DOCUMENT RESUME

ED 089 708 IR 000 420

AUTHOR Volz, Richard A.; And Others
TITLE COINGRAD; Control Oriented Interactive Graphical Analysis and Design.
INSTITUTION Michigan Univ., Ann Arbor. Dept. of Electrical and Computer Engineering.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE 74
NOTE 30p.; Preprint of article to appear in IEEE, Transactions on Education, August 1974

EDRS PRICE MF-$0.75 HC-$1.85 PLUS POSTAGE
DESCRIPTORS *Computer Assisted Instruction; *Computer Graphics; *Computer Programs; Design; Engineering; *Engineering Education; *Engineering Graphics; Higher Education; Interaction; Man Machine Systems; Program Descriptions; Simulation; Time Sharing; Undergraduate Study
IDENTIFIERS Automatic Control; CAD; CCINGRAD; *Computer Aided Design; Control Oriented Interactive Graphic Anal Design

ABSTRACT

The computer is currently a vital tool in engineering analysis and design. With the introduction of moderately priced graphics terminals, it will become even more important in the future as rapid graphic interaction between the engineer and the computer becomes more feasible in computer-aided design (CAD). To provide a vehicle for introducing undergraduate engineering students to the applications of CAD in the automatic control area, a series of three programs--named Control Oriented Interactive Graphical Analysis and Design (COINGRAD)--has been written. These incorporate the basic calculations in this area, including time response, frequency response, and root locus; in addition, a limited optimization facility is included with the time and frequency response programs. The major features of the programs include graphic interactive capability and ease of operation for the user. (Author)

COINGRAD - Control Oriented Interactive Graphical Analysis and Design*

by

Richard Volz+
Michael Dever++
Theodore Johnson**
Darryl Conliffe***

Abstract

The computer has come to be a vital tool in engineering analysis and design. With the introduction of moderate priced graphics terminals it will be even more so in the future. The rapid graphic interaction between the engineer and the computer will become more and more important in computer aided design. To provide a vehicle for introducting students to the power and utility of CAD in the automatic control area, three programs have been written incorporating the basic calculations in this area. These include time response, frequency response and root locus, with a limited optimization facilty included with the time and frequency response programs. The major considerations in the development were the graphic interactive nature of the programs and the ease of operation for the user.

COINGRAD - Control Oriented Interactive Graphical Analysis and Design*

by

Richard Volz+
Michael Dever++
Theodore Johnson**
Darryl Conliffe***

Introduction

During the past two decades engineering has come to rely heavily on the computer for all manner of tasks ranging from scientific computation to on line process control and data management. Emerging from this is a changing view of engineering design. More and more the engineer is coming to rely on the computer to provide computations and analyses which aid in the design process. We are entering an era of computer-aided design. One of the major keys to the continued growth of CAD will be the rapid graphic interaction between the engineer and the computer. The idea that an engineer can enter a set of data for the system he is studying and immediately see the results of the computation displayed, not in awkward tabular arrays, but in easy to understand graphical form will be of utmost importance.

There have been a number of computer-aided design programs developed for control systems during this time. One of the earlier efforts was the Control System Analysis Program (CSAP) developed by Fowler [1] at IBM, which provided root locus and time response calculation for both continuous and sampled data

---

* This work was supported by the Department of Electrical and Computer Engineering, University of Michigan and the National Science Foundation under grant No. GY-8313
+ Systems Engineering Laboratory, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI
++ Education Resources, Department of Dental School, University of Michigan, Ann Arbor, MI
** Power Systems Laboratory, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI
***Proctor Gamble Company, Cincinnati, Ohio

systems. This was modified at the University of Michigan (Volz [2]) to include page printer plotting, frequency response calculations and a simplified user oriented input. Kalman and Englan [3] developed the Automatic Synthesis Program (ASP) for the IBM 7090 computer. One of the major advantages to this program was that it provided the solution to the linear quadratic cost problem.

As an accompaniment to a basic text on automatic control theory, Melsa [4] has produced a collection of analysis and design programs for control systems. Ash and Ash [5] have proposed and programmed a highly efficient method for obtaining root loci. Even high order systems may be handled efficiently. One of the most promising recent efforts has been the INSIGHT program developed by Merritt and Sinclair [6]. This system is a graphical interactive system which provides the user with a convenient way for simulating the time response to a variety of systems. In addition, there are various simulation languages [7] such as IBM's Continuous System Modeling Program (CSMP) or the Continuous System Simulation Language (CSSL), which allows a user to program in relatively simple terms the system he wishes to study.

With the important role that computer tools will play in the future of engineering, it is extremely important that universities make their students aware of the potential of these tools and prepare them for their use. The objective of COINGRAD is to make this possible for the level of material covered in undergraduate control courses. To meet the coverage in the elementary control courses it was decided that the computer aids developed should first provide simulation of time responses for a wide variety of physical systems. Secondly, they should provide the standard analysis and design computations used in classical control theory, frequency response and root locus. Thirdly, a limited capability for system optimization should be incorporated to provide the students an introduction to the use of optimization in design.

What requirements should be imposed on the program developed? What should their operating environment be? How should they appear to the user? The answer to these questions are even more important to the student's view of computer aided design than the range of topics covered.

First and foremost, whatever else is done, the programs must be friendly to the user! That is the input/output and interaction between the user and the computer must be simple and self-explanatory as possible. Whenever possible data should be in forms as close to the natural expressions used as possible. Failure to observe this can result in even very useful programs lying around virtually unused. This consideration is particularly important in an educational environment where a student's objective is not to learn how to code data for CAD programs, but to be able to utilize them almost immediately to enhance his understanding of the material at hand. The typical student has enough to do trying to learn basic control principles without having to learn complicated data coding schemes.

Secondly, to achieve maximum effectiveness, the programs should have a graphic capability and operate in an interactive mode with the user. None of the programs discussed above fully meet these requirements. All are deficient in one or more of the desired characteristics.

This work was begun as a one term senior project course in the Electrical and Computer Engineering Department at the University of Michigan. Students who participated were required to have had as prerequisites both the elementary control systems course and adequate programming experience. At the conclusion of the term all of the basic functions of the systems were operable. The authors then continued the work on the systems under the auspices of NSF to add numerous convenience features and to standardize some of the syntax used.

The system of programs was written primarily to operate with graphics terminals such as those produced by Computek or Tektronix. The College of Engineering maintains a Graphics Laboratory open to the students containing three Computek and one Tektronix 4002 terminals, while the Computing Center maintains

two Tektronix 4010 terminals for student use. In addition, there are other terminals scattered around the campus in various laboratories. It is thus reasonable for advanced undergraduate students to use these in their studies.

