DEVELOPING ROOT LOCUS STABILITY DIAGRAMS USING A PERSONAL COMPUTER

by

Ben C. Svrcek

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

APPROVED:

_____

R. O. Claus, Chairman

_____ _____

T. C. Poon J. G. Tront

September, 1987

Blacksburg, Virginia

DEVELOPING ROOT LOCUS STABILITY DIAGRAMS USING A PERSONAL COMPUTER

by

Ben C. Svrcek

(ABSTRACT)

Companies which design automation control for the metal rolling industry are faced with a growing demand for systems with higher performance standards than ever before. Along with these demanding specifications is always the problem of system stability at any given speed. A multi-ton rolling stand with uncontrolled oscillations not only destroys the product being rolled but may cause serious damage to the plant and endanger the lives of mill personnel. Therefore stability analysis is critical whether modeling individual mills or analyzing old products and strategies so as to invent better, cheaper control methods.

Cost is another major consideration for the firm ordering these systems and the companies which design them. Suppliers are trimming time from design and production schedules wherever possible in order to compete in the world market. It is for these, and other reasons that computer aided stability analysis is so important. The object is to ensure a safe and stable system and yet minimize the time

(and therefore cost) needed for design and installation. This paper describes a program (ROOT LOCUS) which was created to fill this need while using the tools and methods readily at hand. It was written for personal computers as these machines are rapidly proving to be cost effective solutions to problems in computing power.

ACKNOWLEDGEMENTS

There are a number people who have helped make this thesis possible either by their technical guidance or support. My wife, Fran, deserves a special note of appreciation for her continued support of my education and the endurance of the many inconveniences entailed by it. I wish to especially thank Dr. R. O. Claus for without his assistance and encouragement I would have quit the pursuit of my degree long ago. I would also like to thank Paul Dornbusch for the many discussions on system stability that I had with him or had the privilege to sit in on. Finally, a note of thanks to the management of the General Electric Company Drive Systems Operation for not only providing full time employment as an engineer but also for providing the means to continue my education.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

CHAPTER 1

INTRODUCTION

1.0 MOTIVATIONS

In my work at General Electric in Salem, Virginia, I designed computer control systems for major metal rolling companies such as Bethlehem Steel. A typical project is an automatic gage control system for an aluminum rolling mill. Systems such as this have strip speed and tension control along with position regulation of the work rolls and the sideguides which guide the metal into these mills. Tolerances of +/- 1% gage deviation on a strip 15 mills thick are not uncommon along with strip speeds of thousands of feet per minute. Along with these demanding specifications is always the problem of system stability at any given speed. A multi-ton rolling stand with uncontrolled ocillations not only destroys the product being rolled but may cause costly damage to the mill and endanger the lives of mill personnel.

There are many studies modeling mills (e.g. Single-stand And Tandem Cold Mill Dynamic Simulation [1]) which require some form of stability analysis whether it is

for tension regulation of the strip or for automatic gage control. Analysis of old products and control strategies also entail some investigation of stability in order to better understand the systems we automate [2] so as to invent better, cheaper control methods for our customers. As implied, cost is another major consideration. Companies are trimming time from design and production schedules wherever possible to cut the price of their products. The steel industry is looking for the most control for the least amount of money from automation equipment suppliers such as General Electric. It is for these, and other reasons, that stability analysis is so important. The problem is to ensure a safe and stable system and yet minimize the time (and therefore cost) needed for design. This program was created to fill this need for quick stability diagrams to ensure safety and cost effectivness while using the tools and methods readily at hand.

1.1 REASONS FOR DEVELOPING ROOT LOCUS STABILITY DIAGRAMS ON A PERSONAL COMPUTER

Many different tools are used in designing control systems for use in industry. Computers especially have invaded the field and seem to make graphic methods of design and analysis obsolete. When one can have an exact answer to any stability problem in a few minutes it seems to be a

waste of time to obtain a graphic one. However, if the data fed into the computer is inadequate then an exact solution on the system's installation site is worthless. This is the case that companies which design and install automated control systems (such as General Electric) encounter many times.

The reasons for inaccurate data are many and varied. Many of the motors in use are old with the specifications lost in the customer's files. Or the entire system may have been rewired and no longer match the original specifications. Machine wear also takes its toll. All this makes the modeling of any system an exercise in making educated guesses. A system may be well simulated at the factory but for optimum performance it must be tuned in the field. The field engineer may want to increase the loop gain but is uncertain on how far to adjust this gain while still retaining stablity. Root Locus Stability Diagrams provide a quick and convenient means to estimate a needed gain or necessary lead / lag networks. Documentation such as this also does not become obsolete quickly and thus is useful to the customer for a longer period of time. Another feature is that while many engineers in heavy industry readily grasp stability diagrams, newer methods such as State Space Analysis may confuse them.

Using a computer to generate these diagrams is a matter of saving time. First, although the design engineer is

perfectly capable of making these calculations and doing freehand plots, the computer can free the engineer for more creative tasks. Also the computer can plot the results as neatly as any draftsman, thereby freeing that person too. Main frame computers such as Digital Equipment Corporation's VAX series are widely used today both in the design environment and in heavy industry control applications. Root Locus programs utilizing languages such as FORTRAN have existed on these main frames for years [3]. However, the industrial control computer has serious drawbacks in this application for its use by field engineers. Many computers are heavily loaded and cannot be used for other purposes without endangering the control systems performance. The mill owners that buy the main frame may know much about producing steel but nothing about main frame computers. Thus, they may hire personnel who know a great deal about computer science but little or nothing of control systems.

Personal computers offer a cost effective solution to these, and other problems of field personnel. With a personal computer and the Root Locus program, the home office engineer may design a system and then send a Root Locus Diagram to the field engineer through the mails or transmit it via modems and telephone. The field engineer may then update the diagram if the need arises to do so. Most personal computers come equipped for BASIC hence a

program written in this language is immediately useful to an engineer. BASIC has the added advantage of being easily translated from one brand of personal computer to another (e. g. from APPLE BASIC to IBM BASIC).

CHAPTER 2

SOLUTION METHODS

2.1 OVERVIEW OF METHODS

The main task of a Root Locus program is to solve for the roots of the characteristic equation (defined here as being the systems open loop transfer function) for the system being modeled. This equation takes the form:

$$f(s) \equiv a_0 S^n + \ldots + a_{n-1} S + a_n = 0 \qquad (2.1.1)$$

The terms of equation (2.1.1) take the form

$$a_n = b_n + c_n K \qquad (2.1.2)$$

where b and c are constants and K is the system gain. It is important to note that f(s) in (2.1.1) is defined as being equal to zero. The object of ROOT LOCUS is to find the roots of the transfer equation for a particular gain K and plot these roots. The gain is then incremented by some value and the roots of this new equation found. The result is a graph in the S plane that is a sheaf of discrete roots. This plot of discrete roots appears as a continuous function

on the graph. Most methods used for solving an n dimensional polynomial involve some sort of iterative process. Many are not suited to the memory and speed limitations of a personal computer.

Any program for solving this problem faces multiple restrictions. First, it must use as little memory for itself as is possible in order to maximize the memory available for data. Next, it must have as few steps as possible in order to minimize the time needed to execute it. This is due to the slow clock rates of most personal computers. Another restriction is the number of bits used by the microprocessor (i.e. is it an eight bit machine, a sixteen bit one, etc.). Does the processor used by the computer have floating point capablities or not? These things affect the accuracy of the answer. This is the criteria used to evaluate the various methods of solving for the roots of an equation. Using this guideline, there are two methods which have been found useful for the factoring of an n dimensional polynomial. These are the Newton - Ralphson and Lin - Bairstow Methods [4]. Both utilize matrix algebra to obtain derivitives.

Newton - Ralphson is the simpler of the two methods. This, along with the author's familiarity with it, were the reasons for the choice of this method for the ROOT LOCUS program. It works equally well with either complex numbers or real numbers and generally converges quickly. The

Newton - Ralphson method deals with even and odd order polynomials with equal ease. However, problems do arise unless the initial estimate of the first root is close to one of the roots of the equation. Otherwise the method is slow to converge and indeed may not converge at all.

The Lin - Bairstow method works well with even number polynomials and in situations where the roots of the equation are clustered together. It is not as critical to estimate the first root as close to an existing root as in the Newton - Ralphson method.

2.2 LIN - BAIRSTOW METHOD

The Lin - Bairstow method tries to find a quadratic formula which will divide into equation (2.1.1) with no remainder. This can be expressed as

$$f(s) = (S^2 + uS + v)\, g(s) + B_1 S + B_0 \qquad (2.2.1)$$

By manipulating u and v, $B_1S + B_0$ can be reduced to zero. The B terms can be expressed in terms of a change in u and v by the Taylor series expansions

$$-B_1 = \frac{\partial B_1}{\partial u}\Delta u + \frac{\partial B_1}{\partial v}\Delta v \qquad (2.2.2)$$

$$-B_0 = \frac{\partial B_0}{\partial u}\Delta u + \frac{\partial B_0}{\partial v}\Delta v \qquad (2.2.3)$$

The change in u and v is added to those terms respectively

until the iteration converges. The g(s) term of (2.2.1) is given as

$$g(s) = b_0 S^{n-2} + b_1 S^{n-3} + \ldots + b_{n-2} \qquad (2.2.4)$$

where

$$\begin{aligned} b_0 &= a_0 \\ b_1 &= a_1 - u b_0 \\ b_K &= a_K - u b_{K-1} - v b_{K-2} \quad (K = 2,\ldots,n-2) \end{aligned} \qquad (2.2.5)$$

B_1 and B_0 are derived from

$$\begin{aligned} B_1 &= a_{n-1} - u b_{n-2} - v b_{n-3} \\ B_0 &= a_n - v b_{n-2} \end{aligned} \qquad (2.2.6)$$

The a_n term is from equation (2.1.1)

The first step in implementing the Lin - Bairstow method is to estimate the values of u and v in the starting quadratic. This can be done by truncating the higher order terms to obtain a quadratic and then taking its root. There are other means to do this starting root estimate (such as Bernoulli's method) but this involves more code. The terms of equations (2.2.2) and (2.2.3) are calculated next. From this, the change in u and v are then calculated. Convergence is tested by dividing the change in the variable by the variable (e.g. $\Delta u / u$). If the result is less than, or equal to, some small value (e.g. 0.0001) then the equation is said to have converged. This answer is then factored out leaving a polynomial of order n-2. The terms u and v from the previous iteration may be used as the base values of u and v in the next pass [5].

2.3 NEWTON - RALPHSON METHOD

The basis for this method is a Taylor series expansion about an initial estimate of the root. Starting with a graphic representation, let a given function f(x) be plotted as shown in figure 1. A root α of f(x) is found when f(x) = 0. Let x_o be the initial estimate of the root α. X_o is mapped to the function at (x_o, f(x_o)) and a tangent to the curve of f(x) is drawn at this point. Where this tangent strikes the x axis is where a new estimate of α is found. This point is also mapped to f(x) and the procedure is reiterated. From this, it may be seen that the error between the estimate and the root may be given as

$$h = \alpha - x_o \qquad (2.3.1)$$

Therefore the object of Newton - Ralphson is to reduce the error "h" to zero [6].

