# AUTOMATIC GENERATION AND PLOTS OF ROOT LOCI

BY

KEMDI CHRISTIAN OKAFOR,
B.Sc.(Nig).

THESIS

Presented to the Graduate School of
Ahmadu Bello University, Zaria in partial
fulfillment of the requirements for the
award of the Degree of

MASTER OF SCIENCE
IN
ELECTRICAL ENGINEERING

Ahmadu Bello University, Zaria.

October, 1981

ACKNOWLEDGEMENTS

May I express my sincere thanks to Prof. B.J. Olufeagba for the supervision of this work. His interest and insistence on thoroughness more than anything else saw the work through.

My gratitude also goes to the Head of Department, Dr. B.G. Bajoga for providing us with some part-time job in the 1980/81 session. The job was of immense help both in terms of experience and finance.

I would like to seize this opportunity to thank the following members of staff of the Computer Centre for their cooperation and sometimes help:-

(a) The operations manager, Mr. I. Udofia.

(b) Programmers: Mr. W.N. Ekemezie, Mr. E.E. Okoro and Mr. Fulkerson.

(c) All the operators and especially Mr. Joe Ezeaputa and Mr. Theo Mayaki.

Finally, I am grateful to my colleagues Benson D. Jelani, Onyedi D. Oyedum and Cyphas Moneke for their cherished company and moral support throughout the course period. The typing services were performed by Mr. Mike Okupe and I thank him for a job well done.

K.C.O.
A.B.U., Oct. 1981.

# D E D I C A T I O N:

To

Ngoo Nwaimo

## ABSTRACT

A program for the automatic computation of root loci points was implemented. The root loci of a variety of systems was plotted using a plotter also developed as part of the thesis work. And a simple compensator - design was effected using the programs.

# TABLE OF CONTENTS


| CHAPTER | | PAGE |
|---|---|---|
| | LIST OF FIGURES ... ... ... | - x |
| | LIST OF TABLES ... ... ... | xiii |
| I | INTRODUCTION ... ... ... | 1 |
| | 1.1 Introduction ... ... ... | 1 |
| | 1.2 Basic ideas in System Design/Analysis ... | 1 |
| | 1.3 The fundamentals of Root Loci ... | 6 |
| | 1.4 Theory and Properties of Root Loci ... | 10 |
| | 1.5 Root Locus Sketching Rules ... | 15 |
| II | A BRIEF SURVEY OF ALGORITHMS FOR AUTOMATIC GENERATION OF ROOT LOCI ... | 20 |
| | 2.1 Introduction ... | 20 |
| | 2.2 Typical Example of Direct Method Algorithms ... | 23 |
| | 2.2.1 The Recursion Formula ... | 23 |
| | 2.2.2 The Break-away/Break-In Points ... | 28 |
| | 2.3 An Example of a Root Following Algorithm. ... | 32 |
| III | THE ROOT LOCUS GENERATING PROGRAM, ROOT98 ... | 37 |
| | 3.1 Introduction ... | 37 |
| | 3.2 A Bisection Procedure for the Angle Loci ... | 38 |
| | 3.3 The Newton-Raphson Iteration Techniques for Angle Loci ... | 45 |
| | 3.3.1 The Locus Slope Angle ... | 45 |
| | 3.3.2 Quality Check for Computed Points ... | 48 |

| | | Page |
|---|---|---|
| 3.3.3 | Determination of Break-away/Break-in Points ... ... ... | 51 |
| 3.3.3.1 | Branch - Points ... ... | 51 |
| 3.3.3.2 | Determination of Branch-points Based on the Characteristics of the Derivatives of the Angle Function ... | 52 |
| 3.3.3.3 | Determination of Bran-points Based on Properties of the Complex Gain ... | 53 |
| 3.3.4 | Determination of Take-Off Angle ... | 56 |
| 3.3.5 | Specialized Features of Root98 ... | 57 |
| 3.3.6 | Algorithm for Root98 ... ... | 58 |
| 3.4 | Organization and Documentation of Root98 | 64 |
| IV | RESULTS OF TESTS FOR ROOT98 ... | 67 |
| 4.1 | Introduction ... ... | 67 |
| 4.2 | Miscellaneous Examples ... ... | 70 |
| 4.3 | Dependence of Root Loci Shape on Damping Ratio ... ... | 82 |
| 4.4 | Unfactored Open Loop Transfer Function ... ... | 91 |

| | | | Page |
|---|---|---|---|
| V | APPLICATION OF ROOT98 TO SERIES COMPENSATION ... | | 97 |
| | 5.1 | Introduction ... ... ... | 97 |
| | 5.2 | Series Compensator Design Using Root Locus Method ... ... ... | 101 |
| | 5.2.1 | Design Principles ... ... ... | 101 |
| | 5.2.2 | Compensator Transfer Function in Bode Form ... ... ... | 105 |
| | 5.2.3 | Compensator Transfer Function in Root Locus Form ... ... ... | 117 |
| VI | CONCLUSION AND RECOMMENDATION FOR FURTHER WORK | | 128 |
| | 6.1 | Introduction ... ... ... | 128 |
| | 6.2 | Conclusion ... ... ... | 128 |
| | 6.3 | Suggestions for Further Work ... | 130 |
| APPENDIX A | | ... ... ... | 133 |
| | Flow Chart for Routine Root98 ... ... | | 134 |
| | Flow Chart for Routine Angfunc ... ... | | 137 |
| | Flow Chart for Routine Locsang and Displa ... | | 138 |
| | Flow Chart for Routine Breakpt ... ... | | 139 |
| | Flow Chart for Routine Plotz ... ... | | 140 |
| | Flow Chart for Routine Order ... ... | | 142 |
| | Flow Chart for Routines Grad and Dgain ... | | 143 |
| | Flow Chart for Routines Deltita and Gain ... | | 144 |
| | Flow Chart for Routine Direct ... ... | | 145 |
| | Flow Chart for Routine Takeang ... ... | | 147 |

Page

APPENDIX B ... ... ... 148

List of Computed points for System 4 ... 149

List of Computed points for Systems 91 and 92 ... 150

BIBLIOGRAPHY ... ... ... 151

# L I S T  O F  F I G U R E S


| FIGURE | | PAGE |
|---|---|---|
| 1.1 | Illustration of basic ideas of system design/analysis ... ... ... | 2 |
| 1.2 | Typical Feedback control system ... | 2 |
| 1.3 | Graphical construction of Root Loci | 14 |
| 3.1 | Angle Iteration Using bisection Method | 42 |
| 3.2 | Newton-Raphson Iteration ... ... | 42 |
| 3.3 | Illustration of Distance Check ... | 42 |
| 3.4 | Illustration of Distance-check Failure | 42 |
| 3.5 | General Layout of Root98 ... ... | 65 |
| 4.1 | Root Loci Plot for System 1 ... | 71 |
| 4.2 | Root Loci Plot for System 2 ... | 72 |
| 4.3 | Root Loci Plot for System 3 ... | 73 |
| 4.4 | Root Loci Plot for System 4 ... | 74 |
| 4.5 | Root Loci Plot for System 5 ... | 75 |
| 4.6 | Root Loci Plot for System 6 ... | 76 |
| 4.7 | Root Loci Plot for System 7 ... | 77 |
| 4.8 | Root Loci Plot for System 8 ... | 78 |
| 4.9 | Root Loci Plot for System 10 ... | 79 |
| 4.10 | Root Loci Plot for System 12 ... | 80 |
| 4.11 | Root Loci Plot for System 31 ... | 81 |
| 4.12 a | Diagram for $b/a > 1/\sqrt{3}$ ... | 84 |
| 4.12 b | Diagram for $b/a < 1/\sqrt{3}$ ... | 84 |

| FIGURE | | PAGE |
|---|---|---|
| 4.12 c | Diagram for b/a = $1/\sqrt{3}$ ... | 84 |
| 4.12 d | Relating b and a to damping ratio ... | 84 |
| 4.13 | Root Loci plot for System 91 ... | 86 |
| 4.14 | Root Loci plot for System 92 ... | 87 |
| 4.15 | Root Loci plot for System 11 ... | 88 |
| 4.16 | Root Loci plot for System 9 ... | 89 |
| 4.17 | Comparison plot showing dependance of Root Loci Shape on damping ratio. ... | 90 |
| 4.18 | Root Loci plot for System 15 ... | 95 |
| 5.1 a | Series compensation ... | 98 |
| 5.1 b | Parallel compensation ... | 98 |
| 5.2 a | Phase Lead Circuit ... | 100 |
| 5.2 b | Phase Lag Circuit ... | 100 |
| 5.3 | Reshaping Root Loci ... | 100 |
| 5.4 | Regular Root Loci for the uncompensated system ... ... | 107 |
| 5.5 | Root Loci plot for variable parameter T with a = 0 ... | 109 |
| 5.6 | Root Loci plot for T = 0.472; with a as the variable parameter ... | 112 |
| 5.7 | Root Loci plot for T = 0.193, withs a as the variable parameter ... | 113 |
| 5.8 | Root Loci plot for T = 0.052; with a as the variable parameter ... | 114 |

| FIGURE | | PAGE |
|---|---|---|
| 5.9 | Root contour for the 3 different values of T; with a as the variable parameter... | 115 |
| 5.10 | Root Loci plot for $K_c$ = 0.5; with p as the variable parameter ... | 123 |
| 5.11 | Root Loci plot for $K_c$ = 5; with p as the variable parameter ... | 124 |
| 5.12 | Root Loci plot for $K_c$ = 50; with p as the variable parameter ... | 125 |

# L I S T  O F  T A B L E S


| Table | | Page |
|---|---|---|
| 3.1 | Subprograms and Functions they perform ... | 66 |
| 4.1 | Execution time for some Systems ... ... | 67 |
| 4.2 | List of computed points for System 15 ... | 96 |
| 5.1 | Showing values of T and Roots used ... ... | 111 |
| 5.2 | Showing values of $K_c$ and Roots used.. ... | 121 |
| 5.3 | Realistic component values ... ... | 127 |

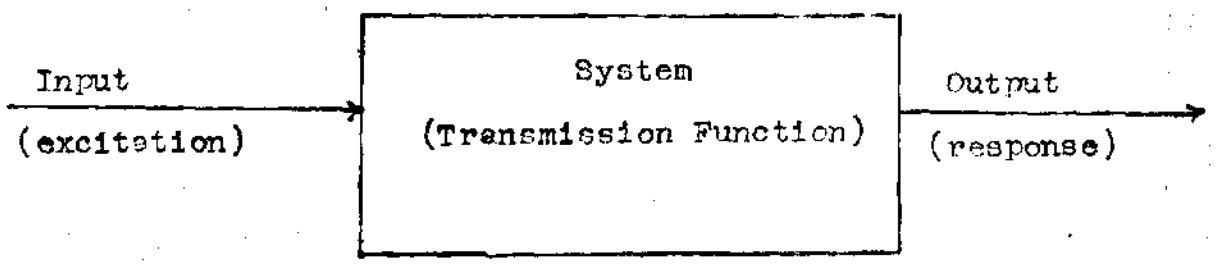



Fig. 1.1: Illustration of basic ideas of system design/analysis.

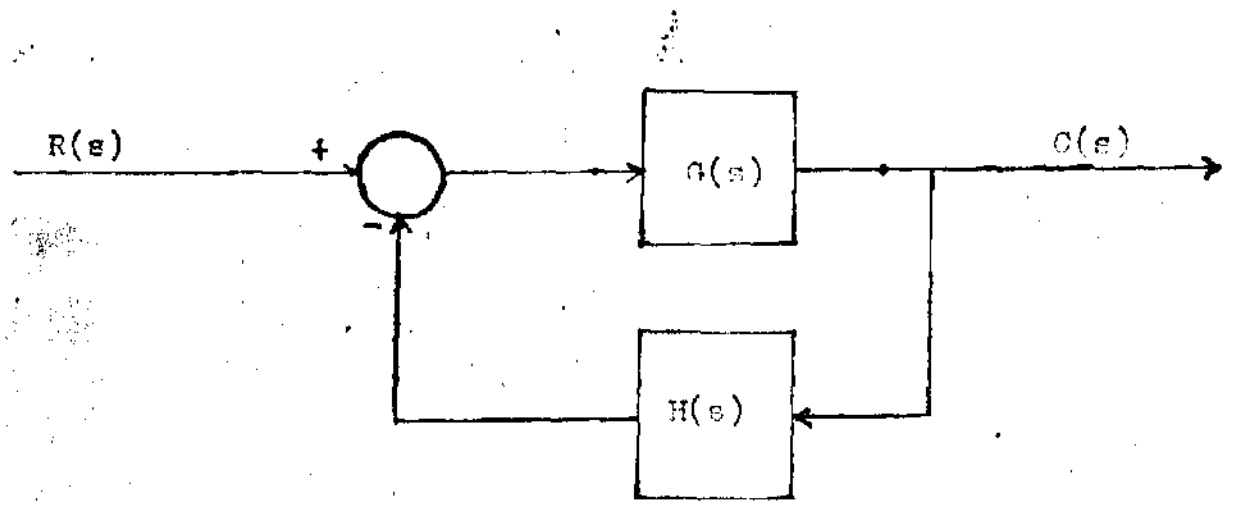



Fig. 1.2: Typical Feedback control system.

On the other hand, if one is interested in synthesizing (ie. building up) the various modules of a system with a view to enable the system perform according to specified criteria, then the corresponding problem is the design one. The problem of design in essence, therefore, is that of synthesizing a transmission function that will invariably yield a desired response to a particular excitation.

Design and analysis could be effected either in the time domain or frequency domain or both. In time domain analysis/design, the excitation signal, the response signal and the transmission functions are all functions of time and one is generally interested in either the transient (initial) or steady-state (final) time response of the system. Whereas for the frequency domain analysis/design, the signals and transmission functions are expressed in terms of frequency and the frequency response as well as other performance criteria (bandwidth, peak response, error coefficients etc.) comprise the major interest.

Working in the time domain, the transmission function or overall transfer function $T(t)$ of a system is defined as the convolution of the input, $u(t)$, with the impulse response, $h(t)$, of the system, that is

$$T(t) = u(t) * h(t) \triangleq \int_{-\infty}^{\infty} u(t-\lambda)\, h(\lambda)\, d\lambda \quad \ldots\ldots \quad 1.1$$

where $\lambda$ is a dummy variable of integration.

It is however much easier to work in the frequency domain because the problem of evaluating the integral of (1.1) reduces to that of multiplication of the fourier transforms since it is possible by transformation of (1.1) to obtain

$$T(w) = u(t) * h(t) = U(w) \times H(w) \quad \ldots\ldots\ldots \quad 1.2$$

where $U(w)$ is defined as the Fourier transform of $u(t)$, $H(w)$ is defined as the Fourier transform of $h(t)$.

This apparently accounts for the fact that the earliest methods of design and analysis attributed to Nyquist and Bode are all frequency domain procedures. Time and experience showed that designs and/or analyses based solely on the frequency domain were sometimes inadequate. This stems from the fact that direct correlation between frequency response and the transient response of a closed loop system[1] is precariously meagre. There was need therefore to evoke methods which could enhance this vital correlation. This was realized by expressing the various signals in terms of the complex frequency variable,

$$s = \sigma + j\omega \quad \ldots\ldots\ldots \quad 1.3$$

and defining the transfer function of a system in the usual manner as

$$T(s) = \frac{C(s)}{R(s)} = \int_0^\infty T(t)\, e^{-st}\, dt \qquad \ldots\ldots\ldots \quad 1.4$$

based on the Laplace transform (see Fig.1.2). In control terms then, we define the analysis problem by

$$C(s) = \frac{G(s)}{1 + G(s)\, H(s)} \cdot R(s) \qquad \ldots\ldots\ldots \quad 1.5$$

and the design problem by

$$T(s) = \frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)\ H(s)} \qquad \ldots\ldots\ldots \quad 1.6$$

where (see Fig. 1.2)

C(s) is defined as the Laplace transform of c(t), the output signal ;

R(s) is defined as the Laplace transform of r(t), the input signal;

G(s) is defined as the forward path transfer function of the system;

H(s) is defined as the feedback or reverse path transfer function of the system;

$G(s)H(s) = GH(s)$ is defined as the open loop transfer function of the system and

$T(s)$ defined as in (1.6) is the closed loop transfer function of the system.

The equation,

$$1 + GH(s) = 0 \qquad \ldots\ldots\ldots \quad 1.7$$

which, for most linear systems, is a polynomial in $s$ is referred to as the characteristic equation (CE) of the system. It performs the role of specifying the nature of the characteristic roots or the eigenvalues of the overall system and hence the response to the inputs.

## 1.3 The Fundamentals of Root Loci

The root locus technique is a graphical procedure[2] for determining the roots (zeros) of the CE of a system and the s-plane loci traversed by the roots as one of the system parameters is varied. It is a very useful technique in the design and analysis of control systems. The method was introduced by W.R. Evans in 1948[1], and can be used to determine the roots of any polynomial[2] provided proper partitioning and rearrangement of the polynomial is organized.

Consider the C E

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

This can often, for linear time invariant systems be expressed in the form,

$$\sum_{i=0}^{n} a_i s^i = 0 \qquad \ldots\ldots\ldots 1.9$$

For $n = 3$, this results in

$$a_3 s^3 + a_2 s^2 + a_1 s + a_0 = 0$$

which can be put in the form

$$s^3 + c s^2 + b s + d = 0 \qquad \ldots\ldots\ldots 1.10$$

The roots of this equation are those values of $s$ for which (1.10) is satisfied. If $c$ and $b$ are fixed, then for each value of $a$, $\lrcorner$ three values of $s$ that are roots. Consequently, if $a$ assumes values in the range $0 \leqslant a \leqslant \infty$, $\lrcorner$ an infinity of triplets of $s$-values which are corresponding roots. If these values are plotted in the $s$-plane, they form a curve or "locus" called the root locus[2] for the polynomial of (1.10) when $a$ is the variable parameter.

<u>Definition:</u> A root locus is therefore defined as the set of all $s$-values which satisfy the CE of a given system.

It should be noted that the root locus gain, K , is not necessarily, the same as the static error coefficient, $K_x$ which, specifically, is the gain of the open loop transfer function, GH(s), of a control system when it is expressed in the Bode frequency form[2] viz:

$$GH(s) = \frac{K_x \prod_{i=1}^{m} (s/z_i + 1)}{s^N \prod_{i=1}^{k} (s/p_k + 1)} \qquad \text{------ 1.14a}$$

where N is the multiplicity of the pole at the origin of the complex frequency-plane.

The two gain constants are related via the equation

$$K_x = \frac{K \prod_{i=1}^{m} z_i}{\prod_{i=1}^{k} p_i} \qquad \text{------ 1.14b}$$

This vital, relationship should be taken into consideration in the design of series compensators using the root locus method with a view to either providing a plant with an appropriate root locus gain or a desired steady-state accuracy. The application of the root locus technique in the design of series compensators is treated in Chapter V.

If a regular root locus is plotted with $K < 0$, the resultant locus is called the inverse root locus of the system. Points on the root locus of a system can be determined either graphically or by computation based on various numerical and analytic schemes.

In this thesis, methods for the automatic generation of root loci will be considered. First a survey of previous algorithms described for the automatic generation of root loci is presented in chapter II. The organization and documentation of the program, Root 98, developed in this thesis is presented in chapter III. In chapter IV, the root loci plots for the systems used in testing-out Root 98 are illustrated while in chapter five, Root 98 is used to implement the design of compensators for some given systems. In the remaining sections of this chapter, the theory and properties of root loci and the simple rules employed in the graphical construction of root loci is discussed.

## 1.4 Theory and Properties of Root Loci

It has been shown earlier on that the CE of the typical system (fig. 1.2) can be put in the form,

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

which implies that

$$KF(s) = -1 \qquad \ldots\ldots \qquad 1.14c$$

In polar form (1.14) can be written as

$$\left|K\,F(s)\right| e^{\,j\,\mathrm{Arg}\left\{KF(s)\right\}} = 1 \cdot e^{j(2n+1)\pi} \quad \ldots\ldots 1.15$$

Hence two equations can be derived from (1.15) viz

$$\mathrm{Arg}\left\{F(s)\right\} = (2n+1)\pi \quad \ldots\ldots 1.16$$

$$n = 0, \pm 1, \pm 2 \ldots\ldots$$

and

$$\left|F(s)\right| = \frac{1}{|K|} \quad \ldots\ldots\ldots 1.17$$

Equation (1.16) is called the angle criterion[13] and arises from the fact that the argument of $F(s)$ must be an odd multiple of $\pi$ for $s$ to be a root locus point while equation (1.17) is called the magnitude criterion and represents the fact that the magnitude of the open loop transfer function

$$GH(s) = KF(s)$$

must be unity for $s$ to be a root locus point. Hence any point, $s_1$ (say), which satisfies equation (1.16) is a root locus point. If $s_1$ further satisfies equation (1.17), then $s_1$ is a characteristic root of the system. Similarly a point $s_2$ (say) which satisfies (1.17) is a root locus point and if $s_2$ further satisfies (1.16), it is also a characteristic root of the system.

For the inverse root locus plot the root locus gain $K < 0$, hence the argument must be an even multiple of $\pi$ and the angle criterion becomes

$$\text{Arg}\left\{F(s)\right\} = 2n\pi \qquad \ldots\ldots \qquad 1.18$$

$$n = 0, \pm 1, \pm 2, \ldots\ldots$$

Subsequently, a regular root locus is sometimes called the $180^{\circ}$ root locus while the inverse plot is called the $0^{\circ}$ root locus.