General COINGRAD Features

COINGRAD is a set of three computer programs developed for the analysis and design of control systems. They will efficiently perform root locus, frequency response, time response and certain optimization computations for linear systems. In addition, certain nonlinearities may be included in the time response calculations. Though intended primarily for use in an interactive environment on a graphics terminal such as a Computek, they may be used on a standard teletype terminal or in batch mode.

The programs are all interactive in nature and basically operate by executing a sequence of commands issued to them by the user. The commands are all kept simple in nature such as RUN (a solution) or PLOT (the result) to make them easy for the user to remember (or look up in a short list). Once the user has indicated the basic function he wishes to perform, the program will interact with him and ask for whatever additional data may be needed, e.g. what is to be plotted. For example, to obtain a root locus plot, only three basic commands are needed:

```
G(s) = .....
RUN
PLOT
```

The program will ask for any remaining data that is needed.

An important feature of COINGRAD is that the manner in which the system is described is simple and kept as close to the natural mathematical language as possible. The programs assume a block-oriented system structure. The user will specify the characteristics of individual blocks of his system and an interconnection scheme between these, from which the program will perform the desired calculations. Each linear block is described by a transfer function of the form

$$G(s) = \frac{\sum_{i=1}^{m} \prod_{j=1}^{r_j} Q_{ij}(s)}{\sum_{i=1}^{m} \prod_{j=1}^{k_i} P_{ij}(s)} \qquad (1)$$

where $P_{ij}(s)$ and $Q_{ij}(s)$ are polynomials in the Laplace operator s. That is, the blocks are expressed as the ratio of sums of products of polynomials. Each polynomial must be presented in order of decreasing powers of s. To enter a transfer function one merely types it in the form one would normally write it, except that as raised superscripts are not allowed in the input character set, powers of s are indicated by integers following the operator s. Coefficients of s may be integers, real numbers, or single character variables to be manipulated later. The function is entered in free format, i.e., blanks are ignored. For example, to enter $G1(s) = \frac{K(s+a)}{(s+b)}$ and $G2(s) = \frac{(s+3)}{(s^2+6.25s+25)(s^2+100s+75}$, one would merely type

```
G1(s) = (k)(s+A)/(s+B)

G2(s) = (s+3)/(s2+6.25s+25)(s2+100s+75) .
```

The programs accept lines of this form as commands to enter the indicated transfer functions into system description arrays. As the line is processed the structure of the transfer function is identified and stored, coefficients are entered into the appropriate arrays, and variable references are set up. If a transfer function is too long to be entered on a single line, it may be continued on succeeding lines by ending unfinished lines with a colon (:) and continuing succeeding lines after the first column.

Following the definition of the linear blocks, one would normally specify the block interconnection. For the time

response program there is a simple command (ROUTE) to do this. The user will be asked to enter a sequence triples (I,J,K) indicating signal flow from block I to block J with gain K. For frequency response a single loop structure is assumed with transfer functions G(s) and P(s) assumed to be in series in the forward loop. One merely specifies whether the system is to be open loop or closed loop. The root locus program simply works with a single transfer function.

The use of variable names allows a part of the system to be modified without having to reenter the entire transfer function. To set (or change) the variable values, one again merely issues a simple command.

Both the time response and frequency response programs allow parameter optimization. When the optimization command is given, the program will ask the user for a desired transfer function. The coefficients given variable names during the system description will automatically be assumed to be the parameters with respect to which the optimization is to be performed, and the user will be asked for initial values for these. For time response, the function to be minimized is based on the integral square error between the desired and actual responses. For frequency response, it is based on a summation of the errors in the phase and logarithm plots of actual and desired transfer function at the frequency points for which the response is calculated. The optimization program used is the PRAXIS routine developed by Brent [8]. This is a conjugate direction procedure without derivatives based on Powell's method [9].

In addition, the COINGRAD programs have the following features.

(1) The programs determine upon evocation whether the initiating device is a graphic terminal, teletype or batch stream and set the input output formats accordingly.

(2) Useful reference lines such as constant damping ratio lines or the unit circle for root locus, or constant M contours for frequency response may be overlaid on the graphs produced.

(3) When appropriate, useful subsidiary information such as rise time, steady state error, gain margin and phase margin are calculated.

(4) The plot boundaries may be altered to "blow-up" interesting portions of the plots.

(5) Plots generated may be saved in MTS (Michigan Terminal System) files and redisplayed later without rerunning the program.

(6) CALCOMP plots may be simply produced (essentially just by asking for them).

(7) For graphic devices having a cursor which can be read, page position is checked before each output and writing off the bottom of the screen is inhibited. The user can erase the screen and continue when he is ready.

(8) In interactive terminal mode (graphic terminal or teletype) both program and attention interrupts are trapped and control is returned to the command structure, not to the computer operating system, thus allowing the user to take some corrective action and make another try.

(9) Both tabular and graphic data may be obtained.

In the following sections, characteristics specific to each of three tasks will be discussed, and an example of each given. In these examples, the commands issued by the user will be printed upper case italics, the responses by the computer in capital letters, and the responses by the user to requests from the program, in lower case italics. Explanatory comments which are <u>not</u> part of the program sequence will be designated by /* comment */ and superimposed on the examples.

Time Response

The time response portion of COINGRAD allows the user considerable freedom in specifying his system and obtaining solutions. Up to six linear blocks and one nonlinear block may be connected in any fashion desired. The linear blocks may each be up to fourth order, and are entered in the manner described above. For convenience, the blocks are numbered,

1 through 6 and block I is entered as GI(s) = ... The nonlinear block allows deadzone and saturation nonlinearities to be considered. The only other restrictions on the linear blocks are that in any block the order of the numerator be less than or equal the order of the denominator and the total order of the numerator around any closed loop be less than the total order of the denominator. The program uses a Adams-Moulton integrator developed by the Jet Propulsion Laboratories.

The output available from the program is also quite flexible. The user, can of course, obtain either tabular arrays or graphs of the solutions. The output of any of the blocks may be obtained, and if it is the output previously identified as the system output, several performance measures, such as rise time, steady state error or peak overshoot will be calculated where appropriate.

There are three ways the user can effectively use the variable names in analyzing a system. The most straightforward is to specify a set of parameter values, obtain a solution, plot it, specify a new set of values, obtain another solution, etc. The second method is similar to this, but allows several curves to be plotted on the same graph. To initiate this mode, the user need only specify the single command PRCH before specifying the first set of parameter values. The program will respond after plotting the first curve on the graph by asking the user for a new set of parameter values. For reference, the program will identify each curve plotted and indicate the parameter values used.