The Taylor expansion about the initial root approximation x_o can be expressed as

$$f(\alpha) = f(x) = f(x_o + h) \qquad (2.3.2)$$

or, by using Taylor's Theorem of the Mean,

$$f(x) = f(x_o) + hf'(x_o) + Rn \qquad (2.3.3)$$

where

x_o is the approximation of the root

$\alpha = x_o + h$

h is the error in the root approximation

Rn is the Taylor series remainder

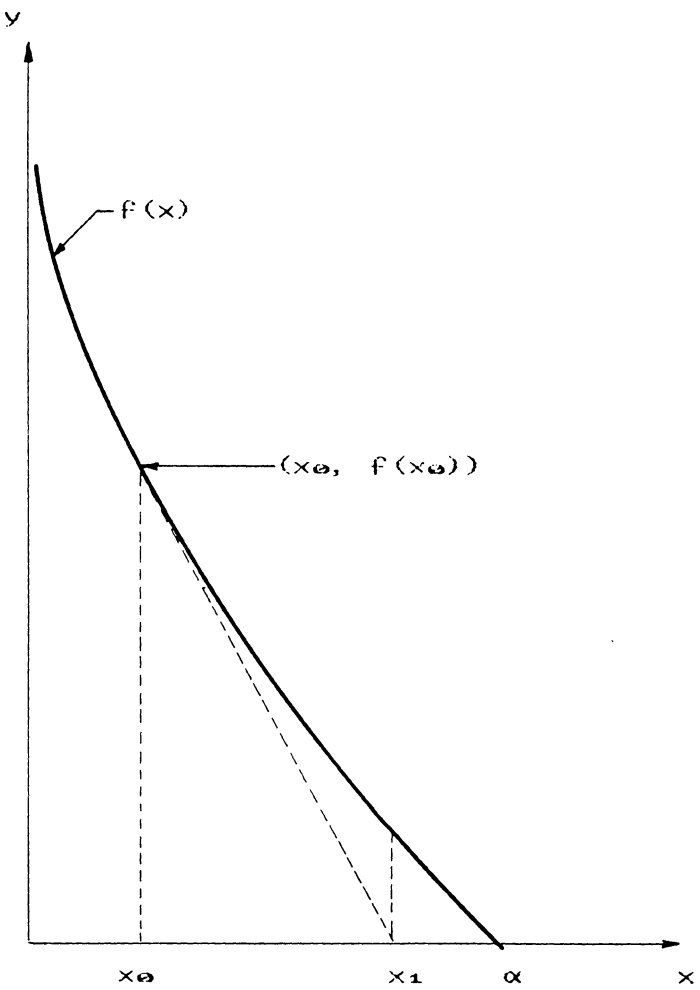


Figure 1: Graphic Basis of Newton - Ralphson

Rn is the truncation error and in Lagrange's form it is given as

$$Rn = (h^2 / 2!)\ f''(\delta) \qquad (2.3.4)$$

where

$$x_0 < \delta < \alpha = x_0 + h \qquad (2.3.5)$$

Let h be sufficiently small such that the remainder term (2.3.5) tends to 0. At $f(x) = 0$ then (2.3.3) becomes

$$0 = f(x_0) + hf'(x_0) \qquad (2.3.6)$$

making the error h

$$h = - \frac{f(x_0)}{f'(x_0)} \qquad (2.3.7)$$

However, the error term is not zero therefore (2.3.1) can be written as

$$x_1 = x_0 + h \qquad (2.3.8)$$

or, by substituting (2.3.7) for h

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \qquad (2.3.9)$$

The next iteration of the root approximation is similar to (2.3.9) and is written as

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad (2.3.10)$$

The truncation error caused by assuming a small h in (2.3.6) affects the convergence and it can be shown [7] that the error between iterations is

$$x_{k+1} - x_k = \frac{h^2\ f''(\delta)}{2\ f'(x_k)} \qquad (2.3.11)$$

There are two conditions for fast convergence; the first being that the error term defined in (2.3.11) is not large. The other condition is that the initial estimate of the root must be close to an existing root. If either of these requirements are not met then convergence will be slow or may not even occur [8].

The procedure for implementing this in a computer starts in the same manner as Lin - Bairstow in that an initial root x_0 is calculated. Next, the terms $f(x_0)$ and $f'(x_0)$ are initialized. The iterative part of the process starts with the evaluation of $f(x_0)$ and $f'(x_0)$ using the sucessive over-relaxation method which makes a change in the unknown variable larger than is required in an effort to converge quicker [9]. It is implemented in a matrix operation by nested multiplications such as indicated in equations (2.3.12) and (2.3.13).

$$f(x_0) = (\ldots((a_0x_0 + a_1)x_0 + a_2)x_0 + \ldots)x_0 + a_n \qquad (2.3.12)$$

$$f'(x_0) = (\ldots(((na_0x_0 + (n-1)a_1)x_0 + (n-2)a_2)x_0 \ldots a_{n-2})x_0 + a_{n-1} \qquad (2.3.13)$$

The convergence is now tested by the formula

$$\left| \frac{x_{k+1} - x_k}{x_k} \right| < \epsilon \qquad (2.3.14)$$

where epsilon is some low number such as 0.0001. Both real and immaginary roots must be checked and if (2.3.14) is false for either type of root then the iteration is repeated. Otherwise the results of equation (2.3.10) are

calculated; this being the fifth step. Step six is a test for the existance of real and complex roots and the storage of these roots in their respective part of the answer array. The next step is a series of "booK Keeping" operations which increment the answer array pointer, decrement the order, and find new values for the real and immaginary part of the starting root approximation. Finally, if the order of the polynomial is zero then all the roots have been found [10]. Appendix A gives the psuedocode for ROOT LOCUS which uses the Newton - Ralphson method.

CHAPTER 3

USER'S GUIDE

3.1 COMPUTER CONSIDERATIONS

These programs were written for an Apple IIe. An 80 column text card is required (must be in card slot 3) as is 64K bytes of random access memory. The programs were developed on a single disk system but a second disk drive may be used with the program disk in disk drive 1 and the data disk in drive 2 provided the appropriate changes are made in the code. If a single drive system is used the program disk must not be write protected and more than one backup disk is recommended.

Hardcopy output is done with a General Electric model TXP-8100 dot matrix printer. Another printer may be used but this would necessitate changing command codes in the plotter output section (lines 420 through 430). The output to the printer is via a parallel output card with text and graphics capability (must be in slot 1).

ROOT LOCUS creates a text data file called ROOT.DAT which may be renamed as needed. The locus program calls a graphics sub-program (PLOTTER) when the data plotting function is executed. Both programs are written in Apple Integer Basic and are listed in the appendices as well as their psuedocode.

3.2 DATA ENTRY

ROOT LOCUS finds the roots of the open loop transfer function for a given system gain and then plots these roots in the S plane. It requires this characteristic equation to be in the form:

$$[a_0 \times (b_0 \times K)]s^n + \ldots + [a_n \times (b_n \times K)] = 0 \quad (3.2.1)$$

where K is the system gain. The range of system gain to be tested and a gain increment are also essential. The program guides the user through entering this data. After running one calculation with one set of gains or equations, another set of results may be obtained by entering the changes desired via the main menu screen. The one exception to this is when the plotting sub-program is run. APPLE graphics memory requirements prohibit the retention of the previously entered equations or the data generated by ROOT LOCUS. Therefore, the plotting program must get it's data from the disk.

ROOT LOCUS is written as a menu driven program in order

make it easier to use. The first action it takes when it is run is to display a title screen. After a short delay, a new screen is displayed which is the Main Menu (figure 2). This screen lists all the major activities available. To perform a specific task, the user selects that function from the list and enters its corresponding number via the keyboard. For example, entering a "1" causes the HELP screen to be shown on the monitor. If the number entered has no listed function (such as the number 9), then the program simply ignores this entry and waits for a new one.

The HELP screen can only be requested via the main menu and cannot be displayed when another function is active. It gives a brief description of the program's purpose and the form the characteristic equation must be in. It also states the data required by the ROOT LOCUS and the usual order this information is entered into the computer. This screen is not meant to be an exhaustive user's guide but rather to serve as a reminder to someone who has used this program before and has forgotten the details.

Before entering the systems's open loop transfer function it is necessary to first enter the degree of this equation (Main Menu choice number 2). The program can accommodate a polynomial of order twenty. Pressing the RETURN key enters the data and brings the main menu screen back. The degree of the polynomial can be altered at any time before returning to the main menu by backspacing with

You can...

1. Ask for help.
2. Enter the degree of the characteristic.
3. Enter the characteristics coefficients.
4. Enter starting, ending, and delta gains.
5. Solve for the root locus.
6. Display the calculation results.
7. Manage data files.
8. Exit.

Your choice is # _

FIGURE 2: MAIN MENU SCREEN

the arrow keys and typing over the error.

The coefficients of the characteristic may be now entered. Selecting item 3 on the main menu displays the Coefficient Entry Screen (refer to figure 3). The terms of the polynomial consist of a constant part and a gain multiplier. It is essential that the user has already entered the equation's degree before entering its coefficients. The constant part of the term is entered first and the gain multiplier second. It is not necessary that each number be entered as a decimal or that a zero preceed a decimal number. For example, 2.0 may be entered as 2 and 0.4567 may be entered as .4567. The highest order term of the characteristic is entered first and proceeds to the lowest order. It is imperative that all terms and their parts be shown even if it means entering a zero for the particular item.

The system gains screen is reached via the main menu (choice 4). A range is indicated by entering the starting and ending gains. The gain increment may also be indicated. The default value for these entries is one. The starting gain is entered first followed by the end gain and then the increment desired.

When all the data is entered to the users satisfaction, the program returns to the main menu where solving the equation is selected (item 5).

You will be asked to enter the highest order coefficient of the polynomial first, lowest order term last. Each coefficient is entered constant term first gain multiplier second. For example, to enter the equation

```
(2.0 + K)S**2 + (1.0 + 2.0K)S + 10.0K = 0
```

one enters:

```
2.0,  1.0 <------ highest order coefficient
1.0,  2.0
0.0,  10.0 <----- lowest order coefficient
```

Note that if either the constant or gain multiplier (or both) is absent a zero must still be entered for that term.

```
Please enter the S** __ term:
```

Figure 3: Coefficient Entry Screen

The program will display

One moment please ...

while the equation is being solved for its roots.

When the computer is finished, it will beep three times and then the message

Your data is on disk under ROOT.DAT
Press any Key to return to the main menu.

will be displayed on the screen. Pressing any Key returns the user to the main menu.