The foregoing purports that it is possible to determine both magnitude and angle loci for any given system.[2] If this is done and the loci are drawn, the roots of the CE i.e. the characteristic roots of the given system are the points in the s-plane (complex frequency - plane) at which the argument and magnitude loci intersect. If the CE is a polynomial of degree, n (say), there will be n roots of the CE corresponding to the n points of intersection of the loci. However, in its original form, the root locus method is based on the angle relation of (1.16). This choice was made by Evans when he first presented the method and used as a basis for the design of a template known as the **spirule**. The spirule operates by summing up algebraically, the angles contributed by the various open loop poles and zeros. Generally poles contribute negative angles while zeros contribute positive

If $\theta = (2n + 1)\pi$

$n = 0, \pm 1, \pm 2, \ldots\ldots$

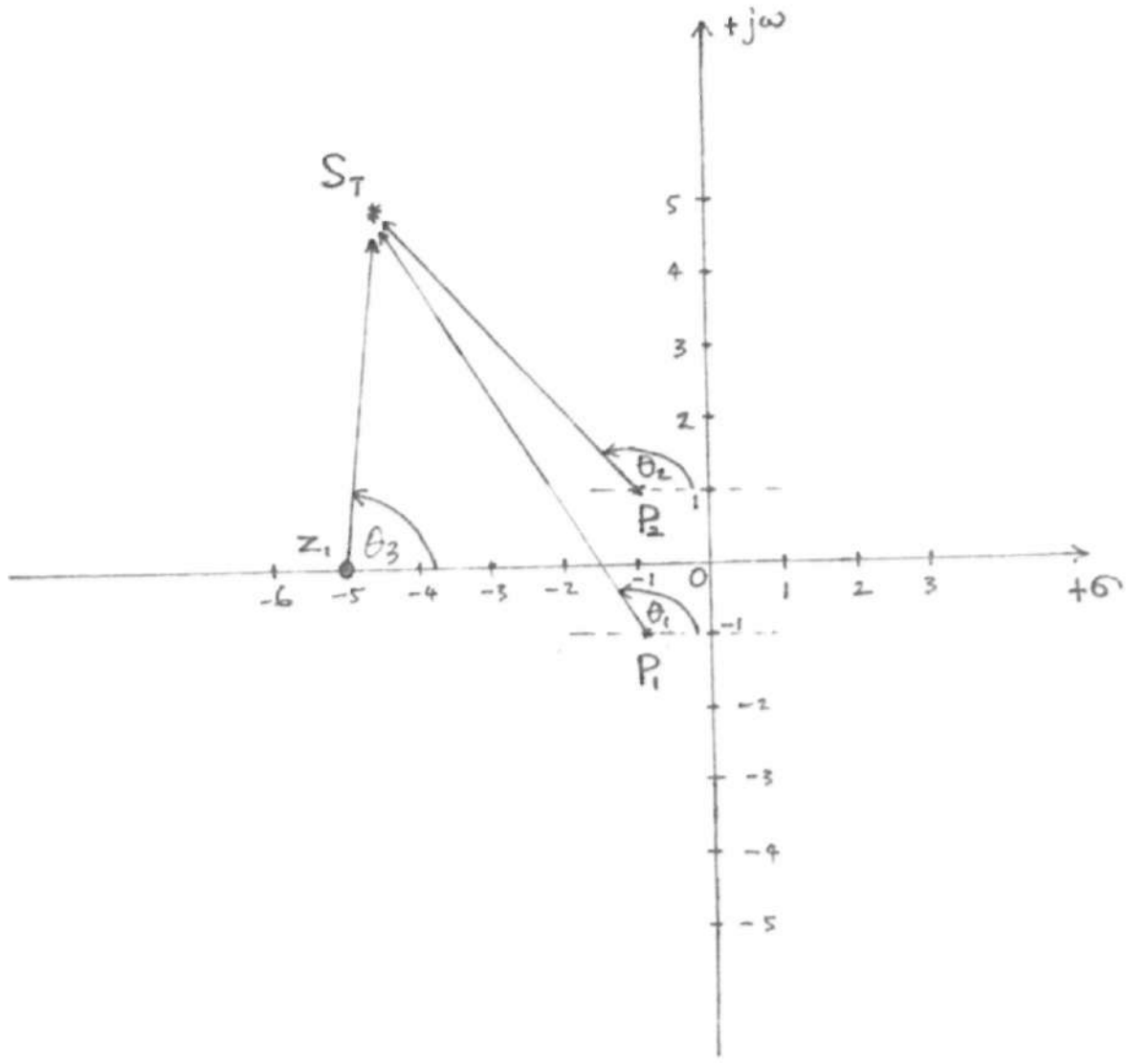



Fig. 1.3: Graphical construction of root loci.

then the point $s_T$ is a root locus point of the given system. Otherwise, a new test point is chosen and tested as previously described. It is apparent that such an exercise can be rather tedius. Despite the fact that the evaluation and subsequent summation of these angles is simplified by the spirule, the process becomes extremely complicated and intractable if the system being handled has a large number of singularities.

It should be noticed that equation (1.19) defines a class of systems and not just a system. This class of system is defined by the set D of root locus gain values where

$$D = \left\{ K: \ 0 \leq K \leq \infty \right\}, \qquad \text{.....1.22}$$

for negative feedback systems.

Fixing the value of K is equivalent to selecting a single system out of the lot and the value of $S_T$ which now satisfies the angle function is a characteristic root of the chosen system.

## 1.5 Root Locus Sketching Rules

In most cases, it is not necessary to obtain the exact root loci of a system; what is required under such circumstances is a sketch of the loci. Certain rules have been derived based on experience and/or properties of root loci in general.

These aid the sketching of root loci and, more important still, help in the formulation of algorithms for the automatic generation of root loci.

Recall, the CE of a system can be written as

$$\frac{K\prod_{i=1}^{m}(s - z_i)}{\prod_{i=1}^{n}(s - p_i)} = -1 \qquad \ldots\ldots\ldots \quad 1.23$$

where K is the root locus gain

m is the number of finite open loop zeros

n is the number of finite open loop poles

$z_i$ $i = 1, \ldots m$ is the set of finite open loop zeros

$p_i$ $i = 1, \ldots n$ is the set of finite open loop poles.

Given this background, the root locus sketching rules can be stated viz:

Rule 1. The total number of root loci branch equals n, the number of finite open loop poles. This pre-supposes that $n \geqslant m$ which is the case for minimum phase systems. If $n < m$, then the total number of root loci equals m.

Rule 2: If $n \geqslant m$, n branches of the root loci of a system start at the n finite open loop poles at $K = 0$; m of these branches terminate at the m finite open loop zeros at

$K = \infty$, while the remaining $n - m$ branches approach infinity as $K \rightarrow \infty$. On the other hand, if $n < m$, $m - n$ of the branches will start at infinity; the rest $n$ branches will start at the $n$ finite open loop poles and all the branches of course, will terminate at the $m$ finite open loop zeros.

Rule 3. Segements of the root loci lie on the real-axis if there exists an odd number of singularities of the system to the right of a given test point.

Rule 4. Branches of the root loci which proceed to infinity or receed from infinity do so assymptotic to a set of straight lines inclined to the positive real-axis at angles given by

$$\alpha_L = \frac{(2L + 1)\pi}{|n - m|} \quad \ldots\ldots \quad 1.24$$

where $L$ is a set defined by

$$\left\{L \text{ is integer: } 0 \leq L < |n - m|\right\}$$

Rule 5. The complex portions of the root loci always occur in pairs provided that the CE is a rational function with real coefficients.(8) This implies that the root loci is symmetric about the real-axis.

Rule 6. The angle of emergence of a root locus from a complex open loop pole is given by:

$$\theta_p = \sum_{i=1}^{m} \arg\{s_k - z_i\} - \sum_{\substack{i=1 \\ i \neq k}}^{n} \arg\{s_k - p_i\} + \pi \qquad \ldots\ldots 1.25$$

while the angle of emergence from a complex zero is given by

$$\theta_z = \sum_{\substack{i=1 \\ i \neq k}}^{m} \arg\{s_k - z_i\} + \sum_{i=1}^{n} \arg\{s_k - p_i\} - \pi \qquad \ldots\ldots 1.26$$

where $s_k$ is a complex singular point.

Rule 7. If segments of a root locus lie between two poles or two zeros, there exists a break-away or a break-in point, $s_b$, at which the two opposite loci separate or meet. The point $s_b$ can be determined by making use of the fact that the complex gain function $K(s)$ has a turning point at $s_b$ such that,

$$\left.\frac{d\,K(s)}{ds}\right|_{s = s_b} = 0 \qquad \ldots\ldots 1.27$$

From the CE, it can be shown that the derivative of the complex gain is

$$\frac{dK(s)}{ds} = K \sum_{i=1}^{n} \frac{1}{s - p_i} - \sum_{i=1}^{n} \frac{1}{s - z_i} \qquad \ldots\ldots 1.28$$

Rule 8. The point at which a root locus crosses the imaginary axis can be evaluated using the Routh - Hurwitz criterion.

# CHAPTER II

## A BRIEF SURVEY OF ALGORITHMS FOR AUTOMATIC GENERATION OF ROOT LOCI

### 2.1 Introduction

Graphical construction of root loci is relatively easy, especially with the aid of the spirule, for systems of order less than four. However, the method possesses a very serious limitation since it requires tremendous effort in handling systems of order higher than four and time delay systems. It also requires considerable effort if any accuracy is to be achieved when performing design. In order to circumvent this problem, various analytic and semi-analytic methods for the construction of root loci have been developed. The resulting methods exploit various properties of the conic sections to realize easier and faster methods for root loci plotting. Power, in (12), described a method for plotting the root loci of systems with a combination of four singularities (poles and zeros) by utilizing a single parabola and a family of hyperbolas. This method similarly faces serious problems since it is equally restricted to low order systems. The advent of fast digital computers has made it possible to compute extensive and complex system root loci automatically.

Numerous algorithms have been developed over the years (from about 1958) for the generation of the root loci of control systems. The basis for any root locus algorithm is the determination of the roots of the CE :

$$1 + K\,F(s) = 0 \qquad \ldots\ldots \quad 2.1$$

As has been indicated earlier on, equation (2.1) generates two conditions namely:

(a) The angle condition

$$\text{Arg}\left\{F(s)\right\} = (2n + 1)\pi \qquad \ldots\ldots \quad 2.2$$

$n = 0, \pm 1; \pm 2, \ldots\ldots$

and

(b) The magnitude condition

$$\left|F(s)\right| = \frac{1}{|K|} \qquad \ldots\ldots \quad 2.3$$

Any efficient root locus algorithm must therefore be a systematic procedure for solving (2.1), (2.2) or (2.3) to an acceptable degree of accuracy and stability. Such a procedure must also provide a workable scheme for determining, precisely, the critical points of the root loci usually referred to as the breakaway and break-in points.

A measure or an index of the performance of any algorithm will have to incorporate the following considerations:

(a) Applicability - Whether algorithm lends itself to easy manipulations and does not have any serious limitations.

(b) Cost - In terms of computer time and memory space required.

(c) Reliability - In terms of the degree of accuracy obtainable and

(d) Adaptability - Whether the algorithm could be modified to handle a wide range of system types.

Presently, almost all the algorithms available can be categorized[11] as follows:-

1. Direct Methods: These include algorithms in which various numerical schemes are used to determine the roots of the polynomial

$$1 + K\,F(s) = 0$$

as $K$ is varied in some prescribed manner. Since these algorithms are limited to polynomials, they cannot handle systems with dead time (i.e. time delay systems).

2. Area Search Algoriths: This class of algorithms generate the root loci by grid-search techniques over a specified area of the complex frequency - plane (s - plane). The grid-search method, in effect, is used to determine the points satisfying the necessary and sufficient conditions for root locus points. Any of equations (2.1), (2.2) and (2.3) could serve as the search criterion.

3. Branch Following Algorithms: Algorithms in this class often presume 'a priori' knowledge of a point on the locus of interest and use the geometric and trigonometric properties of the locus at the given point to predict a new point on the locus.

The algorithm employed in developing the program, Root98, used in this thesis work is a locus following algorithm. It was first presented by R.H. Ash and G.R. Ash in 1968. This algorithm will be discribed in detail in chapter III. But for completeness, two other algorithms will be highlighted here as typical examples of algorithms of categories (1) and (3) respectively.

## 2.2 Typical Example of Direct Method Algorithms

### 2.2.1 The Recursion Formula

In this section, it is intended to represent an algorithm presented in the first instance in 1978 by

C.T. Pan and K.S. Chao.[14] "The approach is based on the concept of continuction methods in which the solution of a parameterized family of algebraic problems is converted into the solution of differential equation".

Recall equations(1.6,1.11 and 1.12),the closed loop system transfer function takes the form

$$T(s) = \frac{B(s)\,G(s)}{B(s) + K\,A(s)} \qquad .. \qquad .. (2.4)$$

Hence the CE of the system becomes

$$g(s,k) \triangleq B(s) + K\,A(s) = 0 \qquad .. \qquad .. (2.5)$$

The root loci of the system represented by this CE consist of a set of points in the s-plane, L(say), for which (2.5) is satisfied i.e.

$$L = \left\{ S : g(s,k) = 0 \right\} \qquad .. \qquad .. (2.6)$$

The simplest direct method of obtaining the set, L, involves the direct determination of the roots of (2.5) for specified gain values. Such a procedure may be attendant with convergence problems and may, worse still, not be capable of handling very large systems since all the roots will be determined each time for every gain value.

However, for the approach described here and presented in (14), s and K are considered functions of a dummy variable, t, and two simultaneous differential equations are formulated , viz:

$$\frac{d}{dt}\, g\left\{s(t),\, K(t)\right\} \;=\; -g\left\{s(t),\, K(t)\right\},\; g\left\{s(o),\, K(o)\right\} = 0 \qquad .. \qquad .. \qquad (2.7)$$

and

$$\frac{d}{dt}\, K(t) = \pm\, 1, \quad K(o) \;=\; K_o \qquad .. \qquad .. \qquad (2.8)$$

where $s(o) = s_o$ is a root of (2.5) for an initial gain $K_o$. Application of chain rule to (2.7) and (2.8) yields

$$\frac{ds}{dt} \;=\; -\left\{g \pm \frac{\partial g}{\partial K}\right\} \Big/ \partial g/\partial s, \quad s(o) = s_o \qquad .. \qquad .. \qquad (2.9)$$

and

$$\frac{dK}{dt} \;=\; \pm\, 1\,, \quad K(o) \;=\; K_o \qquad .. \qquad .. \qquad .. \qquad (2.10)$$

which reduces to

$$\frac{ds}{dt} \;=\; -\;\frac{B(s) + K\,A(s) \;\pm\; A(s)}{B'(s) + K\,A'(s)}\,, \quad s(o) = s_o \qquad .. \qquad (2.11)$$

and

$$\frac{dK}{dt} \;=\; \pm\, 1, \quad K(o) \;=\; K_o \qquad .. \qquad .. \qquad .. \qquad (2.12)$$

The simultaneous equations (2.11) and (2.12) can now be solved by any stable and consistent numerical integration method.

Pan and Chao employed Euler's method and consequently obtained the following recursion equations

$$s_{k+1} = s_k - h \frac{B(s_k) \pm K_k A(s_k) + A(s_k)}{B'(s_k) + K_k A'(s_k)} \quad .. \quad (2.13)$$

and

$$K_{k+1} = K_k \pm h \quad .. \quad .. \quad .. \quad (2.14)$$

where it is obvious from (2.13) and (2.14) that h is the small change, $\triangle K$, in K and is also the step size for the numerical integration.

The $\pm$ sign in (2.13) and (2.14) are used to fix the type of change, $\triangle K$, in the gain constant. If K is to be increased, the + sign is used while the - sign is used if K is to be decreased. This implies that if the computation of locus points is to be commenced at the open loop poles, the + sign is used, and if the starting point is an open loop zero, the - sign is used. For a starting point which is neither an open loop pole nor an open loop zero, either of the signs could be used. The inverse root locus points can be generated using these equations if the - sign is used and the starting point is an open loop pole. It should be noted that algorithm may not work if the +sign is used

when the starting point is an open loop zero. This should be so in view of the fact that if $K_k = \infty$ , then (2.14) gives

$$K_{k+1} = \infty + h = \infty$$

Since $\partial g / \partial s = 0$ at a repeated open loop pole, equation (2.9) is meaningless at such points and as such, a repeated open loop pole cannot be used as a starting point for this algorithm.[14] In order to be able to determine the root loci emanating from a repeated pole, the following modifications are necessary:-

Let the open loop system gain contain a repeated pole, $P_k$ , with multiplicity, $r$ , then the function $g(s, K)$ can be written as

$$g(s, K) = B(s) + K A(s)$$

$$= (s - P_1)(s - P_2) \ldots (s - P_k)^r \ldots (s - P_{n-r}) + K A(s) \quad \text{------} \quad (2.15)$$

If $s$ is now defined by

$$s = P_k + \Delta s = P_k + \delta e^{j\theta} \quad \text{------} \quad (2.16)$$

where $\delta$ is an arbitrarily small real number and $\theta$ is a phase angle to be determined, then

$$g(s,K) = g(P_k + \delta e^{j\theta}, K) \equiv 0 \quad \text{------} \quad (2.17)$$

which implies that

$$\Delta s = \delta e^{j\theta} \simeq (K \rho e^{j\phi})^{\frac{1}{r}} \qquad \text{---------} \qquad (2.18)$$

where

$$\rho e^{j\phi} = - \left. \frac{A(s)\,(s - P_k)^r}{B(s)} \right|_{s = P_k} \qquad \text{-------} \qquad (2.19)$$

The r approximate starting points can then be evaluated from (2.20) as

$$s_i = P_k + \delta e^{\frac{j(\phi + 2\pi i)}{r}} \qquad \text{-------} \qquad (2.20)$$

$$i = 1, 2, \ldots\ldots r - 1$$

and their corresponding gains are

$$K = \frac{1}{\rho} e^r \qquad \text{-------} \qquad (2.21)$$

2.2.2 The Break - away and Break-in Points

An analytic condition for determining the break-away and break-in point, $s_b$, on a root locus is given by the equation

$$\left. \frac{dK}{ds} \right|_{s = s_b} = 0 \qquad \text{-----} \qquad (2.22)$$

$$\left. \frac{dK}{ds} \right|_{s = s_b} = \left. \frac{d}{ds} \left( - \frac{B(s)}{A(s)} \right) \right|_{s = s_b}$$

$$= \frac{A(s_b)\,B'(s_b) - B(s_b)\,A'(s_b)}{A^2(s_b)} = 0 \qquad \text{----- (2.23)}$$

Since A(s) is the numerator of the open loop system transfer function, it cannot be zero for a finite gain, K, and therefore the condition

$$A(s_b)\,B'(s_b) - B(s_b)\,A'(s_b) = A(s_b)\,B'(s_b) - KA'(s_b) = 0 \qquad \text{-----} \quad (2.24)$$

implies that

$$\frac{\partial g}{\partial s}\Big|_{s = s_b} = B'(s_b) + K\,A'(s_b) = 0 \qquad \text{---} \quad (2.25)$$

For such points once more, equation (2.9) does not hold. The usual condition expressed in equation (2.22) cannot therefore be used to determine the break-away/break-in points in this method. Instead the directional derivative, $\frac{dK}{dL}$, as well as its higher derivatives along the tangential direction of the root locus are generally considered. In other words the local extremal values of K along L rather than global extremal values are taken into consideration in other to locate and evaluate all saddle points.

$$\text{Let} \quad K(s) = \frac{B(s)}{A(s)} = U(x,y) + j\,V(x,y) \qquad \ldots.(2.26)$$

where U and V are real valued functions of x and y which are in turn related in the usual manner:

$$s = x + j\,y$$

The directional derivative of K at a point s on the root locus in the unit direction, $v = e^{j\theta} = v_x + j\, v_y$ tangential to the root locus is then

$$\frac{dK}{dL} = \left\langle U(x,y),\ \mathbf{v} \right\rangle = \frac{\partial U}{\partial x} v_x + \frac{\partial U}{\partial y} v_y \qquad \text{---- (2.27)}$$

which can be put in the form

$$\frac{dK}{dL} = \mathrm{Re}\left[\left(\frac{\partial U}{\partial x} - j\,\frac{\partial U}{\partial y}\right)\left(v_x + j\, v_y\right)\right], s \in L \qquad \text{---- (2.28)}$$

By Cauchy - Riemann condition, this reduces to

$$\frac{dK}{dL} = \mathrm{Re}\left[\left(\frac{\partial U}{\partial x} + j\,\frac{\partial V}{\partial x}\right) e^{j\theta}\right] = \mathrm{Re}\left[K'(s)\, e^{j\theta}\right] \qquad \text{---- (2.29)}$$

where

$$K'(s) = \frac{dK}{ds}$$

Generally then, (2.29) can be written as

$$\frac{d^m K}{d L^m} = \mathrm{Re}\left[K^{(m)}(s)\, e^{jm\theta}\right] \qquad \text{---- (2.30)}$$

This directional derivative can then be used to characterize singular points as follows:

Let

$$\left.\frac{d^k K}{ds^k}\right|_{s=s_b} = 0, \quad k = 1 , 2 , \ldots . \quad q - 1 \qquad \text{---- (2.31)}$$

and

$$\left.\frac{d^q K}{ds^q}\right|_{s = s_b} \neq 0, \quad 2 \leq q \leq n \qquad \text{---- (2.32)}$$

then $K(s_b)$ is either a local maximum or a local minimum if $q$ is even. Otherwise $K(s_b)$ is a monotonic function of $s$ in the neighbourhood of $s_b$. The conventional break-away/ break-in points on the real-axis then correspond to the cases for which $q = 2$.

The method relies heavily on the performance of the Euler method for numerical integration. It provides a simple and fast technique because of the explicit nature of the approximation. However since the Euler method suffers from numerical instability, this could lead to very severe restrictions on the allowable step size and may tend to lead to very long computation time. It may be necessary to use improved and more stable procedures such as the predictor/ corrector methods [20,21] with higher order truncation errors.

Finally, because of the fact that the equation

$$g(s, K) = 0$$

may not be satisfied exactly, there could be serious convergence problems since the computed trajectory may diverge away from the actual root locus as a result of error accumulation. This error propagation problem however is taken care of by the ( - ) sign placed in front of the term $g(s,K)$ in equation (2.7).

## 2.3 An Example of a Root Following Algorithm

In this section, a generalized root locus following technique easily adaptable to solving problems in sampled data systems is represented. The technique first published by J.R. Mitchell and W.L. McDaniel (Jr),[13] approximates a locus by a straight line in order to predict another point on the locus.