The third manner of using variable parameters is in the optimization option. This is entered by issuing a single command OPTM. Thereafter, the program will ask the user for the data needed. Essentially, it assumes that block 1 generates the input function, asks the user to specify a desired linear transfer function, takes those parameters given variable names as the unknown parameters, and tries to minimize a cost function based on an approximation to the integral square error between the desired response and the actual response. The user has the option of monitoring the progress of the optimization by comparing

the desired and actual curves every N trials, where the user specifies N. In this way, the program can be stopped when the user feels an adequate solution has been found. As the method requires a large number of solutions to a set of differential equations, however, the cost of using this feature may be high compared to the normal student computing budget. Consequently, only authorized users may use this feature.

To illustrate the use of this program, consider finding the unit step response to the system shown in Figure 1. The interaction with the time response program proceeds as follows.

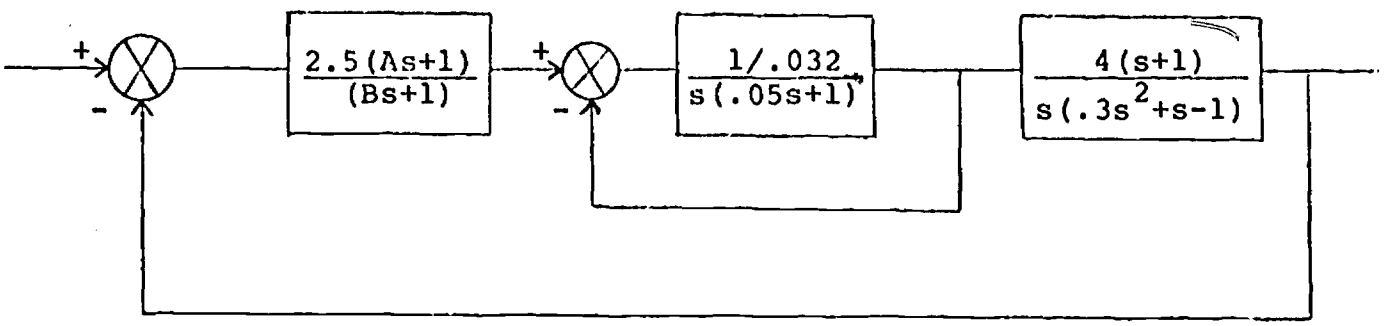


FIGURE 1

```
$SOURCE  E.E.:TIME                    /* This is all that is needed to start the program. */
#EXECUTION BEGINS

>******ENTER COMMAND******
>G1=(1)/(S)                           /* Here we enter our system description. */

>******ENTER COMMAND******             /* Block G1 with a nonzero initial condition is used to
>G2=(2.5)(AS+1)/(3S+1)                   generate the input function */

>******ENTER COMMAND******
>G3=(31.25)/(S)(.Ø5S+1)

>******ENTER COMMAND******
>G4=(4.Ø)(S+1)/(S)(.3S2+S-1)

>******ENTER COMMAND******
>ROUTE
>ENTER BLOCK CONNECTION SCHEME ******
>****** FROM BLOCK [I5] TO BLOCK [I5] WITH GAIN [F15.Ø]
>1,2,1.
>2,3;1.
>3,4,1.
>3,3,-1.
>4,2,-1.
>Ø,Ø,Ø.                                /* The string of zeros indicate that we are done with
>OUTPUT BLOCK = ?                         routing. */
>4

>******ENTER COMMAND******
>PRCH                                  /* This sets us up for multiple answers on a single
                                         plot. */
>******ENTER COMMAND******
>PRAM                                  /* Assign parameter value. */
>B = ?
>.015
>A = ?
>.3

>******ENTER COMMAND******
>DATA
>ENTER UP TO FOUR INITIAL CONDITIONS FOR INPUT [4F7.Ø]
```

```
>STARTING TIME [F4.Ø] ; ENDING TIME [F4.Ø] ; NUMBER OF OUTPUT
>POINTS [I4] ; SLOPE, DEADZONE, AND SATURATION [3F4.Ø]
>1.,Ø.,Ø.,Ø.,Ø.,5.,8Ø,Ø.,Ø.,Ø.                    /* The initial condition gives us a step response.*/

>******ENTER COMMAND******
>SOLN                                               /* This asks for a solution. */

>******ENTER COMMAND******
>OUTP4                                              /* Output from block 4 */
>**ENTER THRU FUNCTION BUTTONS<1-4>::OUTPUT TYPE**
>1=PLOT...2=DATA...3=BOTH...4=RETURN
                                                    /* Function button 1 was depressed. */
>***ENTER TYPE GRID DESIRED FROM FUNCTION BUTTONS 1,2***
>FUNCTION BUTTON 1=OPEN
>FUNCTION BUTTON 2=CROSSHATCHED                     /* 1 was selected */

>***ENTER TYPE PLOT DESIRED FROM FUNCTION BUTTONS 1,2
>FUNCTION BUTTON 1 = POINT BY POINT
>FUNCTION BUTTON 2 = INTERCONNECTED POINTS          /* 2 was selected */
```

/* After plotting curve 1 (see Fig. 2), the message below is printed. Function button 1 is depressed, the variables and their current values are printed. After each new curve, the message is repeated. Up to 5 curves may be drawn on a single plot. */

```
      >VAR-CHANGE..FB 1
>B= 0.015  ?.Ø2 ?.Ø1
>A= 0.300  ?.21 ?.2
```