3.3 DATA DISPLAY

The ROOT LOCUS program can display the results of its calculations in either tabulated numeric form or plot them on the screen using cartesian coordinates. Both forms may be output to the printer. In any case, the data is displayed by selecting item 6 of the main menu. First, the user is prompted for the type of display desired by the Data Display Menu shown in figure 4.

Choosing item 1 of the display menu will plot the roots of the characteristic equation on the "S" plane. The ROOT LOCUS program does this through a subsidiary program called PLOTTER (see appendices C and D). PLOTTER first inquires for the file to plot by displaying the message

DATA DISPLAY MENU

You can...

1. Plot data using rectangular coordinates.
2. Display numeric data.
3. Print numeric data.
4. Return to main menu.

Your choice is # _

Figure 4: Main Data Display Menu

Current selected data file is ROOT.DAT
Enter data file name: <file name> or <RETURN>

As indicated above, the user may enter a new file name or press the RETURN key. The PLOTTER subprogram then plots the selected data file's graph on the screen and then the user may print a hardcopy if desired. Figure 5 gives such a plot.

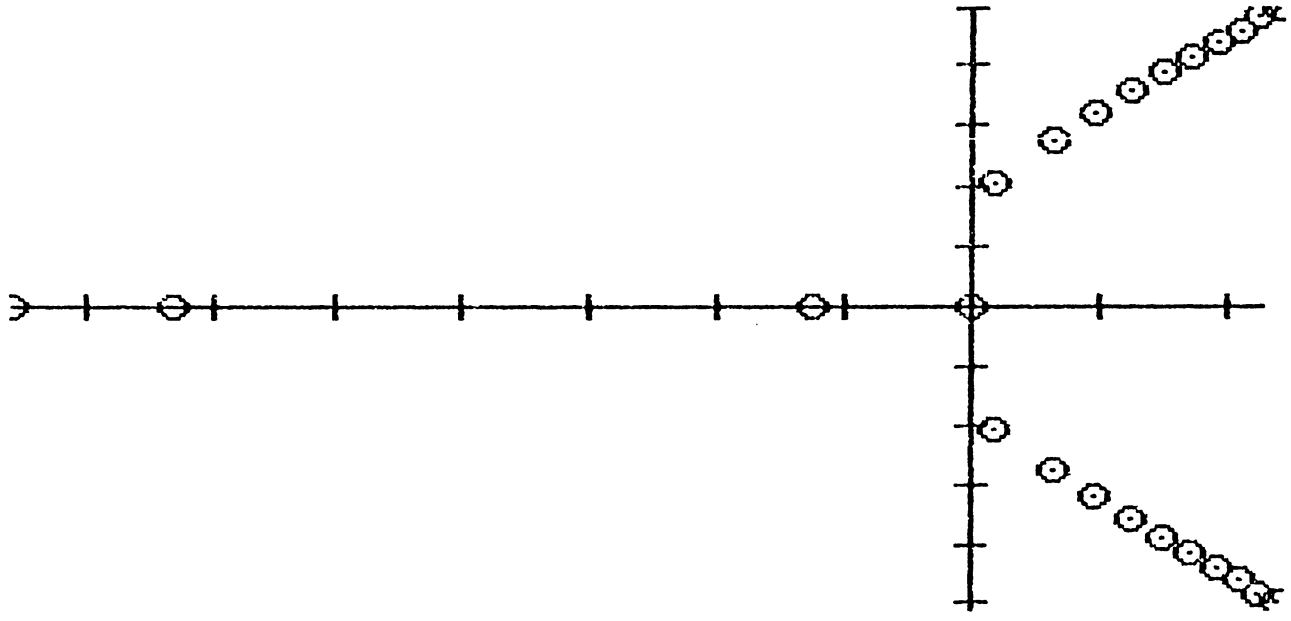
The plotter screen will also display a message asking the user to enter "R" to return to the ROOT LOCUS program or "P" to output the graph to the printer. Again note that using the PLOTTER function erases the equation and gain data used by the ROOT LOCUS program. This information must be reentered if it is to be used again. As mentioned before, this is due to the memory restrictions of the APPLE graphics and it is the only time data used by ROOT LOCUS must be manually reentered via the keyboard.

Selecting item 2 in the Data Display Menu places the numeric data generated by ROOT LOCUS onto the monitor screen. As in the PLOTTING program, the current data file selected to be displayed is shown and a request for a new file is made. ROOT LOCUS then retrieves the data from disk and outputs it to the screen. Selection of item 3, printing numeric data out, is the same as this function with the only difference being the device on which the information shown. The format of a typical display is given in figure 6.

The screen can only display 17 lines of data at a time which means that for most problems only some of the data can be displayed at any one time. At the end of the display will be the message

Press RETURN to continue, A to abort.

Pressing RETURN will always display the next page of data or, if there is no more data, the Main Data Display Menu. An "A" (abort) immediately displays the Main Data Display Menu. From this menu the user may select to print the numeric data by choosing function 3.



X Scale = .791713703 per unit.
Y Scale = 1.2495139 per unit.

File: ROOT.DAT

Figure 5: Typical Cartesian Coordinate Plot

File: ROOT.DAT

GAIN	SIGMA	J-OMEGA
0	0	0
0	-.999991084	0
0	-4.99993851	0
1	.1353734	2.5844633
1	.1353734	-2.5844633
1	-6	0
2	.500000645	3.4278282
2	.500000645	-3.4278282
2	-6	0
3	.774391877	4.01145369
3	.774391877	-4.01145369
3	-6	0
4	.999998259	4.47213857
4	.999998259	-4.47213857
4	-6	0
5	1.19418856	4.85886432
5	1.19418856	-4.85886432
5	-6	0
6	1.36611917	5.19541207
6	1.36611917	-5.19541207
6	-6	0
7	1.52128714	5.49531774
7	1.52128714	-5.49531774
7	-6	0
8	1.66329947	5.76709636
8	1.66329947	-5.76709636
8	-6	0
9	1.79464823	6.01649094
9	1.79464823	-6.01649094
9	-6	0
10	1.91713703	6.24756952
10	1.91713703	-6.24756952
10	-6	0

Figure 6: Numeric Data Display

CHAPTER 4

RESULTS

Let the transfer equation of a system be

$$\frac{GH}{1 + GH} = \frac{42K}{S\ (S+5)\ (S+1) + 42K} \qquad (4.0.1)$$

where K is the gain term [11]. The stability plot for this equation was done using the regular manual techniques with the results shown in figure 7. Appendix E gives the calculations needed to obtain this plot. This not too arduous task took approximately three hours to do both calculations and drafting.

Next, the characteristic equation obtained for the ROOT LOCUS program is

$$1 + GH = S^3 + 6S^2 + 5S + 42K \qquad (4.0.2)$$

Note that this is a third order polynomial. The first step is to enter this into the computer via item 2 of the main screen of ROOT LOCUS. Then the coefficients are entered.

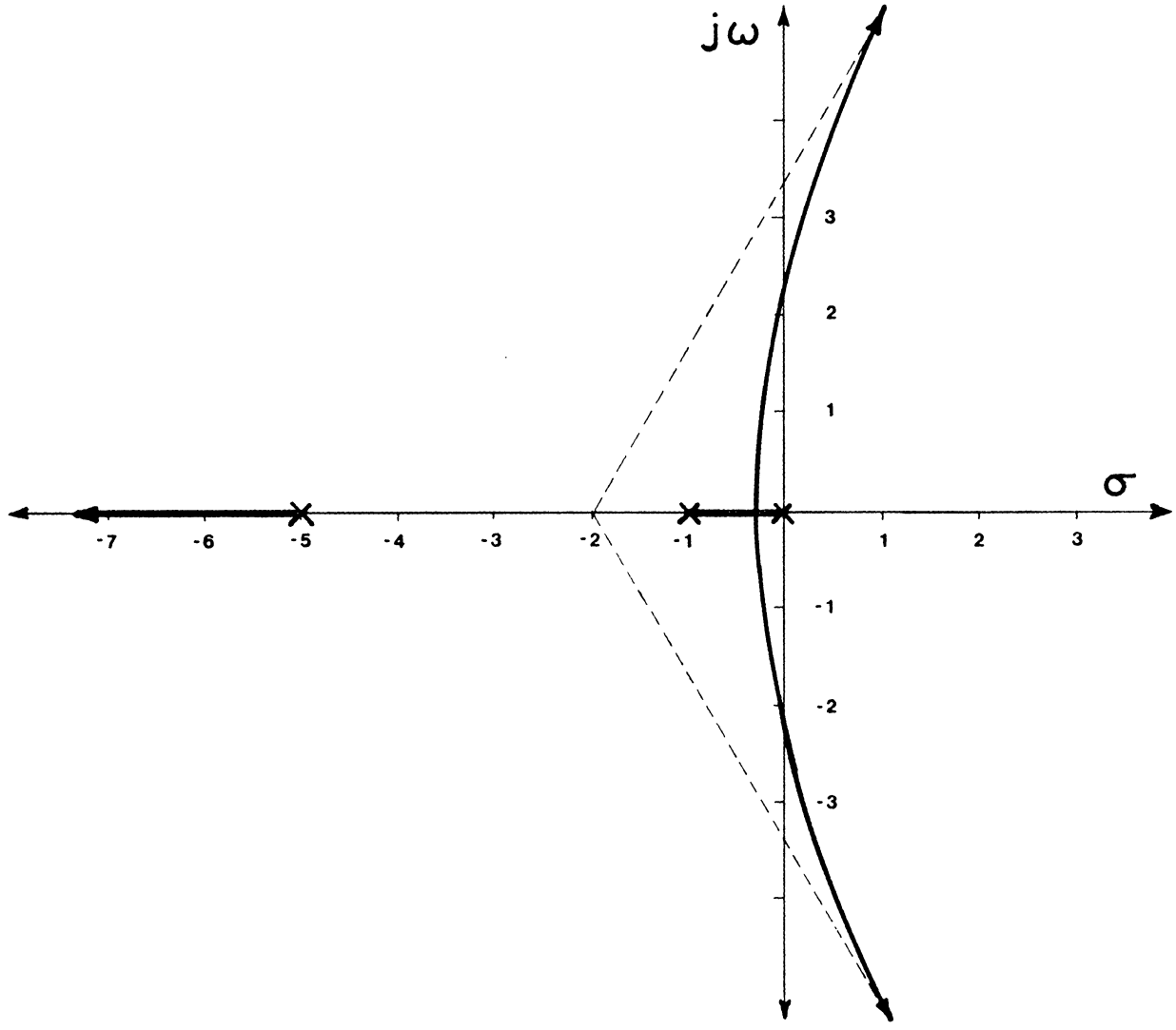


Figure 7: Manual Root Locus Plot

The Data Entry Screen (figure 3) should resemble figure 8 after this equation has been entered.