The CE can now be written as

$$1 \pm g(s,T) = 0 \quad \text{------} \quad (2.33)$$

where $g(s,T)$ is the open loop transfer function and $T$ is the parameter for which root locus is to be plotted. The magnitude and angle criteria now take the forms

$$|g(s,T)| = 1 \quad \text{-----} \quad (2.34)$$

and

$$\text{Arg}\{g(s,T)\} = (2r \cdot 1)\pi \quad \text{-----} \quad (2.35)$$

$n = 0, \pm 1, \pm 2, \ldots\ldots$

respectively

Since the open loop transfer function is a complex quantity, it can be split into its real and imaginary parts viz

$$g(x,y,T) = a(x,y,T) + j\, b(x,y,T) \quad \text{-----} \quad (2.36)$$

where a and b are the real and imaginary parts of g respectively and x and y are related as usual to the complex variable, s viz

$$s = x + jy$$

Let

$$\arg\left\{g(x,y,T)\right\} = \theta \quad \text{------} \quad (2.37)$$

then one can write

$$\theta = \tan^{-1} \frac{b(x,y,T)}{a(x,y,T)}$$

or

$$\tan\theta = F_1(x,y,T) = \frac{b(x,y,T)}{a(x,y,T)} \quad \text{------} \quad (2.38)$$

(2.38) is now the angle condition.

The magnitude condition on the other hand takes the form:

$$|g(x,y,T)| = |a(x,y,T) + j\, b(x,y,T)|$$

i.e.

$$F_2(x,y,T) = a(x,y,T)^2 + b(x,y,T)^2 \quad \text{---} \quad (2.39)$$

Equations (2.38) and (2.39) represent two hypersurfaces which are functions of the three variables x, y and T. The manifold of these hypersurfaces is composed of several lines, namely the 3-dimensional lines of the root loci.(13)

The gradient of the two surfaces are given by

$$\nabla F_1 = \frac{\partial F_1}{\partial x}\hat{i} + \frac{\partial F_1}{\partial y}\hat{j} + \frac{\partial F_1}{\partial T}\hat{k} \quad \text{-----} \quad (2.40)$$

and

$$\nabla F_2 = \frac{\partial F_2}{\partial x}\hat{i} + \frac{\partial F_2}{\partial y}\hat{j} + \frac{\partial F_2}{\partial T}\hat{k} \quad \text{-----} \quad (2.41)$$

Taking their cross products yields

$$\nabla F_1 \times \nabla F_2 = \left(\frac{\partial F_1}{\partial y}\frac{\partial F_2}{\partial T} - \frac{\partial F_2}{\partial y}\frac{\partial F_1}{\partial T}\right)\hat{i} + \left(\frac{\partial F_2}{\partial x}\frac{\partial F_1}{\partial T} - \frac{\partial F_1}{\partial x}\frac{\partial F_2}{\partial T}\right)\hat{j} + \left(\frac{\partial F_1}{\partial x}\frac{\partial F_2}{\partial y} - \frac{\partial F_1}{\partial y}\frac{\partial F_2}{\partial x}\right)\hat{k} \quad \text{----} \quad (2.42)$$

From (2.42) a straight line tangent to the manifold and hence the root locus can be obtained, viz

$$\frac{\Delta x}{\left(\frac{\partial b}{\partial y}\right)\left(\frac{\partial a}{\partial T}\right) - \left(\frac{\partial a}{\partial y}\right)\left(\frac{\partial b}{\partial T}\right)} = \frac{\Delta y}{\left(\frac{\partial a}{\partial x}\right)\left(\frac{\partial b}{\partial T}\right) - \left(\frac{\partial b}{\partial x}\frac{\partial a}{\partial T}\right)}$$

$$= \frac{\Delta T}{\left(\frac{\partial b}{\partial x}\right)\left(\frac{\partial a}{\partial y}\right) - \left(\frac{\partial b}{\partial y}\frac{\partial a}{\partial x}\right)}$$

where $\Delta x = x - x_o$

$\Delta y = y - y_o$

$\Delta T = T - T_o$

and

$$s_o = x_o + j\, y_o$$

is a point on the root locus and $T_o$ is a value of the root locus parameter corresponding to the point $s_o$.

Hence any point $s = x + jy$ in the complex frequency plane (s - plane) on the root locus can be predicted for a given increment, $\Delta T = T - T_o$. For very accurate predictions, $\Delta T$, must be, necessarily, small. In (13), the algorithm was used expressly for sampled data systems and hence no discussion or mention of break-away/break-in point detection procedure was made. However, if the method is to be used for regular root loci (i.e. if the variable parameter, T, is the root locus gain), then the condition

$$\left.\frac{dT}{ds}\right|_{s = s_b} = 0$$

can be used where $s_b$ is the break-away/in point.

The efficiency or strictly, the accuracy of this method will depend on the degree of accuracy achievable in the

evaluation of the partial derivatives of a and b w.r.t. the three variables x, y and T. If this method is to be used in developing a regular root locus program, then a procedure for ensuring that predicted points are acceptably accurate must be introduced. Otherwise, there is the possibility of the occurence of error propagation since it is obvious that the accuracy of a subsequent prediction of a locus point will be affected by the accuracy of a previous locus point prediction.

CHAPTER III

## THE ROOT LOCUS GENERATING PROGRAM, ROOT98

### 3.1 Introduction

The primary objective of this thesis involves the development and implementation, on CDC Cyber 72, of a Computer program for the automatic generation of root loci. The algorithm employed for this purpose, the organization and documentation of the resultant program (Root98) are discussed in this chapter.

Two different algorithms for the automatic computation of root loci have been discussed in chapter II. Either of these algorithms could have been utilized in developing Root98 since both appear, apparently, workable and simple. However, another algorithm published by Ash and Ash in 1968 is used in preference to these two algorithms. This choice is made because the latter promises a more general computation of regular root locus points which are of particular interest here.

The algorithm described in section 2.2, for instance, cannot handle systems with dead time since it employs a direct method. Such systems have characteristic equations that contain both polynomial and exponential terms. Despite

the fact, however, that the algorithm described in section 2.3 can handle systems with dead time, it nevertheless requires the separation of the open loop transfer function into its real and imaginary parts in order to extract the derivatives. This exercise could be very complex in all but the simplest cases. Since the algorithm was used to analyze a low order system,[13] there is doubt whether it is practicable or even advantageous to apply it to very large systems.

More important still, none of these previous algorithms describes in express terms, a procedure for the exact evaluation of the break-away/break-in points. The algorithm for Root98 offers such a provision and, infact, Root98 is capable of evaluating the break-away/break-in points up to an accuracy of order of $10^{-14}$. Before the algorithm is discussed, possibility of using the bisection method for an angle search procedure is examined in the next section.

## 3.2 A Bisection Procedure for the Angle Loci

The algorithm employed in developing Root98 is applicable to the CE of linear systems with a time delay as given in the following equation:

$$1 + KF(s) = 1 + \frac{K \prod_{i=1}^{m} (s - z_i)}{\prod_{i=1}^{n} (s - p_i)} e^{-sT_o} \quad \ldots\ldots \quad 3.1$$

where

$p_i$, i = 1, .... n, is the set of finite open loop poles.

$z_i$, i=1 , .... m, is the set of finite open loop zeros.

$T_o$ is a fixed time delay and

K is the root locus gain.

Since the root locus is the set of points which satisfy the CE as a parameter of the open loop system is varied from - ∞ to +∞, the angle criterion can now be compactly expressed as

$$\text{Arg}\left\{F(s)\right\} = n\pi \quad \ldots\ldots \quad 3.2$$

$$n = \begin{cases} \pm 1, \pm 3, \pm 5, \ldots\ldots & \text{for } K > 0 \\ 0, \pm 2, \pm 4, \ldots\ldots & \text{for } K < 0 \end{cases}$$

Because the algorithm implemented in this thesis is of the angle variety, points in the s-plane must be determined such that they satisfy the angle criterion. Consequently, for an arbitrary point, s, in the complex - plane, let $F^*(s)$ be an angle function defined relative to the gain

normalized open loop transfer function, F(s), as follows:-

$$F^*(s) = \text{Arg}\left\{F(s)\right\} - n\pi$$

$$= \sum_{i=1}^{m} \text{Arg}\left\{s - z_i\right\} - \sum_{i=1}^{n} \text{Arg}\left\{s - p_i\right\} - T_o.\text{Im}(s)$$

$$- n\pi \qquad \ldots\ldots\ldots \quad 3.3$$

where for any point, s, n is chosen according to 3.2 such that:

$$-\pi \leq F^*(s) \leq \pi$$

Let

$$s = x + jy$$

then

$$F^*(x,y) = F^*(s) = F^*(x + jy) \qquad \ldots\ldots\ldots \quad 3.4$$

and so, (3.3) can be written thus

$$F^*(x, y) = \sum_{i=1}^{m} \arctan\,(x - z\,R_i,\ \ y - z\,I_i)$$

$$- \sum_{i=1}^{n} \arctan\,(x - pR_i,\ y - pI_i) - T_o.\ y - n\pi$$

$$\ldots\ldots\ldots \quad 3.5$$

where

$$z_i = zR_i + j\,zI_i \qquad \ldots\ldots\ldots \quad 3.6$$

for i=1, .... m is the set of finite open loop zeroes and

$$p_i = pR_i + jpI_i \qquad \ldots\ldots\ldots \quad 3.7$$

for i = 1, ... n is the set of finite open loop poles.

Hence a point, $s_o$, is on the root locus iff

$$F^*(s_o) = 0 \qquad \ldots\ldots \qquad 3.8$$

(3.8) indicates that the calculation of legitimate root loci points can be achieved by finding the zeros of the angle function, $F^*(s)$. Many numerical schemes are possible. In this section the possibility of using the bisection method is considered.

Let $s_o$ be a point on a given root locus. Then a step size parameter, $\Delta s$, can be calculated such that

$$F^*(s_o + \Delta s) = 0 \qquad \ldots\ldots \qquad 3.9$$

where

$$\Delta s = \Delta x + j \Delta y$$

or using the more convenient Euler representation,

$$\Delta s = (\delta, \theta)$$

where

$$\delta \triangleq |\Delta s| = |s - s_o|$$

and

$$\theta = \arg \left\{ \Delta s \right\}$$

is the angle which the vector $\overline{s_o s}$ makes with the positive direction of the real-axis. (see figure 3.1).

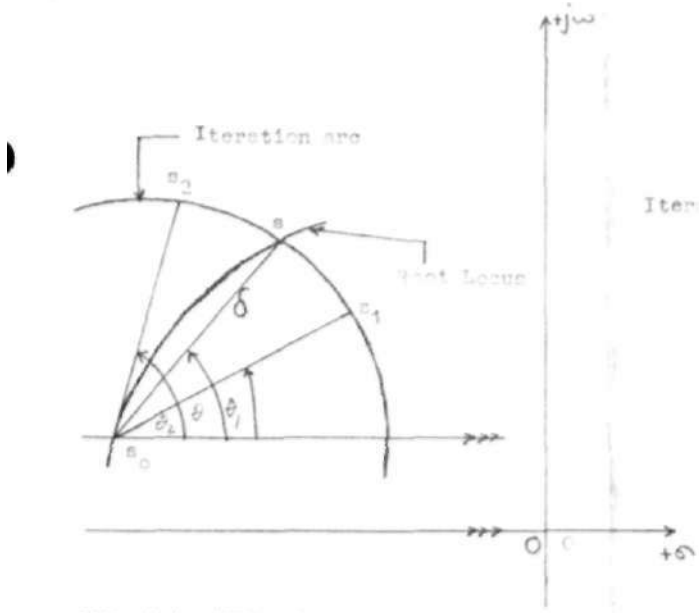



FIG. 3.1: ANGLE ITERATION USING BISECTION METHOD

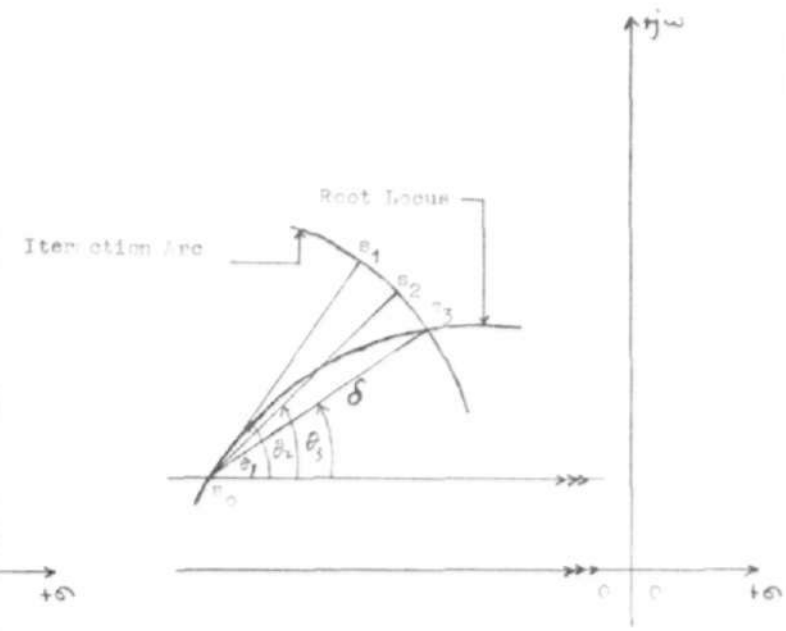



FIG. 3.2: NEWTON-RAPHSON ITERATION

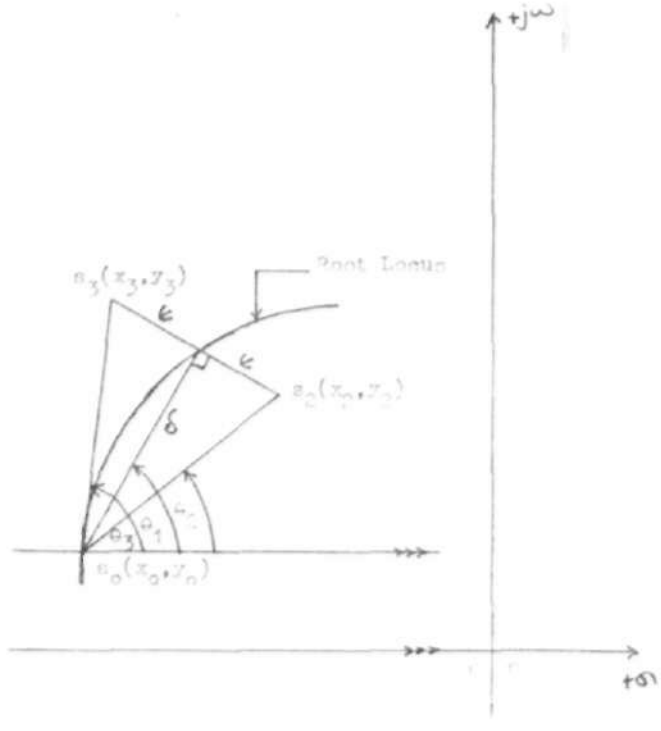



FIG. 3.3: ILLUSTRATION OF DISTANCE-CHECK PROCEDURE

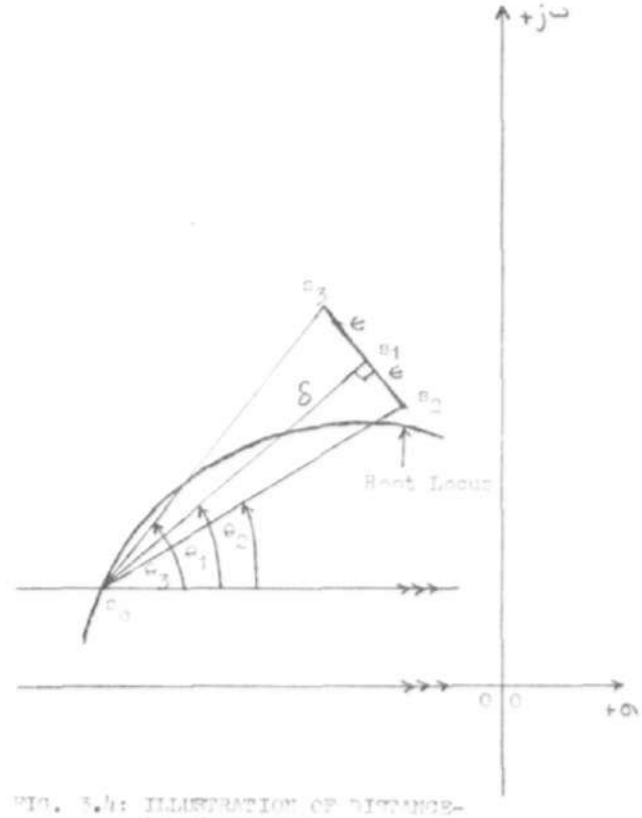



FIG. 3.4: ILLUSTRATION OF DISTANCE-CHECK FAILURE

Hence given any starting value for $\theta_o$, any iterative procedure which is convergent and stable can be used to determine or calculate the improvements on the angle, $\theta$, and subsequently result in finding the corresponding root locus point.

A simple scheme for achieving the iterative process can start with some $\theta$ and then determine the improvements, $\triangle\theta$ , on the initial guess using the "bisection method".(9,20) For this method, the angle, $\theta$, which satisfies the equation,

$$F^*(s) = 0$$

with

$$s = s_o + \triangle s$$

$$\triangle s = (\delta, \theta)$$

is bracketed by $\theta_{1,n}$ and $\theta_{2,n}$ , say at stage n such that

$$F^*(s_o + \triangle s_2) < 0$$

and

$$F^*(s_o + \triangle s_1) > 0$$

where

$$\triangle s_1 = (\delta, \theta_1)$$

and

$$\triangle s_2 = (\delta, \theta_2)$$

Note that since $\delta$ is a constant, iterates are along the arc of a circle with centre at the last point on the root locus

(see figure 3.1). The implementation of the root locus then reduces to the bisection

$$\theta^* = (\theta_1 + \theta_2)/2$$

and

$$\theta_2 = \theta^* ,$$

if $F^*(s_o + \Delta s^*) < 0$

while

$$\theta_1 = \theta^*$$

if $F^*(s_o + \Delta s^*) > 0$

The procedure is repeated until the term $|F^*(\theta)| \leq \epsilon$ where $\epsilon$ is some conveniently small error criterion or tolerance.

It is interesting to note that regardless of the nature of the starting point $s_{o,}^{(k)}$ it is possible to determine some $\Delta s^*$ in the $\delta$ - neighbourhood which is in the $\epsilon$ - neighbourhood of a root locus point provided that $s_o^{(k)}$ is on the root loci. This is equivalent to saying that errors in the calculation of preceeding point do not propagate.

Finally, it is noteworthy to observe that the bisection method does not require the derivatives of the angle function. However, its rate of convergence is first order and consequently slow. The discussion here provides a simple basis for

illustrating the determination of root locus points and highlights the importance of the angle iteration. Despite its poor speed - chanacteristics, the bisection method can be very useful at some points in the s-plane where other derivative methods can lead to singularities.

## 3.3 The Newton-Raphson Iteration Techniques for Angle Loci

### 3.3.1 The Locus Slope Angle

Using the angle function, F*(s), defined in equation (3.3) the angle iteration now requires only one initial guess rather than two. However, it also requires the derivatives of the angle function. Consider a starting point, $s_o$, on a root locus branch. Then a reasonable value for the iteration angle, $\theta$, is given by the angle which the tangent to the root locus makes with the positive direction of the real-axis at the point, $s_o$. The initial value of the angle, $\theta$, designated the locus slope angle[11] is defined as follows:

$$\theta = \arctan \left\{ - F^*_y(x_o, y_o), \; F^*_x(x_o, y_o) \right\} \qquad \ldots\ldots \; 3.10$$

where $F^*_x$ and $F^*_y$ are the partial derivatives of the angle function F*(s) w.r.t. x and y respectively given by

$$F^*_x(x_o, y_o) = \frac{\partial F^*}{\partial x}\bigg|_{(x_o, y_o)} = - \sum_{i=1}^{m} \frac{y_o - z I_i}{(x_o - z R_i)^2 + (y_o - z I_i)^2}$$

$$+ \sum_{i=1}^{n} \frac{y_o - p I_i}{(x_o - p R_i)^2 + (y_o - p I_i)^2} \qquad \ldots\ldots\ldots \; (3.11)$$

and

$$F_y^*(x_o, y_o) = \left.\frac{\partial F^*}{\partial y}\right|_{(x_o,y_o)} = \sum_{i=1}^{m} \frac{x_o - zR_i}{(x_o - zR_i)^2 + (y_o - zI_i)^2}$$

$$- \sum_{i=1}^{n} \frac{x_o - pR_i}{(x_o - pR_i)^2 + (y_o - pI_i)^2} - T_o \qquad \ldots\ldots 3.12$$

The arctangent function used in (3.10) as well as that used in (3.5) is defined as follows:

Let

$\theta = \tan^{-1}(y/x)$

such that

$$-\pi/2 \leqslant \theta \leqslant \pi/2$$

Then

$$\arctan(x, y) = \begin{cases} \theta, & \text{if } x \geqslant 0 \\ \theta + \pi, & \text{if } x < 0, \quad y > 0 \\ \theta - \pi, & \text{if } x < 0, \quad y \leqslant 0 \\ 0, & \text{if } x = 0, \quad y = 0 \end{cases} \qquad \ldots\ldots 3.13$$

i.e. $-\pi \leqslant \arctan(x, y) \leqslant \pi$

The conditions in equation (3.13) are critical to the successful implementation of the algorithm and in Root 98, the Fortran function arctan2(y,x) is used to realize the objective. Note that if the ordinary arctangent function is to be used, then the slope angle, $\theta$, should be defined as :

$$\theta = \begin{cases} \tan^{-1}(y/x) & , \text{ if } x \geqslant 0 \\ \tan^{-1}(y/x) - \pi & , \text{ if } x < 0, \ y > 0 \\ \tan^{-1}(y/x) + \pi & , \text{ if } x < 0, \ y \leqslant 0 \\ 0 & , \text{ if } x = 0, \ y = 0 \end{cases} \quad \ldots\ldots 3.14$$

The sign of the locus slope angle in (3.10) tacitly assumes that computation is in the direction of increasing gain. For computation to be in the direction of decreasing gain, then the initial guess for the iteration angle should be

$$\theta = \arctan \left\{ F^*_y(x_o, y_o), \ - F^*_x(x_o, y_o) \right\} \quad \ldots\ldots 3.15$$

