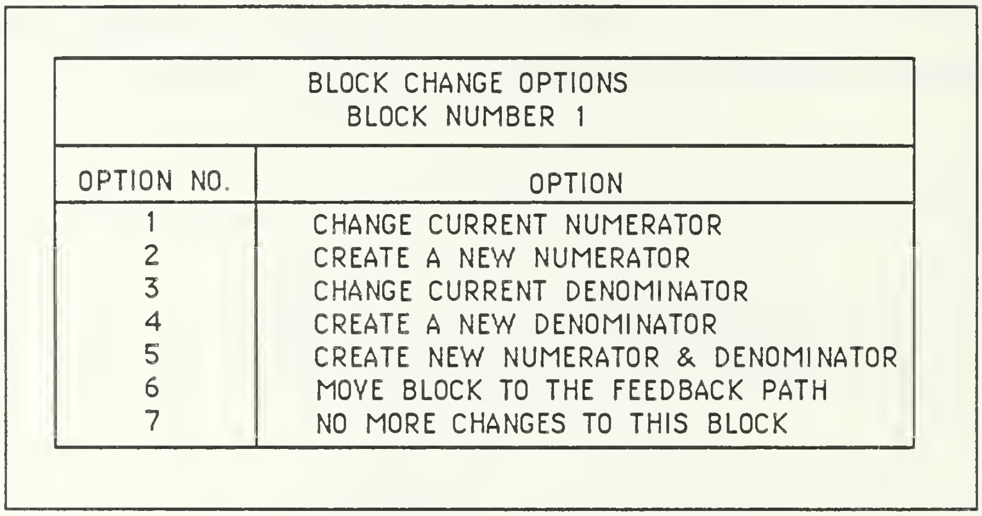


Figure 3.13 Reduced Multiloop System

user may return to the outer loop which now reflects the changes made in the inner loop. This is done by selecting Option 4 to re-expand to the outer loop. The user is asked if the outer loop should be entered from the console or a data file. Since the outer loop was saved under Option 3, its data file can now be loaded. In this case, however, since the program has kept track of the block into which the inner loop was previously placed, it automatically updates that block with the new closed loop transfer function from the inner loop.

Consider other types of changes to the transfer function. When Option 1 of the Transfer Function Change menu is selected, the user will be asked which block is to be changed. Then the following menu of block change options, shown in Figure 3.11 and repeated below, will be presented:



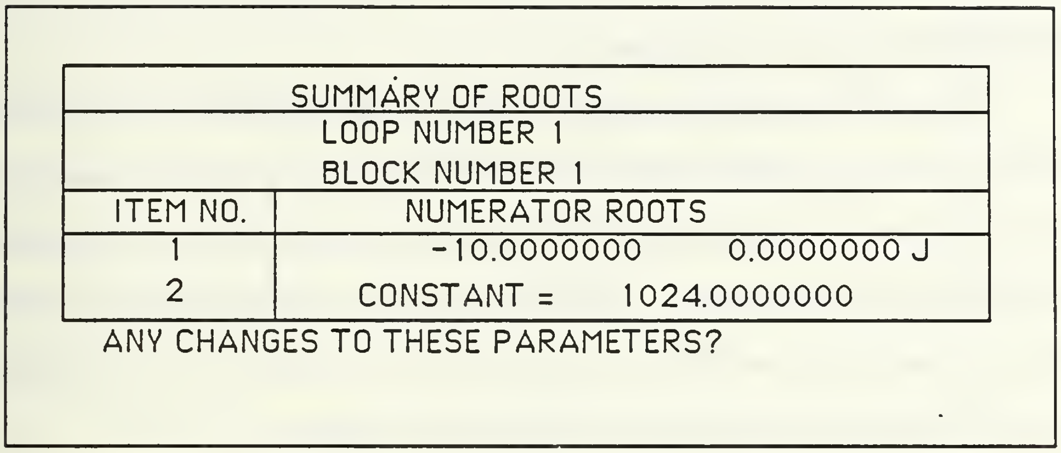
BLOCK CHANGE OPTIONS
BLOCK NUMBER 1

OPTION NO.	OPTION
1	CHANGE CURRENT NUMERATOR
2	CREATE A NEW NUMERATOR
3	CHANGE CURRENT DENOMINATOR
4	CREATE A NEW DENOMINATOR
5	CREATE NEW NUMERATOR & DENOMINATOR
6	MOVE BLOCK TO THE FEEDBACK PATH
7	NO MORE CHANGES TO THIS BLOCK

Figure 3.11 Block Change Menu

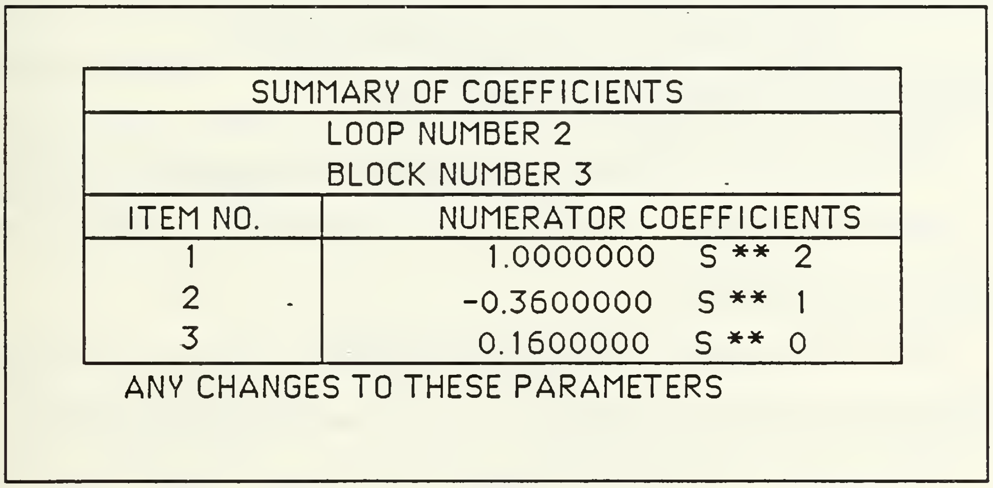
If Options 1 or 3 are selected, a summary of the roots or coefficients of the selected polynomial will be presented in one of the forms shown in Figures 3.14 and 3.15. (These might be part of some system the user has entered.)

By answering yes (Y) to the question following the summary, the user may change any of the parameters displayed. A negative response will return the user to the Block Change Options menu of Figure 3.11.. This method may be used if the user simply wishes to review the contents of a block.



```
                 SUMMARY OF ROOTS
                   LOOP NUMBER 1
                  BLOCK NUMBER 1
 ITEM NO.  |          NUMERATOR ROOTS
     1     |     -10.0000000       0.0000000 J
     2     |   CONSTANT =      1024.0000000
  ANY CHANGES TO THESE PARAMETERS?
```

Figure 3.14 Summary of Numerator Roots



```
              SUMMARY OF COEFFICIENTS
                   LOOP NUMBER 2
                  BLOCK NUMBER 3
 ITEM NO.  |      NUMERATOR COEFFICIENTS
     1     |       1.0000000    S ** 2
     2     |      -0.3600000    S ** 1
     3     |       0.1600000    S ** 0
  ANY CHANGES TO THESE PARAMETERS
```

Figure 3.15 Summary of Numerator Coefficients

If Options 2 or 4 are selected from the Block Change Options menu, the user will be prompted for all new parameters for the selected polynomial. The form of the polynomial (factored or coefficient) will remain the same as it was before. However, all other parameters may be changed including the order of the polynomial. The user will be prompted for the new parameters as he was in the transfer function input routine.

If the user wishes to change the entire character of a block, Option 5 of the Block Change Options menu should be selected. This option essentially erases the current parameters associated with the selected block and prompts the user for all new parameters. The user is free to input the transfer function in any form and entry may be from the console or from a data file. The prompts are exactly as described in the Transfer Function Input section.

Option 6 of the Block Change Options menu simply allows the user to redesignate a block as being in the feedback path if it is currently in the forward path or vice versa. Option 6 will read

```
    |    6    |  MOVE BLOCK TO THE FORWARD PATH     |
```

for blocks which are currently in the feedback path.

When no more changes to the currently selected block are required, Option 7 should be selected. The user is then returned to the Transfer Function Change menu of Figure 3.10.

If the user decided to change the pole (S+1) to (S+2) in the system of Figure 3.9 shown earlier and repeated below, he would proceed as follows.

First, assume the user has loaded the transfer function from LOOP1 where it was saved in a previous terminal session. Also, assume that the user has just responded affirmatively when asked if he would like to make any changes to the transfer function. At this point, he would be presented with the

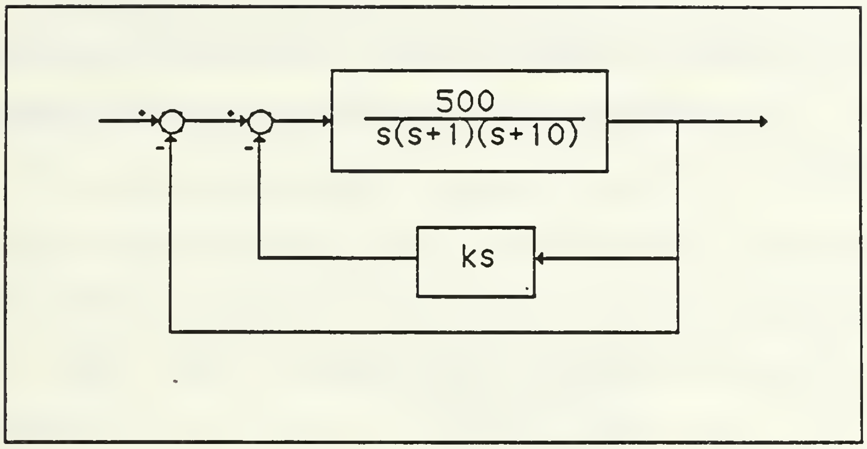
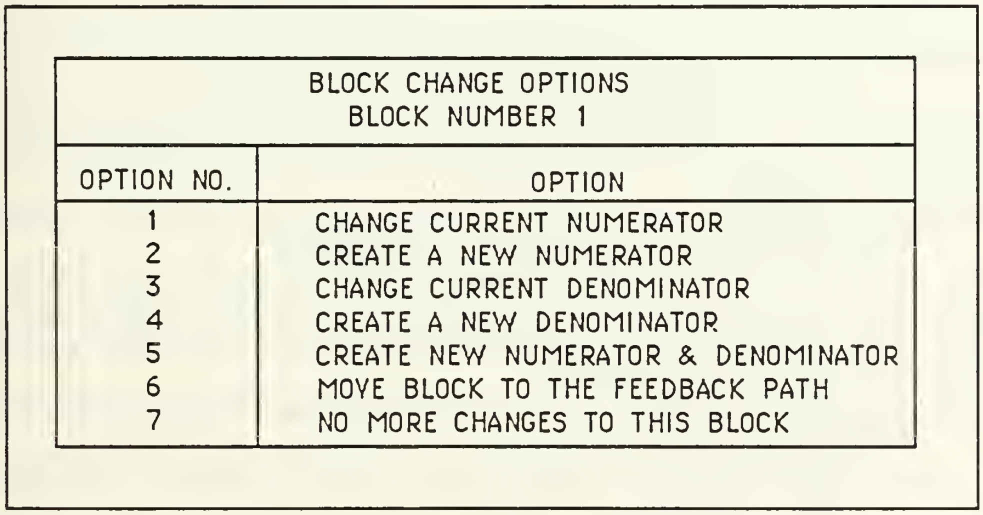


Figure 3.9 Nested Loops

Transfer Function Change Options menu. A transfer function in the current loop is being modified so the user would select Option 1 from the Transfer Function Change Options menu. He would then be presented with the Block Change Options menu of Figure 3.11 repeated here.

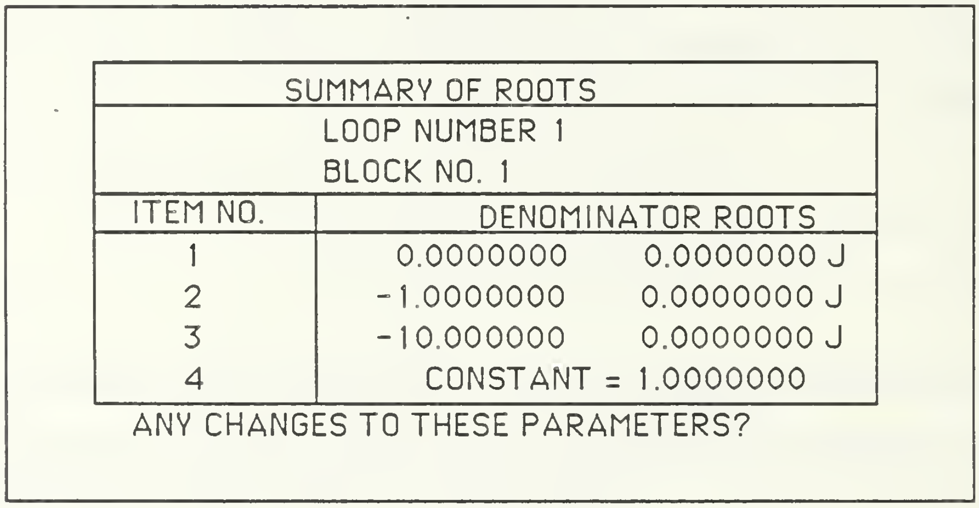


BLOCK CHANGE OPTIONS
BLOCK NUMBER 1

OPTION NO.	OPTION
1	CHANGE CURRENT NUMERATOR
2	CREATE A NEW NUMERATOR
3	CHANGE CURRENT DENOMINATOR
4	CREATE A NEW DENOMINATOR
5	CREATE NEW NUMERATOR & DENOMINATOR
6	MOVE BLOCK TO THE FEEDBACK PATH
7	NO MORE CHANGES TO THIS BLOCK

Figure 3.11 Block Change Menu

The user would select Option 3 since a pole is to be changed in the denominator. The user is then shown a summary of the present denominator as in Figure 3.16.



```
                SUMMARY OF ROOTS
                  LOOP NUMBER 1
                  BLOCK NO. 1
 ITEM NO.  |         DENOMINATOR ROOTS
     1     |   0.0000000       0.0000000 J
     2     |  -1.0000000       0.0000000 J
     3     |  -10.000000       0.0000000 J
     4     |      CONSTANT = 1.0000000
   ANY CHANGES TO THESE PARAMETERS?
```

Figure 3.16 Summary of Denominator Roots

By answering yes (Y) to the question following the summary, the user can change any of the parameters shown. The program then asks which parameter(s) to change.

WHICH ITEM DO YOU WISH TO CHANGE?

The user would enter a 2 which corresponds to (S+1). The program then responds with the following prompt.

ROOT NUMBER 2

REAL PART =

IMAGINARY PART =

In this case, the responses would be -2 and 0.

This results in the display of the new denominator and the question asking the user if he wants to change anything. If he answers no (N), he is returned to the Transfer Function Change Options menu.

One can see that larger multi-loop systems may be designed by creative use of Options 3 and 4 of the Transfer Function Change Menu. If each loop of the system is saved as it is expanded, the user is free to return to any of the inner loops, make changes and have those changes automatically incorporated in the outer loop by successively using Option 4 to re-expand the system.

Consider the system shown in Figure 3.17 as one more example of what can be done with this program.

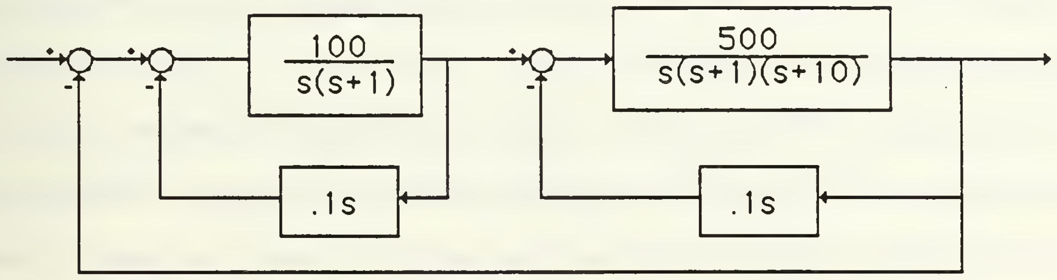
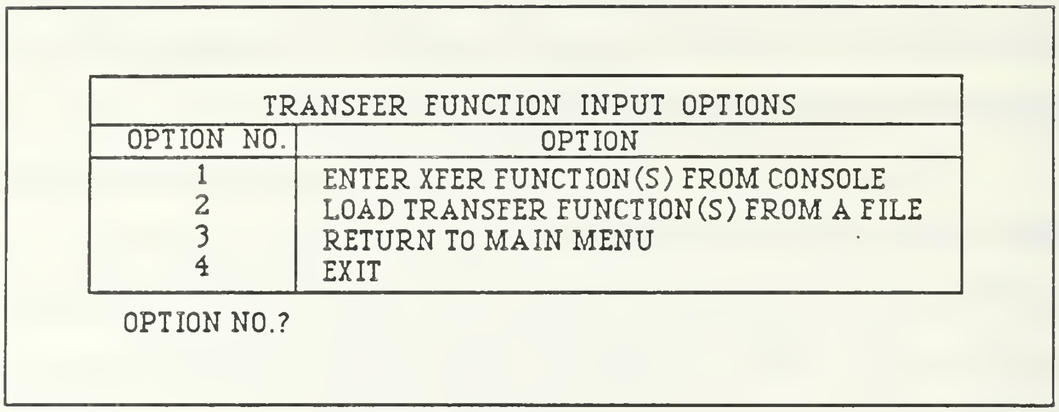


Figure 3.17 Nested and Cascaded Loops

The second inner loop in this diagram corresponds to the first example presented in Figure 3.4 and saved in the file LOOP1. Assume the first inner loop had been saved in a file named LOOP2 using the same procedure described earlier for the second inner loop. After the user has started the program and is presented with the main menu, he should select Option 1. This results in the Transfer Function Input Options menu of Figure 3.5, repeated below, being displayed.



TRANSFER FUNCTION INPUT OPTIONS	
OPTION NO.	OPTION
1	ENTER XFER FUNCTION(S) FROM CONSOLE
2	LOAD TRANSFER FUNCTION(S) FROM A FILE
3	RETURN TO MAIN MENU
4	EXIT

```
OPTION NO.?
```

Figure 3.5 Transfer Function Input Menu

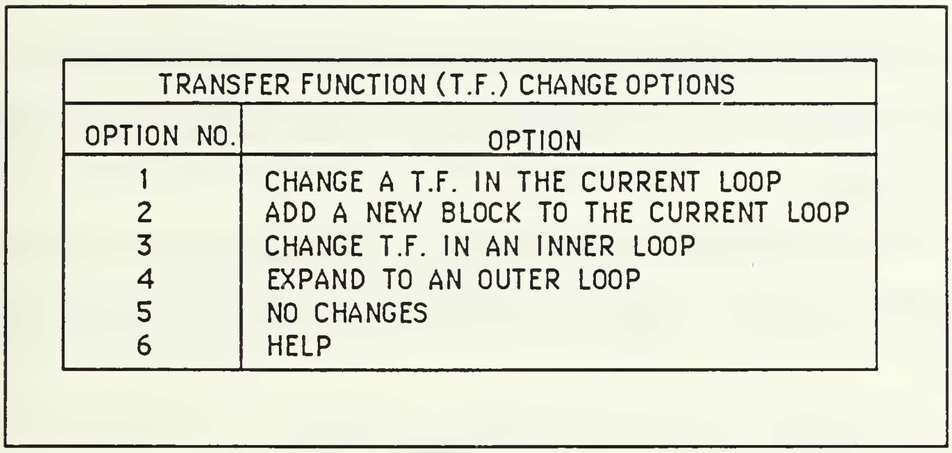
Select Option 2 from this menu. The program then asks for the name of the data file in which the transfer function is stored. (For purposes of this illustration, assume that both files contain the closed loop transfer function equivalents of the inner loops.)

WHAT IS THE NAME OF YOUR DATA FILE? (FILENAME ONLY)

The user would enter LOOP1 or LOOP2. After the program has loaded the transfer function, it will ask the user if he wants to make any changes. At this point he should answer yes (Y). The program will then present the Transfer Function Change Options menu shown in Figure 3.10 and repeated below.

There are two inner loops in cascade in the forward path of the transfer function shown above. To enter the other block, select Option 2. The program will ask the user the following question.

DO YOU WANT TO ENTER THE TRANSFER FUNCTION FOR THIS BLOCK FROM THE CONSOLE OR A DATA FILE? (C OR D)



TRANSFER FUNCTION (T.F.) CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE A T.F. IN THE CURRENT LOOP
2	ADD A NEW BLOCK TO THE CURRENT LOOP
3	CHANGE T.F. IN AN INNER LOOP
4	EXPAND TO AN OUTER LOOP
5	NO CHANGES
6	HELP

Figure 3.10 Transfer Function Change Menu

Since the information needed is assumed to be stored in either LOOP2 or LOOP1, the response should be D. The program then asks for the name of the file. After the transfer function has been read by the program, the user is given a chance to change it. If he wants to make no changes, he will be returned to the Transfer Function Change Options menu. At this point the entire system has been entered and Option 5 should be selected in order to return to the main menu. The user may now perform any analysis desired on the system just entered.

Alternatively, when the user was presented with the Transfer Function Input Options menu, he could have selected Option 1. The program would then respond with the following questions:

WHICH LOOP ARE YOU PREPARING TO INPUT?

HOW MANY BLOCKS ARE IN YOUR OPEN-LOOP?

The user would answer these by entering 1 and 2. The user is then asked if block number is to be entered from the console or from a data file. In this case, both transfer functions are in data files. After the transfer function has been entered, changes to it can be made if desired. The program then asks whether this block is in the forward or feedback path. In this example, both blocks are in the forward path.

As demonstrated above, the user has several options when entering his system information. Hopefully, it is flexible enough to handle most cases of interest.

3. Changing Transfer Functions in an Inner Loop

If the user is currently working on an outer loop of a multiloop system where the inner loop(s) were designed and saved using this package, then the inner loop(s) may be returned to and changed. Later it will be shown how to have these changes reflected in the outer loop(s).

When Option 3 of the Transfer Function Change Options menu shown if Figure 3.10 is selected the user is prompted as follows:

```
IF YOU ARE GOING TO WANT TO COME BACK TO THIS LOOP
LATER, YOU SHOULD SAVE IT.

DO YOU WISH TO SAVE THE TRANSFER FUNCTION(S) IN
THIS LOOP?  (Y  OR  N)
```

As stated, the current loop must be saved if it is to be returned to after changes are made to the inner loop. If it is not saved, its transfer functions will have to be re-entered from the console.

