A NEW COMPUTER TECHNIQUE FOR ROOT LOCUS ANALYSIS

By

BASKARAN SANKARAN

Bachelor of Technology in Mechanical Engineering

Indian Institute of Technology

Madras, India

1977

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1979

Thesis
1979
S227n
Cop. 2

OKLAHOMA STATE
UNIVERSITY
LIBRARY

A NEW COMPUTER TECHNIQUE FOR ROOT LOCUS ANALYSIS

Thesis Approved:

Lynn R. Ebbesen
Thesis Adviser

James H. Taylor

K. N. Reid

Norman N. Durham
Dean of the Graduate College

1031952

ACKNOWLEDGMENTS

During my graduate studies I received a great deal of help and encouragement from a number of individuals. I am very grateful to them all for having made my stay at Oklahoma State University a memorable one.

My thesis adviser, Dr. L. R. Ebbesen, was a constant source of help and inspiration throughout the course of this work and throughout my graduate studies. I sincerely appreciate his patience and understanding through the many hours of discussion that I had with him despite his busy schedule. I am extremely grateful to him.

I thank the other members of my committee, Dr. K. N. Reid and Dr. J. H. Taylor, for their help, advice, and criticism. I also thank my friends, Mr. Vijay K. Maddali and Mr. John E. Perrault, for their help and suggestions. I thank Ms. Charlene Fries for typing the final draft of my thesis.

I am grateful to the School of Mechanical and Aerospace Engineering for their financial support throughout my graduate studies, and the Kirtland Air Force Base for their support during the course of this work.

My parents, my brothers and sisters deserve a special thanks for their advice and encouragement throughout my academic career. I dedicate this thesis to my late father, Mr. Ramayya Sankaran, whose advice and encouragement have inspired me throughout my life so far and will do so in the future too.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Purpose

The root locus is a plot of the roots of the characteristic equation as a function of system parameters. It clearly indicates the effect of parameter adjustment on the roots of the characteristic equation. The principle is based on the fact that the poles of the overall transfer function R(s)/U(s) are related to the zeros and poles of the open-loop transfer function and to the gain. The root-locus method not only indicates whether a system is stable or unstable, but also indicates the degree of stability for a stable system. In designing linear control systems, the root-locus method proves quite useful since it indicates the manner in which the open-loop poles and zeros should be modified so that the response meets system performance specifications. Additional equipment may be introduced into a system to reshape the root locus in order to improve system performance.

1.2 Background

The objective of this study is to develop a computer program for obtaining the roots of the characteristic equation when the system equations are written in the form

$$[P]\ [R] = [Q]\ [U]$$

where

[P] = a lxl transformation matrix;

[R] = an output vector of dimension l;

[Q] = an lxm control gain matrix; and

[U] = a control input vector of dimension m.

A general purpose frequency response program has been developed by the School of Mechanical and Aerospace Engineering, Oklahoma State University (9) by expressing the input-output relations in matrix form. Several models have been developed using this representation to perform frequency response analysis. Hence it was desired to use the same representation to perform root locus analysis.

Evans (1) devised a graphical method for obtaining the root locus plot as he found that the process of calculating the roots of the characteristic equation for various values of the system parameter becomes tedious for systems of higher than second order. As the graphical approach was tedious for higher order systems, Bendrikov and Teodorchik (2) developed an analytic theory of constructing root loci. A semi-analytic approach has been developed by Krishnan (3). Bendrikov and Teodorchik demonstrated that the geometric shape of the root locus of an algebraic equation in the complex plane does not change under linear coordinate transformations. However, their method was very complicated, and a direct digital computer approach was lacking. The introduction of matrix notations for Routh's array by Parks (4) and the formulation of Routh's algorithm for determining the oscillatory modes in feedback systems by Shen-Chen (5) helped Chen and Hsu (6) to develop a digital-computer-oriented root locus method by using Routh's algorithm.

Schulz and Melsa (7) have developed a computer program for plotting the root locus when the characteristic equation is in polynomial form. Robert J. Thomas (8) has indicated that the Jenkins-Traub algorithm is an efficient root-finder when the characteristic equation is in polynomial form. He has also indicated that the Jenkins-Traub algorithm is cheaper and more efficient than the Bairstow and Muller algorithms for higher order polynomials.

The study consists of five main parts. The first part describes the various ways in which a system can be represented. In the second part a description of the solution approach is presented. An explanation of the program in the third part is followed in the fourth part by three examples which illustrate the purpose and use of this program. The last part presents the conclusions, limitations of this program, and recommendations for further study. A listing of the computer program, a description of the sparse matrix method for evaluating the determinant, and Muller's algorithm for solving algebraic equations are presented in the appendices.

CHAPTER II

SYSTEM REPRESENTATION

2.1 Introduction

In this chapter we will consider the different ways in which a system can be represented. We will illustrate the different representations by considering a third order system. We will select a particular representation for the purposes of this work and show that root-locus analysis can be performed when the system is represented in this manner.

2.2 System Representation

Any system can be represented in the following ways:

1. Input-output relation representation (block diagram approach).
2. State variable representation.
3. Input-output relation representation (matrix approach).

The third representation will be discussed in detail in the sections to follow.

One of the principal tools of the input-output representation is the transfer function. This method provides less detail and therefore is less complete than the state variable method. However, the input-output representation is easier to deal with. Since the transfer function representation specifies only the input-output behavior, one can always make an arbitrary selection of state variables for a system specified only by a transfer function. In general, an infinite number of state variable representations exist for a given transfer function. If a state variable representation of a system is known, then the transfer function of the system is completely and uniquely determined. The third type of representation may be used to describe the system using a lesser number of equations. It is helpful to analyze any particular part of the system, especially when performing frequency response analysis.

Some authors of books on control systems (10) (11) define the characteristic equation as the denominator of the transfer function equated to zero, while others (12) define the characteristic equation as the determinant (sI-A) equated to zero, where (sI-A) is obtained from the state variable representation discussed in section 2.3. In this work we will refer to the denominator of the transfer function when equated to zero as the characteristic equation and the determinant (sI-A) equated to zero as a particular form of the characteristic equation. The eigenvlaues of the [A] matrix in the state variable form are identical to the poles of the transfer function.

2.3 Illustrations Using a Third-Order System

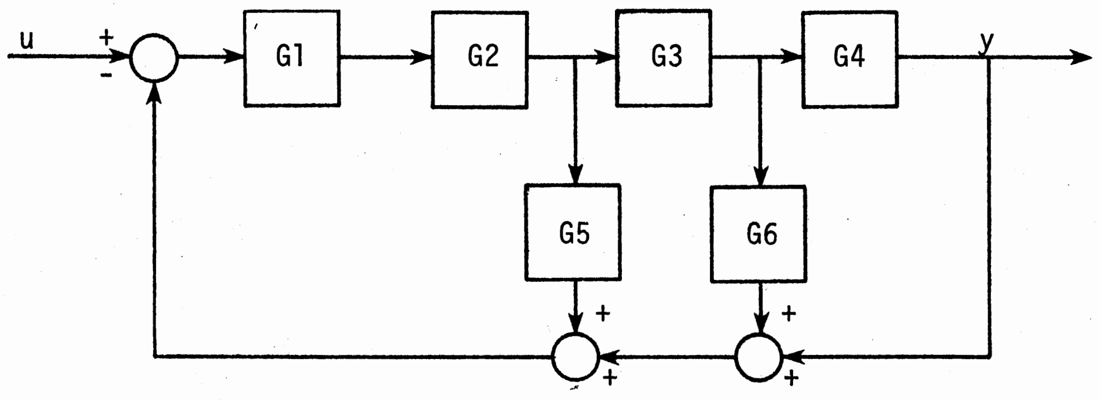
Let us consider a third-order system described by the block diagram shown in Figure 1. Using the block diagram identities, the above block diagram can be reduced as shown in Figure 2. The transfer function for the above system is given by

$$y(s)/u(s) = G(s)/(1 + G(s)) \tag{2.1}$$

where

$y(s)$ = Laplace transform of the output; and

$u(s)$ = Laplace transform of the input.

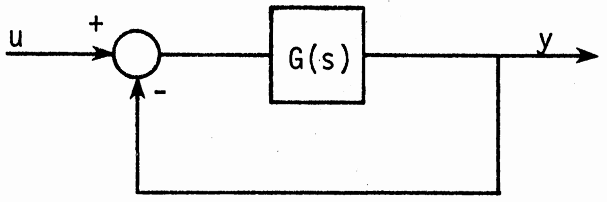


$G1 = K$ $\qquad$ $G4 = \frac{1}{s}$

$G2 = \frac{5}{s+8}$ $\qquad$ $G5 = \frac{3}{20}$

$G3 = \frac{2}{s+2}$ $\qquad$ $G6 = \frac{17}{60}$

Figure 1. Block Diagram of a Third-Order System



$$G(s) = \frac{G1G2G3G4}{1+G1G2G5+G1G2G3G6}$$

Figure 2. Simplified Block Diagram of Figure 1

The denominator of the above transfer function, when equated to zero, is the characteristic equation of the system. Hence the characteristic equation of the above system is

$$1 + G1G2G5 + G1G2G3G6 + G1G2G3G4 = 0 \qquad (2.2)$$

or

$$1 + K(5)(3)/(s + 8)(20) + k(5)(2)(17)/(s + 8)(s + 2)(60)$$
$$+ K(5)(2)(1)/(s + 8)(s + 2)(s) = 0$$

The characteristic equation for the same system can also be obtained by the state-variable method. The block diagram of Figure 1 is redrawn showing all the states in Figure 3.

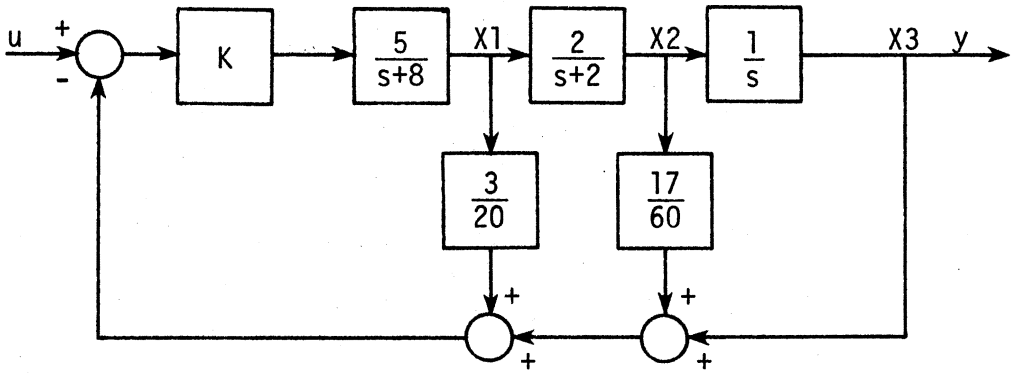


Figure 3. Figure 1 Redrawn Showing All the States

In Figure 3, X1, X2, and X3 are the state variables, u is the scalar input, and y is the scalar output. In the state-variable representation, the state equations can be written as

$$\dot{X}(t) = AX(t) + bu(t) \qquad (2.3)$$

$$y(t) = c^T X(t) + du(t) \tag{2.4}$$

where

$X(t)$ = n dimensional vector known as the state vector;

A = a constant matrix of dimension nxn;

u = a control input vector of dimension m;

b = a control gain matrix of dimension nxm;

c^T = an output gain vector of dimension n;

d = a vector of dimension m; and

y = a scalar output.