Once the initial guess angle has been determined, the Newton - Raphson iteration[9,20]

$$\theta_2 = \theta_1 - \frac{F^*(\theta_1)}{F^{*\prime}(\theta_1)} \quad \ldots\ldots 3.$$

is then used to determine a point $s = x + j\,y$ along an arc of a circle whose centre is at $s_o$ and whose radius is $\delta$ which satisfies equation (3.8). Such a point is then taken as a root locus point. The terms in equation (3.16) are defined (see fig. 3.2) as follows:

$$F^*(\theta_1) = F^*(s_1)\Big|_{\theta = \theta_1} = F^*(x_o + \delta\cos\theta_1 \ , \ y_o + \delta\sin\theta_1) \quad \ldots\ldots 3.17$$

$$F^{*\prime}(\theta_1) = \frac{\partial F^*(\theta_1)}{ds} = \frac{\partial F^*(\theta)}{\partial x}\left.\frac{dx}{d\theta}\right|_{\theta=\theta_1} + \frac{\partial F^*(\theta)}{\partial y}\left.\frac{dy}{d\theta}\right|_{\theta=\theta_1}$$

$$= -F_x^*(x_o + \delta\cos\theta_1,\ y_o + \delta\sin\theta_1).\delta\sin\theta_1$$

$$+ F_y^*(x_o + \delta\cos\theta_1,\ y_o + \delta\sin\theta_1).\delta\cos\theta_1 \quad \ldots\ldots 3.18$$

### 3.3.2 Quality Check for Computed Points

Since the angle criterion could be relatively insensitive (11) to the actual root locus path in certain areas of the s-plane, it is necessary to ensure that any point which has been certified by equation (3.8) as being a valid root locus point is properly scrutinized before it is finally accepted as a root locus point. Otherwise there is the possibility of risking error accumulation, and hence non-convergence of the iteration due to error propagation. A procedure for checking this can be developed as follows:-

Suppose $s_1$ has been determined as a locus point, two points $s_2$ and $s_3$ (see fig. 3.3) at an equal distance, $\epsilon$, from the point $s_1$ and located on a line segment perpendicular to the root locus at $s_1$ are determined where $\epsilon$ is the acceptable error tolerance for the computations. If $s_1$ is in an acceptable neighbourhood of the actual point, then $s_2$ and $s_3$

can be expected to lie on opposite sides of the root locus. This condition can be expressed by one of the following inequalities,

$$F^*(s_1) \cdot F^*(s_2) \leqslant 0 \quad \ldots\ldots \quad 3.19$$

and

$$F^*(s_1) \cdot F^*(s_3) \leqslant 0 \quad \ldots\ldots \quad 3.20$$

The determination of the points $s_2$ and $s_3$ can be realized (see fig. 3.3) as follows:-

$$\theta_3 = \theta_1 + \tan^{-1} \epsilon/\delta \quad \ldots\ldots \quad 3.21$$

$$\theta_2 = \theta_1 - \tan^{-1} \epsilon/\delta \quad \ldots\ldots \quad 3.22$$

The equations of the line segments $\overline{s_2 s_3}$, $\overline{s_o s_3}$ and $\overline{s_o s_2}$ are respectively

$$y - y_1 = -\frac{(x - x_1)}{\tan \theta_1} \quad \ldots\ldots \quad 3.23$$

$$y - y_o = (x - x_o) \tan \theta_3 \quad \ldots\ldots \quad 3.24$$

$$y - y_o = (x - x_o) \tan \theta_2 \quad \ldots\ldots \quad 3.25$$

Since $\theta_2$ and $\theta_3$ are known from equations (3.22) and (3.21), respectively, the pairs of equations (3.23), (3.24) and (3.23), (3.25) could be solved simultaneously to obtain the values of $x_3$, $y_3$ and $x_2$, $y_2$ respectively. Using the matrix notation these set of equation can be written in a more compact form viz:

$$\begin{pmatrix} 1 & \cot\theta_1 \\ 1 & -\tan\theta_3 \end{pmatrix} \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} y_1 + x_1 \cot\theta_1 \\ y_o - x_o \tan\theta_3 \end{pmatrix} \quad \ldots\ldots \quad 3.26$$

and

$$\begin{pmatrix} 1 & \cot\theta_1 \\ 1 & -\tan\theta_2 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \quad \begin{pmatrix} y_1 + x_1 \cot\theta_1 \\ y_o - x_o \tan\theta_2 \end{pmatrix} \quad \ldots\ldots \quad 3.27$$

By Crammers rule, these can then be written viz:

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & \cot\theta_1 \\ 1 & -\tan\theta_3 \end{pmatrix}^{-1} \begin{pmatrix} y_1 + x_1 \cot\theta_1 \\ y_o - x_o \tan\theta_3 \end{pmatrix}$$

$$= \frac{1}{\tan\theta_3 - \cot\theta_1} \begin{pmatrix} -\tan\theta_3 \cot\theta_1 & \\ 1 & -1 \end{pmatrix} \begin{pmatrix} y_1 + x_1 \cot\theta_1 \\ y_o - x_o \tan\theta_3 \end{pmatrix}$$

$\ldots\ldots$ 3.28

and

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \frac{1}{\tan\theta_2 - \cot\theta_1} \begin{pmatrix} -\tan\theta_2 \cot\theta_1 & \\ 1 & -1 \end{pmatrix} \begin{pmatrix} y_1 + x_1 \cot\theta_1 \\ y_o - x_o \tan\theta \end{pmatrix}$$

$\ldots\ldots$ 3.29

which implies that the points $s_2$ and $s_3$ have been determined as

$$s_2 = x_2 + jy_2$$

and

$$s_3 = x_3 + jy_3$$

If $|F^*(s_2)| \leqslant \epsilon$

$|F^*(s_3)| \leqslant \epsilon$

and

$|F^*(s_1)| \leqslant \epsilon$

but neither of the inequalities of (3.19) and (3.20) holds true, then the prevailing situation is as illustrated in Fig. 3.4.

Under such circumstances, it is apparent that the computed point is not very good because the anlge function is rather insensitive in its vicinity. The best thing to do in such a case is to ascertain whichever of $s_2$ and $s_3$ is closer to the root locus by determining how close,

$|F^*(s_2)|$ and $|F^*(s_3)|$ are to zero.

If $|F^*(s_2)| < |F^*(s_3)|$ $s_1$ is replaced by $s_2$ and the distance check is repeated. Otherwise $s_1$ is replaced by $s_3$ and the distance check is once more repeated.

### 3.3.3 Determination of Break-away/Break-in Points

#### 3.3.3.1 Branch-Points

The conventional break-away/break-in points are defined as the points on the real-axis at which a root locus branch leaves (breaks away from) or enters (breaks into) the real-

axis. In this thesis the two terms are used to denote points on the s-plane where two loci intersect or where a root locus branch changes direction. Such points will be referred to generally as branch - points.

If an algorithm is to be capable of descriminating one root locus branch from another, then it must be able to predict the presence of root locus branch points. A logical procedure must, therefore, be introduced into the algorithm based on the properties of root locus branch points in order to identify them. Two possible procedures are described in the next two sub-sections.

3.3.3.2 Determination of Branch Points Based on the Characteristics of the Derivatives of the Angle Function

At a branch - point, the partial derivatives of the angle function[11], $F^*_x$ and $F^*_y$ are each zero and the locus slope angle, $\theta$, defined by the special arctangent function as

$$\theta = \arctan\left(-F^*_y , F^*_x\right) = 0 \qquad \ldots\ldots 3.30$$

More important still, the locus slope angle also changes sign whenever it crosses a branch-point of a root locus unless its value before the branch-point is either 0 or $\pi$. If the locus slope angle before a branch-point is 0, then

its value after the branch-point will be $\overline{+}\pi$. Conversely, if the value of $\theta$ before a branch-point is $\overline{+}\pi$, its value after that branch-point will be 0. Hence the following criteria may be used to detect the branch points:-

Let $\theta_1$ be the locus slope angle before a branch-point, $s_b$, and $\theta_2$ the slope angle (in the same direction) after the branch-point, then

(i) $\quad \text{sgn}\left\{\theta_1\right\} = -\text{sgn}\left\{\theta_2\right\}$

(ii) $\quad \left|\theta_2 - \theta_1\right| > \overline{+}\pi/2$

However, in certain situations the values of the locus slope angle may meet the above conditions at ordinary turning points of the root locus branch (angle function) which are no branch points. The above criteria can cause malfunction of the algorithm under such circumstances.

3.3.3.3 Determination of Branch-Points Based on the Properties of the Complex Gain

From the properties of the root loci (c f. chapter I) the derivatives $\frac{d\,K(s)}{ds}$ of the complex gain, K(s), is zero at a branch-point. If the computation is in the direction of the increasing gain, then $\frac{dK(s)}{ds}$ is + ve before the branch-point, $s_b$, and - ve after the branch point and hence

has a maximum at $s_b$. If the computation is in the direction of decreasing gain, the reverse is the case and K(s) has a minimum at $s_b$. A logical procedure for detecting a branch-point using this scheme may be as follows:-

Let $s_1$ and $s_2$ be any two successive computed points certified to be on a root locus branch. Then if

$$\operatorname{sgn}\left\{\frac{dK(s)}{ds}\right\}_{s=s_2} = -\operatorname{sgn}\left\{\frac{dK(s)}{ds}\right\}_{s=s_1}$$

it implies that a branch-point lies between $s_1$ and $s_2$. Otherwise there is no branch-point between these locus points.

Once any of the above methods has been used to detect the existence of a branch-point between any two successive locus points, either the bisection method or a step cutting method could be used to approach the branch-point close enough in order to enhance the evaluation of the branch-point to a desired accuracy using the conditions (11)

$$F_x^*(s_b) = F_y^*(s_b) = 0$$

or

$$\left.\frac{dK(s)}{ds}\right|_{s=s_b} = 0$$

If the point $s_o$ is sufficiently close to $s_b(x_b, y_b)$ but

$$F^*_x(s_o) \neq F^*_y(s_o) \neq 0,$$

then the Newton-Raphson iteration in 2 - dimension can be used to determine the changes, $\triangle x$ and $\triangle y$, in x and y respectively, such that

$$F^*_x(s_o + \triangle s) = F^*_y(s_o + \triangle s) = 0$$

where

$$\triangle s = \triangle x + j\triangle y$$

and

$$x_b = x_o + \triangle x$$

$$y_b = y_o + \triangle y$$

It can be shown that the required changes in x and y are given by[11]

$$\begin{bmatrix} \triangle x \\ \triangle y \end{bmatrix} = \begin{bmatrix} F^*_{xx} & F^*_{xy} \\ F^*_{xy} & F^*_{yy} \end{bmatrix}^{-1} \begin{bmatrix} F^*_x \\ F^*_y \end{bmatrix} \Bigg|_{x_o,\, y_o}$$

$$= \frac{1}{F^*_{xx} + F^{*2}_{xy}} \begin{bmatrix} -F^*_x F^*_{xx} - F^*_x F^*_{xy} \\ F^*_y F^*_{xx} - F^*_x F^*_{xy} \end{bmatrix} \Bigg|_{x_o, y_o} \quad \ldots\ldots 3.32$$

where $F^*_{xx}$ , $F^*_{yy}$ are the second partial derivatives of $F^*$ w.r.t. x and y respectively and $F^*_{xy}$ is the partial derivative of $F^*_x$ w.r.t. y and are defined viz:-

$$F^*_{xx}(x_o, y_o) = \sum_{i=1}^{m} \frac{2\,(x_o - pR_i)\,(y_o - pI_i)^2}{(x_o - zR_i)^2 + (y_o - zI_i)^2}$$

$$- \sum_{i=1}^{n} \frac{2\,(x_o - pR_i)\,(y_o - pI_i)}{(x_o - pR_i)^2 + (y_o - pI_i)^2} \qquad \ldots\ldots 3.33$$

$$F^*_{yy}(x_o, y_o) = - F^*_{xx}(x_o, y_o) \qquad \ldots\ldots 3.34$$

and

$$F^*_{xy}(x_o, y_o) = - \sum_{i=1}^{m} \frac{(x_o - zR_i)^2 - (y_o - zI_i)^2}{(x_o - zR_i)^2 + (y_o - zI_i)^2}$$

$$+ \sum_{i=1}^{n} \frac{(x_o - pR_i)^2 - (y_o - pI_i)}{(x_o - pR_i)^2 + (y_o - pI_i)^2} \qquad \ldots\ldots 3.35$$

3.3.4 Determination of Take-off Angle

After the computation of a branch-point, the take-off angle from the branch-point can be determined using the relation[11]

$$\phi_n = \overline{\pi}/n \qquad \ldots\ldots \quad 3.36$$

where $n$ is the number of branches leaving or entering the point. Since $n$ is not known before hand, a set of test points

$$s_T(n) = x_b + \delta \cos \phi_n + j\,(y_b + \delta \sin \phi_n) \qquad \ldots\ldots \quad 3.37$$

$$n = \pm 1, \pm 2, \ldots\ldots$$

are generated. If for any $s_T$, $|F^*(s_T)| \leq \epsilon$ and the locus slope angle at $s_T$ is not equal to either the slope angle before or after the branch-point, then the current value of $\phi_n$ is the take-off angle.

3.3.5 Specialized Features of Root98

The program, Root98, has been implemented as a design/ analysis and root locus teaching aid. Consequently it is arranged that the program be capable of generating root loci points by accepting any initial point on the root locus, $s_o$, suitable step size, $\delta$, error tolerance, $\epsilon$, a fixed time delay, $T_o$, and the number/ positions of the open loop poles and zeros. This capability is realized by organizing that the program determines the direction and starting indices $k_1$, $k_2$, $k_3$, $k_4$ and $k_5$ described as follows:-

(a) $k_1$ is the main direction index.

$k_1 = 1$ implies compute in the direction of decreasing gain.

(b) $k_2$ is a starting index.

$k_2 = 1$ implies starting point is an open loop pole;

$k_2 = 0$ otherwise.

If $k_2 = 1$, then $k_1 = 0$.

(c) $k_3$ is another starting index.

$k_3 = 1$ implies starting point is an open loop zero.

$k_3 = 0$ otherwise.

If $k_3 = 1$, then $k_1 = 1$.

(d) $k_4$ is a real-axis branch index.

$k_4 = 1$ implies that a root locus branch lies to the right of the starting point on the real-axis.

(e) $k_5$ is an off-axis branch index.

$k_5 = 1$ implies that the starting point has a non zero imaginary part. Hence locus has a conjugate.

### 3.3.6 Algorithm for Root 98

In summary, the algorithm for Root 98 can be set down viz:

Let $s_o = x_o + j\, y_o$

be the starting point;

$p_i$, $i = 1, \ldots n$ be the set of finite open loop poles;

$z_i$, $i = 1, \ldots m$ be the set of finite open loop zeros;

The step size, error tolerance and time delay be respectively given by

* $\delta = a_1$

* $\epsilon = a_2$

* $T_0 = a_3$

Then:

Step 1. Determine the five direction and starting indices $k_1$, $k_2$, $k_3$, $k_4$, and $k_5$.

Step 2. Compute the gain at $s_o^{(k)}$ if $s_o^{(k)}$ is not an open loop zero. Call this gain $g_1^{(k)}$. If $s_o^{(k)}$ is an open loop zero, assign $g_1^{(k)}$ a very large value say $10^{100}$.

Step 3. Determine the locus slope angle $\theta_1$ at $s_o^{(k)}$. Call this angle $s_{1.1}^{(k)}$.

Step 4. Evaluate $s_1^{(k)}$ such that $s_1^{(k)} = s_o^{(k)} + \triangle s$

where

$$\triangle s = \delta \cos \theta_1 + j \delta \sin \theta_1$$

Hence evaluate the angle function $F^*(s)$ at

$$s = s_1^{(k)}$$

* $a_1$, $a_2$, $a_3$ are any suitable values for the step size, error criterion and time delay respectively.

Step 5. If $\left| F^*(s_1^{(k)}) \right| \geq \epsilon$, compute the change $\Delta\theta_1$ such that

$$\Delta\theta_1 = \frac{- F^*(\theta_1)}{F^{*'}(\theta_1)}$$

where $F^*(\theta_1)$ and $F^{*'}(\theta_1)$ are as defined in (3.17) and (3.18) respectively.

Then

$$\theta_1 := \theta_1 + \Delta\theta_1$$

and repeat steps (4) and (5) until

$$\left| F^*(s_1^{(k)}) \right| \leq \epsilon$$

Step 6. Compute $s_2$ and $s_3$ where

$$s_2 = x_2 + j\, y_2$$

$$s_3 = x_3 + j\, y_3$$

using equations (3.28) and (3.29) and evaluate $F^*(s_2)$ and $F^*(s_3)$.

Step 7. If

$$F^*(s_2) \times F^*(s_1^{(k)}) \leq 0$$

or

$$F^*(s_3) \times F^*(s_1^{(k)}) \leq 0,$$

determine the gain and locus slope angles at $s_1^{(k)}$. Call the gain $g_1^{(k+1)}$ and the slope angle $s_{11}^{(k+1)}$.

Evaluate

$$\left| s_{11}^{(k+1)} - s_{11}^{(k)} \right| \text{ and } \left| g_1^{(k+1)} - g_1^{(k)} \right|$$

(a) If

$$k_2 = 1$$

or

$$k_1 = 0$$

and

$$\left| s_{11}^{(k+1)} - s_{11}^{(k)} \right| < \overline{11}/2$$

or $\left| g^{(k+1)} - g_1^{(k)} \right| > 0$

store the value of $s_1^{(k)}$ as a locus point and then

$$s_o^{(k)} := s_1^{(k)}$$

repeat steps (2) through (7). Otherwise go to step (9)

(b) If

$$k_3 = 1$$

or

$$k_1 = 1$$

and

$$\left| s_{11}^{(k+1)} - s_{11}^{(k)} \right| < \overline{11}/2$$

or

$$\left| g_1^{(k+1)} - g_1^{(k)} \right| < 0$$

store the value of $s_1^{(k)}$ as a true locus point then

$$s_o^{(k)} := s_1^{(k)}$$

and repeat steps (2) through (7). Otherwise go to step (9).

Step 8. If

$$F^*(s_2) \times F^*(s_1^{(k)}) - > 0$$

and

$$F^*(s_3) \times F^*(s_1^{(k)}) - > 0$$

Evaluate

$$\Delta F = \left|F^*(s_3)\right| - \left|F^*(s_2)\right|$$

If $\Delta F$ is + ve

$$s_1^{(k)} := s_3$$

and repeat step 6.

Step 9. $d := a_1 /256$

$$\delta := \delta/4$$

Repeat steps (4) through (8) until

$$\left|s_1^{(k)} - s_o^{(k)}\right| \leq \epsilon$$

or

$$\delta = d$$

Then compute $\Delta x$ and $\Delta y$ as defined by equation (3.32).

$$s_o^{(k)} := s_o^{(k)} + \Delta s$$

where

$$\Delta s = \Delta x + j \Delta y$$

and compute

$$F_x^*(s_o^{(k)}) \quad , \quad F_y^*(s_o^{(k)})$$

If

$$F_x^*(s_o^{(k)}) = F_y^*(s_o^{(k)}) = 0$$

or

$$F_x^*(s_o^{(k)}) \leqslant 10^{-14} ,$$

$$F_y^*(s_o^{(k)}) \leqslant 10^{-14} ,$$

$$\triangle x \leqslant 10^{-14} ,$$

and

$$\triangle y \leqslant 10^{-14}$$

where $10^{-14}$ is the desired accuracy in break-away point computation, then

$$s_b := s_o^{(k)}$$

Store $s_b$ and go to step (10). Otherwise repeat step 9 until stipulated conditions are satisfied.

Step 10. Determine $s_T$ such that

$$s_T = x_T + j\, y_T$$

where

$$x_T = x_b + \delta \cos \phi_n$$

$$y_T = y_b + \delta \sin \phi_n$$

and

$$\phi_n = \overline{11}/n$$

$$n = \pm 1, \pm 2, \ldots\ldots$$

Compute $F^*(s_T)$ . If

$$\left|F^*(s_T)\right| \leqslant \epsilon$$

and neither

$$s_{11}^{(k)} = \phi_n$$

nor

$$s_{11}^{(k+1)} = \phi_n$$

then

$$s_1^{(k)} := s_T \quad .$$

Store $s_1^{(k)}$ as a root locus point.

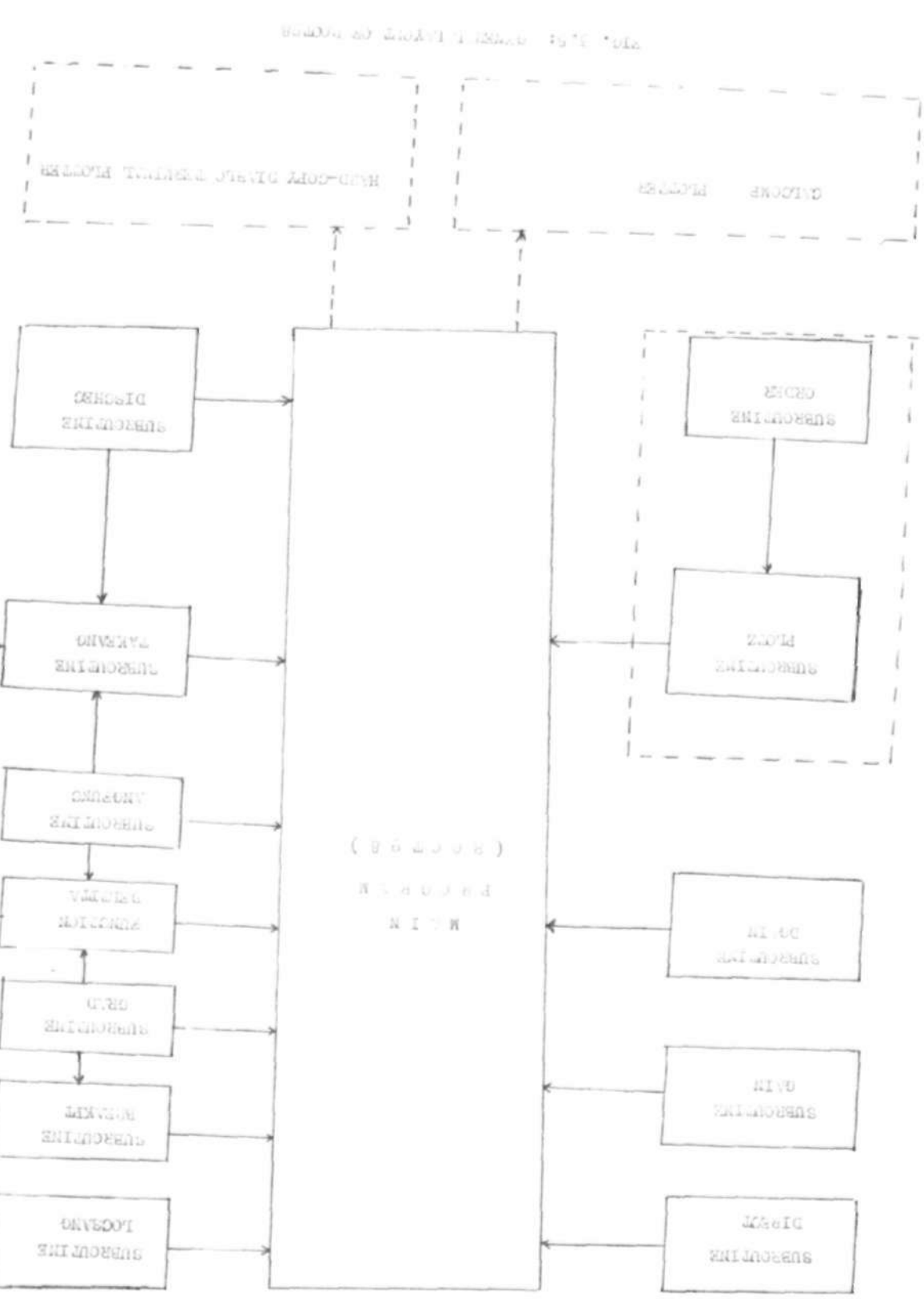
$$s_o^{(k)} := s_1^{(k)}$$

Repeat steps (2) through (10) until relevant number of point has been computed.