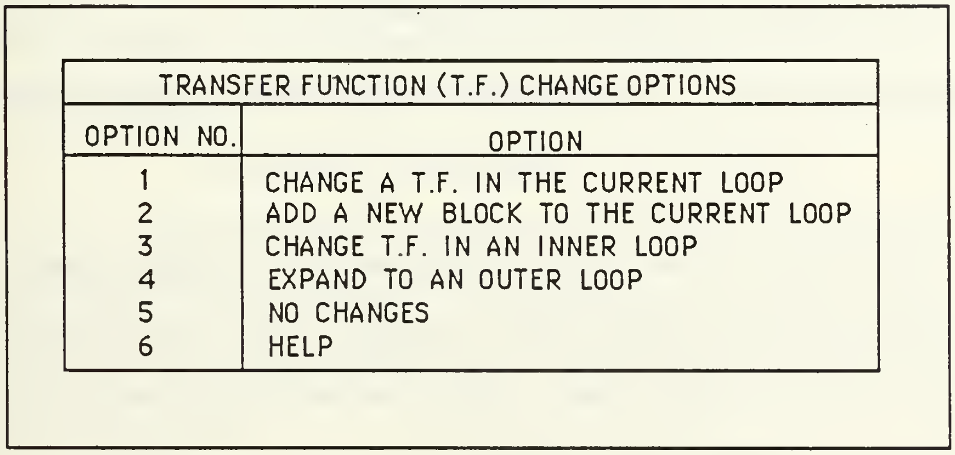
When the user selects to save the current loop he will be prompted for a filename under which it is to be saved. After the outer loop is saved, the user will be prompted for the filename of the inner loop, that loop will be loaded and the following question will appear at the terminal:

```
WANT TO MAKE ANY CHANGES TO YOUR TRANSFER

FUNCTION(S)?   (Y  OR  N)
```

If a negative response is input, the main option menu will be presented and an analysis technique for the inner loop may be selected. An affirmative response will result in the Transfer Function Change menu being presented and the user is free to change or add to the inner loop's transfer functions. It should be noted that if nothing but the forward or feedback path gains are to be changed, Option 4 of the Transfer Function Change menu should be selected to re-expand to the outer loop. The opportunity to change these gains will be given at that time, as explained in the following subsection.

4. Expanding to an Outer Loop

If the system currently being worked on is an inner loop of a multi-loop system, the user may expand the system to an outer loop, letting the program calculate the inner loop closed transfer function and place that transfer function anywhere in the outer loop. This operation can be made to occur by selecting Option 4 from the Transfer Function Change menu of Figure 3.10 repeated below.

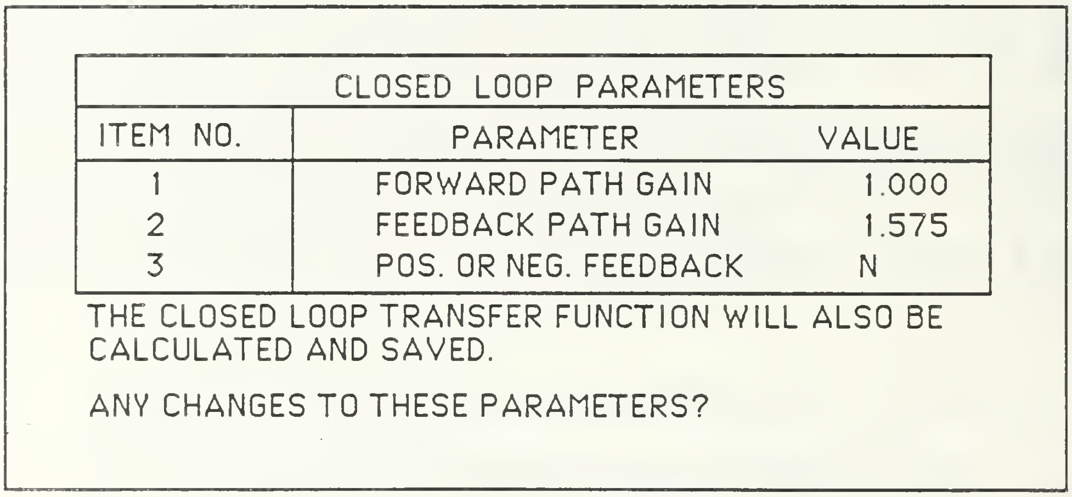


TRANSFER FUNCTION (T.F.) CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE A T.F. IN THE CURRENT LOOP
2	ADD A NEW BLOCK TO THE CURRENT LOOP
3	CHANGE T.F. IN AN INNER LOOP
4	EXPAND TO AN OUTER LOOP
5	NO CHANGES
6	HELP

Figure 3.10 Transfer Function Change Menu

Before expansion to the outer loop occurs, the user is given the opportunity to save the inner loop to a data file. This should be done if the inner loop is ever to be returned to for further analysis or changes. Prompting will be as follows:

```
IF YOU THINK YOU MIGHT WANT TO GO BACK AND MAKE
CHANGES TO THE CURRENT LOOP, YOU SHOULD SAVE IT.

DO YOU WISH TO SAVE THE TRANSFER FUNCTION(S) IN
THIS LOOP? (Y  OR  N)

WHAT NAME DO YOU WISH TO GIVE YOUR DATA FILE?(8 CHAR MAX)
```



CLOSED LOOP PARAMETERS		
ITEM NO.	PARAMETER	VALUE
1	FORWARD PATH GAIN	1.000
2	FEEDBACK PATH GAIN	1.575
3	POS. OR NEG. FEEDBACK	N

```
THE CLOSED LOOP TRANSFER FUNCTION WILL ALSO BE
CALCULATED AND SAVED.

ANY CHANGES TO THESE PARAMETERS?
```

Figure 3.18 Closed Loop Parameters

In the case shown above, the closed loop parameters had already been defined during the analysis of the inner loop. So, the user is simply given a last opportunity to change them. If they had not already been defined, the user would be prompted for them at this time.

At this point it is time to define the outer loop. The following prompt will begin the process:

```
DO YOU WISH TO INPUT THE OUTER LOOP TRAN FUNCTION(S)
FROM A DATA FILE OR FROM THE CONSOLE?  (D  OR  C)
```

Entry from the console will be discussed first.

a. Entering the Outer Loop From the Console

The first time that the system is expanded, the outer loop must be entered from the console. The user will be prompted as follows:

```
INTO WHICH BLOCK OF THE NEW LOOP DO YOU WISH TO PLACE
THE CLOSED LOOP TRANSFER FUNCTION OF THE INNER LOOP?

IS THIS BLOCK IN THE FORWARD OR FEEDBACK PATH? (F  OR  B)

THE CLOSED LOOP TRANSFER FUNCTION FROM THE INNER
LOOP HAS BEEN PLACED IN BLOCK NO.  2 OF THE NEW LOOP.

HOW MANY BLOCKS (INCLUDING BLOCK NO. 2) ARE
IN THE NEW LOOP?
```

In this case, the answer to the first question was 2. The system may have looked like the one shown below in Figure 3.19. So, the closed loop transfer function for the inner loop (enclosed by the dotted line) becomes block number 2 of the outer loop. For this system, the response to the last question would be 3.

The user is now prompted for the transfer function parameters for the remaining blocks in the outer loop (blocks 1, 3 and 4 in the example). These prompts are exactly like those presented in the Transfer Function Input section of this manual.

It is very important to note that the block number containing the inner loop's closed loop transfer function (block number 2 in the example) is always retained as a permanent parameter associated with an outer loop. This becomes important whenever an inner loop is changed and the outer loop is subsequently returned to, as discussed in the next subsection.

b. Entering the Outer Loop From a Data File

This option should be used whenever the user has returned to an inner loop to make changes and is now ready to re-expand to the outer loop. The outer loop

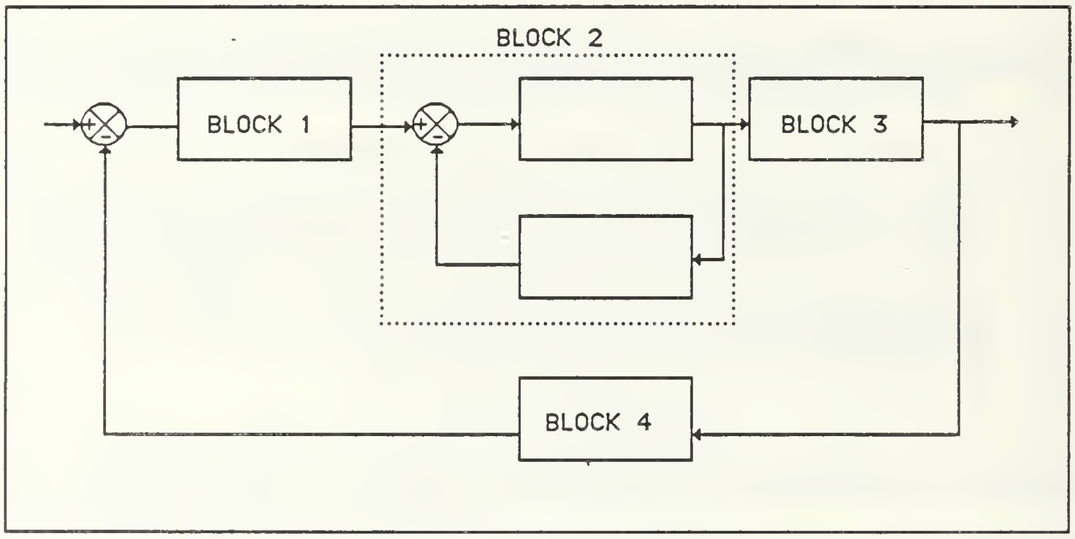


Figure 3.19 Example System

must have been saved before the inner loop was loaded in order for this option to be used.

The first time that the system was expanded to the outer loop, the user specified a block in the outer loop into which the inner would be placed. That block number became a permanent parameter associated with the outer loop. For this reason, when changes have been made to the inner loop, they will automatically be reflected by a change to the correct block in the outer loop.

After the user has chosen to expand to the outer loop (using Option 4 from the Transfer Function Change menu) and defined the closed loop parameters for the inner loop, the following prompts will be presented:

```
DO YOU WISH TO INPUT THE OUTER LOOP TRAN. FUNCTION(S)
FROM A DATA FILE OR FROM THE CONSOLE?  (D  OR  C)

WHAT IS THE NAME OF YOUR DATA FILE?  (FILE NAME ONLY)

THE CLOSED LOOP TRANSFER FUNCTION FROM THE INNER
LOOP HAS BEEN PLACED IN BLOCK N0.  2 OF THIS LOOP.
```

Naturally, one would respond with a D (data file) to the first question then provide the filename for the outer loop. The program then loads the outer loop transfer functions, but replaces the transfer function in the expansion block (block number 2 in the example) with the updated closed loop transfer function for the inner loop. A new open loop transfer function for the outer loop is also calculated using the new closed loop transfer function held in the expansion block. The user is now asked

```
WANT TO MAKE ANY CHANGES TO YOUR TRAN. FUNCTION(S)?
(Y  OR  N)
```

An affirmative response will send the user to the Transfer Function Change menu for the outer loop. A negative response sends the user to the main menu where he may select an analysis technique and observe how the changes to the inner loop have affected the total system's performance.

5. Manipulating Large Multiloop Systems

Much of the power of this package is due to its ability to manipulate large multiloop systems through the use of Options 3 and 4 of the Transfer

Function Change Menu. The key to making these features work is to insure that the system is input starting with the innermost loop then expanding one loop at a time, saving each loop along the way. If this is done, then even the innermost loop may be returned to and changed, having those changes automatically incorporated in the outer loop (via Option 4 of the Transfer Function Change menu). It should be noted that the system must be re-expanded one loop at a time when changes have been made to an inner loop. This is so each successive loop may be updated with the change to the inner loop.

The process is made particularly easy if the user simply wishes to make changes to the gains of an inner loop, which is often the case. After the inner loop is loaded, simply select Option 4 of the Transfer Function Change menu to re-expand to the outer loop. The gain parameters for the inner loop will then be presented and may be changed before the outer loop is loaded and updated.

What if an outer loop has two (or more) inner loops in series, as shown in Figure 3.20? This situation and others like it can be handled by designing the inner loops as subsystems and saving them to data files. Then when designing the outer loop, these subsytems may be loaded into their appropriate blocks. If these subsystems must later be changed, then the outer loop should be saved and the desired inner loop loaded as a new problem (Option 8 of the main menu). The inner loop should then be changed, re-saved and the outer loop loaded as a new problem. Then the old subsystem may be replaced by the new one through Option 5 of the Block Change menu.

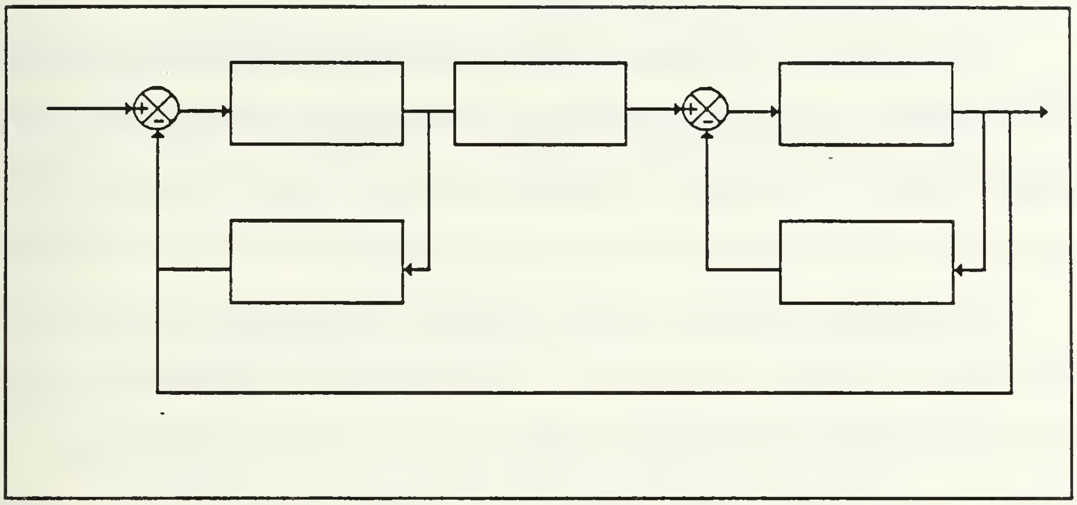


Figure 3.20 Inner Loops in Series

The user may end a terminal session and, later, pick up where he left off if he saves the system before exiting the program. If the user has designed and saved an inner loop and, during a new terminal session, wants to begin work on the next loop (Loop 2, for example), he may do so. Recall that, during the transfer function input sequence, the user is asked for the number of the current loop. In this case, the user would respond with a 2 and would be asked for the block number in Loop 2 which contains the closed loop transfer function for Loop 1. When loading the designated block, the user will be prompted for the filename under which the inner loop (Loop 1) is saved. Changes to the inner loop may then be made using Option 3 and 4 of the Transfer Function Change menu.

One can see that loop and transfer functions may be manipulated to form systems of practically any complexity. Very complex systems may be designed through the imaginative use of the subsystem block load and loop expansion features of this package.

6. Transfer Function Save Routine

The purpose of the save routine is to store all transfer function data to a disk file under a user specified file name. This function is performed by the subroutine TSAVE. TSAVE is called either by the user , through the main option menu, or by the transfer function change routine.

The data file set up by TSAVE contains two sections; the block load section and the system load section. When TSAVE is executed, the user is asked if the closed loop transfer function should be calculated and saved. The response to this question determines which transfer function, open or closed loop, will be stored in the block load section of the data file.

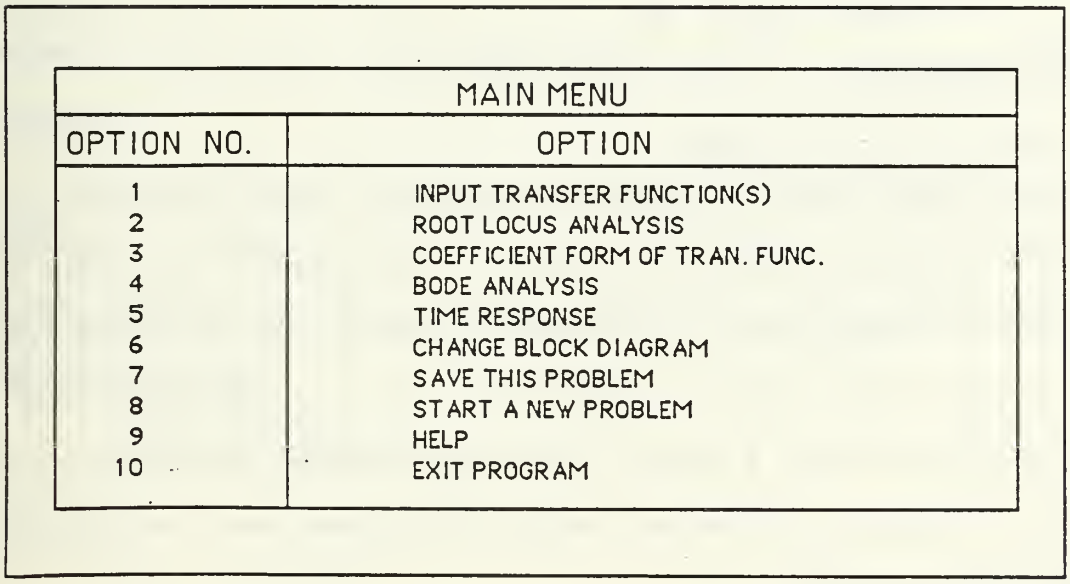
The user's decision to store one or the other is based on the type of system being saved. For example, if the designer uses open loop Bode analysis to design a cascade compensator, the open loop (cascade) transfer function is saved in the block load section. If the inner loop of a multi-loop feedback control system had just been designed, the closed loop transfer function is saved so that it can be loaded into the outer loop.

The system load section contains all data which describes the entire control loop. These parameters include the number of blocks, individual block transfer function polynomials, the open loop coefficients and roots and several other parameters. The system load section is always the same regardless of what is stored in the block load section.

The resulting data file, then, contains all of the data necessary to completely reconstruct the control loop. The entire system may be recalled later for further analysis or just the open or closed loop transfer function may be recalled for insertion into a block of another control loop.

Option 7 from the main menu of Figure 3.1, repeated below, allows the user to store the system's parameters to a data file. This may be done so that the system can be loaded during a subsequent terminal session for further analysis or to continue the design process. The user will be presented with the following two questions:

```
WHAT NAME DO YOU WISH TO GIVE TO YOUR DATA FILE?
(8 CHARACTERS MAX)

DO YOU WISH TO HAVE THE CLOSED LOOP TRANSFER
FUNCTION CALCULATED AND SAVED? (Y OR N)
```



MAIN MENU	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	COEFFICIENT FORM OF TRAN. FUNC.
4	BODE ANALYSIS
5	TIME RESPONSE
6	CHANGE BLOCK DIAGRAM
7	SAVE THIS PROBLEM
8	START A NEW PROBLEM
9	HELP
10	EXIT PROGRAM

Figure 3.1 Main Options Menu

The response to the second question is important. Recall that when inputting transfer functions, any block may be loaded from a data file. The transfer function that is loaded at that time depends on which one is saved now.

If the system being saved is a feedback system, then the closed loop transfer function should be calculated and saved. Then it can be loaded as a subsystem into a block of a larger system.

If the system being saved is a compensator, filter or any other system consisting a single component or components in cascade, then the closed loop transfer function should not be calculated. In this case, the cascade transfer function will be saved and may be loaded into a block of another system. Note that the system will be saved on the user's "A" disk as

'FILENAME' DATA A1

where "FILENAME' is the name specified by the user.

B. ROOT LOCUS ANALYSIS

This portion of the package is an interactive version of the program PAROLE written by Professor Duffin [Ref. 2]. The purpose of this routine is to calculate the complex roots of the system's characteristic polynomial over a user specified range of open loop gain values. The parameters which must be input by the user include a range of open loop gain values, the dimensions and heading for plotting, and information telling the program how to manipulate the characteristic equation.

This routine allows the user to have two variable parameters that appear linearly in the characteristic equation. In some cases, parameters that do not appear linearly can be redefined in terms of two other parameters that will be linearly related. This is left to the designer. All that is considered in this program is the linear case.

From elementary control system theory, it is known that the characteristic equation for a single feedback control loop is

$$1+N(s)/D(s)=0 \tag{3.1}$$

for negative feedback where

N(s) = numerator polynomial of the open loop transfer function

D(s) = denominator polynomial of the open loop transfer function.

Equation 3.1 may be rewritten as

$$D(s) + N(s) = 0. \tag{3.2}$$

This equation takes the form

$$\sum_{n=0}^{N} (A_n + B_nK_2 + C_nK_1)\, s^n = 0 \tag{3.3}$$

where N is the order of the system and K_2 and K_1 are the two allowed parameters.

A simple root locus of a system that does not have two parameters can be obtained by letting K_2 or K_1 equal zero and letting the other parameter correspond to the gain of the open loop system. The characteristic equation for this case would be

$$D(s) + K{*}N(s) = 0 \tag{3.4}$$

or

$$\sum_{n=0}^{N} (A_n + B_nK_2)\, s^n = 0 \tag{3.5}$$

if the user lets K_2 correspond to K.

In order to save the user from extra algebra, the program will calculate the coefficient form of the open loop system. From there, the user can add any desired compensation and easily obtain the characteristic equation.

For an example, suppose the simple root locus of the system in Figure 3.21 is desired.

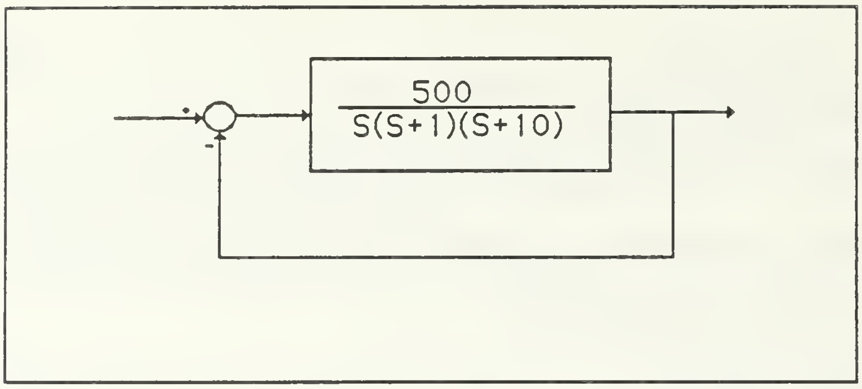


Figure 3.21 Single Loop System

The process is started by selecting Option 2 from the main menu. After being presented with two screens of information concerning the coefficients used by the root locus routine, the user is asked how the coefficients will be entered.

```
1. ENTER COEFFICIENTS FROM CONSOLE
2. ENTER COEFFICIENTS FROM DATAFILE

     (ENTER OPTION)
```

If the system had been saved during a previous root locus, the user could select option two. In that case, the program would ask the user for his file name. In this example, assume this is the first time this system has been examined. The user would enter a 1 in answer to the above question. The next question asked is the order of the system.

```
ENTER THE ORDER OF YOUR SYSTEM  (1-99)
```

In this example, the order is 3. At this point, the program prompts the user for the coefficients.

```
ENTER THE A-COEFFICIENTS

           A(3)=
           A(2)=
           A(1)=
           A(0)=
```

Remember that the A-coefficients correspond to the terms in the characteristic equation that do not contain a variable parameter. The user would enter 1, 11, 10 and 0 in that order.