The initial range of gain tried was from 0 to 1000 with increments of 100. This was found to be too large of a range. A gain range of 0 to 100 (increments of 10) was used next and also found unsatisfactory. Finally, a range of 0 to 10 with increments of 1 was tried with the results shown in figure 9. The numeric results were also printed out and are shown on the following pages in table 1. The three runs of ROOT LOCUS took approximately two hours with the longest time being fourty minutes. This was for the calculations concerning the 0 to 10 range of system gains. The reason that this occurred is because the spacing between the roots was closer than, for example, the polynomial which had gains of 100 in it. ROOT LOCUS took more time in converging to a solution when calculating the roots.

It is significant, however, to note the time involved in using the APPLE as opposed to manual plotting of the root locus. Three root locus plots were obtained using a personal computer in two hours a opposed to one manual plot in three hours. Also, while working on the manual plot no other work could be done, whereas an engineer could be working on other tasks while the computer worked on the root locus plot. The results can be seen to be comparable in accuracy.

You will be asked to enter the highest order coefficient of the polynomial first, lowest order term last. Each coefficient is entered constant term first gain multiplier second. For example, to enter the equation

$$(2.0 + K)S^{**}2 + (1.0 + 2.0K)S + 10.0K = 0$$

one enters:

```
2.0,  1.0 <------- highest order coefficient
1.0,  2.0
0.0,  10.0 <------ lowest order coefficient
```

Note that if either the constant or gain multiplier (or both) is absent a zero must still be entered for that term.

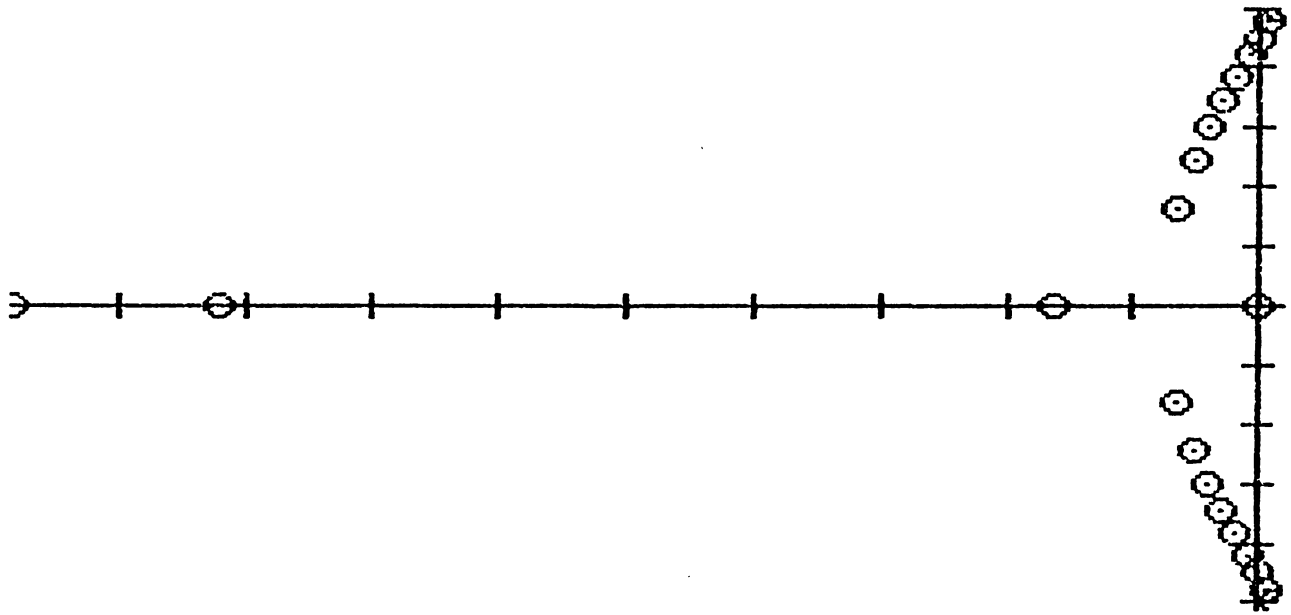
```
Please enter the S**3 term:  1.0,  0.0
Please enter the S**2 term:  6.0,  0.0
Please enter the S**1 term:  5.0,  0.0
Please enter the S**0 term:  0.0,  42.0
```

Figure 8: Entering Coefficients

File: ROOT.DAT

GAIN	SIGMA	J-OMEGA
0	0	0
0	-1	0
0	-5	0
.1	-.403551518	.803705961
.1	-.403551518	-.803705961
.1	-6	0
.2	-.320243003	1.21026886
.2	-.320243003	-1.21026886
.2	-6	0
.3	-.246228535	1.49236157
.3	-.246228535	-1.49236157
.3	-6	0
.4	-.179214795	1.71632639
.4	-.179214795	-1.71632639
.4	-6	0
.5	-.117707231	1.90501396
.5	-.117707231	-1.90501396
.5	-6	0
.6	-.0606693995	2.06954332
.6	-.0606693995	-2.06954332
.6	-6	0
.7	-7.34864549E-03	2.21629832
.7	-7.34864549E-03	-2.21629832
.7	-6	0
.8	.0428214091	2.34933138
.8	.0428214091	-2.34933138
.8	-6	0
.9	.0902792345	2.47139674
.9	.0902792345	-2.47139674
.9	-6	0

Table 1: Numeric Results Of ROOT LOCUS



X Scale = .609027924 per unit.
Y Scale = .494279348 per unit.

File: ROOT.DAT

Figure 9: ROOT LOCUS Plot

CHAPTER 5

CONCLUSIONS

The choice of the algorithm to calculate the root locus was dependent on several key criteria. Accuracy and reliability of the computer generated plots were certainly the most critical. Conservation of memory was important since most personal computers have limited memory available. Algorithm simplicity was also a factor in evaluating any solution method. Simplicity affects the amount of memory needed by the program, how fast the program executes, ease of use, and the maintainability of the program.

Two algorithms were examined, these being the Lin-Bairstow and Newton-Ralphson methods. Using the criteria given above, the Newton-Ralphson algorithm was chosen over the Lin-Bairstow technique. Since the function representing a system will always be continuous, both methods provided accurate and reliable data. However, Newton-Ralphson used less memory than Lin-Bairstow whose matrix sizes must be eight times the order of the polynomial being evaluated [11]. Newton-Ralphson is also a simpler method. For example, it evaluates only one

derivative as opposed to Lin-Bairstow's four. Finally, Newton-Ralphson solved for the roots of a trial third order polynomial (representing a system transfer function) roughly twice as fast as Lin-Bairstow on an Apple IIe.

The novelty of the ROOT LOCUS program lies in its use of numerical methods to plot a root locus diagram. This is a significant departure from many similar personal computer programs currently available which try to adapt manual proceedures to the computer. Laboratory tests and use in the actual design of some heavy industry systems have demonstrated the ROOT LOCUS's value and accuracy.

FOOTNOTES

[1] Steeper, D. E. ; Single-stand And Tandem Cold Mill Dynamic Simulation; General Electric Co. ; Schenectady; 1984.

[2] Kaufman, G. A. ; Analysis Of Feedback Control For A Voltage Source Inverter-induction Machine Drive System; General Electric Co. ; Schenectady; 1981.

[3] Mounfield, Pratt; ROOTL. FOR; General Electric Co. ; Salem; 1981.

[4] Klerer, Melvin and Korn, Granino A. ; Digital Computer User's Handbook; McGraw - Hill Book Co. ; New York; 1967; pg. 2-68.

[5] Digital Computer User's Handbook; pp. 2-64 to 2-66.

[6] Tierney, John A. ; Calculus and Analytic Geometry; Allyn and Bacon, Inc. ; Boston; 1972; pp. 252 - 253.

[7] Scarborough, James B. ; Numerical Mathematical Analysis; Johns Hopkins Press; Baltimore; 1955; pg. 192.

[8] Digital Computer User's Handbook; pg. 2-59.

[9] Ames, William F. ; Numerical Methods for Partial Differential Equations; Academic Press; New York; 1977; pg. 106.

[10] Digital Computer User's Handbook; pp. 2-59 to 2-64.

[11] Mack, Donald R. ; The General Electric A Course Engineering Analysis; General Electric Co. ; Norwalk; 1982; pp. 67 - 75.

[12] Shoup, T. E. ; Applied Numerical Methods for the Microcomputer; Prentice-Hall; New Jersey; 1984; pp. 37 - 39