Step 11. Plot the computed points using the plotting subroutine.

## 3.4 Organization and Documentation of Root 98

The general organization of Root 98 is as shown in fig. 3.5 and the functions of the various sub-programs are as tabulated in Table 3.1. The flow charts for the sub-programs are shown in appendix A.


MAIN PROGRAM
( DRIVER )
SUBROUTINE INPUT
SUBROUTINE GAIN
SUBROUTINE DISPLY
SUBROUTINE PLOT2
SUBROUTINE CROSS
SUBROUTINE DISCRE
SUBROUTINE FAXRAND
SUBROUTINE ANDPUND
SUBROUTINE FUNCTION
SUBROUTINE GRID
SUBROUTINE RUNKUT
SUBROUTINE LOCBAND
MULTI-CONV DIGAL CHANNEL MODELER
CLOCKS MODELER

| SUBPROGRAM | F U N C T I O N |
|---|---|
| SUBROUTINE DIRECT | Determines the direction and starting indices $K_1$, $K_2$, $K_3$, $K_4$ and $K_5$; given the starting point $s_o$, number of poles, np, and the number of zeros, nz and the pole - zero configuration |
| SUBROUTINE GAIN | Evaluates the gain $G^{(k)}$ at any given point $s_o^{(k)} = x_o + j y_o$ |
| SUBROUTINE DGAIN | Evaluates the derivative, $DG^{(k)}$ of the gain |
| SUBROUTINE GRAD | Determines the derivatives $F^*_x$ and $F^*_y$ of the angle function $F^*(x,y)$ at any point $s_o = x_o + j y_o$ |
| SUBROUTINE LOCSANG | Determines, $\theta$ , the locus slope angle at any point $s_o^{(k)} = x_o + j y_o$ |
| FUNCTION DELTITA | Iterates for better values of $\theta$ using the Newton-Raphson technique. |
| SUBROUTINE ANGFUNC | Evaluates the angle function, $F^*(x,y)$ at any given point $s_1^{(k)} = x + \delta\cos\theta + j(y + \delta\sin\theta)$; $\delta$ is the step size |
| SUBROUTINE DISCHEC | Ensures that the value of the computed point $s_1^{(k)}$ is within $\epsilon$ - neighbourhood of the actual root locus; $\epsilon$ is the error criterion. |
| SUBROUTINE BREAKPT | Evalueates the value of the branch-point, $s_b$., to a desired degree of accuracy. Utilizes Newton-Raphson iteration in 2 - D . |
| SUBROUTINE TAKEANG | Determines the possible takeoff-angle, $\phi$, after a branch point has been located and evaluated |
| SUBROUTINE PLOTZ | Realizes the plot of the computed points. Capable of plotting a total of 98 points on one curve (root locus branch). |
| SUBROUTINE ORDER | Arranges the computed points in order of decreasing magnitude of the imaginary parts of the root loci. (Preparatory step to the plotting). |

Table 3.1: Subprograms and the Functions they perform

# CHAPTER IV

## RESULT OF TESTS FOR ROOT98

### 4.1 Introduction

Root98 was tested using a variety of systems. The program proved efficient in terms of computation time and the accuracy of the computed points. Table 4.1 gives the execution time for some of the test systems and a listing of the computed points is included in appendix B for a few of the systems.

| System No. | Open Loop TF | Type | Order | No. of Points per branch | Total No. of Points | Time (Cp sec) |
|---|---|---|---|---|---|---|
| 1 | $\frac{K(s+3)}{(s+1+j)(s+1-j)}$ | 0 | 2 | 200 | 400 | 5.34 |
| 2 | $\frac{K(s+5.5)(s+1.5)}{s(s+1)(s+5)}$ | 1 | 3 | 200 | 600 | 12.2? |
| 3 | $\frac{K}{s(s+4+j2)(s+4-j2)}$ | 1 | 3 | 200 | 600 | 8.00 |
| 4 | $\frac{K(s+1+j2)(s+1-j2)}{s^2(s+0.5)(s+2)(s+8)^2}$ | 2 | 6 | 200 | 1200 | 30.35 |

Table 4.1: Execution time for some systems.

It was possible in certain cases to satisfy, exact.[1]/, the equation

$$F^*(s) = 0 \qquad \text{U.1}$$

In such cases, the accuracy of the computed points is limited hy the computer itself. However, along root loci segments where the actual locus is not sensitive to the angle function, P*(s), the accuracy depends on the ratio' ,4 •_ , of the error tolerance, € , to the step size, 5 • For fixed £, accuracy increases with *0* but for fixed £ , accuracy decreases with C •

The computation time is observed to depend on the following parameters:-

1. The size of the system - that is the no. of singularities (poles and zeros) of the system; computation time is more for large systems.

2. The step size, 5 ; for constant * , the angle iteration converges faster for smaller values of & and this reduces the computation time for a fixed no. of points. However, a reduction in & also reduces the maximum possible distance that could be covered along the locus by the generation of the same no. of points.

3. The error tolerance ; If

$$F^*(s) \neq 0$$

then the computation time for a given number of points and a fixed value of $\delta$ will be inversely proportional to the error tolerance.

4. The number of branch-points: The exact evaluation of the branch points is really time consuming. The program therefore takes a longer time to compute locus points when some branch points are inclusive.

Values of $\epsilon$ and $\delta$ used in the computation lie in the following ranges

$$0 \leq \epsilon \leq 10^{-3}$$

and

$$0.001 \leq \delta \leq 0.25$$

$\epsilon$ and $\delta$ values outside these ranges could be used with caution. However, the optimal ranges recommended for faster computation and better accuracy are:

$$0 \leq \epsilon \leq 10^{-4}$$

$$0.01 \leq \delta \leq 0.125$$

The plotting routine used in the plot of the computed points is a modification of a plotter program developed by