Since all that is needed in this problem is the simple root locus of the system, the only variable parameter will be the gain of the open loop system. The user could choose to let this be K1 or K2. For this example, assume the user chooses to let K2 be the open loop gain, and as such would appear in the numerator of the open loop system. Recall that the B-coefficients correspond to the terms in the characteristic equation with the parameter K2 and C-coefficients correspond to the K1 terms.

```
ENTER THE B-COEFFICIENTS

              B(3)=
              B(2)=
              B(1)=
              B(0)=
```

In response, the user would enter 0, 0, 0 and 500 in that order.

```
ENTER THE C-COEFFICIENTS

              C(3)=
              C(2)=
              C(1)=
              C(0)=
```

The user could enter any numbers he wants for the C-coefficients since they will not be used for this example, but it is recommended that zeros are entered to make things less confusing.

After each set of coefficients are entered, they are displayed on screen and the user is given a chance to change them if necessary. After all coefficients are entered, one last chance to review the coefficients before starting the problem is given. This is also the point to which the program would jump if the coefficients are being loaded from a data file.

```
1. REVIEW COEFFICIENTS
2. START PROBLEM
   (ENTER OPTION)
```

Enter Option 2 in order to start the problem. The user is then asked if he wants a plot or tabulated data for output. The way the program is presently written, the user can obtain only a plot or tabulated data on a single run but not both. He would have to run the routine again in order to obtain whichever was not obtained during the first run. In most cases, the user will want to see a family of curves in order to get a general idea of what values of the parameters will provide desirable results. By first looking at a family of curves, he can limit the amount of tabulation necessary to obtain the specific values needed.

```
PRINT/PLOT OPTIONS
     1. PLOT ROOT LOCATIONS
     2. TABULATE ROOT LOCATIONS
     3. RETURN TO MAIN MENU
ENTER OPTION
```

For this illustration, a plot will be generated so the user would select Option 1 for the first prompt. If Option 2 had been selected, all subsequent questions concerning plotting would be skipped.

The user is then presented will prompts concerning the variable parameters. The user must set one of the parameters to zero in order to obtain a system root locus plot. The other variable will act as the variable gain of the system being plotted. The poles and zeros printed at the top of the root locus plot are those corresponding to the system that results from setting one of the parameters to zero. If a family of curves is being drawn, the user will later be given the chance to select specific values for the parameter initially used as the system gain while the parameter initially set to zero will then correspond to the variable gain.

```
CALCULATION OPTIONS
      1. SET K1=0
      2. SET K2=0
      3. RUN BOTH CASES
ENTER OPTION

ENTER STARTING VALUE FOR K2

ENTER NUMBER OF DECADES TO VARY K2
```

Recall that the open loop gain for this problem corresponds to the parameter K_2 and therefore the B-coefficients are used. The user would answer with Option 1 to the first prompt. The C-coefficients will not be used for this simple root locus. The second prompt asks for a starting value of the variable gain. The answer depends on what the user wants to see. For this problem, assume that 0.1 is entered. The answer to the last prompt is again dependent on how much of the root locus the user wants to see. In this case, consider entering 4 decades. This will plot the root locus for the range of gain values of K_2=.1 to K_2=1000.

The following group of questions pertain to the actual plot.

```
PLOTTING DIMENSIONS
            XMAX=
            XMIN=
            YMAX=
            YMIN=

PLOT HEADING
      HOW MANY LINES OF HEADING WOULD YOU LIKE?  (4 MAX)

      A MAX OF 32 CHARACTERS PER LINE IS ALLOWED.
      LINE 1

WOLD YOU LIKE TO SCALE THE GRAPH?  (Y OR N)
DEFAULT IS 7.5 BY 7.5 INCHES
```

For the plotting dimensions, assume the desired entries are 5, -15, 15 and -5 in that order. After these are entered, the program will present these values to the user and give him a chance to change them. The user may enter up to 4 lines of heading. In this example, one line of heading is used. The title entered was THESIS EXAMPLE. Finally, the user has the option to scale the root locus plot if desired. It is recommended that the size of the plot should not be increased since this could result in some of the plot being cut off.

At this point of the procedure, the program will plot the graph shown in figure 3.22. If the user is at a dual screen graphics terminal, the results can be seen immediately. After the curve is drawn, the program asks if the user wants to select a value for the parameter that was set to zero.

```
CALCULATION OPTIONS
    1. PICK A VALUE FOR K2
    2. QUIT
ENTER OPTION
```

Since all that is desired is the root locus that has been drawn, the user would select Option 2 in order to quit the present problem. This results in the following question.

```
WOULD YOU LIKE TO SAVE THE COEFFICIENTS FOR THIS
PROBLEM?  (Y  OR  N)
```

Answer yes (Y) to save the coefficients for this system. The user is then asked for a file name.

```
WHAT NAME DO YOU WISH TO GIVE TO YOUR DATA FILE?
(8 CHARACTERS MAX)
```

Assume the user names this file EXAMP1. This file will be used in a subsequent example. After the data has been saved, the program asks if the user would like to run another problem.

```
1. RUN ANOTHER PROBLEM
2. QUIT
    ENTER OPTION
```

Figure 3.22 shows the resulting plot.

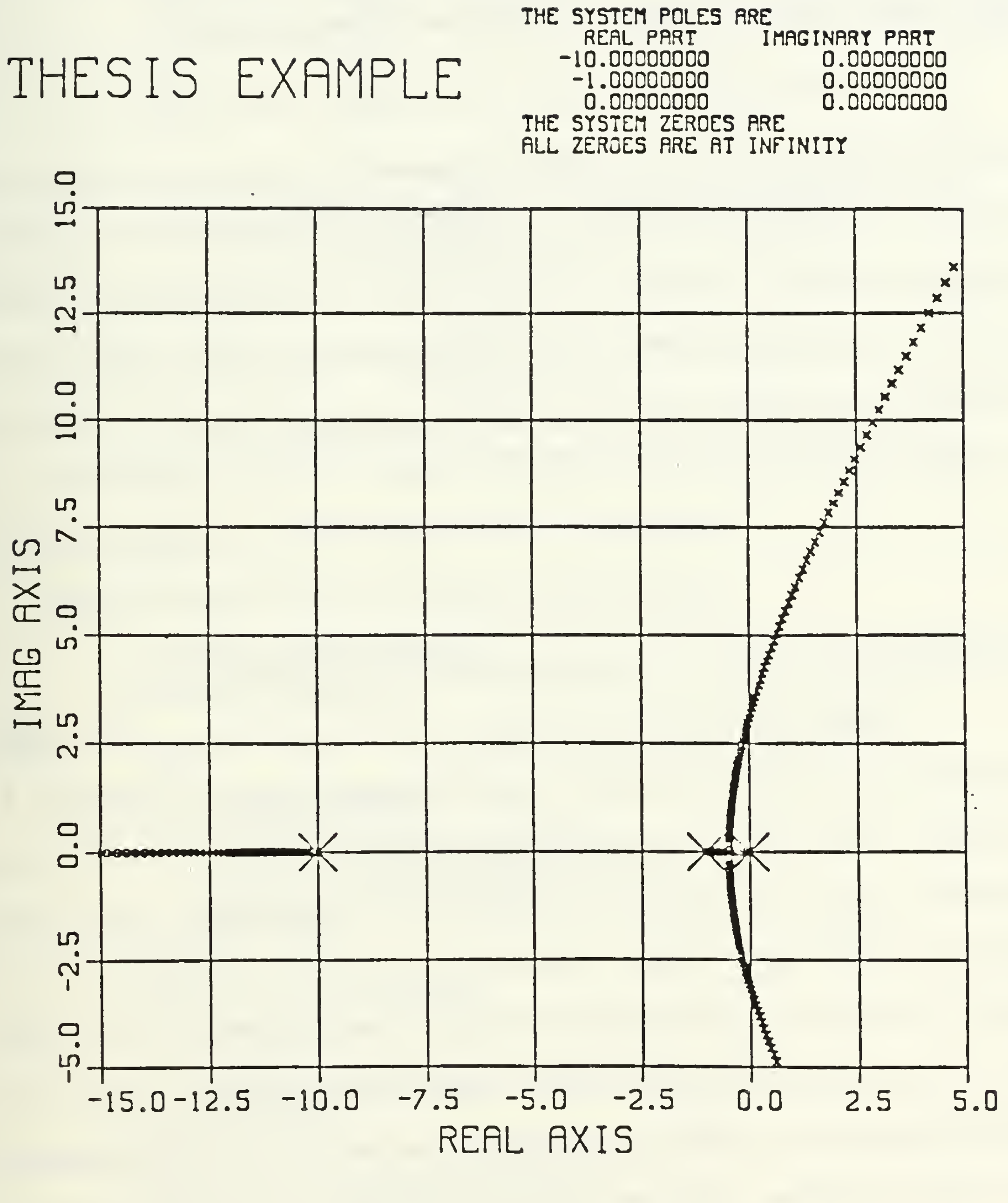


Figure 3.22 Simple System Root Locus

To demonstrate the root locus family capability of the program, answer the above question with 1 in order to analyze the following system.

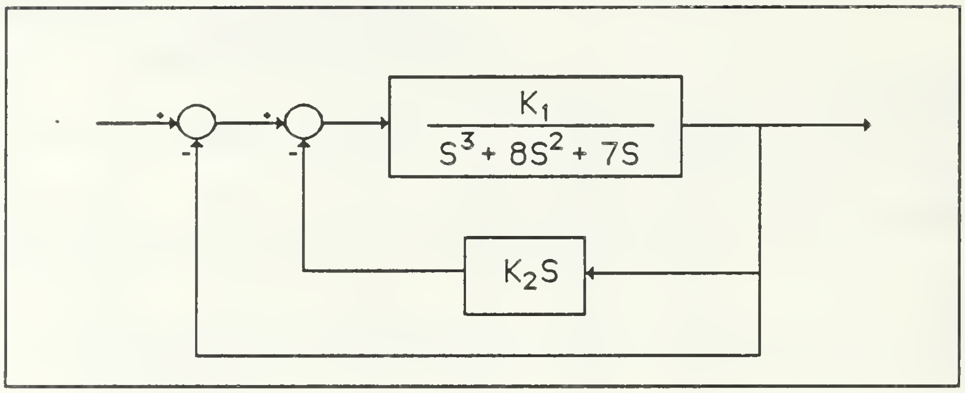


Figure 2.23 Velocity Feedback

The characteristic equation for the above system is as follows:

$$S^3 + 8S^2 + (7+K_1K_2)S + K_1 = 0. \quad (3.6)$$

Recall that the variable parameters must be related linearly. Therefore, it is necessary in this case to redefine the existing parameters in terms of new parameters that do not appear in a product of each other. This example presents a relatively easy case. Let K_1 of the program correspond to K_1 of equation 3.6 and let K_2 of the program correspond to the product of K_1K_2. When the root locus is drawn and the values of the two variable parameters in the redefined system are chosen, the user can then easily solve for the values of the original parameters. This will result in the following characteristic equation.

$$S^3 + 8S^2 + (7+K_2)S + K_1 = 0. \quad (3.7)$$

It will be important for the reader to remember that the K_2 in the revised equation is not the same K_2 appearing in the original equation.

The user would enter the following coefficients:

N	A	B	C
3	1	0	0
2	8	0	0
1	7	1	0
0	0	0	1

Recall that the A-coefficients correspond to the terms with no parameter, the B-coefficients correspond to the terms with the parameter K_2 and the C-coefficients correspond to the terms with the parameter K_1. To get the root locus of the system without any feedback, the user would select to set K_2 to zero for the initial run. Then in subsequent runs the user could select specific values for K_1 and vary K_2. For this example, assume the user has just entered the coefficients above.

The next prompt from the program is the following:

```
1. REVIEW COEFFICIENTS
2. START PROBLEM
```

Enter a 2 to start the problem. This leads to the following questions:

```
PRINT/PLOT OPTIONS
     1. PLOT ROOT LOCATIONS
     2. TABULATE ROOT LOCATIONS
     3. RETURN TO MAIN MENU
ENTER OPTION

CALCULATION OPTIONS
     1. SET K1 = 0
     2. SET K2 = 0
     3. RUN BOTH CASES
ENTER OPTION

ENTER STARTING VALUE FOR K1

ENTER NUMBER OF DECADES TO VARY K1
```

For this case, assume a plot is desired. The user would answer the first question with a 1. The initial root locus desired is obtained by setting K_2

to zero, so the second question is answered with a 2. The starting value for K_1 will be determined by what needs to be seen. In this example, assume the user enters .1 as the starting value and varies it for 4 decades to a value of 1000.

The user is then prompted for information concerning the plot.

```
PLOTTING DIMENSIONS
    XMAX=
    XMIN=
    YMAX=
    YMIN=

PLOT HEADING
    HOW MANY LINES OF HEADING WOULD YOU LIKE? (4 MAX)

    A MAX OF 32 CHARACTERS PER LINE IS ALLOWED.
    LINE 1

WOULD YOU LIKE TO SCALE THE GRAPH? (Y OR N)
```

For plotting dimensions in this example, enter 6, -10,10 and -6 in that order. Only one line of heading is used in this case. The heading is THESIS EXAMPLE. The graph was not scaled. At this point the plotting would begin and the user would see the initial root locus on the graphics screen.

After this plot is drawn, the program then returns the following prompt:

```
CALCULATION OPTIONS
    1. PICK A VALUE OF K1
    2. QUIT
ENTER OPTION
```

If K_2 had initially been set to zero, the user could now select specific values for that parameter. If both cases had been run, the user could select a value for either parameter. For this example, select a value of 50 for the parameter K_1. The following prompts are then displayed.

```
ENTER STARTING VALUE FOR K2

ENTER NUMBER OF DECADES TO VARY K2
```

Again, enter a starting value of .1 and vary for 4 decades. The system will then calculate the first set of roots and ask the user the following questions concerning the label for the curve corresponding to the chosen value of this parameter.

```
BY WHICH ROOT WOULD YOU LIKE TO PLACE THE LABEL?
     1.   -0.779528990D+01 +    - 0.1907348630D-05 J
     2.   -0.102357864D+00 +      0.2530540470D+01 J
     3.   -0.102357864D+00 +    - 0.2430540470D+01 J
     4. NO CURVE LABEL
ENTER OPTION

1. PLACE LABEL TO THE RIGHT OF THIS POINT
2. PLACE LABEL TO THE LEFT OF THIS POINT
     ENTER OPTION
```

Note in the above list of roots that the first root is actually a real root and the imaginary part should be zero, but due to the way the root calculation routines are written, a small value (approximately zero) is returned for the imaginary part. The user should be aware that all roots with nonzero imaginary parts will appear in pairs.

Placement of the label will be a matter of preference and experience. The user may choose to leave off the labels for this plot and insert them on a subsequent plot when the curve characteristics are known. Also, the curve for any value of the parameter could be drawn with no label and then immediately redrawn with a label. In most cases, the root of interest will be the rightmost root in the upper half of the plane. For this example, the root chosen was number 2 and since the subsequent roots move to the left, the label is placed to the right of the chosen point. The program will now draw the curve for the chosen value of the parameter.

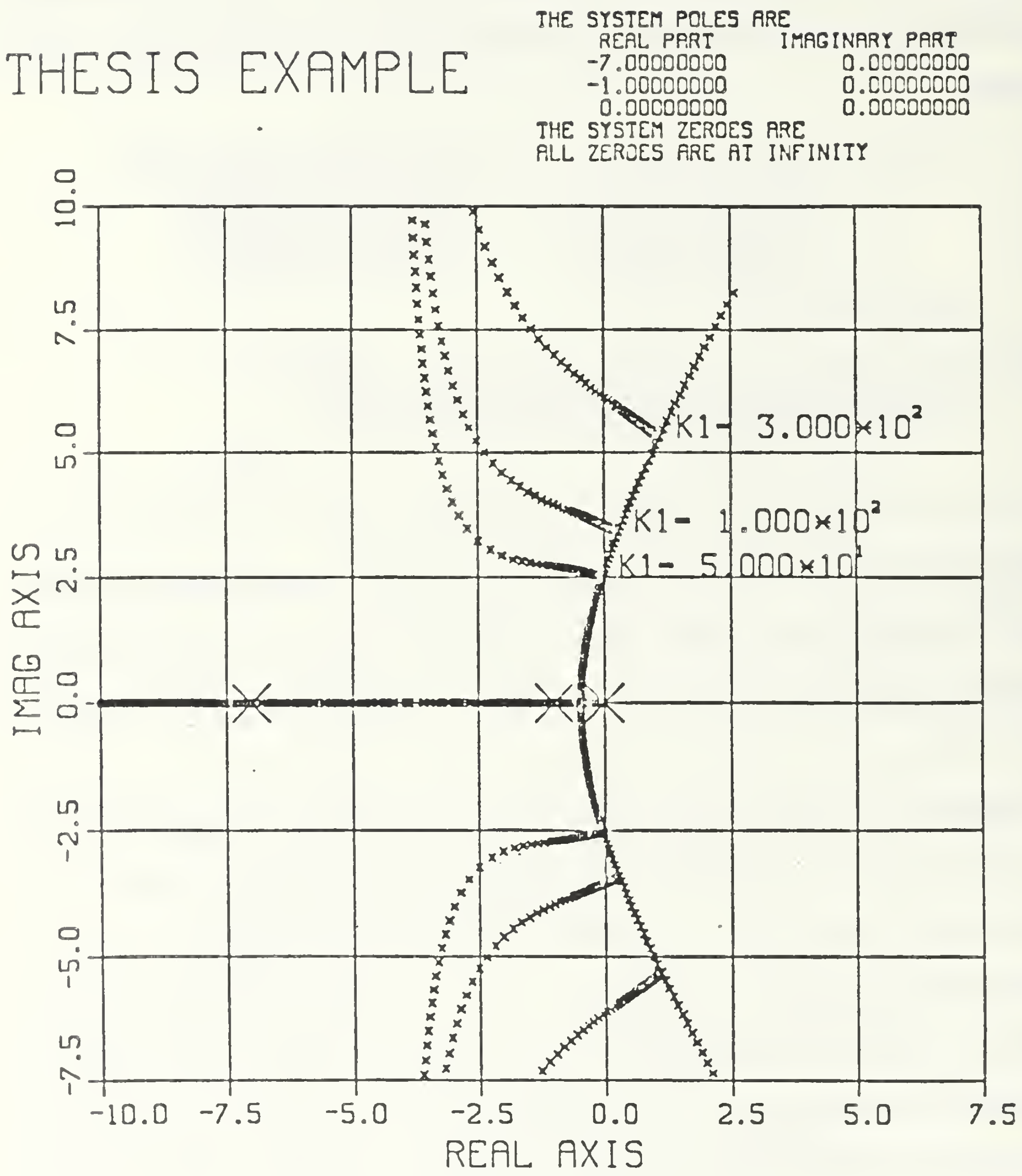


Figure 3.24 Example of Root Locus Family

When this curve is drawn, the program will again ask the user if he wants to select a value for K_1 or if he wants to quit this problem. Figure 3.24 shows the plot corresponding to this example. It resulted from choosing values of 50, 100 and 300.

To obtain the exact value of the parameters, the user may have to examine tabular output. To do this, the user would have selected TABULATION OF ROOT LOCATIONS early in the problem. The tabulated roots are placed in a file named SYSTEM ROOTS on the users disk. This file can become very large and the users disk may fill up very fast. It might be necessary to define temporary disk space to allow for tabulations of a large number of roots.

C. FREQUENCY RESPONSE ANALYSIS

This routine determines the frequency response of the system's open or closed loop transfer function. The output may be displayed graphically or in tabular form. Relative stability analysis may be performed by examining the system's open loop frequency response. The total system's steady state response to a sinusoidal input may be analyzed using closed loop Bode plots. The major routine involved in frequency response analysis is BODPLT.

1. Open Loop Analysis

The system's open loop frequency response is obtained by substituting jv for s in the open loop transfer function, GH(s). The value of GH(s) at $j\omega$ is

$$GH(j\omega) = R(\omega) + jX(\omega) \tag{3.8}$$

where

$R(\omega)$ = real part of $GH(j\omega)$

$X(\omega)$ = imaginary part of $GH(j\omega)$.

GH(s) may also be expressed as

$$GH(j\omega) = |GH(j\omega)| \exp\{j\varphi(j\omega)\} \quad (3.9)$$

where

$$|GH(j\omega)| = \{R^2(\omega) + X^2(\omega)\}^{1/2} \quad (3.10)$$

and

$$f(\omega) = \arctan\{ X(\omega)/R(\omega) \} \quad (3.11)$$

are the magnitude and phase, respectively, of GH(jω).

The coefficients of the open loop transfer function are known from TINPUT. The user need only input the open loop gain and the range of frequencies, v, over which the response is to be observed. Once all of the data has been input, the program evaluates GH(jv) at values of v within the frequency range selected by the user.

As an example, consider a previous system shown in Figure 3.21 and repeated below.

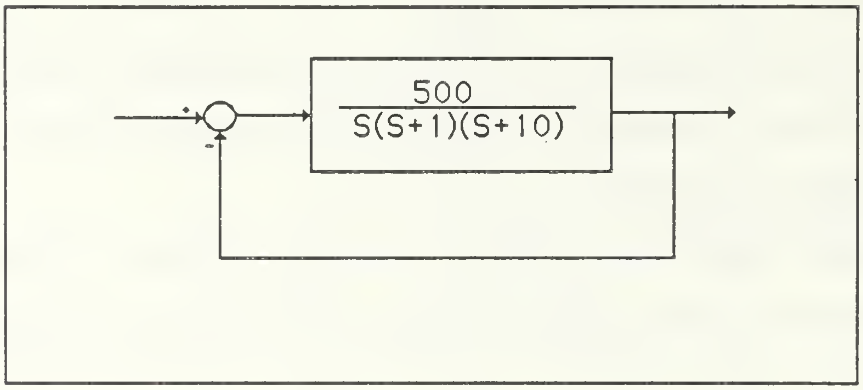


Figure 3.21 Single Loop System

By examining tabulated root locus data, it was determined that the open loop gain must be lowered to stabilize the system; use 20. The frequency response analysis can be started by selecting Option 4 from the main menu. The user is first presented with a screen that briefly describes this routine. It was originally written by Habib Ismail [Ref. 2].

The user is then asked for the necessary plotting information .

```
ENTER THE LOWER LIMIT OF THE FREQ AXIS AS FOLLOWS:

    IF LOWER LIMIT IS 10**(-02), ENTER -02
    IF LOWER LIMIT IS 10**(+11), ENTER +11

THE LOWER LIMIT IS 10**  -1       CORRECT?  (Y/N)
```

For the problem above the poles are 1 and 10. Therefore, if the user plots the response from v=.1 to v=100, everything of interest should be included. In this case the user would enter -01 for the above prompt after which the system will ask if that is what was intended. Next, the user is prompted for the number of decades to be plotted.

```
ENTER AS A SINGLE DIGIT INTEGER, THE NUMBER OF DECADES
OF FREQUENCY TO BE SPANNED.
MAXIMUM  :  9             MINIMUM  :  1
```

The user would enter 3 in order to plot to 100 from 0.1 and the program would respond will the message below.

```
3 DECADES OF FREQUENCY WILL BE SPANNED.    CORRECT?
(Y/N)
```

The next prompt is for the heading. This routine allows two lines of heading. For this particular routine, there are always two lines of heading. If the user has nothing to enter, he must still enter at least a blank for the heading line.

```
YOU MAY WRITE TWO LINES OF TEXT AS HEAD ON THE BODE
PLOT. ENTER FIRST LINE   (MAXIMUM OF 20 CHARACTERS)

ENTER SECOND LINE   (MAXIMUM OF 20 CHARACTERS)
```

For this example, enter THESIS EXAMPLE for the first line and OPEN LOOP BODE for the second line.