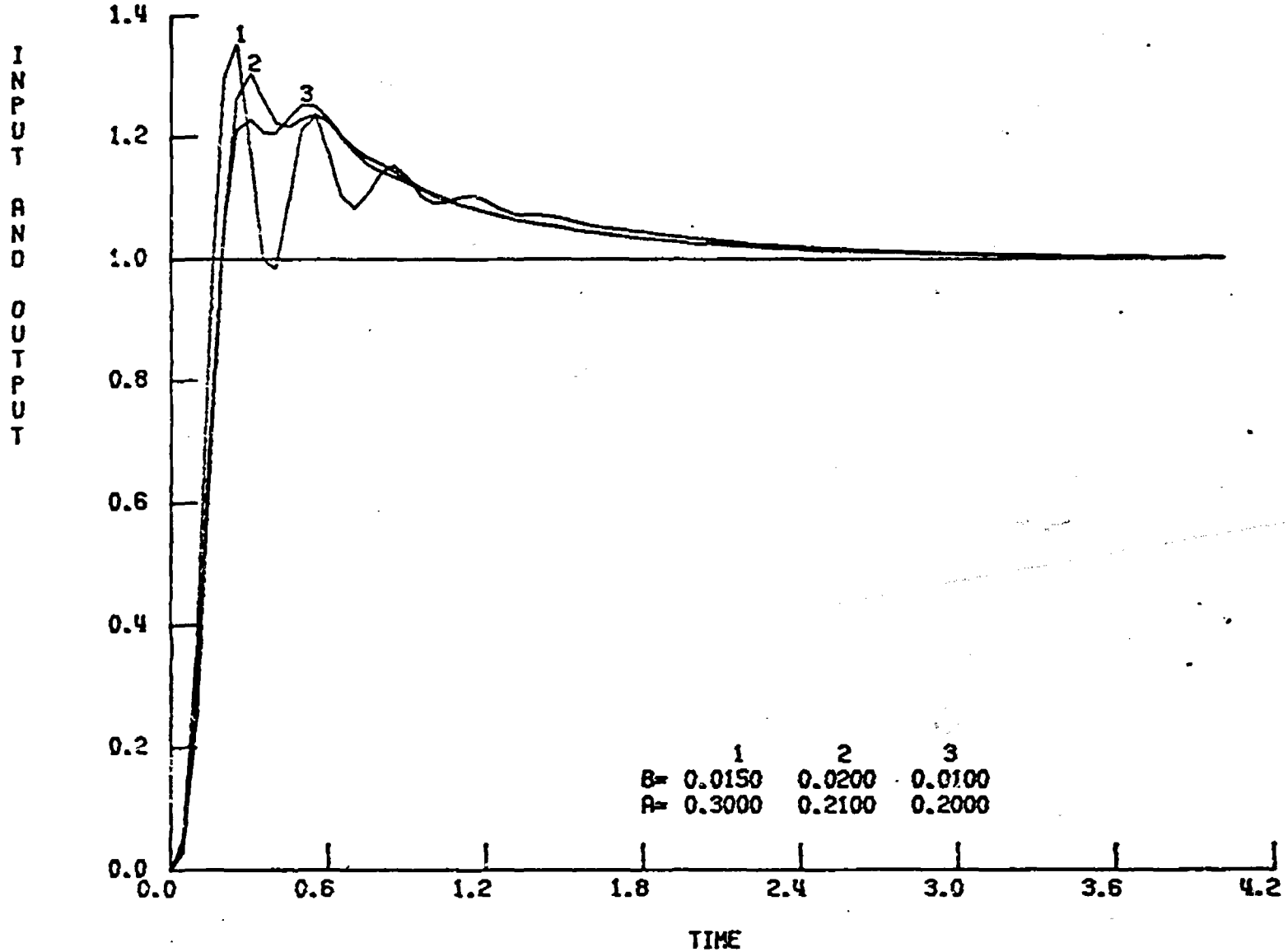
/* The user would, of course, see the answers displayed as they are produced on his terminal. The final composite curve is shown in Fig. 2 as produced by the Calcomp plotting feature. Note that each curve is labelled and the corresponding parameter values given. */

/* After finishing with a plot, this menu is printed. Function button 4 was selected. */

```
>FUNCTION     BUTTON   FUNCTION    BUTTON
>INPUT PLOT      1     SAVE           4
>DESIRED PLOT    2     DATA           5
>BEST FOUND      3     RELEASE        6
```

/* Whereupon this message is printed. */

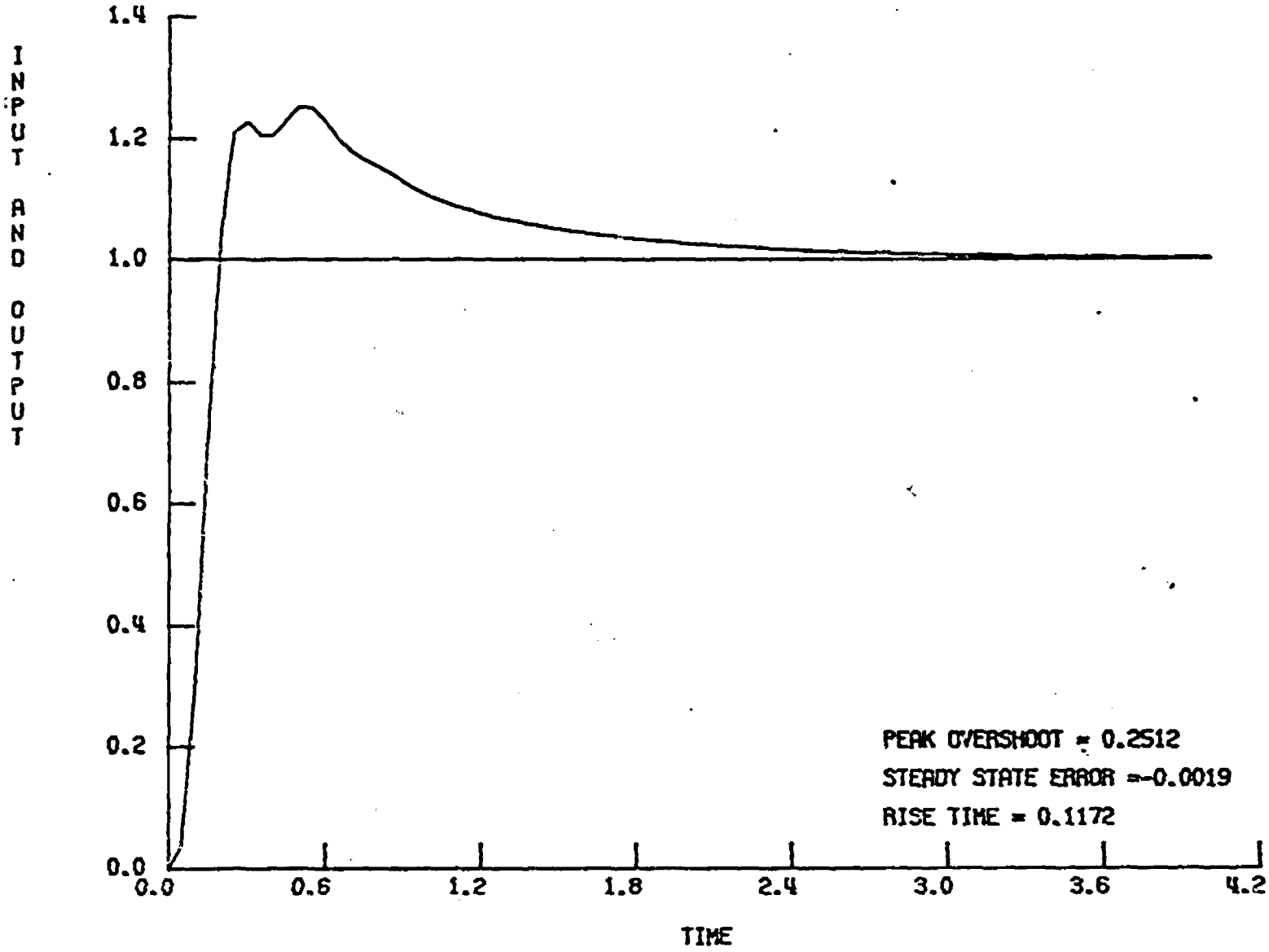
```
>**ENTER PLOT CODE NAME<<.LE.A4>>
                >ted3
```