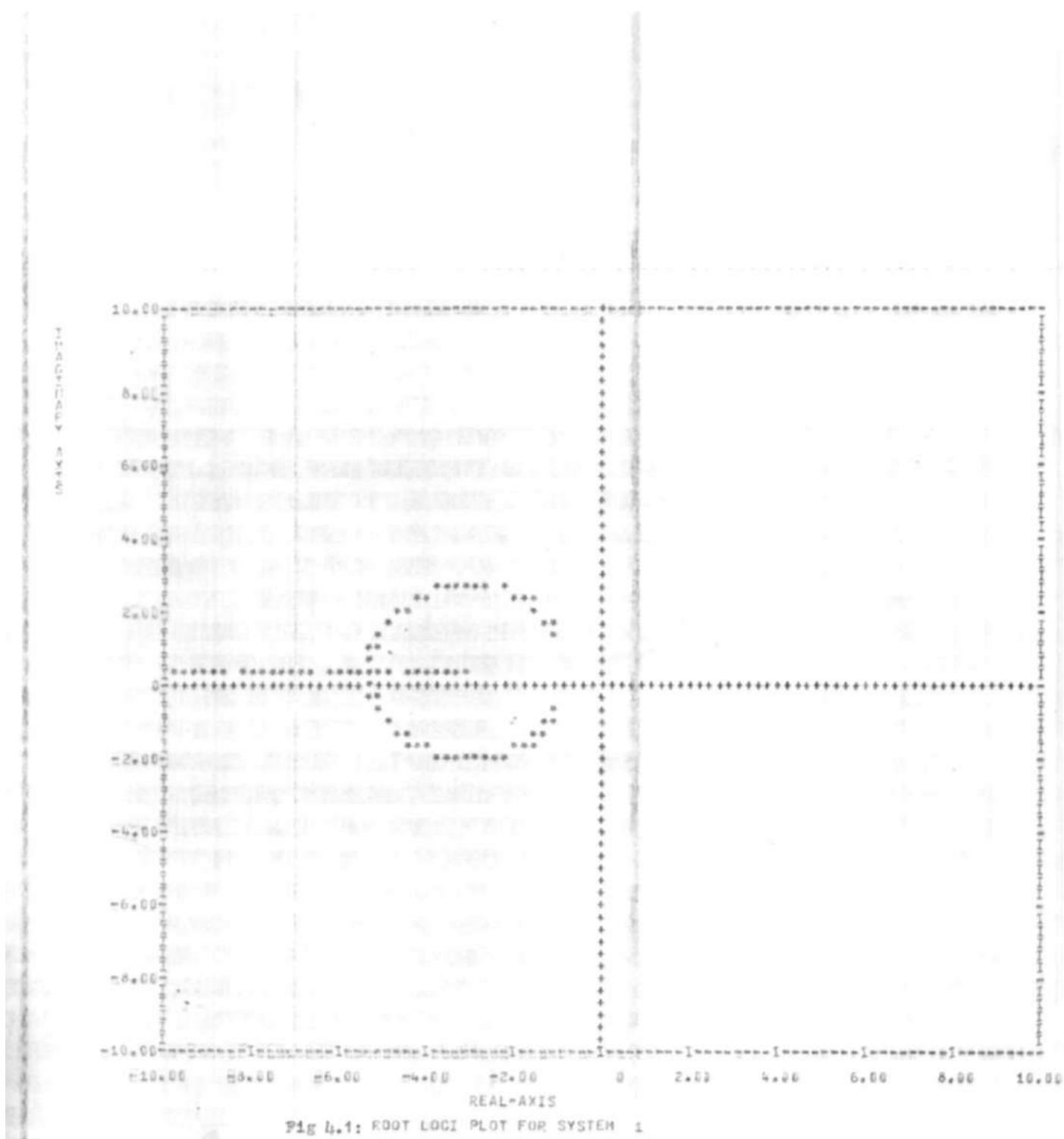



Fig 4.1: ROOT LOCI PLOT FOR SYSTEM 1

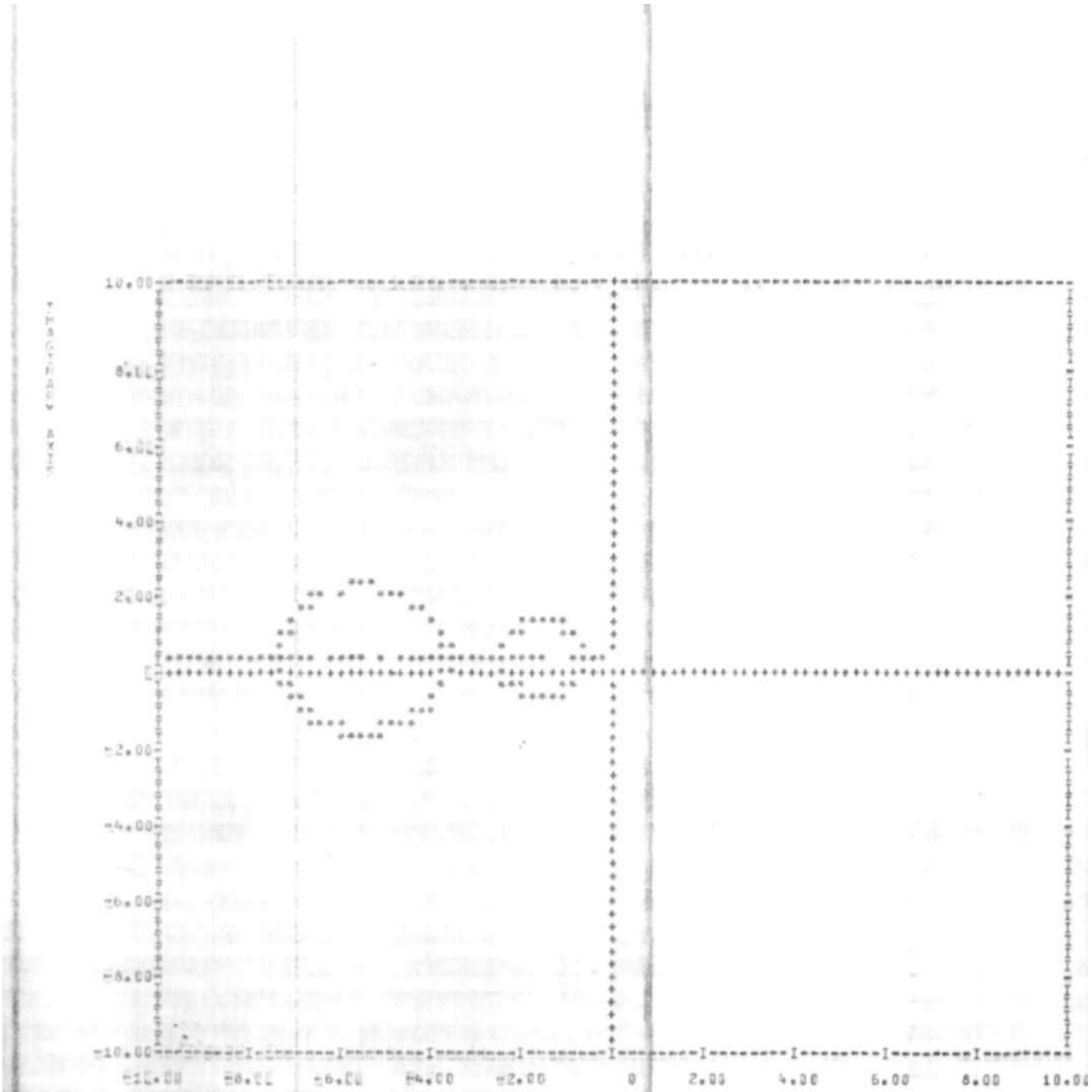



Fig 4.2: ROOT LOCI PLOT FOR SYSTEM 2

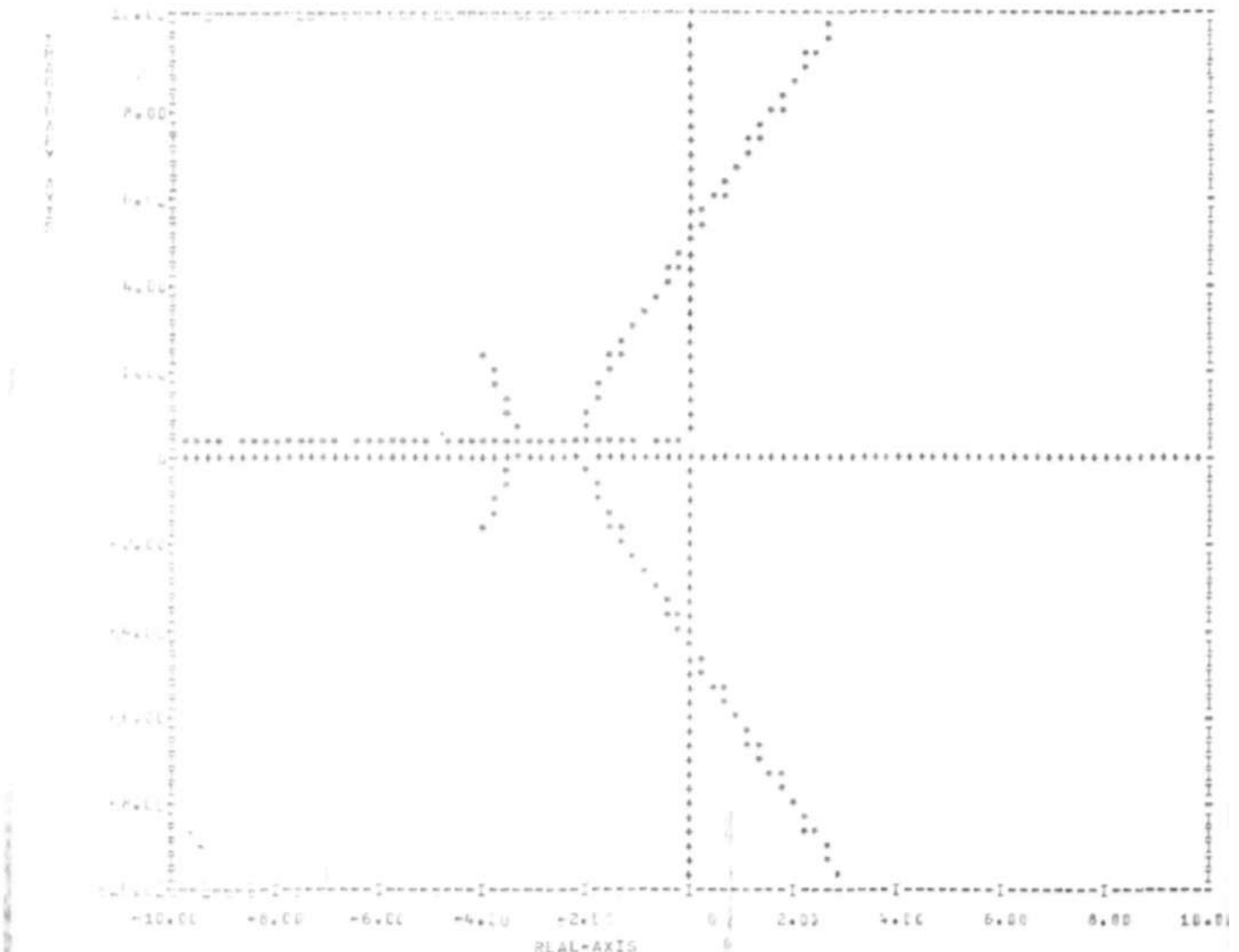



**Fig. 4.3:** ROOT LOCI PLOT FOR SYSTEM 3

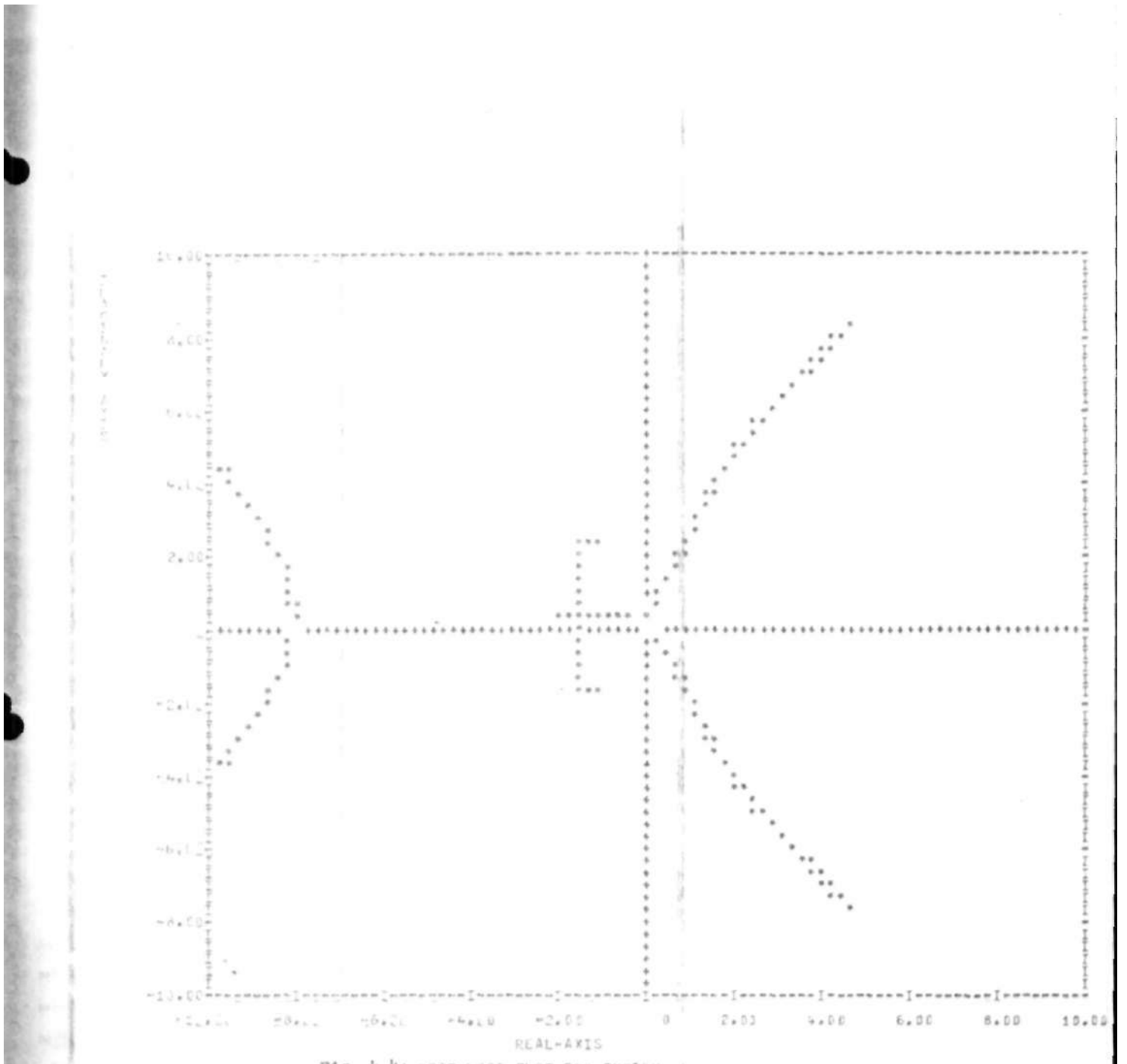



Fig. 4.4: ROOT LOCI PLOT FOR SYSTEM 4

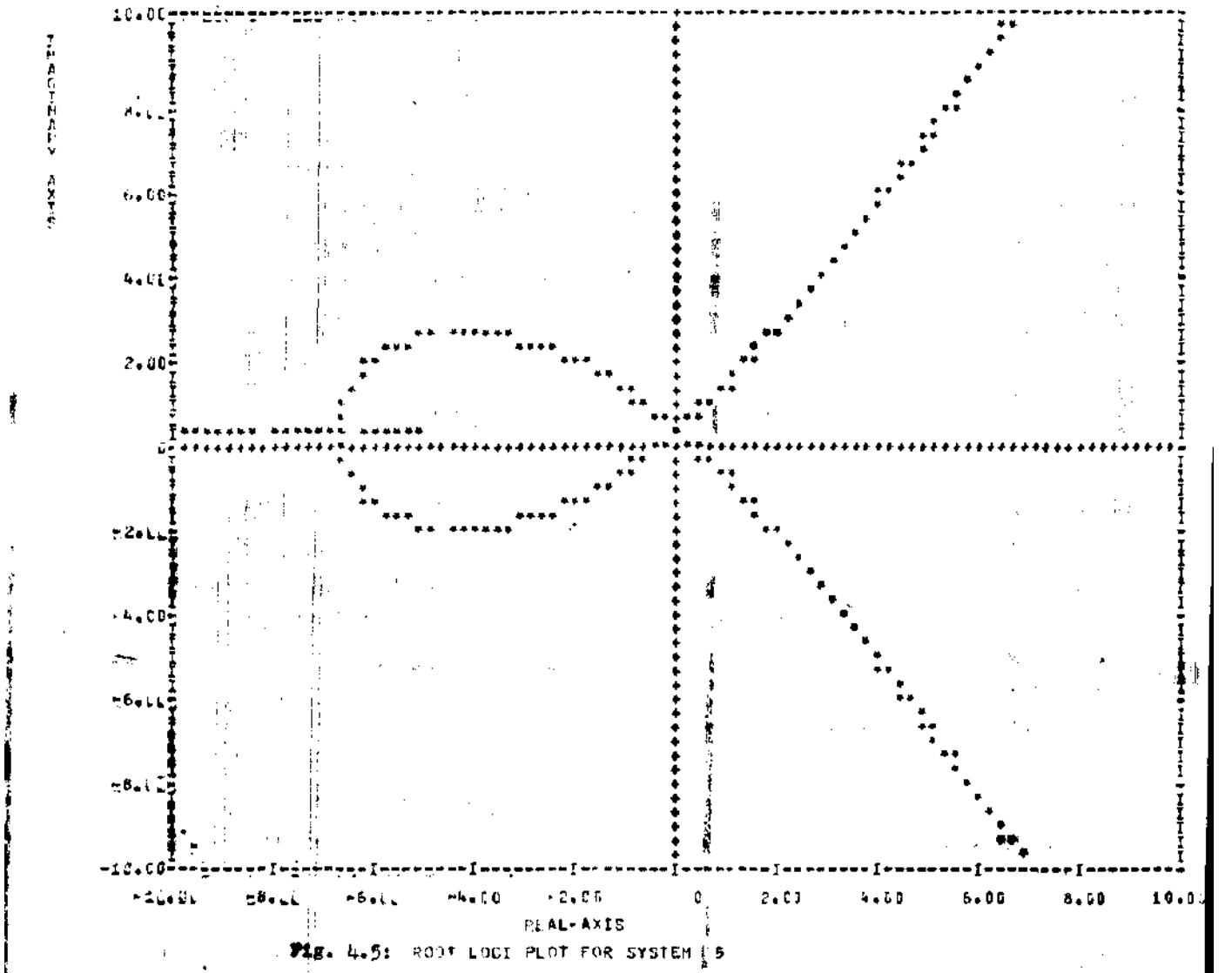



Fig. 4.5: ROOT LOCI PLOT FOR SYSTEM 5

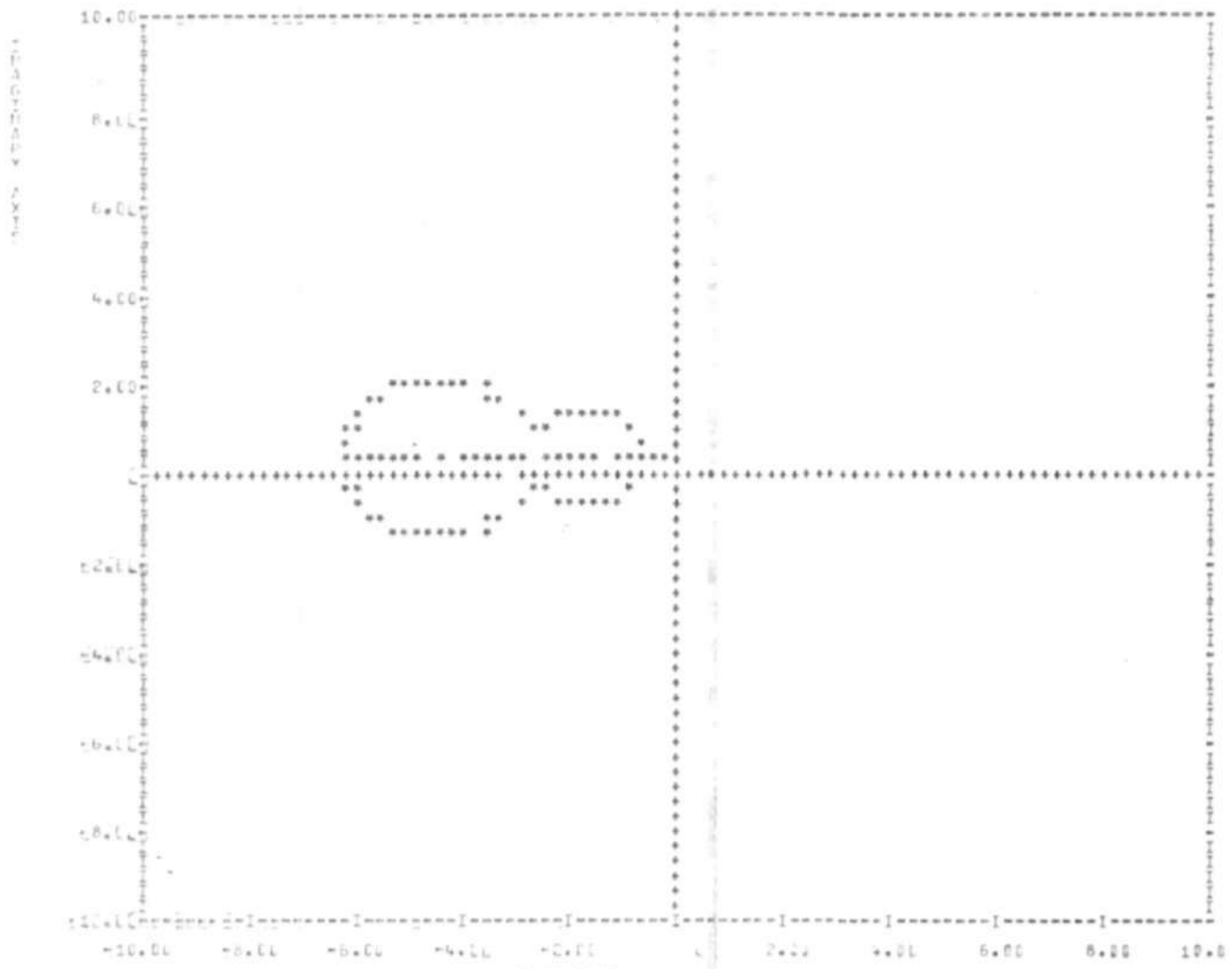



Fig. 4.6: ROOT LOCI PLOT FOR SYSTEM 6

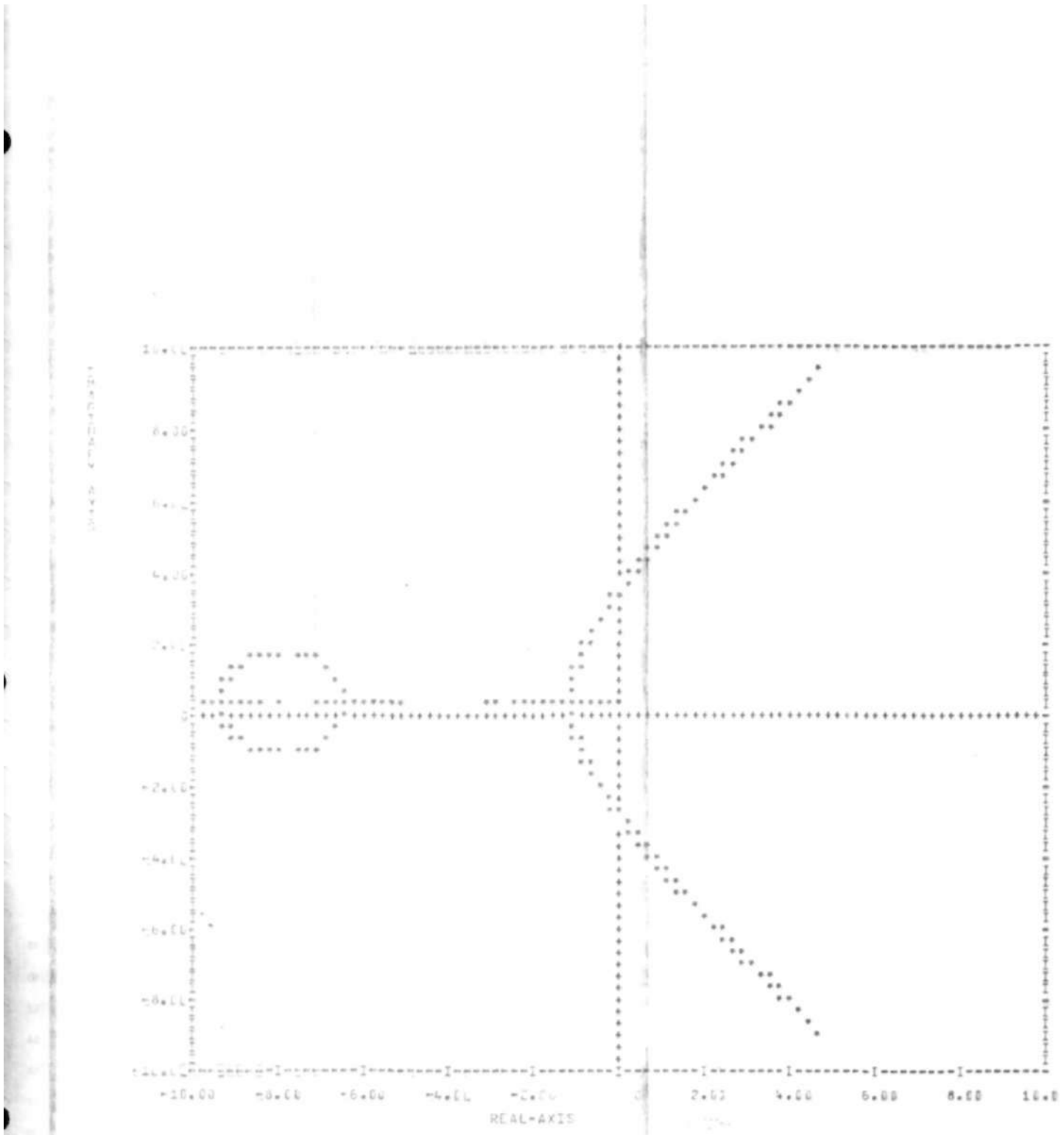



Fig.4.7: ROOT LOCI PLOT FOR SYSTEM 7

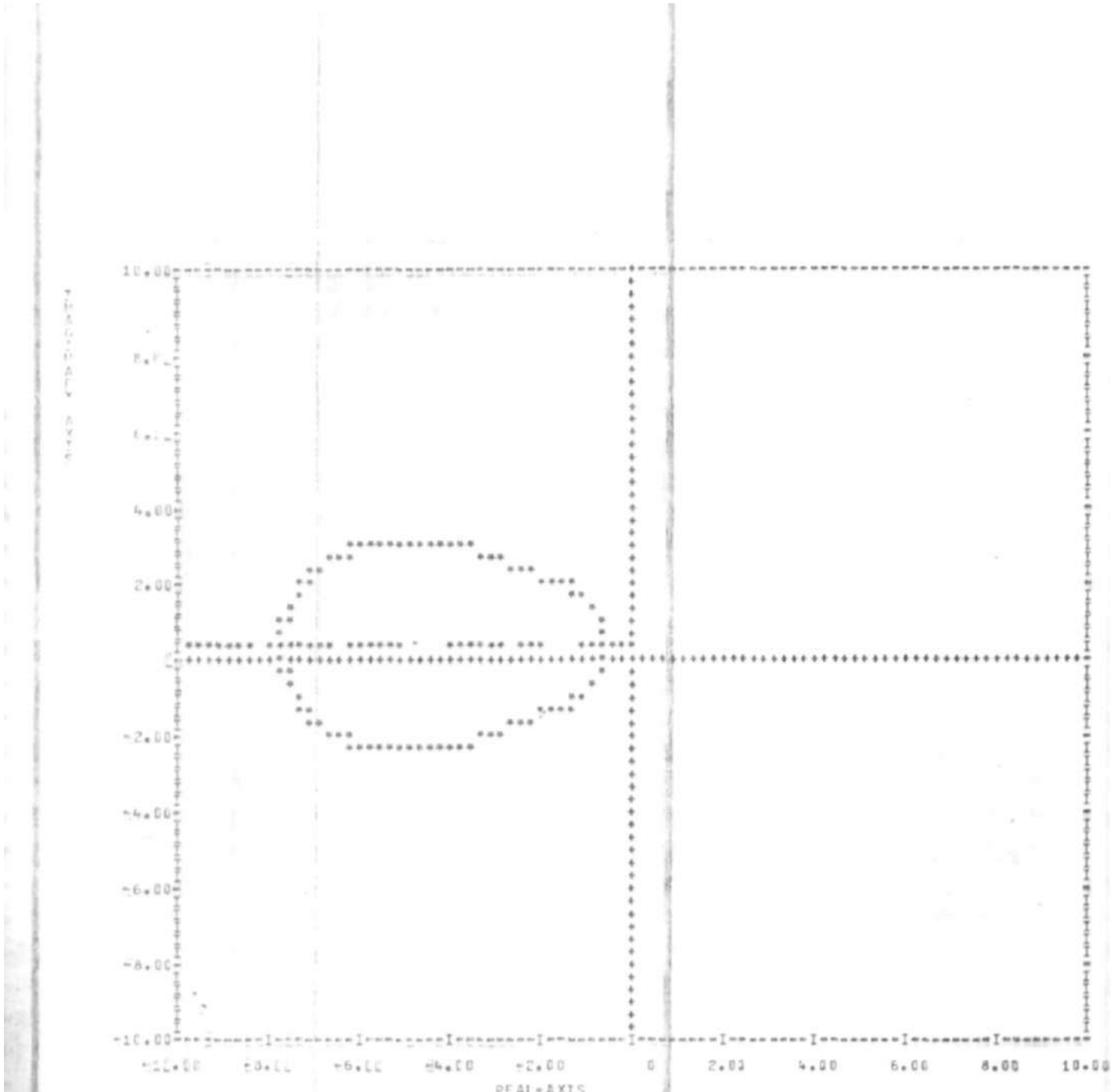



Fig. 4.8: ROOT LOCI PLOT FOR SYSTEM 8

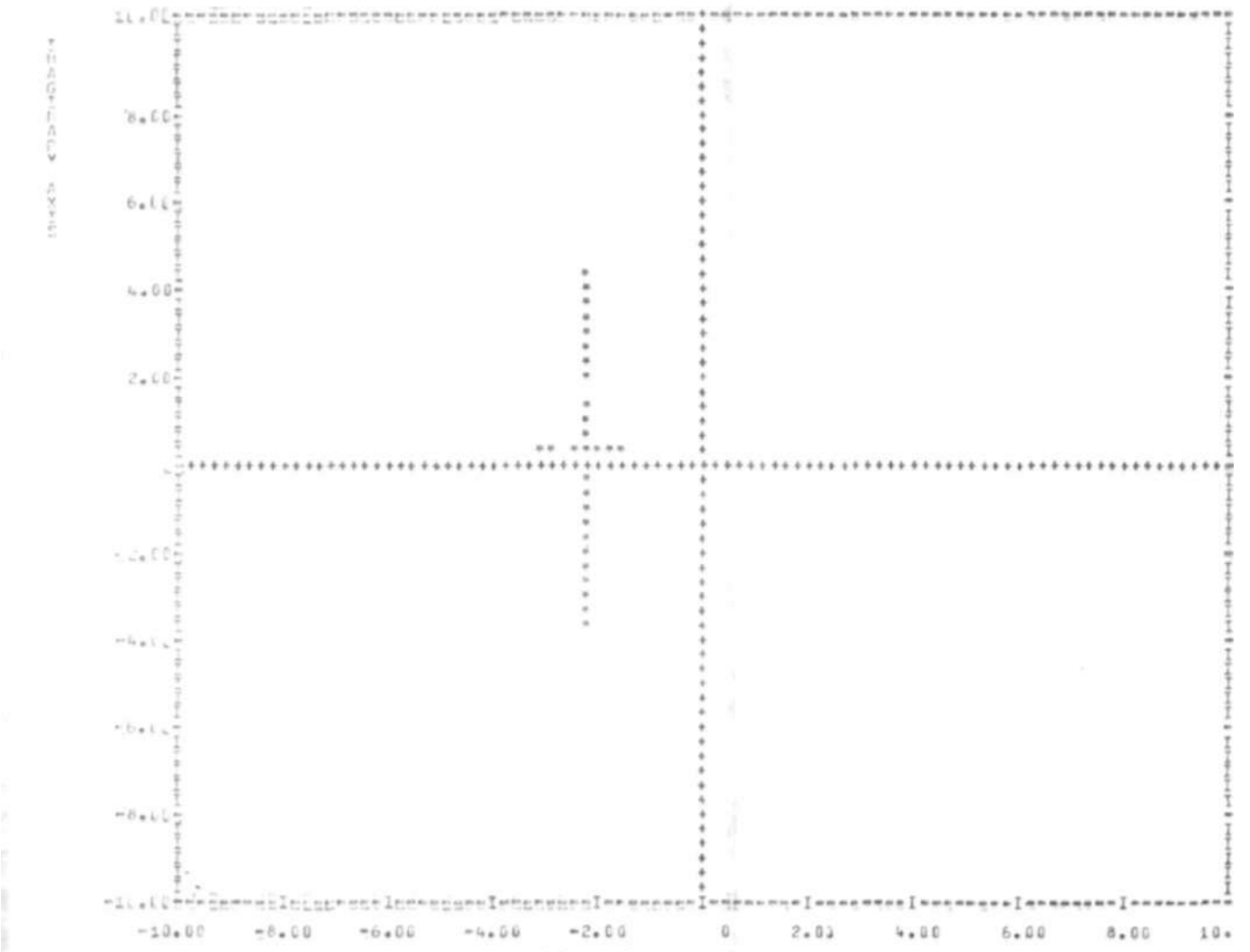



Fig. 4.9: ROOT LOCI PLOT FOR SYSTEM 10

System 10: $GH(s) = \frac{K}{(s+1.5)(s+3)}$

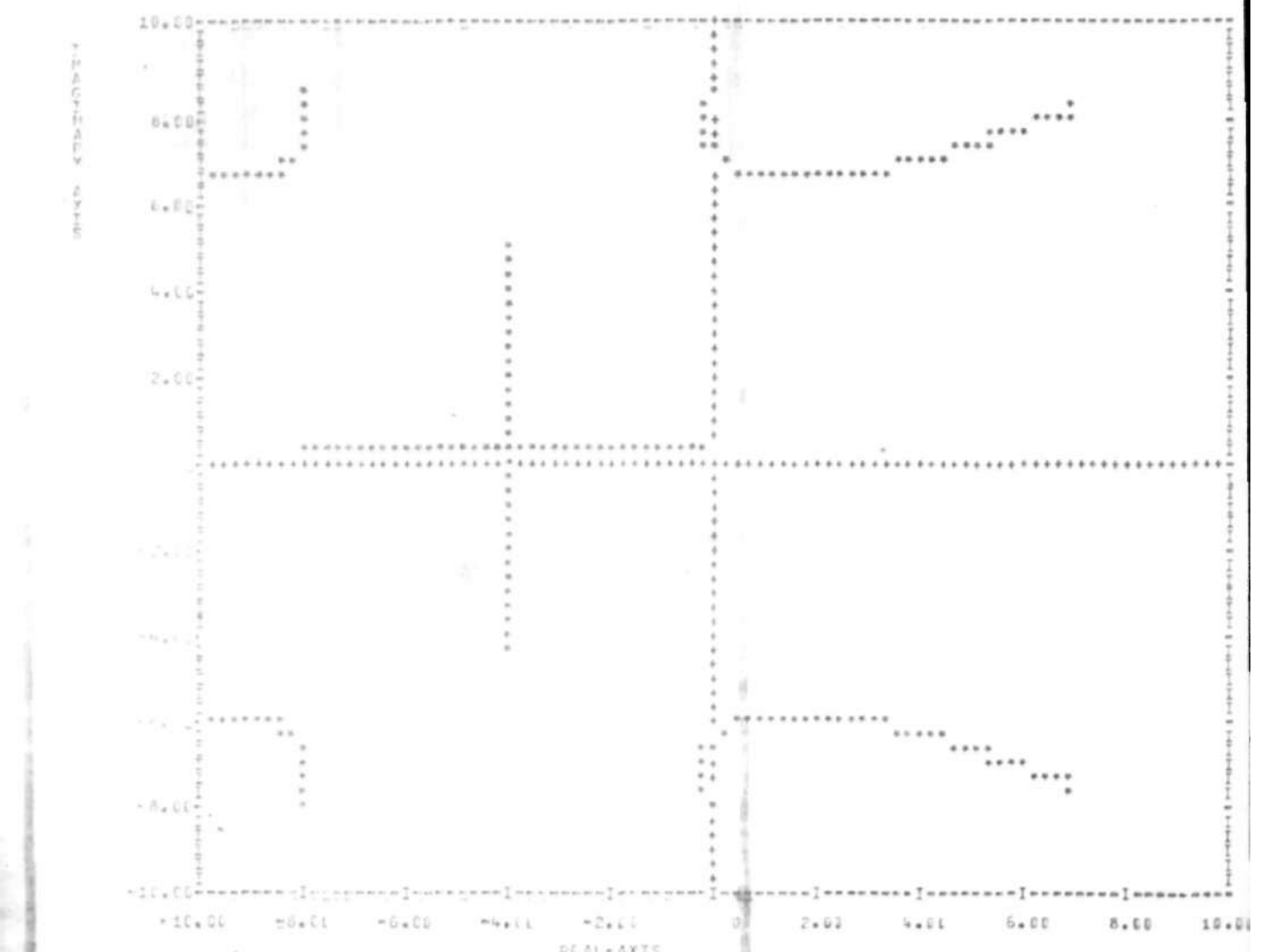



Fig.4.10: ROOT LOCI PLOT FOR SYSTEM 12

System 12: $GH(s) = \dfrac{K}{s(s+j8)(s-j8)(s+8+j8)(s+8-j8)(s+8)}$

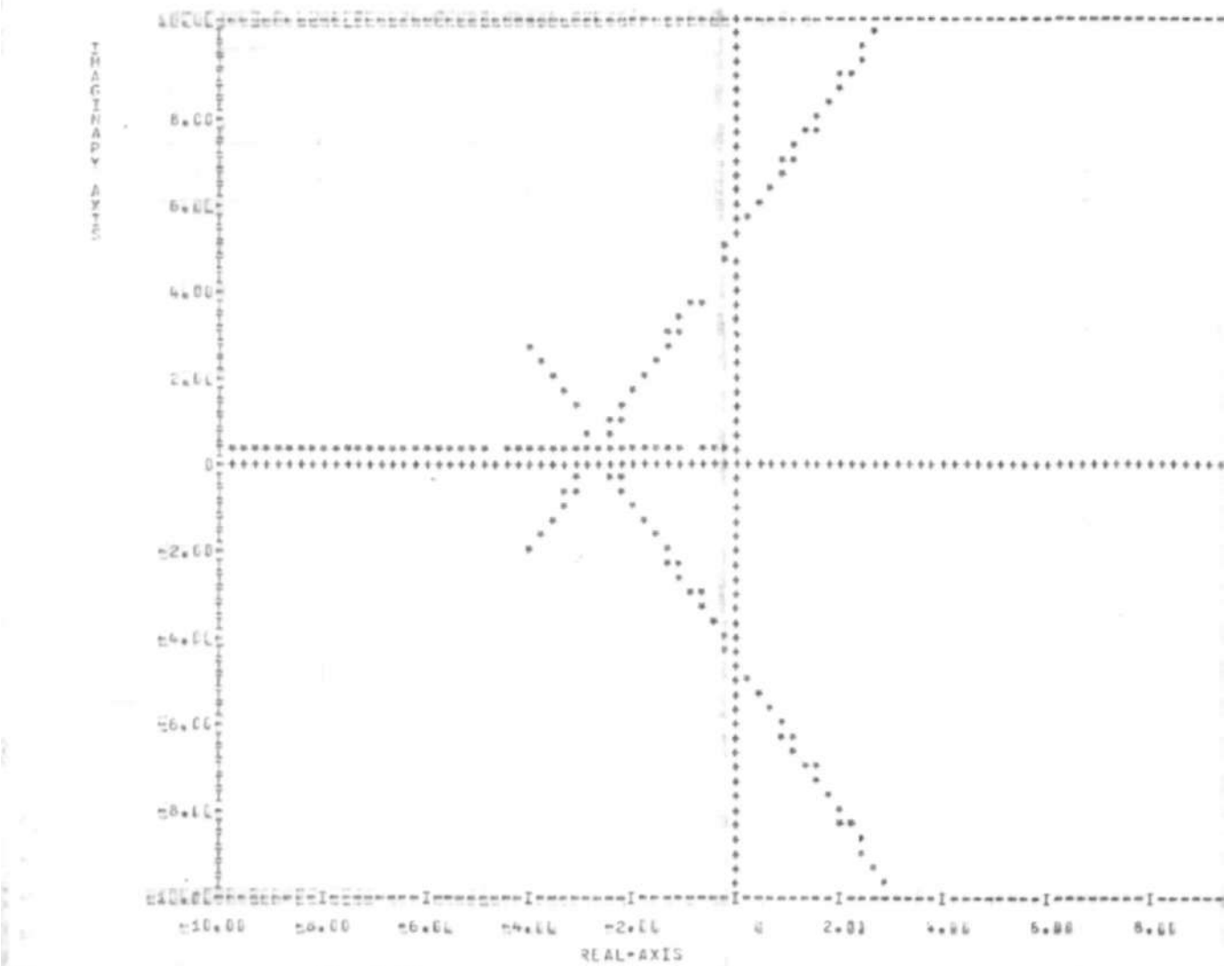



Fig. 4.11: ROOT LOCI PLOT FOR SYSTEM 31

System 31: $GH(s) = \dfrac{K}{s(s+4+j4/\sqrt{3})(s+4-j4/\sqrt{3})}$

$$\text{System 4:}\quad GH(s) = \frac{K(s+1+j2)(s+1-j2)}{s^2(s+0.5)(s+2)(s+8)^2}$$

$$\text{System 5:}\quad GH(s) = \frac{K(s+5)}{s^4}$$

$$\text{System 6:}\quad GH(s) = \frac{K(s+1.5)(s+4.5)}{s(s+1)(s+4)}$$

$$\text{System 7:}\quad GH(s) = \frac{K(s+8)}{s(s+3)(s+7)(s+15)(s+5)}$$

$$\text{System 8:}\quad GH(s) = \frac{K(s+2)(s+5)}{s(s+4)(s+1)}$$

Their root loci and that of some other systems are shown in figures 4.1 through 4.17.

## 4.3 Dependence of Root Loci Shape on Damping Ratio

It was observed that the shape of the root loci of systems with open loop transfer function given by