The program then asks if the user wants the open loop response or the closed loop response.

```
DO YOU WANT OPEN OR CLOSED LOOP RESPONSE?
     ENTER O FOR OPEN LOOP
     ENTER C FOR CLOSED LOOP
```

For this example, enter O to obtain the open loop plot. The final question allows the user to get tabular data if desired. In this case, assume that no tabular output is desired.

The Bode plot for this example is shown in Figure 3.25.

2. Closed Loop Analysis

The calculations involved in closed loop frequency response analysis are very similar to those for open loop frequency response analysis, as discussed in the previous subsection. In this case, however, s is replaced by jv in the system's closed loop transfer function.

Continuing with the above example, assume that the closed loop Bode is desired after the open loop Bode has been drawn. After the program completes the open loop drawing, it will give the user a chance to change the plot with the following question:

```
DO YOU WISH TO CHANGE THE PLOT JUST DRAWN?  (Y/N)
```

By answering yes (Y) to this question, the user is presented with the following change options:

```
ENTER CATEGORIES FOR CHANGE AS FOLLOWS:
     W   NUMBER OF DECADES TO BE SPANNED
     L   LOWER LIMIT OF FREQUENCY AXIS
     T   TITLES
     P   NO CHANGES
```

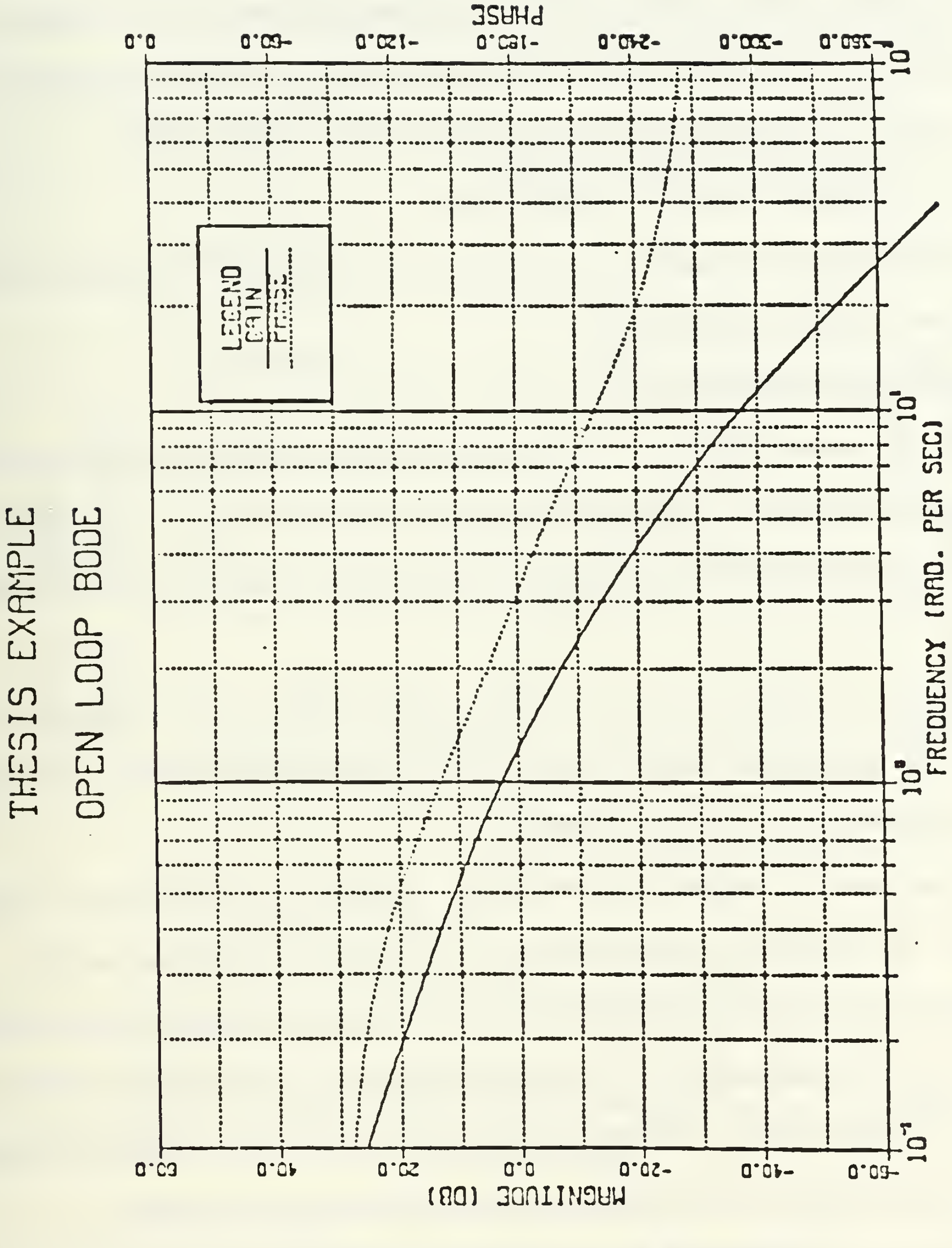


Figure 3.25 Open Loop Bode Plot

For the closed loop plot, the user will want to change the titles. He should enter a T. The program then prompts him for the new titles. In this case, assume he enters THESIS EXAMPLE for the first line and CLOSED LOOP BODE for the second line. The user is then returned to the change options shown above. He should now select P for no changes. This will result in the following prompt:

```
DO YOU WANT OPEN OR CLOSED LOOP RESPONSE?
    ENTER O FOR OPEN LOOP
    ENTER C FOR CLOSED LOOP
```

C will be entered for the closed loop Bode plot. The closed loop Bode plot for this problem is shown in Figure 3.26.

D. TIME RESPONSE ANALYSIS

The routines described in this section provide the user with a time history of a closed loop system's response to a user specified input. The user may choose to excite the system with either a unit impulse, a step function or a ramp function.

The major routines involved in evaluating system time response are TIMRSP, the parameter input routine, STIME and DVERK, the calculation routines, and TPLOT, the plotting routine.

The parameters which are input by TIMRES are the type of input (excitation), the closed loop parameters and the amount of time over which the response is to be observed. The closed loop parameters are values for the forward path, the feedback path gain and the type of feedback (positive or negative). The subroutine CLOOP calculates the coefficients of the closed loop transfer function based on these parameters and the open loop roots.

In order to calculate the time response, it is desirable to perform the inverse Laplace transform for the closed loop transfer function, thus obtaining

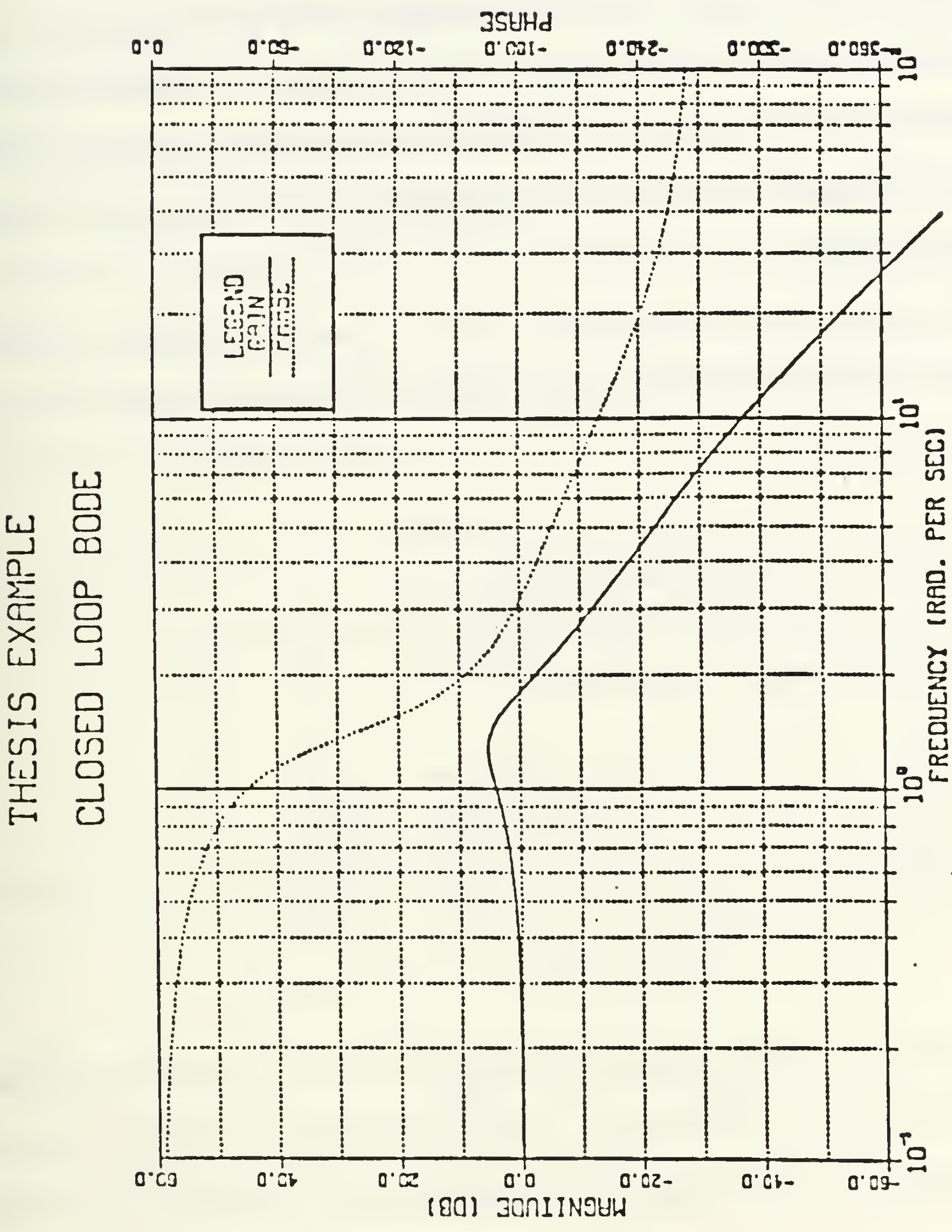


Figure 3.26 Closed Loop Bode Plot

an expression for the output as a function of time. However, numerical methods for obtaining inverse Laplace transforms sometimes break down and cannot be relied upon in all cases. An alternate method is to generate a set of state differential equations, and associated output equation, corresponding to the closed loop transfer function. A numerical method for solving that set of state differential equations can then be employed and the output, as a function of time, is calculated.

From elementary linear control theory, a transfer function, T(s), can be changed to an equivalent set of state equations as follows. Given, for example, the transfer function

$$T(s) = \frac{s^3 + b_2 s^2 + b_1 s + b_0}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0} \tag{3.12}$$

the corresponding matrix equations are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \delta \tag{3.13}$$

$$y = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{3.14}$$

where δ is the magnitude of the input at time t. Equation 3.13 is the phase variable canonical form or companion matrix form of the state equation and is based on the denominator coefficients of the transfer function. Equation 3.14 is the output equation and is based on the zeros' of the transfer function.

The subroutine STIME calculates the coefficients of the matrices in Equations 3.13 and 3.14. These coefficients are then used by the IMSL routine DVERK to create and solve the corresponding set of differential equations for values of time over the user specified time interval. Common arrays are established for the output amplitude and time values are made available for the plotting routine, TPLOT.

As an example, consider the system presented in the root locus section in Figure 3.23 and repeated below.

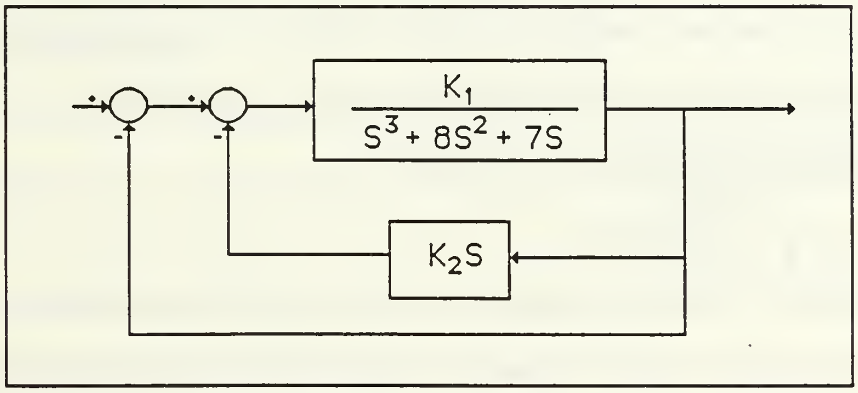
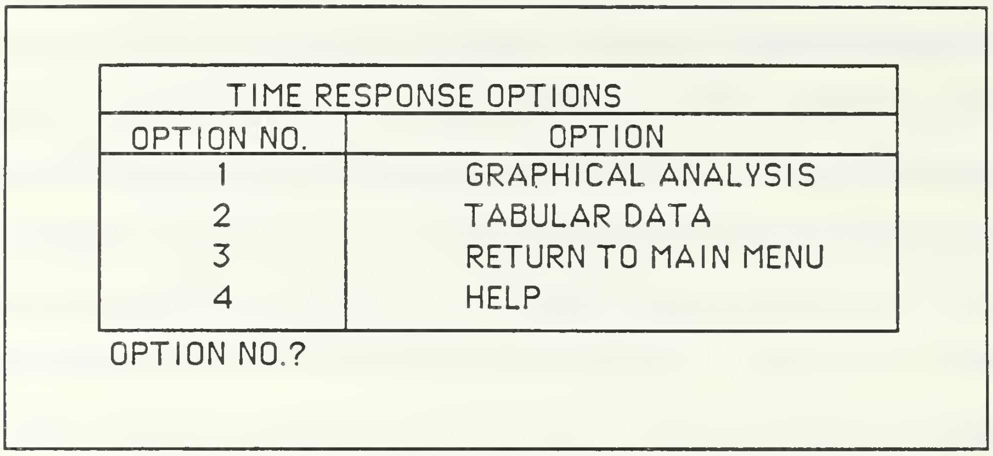


Figure 2.23 Velocity Feedback

Assume that through use of the root locus routine the values chosen for K_1 and K_2 were 50 and 10 respectively. Also, assume that the system has been entered as explained in the transfer function input section and that the user is now looking at the main menu.

To begin the time response analysis, the user must select Option 5 from the main menu. This results in the user being presented with the Time Response Options menu shown below in Figure 3.27.



TIME RESPONSE OPTIONS	
OPTION NO.	OPTION
1	GRAPHICAL ANALYSIS
2	TABULAR DATA
3	RETURN TO MAIN MENU
4	HELP

OPTION NO.?

FIGURE 3.27 Time Response Options Menu

In this example, the user wants to produce a plot, so he would select Option 1. He will then be presented with the following group of questions:

```
TIME RESPONSE PARAMETERS:
      1 = STEP
      2 = IMPULSE
      3 = RAMP
SELECT TYPE OF INPUT TO YOUR SYSTEM (1, 2 OR 3)

WHAT IS THE AMPLITUDE OF YOUR INPUT?(PER SEC FOR RAMP)

WHAT IS THE FORWARD PATH GAIN?

WHAT IS THE FEEDBACK PATH GAIN?

POSITIVE OR NEGATIVE FEEDBACK?  (P OR N)

FOR HOW MANY SECS  DO YOU WISH TO SEE THE RESPONSE?
```

For this problem, use the step input option. Select the amplitude of the input to be unity. The two questions regarding path gain allow the user to adjust the gain of a system and see the response without returning to one of the other analysis routines. For this problem, enter 1 for both questions. Negative feedback was used, so enter N. The answer to the last question will depend on how fast the response is. For this case, five seconds is sufficient.

At this point, the program will present the parameters just entered and give the user a chance to change his responses.

Next, the program asks for information concerning the heading.

```
PLOT HEADING
     HOW MANY LINES OF HEADING WOULD YOU LIKE?  (4 MAX)

A MAXIMUM OF 32 CHARACTERS PER LINE IS ALLOWED.
     LINE 1 =
```

Enter 1 for the number of lines and enter STEP RESPONSE for the plot heading. The resulting plot is shown in figure 3.28.

E. HELP ROUTINE

The word HELP appears as an option in many of the option menus presented to the user. If this option is chosen from any menu other than the main menu, several pages of information pertaining to the currently chosen routine appears on the screen. This information includes a brief description of the routine, a list of the required input data and very brief operating instructions for the routine. After the information has been read, the program returns the user to the menu from which HELP was selected.

If HELP is selected from the main menu, a brief description of the package appears on the screen followed by a menu of options. From this menu the user may choose to read any of the previously mentioned HELP files or print a hard-copy of the entire HELP routine. A sample hardcopy printout is contained in Appendix B.

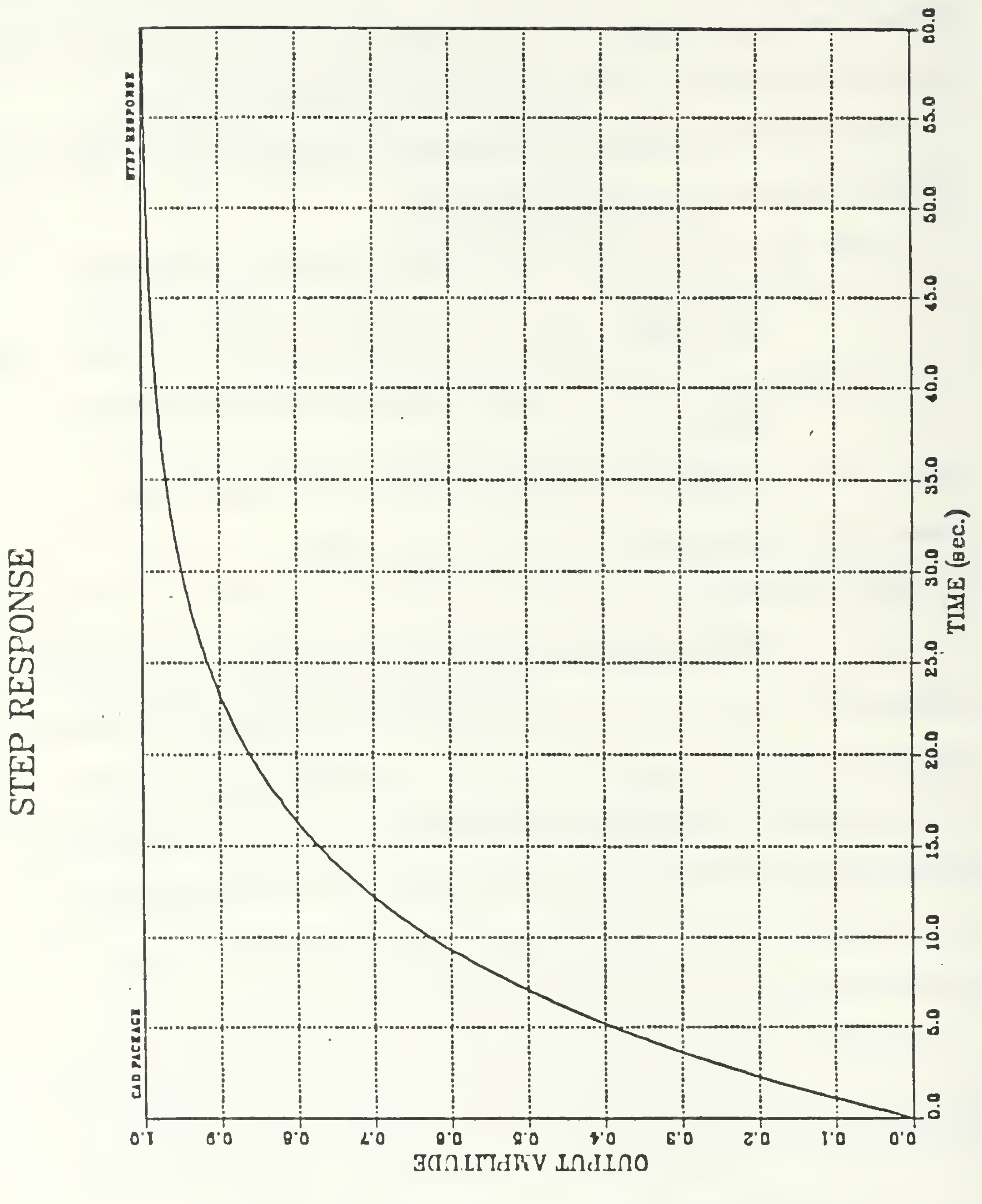


Figure 3.28 Time Response Example

IV. CONCLUSIONS

The primary objective of this thesis was to develop a comprehensive computer aid for the design and analysis of linear control systems. The program was aimed at reducing or eliminating the large number of calculations involved in the design of control systems. To this end, the program was to allow for the entry of transfer functions as they appear in a block diagram and to incorporate common control system techniques.

A. SUMMARY OF RESULTS

The result of the thesis is a computer design tool that incorporates previously written programs, interactive and batch. This package is a fully interactive design aid which includes a variety of methods for block diagram manipulation and control system analysis. The following list outlines the program's capabilities:

1) Interactive, menu driven program control.
2) Block diagram manipulations.
3) Root locus analysis including the capability for user defined variable parameters.
4) Frequency response analysis.
5) Time response analysis.
6) Mass storage data file routines which allow the user to store the entire block diagram plus the open or closed loop transfer functions for use during a subsequent run of the program.
7) Graphical output produced at the user's terminal allowing real time analysis of the system and true interactive design.
8) A HELP routine.

The program is currently available for use by all users of the Naval Postgraduate School's IBM 370 mainframe computer and is being used by students and faculty.

B. OTHER PROGRAMS

Some existing programs that aid in the design process of control systems are CONTROLS, TUTSIM, DSL and IODE.

As stated earlier, this package was written using CONTROLS as a model. In fact, it is hoped that this package will be integrated into CONTROLS. The main advantage of the package described in this thesis over CONTROLS is its ability to perform analysis on systems with two variable parameters. Operation of these two packages is very similar.

TUTSIM and DSL are both simulation languages that require the user to learn special codes in order to program a system for analysis. Neither package is interactive. This feature alone makes them less desirable. The design package of this thesis is faster and easier to use.

Another package used for analysis is IODE. This package performs time response analysis on a system entered by the user. This package is interactive but it is very slow. It is also difficult to recover from an error made in data entry. The user must have the system being analyzed in state-variable form. Also, since nonlinearities are allowed through the use of special functions, the user must be familiar with FORTRAN. Graphical output from this package is very crude compared to that of the program described in this thesis.

The user should find this package easier and faster to use than others that are currently available.

APPENDIX A

GETTING STARTED

To access this package the user must link to the disk on which it resides, disk 0171P, which has a read password of EWALD. It is advised that the user set up an EXEC file that contains the following commands:

```
CP LINK 0171P 191 AS 193 RR EWALD
ACCESS 193 C
```

This EXEC file can then be invoked by merely typing its name in the CMS environmentand pressing the ENTER key.