APPENDIX A
ROOT LOCUS PROGRAM PSUEDOCODE

```
/* MAIN PROGRAM */

   OUTPUT TITLE PAGE
   WAIT A FEW SECONDS
   DEFINE VARIABLES AND ARRAYS
   SET CONSTANT DEFAULTS

   DO WHILE (NOT EXIT)
     CLEAR SCREEN AND HOME CURSOR
     DO WHILE (KEYBOARD NOT IN RANGE)
       OUTPUT MAIN MENU TO SCREEN
       PRINT "Enter action code ... "
       INPUT KEYBOARD
       CONVERT ASCII TO AN INTEGER NUMBER
     ENDDO
     CASE 1: RUN HELP SUBROUTINE
          2: RUN CHARACTERISTIC DEGREE ENTRY SUBROUTINE
          3: RUN CHARACTERISTIC COEFFICIENT ENTRY
             SUBROUTINE
          4: RUN GAIN ENTRY SUBROUTINE
          5: RUN SOLVE ROOT LOCUS SUBROUTINE
          6: RUN DATA DISPLAY SUBROUTINE
          7: RUN DATA FILE MANAGEMENT SUBROUTINE
          8: EXIT
   ENDDO
   END

/* HELP SUBROUTINE */

   RETURN TO TEXT MODE AND CLEAR SCREEN
   OUTPUT HELP PAGE TO SCREEN
   INPUT KEYBOARD
   RETURN TO MAIN PROGRAM
```

```
/* DEGREE ENTRY SUBROUTINE */

   RETURN TO TEXT MODE AND CLEAR SCREEN
   OUTPUT DATA ENTRY INSTRUCTIONS TO SCREEN
   PRINT "Please enter the characterisic equations degree"
   INPUT DEGREE FROM KEYBOARD
   NUMBER OF EQUATION TERMS = DEGREE + 1
   RETURN TO MAIN PROGRAM

/* COEFFICIENT ENTRY SUBROUTINE */

   RETURN TO TEXT MODE AND CLEAR SCREEN
   OUTPUT DATA ENTRY INSTRUCTIONS TO SCREEN
   SKIP A LINE ON SCREEN

   FOR I = NUMBER OF TERMS TO 1 IN STEPS OF -1
     PRINT "Enter the coefficients of the s**"(I-1)" term."

     INPUT CONSTANT AND GAIN MULTIPLIER PARTS OF TERM
   NEXT I

   RETURN TO MAIN PROGRAM

/* GAIN ENTRY SUBROUTINE */

   RETURN TO TEXT MODE AND CLEAR SCREEN
   PRINT "Enter starting, ending, and delta gains."
   INPUT STARTING GAIN, ENDING GAIN, AND DELTA GAIN
   RETURN TO MAIN PROGRAM

/* ROOT SOLVER SUBROUTINE */

   /* In the interests space, the following              */
   /* abbreviations are used:                            */
   /*                                                    */
   /*     N:  order of the polynomial being solved       */
   /*     Xo: real component of the starting root.       */
   /*     Yo: imaginary component of the starting root.  */
   /*     X1: real component of the delta root.          */
   /*     Y1: imaginary component of the delta root.     */

   FOR GAIN = GAIN TO ENDING GAIN IN STEPS OF DELTA GAIN

     FOR TERM = 1 TO NUMBER OF TERMS
       COEFFICIENT = CONSTANT COMPONENT + GAIN (MULTIPLIER)
     NEXT TERM
```

```
/* SET UP INITIAL ROOT ESTIMATE */

IF a(n-2) IS NOT ZERO THEN

  /* Use the three lowest terms to form a        */
  /* a quadratic and estimate the first root.    */

  RADICAL = a(n-1)^2 - 4(a(n-2))(a(n))
  IF RADICAL < 0 THEN

          sqrt(|RADICAL|)
     Xo = ---------------
          [2(a(n-2))]

          -a(n-1)
     Yo = ---------------
          [2(a(n-2))]
  ELSE
     Xo = 0
          [ -a(n-1) + sqrt(RADICAL) ]
     Yo = ----------------------------
                  [2(a(n-2))]
  ENDIF
ELSE
  Xo = 1.5                ! if a(n-2) is zero then
  Yo = 1.5                !   arbitrarily set to 1.5
ENDIF

N = DEGREE
INITIALIZE ANSWER ARRAY POINTER TO 1
B(0) = A(0)                          ! approximation real
C(0) = 0                             ! approximation imag
H(0) = 0                             ! 1st derivative real
G(0) = N * A(0)                      ! 1st derivative imag

FOR N = DEGREE TO 0 IN STEPS OF -1
  FOR ITERATION = 1 TO 500

    /* NESTED MULTIPLICATION TO EVALUATE f(x)    */
    /* AND f'(x)                                 */

    FOR I = 1 TO N
      B(I) = A(I) + [(Xo * B(I-1)] - [Yo * C(I-1)]
      C(I) = [Xo * C(I-1)] + [Yo * B(I-1)]
      G(I) = [(N-1) * A(I)] + [Xo * G(I-1)]
             - [Yo * H(I-1)]
      H(I) = [Xo * H(I-1)] + [Yo * G(I-1)]
    NEXT I
```

```
  /* FIGURE ERROR */

  DENOMINATOR = G(N-1)² + H(N-1)²

        Xo - { [B(N) * G(N-1)] + [C(N) * H(N-1)] }
  X1 = ------------------------------------------
                        DENOMINATOR

        Yo - { [C(N) * G(N-1)] - [B(N) * H(N-1)] }
  Y1 = ------------------------------------------
                        DENOMINATOR

  IF [¦(X1 - Xo) / Xo¦ < EPSILON] OR
    IF [¦(Y1 - Yo) / Yo¦ < EPSILON] THEN
      Xo = X1
      Yo = Y1
  ELSE
      ITERATION = 500
  ENDIF
NEXT ITERATION

IF [¦Y1 / X1¦ - EPSILON] > 0 THEN
  /* SAVE REAL ROOTS IN ANSWER ARRAY */
  REAL ROOT (ANSWER POINTER) = X1
  REAL ROOT (ANSWER POINTER + 1 = X1

  /* SAVE IMAGINARY ROOTS */
  IMAG ROOT (ANSWER POINTER) = Y1
  IMAG ROOT (ANSWER POINTER + 1) = -Y1
  INCREMENT ANSWER POINTER BY 2
  DECREMENT ORDER BY 2
  Xo = X1
  Yo = Y1

  /* SYNTHETIC DIVISION TO ELIMINATE ROOTS     */
  /* JUST FOUND                                */

  Xb = X1 + X1
  Xc = - (X1² + Y1²)
  A(1) = A(1) + [Xb * A(0)]
  FOR I = 2 TO N
    A(I) = A(I) + [Xb * A(I-1)] + [Xc * A(I-2)]
  NEXT I

ELSE

  REAL ROOT (ANSWER POINTER) = X1      ! real root
  IMAG ROOT (ANSWER POINTER) = 0       ! imag root
  INCREMENT ANSWER POINTER BY 1
  DECREMENT ORDER BY 1
  Xo = X1
```

```
          Yo = Y1

          /* SYNTHETIC DIVISION TO ELIMINATE ROOTS     */
          /* JUST FOUND                                */

          FOR I = 1 TO N
            A(I) = A(I) + [X1 * A(I-1)]
          NEXT I

        ENDIF
      NEXT N
    NEXT GAIN

    /* SAVE ANSWER ARRAY TO DISK */

    NUMBER OF ELEMENTS IN FILE = ANSWER POINTER - 1
    OPEN DATA FILE
    WRITE NUMBER OF ELEMENTS TO DATA FILE

    FOR I = 1 TO NUMBER OF ELEMENTS
      WRITE GAIN(I), REAL(I), IMAGINARY(I) TO DISK DATA FILE
    NEXT I

    CLOSE DATA FILE
    PRINT "Your data is in ROOT.DAT."
    DISK FILE = "ROOT.DAT"
    RETURN TO MAIN PROGRAM

/* DISPLAY DATA SUBROUTINE */

    RETURN TO TEXT MODE AND CLEAR SCREEN
    DO WHILE (NOT EXIT)
      OUTPUT DISPLAY MENU TO SCREEN
      PRINT "Current data file is ", DISK FILE
      DO WHILE (KEYBOARD NOT IN RANGE)
        PRINT "Enter action code: "
        INPUT KEYBOARD
        CONVERT ASCII TO INTEGER
      ENDDO
      CASE 1: RUN PLOTTER PROGRAM
           2: RUN DISPLAY SUBROUTINE
           3: RUN PRINT DATA SUBROUTINE
           4: EXIT
    ENDDO
    RETURN TO MAIN PROGRAM

/* DISPLAY DATA CASE 1 - CARTESIAN COORDINATE PLOT */

    CALL PLOTTER PROGRAM
    RETURN TO MAIN PROGRAM
```

```
/* DISPLAY DATA CASE 2 - DATA DISPLAY SUBROUTINE */

   INITIALIZE ELEMENT COUNTER
   PRINT "Data file name: "
   INPUT FILE NAME
   IF (FILE NAME NOT BLANK) THEN
     DISK FILE = FILE NAME
   ENDIF

   CLEAR ABORT AND DONE FLAGS
   OPEN DISK FILE
   READ NUMBER OF RECORDS FROM DISK
   DO WHILE [(NOT DONE) AND (NOT ABORT)]
     PRINT HEADER TO SCREEN
     FOR LINE = 3 TO 20
       INPUT ANSWER'S GAIN, REAL, AND IMAGINARY PARTS
       PRINT GAIN, REAL, AND IMAGINARY PARTS TO SCREEN
       IF ELEMENT COUNTER ≥ NUMBER OF RECORDS THEN
         SET DONE FLAG
         LINE = 20
       ELSE
         ELEMENT COUNTER = ELEMENT COUNTER + 1
       ENDIF
     NEXT LINE
     PRINT "Press RETURN  to continue, A to abort."
     INPUT KEYBOARD
     IF KEYBOARD = A THEN
       SET ABORT FLAG
     ENDIF
   ENDDO

   CLOSE FILE
   RETURN TO DISPLAY MENU

/* DISPLAY DATA CASE 3 - DATA PRINTOUT SUBROUTINE */

   RETURN TO TEXT MODE AND CLEAR  SCREEN

   INITIALIZE ELEMENT COUNTER
   PRINT "Data file name: "
   INPUT FILE NAME
   IF (FILE NAME NOT BLANK) THEN
     DISK FILE = FILE NAME
   ENDIF

   CLEAR ABORT AND DONE FLAGS
   OPEN DISK FILE
   READ NUMBER OF RECORDS FROM DISK
```

```
    DO WHILE [(NOT DONE) AND (NOT ABORT)]
      ENABLE PRINTER
      DISABLE SCREEN AND SET PRINTER FOR 80 COLUMNS
      PAGE = 1
      PRINT "File: ", DISK FILE TO PRINTER
      PRINT HEADER TO PRINTER

      FOR LINE = 9 TO 55
        INPUT ANSWER'S GAIN, REAL, AND IMAGINARY PARTS
        PRINT GAIN, REAL, AND IMAGINARY PARTS TO PRINTER
        IF ELEMENT COUNTER ≥ NUMBER OF RECORDS THEN
          SET DONE FLAG
          LINE = 55
        ELSE
          ELEMENT COUNTER = ELEMENT COUNTER + 1
        ENDIF
      NEXT LINE
      IF (NOT DONE) THEN
        INCREMENT PAGE COUNTER
      ENDIF
    ENDDO

    CLOSE FILE
    DISABLE PRINTER
    ENABLE SCREEN
    RETURN TO DISPLAY MENU

/* DATA FILES SUBROUTINE */

    DO WHILE (NOT EXIT)
      RETURN TO TEXT AND CLEAR SCREEN
      PRINT FILE MANAGEMENT MENU TO SCREEN
      DO WHILE (KEYBOARD NOT IN RANGE)
        PRINT "Enter action code: " TO SCREEN
        INPUT KEYBOARD
        CONVERT ASCII TO INTEGER
      ENDDO

      /* CASE 1 - RENAME A FILE */

      PRINT "Original file name: " TO SCREEN
      INPUT OLD FILE NAME
      PRINT "New file name: " TO SCREEN
      INPUT NEW FILE NAME
      RENAME OLD FILE TO NEW NAME

      /* CASE 2 - DELETE A FILE */

      PRINT "File to be deleted: " TO SCREEN
      INPUT FILE NAME
      PRINT "Confirm order to delete " FILE NAME " (Y/N): "
```

```
  INPUT KEYBOARD
  IF (KEYBOARD = Y) THEN
    DELETE FILE
  ELSE
    PRINT "ABORT DELETION" TO SCREEN
  ENDIF

ENDDO
RETURN TO MAIN PROGRAM
```