$$GH(s) = \frac{K}{s(s+a+jb)(s+a-jb)} \qquad \text{---- 4.1}$$

for $a, b \neq 0$, depends on the ratio $b/a$ of the imaginary part to the real part of the complex poles.

Three different root loci shapes are identified for this class of systems depending on whether

(a) $b/a > 1/\sqrt{3}$ ---- 4.2

(b) $b/a = 1/\sqrt{3}$ ---- 4.3

(c) $b/a < 1/\sqrt{3}$ ---- 4.4

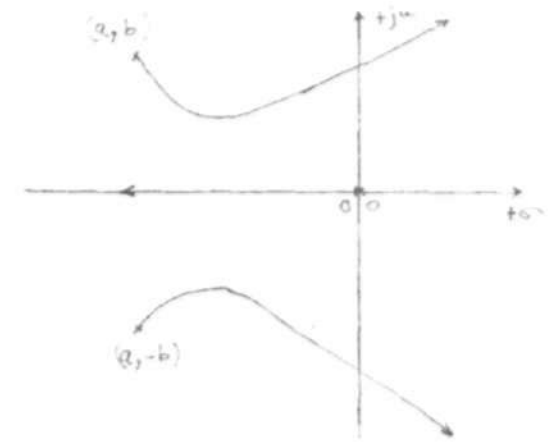



Fig. 4.12a $b/a > 1/\sqrt{3}$

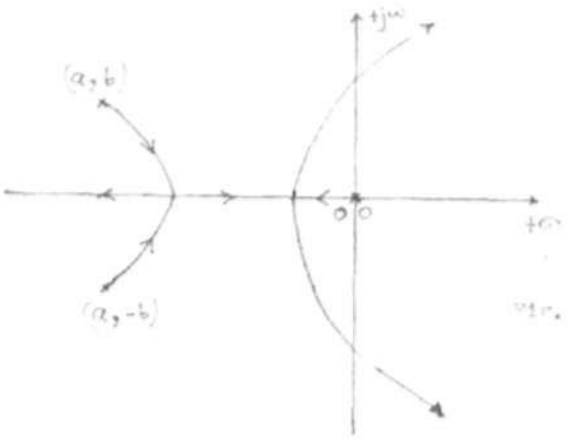



Fig. 4.12b $b/a < 1/\sqrt{3}$

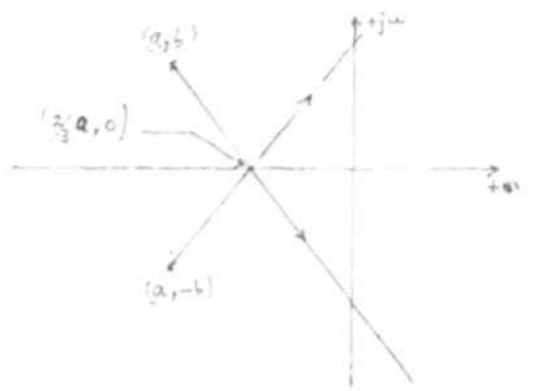



Fig. 4.12c $b/a = 1/\sqrt{3}$



1. Systems of the type and class being considered will have a root loci shape as in figure 4.12 a if one of their complex poles falls within the region A.

2. Systems of the type and class considered will have a root loci shape as in figure 4.12 b if one of their complex poles falls within region B and

3. Systems with one of their complex poles on the line

$$b = {}^{1}/\sqrt{3}\ a \qquad \text{-------} \qquad 4.8$$

will have a root loci of the type shown in figure 4.12 c.

Corollary: It is possible to estimate the damping ratio of systems of this kind from the shape of their root loci. If such a system has a root loci shape as in fig. 4.12 a, then the damping ratio, $\zeta$, lies in the range

$$0 \leq \zeta < 0.866$$

For such a system with a root loci shape as in fig. 4.12 b, the damping ratio, $\zeta$, is such that

$$0.866 < \zeta \leq 1$$

while the damping ratio, $\zeta$, can be infered directly as 0.866 if the shape of the root loci is as in figure 4.12 c.