Transforming both sides of Equations (2.3) and (2.4) into the Laplace domain,

$$sX(s) = AX(s) + bU(s) \tag{2.5}$$

$$y(s) = c^T X(s) + du(s) \tag{2.6}$$

Equation (2.5) can be written as

$$X(s) = (sI-A)^{-1} bu(s) \tag{2.7}$$

Substituting the expression for X(s) into the relation for y(s), we get

$$y(s) = c^T \{(sI-A)^{-1} bu(s)\} + du(s)$$

or

$$y(s) = [c^T \{(sI-A)^{-1} b\} + d]u(s) \tag{2.8}$$

The characteristic equation is given by the determinant $(sI-A) = 0$.

For the system considered above,

$$[A] = \begin{bmatrix} 8 - \frac{3k}{4} & \frac{-17}{12}k & -5k \\ 2 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$[sI-A] = \begin{bmatrix} s + \frac{3k}{4} + 8 & \frac{17}{12}k & 5k \\ -2 & s+2 & 0 \\ 0 & -1 & s \end{bmatrix}$$

The characteristic equation of the above system is given by

$$(s+8+3K/4)(s+2)(s) + (17K/12)(2s) + (5K)(2) = 0 \qquad (2.9)$$

The above equation is just a simplified form of Equation (2.2).

We will now write the system equations using the input-output relations (matrix approach). The block diagram shown in Figure 1 can be redrawn as in Figure 4 showing all of the inputs and outputs that we are going to consider.

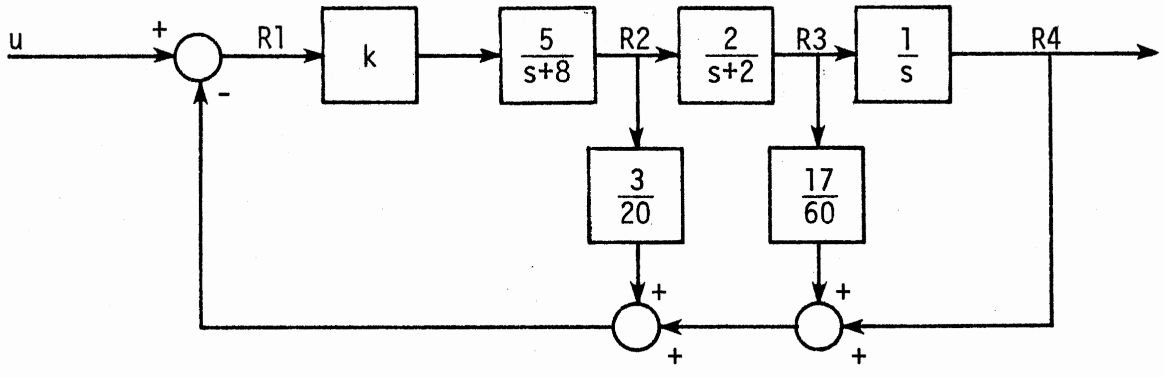


Figure 4. Block Diagram With Four Outputs and One Input

In Figure 4, u is the scalar input, R1 is the error signal, and R4 is the output. In the matrix approach, the system equations can be written in the form

$$[P][R] = [Q][U] \tag{2.10}$$

where

[P] = a 4x4 transformation matrix which is a function of s;

[R] = an output vector of dimension 4;

[Q] = a 4x1 control gain matrix which is a function of s; and

[U] = a control input vector of dimension 1.

For the above system,

$$P = \begin{bmatrix} 1 & G5 & G6 & 1 \\ -G1G2 & 1 & 0 & 0 \\ 0 & -G3 & 1 & 0 \\ 0 & 0 & -G4 & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Inspection of Figures 3 and 4 indicates that the following equations can be written:

$$R1 = U - X1G5 - X2G6 - X3$$

$$R2 = X1$$

$$R3 = X2$$

$$R4 = X3$$

The above equations can be written in matrix form as shown below.

$$\begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \end{bmatrix} = \begin{bmatrix} -G5 & -G6 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \\ X3 \\ \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u$$

If we consider l outputs, n states, and m inputs, the matrix equation can be generalized and written in the form

$$[R] = [E][X] + [F][U] \tag{2.11}$$

where

[E] = a lxn transformation matrix on X; and

[F] = a lxm transformation matrix on U.

Equation (2.10) can be rewritten as

$$[R] = [P]^{-1}[Q][U] \tag{2.12}$$

Substituting the expression for [X] in Equation (2.7) into the expression for [R] in Equation (2.11), we get

$$[R] = \{[E][sI-A]^{-1}b + [F]\}[U] \tag{2.13}$$

Comparing Equations (2.12) and (2.13), we know that determinant [P] = 0 is the characteristic equation.

We will now write the system equations using the above approach and show that irrespective of the number of equations we consider, we obtain the same form of characteristic equation for the system. We will now demonstrate three alternate forms using Equation (2.7), where u is as for the state variable model, [R] = [E][X] + [F][U], and y is as for the state variable model. We will show that irrespective of the form in which Equation (2.7) is written, determinant [P] is the same for all of the cases and when equated to zero, it represents the characteristic equation.

For the block diagram shown in Figure 4, the system equations can be written in matrix form as

$$\begin{bmatrix} 1 & G5 & G6 & 1 \\ -G1G2 & 1 & 0 & 0 \\ 0 & -G3 & 1 & 0 \\ 0 & 0 & -G4 & 1 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u$$

As shown earlier, determinant [P] = 0 is the characteristic equation. Hence the characteristic equation is

$$1 + G1G2G5 + G1G2G3G6 + G1G2G3G4 = 0 \qquad (2.14)$$

For the block diagram shown in Figure 5, the system equations can be written in matrix form as

$$\begin{bmatrix} 1 + G1G2G5 & G6 & 1 \\ G1G2G3 & -1 & 0 \\ 0 & G4 & -1 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \\ R3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u$$

On evaluating the determinant of [P], we see that we obtain the same form of characteristic equation as in Equation (2.14).

For the block diagram shown in Figure 6, the two system equations are written in matrix form as

$$\begin{bmatrix} 1 + G1G2G5 + G1G2G3G6 & 1 \\ -G1G2G3G4 & 1 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

We can see that determinant [P] = 0 has the same form of characteristic equation as for the two cases shown above.

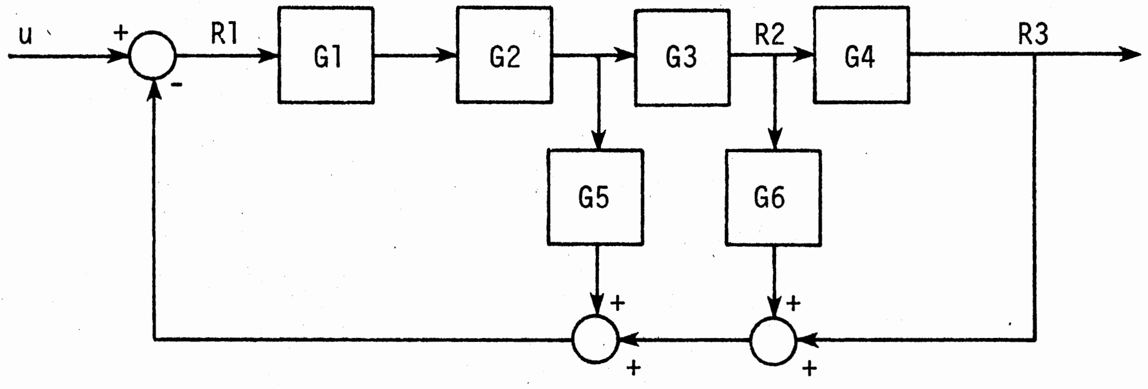


Figure 5. Block Diagram With Three Outputs and One Input

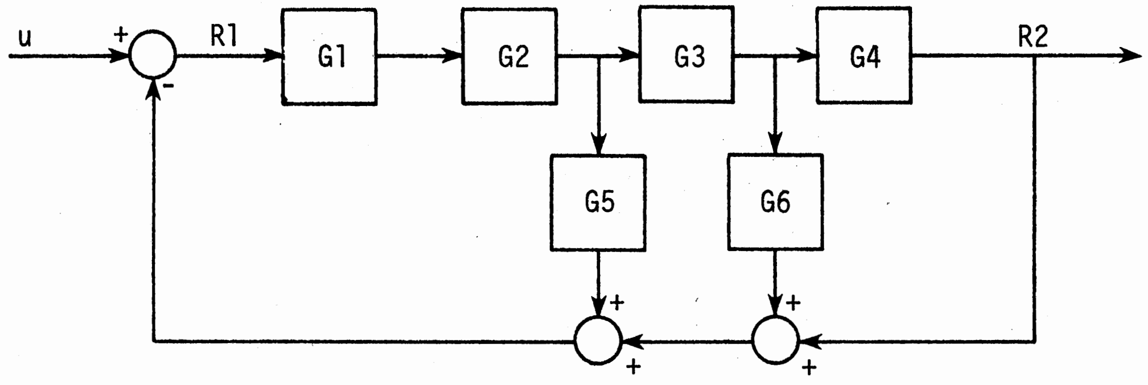


Figure 6. Block Diagram With Two Outputs and One Input

CHAPTER III

THE SOLUTION APPROACH

3.1 Introduction

In the previous chapter, we have shown that determinant [P] = 0 is the characteristic equation. Hence, to solve our problem, a method for evaluating the determinant and a method for evaluating the roots of the characteristic equation have to be found. For large complicated systems, the [P] matrix will be large with a number of zero elements in it and hence the computer storage and time for evaluating the determinant will be large. In order to reduce computer storage and time, a sparse matrix method has been used to evaluate the determinant. The characteristic equation as obtained by evaluating the determinant of the [P] matrix is complex and is not in polynomial form. To determine the roots of the characteristic equation, Muller's method has been used. In this chapter, the sparse matrix method for evaluating the determinant of the [P] matrix, a brief description of Muller's method, and a brief description of the computer program will be presented.

3.2 Evaluation of the Determinant

A computer program has been developed by Key (13) for the solution of large, sparse, unsymmetric systems of linear equations. The program can solve large systems of sparse arrays in core with minimum computer storage and computer time. It stores only a limited number of zeros in addition to non-zero coefficients.

Key tried several techniques for selecting the pivotal element which would reduce storage and yield reasonable accuracy and computer time. The minimum row-minimum column pivot selector was found to be consistently faster. From an accuracy point of view also, it ranked the best. A slightly modified version of Key's program, called subroutine SPARSE, has been used in the frequency response analysis program developed by the School of Mechanical and Aerospace Engineering, Oklahoma State University (9). A brief description of the program along with the modifications made in it is given below. For a detailed description of this procedure, refer to Appendix B.

Subroutine SPARSE is a linear equation solver which is designed to use a sparse coefficient matrix A(s). It is a modification of the method by Key (13) and it allows the solution of complex equations and the presence of zero terms in the [A] matrix passed to SPARSE.

The actual solution of the compressed equations consists of "NEQN" passes through a loop in the routine. On each pass a pivotal element is chosen as the term at the intersection of the row with the least number of non-zero terms and the column in that row with the least number of non-zero terms. In the case of a tie, the first minimum row (or column) is used for the pivot selection. Before entering the routine, an initial value is assigned to the determinant. Every time a new pivot element is selected, it is multiplied with the previous value of the determinant. A final value for the determinant is obtained when all of the pivot elements are selected.

3.3 Solution of the Characteristic Equation

Many methods have been developed for the solution of algebraic equations by an automatic computer. One of the most powerful methods has been developed by Muller (14). It is capable of solving algebraic equations of high degree with complex coefficients in a relatively small number of iterations. A brief description of the method is given below followed by a discussion of the advantages and disadvantages of this method.