CONTROL SYSTEM TIME RESPONSE

FIGURE 2

```
                                                    /* Whereupon this menu is printed. */
>FUNCTION     BUTTON
>VIEW         1
>EXPLAIN      2
>RELEASE      3
>CALCOMP PLOT 4

                                                    /* Function button 4 was selected. */
>ENTER SCALE FACTOR: > 1.                           /* This is the multiple of screen size to be used in
                                                       the plot.  This is all that is needed at this point
>******ENTER COMMAND******                           to botain the Calcomp plot. */
>PRAM
>B = ?
>.01                                                /* We will run one more solution with the best points
>A = ?                                                 found. */
>.2

>******ENTER COMMAND******
>SOLN

>******ENTER COMMAND******
>OUTP4
>**ENTER THRU FUNCTION BUTTONS<1-4>::OUTPUT TYPE**
>1=PLOT...2=DATA...3=BOTH...4=RETURN
                                                    /*The plot obtained is in Fig. 3.  Note the supplement
                                                      information provided. */
```



CONTROL SYSTEM TIME RESPONSE

FIGURE 3

Frequency Response

The section of COINGRAD called DUFS (Design Using Frequency Spectrum) may be used to display the frequency response of a system, or modify the response to match a desired frequency response by parameter optimization. The frequency response can be plotted in any or all of the following formats: (1) gain in db.vs.frequency; (2) phase vs. frequency; (3) imaginary part of magnitude vs. real part; (4) gain in db.vs. phase. Constant M contours may be placed on the Nyquist plot and the gain vs. phase plot (selectively generating lines that would appear on a Nichols Chart) as design aids. On a graphic terminal each of the graphs may be viewed individually, or each may be placed in a quardant of the screen so that any combination may be viewed simultaneously.

As stated earlier, the system structure is assumed to be a single loop feedback system for which both open and closed loop calculation are possible. The program recognizes the names P(s) and G(s) for transfer functions, and each may be order $n \leq 30$. The default value of the transfer functions is identity. Thus, they need not be specified if unused.

The number of basic commands is small. To obtain and plot a simple frequency one would need only issue the sequence P(s) = ..., PNTS, RUN, PLOT. The flexibility of the system arises through the options presented to the user as a result of the basic commands. This is perhaps best illustrated by taking a simple example such as a beginning student student might encounter.

Consider the system of Figure 4. Choose the parameters A and B in the compensator to yield a resonant peak, M_p, of 3db. To save space only the final check on the solution is shown. The conversation with the computer to do so might proceed as shown below. Recall that the only basic commands that the user need remember or look up are shown in capital italics.

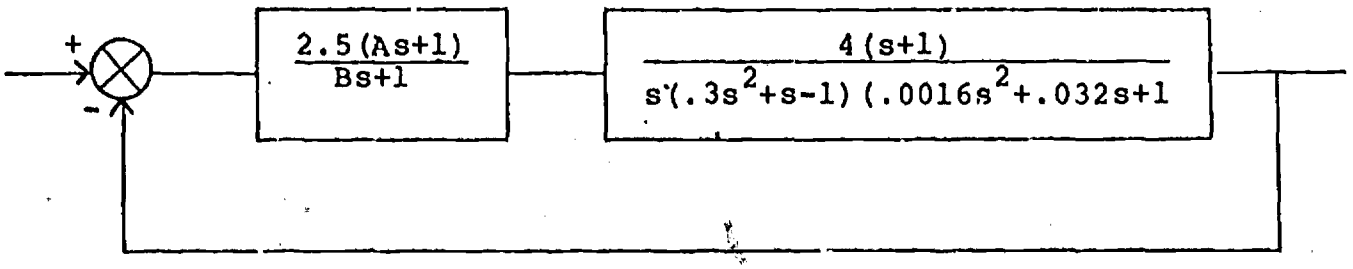


FIGURE 4

```
$SIGNON K2AT  PW=
$SOURCE E.E.:DUFS
PROJECT DUFS:
    DESIGN USING FREQUENCY SPECTRUM:  VERSION 04/06/73
USER K2AT SIGNED ON AT 10:59.41  11-08-73
11:00.42  NOV  8, 1973
$COST =    $.32                          /* This was the cost of signing on the computer and loading
?G(S)=(2.5)(AS+1)/(BS+1)                    the program. */
$G(S)=(2.5)(AS+1)/(BS+1)                 /* Here we enter the transfer functions. */
?P(S)=(4)(S+1)/(S)(.3S2+S-1)(.0016S2+.032S+1)
$P(S)=(4)(S+1)/(S)(.3S2+S-1)(.0016S2+.032S+1)   /* They are echoed by the program for
?PRAM                                              verification. */
$PRAM
A =   0.0
 A = .2
B =   0.0
 B = .02
?PNTS .5 20 2                            /* Here we set the lower frequency to  =.5 rad/sec, ask
$PNTS                                       for 20 points per decade (logarithmically spaced), and
?RUN                                        request data for 2 decades. */
$RUN                                     /* This actually requests the solution. */
?PLOT                                    /* Request to enter plotting mode. */
$PLOT

(1) DESIGN FUNCTION
(2) TABLE OF POINTS                      /* These are options which we may select. */
(3) *PRINT* OUTPUT
(4) BODE PLOT
(5) PHASE PLOT
(6) NYQUIST PLOT
(7) NICHOLS (GAIN VS. PHASE) PLOT
(8) M - CIRCLES
(9) COMBINATION OF PLOTS
(10) LIST OF SAVED PLOTS
?7 8                                     /* Here we ask for a gain vs. phase plot and indicate
$7 8                                        we want constant M contour for reference. */

NICHOLS PLOT
Y MAX/MIN:       25.5561          -27.3310
X MAX/MIN:     0.0                -281.903
 ENTER (0)  TO KEEP: (1) TO CHANGE: 1
** ENTER YMAX. YMIN. XMAX. XMIN:
25.,-25.,-90.,-225.                      /* This will restrict the portion of the results plotted
                                            to the area of primary interest. */
```

```
ENTER DB-CURVE NUMBER:  3.              /* This will cause a 3 db line to be printed for
DB-CURVE  3.00000                          reference. */
M-CIRCLE  1.41254
   POINTS PLOTTED WITH CHARACTER: 1
ANY MORE (Y OR N)? n.                   /* After this, the graph shown on Figure 5 is plotted on
                                           the screen. */
PHASE -DEG.-                            /* After the plot, the program prints: */
(F1)SAVE  (F2)CALCOMP  (F3)BOTH  (F4)CONTINUE
                                        /* Function button 2 was depress. */
                                        /* Whereup the computer responds. */
ENTER FIXED PT. SCALE FACTOR. MAX = 4.
(1. = SIZE OF SCREEN)
?1.

? STOP                                  /* After we give the scale factor, we are returned to
$STOP                                      the command mode. */
NO GRAPHS HAVE BEEN SAVED!!
 CALCOMP PLOTS IN -CKPLOT/ SAVED PLOTS IN -SAVEPLOT
$CREATE CKPLOT                          /* This is some final clean up needed to actually obtain
$COPY -CKPLOT TO CKPLOT                    a Calcomp plot. A permanent file must be created and
$PERMIT CKPLOT READ ALL                    the file -CKPLOT copies into it. The file must be
                                           permitted so that the plotting program can read it.
$RUN *CCQUEUE PAR=CKPLOT                   Finally, we run the system program which does the
                                           plotting. */
 YOUR PLOT FILES SHOULD BE PERMITTED (VIA $PERMIT)
 ARE THEY(*YES* OR *NO*)?
?yes

 PLOTTING WILL TAKE APPROX.
 TYPE *YES* FOR POST-PROCESS QUEUEING: OTHERWISE TYPE *NO*.
?yes

 *** ASSIGNED CALCOMP PLOT RECEIPT $ 508060 (11:11.12) ***

 YOU HAVE BEEN BILLED FOR THE ABOVE PLOT TIME.
$DISPLAY COST                           /* This is just to see how much money we spent.  While
$COST =  $1.89                             it looks high, it can actually be broken down as
                                           follows.
                                                signing on and loading program      .32
                                                entering data and obtaining solution .14
                                                plotting solution on Computek screen .22
                                                generating Calcomp plot             1.21  */
```