Alternatively, the user can issue the following commands while in the CMS environment:

```
LINKTO 0171P 191 AS 193 RR
      (The computer will then prompt for the read password.)
EWALD
ACCESS 193 C
```

This package, in combination with DISSPLA subroutines, takes nearly one megabyte of memory to operate. Therefore, the user should define his virtual memory as one megabyte prior to starting the program by issuing the command

```
DEFINE STORAGE 1M
```

from the CMS environment. If this is not done, the program will define the storage and leave the user in the CP environment along with instructions on how to start the program. When program execution has begun, the user will be asked to select an output device as shown below.

```
GRAPHICAL OUTPUT DEVICE OPTIONS:

     1. TEK 618
     2. TEK 4113 OR 4114
     3. IBM 3278
     4. DISSPLA METAFILE

PLEASE SELECT A GRAPHICAL OUTPUT DEVICE (1-4).
```

Since the package is meant to be an interactive design tool, it is best utilized when output is to a high speed, high resolution device such as the TEK 618 or TEK 4113/4114 with a high speed modem. If one wishes to input a large system, make changes to a system or obtain tabular data prior to performing graphical analysis, and do so without typing up a graphics terminal, then the IBM 3278 terminal may be selected. Graphical output to these terminals is possible, but it is in the form of an unlabelled, low resolution, line printer style plot. The general shape of curves can be observed, but detailed analysis of any kind is practically impossible.

Output to a DISSPLA metafile is available to those users who desire hardcopy, publication quality plots for use in theses, technical reports, etc. After the metafile has been formed the user is free to select any of a number of output devices available through the DISSPOP routine. The figures for this document were produced using this option. The program may be run from any terminal if this option is chosen since no graphical output comes to the terminal.

APPENDIX B

HELP ROUTINE PRINTOUT

*** THESIS PROJECT OF TERRENCE L. EWALD ***

FOR DR. GEORGE THALER

THIS IS AN INTERACTIVE COMPUTER AID FOR THE DESIGN AND ANALYSIS OF SINGLE INPUT / SINGLE OUTPUT CONTROL SYSTEMS. CLASSICAL CONTROL SYSTEM ANALYSIS METHODS ARE AVAILABLE IN THREE (3) MAIN AREAS: ROOT LOCUS, FREQUENCY RESPONSE, AND TIME RESPONSE.

LARGE, MULTI-LOOP FEEDBACK CONTROL SYSTEMS MAY BE INPUT AND ANALYZED. EACH LOOP IS SAVED SO THAT IT MAY BE RETURNED TO AND CHANGED. ESTABLISHED (PREVIOUSLY SAVED) SUBSYSTEMS MAY BE LOADED INTO ANY BLOCK OF ANOTHER CONTROL SYSTEM. IN THIS MANNER, FEEDBACK CONTROL SYSTEMS OF NEARLY ANY SHAPE OR SIZE MAY BE GENERATED AND ANALYZED.

THE PRIMARY OUTPUT OF THIS PACKAGE IS GRAPHICAL, THEREFORE, THE PROGRAM SHOULD BE RUN FROM A GRAPHICS TERMINAL (TEK 618, TEK 4113, OR TEK 4114).

*** TRANSFER FUNCTION INPUT ROUTINE ***

PURPOSE:

THIS ROUTINE INPUTS THE TRANSFER FUNCTION DATA REQUIRED FOR THE FREQUENCY RESPONSE AND TIME RESPONSE ANALYSIS ROUTINES.
THE INPUT DATA MUST BE AN OPEN-LOOP TRANSFER FUNCTION WHICH WILL BE INPUT AS A RATIO OF POLYNOMIALS DESCRIBED BY EITHER POLYNOMIAL COEFFICIENTS OR POLYNOMIAL ROOTS (FACTORED FORM).

REQUIRED DATA:

1. TRANSFER FUNCTION(S) WHOSE POLYNOMIALS ARE IN FACTORED OR COEFFICIENT FORM.
2. COEFFICIENTS MUST BE REAL NUMBERS ENTERED IN DESCENDING POWERS OF S.
3. IN FACTORED FORM, THE COMPLEX ROOTS OF THE POLYNOMIAL MUST BE ENTERED ALONG WITH A REAL GAIN CONSTANT (MULTIPLICATION FACTOR).

PROCEDURE:

1. COEFFICIENT FORM;

   GIVEN THE POLYNOMIAL

   ```
   2(S**2) + (5.656)S + 8
   ```

   THE COEFFICIENTS WOULD BE ENTERED IN THE FOLLOWING ORDER

   ```
   2
   5.656
   8
   ```

   A PROMPT WILL APPEAR BEFORE EACH ENTRY.

2. FACTORED FORM;

   THE SAME POLYNOMIAL MAY BE REPRESENTED AS

   ```
   2(S+1.414+1.414J)(S+1.414-1.414J)
   ```

   AGAIN, THE USER WILL BE PROMPTED. INPUT DATA WILL BE

   ```
   CONSTANT = 2
   ROOT 1:       REAL PART = -1.414
   ```

```
                IMAG PART = -1.414
ROOT 2:         REAL PART = -1.414
                IMAG PART = 1.414
```

3. THE FOLLOWING RESTRICTIONS APPLY TO THE TRANSFER FUNCTIONS;

   A. UP TO 20 BLOCKS MAY BE ENTERED INTO THE CONTROL LOOP.
   B. EACH BLOCK MAY BE UP TO 20TH ORDER.
   C. THE SIZE OF THE TOTAL SYSTEM MUST NOT EXCEED 100TH ORDER.
   D. THE NUMERATOR AND DENOMINATOR OF ANY GIVEN BLOCK MUST BE OF THE SAME FORM (COEFFICIENT OR FACTORED).
   E. IF ONLY ONE BLOCK IS ENTERED, IT IS ASSUMED TO BE IN THE FORWARD PATH FOR CLOSED-LOOP CALCULATIONS.
   F. UNITY FEEDBACK IS ASSUMED IF NO BLOCKS ARE ASSIGNED TO THE FEEDBACK PATH.

OPTIONS / COMMENTS

1. ANY TRANSFER FUNCTION MAY BE CHANGED FROM ANY OF THE ANALYSIS ROUTINES OR THE MAIN MENU.
2. IN ORDER TO CHANGE THE FORM (COEFFICIENT OR FACTORED) OF A BLOCK, THE USER CREATES BOTH A NEW NUMERATOR AND DENOMINATOR FOR THAT BLOCK.
3. AN ENTIRE CONTROL LOOP MAY BE LOADED FROM A FILE IF IT WAS SAVED DURING A PREVIOUS TERMINAL SESSION.
4. WHEN A SYSTEM OF TRANSFER FUNCTIONS IS SAVED, THE CLOSED LOOP TRANSFER FUNCTION FOR THAT SYSTEM IS ALSO SAVED.
5. THAT CLOSED LOOP TRANSFER FUNCTION CAN, THEN, BE LOADED INTO ANY BLOCK OF ANOTHER CONTROL SYSTEM. THIS ALLOWS THE USER DESIGN AND ANALYZE CONTROL SYSTEMS OF ALMOST ANY SIZE AND SHAPE.
6. REFER TO THE TRANSFER FUNCTION CHANGE ROUTINE HELP SECTION FOR MORE INFORMATION.

*** TRANSFER FUNCTION CHANGE ROUTINE ***

PURPOSE:

THIS ROUTINE PROVIDES THE USER WITH THE ABILITY TO EXAMINE AND / OR CHANGE TRANSFER FUNCTIONS IN THE CONTROL LOOP CURRENTLY BEING DESIGNED OR ANALYZED. THE ROUTINE ALSO ALLOWS THE USER TO EXPAND TO OUTER LOOP OR RETURN TO AND CHANGE INNER LOOPS.

REQUIRED DATA:

1. POLYNOMIAL(S) WHICH ARE TO BE CHANGED OR ADDED WILL BE INPUT IN THE SAME MANNER AS DESCRIBED IN THE TRANSFER FUNCTION INPUT SECTION.
2. IF EXPANDING TO OUTER LOOPS, THE INNER LOOP DATA MUST BE SAVED IF THE USER EVER INTENDS ON RETURNING TO AND CHANGING ANYTHING IN THE INNER LOOP. THE USER WILL BE PROMPTED FOR A FILE NAME DURING THIS PROCEDURE.
3. FORWARD AND FEEDBACK PATH GAINS AND THE TYPE OF FEEDBACK ARE REQUIRED WHEN LOOP ARE SAVED. THIS DATA IS NEEDED IN ORDER TO CALCULATE THE LOOP CLOSED LOOP TRANSFER FUNCTION.

PROCEDURE:

1. SELECT THE DESIRED OPERATION FROM THE TRANSFER FUNCTION CHANGE MENU.
2. THE USER WILL BE PROMPTED WHILE MAKING CHANGES TO ANY BLOCK IN THE CURRENT LOOP.
3. IF THE FORM OF AN EXISTING BLOCK IS TO BE CHANGED, THE USER MUST SELECT THE OPTION TO CREATE BOTH A NEW NUMERATOR AND DENOMINATOR FOR THAT BLOCK.
4. WHEN RE-ENTERING INNER LOOPS, SAVE THE CURRENT LOOP AND PROVIDE THE FILE NAME OF THE LOOP TO BE RE-EXAMINED. EXPANDING BACK OUT TO THE OUTER-MOST LOOP MUST BE DONE ONE LOOP AT A TIME. THIS WILL ALLOW THE PROGRAM TO AUTOMATICALLY UPDATE THE OUTER LOOPS WITH THE CHANGES MADE IN THE INNER LOOP.
5. SAVE EACH LOOP DURING THE EXPANSION PROCESS SO THAT THE LOOP FILES CONTAINS THE MOST UP-TO-DATE INFORMATION.

OPTIONS / COMMENTS:

1. EXISTING TRANSFER FUNCTIONS MAY BE CHANGED OR MOVED.
2. NEW TRANSFER FUNCTION(S) MAY BE ADDED TO THE CONTROL LOOP.

3. THE CONTROL SYSTEM MAY BE EXPANDED TO MULTIPLE LOOPS. THE INNER LOOP CLOSED LOOP TRANSFER FUNCTION IS AUTO-MATICALLY PLACED INTO A USER DEFINED BLOCK IN THE OUTER LOOP.
4. INNER LOOPS WHICH HAVE BEEN SAVED MAY BE LOADED AND CHANGED. ONCE CHANGES HAVE BEEN MADE, THE OUTER LOOPS CAN AUTOMATICALLY BE UPDATED IF EXPANSION IS DONE ONE LOOP AT A TIME.

*** ROOT LOCUS ROUTINE ***

PURPOSE:

THIS ROUTINE CALCULATES THE SYSTEM CLOSED LOOP CHARACTERISTIC EQUATION WHOSE COEFFICIENTS ARE ENTERED BY THE USER. THE ROOTS OF THE CHARACTERISTIC EQUATION ARE DETERMINED OVER A
USER DEFINED RANGE OF OPEN LOOP GAIN VALUES AND MAY BE PLOTTED AND / OR PRESENTED IN TABULAR FORM.

REQUIRED DATA:

1. COEFFICIENTS OF THE CHARACTERISTIC EQUATION
2. A RANGE OF OPEN LOOP GAIN VALUES.
3. PLOTTING DIMENSIONS. RECTANGULAR DIMENSIONS (IE. XMAX-XMIN=YMAX-YMIN) SHOULD BE USED IF ANGLES OR MAGNITUDES WILL BE MEASURED DIRECTLY FROM THE PLOT. IF UNSURE OF THE PLOTTING DIMENSIONS, SELECT TABULAR DATA FIRST. POLE AND ZERO LOCATIONS WILL BE PROVIDED SO THAT PLOTTING DIMENSIONS MAY BE DETERMINED.

PROCEDURE:

1. SELECT PLOTTED OR TABULAR DATA OPTION
2. INPUT GAIN AND PLOTTING PARAMETERS WHEN PROMPTED.
3. INPUT CHARACTERISTIC EQUATION COEFFICIENTS WHEN PROMPTED.
4. ONCE THE PLOTTING OR TABULATION IS COMPLETE, THE USER WILL BE GIVEN THE CHANCE TO PICK VALUES FOR VARIABLE PARAMETERS.

OPTIONS / COMMENTS:

1. 0-4 LINES OF HEADING MAY BE SPECIFIED FOR EACH PLOT.

*** FREQUENCY RESPONSE ROUTINE ***

PURPOSE:

THIS ROUTINE CALCULATES AND PLOTS THE FREQUENCY RESPONSE OF
AN OPEN OR CLOSED LOOP SYSTEM OVER A USER SPECIFIED RANGE OF FREQUENCIES. OPEN LOOP OR CLOSED LOOP RESPONSES MAY BE ANALYZED USING BODE PLOTS.

REQUIRED DATA:

1. A PREVIOUSLY ENTERED OPEN LOOP TRANSFER FUNCTION.
2. OPEN LOOP GAIN
3. RANGE OF FREQUENCIES.

PROCEDURE:

1. SELECT PLOTTED OR TABULATED DATA OPTION.
2. INPUT THE REQUIRED DATA (SEE ABOVE) WHEN PROMPTED.
3. ALL PLOTS ARE AUTOMATICALLY SCALED.

OPTIONS / COMMENTS:

1. ONLY TWO LINES OF HEADING MAY BE SPECIFIED FOR EACH PLOT.

*** TIME RESPONSE ***

PURPOSE:

THIS ROUTINE CALCULATES AND PLOTS THE TRANSIENT RESPONSE OF THE CLOSED LOOP SYSTEM TO A USER SPECIFIED INPUT. THE ROUTINE CALCULATES THE CLOSED LOOP TRANSFER FUNCTION FROM OPEN LOOP DATA AND INPUTS TO THIS ROUTINE.

REQUIRED DATA:

1. A PREVIOUSLY ENTERED OPEN LOOP TRANSFER FUNCTION.
2. TYPE AND MAGNITUDE OF THE INPUT SIGNAL.
3. FORWARD AND FEEDBACK PATH GAINS AND THE TYPE OF FEEDBACK.
4. THE NUMBER OF SECONDS OVER WHICH THE TIME RESPONSE IS TO BE OBSERVED.

PROCEDURE:

1. SELECT PLOTTED OR TABULATED DATA OPTION.
2. ENTER THE REQUIRED DATA (SEE ABOVE) WHEN PROMPTED.
3. TABULAR DATA MAY BE OBTAINED AFTER PLOTTING.

OPTIONS / COMMENTS:

1. TRANSIENT RESPONSE MAY BE OBTAINED FOR IMPULSE, STEP, OR RAMP INPUTS.
2. 0-4 LINES OF HEADING MAY BE SPECIFIED FOR EACH PLOT.

APPENDIX C

DISSPLA AND GRAPHICAL OUTPUT DEVICES

The main purpose of this package is to provide the user with graphical systems analysis tools such as root locus plots, frequency response plots and time response plots. It takes user specified parameters and performs calculations which result in data which is to be plotted. The actual plotting, however, is done by a library of graphics subroutines called DISSPLA (Display Integrated Software System and Plotting Language). DISSPLA is a web of FORTRAN subroutines to which plotting parameters are passed and which, in turn, provide the instructions necessary to produce high quality plots, graphs, charts or other specialized graphics.

This package passes data which is to be plotted to one of its plotting routines. These routines, then, pass the parameters to DISSPLA by calling a variety of subroutines in the proper sequence to produce the desired plot. By examining one of the plotting routines, the reader can see that, in some cases, no more than 30 instructions are required to produce a high quality plot with curve smoothing, plotting surface grid, labelled axes and headings.

In general, DISSPLA subroutine names are indicative of their function. For example, the subroutine CURVE passes to DISSPLA the array names containing the data which defines a curve. XNAME and YNAME pass the x and y axis labels, respectively. GRAF, XLOG, and POLAR define the dimensions for rectangular, semi-log and polar surfaces, respectively. Most of DISSPLA's subroutines follow this pattern, thus, making the graphics routines very easy to implement. Even if the user is unfamiliar with DISSPLA, he could examine one of the plotting routines and follow the method used to produce a plot.

In general, the program controls the size, shape, scaling and nature of information presented on all plots. Thus, the user is faced with far fewer questions from the program, but because all plotting decisions are made by the program, he has little control over the appearance of the plot. The only parameter controlled by the user is the plot heading. DISSPLA allows up to four lines of heading per plotted page. This program accommodates this capability, restricting the length of each line to 32 characters.

DISSPLA produces plots on a number of output devices. The user of this package is given four output device options; TEK 618, TEK 4113/4114, IBM 3278 and DISSPLA METAFILE. Since this is meant to be an interactive design tool, it is best utilized when output is to a high speed, high resolution device such as TEK 618 or TEK 4113/4114, using a high speed modem.

Output to the IBM 3278 is in the form of a low resolution, unlabelled, line printer style plot. The general shape of curves can be observed, but real analysis of these plots is practically impossible. However, this option may be useful if the user wishes to input the transfer functions for a large system, make changes to a system or obtain tabular output, and do so without tieing up a graphics terminal.

The option to output to a METAFILE is available to those users who wish to create publication quality plots using a postprocessor and high resolution plotting device such as the VERSATEC. The plots, in this case, are sized so as to fit comfortably on an 8.5 by 11 inch sheet of paper. All figures in this thesis were produced using the METAFILE option. This option may not be used for analysis, however, since output does not appear at the terminal. Separate terminal sessions are required; the first to analyze the system and the second to produce plots based on parameters obtained in the analysis section.s

APPENDIX D

PROGRAM LOCATION AND FORMAT

All blocks containing portions of this package reside on disk 0171P which belongs to Professor Thaler. Any user may access this disk by linking to the above user number and entering EWALD as the read password. This will allow any user interested to copy the package to his own disk and make any desired changes. These changes will not be reflected on disk 0171P. Anyone desiring to make changes to the original package must obtain permission from Professor Thaler.

The three files containing the FORTRAN code for this package are P2 FORTRAN, CADPRG FORTRAN and TIMRSP FORTRAN. These files have been packed using the PACK ON option while in the XEDIT mode. Packing a FORTRAN file saves the user valuable disk space. If the program is changed and the user plans to compile the file in order to create a TEXT file, the user must use the PACK OFF option while in XEDIT before filing the program.

In order to save disk space and to increase execution speed, the executable code has been put into a MODULE. When a program is normally loaded and executed, the system forms a MODULE that contains all the information needed for execution. By creating this MODULE, the step of loading the three text files and creating a map of all entry points has been deleted. Due to this, the user does not need to keep the TEXT files after the MODULE has been created. The FORTRAN files are not needed either except for future revisions.

An EXEC file named MAKE EXEC has been placed on disk 0171P that will create a MODULE. For more information on creating MODULES, use the HELP facility included on the IBM 370 system or consult the advisors at the computer center.

It should be noted that when a user puts a routine on his own disk and makes changes, these changes will not be reflected in subsequent runs of the program due to the MODULE form being used. The only way a user can makes changes that will be reflected in subsequent runs is to recompile all FORTRANS files and create a new MODULE.

APPENDIX E
THESIS EXAMPLES

A few examples are included in this appendix in order to demonstrate the use of the program, the appearance of a terminal session and the capabilities of the analysis routines. The first example is the most complete in that it includes all user and program screen entries. This example will be the most helpful to a first time user. The subsequent examples will explain what the user enters in order to obtain the desired output but the program prompts will not be included.

A. SINGLE LOOP SYSTEM

This first example is meant to show how one would obtain the simple root locus of a given transfer function and subsequently obtain the frequency and time responses.

The transfer function of this example is

$$\frac{1024(s+10)}{S(S+8)(S+16)(S+80)}$$

which is the only block in a single loop system with unity feedback.

Start by typing the name of the EXEC (EWALD) which will invoke the program. The first screen to appear gives a brief explanation of the project.

```
*** THESIS PROJECT OF TERRENCE L. EWALD ***
              FOR DR. THALER

THIS IS A FULLY INTERACTIVE COMPUTER AID FOR THE
DESIGN AND ANALYSIS OF SISO CONTROL SYSTEMS. ALL
OPTIONS ARE PRESENTED VIA OPTION MENUS AND YOU WILL BE
PROMPTED FOR ALL INPUT DATA. IF YOU ARE UNSURE OF THE
PURPOSE OF AN OPTION OR THE REQUIRED INPUT DATA FOR AN
OPTION, SELECT THE "HELP" OPTION FOR AN EXPLANATION. FOR A
DETAILED OUTLINE OF OPERATING PROCEDURES, REFER TO THE
USERS MANUAL. THE OPTION TO EXIT THE PROGRAM IS
AVAILABLE THROUGH THE MAJOR OPTION MENUS. AS A LAST
RESORT, THE PROGRAM CAN BE KILLED BY PRESSING "ENTER"
TWICE IN RESPONSE TO ANY PROMPT. ANALYSIS ROUTINES
INCLUDED ARE ROOT LOCUS, FREQUENCY RESPONSE, AND
TIME RESPONSE. TWO ROUTINES ARE ALSO INCLUDED TO
CALCULATE THE ROOTS OF THE CHARACTERISTIC EQUATION
 OF THE EXISTING TRANSFER OR OF A USER SUPPLIED
POLYNOMIAL.

          CLEAR SCREEN TO CONTINUE
```

The next screen gives the user a choice of output devices. The TEK618 dual screen system is recommended since it allows the user to make multiple runs and see the graphical results. The DISSPLA METAFILE option allows the user to set up a file of graphical information that can later be plotted on a variety of devices through the use of DISSPOP. In this example, Option 1 is used.

```
GRAPHICAL OUTPUT DEVICE OPTIONS:
        1. TEK 618
        2. TEK  4113/4114
        3. IBM 3278
        4. DISSPLA METAFILE
PLEASE SELECT A GRAPHICAL OUTPUT DEVICE (1-4).
```

Next, the main option screen is shown.

```
+---------------------------------------------------------------+
|                         MAIN   MENU                           |
+---------------+-----------------------------------------------+
|  OPTION  NO.  |               OPTION                          |
+---------------+-----------------------------------------------+
|       1       |    INPUT TRANSFER FUNCTION(S)                 |
|       2       |    ROOT LOCUS ANALYSIS                        |
|       3       |    COEFFICIENT FORM OF TRANSFER FUNCTION      |
|       4       |    BODE ANALYSIS                              |
|       5       |    TIME RESPONSE                              |
|       6       |    CHANGE BLOCK DIAGRAM                       |
|       7       |    SAVE THIS PROBLEM                          |
|       8       |    START A NEW PROBLEM                        |
|       9       |    HELP                                       |
|      10       |    EXIT PROGRAM                               |
+---------------+-----------------------------------------------+
      OPTION  NO?
```

To enter the transfer function data, choose option 1. This brings up the transfer function option screen.

```
+-----------------------------------------------------------+
|            TRANSFER  FUNCTION  INPUT  OPTIONS             |
+--------------+--------------------------------------------+
| OPTION  NO.  |                 OPTION                     |
+--------------+--------------------------------------------+
|       1      |  ENTER XFER FUNCTION(S) FROM CONSOLE       |
|       2      |  LOAD TRANSFER FUNCTION(S) FROM A FILE     |
|       3      |  RETURN TO MAIN MENU                       |
|       4      |  EXIT                                      |
+--------------+--------------------------------------------+
 OPTION NO.?
```

Since the data has not been saved, choose Option 1 to enter data from the console. There is only one loop and one block in this system, so enter a "1" in response to each of the next questions presented on screen.

```
WHICH LOOP ARE YOU PREPARING TO INPUT?

HOW MANY BLOCKS ARE IN YOUR OPEN-LOOP?
```

The program then prompts with the following question.

```
TRANSFER FUNCTION NO. 1

DO YOU WISH TO INPUT THE TRANSFER FUNCTION FOR THIS
BLOCK FROM A DATA FILE OR FROM THE CONSOLE? (D OR C)
```

Since this is the first time the system has been entered, select Option C in order to enter data from the console.

```
WHAT IS THE ORDER OF THE NUMERATOR POLYNOMIAL?
```

Enter 1 for the order of the numerator.

```
WHAT IS THE ORDER OF THE DENOMINATOR?
```

Enter 4 for the order of the denominator.

```
IS THIS TRANSFER FUNCTION IN FACTORED OR COEFFICIENT
        FORM?   (F OR C)
```

This system is in factored form so enter F to the above question.