Muller's method finds the roots of an algebraic equation by an iterative procedure. Successive iterations towards a particular root are obtained by finding the nearer root of a quadratic whose curve passes through the last three points. The quadratic will, in general, have complex coefficients and complex roots. A detailed description of this method can be found in Appendix C.

3.4 Advantages and Disadvantages of Muller's Method

The advantages of Muller's method are:

1. Although the method is rather complicated, no evaluation of the derivatives of the function and only one evaluation of the function is required per iteration. This makes it suitable for solving algebraic equations by an automatic computer.

2. The time spent per iteration is less with this process than with iterative schemes which require the calculation of derivatives, whenever the degree of the equation is large.

3. Convergence occurs for most polynomial equations, in spite of

the fact that convergence has only been proved for single and double roots when the process has brought one to the neighborhood of a root.

4. Muller's method has a high speed of convergence. It can solve equations of high degree in a relatively small number of iterations.

The disadvantages of Muller's method are:

1. No general proof of convergence has been obtained for this method, although convergence can be shown to occur whenever the process leads one sufficiently close to a single or double root.

2. As shown in the previous chapter, the characteristic equation to be solved is of the form 1 + N1/D1 + N2/D2 + . . . Np/Dp = 0, where N1/D1, N2/D2, . . . Np/Dp are ratios of polynomials. Muller's method does not converge in this case unless the initial guesses are provided very close to the actual roots. This is due to the fact that the value of the function tends to infinity whenever the root tends towards one of the open-loop poles. Since it is very difficult to provide initial guesses close to the actual roots, especially for complicated systems, the above form of the characteristic equation is premultiplied by the denominators of all the G's, the individual transfer functions. The characteristic equation, premultiplied as above, will have the same roots as the original characteristic equation.

3.5 A Brief Description of the Program

The program RTLOC is a general purpose program which can be used to perform root locus analysis. It consists of a main program RTLOC, a complex function subprogram F(s), subroutine SPARSE and subroutine ZANLYT. The main program RTLOC reads the input variables and calls subroutine ZANLYT to solve for the roots of the characteristic equation. The complex function subprogram F(s) defines the characteristic equation whose roots are to be found by ZANLYT. Subroutine SPARSE is a linear equation solver which has been modified so that it can evaluate the determinant of any matrix. Subroutine ZANLYT is an International Mathematical and Statistical Library (15) subroutine which uses Muller's method to evaluate the roots of the characteristic equation. In addition to the above routines, the user has to write two subprograms, SMCON or AAECON and SMCOEF or AAECOEF. In subroutine SMCOEF or AAECOEF, the user has to define the [P] matrix and in subroutine SMCON or AAECON, the user has to define all the transfer functions.

Users who wish to use the program need to do the following:

1. Develop a matrix system model of the form

$$P(s)R(s) = Q(s)U(s)$$

where R(s) is the vector of unknown signals, Q(s)U(s) describes the input to the system, and P(s) describes the system.

2. Code a subroutine SMCOEF or AAECOEF which defines Q(s) and a compressed P(s).

3. Code a subroutine SMCON or AAECON which defines all the transfer functions.

4. The user has to input the following variables:

EPS	first stopping criterion for ZANLYT (1.0e-7 is a suggested value)
NSIG	second stopping criterion for ZANLYT. A root is accepted if two successive approximations to a given root agree in the first NSIG digits.
KN	the number of known roots which must be stored in X(1), X(2) . . . X(KN), prior to entry to ZANLYT.

NGUESS - the number of initial guesses provided.

ITMAX - the maximum number of iterations allowed per root.

L - subscript of the parameters being varied.

M - the number of new roots to be found.

NEQN - the number of equations to be solved.

ZTEST - absolute minimum at which terms are set to zero in SPARSE.

SOMVAL - the final value of the parameter.

P(L) - the initial value of the parameter which is varied.

5. To premultiply the determinant the user has to provide the poles of all the individual transfer functions.

6. All the arrays in the dimension, complex, and integer statements need to be checked for each individual problem. The NROW and NCOL arguments in the call statements for SMCOEF or AAECOEF and ZANLYT have to be changed for each individual problem.

The denominators of the transfer functions can be of any of the following forms.

1. $As^2 + Bs + C$ where A, B, and C may be expressed as constants or in terms of parameter values.

2. $s^2 + 2\tau w_n s + w_n$ where τ, w_n may be expressed as constants or in terms of parameter values.

3. $\tau s + 1$ where τ may be a constant or a parameter value.

4. s+CN where CN may be a complex number or a parameter.

On output, X refers to the roots of the characteristic equation, P refers to the parameter value, INFER refers to the number of iterations required per root, and IER provides the error code. A complete listing of the program is provided in Appendix A.

CHAPTER IV

ILLUSTRATIVE EXAMPLES

To illustrate the use of the program RTLOC, three examples will be worked in this chapter. Each example will comprise of a block diagram, the system equations, the equations in matrix form, user input, the results, and a brief discussion. Results for the first two are plotted.

4.1 Example 1

A block diagram of a third-order system (7) whose root locus is desired is shown in Figure 7.

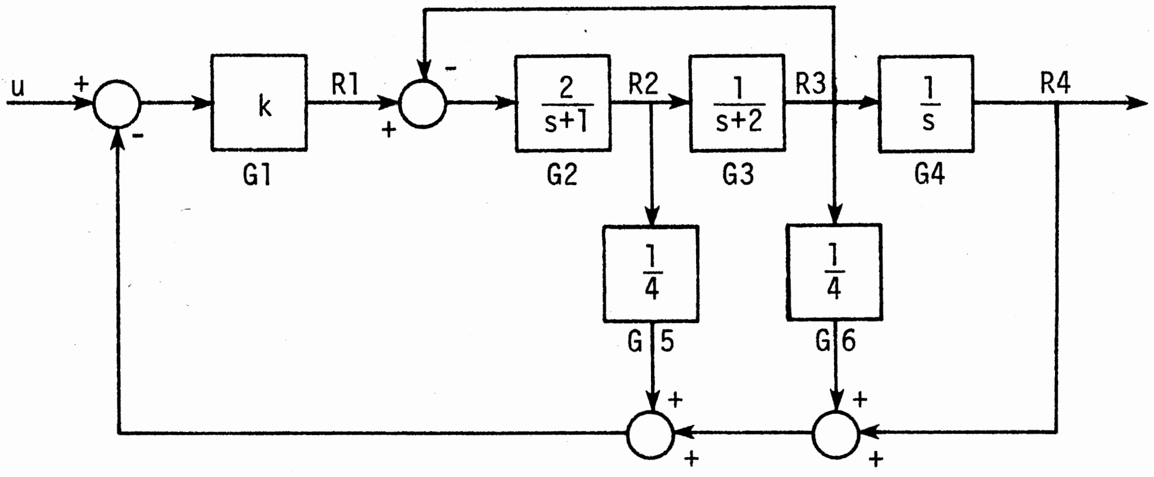


Figure 7. Block Diagram for Example 1

The system equations are:

$R1 = (U - R2G5 - R3G6 - R4)G1$

$R2 = (R1 - R3)G2$

$R3 = R2G3$

$R4 = R3G4$

The system equations in matrix form are:

$$\begin{bmatrix} \frac{1}{G1} & G5 & G6 & 1 \\ -G2 & 1 & G2 & 0 \\ 0 & -G3 & 1 & 0 \\ 0 & 0 & -G4 & 1 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u$$

The SMCON and SMCOEF listings for this example are given in Appendix D.

The poles of the individual transfer functions are:

AL(1) = (-1.0, 0.0)

AL(2) = (-2.0, 0.0)

AL(3) = (0.0, 0.0)

The input variables for this example are given below:

EPS = 1.0 E - 7	M = 3
NSIG = 9	NEQN = 4
KN = 0	ZTEST = 0.0
NGUESS = 0	SOMVAL = 0.12
ITMAX = 100	P(L) = 2048.0
L = 20	

A listing of the output is presented in Table I and a plot of the root locus is presented in Figure 8.

TABLE I

RESULTS FOR EXAMPLE 1

```
THE ROOTS OF THE EQUATION FOR P( 20)=       2048.0000 ARE
    X(  1)=         -1.5000         1.3229
    X(  2)=         -1.5000        -1.3229
    X(  3)=      -1024.0000          .0000
THE ROOTS OF THE EQUATION FOR P( 20)=        512.0000 ARE
    X(  1)=         -1.5000         1.3229
    X(  2)=         -1.5000        -1.3229
    X(  3)=       -256.0000          .0000
THE ROOTS OF THE EQUATION FOR P( 20)=        128.0000 ARE
    X(  1)=         -1.5000         1.3229
    X(  2)=         -1.5000        -1.3229
    X(  3)=        -64.0000         -.0000
THE ROOTS OF THE EQUATION FOR P( 20)=         32.0000 ARE
    X(  1)=         -1.5000         1.3229
    X(  2)=         -1.5000        -1.3229
    X(  3)=        -16.0000          .0000
THE ROOTS OF THE EQUATION FOR P( 20)=          8.0000 ARE
    X(  1)=         -1.5000         1.3229
    X(  2)=         -1.5000        -1.3229
    X(  3)=         -4.0000         -.0000
THE ROOTS OF THE EQUATION FOR P( 20)=          2.0000 ARE
    X(  1)=         -1.0000          .0000
    X(  2)=         -1.5000        -1.3229
    X(  3)=         -1.5000         1.3229
THE ROOTS OF THE EQUATION FOR P( 20)=           .5000 ARE
    X(  1)=          -.2500         0.0000
    X(  2)=         -1.5000         1.3229
    X(  3)=         -1.5000        -1.3229
THE ROOTS OF THE EQUATION FOR P( 20)=           .1250 ARE
    X(  1)=          -.0625         0.0000
    X(  2)=         -1.5000         1.3229
    X(  3)=         -1.5000        -1.3229
```

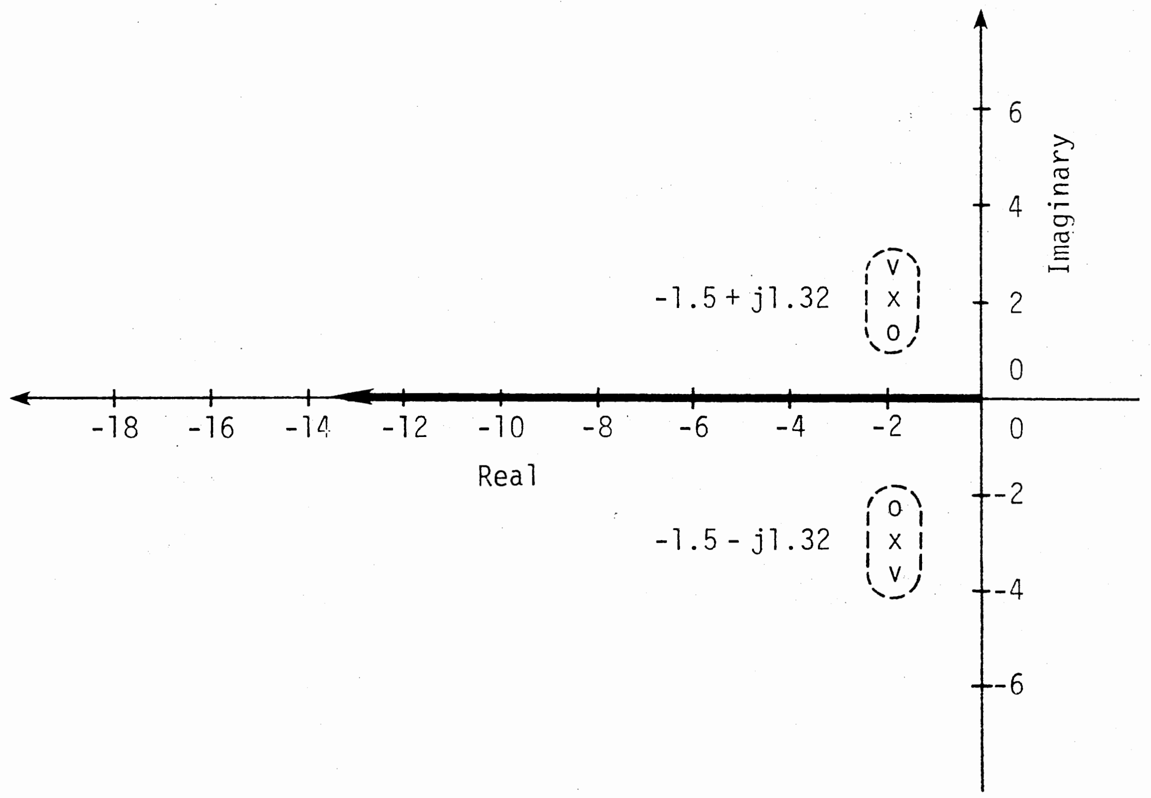


Figure 8. Root Locus Plot for Example 1

In the above problem, the complex conjugate poles are insensitive to the gain because of the pole-zero cancellation. It can be seen that one of the roots of the characteristic equation moves along the negative real axis as the gain is varied. The root locus moves through (0.0,0.0), (-1.0,0.0), and (-2.0,0.0), which are the poles of G's, the individual transfer functions.