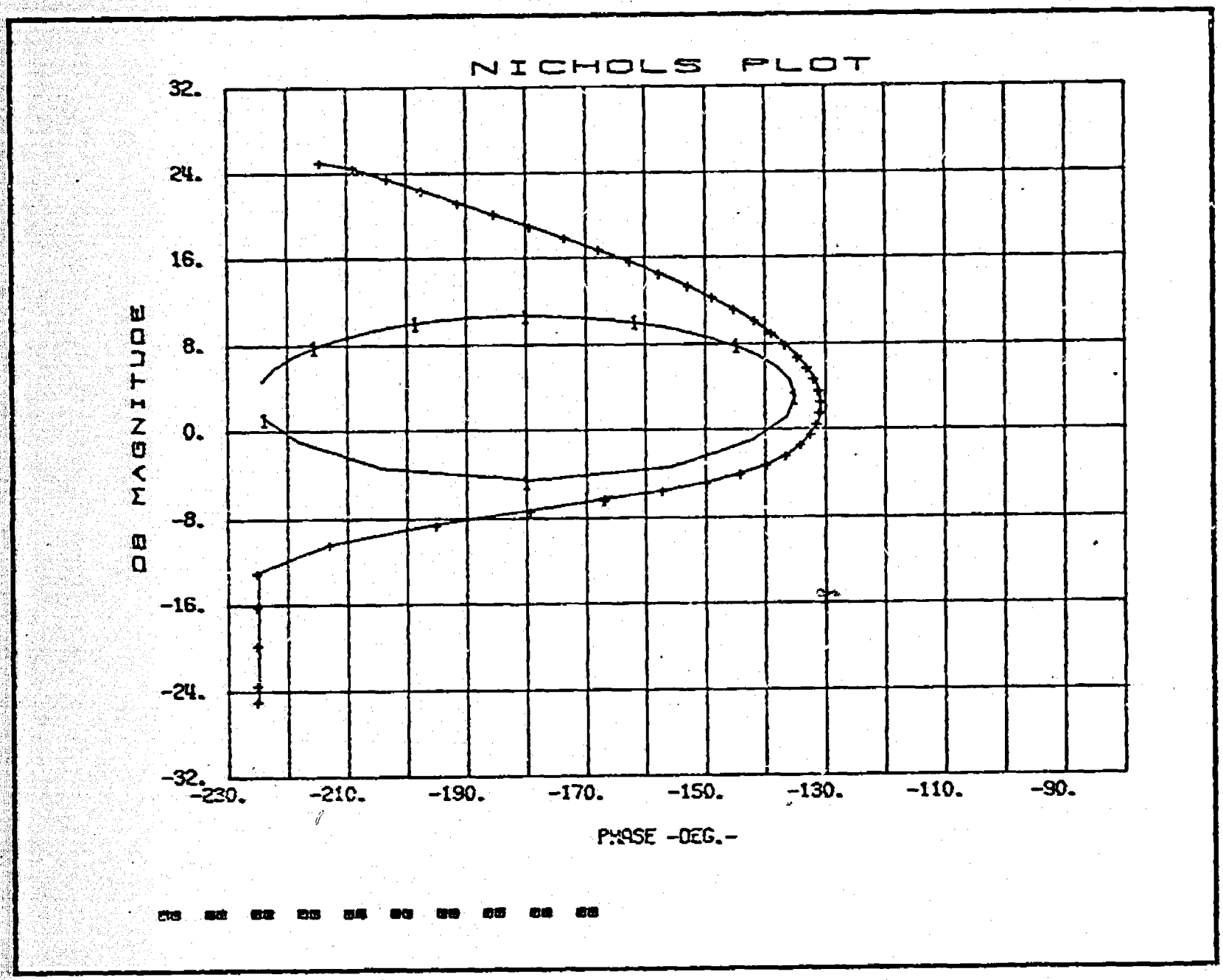

NICHOLS PLOT
32.
24.
16.
8.
0.
-8.
-16.
-24.
-32.
DB MAGNITUDE
-230.
-210.
-190.
-170.
-150.
-130.
-110.
-90.
PHASE -DEG.-

FIGURE 5

Root Locus

The root locus program provides the user with a simple yet efficient means of calculating and displaying root loci. It finds the roots of

$$1 + KG(s) = 0$$

lying in a given rectangle and the values of K corresponding to these. The function G(s) can be entered in either of two ways, through the command G(s) = ..., as with the time or frequency response programs, or, if the user wishes, by entering a list of the poles and zeros. The transfer function G(s) may be up to 29th order, and the roots may be calculated for either positive or negative gain. A pure time delay, e^{-Ts}, is also allowed in the transfer function.