The program then asks the user to enter the gain coefficients and roots of the numerator and denominator.

```
WHAT IS THE NUMERATOR GAIN CONSTANT?
```

The gain coefficient is 1024 in this case.

```
WHAT ARE THE ROOTS OF NUMERATOR?


ROOT NO. 1
     REAL PART =

     IMAGINARY PART =
```

When entering the roots, be sure to enter the sign of the number correctly. For example, the term (S+8) means a real root at -8. For this example, enter -10 for the real part and 0 for the imaginary part of the numerator root. The program shows the user what has been entered and allows changes to be made if necessary.

```
+--------------------------------------------------------------+
|                      SUMMARY  OF  ROOTS                      |
+--------------------------------------------------------------+
|                        LOOP NUMBER 1                         |
|                        BLOCK NUMBER 1                        |
+------------+-------------------------------------------------+
| ITEM NO.   |                NUMERATOR  ROOTS                 |
+------------+-------------------------------------------------+
|     1      |        -10.0000000               0.0000000 J    |
|     2      |  CONSTANT  =       1024.000000                  |
+------------+-------------------------------------------------+
  ANY CHANGES TO THESE PARAMETERS?
```

The same procedure is followed for the denominator.

```
WHAT IS THE DENOMINATOR GAIN CONSTANT?
```

Enter a 1 for the denominator gain of this example.

```
WHAT ARE THE ROOTS OF THE DENOMINATOR?

ROOT NO. 1
     REAL PART =
```

Enter 0 for the S term.

```
     IMAGINARY PART =
```

Enter 0.

```
          ROOT NO. 2
               REAL PART =
```

Enter -8 for the term (S+8).

```
               IMAGINARY PART =
```

Enter 0.

```
          ROOT NO. 3
               REAL PART =
```

Enter -16 for the term (S+16).

```
               IMAGINARY PART =
```

Enter 0.

```
          ROOT NO. 4
               REAL PART =
```

Enter -80 for the term (S+80).

```
          IMAGINARY PART =
```

Enter 0.

The summary of the entries just made is shown next.

```
+-----------------------------------------------------+
|                  SUMMARY OF ROOTS                   |
+-----------------------------------------------------+
|                   LOOP NUMBER 1                     |
|                   BLOCK NUMBER 1                    |
+-----------+-----------------------------------------+
| ITEM NO.  |          DENOMINATOR ROOTS              |
+-----------+-----------------------------------------+
|     1     |       0.0000000        0.0000000 J      |
|     2     |      -8.0000000        0.0000000 J      |
|     3     |     -16.0000000        0.0000000 J      |
|     4     |     -80.0000000        0.0000000 J      |
|     5     |    CONSTANT =    1.0000000              |
+-----------+-----------------------------------------+
 ANY CHANGES TO THESE PARAMETERS?
```

Enter Y if all values shown are correct. After the entire transfer has been entered, the user is given one last chance to change the transfer function.

```
WANT TO MAKE ANY CHANGES TO YOU TRAN. FUNCTION(S)?
(Y OR N)
```

Enter N to continue with problem. Enter Y to review entries or make any changes. After entering N, the user is returned to the main menu.

MAIN MENU	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	COEFFICIENT FORM OF TRANSFER FUNCTION
4	BODE ANALYSIS
5	TIME RESPONSE
6	CHANGE BLOCK DIAGRAM
7	SAVE THIS PROBLEM
8	START A NEW PROBLEM
9	HELP
10	EXIT PROGRAM

```
OPTION NO.?
```

In order to get the root locus, the user must have the coefficient form of the transfer function. To have the program perform the calculations, choose option 3.

OPEN-LOOP TRANSFER FUNCTION COEFFICIENTS		
DEGREE	NUMERATOR	DENOMINATOR
4		0.1000000E+01
3		0.1040000E+03
2		0.2048000E+04
1	0.10240000E+04	0.1024000E+05
0	0.10240000E+05	0.0000000E+00

These coefficients will be used as the coefficients for the simple root locus case as in Equation 3.4. After clearing the screen the user is again returned to the main menu.

MAIN MENU	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	COEFFICIENT FORM OF TRANSFER FUNCTION
4	BODE ANALYSIS
5	TIME RESPONSE
6	CHANGE BLOCK DIAGRAM
7	SAVE THIS PROBLEM
8	START A NEW PROBLEM
9	HELP
10	EXIT PROGRAM

To start the root locus analysis, select Option 2. The user is first prompted to enter how the system coefficients will be entered.

```
1. ENTER COEFFICIENTS FROM CONSOLE
2. ENTER COEFFICIENTS FROM DATAFILE

     (ENTER OPTION)
```

Enter 1 in order to enter the coefficients from the keyboard.

```
ENTER THE ORDER OF YOUR SYSTEM (1-99)
```

Enter a 4 since this problem is of fourth order.

```
THE ORDER OF YOUR SYSTEM IS 4
IS THAT CORRECT?   (Y/N)
```

Enter a Y. The user is now ready to enter the coefficients. Remember that the A-coefficients correspond to the terms with no parameters.

```
ENTER THE A-COEFFICIENTS
A(4) =

A(3) =

A(2) =

A(1) =

A(0) =
```

After the A-coefficients have been entered, they will be presented to the user and changes may be made if needed.

```
N                A(N)
4            1.000000
3          104.000000
2         2048.000000
1        10240.000000
0            0.000000
ARE THESE CORRECT?   (Y/N)
```

The same process is followed for the B-coefficients, which coerrespond to the terms with the K_2 parameter, and for the C-coefficients, which correspond to the terms with the K_1 parameter. For this example, let the K_2 parameter correspond to the variable gain used for the system root locus. Therefore the numerator of the system of interest corresponds to the B-coefficients. The C-coefficients are not used but enter zeros to make things less confusing.

```
ENTER THE B-COEFFICIENTS

B(4) =

B(3) =
B(2) =

B(1) =

B(0) =

N                  B(N)
4                  0.000000
3                  0.000000
2                  0.000000
1               1024.000000
0              10240.000000
ARE THESE CORRECT?  (Y.N)

ENTER THE C-COEFFICIENTS
C(4) =

C(3) =

C(2) =

C(1) =

C(0) =

N                  C(N)
4                  0.000000
3                  0.000000
2                  0.000000
1                  0.000000
0                  0.000000
ARE THESE CORRECT?   (Y/N)
```

After the coefficients have all been entered, the user is given one last opportunity to change things if necessary. If not, he is ready to start the problem.

```
1. REVIEW COEFFICIENTS
2. START PROBLEM

     (ENTER OPTION)
```

Enter 2 to start the root locus analysis.

```
PRINT/PLOT OPTIONS
      1. PLOT ROOT LOCATIONS
      2. TABULATE ROOT LOCATIONS
      3. RETURN TO MAIN MENU
ENTER OPTION
```

Enter 1 to draw a plot.

```
CALCULATION OPTIONS
      1. SET K1=0
      2. SET K2=0
      3. RUN BOTH CASES
ENTER OPTION
```

The B-coefficients are being used for this simple root locus so select Option 1 to let K_1 be set to zero.

```
ENTER STARTING VALUE FOR K2
```

Start the root locus at K_2=.1.

```
ENTER NUMBER OF DECADES TO VARY K2
```

Enter 5 to vary the values of the gain over five decades from .1 to 10,000.

```
PLOTTING DIMENSIONS
      XMAX =
```

Set the right limit on the real axis by entering 10.

```
      XMIN =
```

Enter -90 for the left limit.

```
      YMAX =
```

Set the upper limit on the imaginary axis by entering 90.

```
      YMIN =
```

Enter -10 for the lower limit.

```
XMAX = 10
XMIN = -90
YMAX = 90
YMIN = -10
```

```
            ARE THESE CORRECT?  (Y/N)
```

These values are correct so answer yes (Y).

```
            PLOT HEADING
                  HOW MANY LINES OF HEADING WOULD YOU LIKE?
                  (4 MAX)
```

Only one line of heading is used in this example.

```
            A MAX OF 32 CHARACTERS PER LINE IS ALLOWED.
                  LINE  1
```

THESIS EXAMPLE is used as the title.

```
            WOULD YOU LIKE TO SCALE THE GRAPH?   (Y/N)
            DEFAULT IS 7.5 BY 7.5 INCHES
```

The user may scale the graph to make it fit in areas smaller than 8" by 11". Accept the default value for this example. The resulting graph is shown in Figure E.1.

```
            CALCULATION OPTIONS
                  1. PICK A VALUE FOR K2
                  2. QUIT
            ENTER OPTION
```

If a root locus family were being drawn, the user could now select specific values for the variable initially used as the system gain and vary the parameter initially set to zero. For this example, select Option 2 to quit this problem.

```
            WOULD YOU LIKE TO SAVE THE COEFFICIENTS FOR THIS
                  PROBLEM?  (Y/N)
```

At this point, the coefficients may be saved for use at some later date. A response of N was entered in this example.

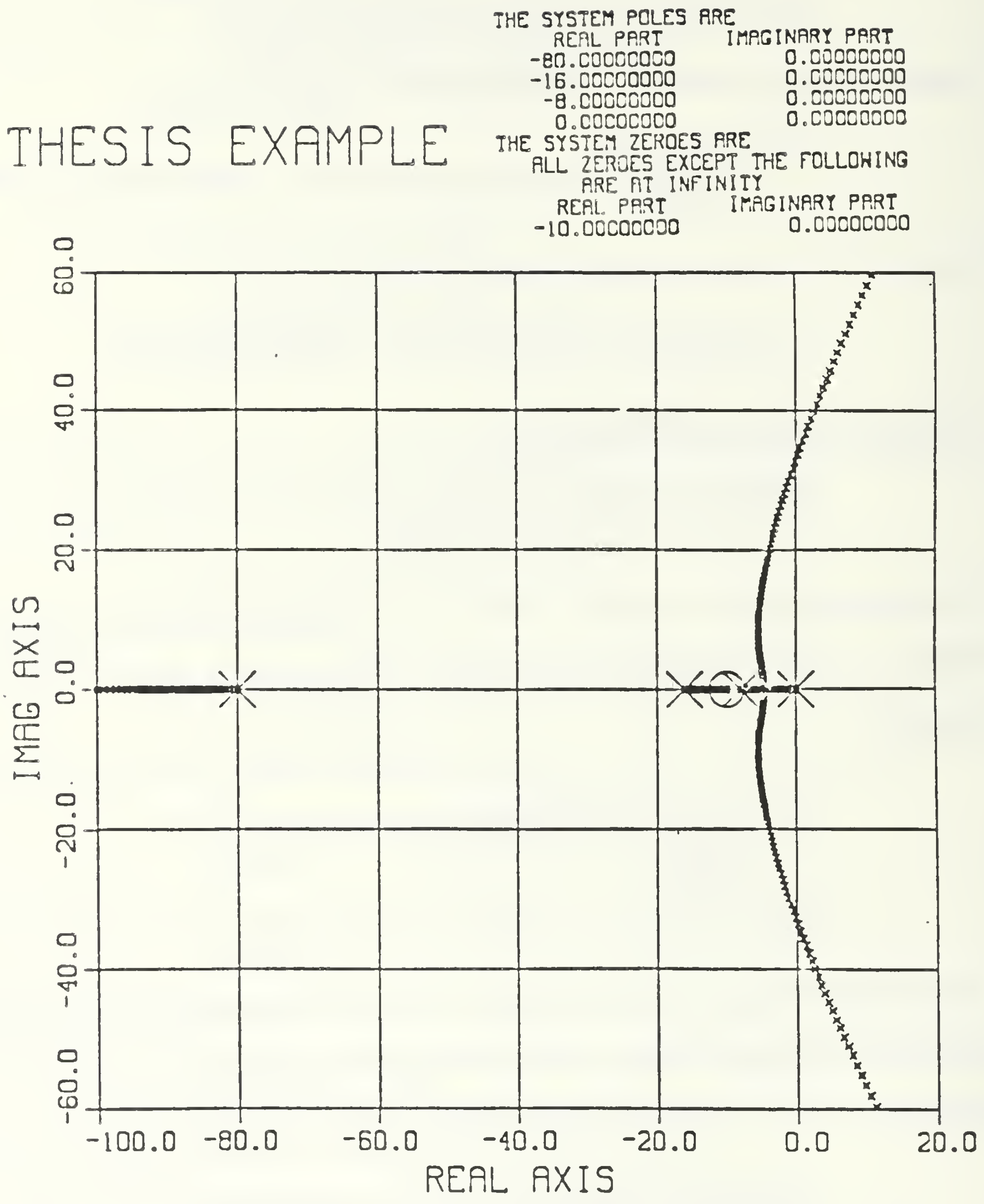


Figure E.1 Simple System Root Locus

This results in the following prompt.

```
1. RUN ANOTHER PROBLEM
2. QUIT
      (ENTER OPTION)
```

The user could run another, run the same problem again in order to get tabulated data or simply quit and return to the main menu.

```
                    MAIN   MENU
  OPTION NO.            OPTION
      1          INPUT TRANSFER FUNCTION(S)
      2          ROOT LOCUS ANALYSIS
      3          COEFFICIENT FORM OF TRANSFER FUNCTION
      4          BODE ANALYSIS
      5          TIME RESPONSE
      6          CHANGE BLOCK DIAGRAM
      7          SAVE THIS PROBLEM
      8          START A NEW PROBLEM
      9          HELP
      10         EXIT PROGRAM

OPTION NO?
```

Assume that a Bode plot is also desired. The user would select Option 4 from the main menu. This sends him to the routine BODPLOT.

```
ENTER THE LOWER LIMIT OF THE FREQ AXIS AS FOLLOWS:

      FOR EXAMPLE

      IF LOWER LIMIT IS 10**(-02), ENTER -02
      IF LOWER LIMIT IS 10**(+11), ENTER +11
```

For this example -01 was entered.

```
ENTER AS A SINGLE DIGIT INTEGER, THE NUM OF DECADES
OF FREQUENCY TO BE SPANNED.
MAXIMUM  :  9            MINIMUM  :  1
```

In this case 3 was entered.

```
YOU MAY WRITE TWO LINES OF TEXT AS HEADING

ENTER FIRST LINE (MAXIMUM 20 CHARACTERS)
```

For this plot, THESIS EXAMPLE was entered.

```
ENTER SECOND LINE (MAXIMUM 20 CHARACTERS)
```

OPEN LOOP BODE was typed in by the user.

```
DO YOU WANT OPEN OR CLOSED LOOP RESPONSE?

ENTER   O   FOR OPEN LOOP
ENTER   C   FOR CLOSED LOOP
```

The open loop response was desired so O was entered.

```
DO YOU WANT TABULAR OUTPUT AT THE TERMINAL? (Y/N)
```

For illustration purposes, Y was entered. At this point, tabular data is displayed on the screen. This output is shown in Table E.1. After the output to the screen is complete, the program continues with the Bode plot. The resulting graph is shown in Figure E.2. After the plot is drawn, the user is given a chance to make changes to the plot which include changing the lower limit of the frequency axis, the number of decades spanned or the titles.

```
DO YOU WISH TO CHANGE THE PLOT JUST DRAWN?   (Y/N)
```

The closed loop response is desired. The program can be run again or the user can just go through the change menu. Enter Y.

```
ENTER CATEGORIES FOR CHANGES AS FOLLOWS:
W       NUMBER OF DECADES TO BE SPANNED
L       LOWER LIMIT OF FREQUENCY AXIS
T       TITLES
P       NO CHANGES
RETURN TO THE MENU TO CHANGE THE TRAN FUNCTION(S)
```

FREQ	MAGNITUDE	PHASE
0.100000E+00	0.199996E+02	-0.905718E+02
0.107977E+00	0.193328E+02	-0.906174E+02
0.116591E+00	0.186661E+02	-0.906668E+02
0.125892E+00	0.179993E+02	-0.907201E+02
0.135935E+00	0.173326E+02	-0.907776E+02
0.146780E+00	0.166658E+02	-0.908397E+02
0.158489E+00	0.159989E+02	-0.909068E+02
0.171133E+00	0.153321E+02	-0.909792E+02
0.184785E+00	0.146652E+02	-0.910574E+02
0.199526E+00	0.139983E+02	-0.911418E+02
0.215443E+00	0.133314E+02	-0.912330E+02
0.232630E+00	0.126644E+02	-0.913314E+02
0.251188E+00	0.119974E+02	-0.914377E+02
0.271227E+00	0.113302E+02	-0.915524E+02
0.292864E+00	0.106631E+02	-0.916762E+02
0.316227E+00	0.999579E+01	-0.918099E+02
0.341454E+00	0.932844E+01	-0.919543E+02
0.368694E+00	0.866097E+01	-0.921101E+02
0.398107E+00	0.799336E+01	-0.922783E+02
0.429865E+00	0.732559E+01	-0.924599E+02
0.464158E+00	0.665763E+01	-0.926559E+02
0.501187E+00	0.598947E+01	-0.928675E+02
0.541169E+00	0.532106E+01	-0.930958E+02
0.584341E+00	0.465236E+01	-0.933423E+02
0.630956E+00	0.398331E+01	-0.936082E+02
0.681291E+00	0.331390E+01	-0.938951E+02
0.735641E+00	0.264401E+01	-0.942046E+02
0.794327E+00	0.197359E+01	-0.945385E+02
0.857695E+00	0.130258E+01	-0.948987E+02
0.926117E+00	0.630841E+00	-0.952870E+02
0.999998E+00	-0.417236E-01	-0.957056E+02
0.107977E+01	-0.715219E+00	-0.961569E+02
0.116591E+01	-0.138988E+01	-0.966431E+02
0.125892E+01	-0.206579E+01	-0.971669E+02
0.135935E+01	-0.274324E+01	-0.977309E+02
0.146780E+01	-0.342238E+01	-0.983378E+02
0.158489E+01	-0.410353E+01	-0.989906E+02
0.171132E+01	-0.478699E+01	-0.996925E+02
0.184785E+01	-0.547309E+01	-0.100447E+03
0.199525E+01	-0.616222E+01	-0.101256E+03
0.215443E+01	-0.685485E+01	-0.102124E+03
0.232630E+01	-0.755146E+01	-0.103055E+03
0.251188E+01	-0.825262E+01	-0.104050E+03
0.271226E+01	-0.895891E+01	-0.105115E+03
0.292864E+01	-0.967106E+01	-0.106251E+03
0.316227E+01	-0.103898E+02	-0.107462E+03

Table E.1 Tabular Frequency Respnse Data

FREQ	MAGNITUDE	PHASE
0.341454E+01	-0.111158E+02	-0.108750E+03
0.368694E+01	-0.118501E+02	-0.110118E+03
0.398106E+01	-0.125936E+02	-0.111568E+03
0.429865E+01	-0.133471E+02	-0.113102E+03
0.464157E+01	-0.141116E+02	-0.114720E+03
0.501186E+01	-0.148882E+02	-0.116423E+03
0.541168E+01	-0.156777E+02	-0.118211E+03
0.584340E+01	-0.164813E+02	-0.120085E+03
0.630955E+01	-0.172999E+02	-0.122042E+03
0.681290E+01	-0.181343E+02	-0.124082E+03
0.735640E+01	-0.189855E+02	-0.126204E+03
0.794326E+01	-0.198545E+02	-0.128406E+03
0.857693E+01	-0.207419E+02	-0.130685E+03
0.926116E+01	-0.216487E+02	-0.133040E+03
0.999996E+01	-0.225757E+02	-0.135469E+03
0.107977E+02	-0.235237E+02	-0.137967E+03
0.116591E+02	-0.244934E+02	-0.140534E+03
0.125892E+02	-0.254858E+02	-0.143164E+03
0.135935E+02	-0.265014E+02	-0.145855E+03
0.146779E+02	-0.275409E+02	-0.148601E+03
0.158489E+02	-0.286049E+02	-0.151399E+03
0.171132E+02	-0.296940E+02	-0.154243E+03
0.184784E+02	-0.308084E+02	-0.157127E+03
0.199525E+02	-0.319485E+02	-0.160047E+03
0.215443E+02	-0.331144E+02	-0.162998E+03
0.232630E+02	-0.343061E+02	-0.165975E+03
0.251188E+02	-0.355238E+02	-0.168975E+03
0.271226E+02	-0.367672E+02	-0.171994E+03
0.292864E+02	-0.380365E+02	-0.175030E+03
0.316226E+02	-0.393316E+02	-0.178079E+03
0.341454E+02	-0.406526E+02	-0.181146E+03
0.368694E+02	-0.419997E+02	-0.184219E+03
0.398106E+02	-0.433731E+02	-0.187301E+03
0.429865E+02	-0.447731E+02	-0.190391E+03
0.464157E+02	-0.462002E+02	-0.193484E+03
0.501186E+02	-0.476548E+02	-0.196578E+03
0.541167E+02	-0.491374E+02	-0.199669E+03
0.584340E+02	-0.506486E+02	-0.202750E+03
0.630955E+02	-0.521885E+02	-0.205815E+03
0.681290E+02	-0.537576E+02	-0.208857E+03
0.735639E+02	-0.553560E+02	-0.211866E+03
0.794326E+02	-0.569835E+02	-0.214834E+03
0.857692E+02	-0.586398E+02	-0.217750E+03
0.926116E+02	-0.603245E+02	-0.220604E+03
0.999998E+02	-0.620367E+02	-0.223388E+03