4.2 Example 2

The block diagram for this problem (7) is shown in Figure 9.

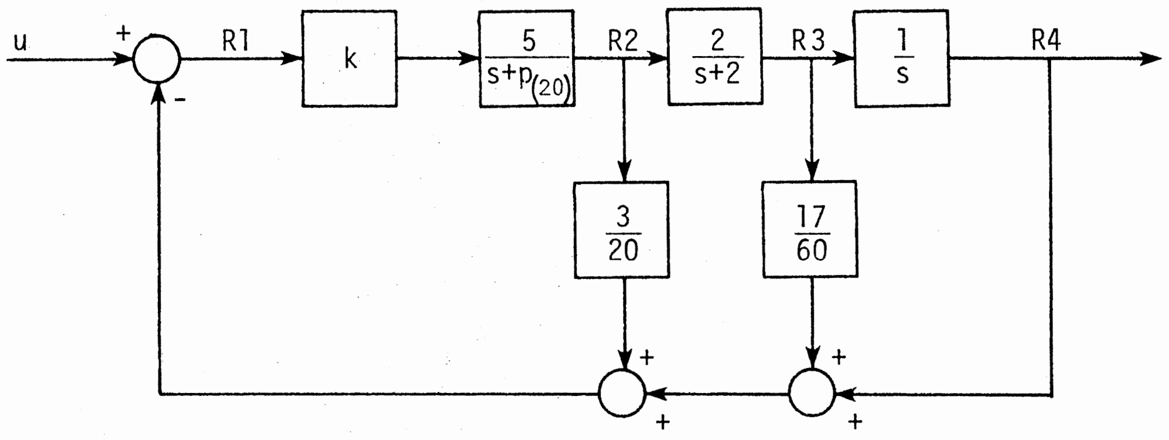


Figure 9. Block Diagram for Example 2

The system equations are:

R1 = U - R3G6 - R4 - R2G5

R2 = R1G1G2

R3 = R2G3

R4 = R3G4

The system equations in matrix form are:

$$\begin{bmatrix} 1 & G5 & G6 & 1 \\ -G1G2 & 1 & 0 & 0 \\ 0 & -G3 & 1 & 0 \\ 0 & 0 & -G4 & 1 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u$$

The SMCON and SMCOEF listings for this example are given in Appendix E.

The poles of the individual transfer functions are:

AL(1) = (-8.0, 0.0)

AL(2) = -P(20)

AL(3) = (0.0, 0.0)

The input variables for this example are given below:

EPS = 1.0 E - 7	M = 3
NSIG = 9	NEQN = 4
KN = 0	ZTEST = 0.0
NGUESS = 0	SOMVAL = 0.12
ITMAX = 100	P(L) = 512.0
L = 20	

A listing of the output is presented in Table II and a plot of the root locus is presented in Figure 10.

In the conventional root locus, the system parameter that is allowed to vary is the gain K. In the above example the effect of varying a parameter other than the gain K on the roots of the characteristic equation has been illustrated.

TABLE II

RESULTS FOR EXAMPLE 2

```
THE ROOTS OF THE EQUATION FOR P( 20)=        512.0000 ARE
    X(  1)=          -.0137        0.0000
    X(  2)=        -17.0545        0.0000
    X(  3)=       -511.9318        0.0000
THE ROOTS OF THE EQUATION FOR P( 20)=        128.0000 ARE
    X(  1)=          -.0545        0.0000
    X(  2)=        -17.2442        0.0000
    X(  3)=       -127.7014        0.0000
THE ROOTS OF THE EQUATION FOR P( 20)=         32.0000 ARE
    X(  1)=          -.2114        0.0000
    X(  2)=        -19.1602        0.0000
    X(  3)=        -29.6284        0.0000
THE ROOTS OF THE EQUATION FOR P( 20)=          8.0000 ARE
    X(  1)=          -.7961        0.0000
    X(  2)=        -12.1019       -2.0673
    X(  3)=        -12.1019        2.0673
THE ROOTS OF THE EQUATION FOR P( 20)=          2.0000 ARE
    X(  1)=         -2.0000       -2.0000
    X(  2)=         -2.0000        2.0000
    X(  3)=        -15.0000         .0000
THE ROOTS OF THE EQUATION FOR P( 20)=           .5000 ARE
    X(  1)=         -1.1368       -2.5669
    X(  2)=         -1.1368        2.5669
    X(  3)=        -15.2264        -.0000
THE ROOTS OF THE EQUATION FOR P( 20)=           .1250 ARE
    X(  1)=          -.9254       -2.6458
    X(  2)=          -.9254        2.6458
    X(  3)=        -15.2743         .0000
```

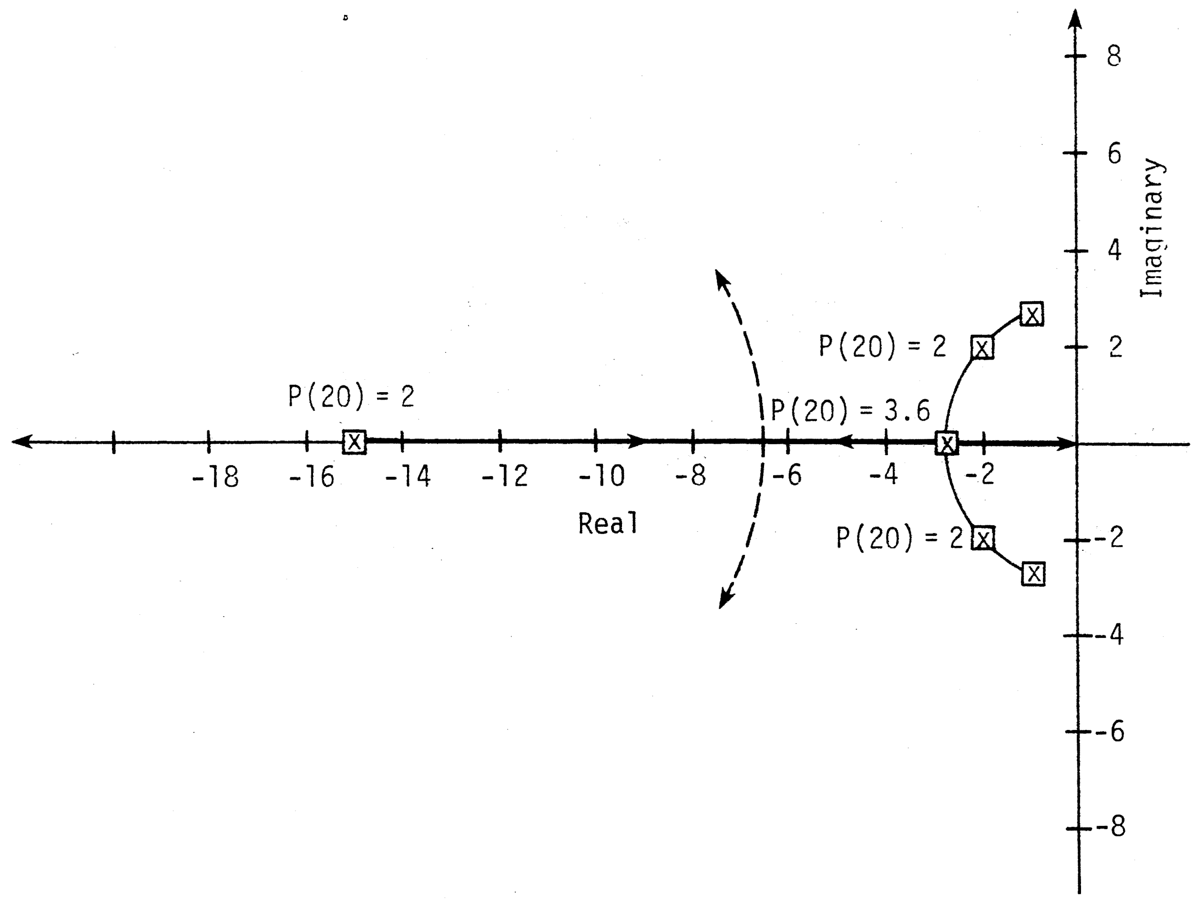


Figure 10. Root Locus Plot for Example 2

4.3 Example 3

The block diagram for this problem is shown in Figure 11. The poles of the individual transfer functions are defined in subroutine AAECON. The AAECON and AAECOEF listings for this example are given in Appendix F.

The input variables for this example are:

EPS = 1.0 E - 7	M = 32
NSIG = 7	NEQN = 16
KN = 0	ZTEST = 0.0
NGUESS = 0	SOMVAL = 870.0
ITMAX = 100	P(L) = 3492.0
L = 30	

A listing of the output is presented in Table III.

This example illustrates the fact that our technique can be used even for very high order systems. The system considered in this example is of order 32, and the effect of varying the various parameters in the system on the roots of the characteristic equation can be clearly seen.