Contrary to most root locus programs, this one accurately find root loci at equally spaced points on the locus. A rectangle is specified within which the roots are desired. The program scans along the boundary finding all points where loci enter or leave the rectangle, and factors the function G(s). The Newton Raphson procedure proposed by Ash and Ash [5] is then used to follow the various loci, either from poles within the rectangle, or points where the loci enter the boundary. Absolute accuracy to within a specified tolerance is maintained.

The boundarys of the rectangle, spacing between locus points, accuracy, and time delay may all be varied by the user. For his converience, however, all parameters have useful default values and need be changed only if required by a particular transfer function. Upon occasion a root locus will have a sharp turn relative to the spacing between points. The program will detect this and automatically reduce the spacing while the locus is moving around the turn.

For reference, the program will superimpose on the output graphs both constant damping ratio lines and the unit circle, if desired. Use of root loci often involves locating a point on the locus and then determining the gain corresponding to this. An option available to the user is to have the program aid in this. The user may type in a set of coordinates and

have the program find the nearest point on the locus (among those calculated) and print the resultant coordinates and gain. Of course, the graph saving and Calcomp options are available here as with the other programs.

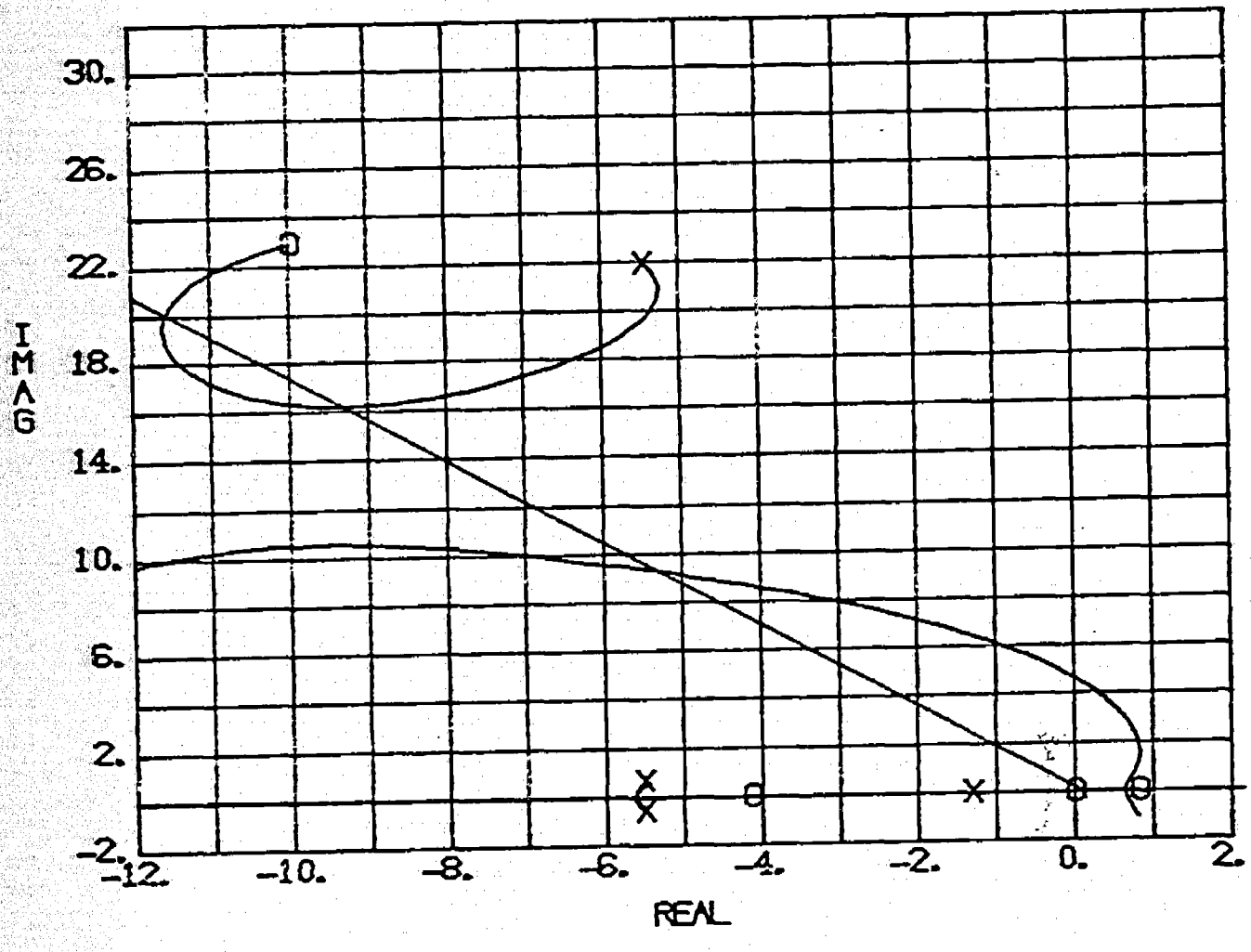
To illustrate, suppose it is desired to run a family of root loci for the function

$$G(s) = \frac{(s^2)(.0016s^2+.032s+1)(.3s^2+s-1)}{(s)(.0016s^2+.032s+1)(.3s^2+s-1)+(10)(as^2+1)(s+1)}$$