APPENDIX B
ROOT LOCUS PROGRAM LISTING

NOTE: DELETE ALL COMMENTS PRECEDED BY AN EXCLAMATION MARK.
The only comment delimiter the APPLE IIe recognizes is REM.
The comments with the exclamation mark are for clarity only.

```
4   REM ---------------- ROOT LOCUS TITLE PAGE -------------

5   IF ROOT = 1 THEN  GOTO 21:
    REM     If plotting return, go to display
6   HOME :                                  ! Clear screen
    TEXT :                                  ! Text mode
    VTAB 6:                                 ! Tab down 6 lines
    INVERSE                                 ! Inverse screen

7   HTAB 25: PRINT "                                    "
8   HTAB 25: PRINT " ROOT LOCUS STABILITY PLOT "
9   HTAB 25: PRINT "                                    "
10   HTAB 25: PRINT "      By: Ben C.  Svrcek      "
11   HTAB 25: PRINT "                                    "
12   HTAB 25: PRINT "    DOS Version 3.3; 1985    "
13   HTAB 25: PRINT "                                    "
14   HTAB 25: PRINT "                                    "
15   HTAB 25: PRINT "                                    "
16   HTAB 25: PRINT " Written for the Apple IIe "
17   HTAB 25: PRINT "  using an 80 column card  "
18   HTAB 25: PRINT "                                    "
19   NORMAL :
     FOR I = 1 TO 2500:
     NEXT I:
     REM          Leave entry header page onscreen for a while
```

```
20  REM -------- DECLARE VARIABLES AND ARRAYS -------------

21 D$ =  CHR$ (4):                          ! Control D
   F$ = "ROOT.DAT":                         ! File name is ROOT.DAT
   SG = 1.0:                                ! Default starting gain
   EG = 1.0:                                ! Default ending gain
   DG = 1.0:                                ! Default delta gain
   EP = 0.0000000001:                       ! Convergence test value
   CER = 0.0001:                            ! Complex no. test value
   LF$ =  CHR$ (10):                        ! Line feed
   FF$ =  CHR$ (12)                         ! Form feed

23  DIM A(20): REM      Polynomial coefficient array
24  DIM ANS(500,3): REM Answer array -
                           1) Gain,
                           2) Real,
                           3) Immaginary
27  DIM B(20): REM      1st approximation array (real &
                           remainder)
30  DIM C(20): REM      1st approximation array (imaginary)
31  DIM COF(20,2):  REM Coefficient entry array -
                           1) constant,
                           2) multiplier
32  DIM G(20): REM      1st derivative real array
33  DIM H(20): REM      1st derivative imaginary array



105  REM ----------------- MAIN PROGRAM -------------------


110  HOME : HTAB 30:                 ! Do until EXIT is chosen
       PRINT "ROOT LOCUS MAIN MENU":
       VTAB 3: HTAB 17: PRINT "You can...":
       PRINT : HTAB 21: PRINT "1. Ask for help.":
       PRINT : HTAB 21:
       PRINT "2. Enter the degree of the characteristic."
115    PRINT : HTAB 21:
       PRINT "3. Enter the characteristics coefficients.":
       PRINT : HTAB 21:
       PRINT "4. Enter starting, ending, and delta gains.":

       PRINT : HTAB 21:
       PRINT "5. Solve for the root locus."
120    PRINT : HTAB 21:
       PRINT "6. Display the calculation results.":
       PRINT : HTAB 21:
       PRINT "7. Manage data files.":
       PRINT : HTAB 21:
```

```
        PRINT "8. Exit.":
        PRINT : PRINT
145     HTAB 17: PRINT "Your choice is # ";:
        GET C$:                     ! Wait for Keyboard
        C =  ASC (C$) - 48:         ! Change ASII to #
        IF C < 1 OR C > 8 THEN  GOTO 145
155     ON C GOSUB 500,             ! Case 1: HELP
                   1000,            !       2: DEGREE ENTRY
                   1100,            !       3: COEFFICIENT ENTRY
                   1500,            !       4: GAIN ENTRY
                   2000,            !       6: SOLVE FOR ROOT
                   3100,            !       7: DATA DISPLAY
                   4000,            !       8: FILE MANAGEMENT
                   10000:           !       9: EXIT
     GOTO 110:                      ! End do loop



500  REM ----------- HELP INSTRUCTIONS SUBROUTINE ---------


505  TEXT : HOME : VTAB 4:
     PRINT "ROOT LOCUS finds the roots of a control system's
            open loop transfer function": PRINT "(1 + GH)
            and plots them on the 'S' plane.  The equation
            is of the form:"
510  PRINT : HTAB 13:
     PRINT
     "(a0+b0K)S**(n) + (a1+b1K)S**(n-1) +...+ (an+bnK) = 0":
     PRINT "where": HTAB 22: PRINT "ax is the constant part
            of the term.": HTAB 22: PRINT "bx is the gain
            multiplier."
515  HTAB 22:
     PRINT "K  is the gain factor.": PRINT :
     PRINT "The transfer function will also be refered to
            in this program as the system's":
            PRINT "CHARACTERISTIC.  You will be asked to
            enter:":
            PRINT : HTAB 20
520  PRINT "* The degree (n) of this polynomial.":
     PRINT : HTAB 20:
     PRINT "* The polynomial's coefficients.":
     PRINT : HTAB 20:
     PRINT "* Starting, ending, and change in gain."
525  VTAB 24: HTAB 27:
     PRINT "Press any key to exit ...";:
     GET A$:
     RETURN
```

```
1000  REM ------- POLYNOMIAL DEGREE ENTRY SUBROUTINE -------

1005  TEXT : HOME : VTAB 8:
      PRINT "The 'degree' of a polynomial is the largest
            power any one term in the equation":
      PRINT "is raised to.  For example, the expression:":
      PRINT
1010  HTAB 30:
      PRINT "aX**2 + bX**1 + c = d":
      PRINT "has a degree of 2.":
      PRINT : HTAB 19:
      INPUT "Please enter the characteristic's degree:";D:
      NTERMS = D + 1:
      RETURN :

1100  REM ------------ COEFFICIENT ENTRY SUBROUTINE --------

1105  TEXT : HOME :
      PRINT "You will be asked to enter the highest order
            coefficient of the polynomial": PRINT "first,
            lowest order term last.  Each coefficient is
            entered constant term"
1110  PRINT "first gain multiplier second.  For example,
            to enter the equation": PRINT : HTAB 20:
      PRINT "(2.0 + K)S**2 + (1.0 + 2.0K)S + 10.0K = 0":
      PRINT "one enters": HTAB 20:
      PRINT "2.0, 1.0 <------ highest order coefficient":
      HTAB 20
1115  PRINT "1.0, 2.0": HTAB 20:
      PRINT "0.0, 10.0 <----- lowest order coefficient":
      PRINT : PRINT "Note that if either the constant or
            gain multiplier (or both) is absent a zero":
      PRINT "must still be entered for that term.": PRINT
1125  FOR TC = 1 TO NTERM:
        PRINT "Please enter the S**"(NTERM - TC)" term: ";:
        INPUT " ";COF(TC,1),COF(TC,2):
      NEXT TC
1135  RETURN
```

```
1500  REM ---------------- GAIN ENTRY SUBROUTINE ------------

1505  TEXT : HOME : VTAB 9:
      PRINT "Please enter the starting, ending, and delta
             gains, making sure there is a comma between
             each piece of data.": PRINT : HTAB 20:
      INPUT SG,EG,DG:
      RETURN



2000  REM ------- ROOT SOLVER AND PLOTTER SUBROUTINE -------

2005  HOME : VTAB 12: HTAB 30:
      PRINT "One moment please ...":AP = 1

2010  FOR KGA = SG TO EG STEP DG ! Find roots for gain range

2020    FOR TC = 1 TO NTERM:
          A(TC) = COF(TC,1) + (COF(TC,2) * KGA):
        NEXT TC:
        REM Determine coeffients with gain

2030    N = D + 1:                    ! Find initial trial root
        IF A(N - 2) = 0 THEN          ! If not quadratic then
          X0 = 1.5:                   !    Arbitrarily assign
          Y0 = 1.5:                   !    roots
        GOTO 2050                     ! Else
2040      RAD = (A(N-1) * A(N-1)) - (4.0 * A(N-2) * A(N)):
          IF RAD < 0 THEN             !    If negative radical
            Y0 =  SQR ( ABS (RAD)) / (2.0 * A(N-2)):
            X0 =  -A(N-1) / (2.0 * A(N-2)):
          GOTO 2050
2045        X0 = ( -A(N-1) +  SQR (RAD)) / (2.0 * A(N-2)):
            Y0 = 0.0
2050      K = 1:      B(1) = A(1):   ! Initalize all arrays
          C(1) = 0:   H(1) = 0:
          G(1) = D * A(1)

2055      FOR N = N TO 2 STEP  -- 1

2060        FOR ITER = 1 TO 1000
2065          FOR I = 2 TO N:    ! Evaluate f(x) & f'(x)
                B(I) = A(I) + (X0 * B(I-1)) - (Y0 * C(I-1)):
                C(I) = (X0 * C(I-1)) + (Y0 * B(I-1)):
```

```
              G(I) = ( (N-1) * A(I)) + (X0 * G(I-1))
                     - (Y0 * H(I-1)):
              H(I) = (X0 * H(I-1)) + (Y0 * G(I-1)):
            NEXT I

2070        VTAB 20: HTAB 27:
            PRINT "Gain: "KGA"    Answer Element: "AP

2100        DEN = (G(N-1) * G(N-1)) + (H(N-1) * H(N-1)):
            IF DEN = 0 THEN
              X1 = 0.0:
              Y1 = 0.0:
            GOTO 2115
2110          X1 = X0 - ((B(N) * G(N-1)) +
                  (C(N) * H(N -1))) / DEN:
              Y1 = Y0 - ((C(N) * G(N-1)) -