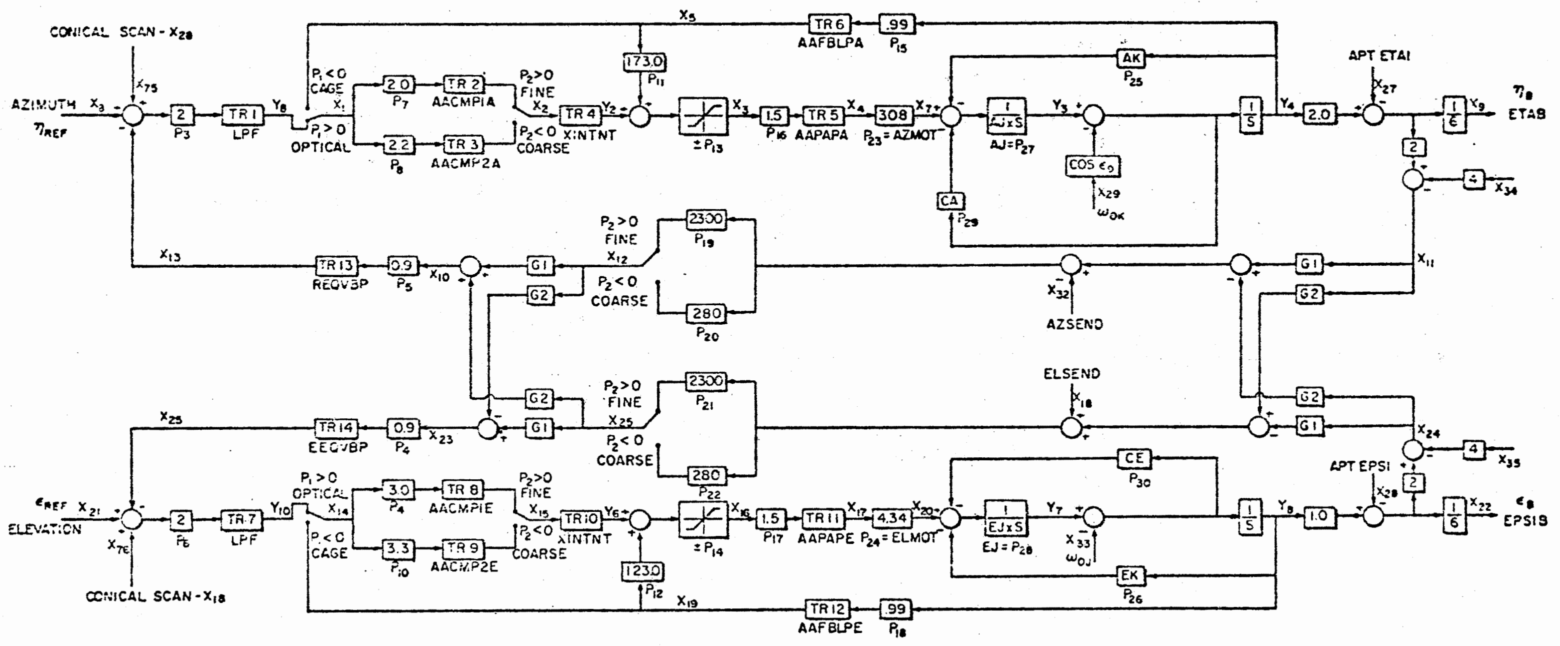


Figure 11. Block Diagram for Example 3

TABLE III

RESULTS OF EXAMPLE 3

```
THE ROOTS OF THE EQUATION FOR P( 30)=        3492.0000 ARE
    X(  1)=      -12.6574          -.0000
    X(  2)=      -12.5629           .0000
    X(  3)=    -1156.0517      -1074.9228
    X(  4)=    -1341.5680       1102.8207
    X(  5)=    -1156.0517       1074.9228
    X(  6)=    -1341.5680      -1102.8207
    X(  7)=    -1565.8951      -2724.8890
    X(  8)=    -1566.4887      -2724.1582
    X(  9)=    -6912.5345           .0000
    X( 10)=    -5570.9118          -.0000
    X( 11)=    -7933.6359      -6662.1017
    X( 12)=    -7550.9299       6495.4344
    X( 13)=   -31400.0041          -.0003
    X( 14)=   -31399.9992           .0009
    X( 15)=    -1566.4887       2724.1582
    X( 16)=    -1565.8951       2724.8890
    X( 17)=    -7550.9299      -6495.4344
    X( 18)=    -7933.6359       6662.1017
    X( 19)=   -29258.7881     -13332.8473
    X( 20)=   -29263.6009     -13196.5353
    X( 21)=   -41888.1943      -6846.4394
    X( 22)=   -41780.1803      -6779.9839
    X( 23)=   -29258.7881      13332.8473
    X( 24)=   -41888.1943       6846.4394
    X( 25)=   -29263.6009      13196.5353
    X( 26)=   -41780.1803       6779.9839
    X( 27)=   -50000.5301     -86602.6806
    X( 28)=   -50000.5301      86602.6806
    X( 29)=   -50000.5016     -86602.6731
    X( 30)=   -50000.5016      86602.6731
    X( 31)=  -250000.0019           .0000
    X( 32)=  -250000.0020           .0000
THE ROOTS OF THE EQUATION FOR P( 30)=         873.0000 ARE
    X(  1)=      -12.6574          -.0000
    X(  2)=      -12.5629           .0000
    X(  3)=    -1156.0517      -1074.9228
    X(  4)=    -1341.5680       1102.8207
    X(  5)=    -1156.0517       1074.9228
    X(  6)=    -1341.5680      -1102.8207
    X(  7)=    -1565.8951      -2724.8890
    X(  8)=    -1566.4887      -2724.1582
    X(  9)=    -6912.5345           .0000
    X( 10)=    -5570.9118          -.0000
    X( 11)=    -7933.6359      -6662.1017
    X( 12)=    -7550.9299       6495.4344
    X( 13)=   -31400.0041          -.0003
    X( 14)=   -31399.9992           .0009
```

TABLE III (CONTINUED)

```
X( 15)=      -1566.4887       2724.1582
X( 16)=      -1565.8951       2724.8890
X( 17)=      -7550.9299      -6495.4344
X( 18)=      -7933.6359       6662.1017
X( 19)=     -29258.7881     -13332.8473
X( 20)=     -29263.6009     -13196.5353
X( 21)=     -41888.1943      -6846.4394
X( 22)=     -41780.1803      -6779.9839
X( 23)=     -29258.7881      13332.8473
X( 24)=     -41888.1943       6846.4394
X( 25)=     -29263.6009      13196.5353
X( 26)=     -41780.1803       6779.9839
X( 27)=     -50000.5301     -86602.6806
X( 28)=     -50000.5301      86602.6806
X( 29)=     -50000.5016     -86602.6731
X( 30)=     -50000.5016      86602.6731
X( 31)=    -250000.0019          .0000
X( 32)=    -250000.0020          .0000
```

CHAPTER V

RECOMMENDATIONS AND CONCLUSIONS

A new computer-oriented technique for root-locus analysis has been developed. It is capable of determining the roots of the characteristic equation for systems of high order. The system equations have been represented in the form [P][R] = [Q][U], where [R] is an output vector of dimension l, [U] is a control input vector of dimension m, [P] is a lxl transformation matrix, and [Q] is a lxm control gain matrix. It has been shown that when the system is represented in this manner, determinant [P] = 0 is the characteristic equation. [P] is a function of the Laplace variables, and its determinant is evaluated using a sparse matrix method which reduces computer storage and computer time. The roots of the characteristic equation have been evaluated using Muller's algorithm. One of the main features of this technique is that the characteristic equation need not be in polynomial form. Muller's algorithm is one of the fastest converging algorithms (8) developed so far and it enables the roots of the characteristic equation to be obtained in a relatively small number of iterations.

Although Muller's method is supposed to be capable of solving complex algebraic equations, no general proof of convergence has been obtained for this method so far. It has been found that this method does not converge when there are discontinuities in the characteristic equation unless initial guesses are provided very close to the actual roots.

The effectiveness of this technique will be considerably improved if a method more efficient than Muller's method and which has a definite proof of convergence is developed.

BIBLIOGRAPHY

(1) Evans, W. R. "Control System Synthesis by Root Locus Method." AIEE Trans., Part II, Vol. 69 (1950), pp. 66-69.

(2) Bendrikov, G. A., and K. F. Teodorchik. "The Analytic Theory of Constructing Root Loci." Automat. Remote Control (March, 1959), pp. 340-344.

(3) Krishnan, V. "Semi-Analytic Approach to Root Locus." IEEE Trans. on Automat. Control, Vol. AC-11, No. 1 (January, 1966), pp. 102-108.

(4) Parks, P. C. "A New Proof of the Routh-Hurwitz Stability Criterion Using the Second Method of Liapunov." Proc., Cambridge Phil. Soc., Math. and Phys. Ser., Vol. 58, Part 4 (October, 1962), pp. 694-702.

(5) Shen, D. W. C., and C. F. Chen. "Routh's Method for Approximating Oscillatory Methods in Linear Systems." IEEE Inter. Conv. Rec., Part 2 (March, 1963), pp. 10-15.

(6) Chen, C. F., and C. Hsu. "The Determination of Root Loci Using Routh's Algorithm." Journal of the Franklin Institute, Vol. 281, No. 2 (February, 1966), pp. 114-121.

(7) Schultz, D. G., and J. L. Melsa. Computer Programs for Computational Assistance in the Study of Linear Control Theory. New York: McGraw-Hill, 1970.

(8) Thomas, Robert J. "An Efficient Root Locus Program for Feedback Control System Design." IEEE Trans. on Education (May, 1976), pp. 76-77.

(9) "User Information for the FRQRSP Frequency Response Program Report ER-75-R-109-010." School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, Oklahoma, January, 1978.

(10) Ogata, K. State Space Analysis of Control Systems. Englewood Cliffs, N.J.: Prentice-Hall, 1970.

(11) D'Azzo, J. J., and C. H. Houpis. Feedback Control System Analysis and Synthesis. New York: McGraw-Hill, 1966.

(12) Schultz, D. G., and J. L. Melsa. Linear Control Systems. New York: McGraw-Hill, 1969.

(13) Key, J. E. "Computer Program for Solution of Large, Sparse, Unsymmetric Systems of Linear Equations." Intl. J. for Num. Meth. in Eng., Vol. 6 (1973), pp. 497-509.

(14) Muller, D. E. "A Method for Solving Algebraic Equations Using an Automatic Computer." MTAC (1956), pp. 208-215.

(15) IMSL Reference Manual. 6th ed. Houston: International Mathematical and Statistical Libraries Inc., 1977.