for different values of the parameter a. It is also desired to look at conditions near the .707 and .5 constant damping ratio lines. The interactive computer sequence is as follows.

```
$SOURCE E.E.:ROOT                    /* Tell the computer to look in the file E.E.:ROOT for
                                        it's commands. */
#EXECUTION BEGINS

*** WARNING DO NOT ENTER DATA WITHOUT BEING PROMPTED!!
ENTER COMMAND:  $
#$DIS COST
#COST =    $.26
ENTER COMMAND:  G(S)=(S2)(.0016S2+.032S+1)(.3S2+S-1)/(S)(.0016S2:   /* A colon placed anywhere
?  +.032S+1)(.3S2+S-1)+(10)(AS+1)(S+1)                              acts as a line continuation. */
ENTER COMMAND:  ALTER
? a=.23033
? @                                  /* End of assignment of parameter values. */
ENTER COMMAND:  BOUNDS=1,-1./-12 30  /* Actual problem suggests boundary changes--Note that the
ENTER COMMAND:  RUN                     opposite corners are entered in free format. */
HIT ANY FUNCTION BUTTON OR NUMERIC TO CLEAR SCREEN

     STEP SIZE =      0.2500000000
     TOLERANCE =      0.2500000000D-03
     TIME DELAY =      0.0
     XMAX =         1.000000000
     YMAX =         30.00000000
     XMIN =        -12.00000000
     YMIN =        -1.000000000
     DEVICE = COMPUTEK
     MODE = SLOW

THE ZEROS ARE:
   REAL       IMAG          REAL       IMAG          REAL       IMAG
    0.0       0.0            0.0       0.0          0.8054     0.0
   -4.1387    0.0          -10.0000  -22.9129      -10.0000   22.9129
THE POLES ARE:
   REAL       IMAG          REAL       IMAG          REAL       IMAG
  -1.3321     0.0          -5.5201   -0.6041       -5.5201    0.6041
  -5.4805   -21.8436         5.4805   21.8436

EVERYTHING ACCEPTABLE? (Y OR N)
? y                                  /* If "NO" then the user is returned to the command mode. */

***CALCULATION BEGINS
ENTER COMMAND:  PLOT V+ρR
HIT RETURN TO CONTINUE
ENTER DAMPING RATIO:    ? .5         /* Upon request the user may add damping ratio lines. */
```

```
ENTER DAMPING RATIO:    ? @          /* The resulting plot in shown in Fig. 6.  After plotting
                                        this, the following menu is presented. */

1=GAIN          4=PLOT               /* A 1 was depressed. */
2=SAVE          5=REPLOT
3=CALCOMP       6=RETURN

                                     /* The user types approximate coordinates and the program
                                        finds the nearest point on the locus. */

? -5,10
X=   -5.4037416273 Y=    9.2842310673 GAIN=  .602265390D-01
? 0,0
X=    0.            Y=    0.           GAIN=  ZERO        /* Indicates a zero at the origin */
? @

1=GAIN          4=PLOT               /* A 3 was depressed. */
2=SAVE          5=REPLOT
3=CALCOMP       6=RETURN

ENTER FIXED PT. SCALE FACTOR. MAX = 4
(1. = SIZE OF SCREEN)                /* The plotting data is placed in a temporary file
? 1.                                    -CKPLOT. */

ENTER COMMAND:  stop
```



RT LOCUS

FIGURE 6

Operating Costs and Program Availability

COINGRAD has been in operation for aobut a year on the dual processor 360/67 at the University of Michigan. The costs of running vary significantly with system load and how the user operates the programs. The frequency response example given previously included the cost of running the job. The costs shown are typical of running during a heavily loaded period. The programs cost $.25 to $.35 to load, plots on the graphic terminal screen run about $.15 to $.25, and the calculation of the frequency response vary with the size of the system, but typically runs less than $.30. The costs for root locus and time response calculation again depend on the size system being considered (root locus) and/or eignvalue split (time response), but typically fall in the $.20 - $.75 range. The Michigan Computing Center charges $2.20 to $2.69/hr. (depending on time of day) simply to have a terminal connected to the computer, exclusive of any computation. For most student sessions, this is the dominant cost.

The source programs can be obtained from the first author. With the single exception of a short routine to trap interrupts all of the programs written by the authors for COINGRAD are in IBM level G Fortran IV. However, extensive use was made of existing graphics libraries for the graphic terminals, page printer plotting and Calcomp plot generation which were originally written in 360 assembly language. The page printer plotting routines have since been translated by their author to Fortran, and the graphic routines for a Tektronix 4010 are currently in the process of being translated into Fortran. The major portion of the system thus either are, or will shortly be, available in Fortran. In addition, however, COINGRAD makes use of few system routines in MTS (Michigan

Terminal System) for such tasks as determining what type of device is using the program. If COINGRAD were to be transferred to another system, these would have to be emulated or replaced by dummy routines. Efforts to simplify this are currently under way.

Conclusion

The programs described here provide a means of introducing the power and utility of computer aided analysis and design tools for automatic control to undergraduate students. They perform the basic mundane calculations required for classical control system analysis, making it possible for students to concentrate on the principles involved, not on repeated calculation of system behavior. Too often students without computer aids become more embroided in eliminating arithmetic errors than in understanding the subject. With computer aids it is possible to study more realistic and meaningful problems.

Most important in this implementation of the basic control calculations are the graphical interaction which takes place between the user and computer, and the simplicity of use. The student can enter his problem description simply and see the pertinent calculations displayed almost immediately in easy to interpret graphical form. Errors can be immediately corrected and a number of trials investigated rapidly, making it possible to deal with problems efficiently. The user oriented data formats have proven to be very easy to learn, and students have been able to begin using COINGRAD with minimal instruction. This ease of use is important, for without it much of the advantage of computer aids for instruction would be lost by time needed to learn how to use them.

Finally, it should be noted that not only were a useful set of programs produced, but the mechanics of doing much of the development in a project course provided an excellent learning opportunity for the students involved.

Acknowledgements

The authors are indebted to the National Science Foundation (Grant GY 8313) and the Department of Electrical and Computer Engineering for financial support of the project.

The second half of this work, COINGRAD, is the outgrowth of an ECE 473 design project taught under the direction of Professor R. A. Volz during the Winter Term of 1972. The members of the class and the portion of the system they developed are: (1) root locus; Carl Romanik, Daniel White and Mike Dever, team leader; (2) time response; Ted Johnson, Randy Penning and Bob Weiss, team leader; (3) frequency response; Roger Tanner, Marvin Spears and Darryl Conliffe, team leader.

References

1. Fowler, M., "Control Systems Analysis Program II (CSAP II)," IBM System Research and Development Center Technical Report.

2. Volz, Richard A., "Control Systems Analysis Program - Simplified Input," University of Michigan, Electrical and Computer Engineering Department Report. 1967.

3. Kalman, R. E. and T. Engler, "ASP - The Automatic Synthesis Program (Program C)," NASA Contractor Report CR-475, June 1966.

4. Melsa, James L., Computer Programs for Computations/Assistance McGraw-Hill, New York. 1970.

5. Ash, R. H. and G. R. Ash, "Numerical Computations of Root Loci Using the Newton-Raphson Technique," IEEE Transactions on Automatic Control, vol. AC-13, no. 5, October 1968, pp. 576-581.

6. Merritt, M. J. and R. Sinclair, "Insight: An Interactive Graphic Instructional Aid for Systems Analysis," Tech. Report No. 71-21, University of Southern California, May 1971.

7. Stephenson, Robert E., Computer Simulation for Engineers, Harcourt Brace Jovanovich, Inc.

8. Brent, Richard P., "Algorithms for Finding Zeros and Extreme of Functions Without Calculating Derivatives," Ph.D. Dissertation Computer Science Dept., Stanford University, Feburary 1971.

9. Powell, M. J. D. "An Efficient Method for Finding the Minimum of Several Variables Without Calculating Derivatives," The Computer Journal, Vol. 7, pp. 155-163, July 1964.