                  (B(N) * H(N-1))) / DEN
2115        IF ((X0 = 0) AND ( ABS (X1 - X0) > EP) OR
              ((Y0 = 0) AND ( ABS (Y1 - Y0) > EP)) THEN
                 X0 = X1:
                 Y0 = Y1:
                 NEXT ITER
2120        IF (X0 <  > 0) THEN
              IF ABS ((X1 - X0) / X0) > EP THEN
                X0 = X1:
                Y0 = Y1:
                NEXT ITER
2130        IF (Y0 <  > 0) THEN
              IF ABS ((Y1 - Y0) / Y0) > EP THEN
                X0 = X1:
                Y0 = Y1:
                NEXT ITER

2145      IF X1 = 0 THEN  GOTO 2165
2150        IF ( ABS (Y1 / X1) < CER) THEN GOTO 2165

                                      ! Complex conjugate root
2155          ANS(AP,1) = KGA:        ! Save the roots
              ANS(AP + 1,1) = KGA:
              ANS(AP,2) = X1:
              ANS(AP + 1,2) = X1:
              ANS(AP,3) = Y1:
              ANS(AP + 1,3) = -Y1:
              N = N - 1:              ! Decrement degree
              X0 = X1:                ! Save new intial root
              Y0 = Y1:

              XB = 2 * X1:            ! Synthetic division
              XC = -((X1 * X1) + (Y1 * Y1)):
              A(2) = A(2) + (XB * A(0)):
              AP = AP + 2       ! Increment answer pointer
```

```
2160          FOR I = 3 TO N:
                A(I) = A(I) + (XB * A(I-1)) + (XC * A(I-2)):
              NEXT I:
              GOTO 2180

2165        ! Else of line 2150  -  real roots
              ANS(AP,1) = KGA:
              ANS(AP,2) = X1:
              ANS(AP,3) = 0:
              X0 = X1:
              Y0 = Y1:
              AP = AP + 1
2175         FOR I = 2 TO (N-1): ! Synthetic division
              A(I) = A(I) + (X1 * A(I-1)):
             NEXT I
2180      NEXT N: REM    Endif of line 2150
2190   NEXT KGA

       ! Save data to disk

3000   ENUM = AP - 1:                    ! Number of records
       PRINT D$;"OPEN ROOT.DAT":         ! Open file ROOT.DAT
       PRINT D$;"WRITE ROOT.DAT":
       PRINT ENUM:                       ! Write number of records
       FOR I = 1 TO ENUM:                ! Save the answer array
          PRINT ANS(I,1):
          PRINT ANS(I,2):
          PRINT ANS(I,3):
       NEXT I
3020   PRINT D$;"CLOSE ROOT.DAT":        ! Close the file
       F$ = "ROOT.DAT":

       PRINT  CHR$ (7), CHR$ (7), CHR$ (7), CHR$ (7):
                                         ! Bell to notify user
       HOME : VTAB 12:
       PRINT "Your data is on disk under ROOT.DAT":
       PRINT "Press any key to return to the main menu.";:
       GET C$:
       PRINT C$:
       RETURN



3100   REM  ---------------- DISPLAY DATA ROUTINES -----------


3105   HOME : VTAB 5: HTAB 31:
       PRINT "DATA DISPLAY MENU":
       PRINT : HTAB 22:
```

```
      PRINT "You can. . . ":
      PRINT : HTAB 22:
      PRINT "1. Plot data using rectangular coordinates. ":
      PRINT : HTAB 22: PRINT "2. Display numeric data. "
3110  PRINT : HTAB 22: PRINT "3. Print numeric data. ":
      PRINT : HTAB 22: PRINT "4. Return to main menu. ":
      PRINT : PRINT
3115  HTAB 30: PRINT "Your choice is # ";:
      GET C$: PRINT C$:
      C =  ASC (C$) - 48:
      IF C < 1 OR C > 4 THEN  GOTO 3115
3140     ON C GOSUB 3200,              ! Case 1: Plot data
                    3500,              !       2: CRT data disp
                    3600:              !       3: Print data
      IF C <  > 4 THEN  GOTO 3105      !       4: Exit
3150  RETURN


3199  REM - - - - - - - - - - - - plotter routine

3200  PRINT D$;"RUN PLOTTER": REM    Call PLOTTER subroutine

3499  REM - - - - - - - - - - - - crt data display

3500  ECTR = 1:                      ! Initialize record ctr
      HOME : VTAB 12: HTAB 20:
      PRINT "Current selected data file is "F$:
      PRINT : HTAB 20:
      INPUT "Enter data file name: ";A$:    ! Get Data file
      IF A$ <  > "" THEN F$ = A$   ! Check if default

3505  ABT = 0: DNE = 0:              ! Clear ABORT & DONE Flags
      PRINT D$;"OPEN "F$:            ! Open file
      PRINT D$;"READ "F$:
      INPUT ENUM                     ! Input # records in file

      ! Do until ABORT or DONE

3515     HOME : HTAB 8:              ! Clear screen/show header

         PRINT "GAIN" SPC( 17)"SIGMA" SPC( 15)"J-OMEGA":
         PRINT D$:                      ! Set for data input
         PRINT D$;"READ "F$

3520     FOR LINE = 3 TO 20:
            INPUT K,R,I:             ! Retrieve data from disK
            A = 15 -  LEN (STR$ (K)):! Find spacing for disp
            B = 20 -  LEN (STR$ (R)):
            C = 20 -  LEN (STR$ (I)):
            VTAB LINE:               ! Drop to correct line
            PRINT  SPC( A)K SPC( B)R SPC( C)I: ! Output data
```

```
            IF ECTR >  = ENUM THEN    ! If ctr = # records
               DNE = 1:               !   Set DONE flag
               LINE = 20              !   Drop to last line
3525        IF ECTR < ENUM THEN       ! Else
               ECTR = ECTR + 1        !   Increment ctr
3530     NEXT LINE:                   ! Endif

         PRINT D$:                    ! Set for CRT entry
         VTAB 24:
         PRINT "Press RETURN to continue, A to abort.";:
         GET C$:                      ! Get keyboard entry
         IF C$ = "A" THEN ABT = 1     ! If A then set abort

3540  IF ( NOT ABT) AND ( NOT DNE) THEN GOTO 3515  ! End do

3545  PRINT D$;"CLOSE "F$:            ! Close data file
      RETURN                          ! Return to sub menu

3599  REM -  -  -  -  -  -  -  -  -  -  - print numeric data

3600  ECTR = 1: HOME : VTAB 12: HTAB 20:
      PRINT "Current selected data file is "F$:
      PRINT : HTAB 20:
      INPUT "Enter data file name: ";A$:
      IF A$ <  > "" THEN F$ = A$

3605  ABT = 0: DNE = 0:
      PRINT D$;"OPEN "F$:          ! Open data file
      PRINT D$;"READ "F$:
      INPUT ENUM:                  ! Input # of records

      PRINT D$;"PR#1":             ! Output is printer
      PRINT  CHR$ (9);"80N":       ! Set spacing
      PAGE = 1                     ! Initialize page counter

3615  PRINT  SPC( 55)PAGE,LF$,LF$,
             SPC( 8)"File: "F$,LF$,LF$,
             SPC( 8)"GAIN" SPC( 17)"SIGMA"
             SPC( 15)"J-OMEGA"LF$,LF$:     ! Output header
      PRINT D$:                    ! Set up for data
      PRINT D$;"READ "F$

      ! Do while not DONE or ABORTED
3620     FOR LINE = 9 TO 55:
            INPUT K,R,I:
            A = 15 -  LEN (STR$ (K)):
            B = 20 -  LEN (STR$ (R)):
            C = 20 -  LEN (STR$ (I)):
            PRINT  SPC( A)K SPC( B)R SPC( C)I:
            IF ECTR >  = ENUM THEN
               DNE = 1:
```

```
            LINE = 55
3625        IF ECTR < ENUM THEN ECTR = ECTR + 1
3630     NEXT LINE
3640     IF ( NOT ABT) AND ( NOT DNE) THEN ! If not done
            PAGE = PAGE + 1:               !   Incr page
            PRINT FF$:                     !   form feed
      GOTO 3615                       ! End do

3645  PRINT LF$,LF$,LF$:              ! Output line feeds
      PRINT D$;"CLOSE "F$:            ! Close File
      PRINT D$;"PR#3":                ! Reset output to CRT
      RETURN                          ! Return to sub menu



4000  REM  ---------- FILE MANAGEMENT ROUTINES -------------


4005  HOME :                          ! Clear screen
      VTAB 3: HTAB 30:
      PRINT "FILE MANAGEMENT MENU": PRINT :
      HTAB 26: PRINT "You can ...": PRINT :
      HTAB 29: PRINT "1. Rename data files.": PRINT :
      HTAB 29: PRINT "2. Delete data files.": PRINT
4025  HTAB 29: PRINT "3. Return to the main menu.": PRINT :
      PRINT
4030  HTAB 26: PRINT "Your choice is # ";:
      GET C$:                         ! Monitor Keyboard
      PRINT C$:                       ! Output reply to CRT
      C =  ASC (C$) - 48:             ! Convert ASCII to numeric
      IF C < 1 OR C > 3 THEN  GOTO 4030
4035  ON C GOSUB 4100,                ! Case 1: Rename files
                 4150:                !       2: Delete Files
      IF C <  > 3 THEN GOTO 4005      !       3: Exit
4040  RETURN                          ! Return to main program

      ! Case 1: Rename files

4100  VTAB 16: HTAB 26:
      INPUT "Original file name: ";OF$: HTAB 26:
      INPUT "New file name: ";NF$:
      PRINT D$;"RENAME "OF$","NF$:
      RETURN                    ! Return to file management

      ! Case 2: Delete files

4150  VTAB 16: HTAB 26:
      INPUT "File to be deleted: ";OF$: VTAB 16: HTAB 20:
      PRINT "Confirm order to delete "OF$" (Y/N): ";:
```

```
      GET C$:
      PRINT C$:
      IF C$ = "Y" THEN  PRINT D$;"DELETE "OF$
4160  IF C$ <  > "Y" THEN
      HTAB 20:
      PRINT "       DELETION ABORTED                          ":
      FOR I = 1 TO 1500:          ! Wait a bit
      NEXT I
4165  RETURN                      ! Return to file management



10000  REM ------------------- EXIT ROUTINE ----------------
10010  HOME : END
```