APPENDIX A

PROGRAM LISTINGS

```
      PROGRAM RTLOC(INPUT,OUTPUT=80,TAPE5=INPUT,TAPE6=OUTPUT)
C
      EXTERNAL F
C
      COMPLEX DETRET,X(3)
C
      COMMON /FRQALL/ GLOBAL,NR,NW,ISTART,METHOD
C
      COMMON /SAME/ ITMAX,L,NEQN,ZTEST,M
C
      REAL GLOBAL,P
C
      INTEGER NR,NW,ISTART,METHOD,INFER(3)
C
      DIMENSION P(100),GLOBAL(1000)
C
      EQUIVALENCE (GLOBAL(1),P(1))
C
      READ(5,*)EPS,NSIG,KN,NGUESS,ITMAX,L,M,NEQN,
     1ZTEST,SOMVAL,P(L)
C
10    CALL ZANLYT(F,EPS,NSIG,KN,NGUESS,M,X,ITMAX,INFER,IER)
      PRINT(6,20)L,P(L)
20    FORMAT(5X,33H THE ROOTS OF THE EQUATION FOR P(,I3,2H)=,
     1F15.4,4H ARE)
C
      DO 40 I=1,M
      PRINT(6,30)I,X(I)
30    FORMAT(10X,2HX(,I3,2H)=,(F15.4,F15.4))
40    CONTINUE
C
      P(L)=P(L)/4.0
      IF(P(L).LT.SOMVAL)GO TO 50
      GO TO 10
50    STOP
      END
      COMPLEX FUNCTION F(S)
C
      COMMON AL(3)
C
      COMPLEX S,SS,DETRET,A(4,4),B(4),R(4),C,X,AL
C
      DIMENSION ICOL(4,4),IRENT(4),ICENT(4)
C
      COMMON /SAME/ ITMAX,L,NEQN,ZTEST,M
C
      SS=S
      N=1
      DO 10 I=1,M
      IF(REAL(S).NE.REAL(AL(I)).AND.AIMAG(S).NE.AIMAG(AL(I)))
     1GO TO 10
      IF(CABS(S).EQ.0.0)SS=1.0E-10
10    CONTINUE
C
      CALL SMCON(SS,N)
C
      CALL SMCOEF(A,B,ICOL,4,4,4,S,N,R,-1)
C
      DETRET=(1.0,0.0)
C
C     MULTIPLY THE DETERMINANT BY THE DENOMINATORS OF ALL THE
C     TRANSFER FUNCTIONS
C
      DO 20 I=1,M
      DETRET=DETRET*(SS-AL(I))
20    CONTINUE
C
      CALL SPARSE(A,R,B,ICOL,4,ZTEST,IRENT,ICENT,4,4,
     1IER,DETRET)
C
      IF(IER.NE.2)GO TO 30
C
      STOP
  30  F=DETRET
      RETURN
      END
      SUBROUTINE SPARSE(A,R,B,ICOL,NEQN,ZTEST,IRENT,ICENT,NROW,NCOL,IER
     1,DETRET)
C
C
C     A     = MATRIX CONTAINING (NON-ZERO) COEFFICIENTS OF EQUATIONS
C     R     = SOLUTION VECTOR
C     B     = CONSTANT VECTOR FOR EQUATIONS TO BE SOLVED
C     ICOL  = POSITION OF COEFFICIENTS IN EQUATIONS
C     NEQN  = NUMBER OF EQUATIONS TO BE SOLVED
C     ZTEST= ABSOLUTE MINIMUM AT WHICH TERMS ARE SET TO ZERO
C     IRENT= WORKING VECTOR OF LENGTH NROW
C     ICENT= WORKING VECTOR OF LENGTH NROW
C     NROW  = NUMBER OF ROWS IN A AND ICOL
C     NCOL  = NUMBER OF COLUMNS IN A AND ICOL
C     IER   = ERROR FLAG
C             0 - NO ERRORS
C             1 - A IS SINGULAR
C             2 - NCOL IS TOO SMALL TO ALLOW A SOLUTION
C     DETRET=VALUE OF THE DETERMINANT
C
      DIMENSION A(NROW,NCOL),R(NROW),B(NROW),ICOL(NROW,NCOL),
     1IRENT(NROW),ICENT(NROW)
C
      INTEGER PIVROW,PIVCOL
```

```
C
      REAL XX(2)
C
C
      COMPLEX A,R,B,C,X,DET,DETRET
C
      EQUIVALENCE (XX(1),X)
C
C     ZERO COLUMN ENTRY COUNTER (IF NOT DONE BY CALLING PROGRAM)
C
      DO 5 I=1,NEQN
    5 ICENT(I)=0
      NK=NCOL-1
      DO 70 I=1,NEQN
C
C     REMOVE ZEROS FROM A
C
      DO 40 J=1,NCOL
   10 IF (ICOL(I,J).EQ.0) GO TO 50
      X=A(I,J)
      IF (XX(1).NE.0.0) GO TO 40
C
C     THE FOLLOWING TEST IS ONLY REQUIRED FOR COMPLEX EQUATIONS
C
C
      IF (XX(2).NE.0.0) GO TO 40
C
      IF (J.EQ.NCOL) GO TO 30
      DO 20 K=J,NK
      ICOL(I,K)=ICOL(I,K+1)
      IF (ICOL(I,K).EQ.0) GO TO 10
   20 A(I,K)=A(I,K+1)
   30 ICOL(I,NCOL)=0
      GO TO 10
   40 CONTINUE
C
C     COUNT ROW AND COLUMN ENTRIES
C
   50 DO 60 J=1,NCOL
      IC=ICOL(I,J)
      IF (IC.EQ.0) GO TO 70
      ICENT(IC)=ICENT(IC)+1
   60 CONTINUE
      J=NCOL+1
   70 IRENT(I)=J-1
      DET=(1.0,0.0)
      IF(LOCF(DETRET).NE.0)DET=DETRET
      DO 230 LKJ=1,NEQN
C
C     SELECT FIRST MIN ROW THEN FIRST MIN COL
```

```
C
C     SELECT ROW WITH MINIMUM ENTRIES
      IK=1000000
      DO 80 I=1,NEQN
      IR=IRENT(I)
      IF (IR.LE.0.OR.IR.GE.IK) GO TO 80
      PIVROW=I
      IK=IR
   80 CONTINUE
C
C     SELECT SMALLEST AVAILABLE COLUMN FROM PIVROW
C
      IK=1000000
      IR=IRENT(PIVROW)
      DO 90 I=1,IR
      II=ICOL(PIVROW,I)
      IC=ICENT(II)
      IF (IC.LE.0.OR.IC.GE.IK) GO TO 90
      PIVCOL=II
      IK=IC
      IY=I
   90 CONTINUE
C
C     END FIRST MIN ROW THEN FIRST MIN COL
C
C
C     NORMALIZE PIVROW
C
      DET=DET*A(PIVROW,IY)
      X=1.0/A(PIVROW,IY)
      IC=IRENT(PIVROW)
      DO 100 J=1,IC
  100 A(PIVROW,J)=A(PIVROW,J)*X
      A(PIVROW,IY)=1.0
      B(PIVROW)=B(PIVROW)*X
C
C     SELECT ROWS THAT CAN BE OPERATED ON
C
      DO 210 I=1,NEQN
      IF (ICENT(PIVCOL)-1) 250,220,110
  110 IF (I.EQ.PIVROW) GO TO 210
      IC=IABS(IRENT(I))
      DO 120 J=1,IC
      IF (ICOL(I,J).EQ.PIVCOL) GO TO 130
  120 CONTINUE
      GO TO 210
C
C     IF YOU CAN GET TO THIS POINT ROW I CONTAINS PIVOTAL COLUMN
C
  130 C=-A(I,J)
```

```
      B(I)=B(I)+C*B(PIVROW)
      DO 190 JKPI=1,NCOL
      II=ICOL(PIVROW,JKPI)
      IF (II.EQ.0) GO TO 200
      DO 140 NK=1,IC
      IF (ICOL(I,NK).EQ.II) GO TO 150
  140 CONTINUE
C
C     ROW I DOES NOT CONTAIN THIS ELEMENT - ADD ELEMENT TO ROW I
C
      IC=IC+1
C     IF (IC.GT.JCOL) JCOL=IC
      IF (IC.GT.NCOL) GO TO 260
      A(I,IC)=C*A(PIVROW,JKPI)
      ICOL(I,IC)=II
      ICENT(II)=ICENT(II)+ISIGN(1,ICENT(II))
      GO TO 190
C
C     ROW I CONTAINS THIS ELEMENT - ADD TO IT
C
  150 IF (II.EQ.PIVCOL) GO TO 160
      AMIN=AMIN1(CABS(A(I,NK)),CABS(A(PIVROW,JKPI)))
      A(I,NK)=A(I,NK)+C*A(PIVROW,JKPI)
C
C     TEST THE ELEMENT TO SEE IF IT WAS ELIMINATED
C
      IF (CABS(A(I,NK))/AMIN.GT.ZTEST) GO TO 190
  160 ICENT(II)=ICENT(II)-ISIGN(1,ICENT(II))
      IF (ICENT(II).EQ.0) GO TO 250
      IC=IC-1
      IF (IC.EQ.0) GO TO 250
      IF (NK.GT.IC) GO TO 180
      DO 170 J=NK,IC
      A(I,J)=A(I,J+1)
  170 ICOL(I,J)=ICOL(I,J+1)
  180 ICOL(I,IC+1)=0
  190 CONTINUE
  200 IRENT(I)=ISIGN(IC,IRENT(I))
  210 CONTINUE
C
C     ELIMINATE PIVROW AND PIVCOL FROM BEING CONSIDERED AGAIN
C
  220 IRENT(PIVROW)=-IRENT(PIVROW)
  230 ICENT(PIVCOL)=-ICENT(PIVCOL)
C
C     UNSCRAMBLE AND STORE SOLUTION IN R
C
      DO 240 I=1,NEQN
      II=ICOL(I,1)
  240 R(II)=B(I)
```

```
      IF(LOCF(DETRET).NE.0)DETRET=DET
      IER=0
      GO TO 270
  250 IER=1
      IF(LOCF(DETRET).NE.0)DETRET=(0.0,0.0)
      GO TO 270
  260 IER=2
C
  270 RETURN
      END
*EOR  END-OF-FILE  1.
```

APPENDIX B

SPARSE MATRIX METHOD FOR EVALUATING THE DETERMINANT

A computer program has been developed by Key (13) for the solution of large, sparse, unsymmetric systems of linear equations. The program can solve large systems of sparse arrays in core with minimum computer storage and computer time. It stores only a limited number of zeros in addition to non-zero coefficients.

Key tried several techniques for selecting the pivotal element which would reduce storage and yield reasonable accuracy and computer time. From this selection, one technique was found to be the best for the test cases tried.

Basically the method is based on three elementary row operations:

1. Interchange of any two rows.
2. Multiplication of a row by a scalar.
3. Addition of a multiple of one row to another row.

To aid in the selection of the pivotal element, two additional arrays are generated. One array contains the number of non-zero elements in each row and the other contains the number of non-zero elements in each column. These arrays are updated each time an element is eliminated or generated, so that the current row and column count are available for pivot selection. The pivot selector determines the pivotal row and pivotal column relative to the full matrix and the actual position in storage of the condensed array.

In subroutine SPARSE, the minimum row-minimum column pivot selection technique is used. In this pivot selection technique, the IRENT array is searched to find the row with the least number of non-zero coefficients that has not been previously selected as the pivotal row. In the event that two or more rows satisfy the above requirements, the row with the smallest row index is selected. Once the pivot row has been established, the columns that correspond to non-zero elements in the pivotal row are examined to select the column with the least number of entries. This is accomplished by testing selected elements of the ICENT array. In the event that two or more columns contain the same number of elements, the column with the smallest index is selected as the pivotal column.

The program requires the input of three arrays: an array A, containing non-zero coefficients of equations; an array B, a constant equation for equations to be solved; and an array ICOL containing the position of the coefficients in the equations. The number of columns required for storage must be equal to or greater than the number of entries in the longest row.

Upon entry, the number of non-zero elements in each column and the number of non-zero elements in each row are computed and stored in the ICENT and IRENT arrays, respectively, and the initial value of the determinant is set to one. At this point the initialization is complete and the remainder of the program is contained within three nested loops. The outer loop selects a new pivotal element on each pass. The determinant is obtained by multiplying all the pivotal elements.

Once the pivotal element has been selected, the pivotal row is normalized by dividing the row by the pivotal element. Since the pivotal element is known to be unity after normalization, it is set to one as a precaution against round-off error.

The second loop is entered, which involves a row-by-row search for rows containing elements in the pivotal column. If the number of entries in the pivotal column has been reduced to one entry, there is no need to continue the row-by-row search and the program is branched back to the pivot selector to select a new element. Also, if the pivotal row is selected in the row-by-row search, all further tests are by-passed and the next row is selected since operations are not permitted on the pivotal row. Finally, the inner loop is a column-by-column search of each row to determine if the row contains the pivotal element. At this point there are three alternatives available:

1. If the column index in the row being searched is less than the pivotal column, it is necessary to continue searching the row.

2. If the column index is greater than the pivotal column, the row does not contain the pivotal column and a new row must be selected.

3. If the column index is equal to the pivotal column, the row contains the pivotal element and the row can be operated on by the pivotal row.

If the conditions of the third alternative are met, the pivotal row is multiplied by the negative of the element in the pivotal column of the row being operated on. Then the two rows are added in a manner that is consistent with the storage scheme. The element being eliminated is simply dropped from consideration by moving all entries to its right one space to the left. All elements remaining in the row are compared to ZTEST to see if any elements other than the element in the pivotal column were eliminated. If so, the row is further compressed to eliminate the zero entry from the row. Finally, the row is tested to see if the row count is zero, which indicates a singularity.