Table E.1 (Continued)   Tabular Frequency Respnse Data

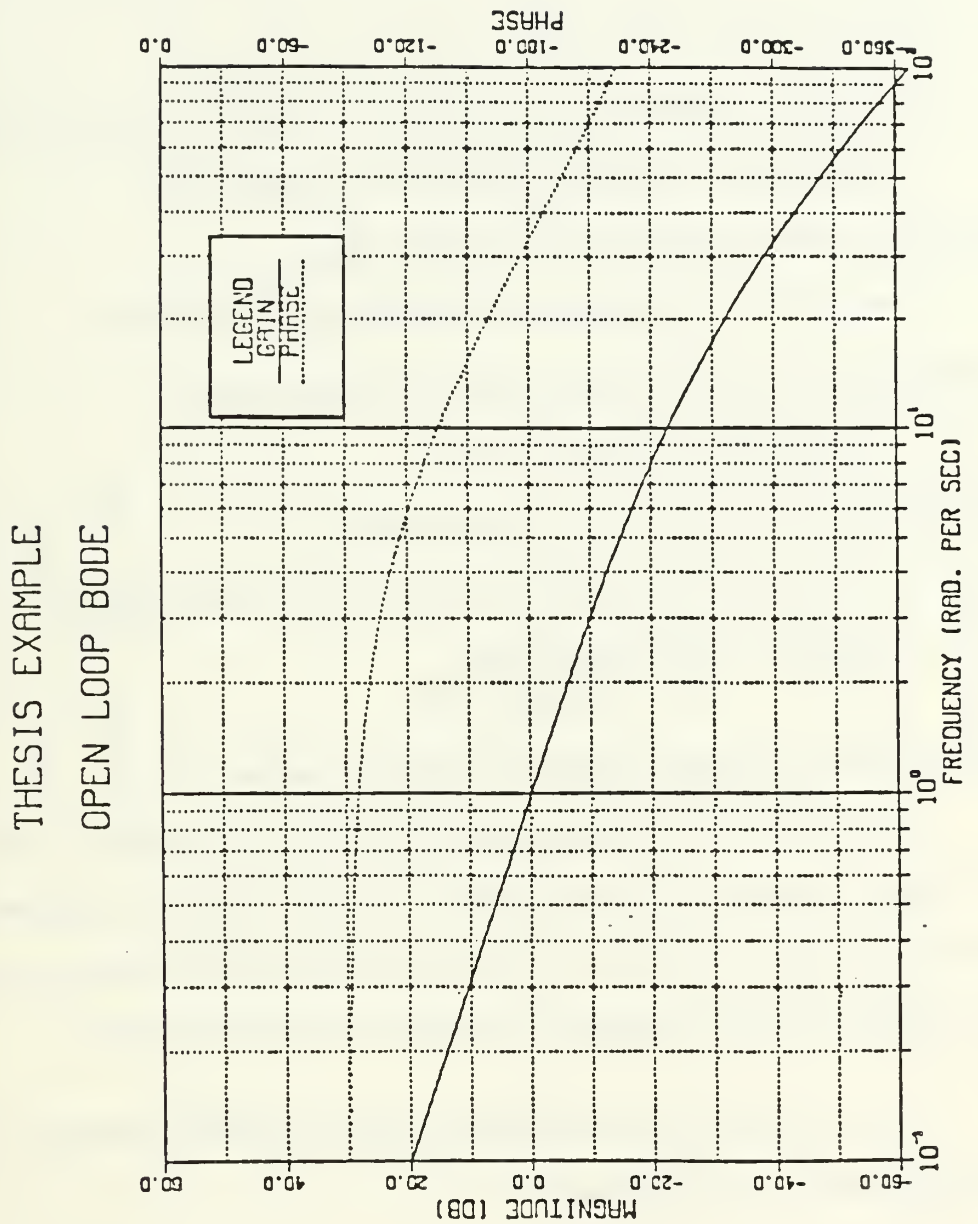


Figure E.2 Open Loop Frequency Response

First, change the titles to reflect that a closed loop Bode is being drawn. Enter a T.

```
        YOU MAY WRITE TWO LINES OF TEXT AS HEADING ON THE PLOT

        ENTER FIRST LINE (MAXIMUM 20 CHARACTERS)
```

For this run, THESIS EXAMPLE was entered again.

```
        ENTER SECOND LINE (MAXIMUM 20 CHARACTERS)
```

CLOSED LOOP BODE was typed in as the desired change.

```
        ANY MORE CHANGES?  (Y/N)
```

The user is now ready to draw the plot, so enter N

```
        DO YOU WANT OPEN OR CLOSED LOOP RESPONSE?

        ENTER   O   FOR OPEN LOOP
        ENTER   C   FOR CLOSED LOOP
```

The closed loop response is desired on this run so enter a C.

```
        DO YOU WANT TABULAR OUTPUT AT THE TERMINAL? (Y/N)
```

For this run, no tabular output is desired so enter N. The resulting closed loop Bode plot is shown in Figure E.3. After the graph has been drawn, the user is again asked if he wants to make any changes to the plot just drawn.

```
            DO YOU WISH TO CHANGE THE PLOT JUST DRAWN?  (Y/N)
```

Enter N to return to the main menu.

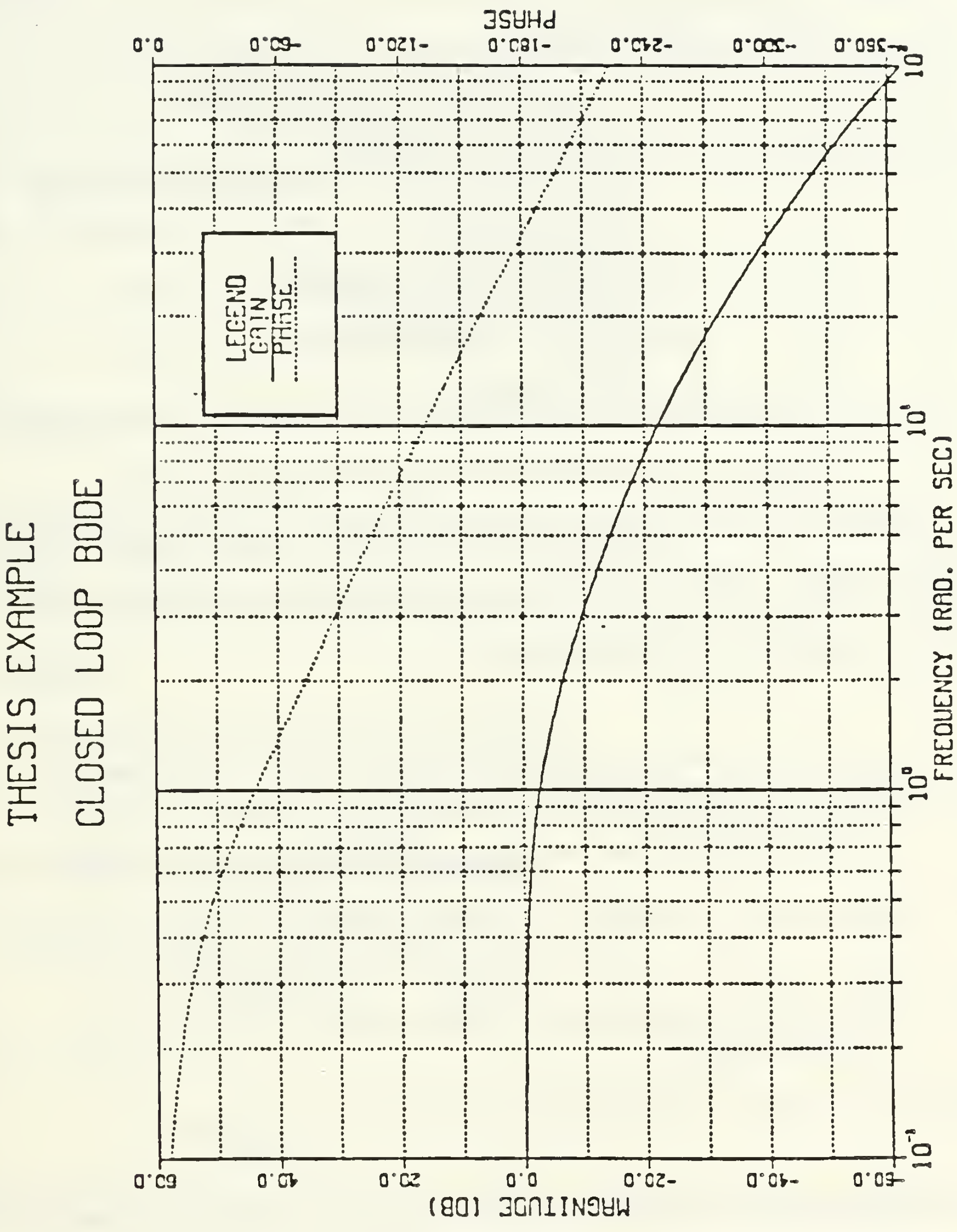


Figure E.3 Closed Loop Frequency Response

MAIN MENU	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	COEFFICIENT FORM OF TRANSFER FUNCTION
4	BODE ANALYSIS
5	TIME RESPONSE
6	CHANGE BLOCK DIAGRAM
7	SAVE THIS PROBLEM
8	START A NEW PROBLEM
9	HELP
10	EXIT PROGRAM

To begin the time response analysis, the user must select Option 5 from the main menu. This results in the user being presented with the Time Response Options menu shown below.

TIME RESPONSE OPTIONS	
OPTION NO.	OPTION
1	GRAPHICAL ANALYSIS
2	TABULAR DATA
3	RETURN TO MAIN MENU
4	HELP

```
OPTION NO.?
```

In this example, the user wants to produce a plot, so he would select Option 1. He will then be presented with the following prompt.

```
TIME RESPONSE PARAMETERS:
      1 = STEP
      2 = IMPULSE
      3 = RAMP
SELECT TYPE OF INPUT TO YOUR SYSTEM (1, 2 OR 3)
```

Enter Option 1 to obtain the step response. The user is then asked for the step amplitude.

```
WHAT IS THE AMPLITUDE OF YOUR INPUT?
     (PER SECOND FOR RAMP)
```

Let the step input amplitude be unity. The user then has the chance to insert a forward or feedback path gain.

```
WHAT IS THE FORWARD PATH GAIN?

WHAT IS THE FEEDBACK PATH GAIN?
```

In both cases enter 1 to accept the system as it was entered.

```
POSITIVE OR NEGATIVE FEEDBACK? (P OR N)
```

This problem used negative feedback so enter an N.

```
FOR HOW MANY SECONDS DO YOU WISH TO SEE THE RESPONSE?
```

Look at the response over the first 5 seconds.

At this point, the program will present the parameters just entered and give the user a chance to change his responses.

Next, the program asks for information concerning the heading.

```
PLOT HEADING
     HOW MANY LINES OF HEADING WOULD YOU LIKE? (4 MAX)

A MAXIMUM OF 32 CHARACTERS PER LINE IS ALLOWED.
     LINE 1 =
```

Enter 1 for the number of lines and enter STEP RESPONSE for the plot heading. The resulting plot is shown in Figure E.4.

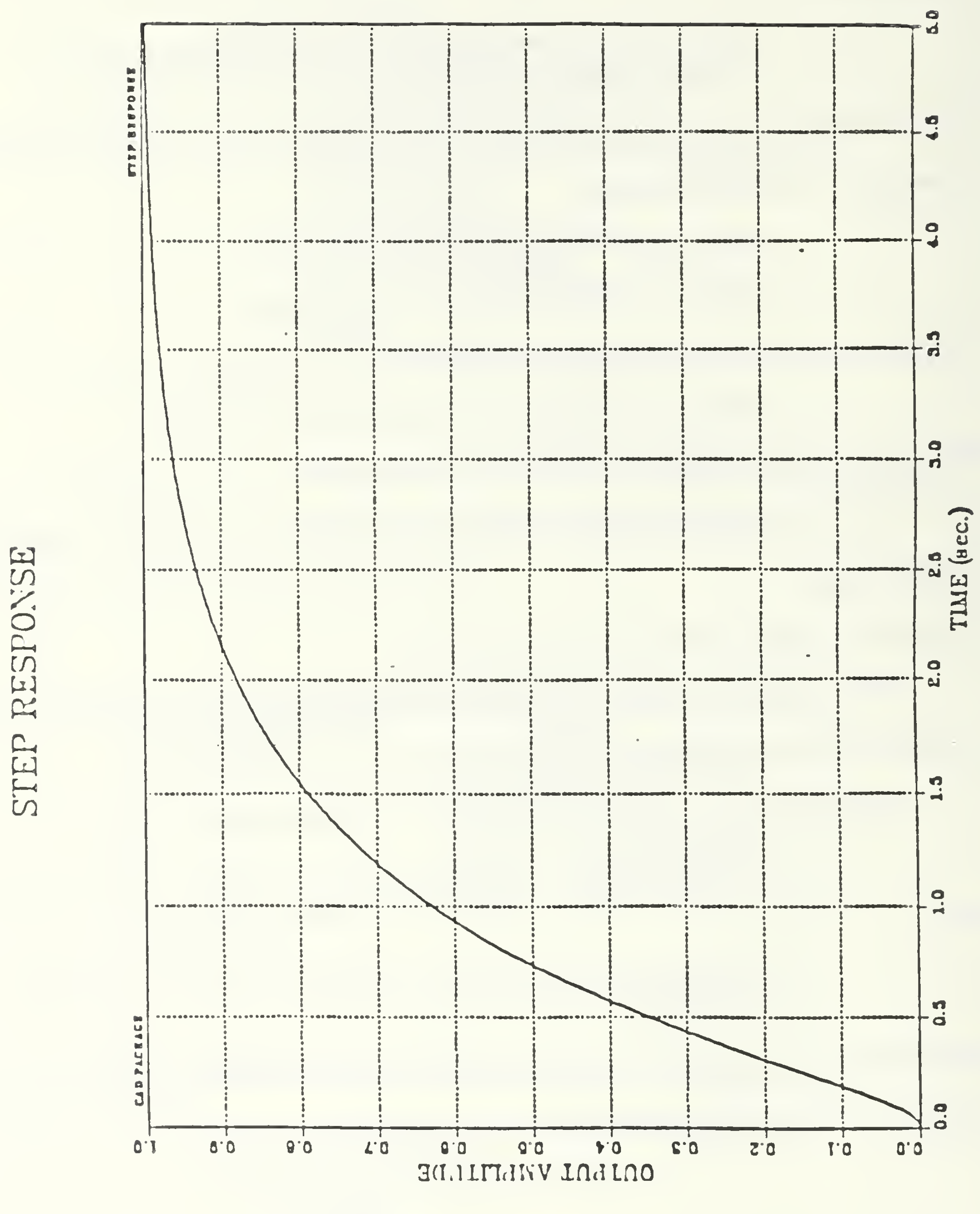


Figure E.4 Time Response

B. ROOT LOCUS FAMILY

This example demonstrates the capability of the program to draw a family of root loci for a characteristic equation containing two variable parameters. Consider the characteristic equation

$$s^3 + K_2s^2 + K_1s + K_1 = 0 \qquad \text{(E.1)}$$

where K_1 and K_2 are the variable parameters with values that lie between zero and infinity.

The user desiring to analyze this equation using root locus methods would enter the program and select Option 2 (ROOT LOCUS ANALYSIS) from the main menu. After answering 3 to the question asking for the order of the system, the user would enter the coefficients as follows:

N	A	B	C
3	1	0	0
2	0	1	0
1	0	0	1
0	0	0	1

As a first step let K_2=0; Equation E.1 then reduces to

$$s^3 + K_1s + K_1 = 0 \qquad \text{(E.2)}$$

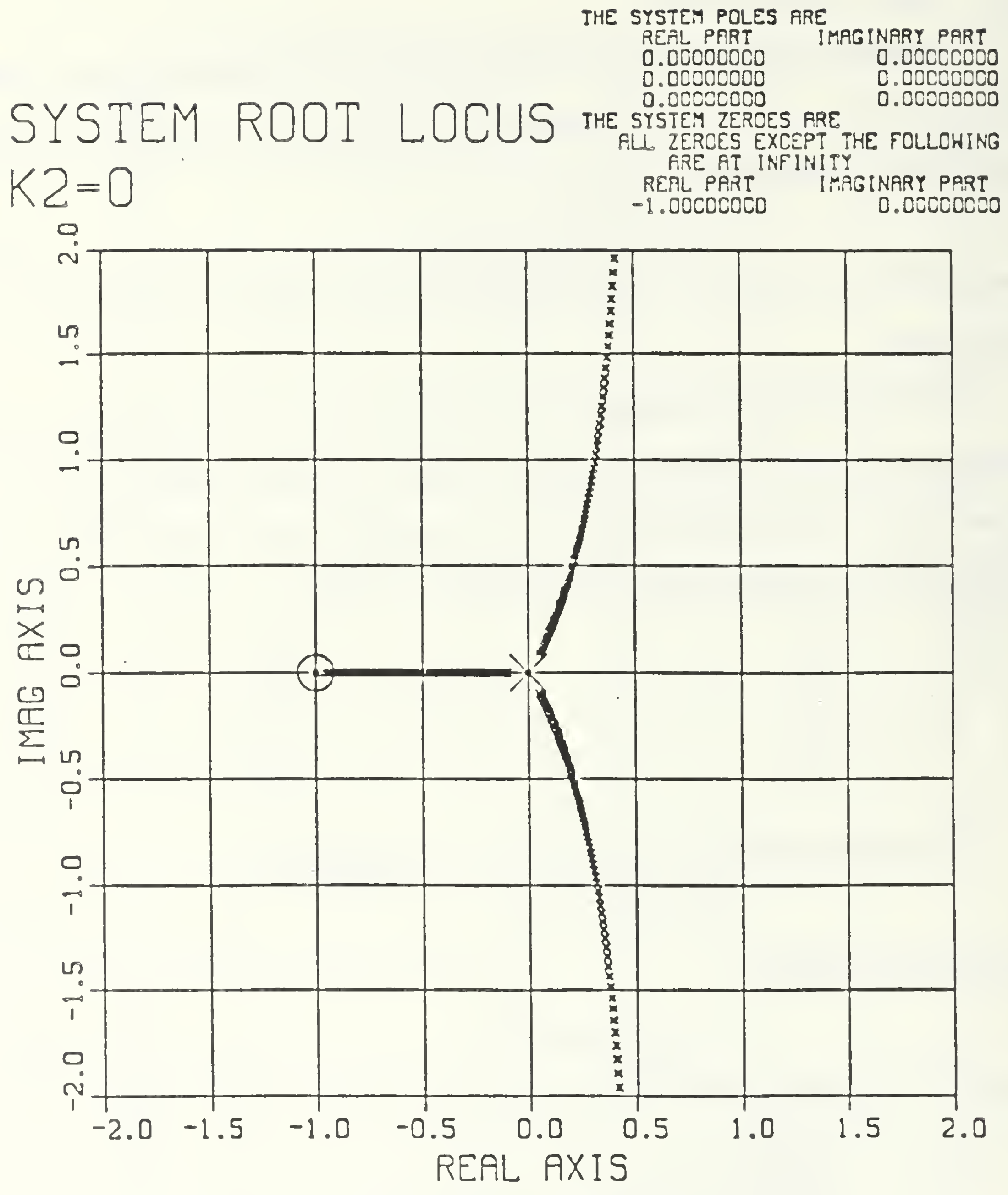
which is converted to

$$1 + [K_1(s+1)/s^3] = 0. \qquad \text{(E.3)}$$

The plot dimensions were entered as follows:

```
XMAX =  2
XMIN  = -2
YMAX =  2
XMIN  = -2.
```

The root loci of Equation E.2 is shown in Figure E.5.



```
THE SYSTEM POLES ARE
     REAL PART         IMAGINARY PART
     0.00000000            0.00000000
     0.00000000            0.00000000
     0.00000000            0.00000000
THE SYSTEM ZEROES ARE
   ALL ZEROES EXCEPT THE FOLLOWING
       ARE AT INFINITY
     REAL PART         IMAGINARY PART
    -1.00000000            0.00000000
```

Figure E.5 System Rooot Locus (K_2=0)

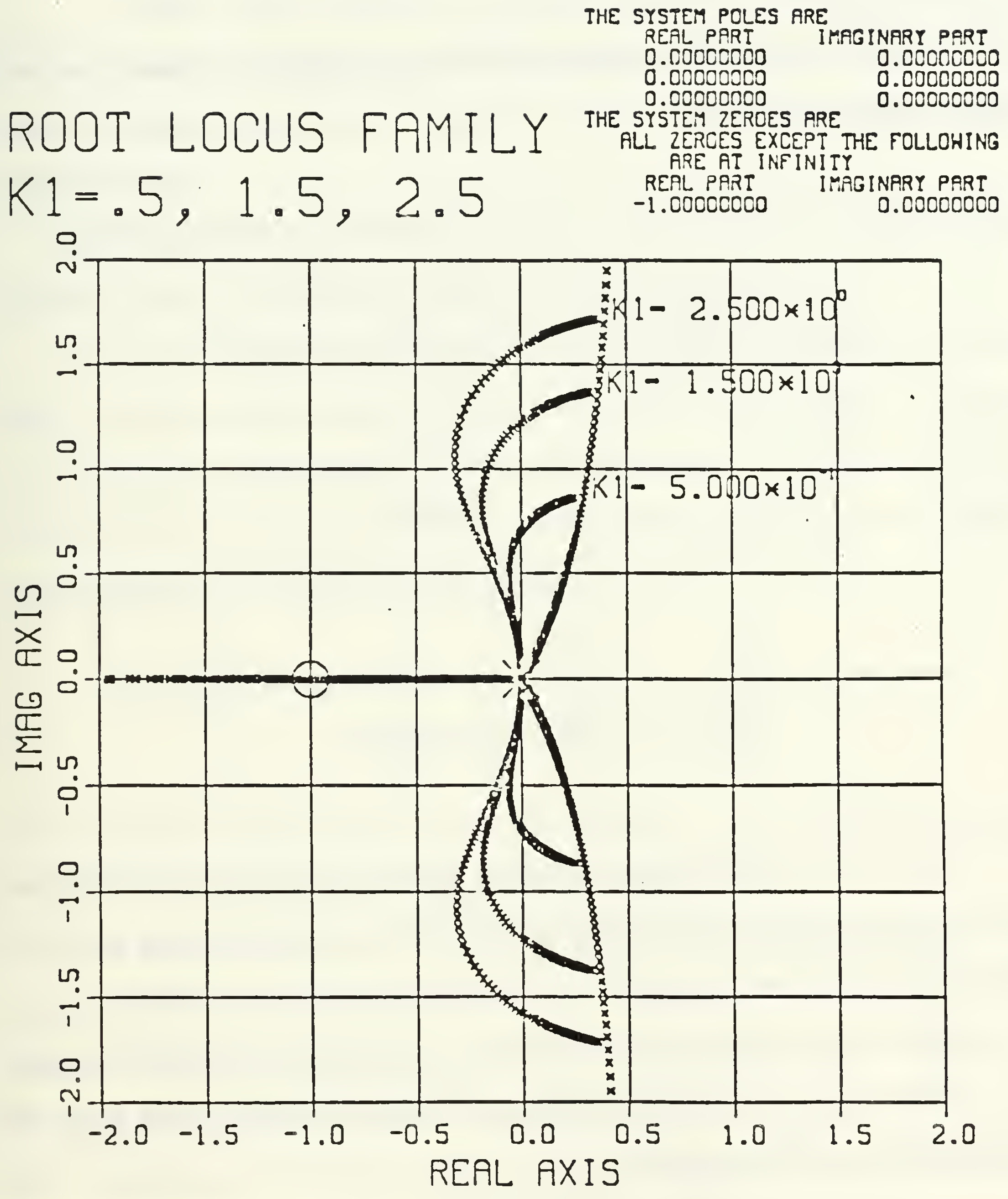
Next, let K_2 vary between zero and infinity while holding K_1 at a constant nonzero value. Dividing both sides of Equation E.1 by the terms that do not contain K_2, the result is

$$1 + [K_2 s^2/(s^3+K_1 s+K_1)] = 0. \qquad (E.4)$$

The values of K_1 used for this illustration were 0.5,1.5 and 2.5. The resulting graph is shown in Figure E.6. Table E.2 shows a limited portion of the file SYSTEM ROOTS established on the user's A-disk when tabular output is requested. Note that the output will print a message when the locus crosses the imaginary axis making it a little easier for the user to analyze tabular data.

```
 ROOTS FOR THE SPECIFIED VALUES OF THE VARIABLE ARE AS FOLLOWS

  VAR      REAL PART  IMAG PART      REAL PART  IMAG PART      REAL PART  IMAG PART
 0.10000000D-01
           0.100D+00  0.200D+00      0.100D+00 -0.200D+00     -0.200D+00  0.000D+00
 0.10402623D-01
           0.101D+00  0.203D+00      0.101D+00 -0.203D+00     -0.202D+00  0.000D+00
 0.10821456D-01
           0.102D+00  0.206D+00      0.102D+00 -0.206D+00     -0.205D+00  0.000D+00

(DATA DELETED TO SAVE SPACE IN THESIS

 0.85393904D+01
           0.456D+00  0.303D+01      0.456D+00 -0.303D+01     -0.911D+00  0.000D+00
 0.88832056D+01
           0.457D+00  0.308D+01      0.457D+00 -0.308D+01     -0.914D+00  0.000D+00
 0.92408636D+01
           0.458D+00  0.314D+01      0.458D+00 -0.314D+01     -0.917D+00  0.000D+00
 0.96129218D+01
           0.460D+00  0.320D+01      0.460D+00 -0.320D+01     -0.919D+00  0.000D+00

 ROOTS FOR THE SPECIFIED VALUES OF THE VARIABLE K1 ARE AS FOLLOWS
(K1=.5)
  VAR      REAL PART  IMAG PART      REAL PART  IMAG PART      REAL PART  IMAG PART
 0.10000000D-01
           0.291D+00  0.872D+00      0.291D+00 -0.872D+00     -0.592D+00  0.000D+00
 0.10132446D-01
           0.291D+00  0.872D+00      0.291D+00 -0.872D+00     -0.592D+00  0.000D+00
 0.10540401D-01
           0.291D+00  0.872D+00      0.291D+00 -0.872D+00     -0.592D+00  0.000D+00

(DATA DELETED TO SAVE SPACE)

 0.92408760D+00
          -0.951D+00  0.000D+00      0.135D-01  0.725D+00      0.135D-01 -0.725D+00
 0.97402538D+00
          -0.983D+00  0.000D+00      0.443D-02  0.713D+00      0.443D-02 -0.713D+00
0A LOCUS HAS JUST CROSSED THE IMAGINARY AXIS

 0.10266618D+01
          -0.102D+01  0.000D+00     -0.434D-02  0.701D+00     -0.434D-02 -0.701D+00
 0.10821427D+01
          -0.106D+01  0.000D+00     -0.127D-01  0.688D+00     -0.127D-01 -0.688D+00
```

Table E.2 Edited Root Locus Tabular Data



```
THE SYSTEM POLES ARE
     REAL PART          IMAGINARY PART
     0.00000000              0.00000000
     0.00000000              0.00000000
     0.00000000              0.00000000
THE SYSTEM ZEROES ARE
   ALL ZEROES EXCEPT THE FOLLOWING
        ARE AT INFINITY
     REAL PART          IMAGINARY PART
    -1.00000000              0.00000000
```

Figure E.6 Root Locus Family Example

C. MULTILOOP SYSTEM

For this final example consider the multiloop system of limited complexity shown below in Figure E.5.