APPENDIX C
PLOTTER PROGRAM PSUEDOCODE

NOTE:

This program is called by the other programs. No data is passed to it but it does open and read a data file from the disk. After the data is plotted in Cartesian coordinates, the PLOTTER program calls the program specified by the user and passes it a flag which disables the title screen and its attendant wait.

```
/* PLOTTER PROGRAM */

     DEFINE X POINT PLACEMENT EQUATION (XMARK)
     DEFINE Y POINT PLACEMENT EQUATION (YMARK)
     DEFINE ARRAYS
     SET VARIABLE DEFAULT VALUES

     RETURN TO TEXT MODE AND CLEAR SCREEN

     PRINT "Currently selected data file: " FILE NAME
     PRINT "Data file to be plotted: " ON SCREEN

     INPUT FILE NAME
     IF (FILE NAME NOT BLANK) THEN
        FILE = FILE NAME
     ELSE
        FILE = DISK FILE
     ENDIF

     LOAD MARKER SYMBOL TABLE

     OPEN REQUESTED FILE
     INPUT NUMBER OF RECORDS IN FILE
```

```
/* SEARCH FOR MAX AND MIN VALUES */

     FOR I = 1 TO NUMBER OF RECORDS
        INPUT ELEMENT 1, ELEMENT 2, ELEMENT 3
        IF MAXIMUM X < ELEMENT 2 THEN MAX X = ELEMENT 2
        IF MINIMUM X > ELEMENT 2 THEN MIN X = ELEMENT 2
        IF MAXIMUM Y < ELEMENT 3 THEN MAX Y = ELEMENT 3
        IF MINIMUM Y > ELEMENT 3 THEN MIN Y = ELEMENT 3
     NEXT I
     CLOSE FILE


/* KEEP AXIS ON PLOT */

     IF MIN X ≥ 0 THEN MIN X = -0.0001
     IF MIN Y ≥ 0 THEN MIN Y = -0.0001
     IF MAX X ≤ 0 THEN MAX X =  0.0001
     IF MAX Y ≤ 0 THEN MAX Y =  0.0001


/* AUTORANGING */

     CALCULATE RANGE OF X
     CALCULATE RANGE OF Y
     CALCULATE X SCALING FACTOR
     CALCULATE Y SCALING FACTOR

     X AXIS = FUNCTION YMARK @ POINT 0
     Y AXIS = FUNCTION XMARK @ POINT 0

     SET SCREEN FOR HIGH RESOLUTION GRAPHICS
     PLOT X, Y AXIS

     X TICK SIZE = (RANGE OF X) / 10 SEGMENTS
     FOR I = 1 TO 10
        TICK POINT = FUNCTION XMARK @ (X TICK SIZE * I)
        PLOT POSITIVE TICK POINT USING SHAPE 1
        PLOT NEGATIVE TICK POINT USING SHAPE 1
     NEXT I

     Y TICK SIZE = (RANGE OF Y) / 10 SEGMENTS
     FOR I = 1 TO 10
        TICK POINT = FUNCTION YMARK @ (Y TICK SIZE * I)
        PLOT POSITIVE TICK POINT USING SHAPE 1
        PLOT NEGATIVE TICK POINT USING SHAPE 1
     NEXT I

     REOPEN REQUESTED FILE
     INPUT NUMBER OF RECORDS IN FILE
```

```
FOR I = 1 TO NUMBER OF RECORDS
   INPUT ELEMENT 1, ELEMENT 2, ELEMENT 3
   X POINT = FUNCTION XMARK @ ELEMENT 2
   Y POINT = FUNCTION YMARK @ ELEMENT 3
   PLOT X POINT, Y POINT
NEXT I
CLOSE FILE

PRINT "X Scale = " X TICK " per unit." ON SCREEN
PRINT "Y Scale = " Y TICK " per unit." ON SCREEN

DO WHILE (KEYBOARD NOT "R" AND NOT "F" AND NOT "P"
   PRINT "Press R for ROOT LOCUS, F for FREQUENCY,
          P to print display." TO BOTTOM OF SCREEN
   INPUT KEYBOARD

   /* CASE 1 - RETURN TO ROOT LOCUS PROGRAM */

     SET ROOT FLAG
     CALL ROOT LOCUS PROGRAM

   /* CASE 2 - RETURN TO FREQUENCY RESPONSE PROGRAM */


     SET FREQUENCY FLAG
     CALL FREQUENCY PROGRAM

   /* CASE 3 - PRINT DISPLAY */

     BLANK COMMAND LINE
     ENABLE PRINTER
     DISABLE SCREEN
     SET PRINTER FOR GRAPHICS
     OUTPUT SCREEN DISPLAY TO PRINTER
     DISABLE PRINTER
     ENABLE SCREEN

     RETURN TO TEXT MODE AND CLEAR SCREEN

ENDDO
```

APPENDIX D
PLOTTING PROGRAM LISTING

```
5  REM ----------------- PLOTTER PROGRAM ------------------

20 REM - DECLARE VARIABLES AND ARRAYS

40  DEF  FN XMARK(XP) =
          (279 * (XP +  ABS (XN))) / ( ABS(XRNG) + .0001):
    DEF  FN YMARK(YP) =
          (159 *  ABS (YX -  YP)) / (ABS (YRNG) + .0001):
    D$ =  CHR$ (4):
    DIM AS(3):
    XX = 0:
    XN = 0:
    YX = 0:
    YN = 0:
    F$ = "ROOT.DAT":
    REM  Define tic mark equations for the plotting program



100  REM ---------------- MAIN PLOTTER PROGRAM -------------

105  HOME : TEXT : VTAB 9: HTAB 23:
     PRINT "Currently selected data file: "F$:
     VTAB 12: HTAB 25:
     PRINT "Data file to be plotted: ":
     VTAB 12: POKE 36,49:
     INPUT "";A$:
     IF A$ <  > "" THEN F$ = A$

110  PRINT D$;"BLOAD ROOT.CS,A$6000":
     POKE 232, PEEK (43634):
     POKE 233, PEEK (43635):
     PRINT D$;"OPEN "F$:
     PRINT D$;"READ "F$:
     INPUT NREC:
     REM Input plotter shapes then # records in file
```

```
120  FOR I = 1 TO NREC:
     INPUT AS(1),AS(2),AS(3):
     IF XX < AS(2) THEN XX = AS(2): REM Max or min of plot
125  IF XN > AS(2) THEN XN = AS(2)
130  IF YX < AS(3) THEN YX = AS(3)
135  IF YN > AS(3) THEN YN = AS(3)
140  NEXT I
145  PRINT D$;"CLOSE "F$: REM                Close data file

150  IF XN >  = 0 THEN XN =  - 0.0001: REM Keep axis on
155  IF YN >  = 0 THEN YN =  - 0.0001: REM   on display
160  IF XX <  = 0 THEN XX = 0.0001
165  IF YX <  = 0 THEN YX = 0.0001

180 XRNG = (XX - XN):
    YRNG = (YX - YN):
    XSC = 280 / ( ABS (XRNG) + .0001):
    YSC = 160 / ( ABS (YRNG) + .0001):
    XAXIS =  FN YMARK(0):
    YAXIS =  FN XMARK(0):
    REM        Establish scaling factors and axis locations



190  REM ------------------- SET UP GRAPH -----------------

195  ROT= 1: SCALE= 1: HCOLOR= 3: HGR :
     HPLOT 0,XAXIS TO 279,XAXIS:
     HPLOT YAXIS,0 TO YAXIS,159:
     REM                              Set up graph and plot axis

200  TIC = XRNG / 10:
     FOR I = 0 TO 10:
     XTIC = TIC * I:
     PX =  FN XMARK(XTIC):
     IF PX <  = 279 THEN  DRAW 1 AT PX,XAXIS

210  PX =  FN XMARK(-XTIC):
     IF PX <  = 0 THEN  DRAW 1 AT PX,XAXIS:
     REM                            Calculate / plot x ticks
215  NEXT I: REM                    Get next x tick

230  TIC = YRNG / 10:
     FOR I = 1 TO 10:
     YTIC = TIC * I:
     PY =  FN YMARK(YTIC):
     IF PY <  = 0 THEN  DRAW 1 AT YAXIS,PY

232  PX =  FN YMARK(-YTIC):
```

```
     IF PY <  = 159 THEN  DRAW 1 AT YAXIS,PY:
     REM                          Calculate / plot y ticks
235  NEXT I: REM                  Get next y tick



300  REM ------------------ PLOT DATA ---------------------

310  PRINT D$;"OPEN "F$: PRINT D$;"READ "F$: INPUT NREC:
     REM       Open data file and read the number of records
               to be used

315  FOR I = 1 TO NREC:
     INPUT AS(1),AS(2),AS(3):
     XPT =  FN XMARK(AS(2)):
     YPT =  FN YMARK(AS(3)):
     DRAW 2 AT XPT,YPT: NEXT I:
     REM                      Input a record and plot on graph
320  PRINT D$;"CLOSE "F$: REM     Close data file



330  REM -- OUTPUT SCALE AND PREPARE TO RETURN TO MAIN --

350  XTIC = XRNG / 10: YTIC = YRNG / 10: VTAB 24:
     PRINT "X Scale = "XTIC" per unit.":
     PRINT "Y Scale = "YTIC" per unit.":
     REM                          Output scaling factors to CRT

360  VTAB 21:
     PRINT "Press R for ROOT LOCUS, F for FREQ, P to
            print display.";:
     GET K$:
     PRINT K$:
     IF (K$ <  > "R")
       AND (K$ <  > "F")
       AND (K$ <  > "P") THEN
         HOME : TEXT : GOTO 360: REM    Ask for next duty

400  IF K$ = "R" THEN
        ROOT = 1:
        PRINT D$;"BLOAD CHAIN,A520":
        CALL 520"ROOT LOCUS": REM       Return to ROOT LOCUS

410  IF K$ = "F" THEN
        FQ = 1:
        PRINT D$;"BLOAD CHAIN,A520":
```

```
        CALL 520"FREQ": REM              Return to FREQUENCY

/* The following are printer commands specific to the     */
/* APPLE IIe and the General Electric model TXP-8100      */
/* printer for setting up the graphics mode & then        */
/* returning to print mode.                               */

420   IF K$ = "P" THEN
         PRINT  CHR$ (17): VTAB 21:
         PRINT "                                          ":
         PRINT D$;"PR#1": PRINT  CHR$ (9)"Z":
         PRINT  CHR$ (27); CHR$ (24):
         PRINT  CHR$ (27);"A"; CHR$ (7):
         PRINT  CHR$ (27);"2"
425      PRINT D$;"PR#0":
         PRINT D$;"PR#1":
         PRINT  CHR$ (9)"14L":
         PRINT  CHR$ (9)"GML":
         PRINT D$;"PR#0": PRINT D$;"PR#1":
         PRINT  CHR$ (27); CHR$ (27):
         PRINT  "                    File: "F$
430      PRINT  CHR$ (17):
         PRINT D$;"PR#3":
         TEXT : HOME : GOTO 360:
      REM  Restore CRT to normal mode.
```

APPENDIX E

CALCULATIONS USED IN CONSTRUCTING FIGURE 7

Many books have been published describing graphic methods for producing root locus plots [4] so only the actual calculations will be shown here. The open loop transfer function of (3.4.1) is given by

$$1 + GH = 1 + \frac{K\ (42)}{S\ (S+5)\ (S+1)} \qquad \text{(E.1)}$$

Examination of (E.1) shows no zeros and three poles. The number of segments stretching to infinity is given by substracting the number of zeros from the number of poles, which in this case gives three. The fact that this is an odd number indicates that one of these segments will lie on the real axis and extend in a negative direction. Next, the angle between these segments is

$$\frac{360 \text{ degrees}}{(\#\text{ of Poles}) - (\#\text{ of zeros})} = \frac{360}{3} = 120 \text{ degrees} \qquad \text{(E.2)}$$

The asymptotes intercept point on the σ axis is given as

$$\frac{\Sigma \text{zeros} - \Sigma \text{poles}}{\text{Excess Of Poles Over Zeros}} = \frac{0 - 0 - 5 - 1}{3 - 0} = -2 \qquad \text{(E. 3)}$$

The breaK away point is given by setting equation (E. 1) to be equal to -1. This leads to the equation

$$42K = -S^3 - 6S^2 - 5S \qquad \text{(E. 4)}$$

TaKing the first derivitive of this with respect to S leads to

$$\frac{dK}{dS} = 0 = -3S^2 - 12S - 5 \qquad \text{(E. 5)}$$

The roots of equation (E. 5) are therefore

$$r = \frac{-(-12) + \sqrt{(12)^2 - 4(-3)(-5)}}{2(-3)} \qquad \text{(E. 6)}$$

or -2. 1165 and -0. 28345. Since the first root exceeds the asymptote intercept point it cannot be correct which leaves the second root to be the correct one.

The 4 page vita has been removed from the scanned document

The 4 page vita has been removed from the scanned document

The 4 page vita has been removed from the scanned document

The 4 page vita has been removed from the scanned document