The minimum row-minimum column pivot selector has been found to be consistently faster and eliminates fewer terms than the other pivot selectors tested. From an accuracy point of view also, this pivot selector always ranked among the best.

APPENDIX C

MULLER'S METHOD

Many methods have been developed for solving algebraic equations and several of these have been used with automatic computers. One of the most powerful methods has been developed by Muller (14). A description of the method is provided in the following paragraphs.

Each root of any complex algebraic equation is found by an iterative procedure. Successive iterations towards a particular root are obtained by finding the nearer root of a quadratic whose curve passes through the last three points. The quadratic will in general have complex coefficients and complex roots. This solution is accompanied by a variation of the standard quadratic formula.

Let

$$f(x) = a_0x^n + a_1x^{n-1} + \ldots + a_n = 0$$

where the coefficients $a_0, a_1, \ldots, a_n$ are complex numbers. Suppose that (x_{i-2}, f_{i-2}), (x_{i-1}, f_{i-1}) and (x_i, f_i) are three points on a curve. We can then find a parabola $y = ax^2 + bx + c$ passing through these points. The equation of the parabola can be written directly.

$$y = \frac{(x - x_{i-1})(x - x_i)}{(x_{i-2} - x_{i-1})(x_{i-2} - x_i)} f_{i-2} + \frac{(x - x_{i-2})(x - x_i)}{(x_{i-1} - x_{i-2})(x_{i-1} - x_i)} f_{i-1}$$
$$+ \frac{(x - x_{i-2})(x - x_{i-1})}{(x_i - x_{i-2})(x_i - x_{i-1})} f_i$$

This equation is of the second degree and, as is easily found, it is satisfied by the coordinates of the three points. Putting $h = x - x_i$, $h_i = x_i - x_{i-1}$, $h_{i-1} = x_{i-1} - x_{i-2}$, we obtain

$$y = \frac{(h + h_i)h}{-h_{i-1}(-h_{i-1} - h_i)} f_{i-2} + \frac{(h + h_i + h_{i-1})}{h_{i-1}(-h_i)} f_{i-1}$$

$$+ \frac{(h + h_i + h_{i-1})(h + h_i)}{(h_i + h_{i-1})h_i} f_i$$

Further introducing $\lambda = h/h_i$, $\lambda_i = h_i/h_{i-1}$ and $\delta = 1 + \lambda_i$, we get:

$$y = \frac{1}{\delta_i} [\lambda(\lambda+1)\lambda_i^2 f_{i-2} - \lambda(\lambda+1+\lambda_i^{-1})\lambda_i \delta_i f_{i-1}$$

$$+ (\lambda+1)(\lambda+1+\lambda_i^{-1})\lambda_i f_i]$$

$$= \lambda^2 \delta_i^{-1} [f_{i-2}\lambda_i^2 - f_{i-1}\lambda_i\delta_i + f_i\lambda_i]$$

$$+ \lambda\delta_i^{-1}[f_{i-2}\lambda_i^2 - f_{i-1}\delta_i^2 + f_i(\lambda_i + \delta_i)] + f_i$$

This equation is equated to zero and divided by $f_i\lambda^2$, and then solved for $1/\lambda$. With $g_i = f_{i-2}\lambda_i^2 - f_{i-1}\delta_i^2 + f_i(\lambda_i + \delta_i)$, we get

$$\lambda = \lambda_{i+1} = \frac{-2f_i\delta_i}{g_i \pm \sqrt{g_i^2 - 4f_i\delta_i\lambda_i[f_{i-2}\lambda_i - f_{i-1}\delta_i + f_i]}} \qquad (C.1)$$

Since $\lambda = h/h_i = (x - x_i)/(x_i - x_{i-1})$, a small value of x will give a value of x close to x_i. For this reason the denominator is made as large as possible by choosing the sign accordingly. Hence the result of the extrapolation is

$$x_{i+1} = x_i + \lambda_{i+1}h_i = x_i + h_{i+1}$$

This process is conveniently started by making $x_0 = -1$, $x_1 = 1$, and $x_2 = 0$, and further by using $a_n - a_{n-1} + a_{n+2}$ for f_0, $a_n + a_{n-1} + a_{n-2}$ for f_1, a_n for f_2, that is, $\lambda_2 = -1/2$ and $h_2 = -1$. This corresponds to the approximation $f \simeq a_n + a_{n-1}x + a_{n-2}x$ close to the origin.

A final value of the root x_i is taken when $|x_i - x_i - 1|/|x_i|$ becomes less than some preassigned number. Such a convergence is consistent with the use of floating point arithmetic in the calculation. As a result of this criterion we see that convergence occurs if $x_{i-1} = x_i$. This means that before convergence, no two iterative results will be equal. Furthermore, if $x_i = x_{i-2}$, we have $\delta_i = (x_i - x_{i-2})/(x_{i-1} - x_{i-2}) = 0$, so $\lambda_{i+1} = 0$ and $x_{i+1} = x_i$ also giving convergence unless $x_i = 0$. Thus in normal operation of the process, x_i, x_{i-1}, and x_{i-2} are distinct.

As each root is found, the function f(x) may be divided by it, thus reducing the degree of the equation by one. The algorithm for this reduction is the commonly used one,

$$a_i' = ra_{i-1} + a_i \qquad (i = 0, 1, 2, \ldots)$$

where a_i' is the new coefficient to replace a_i and r is the root which has just been found. Errors introduced by this process will be reduced if the roots are eliminated in order of increasing magnitude. By always starting at the point x = 0, one will tend to find roots roughly in this order.

Muller has proved that there is little to be gained in speed of convergence by fitting a curve of degree higher than two. If the degree of the equation is one, Equation (C.1) is greatly simplified. However, it suffers from a disadvantage if all the coefficients of the original system are real. If one starts from a real point x, then all successive iterative results x will also be real and hence only real roots will be found.

APPENDIX D

SUBROUTINE LISTINGS FOR EXAMPLE 1

```
      SUBROUTINE SMCOEF (A,B,ICOL,NEQN,NROW,NCOL,S,N,R,DISABR)
C
      IMPLICIT COMPLEX (A-H,O-Z)
C
      DIMENSION GLOBAL(1000)
      COMMON /FRQALL/ GLOBAL,NR,NW,ISTART,METHOD
      REAL GLOBAL
      INTEGER NR,NW,ISTART,METHOD
      DIMENSION P(20)
C
      EQUIVALENCE  (GLOBAL(1),P(1))
C
      REAL P
C
      COMPLEX G1,G2,G3,G4,G5,G6
C
      COMMON /SMFALL/ G1,G2,G3,G4,G5,G6
C
C     NO P EQUIVALENCES FOR SM SECONDARY MIRROR MODEL.
C
      COMPLEX A(NROW,NCOL),B(NROW),R(NROW)
C
      INTEGER ICOL(NROW,NCOL),DISABR
      ONE=(1.0,0.0)
C
      A(1,1)=-G2
      A(1,2)=ONE
      A(1,3)=G2
      A(1,4)=(0.0,0.0)
      A(2,1)=(0.0,0.0)
      A(2,2)=-G3
      A(2,3)=ONE
      A(2,4)=(0.0,0.0)
      A(3,1)=(0.0,0.0)
      A(3,2)=(0.0,0.0)
      A(3,3)=-G4
      A(3,4)=ONE
      A(4,1)=1.0/G1
      A(4,2)=G5
      A(4,3)=G6
      A(4,4)=ONE
      B(1)=P(1)
      ICOL(1,1)=1
      ICOL(1,2)=2
      ICOL(1,3)=3
      ICOL(1,4)=4
      ICOL(2,1)=1
      ICOL(2,2)=2
      ICOL(2,3)=3
      ICOL(2,4)=4
      ICOL(3,1)=1
      ICOL(3,2)=2
      ICOL(3,3)=3
      ICOL(3,4)=4
      ICOL(4,1)=1
      ICOL(4,2)=2
      ICOL(4,3)=3
      ICOL(4,4)=4
C
      RETURN
      END
      SUBROUTINE SMCON (S,N)
C
      IMPLICIT COMPLEX (A-H,O-Z)
      DIMENSION GLOBAL(1000)
      COMMON AL(3)
      COMMON /FRQALL/ GLOBAL,NR,NW,ISTART,METHOD
      REAL GLOBAL
      INTEGER NR,NW,ISTART,METHOD
C
      DIMENSION P(20)
C
      EQUIVALENCE  (GLOBAL(1),P(1))
C
      REAL P
C
      COMPLEX G1,G2,G3,G4,G5,G6,AL
C
      COMMON /SMFALL/ G1,G2,G3,G4,G5,G6
C
C     NO P EQUIVALENCES FOR SM SECONDARY MIRROR MODEL.
C
C     DEFINE FUNCTIONS
C
      FIRST(Z)=ONE+S/Z
      QUAD(Z,WN)=(S/WN)**2+2.0*Z*S/WN+ONE
      AL(1)=(-1.0,0.0)
      AL(2)=(-2.0,0.0)
      AL(3)=(0.0,0.0)
      P(1)=0.0
      P(2)=0.0
      P(3)=297.0
      P(4)=5.0
      P(5)=45.454
      P(6)=220.26
      P(7)=3891.0
      P(8)=.4E-2
      P(9)=100.0
      P(10)=0.75
      P(11)=12500.0
      P(12)=0.15125
      P(13)=12500.0
      P(14)=0.0
      P(15)=89.443
      P(16)=0.22
      P(17)=2.60
      P(18)=0.8982
      P(19)=8164.9
C
      ONE=(1.0,0.0)
C
      G1=P(20)
      G2=2.0/(S+1.0)
      G3=1.0/(S+2.0)
      G4=1.0/S
      G5=0.25
      G6=0.25
C
      RETURN
      END
```