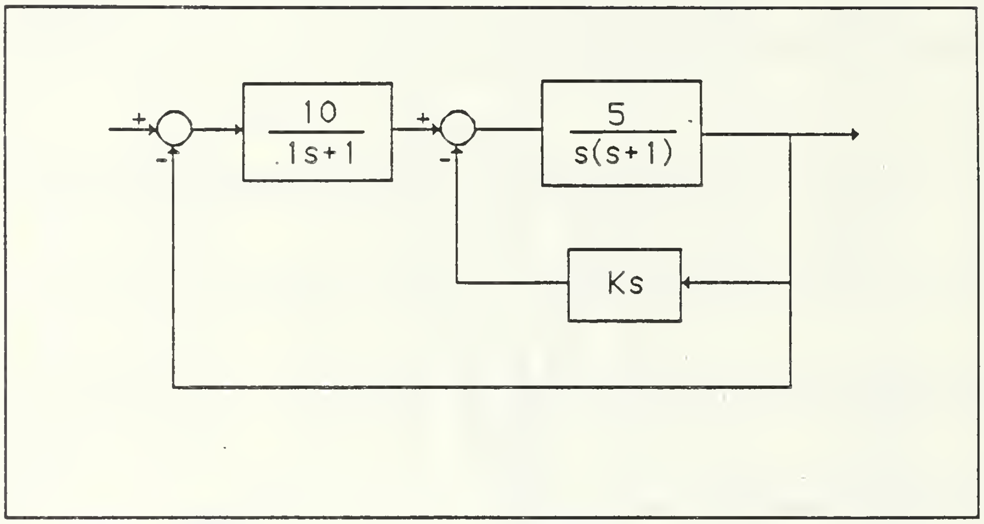


Figure E.5 Multiloop System

The user could enter this system selecting a specific value for K and analyze it using root locus, Bode or time response methods. He could then go back to the inner loop and change the value of K and analyze the resulting system.

Alternatively, the user could calculate the characteristic equation retaining the variable K and use a method called partitioning to select a value for K. The characteristic equation would be

$$.1s^3 + (1.1 + .5K)s^2 + (1.5 + 5K)s + 55 = 0. \qquad \text{(E.5)}$$

In partitioning, Equation E.5 would be rearranged so that the terms that correspond to the variable K would be grouped separately from the rest as in Equation E.6.

$$(.1s^3 + 1.1s^2 + 1.5s + 55) + K(.5s^2 + 5s) = 0. \qquad \text{(E.6)}$$

The root locus for the system of Equation E.6 is then drawn. The user could examine tabular output in order to select a value for K that results in the desired root placement.

At this point, the user would enter the system transfer function with the selected value of K and plot the frequency and time responses.

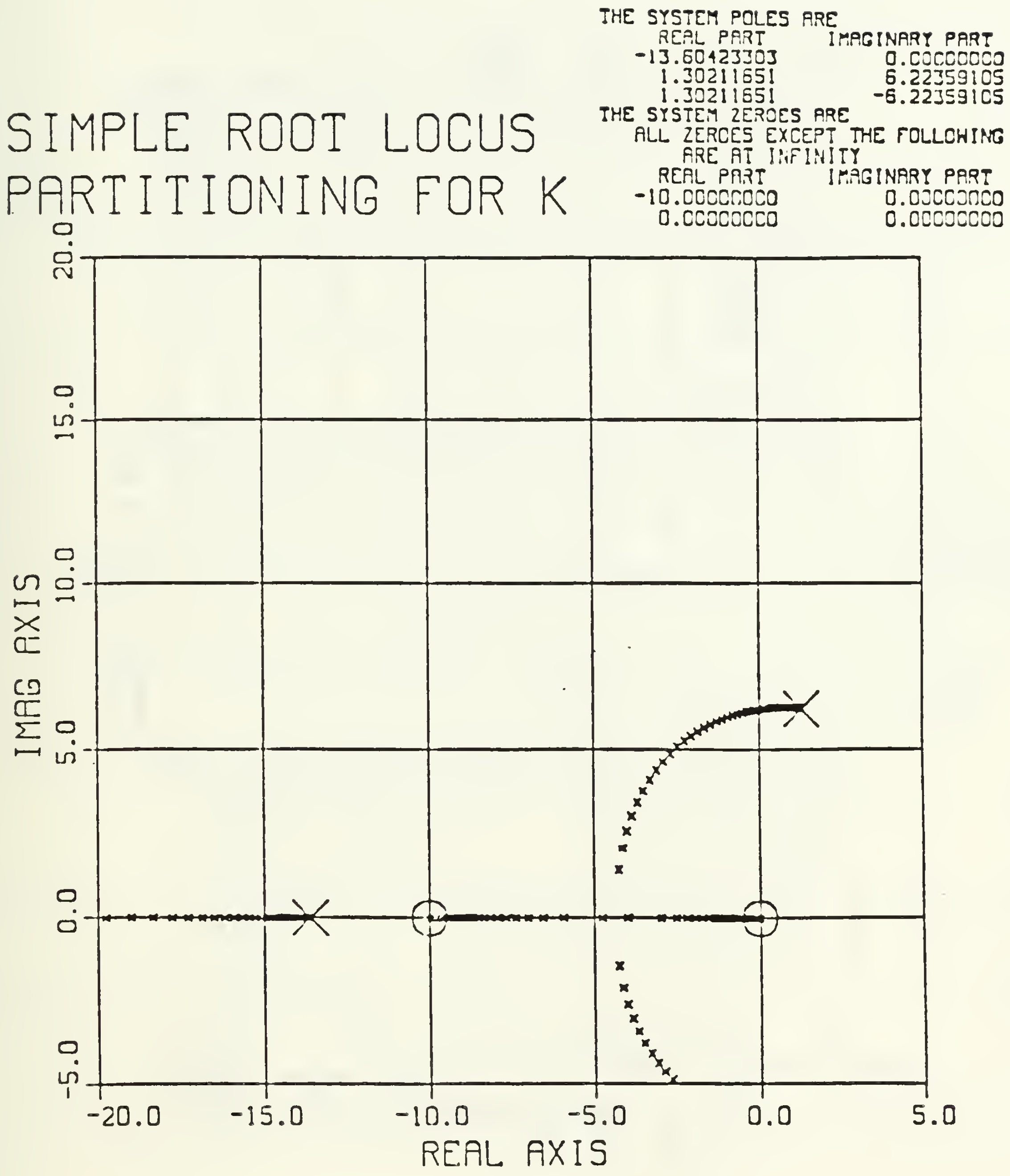
In order the draw the root locus for the system of Equation E.6, select Option 2 from the main menu. Enter 3 for the order of the system. Only one of the program's variable parameters will be needed as in the case of drawing a simple root locus. For this example, assume that K_2 is used. In that case the user would enter the coefficients as follows:

N	A	B	C
3	0.1	0	0
2	1.1	0.5	0
1	1.5	5.0	0
0	55.	0	0

The resulting root locus plot is shown in Figure E.6. For this illustration, assume the desired root locations correspond to the value K=2.25.

The system of Figure E.5 can now be entered using the value of K just selected. The user must start with the innermost loop. This loop contains two blocks, one in the forward path and one in the feedback path. Coefficient or factored form could be selected. Negative feedback is used. The user must then expand to an outer loop. The closed loop equivalent of the inner loop just entered should be placed in block number two of the outer loop. Block number one of the outer loop can then be entered. At this point, the user may use any of the analysis routines to analyze the system just entered.

The resulting Bode and time response plots are shown in Figures E.7 and E.8 respectively.



```
THE SYSTEM POLES ARE
      REAL PART          IMAGINARY PART
   -13.60423303            0.00000000
     1.30211651            6.22359105
     1.30211651           -6.22359105
THE SYSTEM ZEROES ARE
    ALL ZEROES EXCEPT THE FOLLOWING
        ARE AT INFINITY
      REAL PART          IMAGINARY PART
   -10.00000000            0.00000000
     0.00000000            0.00000000
```

Figure F.7 Simple System Root Locus

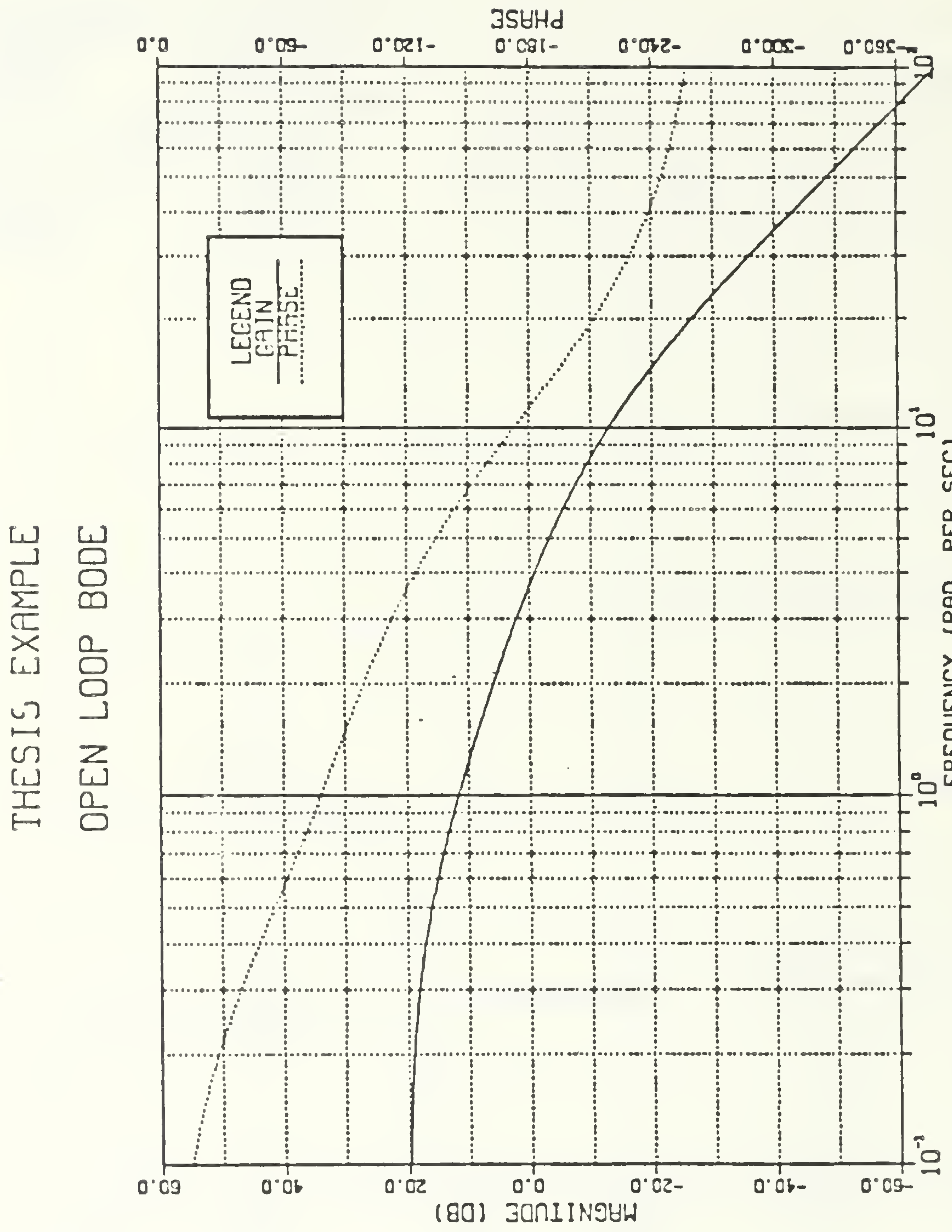


Figure E.8 Open Loop Bode Plot

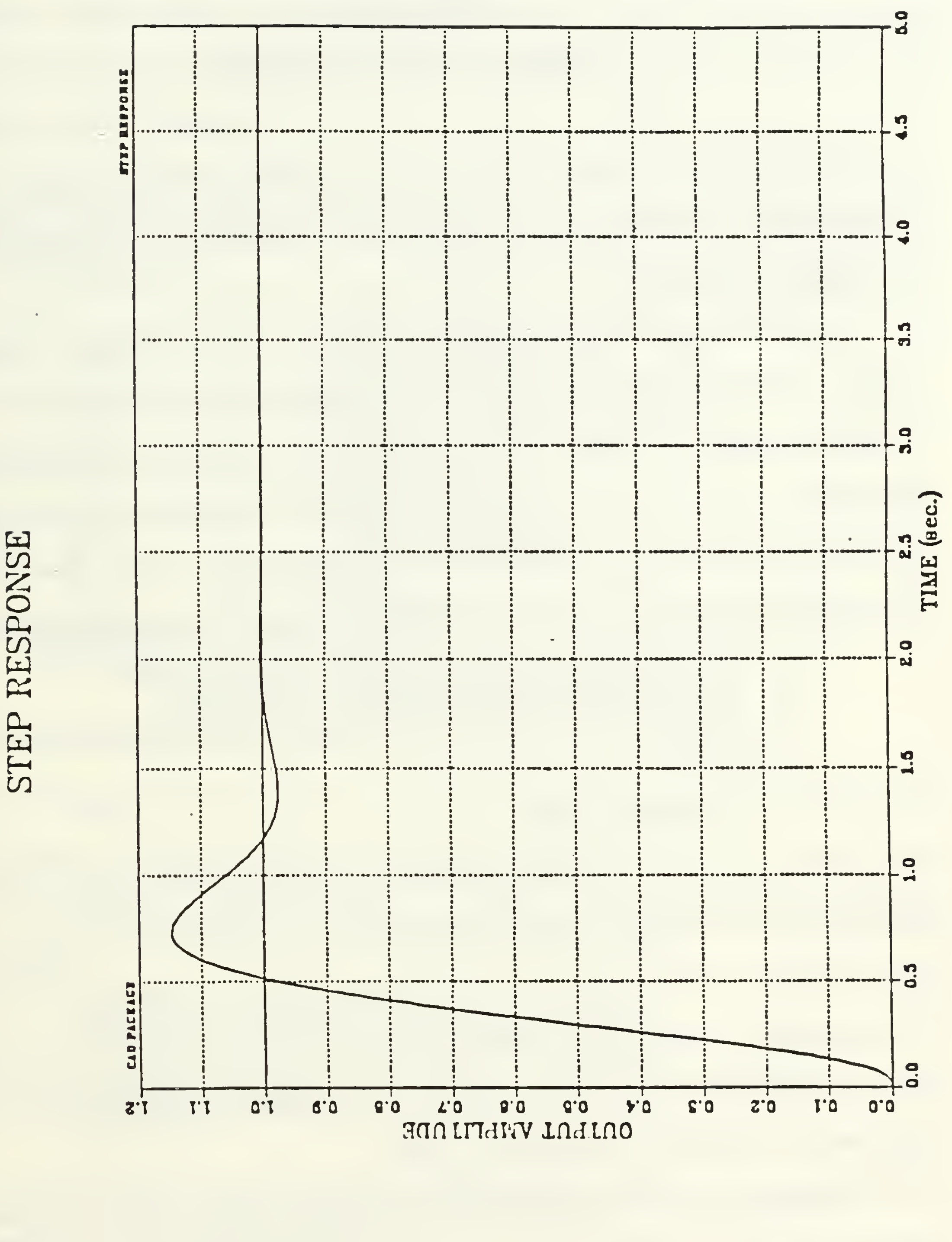


Figure E.9 Closed Loop Bode Plot

APPENDIX F

PROGRAM IMPROVEMENTS

This appendix contains information concerning changes to the program subsequent to the completion of the thesis. There are only three additions that are apparent to the user.

The first improvement makes it easier for a user to obtain a simple root locus. Originally, the user would have to obtain the coefficient form of the transfer function. He then must enter the coefficients of the characteristic equation in the root locus routine being careful to assign one of the allowed variable parameters to correspond to the numerator gain. Now, when the user first enters the root locus routine, he is presented with the following options:

```
1.  ENTER COEFFICIENTS FROM THE CONSOLE
2.  ENTER COEFFICIENTS FROM A DATAFILE
3.  SIMPLE ROOT LOCUS OF EXISTING TRANSFER FUNCTION
4.  HELP

    ENTER OPTION
```

By selecting Option 3 (the previous menu did not include this option), the coefficients are assigned automatically using the information entered by the user previously in the transfer function input routine. The user must still enter the parameters necessary for plotting dimensions and heading, but he is freed from forming the characteristic equation and assigning coefficients (this is still necessary in the case of a root locus family).

The second and third improvements involve finding roots. The designer wants to know the location of the roots of the characteristic equation. The main menu now includes the option to calculate the roots of the characteristic equation of the current transfer function entered by the user. When adding this feature, it was simple to add an option to allow the user to find the roots of a polynomial for which he must simply enter the coefficients when prompted. This option was also added to the main menu as shown below.

```
                          MAIN MENU
OPTION NO.                  OPTION

    1          INPUT TRANSFER FUNCTION(S)
    2          ROOT LOCUS ANALYSIS
    3          ROOTS OF USER POLYNOMIAL
    4          ROOTS OF CHAR EQUATION OF TRAN FUNC
    5          COEFFICIENT FORM OF TRAN. FUNC.
    6          BODE ANALYSIS
    7          TIME RESPONSE
    8          CHANGE BLOCK DIAGRAM
    9          SAVE THIS PROBLEM
   10          START A NEW PROBLEM
   11          HELP
   12          EXIT PROGRAM
OPTION NO?
```

If the user selects Option 4, he is presented with a screen listing the roots of the characteristic equation of the existing transfer function.

If Option 3 is selected, the user is first prompted for the order of his polynomial.

ENTER THE ORDER OF YOUR POLYNOMIAL

For an example, assume we want to know the roots of the polynomial

$s^2+11s+10$.

The user would enter 2 in answer to the above prompt. The program then prompts for the polynomial coefficients.

ENTER POLYNOMIAL COEFFICIENTS

```
X ** 2 =

X ** 1 =

X ** 0 =
```

For this example, the user would enter 1, 11 and 10 in that order. After the coefficients have been entered, the program presents the coefficients and asks if they are correct.

```
THE COEFFICIENTS ARE
     A(2) = 0.10000E+01
     A(1) = 0.11000E+02
     A(0) = 0.10000E+02

ARE THESE CORRECT?  (Y/N)
```

After responding with Y, the user is presented with the roots of his polynomial.

```
THE ROOTS ARE AS FOLLOWS
     REAL PART                IMAG PART
     -1.0000000               0.000000
     -10.000000               0.000000

CLEAR SCREEN TO CONTINUE
```

The user is then returned to the main menu.

LIST OF REFERENCES

1. Cooksey, C. M., An Interactive Computer Aid for the Design and Analysis of Linear, Single Input / Single Output Digital and Continuous Controls Systems, M. S. Thesis, Naval Postgraduate School, Monterey, California, December 1984.

2. Duffin, J., PAROLE, paper explaining the use of the batch version of the two-parameter root locus program distributed in his classes at the Naval Postgraduate School, July 1967.

3. Ismail, H., An Interactive Program, M. S. Thesis, Naval Postgraduate School, Monterey, California, March 1984.

BIBLIOGRAPHY

Kuo, B., Automatic Control Systems, Prentice Hall, 1982.

Ogata, K., Modern Control Engineering, Prentice Hall, 1970.

Thaler, G. J., Design of Feedback Systems, Dowden, Hutchinson & Ross, Inc., 1973.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Distiguished Professor G. J. Thaler Code 62Tr Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	25
4.	Professor J. H. Duffin Code 62Dn Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	4
5.	Adjunct Professor A. Rigas Code 62Rx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	2
6.	Professor H. A. Titus Code 62Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
7.	Associate Professor A. Gerba, Jr. Code 62Gz Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
8.	Professor D. E. Kirk Code 62Ki Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1

9. Professor R. D. Strum Code 62St
Department of Electrical and Computer Engineering 1
Naval Postgraduate School
Monterey, California 93943

10. Adjunct Professor S. Turajlic Code 62Tc
Department of Electrical and Computer Engineering 1
Naval Postgraduate School
Monterey, California 93943

11. LT T. L. Ewald
82 N. Circle Dr. 1
Wright City, MO 63390