APPENDIX E

SUBROUTINE LISTINGS FOR EXAMPLE 2

```
      SUBROUTINE SMCOEF (A,B,ICOL,NEQN,NROW,NCOL,S,N,R,DISABR)
C
      IMPLICIT COMPLEX (A-H,O-Z)
C
      DIMENSION GLOBAL(1000)
      COMMON /FRQALL/ GLOBAL,NR,NW,ISTART,METHOD
      REAL GLOBAL
      INTEGER NR,NW,ISTART,METHOD
      DIMENSION P(20)
C
      EQUIVALENCE (GLOBAL(1),P(1))
C
      REAL P
C
      COMPLEX G1,G2 G3,G4,G5,G6
C
      COMMON /SMFALL/ G1,G2,G3,G4,G5,G6
C
C     NO P EQUIVALENCES FOR SM SECONDARY MIRROR MODEL
C
      COMPLEX A(NROW,NCOL),B(NROW),R(NROW)
C
      INTEGER ICOL(NROW,NCOL),DISABR
      ONE=(1.0,0.0)
C
      A(1,1)=ONE
      A(1,2)=G5
      A(1,3)=G6
      A(1,4)=ONE
      A(2,1)=-G1*G2
      A(2,2)=ONE
      A(2,3)=(0.0,0.0)
      A(2,4)=(0.0,0.0)
      A(3,1)=(0.0,0.0)
      A(3,2)=-G3
      A(3,3)=ONE
      A(3,4)=(0.0,0.0)
      A(4,1)=(0.0,0.0)
      A(4,2)=(0.0,0.0)
      A(4,3)=-G4
      A(4,4)=ONE
      B(1)=P(1)
      ICOL(1,1)=1
      ICOL(1,2)=2
      ICOL(1,3)=3
      ICOL(1,4)=4
      ICOL(2,1)=1
      ICOL(2,2)=2
      ICOL(2,3)=3
      ICOL(2,4)=4
      ICOL(3,1)=1
      ICOL(3,2)=2
      ICOL(3,3)=3
      ICOL(3,4)=4
      ICOL(4,1)=1
      ICOL(4,2)=2
      ICOL(4,3)=3
      ICOL(4,4)=4
C
      RETURN
      END
      SUBROUTINE SMCON (S,N)
C
      IMPLICIT COMPLEX (A-H,O-Z)
      DIMENSION GLOBAL(1000)
      COMMON AL(3)
      COMMON /FRQALL/ GLOBAL,NR,NW,ISTART,METHOD
      REAL GLOBAL
      INTEGER NR,NW,ISTART,METHOD
C
      DIMENSION P(20)
C
      EQUIVALENCE (GLOBAL(1),P(1))
C
      REAL P
C
      COMPLEX G1,G2,G3,G4,G5,G6,AL
C
      COMMON /SMFALL/ G1,G2,G3,G4,G5,G6
C
C     NO P EQUIVALENCES FOR SM SECONDARY MIRROR MODEL.
C
C     DEFINE FUNCTIONS
C
      FIRST(Z)=ONE+S/Z
      QUAD(Z,WN)=(S/WN)**2+2.0*Z*S/WN+ONE
      AL(1)=(-8.0,0.0)
      AL(2)=-P(20)
      AL(3)=(0.0,0.0)
      P(1)=0.0
      P(2)=0.0
      P(3)=297.0
      P(4)=5.0
      P(5)=45.454
      P(6)=220.26
      P(7)=3891.0
      P(8)=.4E-2
      P(9)=100.0
      P(10)=0.75
      P(11)=12500.0
      P(12)=0.15125
      P(13)=12500.0
      P(14)=0.0
      P(15)=89.443
      P(16)=0.22
      P(17)=2.60
      P(18)=0.8982
      P(19)=8164.9
C
      ONE=(1.0,0.0)
C
      G1=12.0
      G2=5.0/(S+8.0)
      G3=2.0/(S+P(20))
      G4=1.0/S
      G5=3.0/20.0
      G6=17.0/60.0
C
      RETURN
      END
```

APPENDIX F

SUBROUTINE LISTINGS FOR EXAMPLE 3

```
      SUBROUTINE AAECOEF(A,B,ICOL,NEQN,NROW,NCOL,S,N,R)
C
C     THIS SUBROUTINE CODED FOR COUPLED
C     TWO CHANNEL ELECTRICAL AA
C     SYSTEM FOR CYCLE III.
C
      INTEGER ICOL(NROW,NCOL)
C
      COMPLEX A(NROW,NCOL),B(NROW),R(NROW),S
      DIMENSION GLOBAL(1000)
      COMMON /FRQALL /GLOBAL,NR,NW,ISTART,METHOD
      REAL GLOBAL
      INTEGER NR,NW,ISTART,METHOD
C
      DIMENSION P(40)
C
      EQUIVALENCE (GLOBAL(1),P(1))
C
      REAL P
C
      COMMON/AAEFALL/AK,EK,CA,CE,XKA,XKE,AZMOT,ELMOT,AJ,EJ,
     1 AOPTG,EOPTG,XG1,XG2,LPF,AAACMP,AAECMP,APAP,EPAP,AFBLP,EFBLP,
     2 XINTNT,EEQVBP,AEQVBP
C
      COMPLEX GI,LPF,AAACMP,AAECMP,APAP,EPAP,AFBLP,EFBLP,
     1 AEQVBP,EEQVBP,XINTNT
C
C
C     FORCING FUNCTION FOR CKOUT
C
      A(1,1)=1.                      $ ICOL(1,1)=1
      A(1,2)=AEQVBP*XG1*AOPTG        $ ICOL(1,2)=8
      A(1,3)=AEQVBP*XG2*EOPTG        $ ICOL(1,3)=16
C
      A(2,1)=-XINTNT*AAACMP*LPF      $ ICOL(2,1)=1
      A(2,2)=1.                      $ ICOL(2,2)=2
      A(2,3)=-XKA*AFBLP/S            $ ICOL(2,3)=6
C
      A(3,1)=-AZMOT*APAP             $ ICOL(3,1)=2
      A(3,2)=1.                      $ ICOL(3,2)=3
      A(3,3)=AK/S+CA                 $ ICOL(3,3)=6
C
      A(4,1)=-(1./AJ)                $ ICOL(4,1)=3
      A(4,2)=1.                      $ ICOL(4,2)=4
C
      A(5,1)=-2./S**2                $ ICOL(5,1)=4
      A(5,2)=1.                      $ ICOL(5,2)=5
C
      A(6,1)=-1.0/S                  $ ICOL(6,1)=4
      A(6,2)=1.                      $ ICOL(6,2)=6
```

```
C
      A(7,1)=-2.                    $ ICOL(7,1)=5
      A(7,2)=1.                     $ ICOL(7,2)=7
C
      A(8,1)=-XG1                   $ ICOL(8,1)=7
      A(8,2)=1.                     $ ICOL(8,2)=8
      A(8,3)=XG2                    $ ICOL(8,3)=15
C
      A(9,1)=-XG2*AOPTG*EEQVBP      $ ICOL(9,1)=8
      A(9,2)=1.                     $ ICOL(9,2)=9
      A(9,3)=XG1*EOPTG*EEQVBP       $ ICOL(9,3)=16
C
      A(10,1)=-XINTNT*AAECMP*LPF    $ ICOL(10,1)=9
      A(10,2)=1.                    $ ICOL(10,2)=10
      A(10,3)=-XKE*EFBLP/S          $ ICOL(10,3)=14
C
      A(11,1)=-ELMOT*EPAP           $ ICOL(11,1)=10
      A(11,2)=1.                    $ ICOL(11,2)=11
      A(11,3)=EK/S+CE               $ ICOL(11,3)=14
C
      A(12,1)=-(1./EJ)              $ ICOL(12,1)=11
      A(12,2)=1.                    $ ICOL(12,2)=12
C
      A(13,1)=-1.0/S**2             $ ICOL(13,1)=12
      A(13,2)=1.                    $ ICOL(13,2)=13
C
      A(14,1)=-1.0/S                $ ICOL(14,1)=12
      A(14,2)=1.                    $ ICOL(14,2)=14
C
      A(15,1)=-2.                   $ ICOL(15,1)=13
      A(15,2)=1.                    $ ICOL(15,2)=15
C
      A(16,1)=-XG2                  $ ICOL(16,1)=7
      A(16,2)=-XG1                  $ ICOL(16,2)=15
      A(16,3)=1.                    $ ICOL(16,3)=16
C
      RETURN
      END
      SUBROUTINE AAECON (S,N)
C
C     THIS ROUTINE CODED FOR COUPLED ELECTRICAL CYCLE III AA
C
      COMPLEX S
      DIMENSION GLOBAL(1000)
      COMMON /FRQALL/ GLOBAL,NR,NW,ISTART,METHOD
      COMMON AL(32)
      REAL GLOBAL
      INTEGER NR,NW,ISTART,METHOD
C
      DIMENSION P(40)
```

```
C
      EQUIVALENCE (GLOBAL(1),P(1))
C
      REAL P
C
      COMMON/AAEFALL/AK,EK,CA,CE,XKA,XKE,AZMOT,ELMOT,AJ,EJ,
     1 AOPTG,EOPTG,XG1,XG2,LPF,AAACMP,AAECMP,APAP,EPAP,AFBLP,EFBLP,
     2 XINTNT,EEQVBP,AEQVBP
C
      COMPLEX GI,LPF,AAACMP,AAECMP,APAP,EPAP,AFBLP,EFBLP,
     1 AEQVBP,EEQVBP,XINTNT,AL,Q(3)
C
      Q(1)=(-9400.0,0.0)
      Q(2)=(-3142.0,0.0)
      Q(3)=(-100000.0,0.0)
      AL(1)=(Q(1)+(CSQRT(Q(1)**2-4.0*Q(1)**2)))/2.0
      AL(2)=CONJG(AL(1))
      AL(3)=AL(1)
      AL(4)=AL(2)
      AL(5)=(Q(2)+(CSQRT(Q(2)**2-4.0*Q(2)**2)))/2.0
      AL(6)=CONJG(AL(5))
      AL(7)=AL(5)
      AL(8)=AL(6)
      AL(9)=(-31400.0,0.0)
      AL(10)=AL(9)
      AL(11)=(Q(3)+(CSQRT(Q(3)**2-4.0*Q(3)**2)))/2.0
      AL(12)=CONJG(AL(11))
      AL(13)=AL(11)
      AL(14)=AL(12)
      AL(15)=(0.0,0.0)
      AL(16)=AL(15)
      AL(17)=AL(16)
      AL(18)=AL(17)
      AL(19)=AL(18)
      AL(20)=AL(19)
      AL(21)=(-31400.0,0.0)
      AL(22)=AL(21)
      AL(23)=AL(22)
      AL(24)=AL(23)
      AL(25)=AL(24)
      AL(26)=AL(25)
      AL(27)=AL(26)
      AL(28)=AL(27)
      AL(29)=(-250000.0,0.0)
      AL(30)=(-31400.0,0.0)
      AL(31)=AL(29)
      AL(32)=AL(30)
      P(34)=(0.0,0.0)
      EPSO=P(34)
      P(36)=(0.0,0.0)
      ETAO=P(36)
      AK=P(32)
      EK=873.
      CA=1.87
      CE=1.87
      XKA=173.
      XKE=123.
      AZMOT=3.08
      ELMOT=4.34
      AJ=.1
      EJ=.1
      AOPTG=2300.
      EOPTG=2300.
C
      XG1=COS(ETAO)*COS(EPSO)-SIN(ETAO)*SIN(EPSO)
      XG2=COS(ETAO)*SIN(EPSO)+SIN(ETAO)*COS(EPSO)
C
      LPF=2./(S**2/9400.**2+S/9400.+1.)
C
C     USE LPF AS POINT TO ADJUST LOOP GAIN
C
      LPF=0.77*LPF
      AAACMP=2.*(S**2/31400.**2+1.)*(S/754.+1.)
     1 /(S**2/31400.**2+2.*S/31400. +1.)/(S/31400.+1.)**2
      AAECMP=1.5*AAACMP
      APAP=1.5/(S**2/100000.**2+S/100000.+1.)
      EPAP=APAP
      AFBLP=0.99/(S/31400. + 1.) /(S**2/3142.**2+S/3142.+1.)
      EFBLP=AFBLP
      AEQVBP=0.9/(S/250000.+1.)/(S/31400.+1.)
      EEQVBP=AEQVBP
C
      XINTNT=(S/12.57 + 1.)/(S/12.57)
C
      RETURN
      END
*EOR  END-OF-FILE  1.
```

VITA

Baskaran Sankaran

Candidate for the Degree of

Master of Science

Thesis: A NEW COMPUTER TECHNIQUE FOR ROOT LOCUS ANALYSIS

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Madras, India, May 30, 1955, the son of Mr. and Mrs. R. Sankaran.

Education: Graduated from Hindu High School, Madras, India, in June, 1971; received the Pre-University Certificate from Loyola College, Madras, India, in June, 1972; received the Bachelor of Technology in Mechanical Engineering degree from the Indian Institute of Technology, Madras, India, in June, 1977; completed the requirements for the Master of Science degree at Oklahoma State University in May, 1979.

Professional Experience: Teaching and research assistant, School of Mechanical and Aerospace Engineering, Oklahoma State University, from September, 1977, to December, 1978.