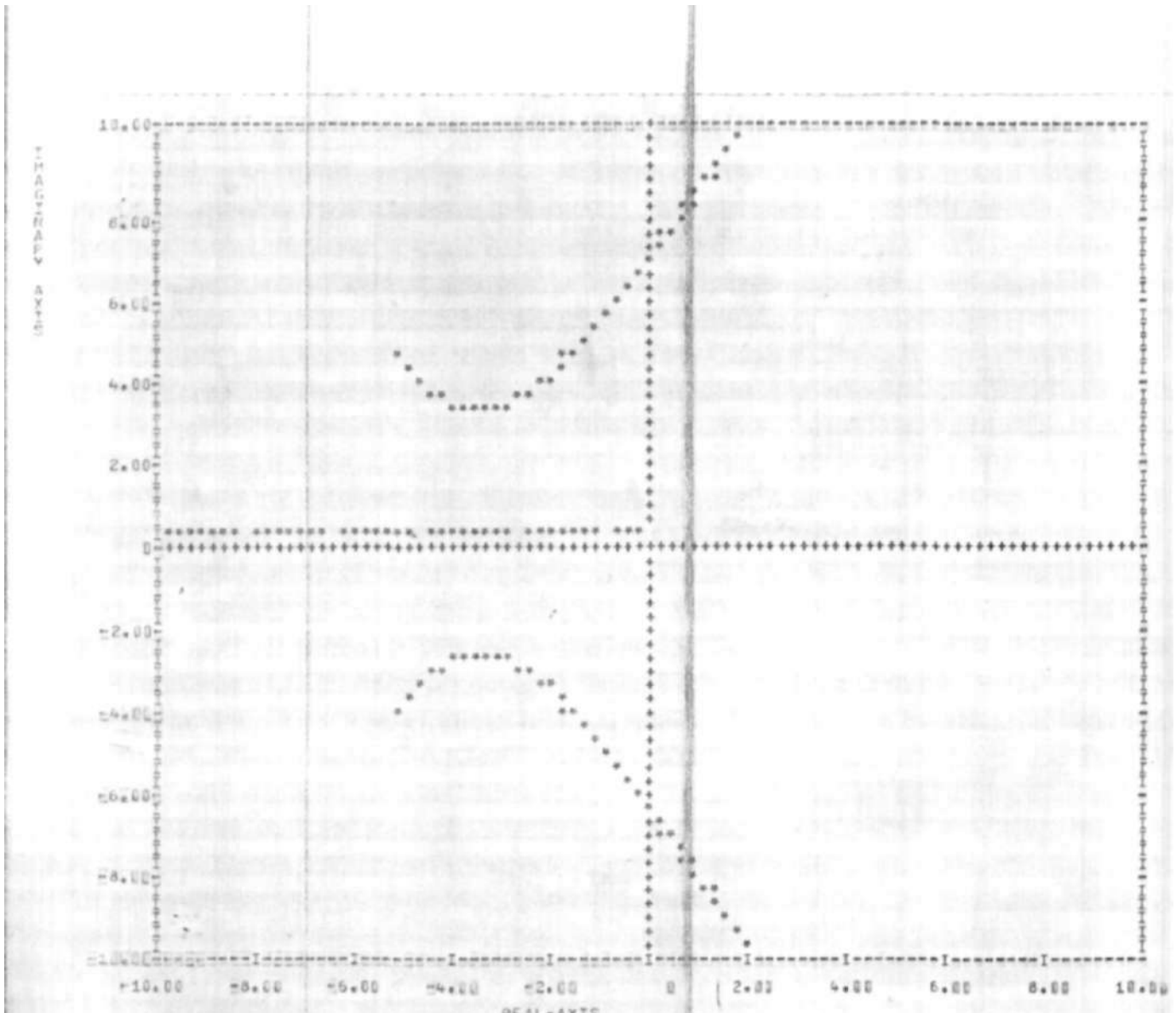



Fig. 4.13: ROOT LOCI PLOT FOR SYSTEM 91

System 91: $GH(s) = \dfrac{K}{s(s+5+j4)(s+5-j4)}$

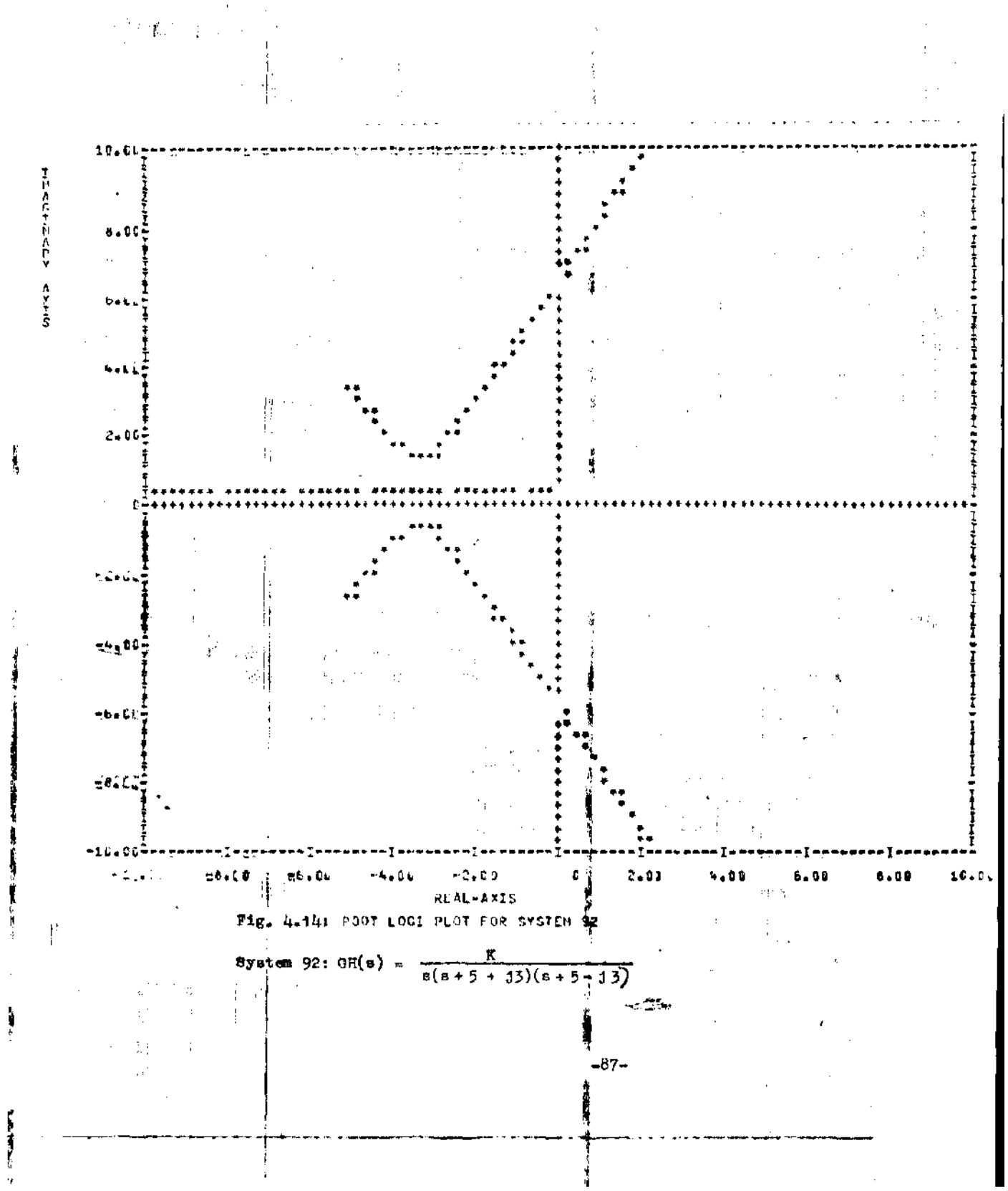



Fig. 4.14: ROOT LOCI PLOT FOR SYSTEM 92

System 92: $GH(s) = \frac{K}{s(s+5+j3)(s+5-j3)}$

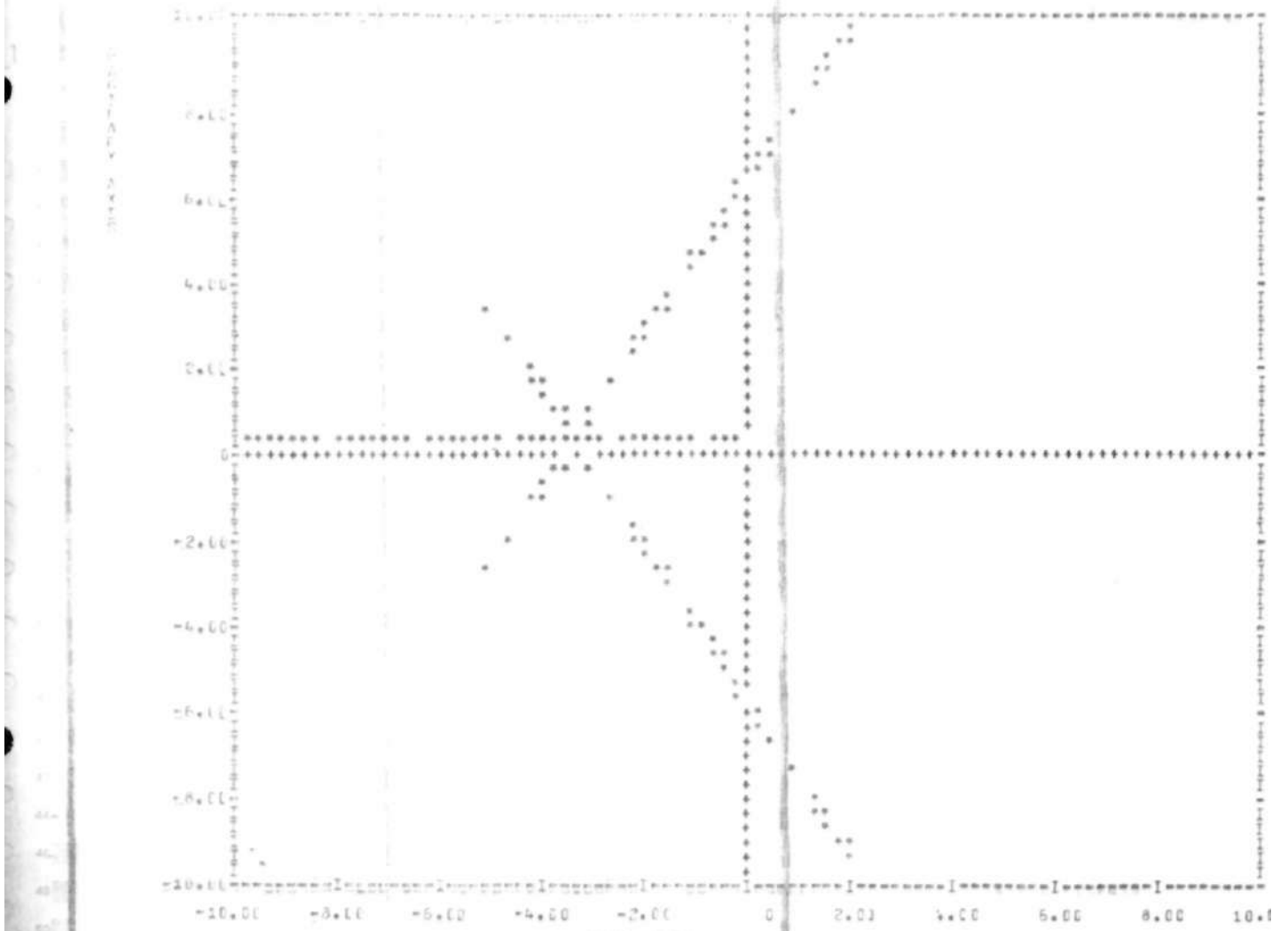



Fig 4.15: ROOT LOCI PLOT FOR SYSTEM 11

System 11: $GH(s) = \dfrac{K}{s(s+5+35/\sqrt{3})(s+5-35/\sqrt{3})}$

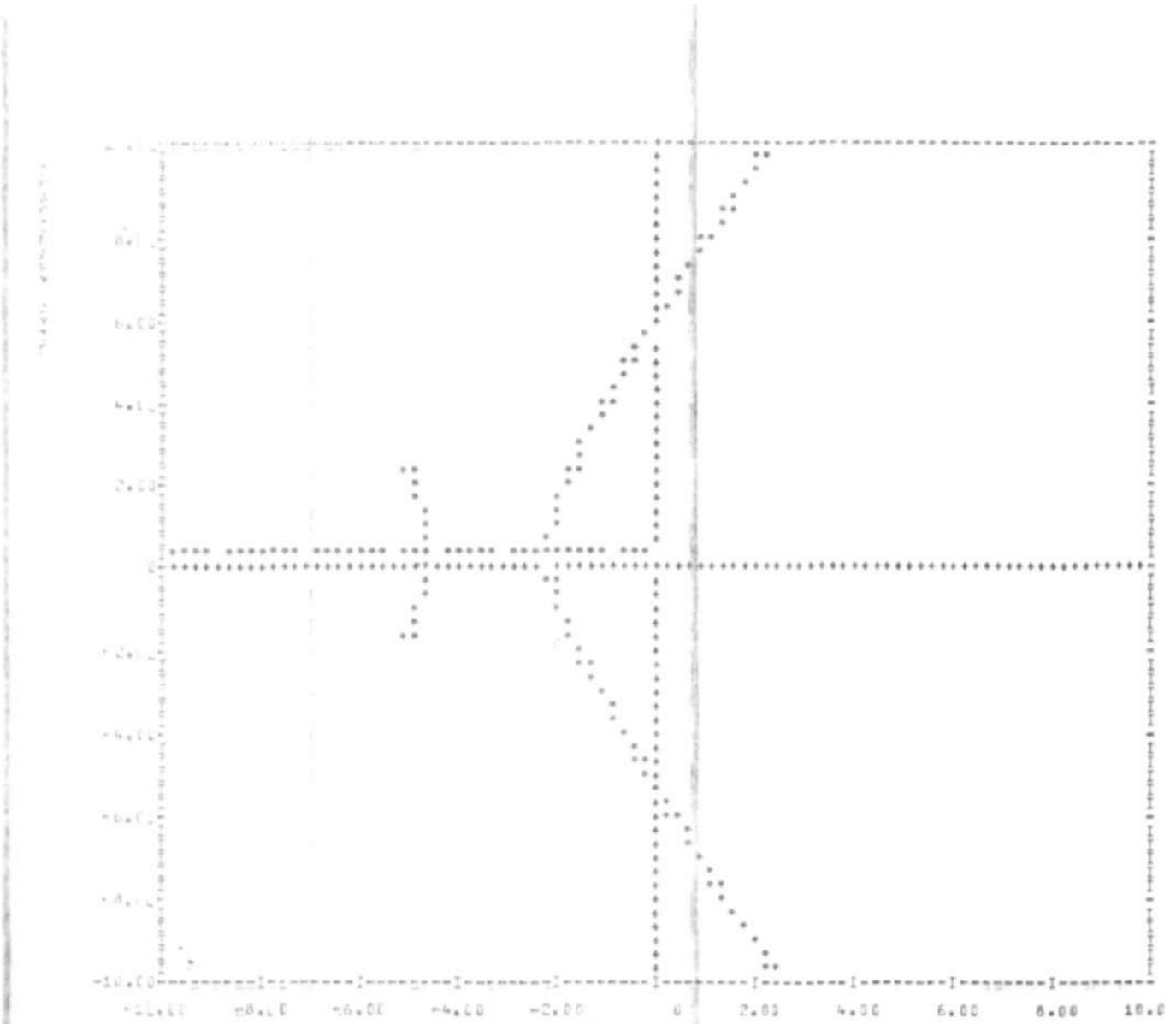



Fig. 4.16: ROOT LOCI PLOT FOR SYSTEM 9

System 9 : $GH(s) = \frac{K}{(s+5+j2)(s+5-j2)}$

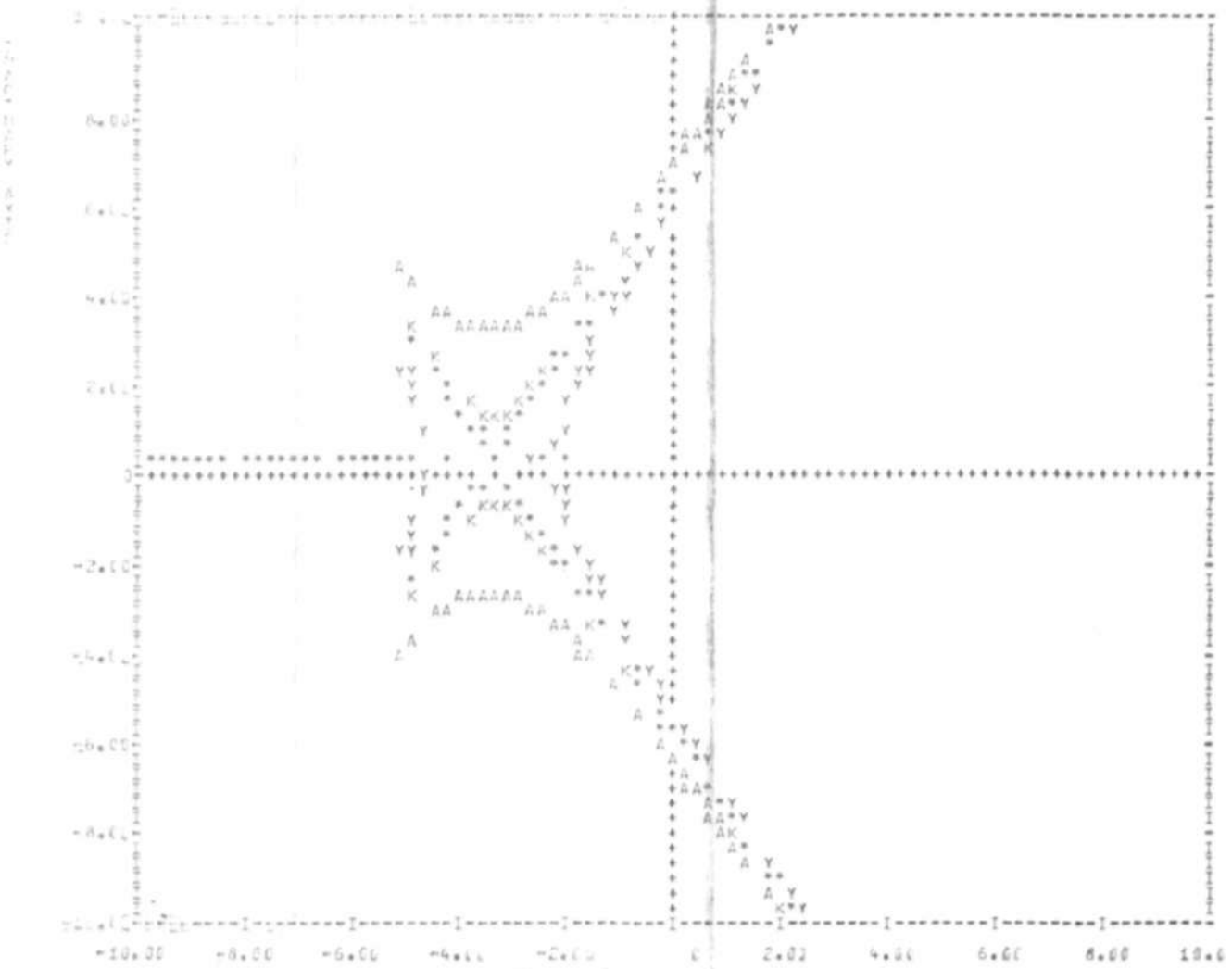



Fig. 4.17: ROOT CONTOUR SHOWING DEPENDENCE OF ROOT LOCI SHAPE ON THE RATIO B/A OF THE IMAG TO THE REAL PART OF THE COMPLEX POLES.A IS FIXED AT -5.0 WHILE B TOOK ON THE ABSOLUTE VALUES:- 4.3,2.886.2

## 4.4 Unfactored Open Loop Transfer Function

Many a times in control system design/analysis, one is faced with handling open loop transfer functions which are not factored. In this section it is intended to indicate how the root loci of such systems can be arrived at using Root98 ®.

Consider a system with

$$GH(s) = s^6 + 20\,s^5 + 13\,s^4 + 50\,s^3 + 15\,s^2 + 10s + 20 \qquad \text{------ 4.9}$$

The CE is then given by

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

$$\text{i.e.,}\quad s^6 + 20\,s^5 + 13\,s^4 + 50\,s^3 + 15\,s^2 + 10s + 21 = 0 \qquad \text{------ 4.10}$$

Some of the ways to rearrange and partition [2] the CE such that it will be possible to plot the root loci for the given system and hence deduce the characteristic roots are as follows:-

(a) $$\frac{50\,s^3 + 15\,s^2 + 10\,s + 21}{s^6 + 20\,s^5 + 13\,s^4} = -1 \qquad \text{-------- 4.11}$$

(b) $$\frac{15\,s^2 + 10s + 21}{s^6 + 20\,s^5 + 13\,s^4 + 50\,s^3} = -1 \qquad \text{-------- 4.12}$$

(c) $$\frac{10s + 21}{s^6 + 20s^5 + 13s^4 + 50s^3 + 15s^2} = -1 \qquad \text{------ 4.13}$$

A host of other arrangements are possible. The problem is tackled here using equation (4.12).

The equation can be rewritten viz:

$$\frac{15(s^2 + \frac{2}{3}s + 7/5)}{s^3(s^3 + 20s^2 + 13s + 50)} = -1. \qquad \text{------ 4.14}$$

The term

$$s^3 + 20s^2 + 13s + 50$$

in the denominator could be further partitioned to give

$$\frac{13(s + 50/13)}{s^2(s + 20)} = -1 \qquad \text{------ 4.15}$$

The solution of equation (4.15) can be effected by plotting the root loci for a system with open loop transfer function given by

$$GH(s) = \frac{K(s + 50/13)}{s^2(s + 20)} \qquad \text{------ 4.16}$$

and then determine the characteristic roots at k = 13.

Suppose the characteristic roots for the system is determined as $r_1$, $r_2$, $r_3$, then equation (4.14) can be written as:

$$\frac{15\,(s^2 + \frac{2}{3}s + 7/5)}{s^3(s + r_1)(s + r_2)(s + r_3)} = -1$$

i.e. $$\frac{15(s + \frac{1}{3} - j\ \frac{232}{45})(s + \frac{1}{3} + j\ \frac{232}{45})}{s^3\,(s + r_1)\,(s + r_2)\,(s + r_3)} = -1 \qquad \text{----- 4.17}$$

The characteristics roots of the given system $r_4$, $r_5$, $r_6$, $r_7$, $r_8$, and $r_9$ can therefore be obtained by plotting the root loci of the class systems with open loop transfer function given by:

$$GH(s) = \frac{K\,(s + \frac{1}{3} - j\ 2.271)(s + \frac{1}{3} + j\ 2.271)}{s^3(s + r_1)(s + r_2)(s + r_3)} \qquad \text{----- 4.18}$$

The values of $r_1$, $r_2$ and $r_3$ were determined as

$$r_1 = -0.269 + j\ 1.582$$

$$r_2 = -0.269 - j\ 1.582$$

and

$$r_3 = -19.463 + j\ 0$$

Hence the factored form of the transfer function is

$$GH(s) = \frac{15(s + 0.333 - j\ 2.271)(s + 0.333 + j2.271)}{s^3(s + 0.269 + j\ 1.582)(s + 0.269 - j1.582)(s + 19.463)} \qquad \text{---- 4.18}$$

The closed loop roots of the system were determined (using Root98) as

$$r_4 = -0.669 + j0$$

$$r_5 = 0.378 - j0.53164$$

$$r_6 = 0.378 + j0.53164$$

$$r_7 = -0.312 + j1.616$$

$$r_8 = -0.313 + j1.616$$

and

$$r_9 = -19.463 + j0$$

A possible root loci for the system is as shown in figure 4.18. A larger area of the s-plane has been covered in this particular plot to ensure that the six branches of the root loci are included. The two complex root loci which appear to be parallel to the imaginary axis are not strictly so. This could be realized if the plot is performed for portions of loci lying between $\pm$ 1 for the real-axis and $\pm$ 3 for the imaginary-axis. A listing of the computed points for this system is given in table 4.2. The variation in the real-axis values for these branches (4 and 5) is seen to be really small. Note that because of the triple pole at the origin, the computation for the third branch emanating from the origin cannot be started at exactly (0, 0). Hence a starting point of (0.15, 0) was used instead. The step size used for branches 4 and 5 is 0.075 while a step size of 0.125 was used for the rest of the branches.

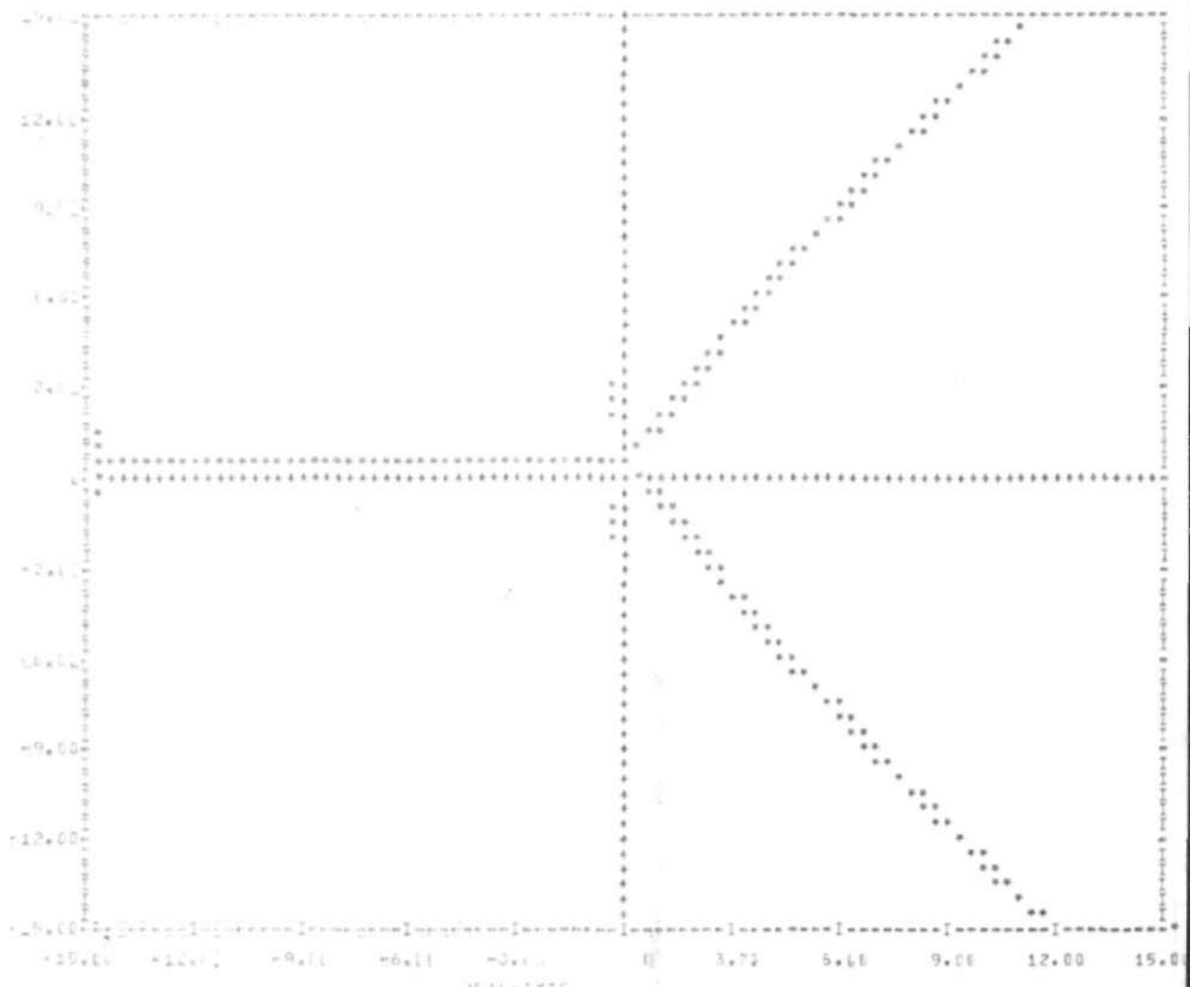



Fig.4.18: ROOT LOCI PLOT FOR SYSTEM 15

System 15: $GH(s) = \dfrac{15(s + 0.333 - j2.271)(s + 0.333 + j2.271)}{s^3(s + 0.269 + j\,1.582)(s + 0.269 - j1.582)(s + 19.463)}$

BRANCH NUMBER 1

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| -.1500 | 0 | .1316 |
| -1.4000 | 0 | 29.7736 |
| -2.6500 | 0 | 242.9196 |
| -3.9000 | 0 | 1809.9144 |
| -5.1500 | 0 | 12454.8188 |
| -6.4000 | 0 | 83271.4665 |
| -7.6500 | 0 | 534.2.2165 |
| -8.9000 | 0 | 7299.2649 |
| -10.1500 | 0 | 9604.6931 |
| -11.4000 | 0 | 11830.1327 |
| -12.6500 | 0 | 13697.1225 |
| -13.9000 | 0 | 14867.9193 |
| -14.0734 | 0 | 15073.3250 |
| -14.6751 | -1.1231 | 15621.2234 |
| -14.8034 | -1.3572 | 17565.9723 |
| -15.2864 | -3.5654 | 21195.4.75 |
| -15.6185 | -4.7421 | 20504.2345 |
| -16.1161 | -5.8842 | 34166.9774 |
| -16.6926 | -6.9926 | 44624.9134 |
| -17.3287 | -8.1737 | 98374.1589 |

BRANCH NUMBER 2

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| 0 | 0 | 0 |
| .7416 | 1.1118 | 88.1178 |
| 1.5316 | 1.9696 | 274.3186 |
| 2.2163 | 3.0147 | 1102.9866 |
| 2.9112 | 4.1537 | 2799.8539 |
| 3.6371 | 5.0713 | 5728.0254 |
| 4.3886 | 6.1710 | 11317.6936 |
| 5.1581 | 7.1551 | 17054.6064 |
| 5.9433 | 8.1278 | 26480.2417 |
| 6.7486 | 8.9965 | 39192.1862 |
| 7.5479 | 9.9448 | 55845.3114 |
| 8.3635 | 10.8921 | 77252.1383 |
| 9.1861 | 11.8338 | 103883.1356 |
| 10.0147 | 12.7691 | 136866.4493 |
| 10.8466 | 13.7013 | 175989.1632 |
| 11.6869 | 14.6276 | 225195.8848 |
| 12.5291 | 15.5512 | 282491.3122 |
| 13.3748 | 16.4717 | 349934.1831 |
| 14.2235 | 17.3895 | 428647.8106 |
| 15.1749 | 18.3047 | 519810.0194 |

BRANCH NUMBER 3

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| 0 | 0 | 0 |
| .7416 | -1.1018 | 88.1178 |
| 1.5316 | -1.9696 | 274.3186 |
| 2.2163 | -3.0147 | 1102.9866 |
| 2.9112 | -4.1537 | 2799.8539 |
| 3.6371 | -5.0713 | 5728.0254 |
| 4.3880 | -6.1710 | 11317.6936 |
| 5.1581 | -7.1551 | 17054.6064 |
| 5.9433 | -8.1278 | 26480.2417 |
| 6.7486 | -8.9965 | 39192.1862 |
| 7.5479 | -9.9448 | 55845.3114 |
| 8.3635 | -10.8921 | 77252.1383 |
| 9.1861 | -11.8338 | 103883.1356 |
| 10.0147 | -12.7691 | 136866.4493 |
| 10.8466 | -13.7013 | 175989.1632 |
| 11.6869 | -14.6276 | 225195.8848 |
| 12.5291 | -15.5512 | 282491.3122 |
| 13.3748 | -16.4717 | 349934.1831 |
| 14.2235 | -17.3895 | 428647.8106 |
| 15.1749 | -18.3047 | 519810.0194 |

BRANCH NUMBER 4

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| -.1500 | 0 | .1316 |
| -.3426 | -1.6487 | 12.0562 |
| -.3926 | -1.735. | 32.5789 |
| -.4249 | -1.8296 | 68.4638 |
| -.4419 | -1.9281 | 133.0573 |
| -.4431 | -2.1279 | 257.8836 |
| -.4252 | -2.1261 | 543.6667 |
| -.3832 | -2.2165 | 1109.1158 |
| -.3441 | -2.2617 | 9250.2297 |
| -.3441 | -2.2617 | 9250.2297 |
| -.3441 | -2.2617 | 9250.2297 |
| -.3441 | -2.2617 | 9250.2297 |
| -.3441 | -2.2617 | 9250.2297 |
| -.3441 | -2.2617 | 9250.2297 |
| -.3441 | -2.2617 | 9250.2297 |
| -.3441 | -2.2617 | 9250.2297 |
| -.3441 | -2.2617 | 9250.2297 |
| -.3441 | -2.2617 | 9250.2297 |

BRANCH NUMBER 5

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| -.1500 | 1.5812 | 1 |
| -.3426 | 1.6487 | 12.0562 |
| -.3926 | 1.7350 | 32.5789 |
| -.4249 | 1.8296 | 68.4638 |
| -.4419 | 1.9280 | 133.0573 |
| -.4431 | 2.1279 | 257.8836 |
| -.4252 | 2.1261 | 543.6667 |
| -.3832 | 2.2165 | 1109.1158 |
| -.3441 | 2.2617 | 9250.2297 |
| -.3441 | 2.2617 | 9250.2297 |
| -.3441 | 2.2617 | 9250.2297 |
| -.3441 | 2.2617 | 9250.2297 |
| -.3441 | 2.2617 | 9250.2297 |
| -.3441 | 2.2617 | 9250.2297 |
| -.3441 | 2.2617 | 9250.2297 |
| -.3441 | 2.2617 | 9250.2297 |
| -.3441 | 2.2617 | 9250.2297 |
| -.3441 | 2.2617 | 9250.2297 |

BRANCH NUMBER 6

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| -13.4030 | 0 | 0 |
| -13.8212 | 0 | 7543.4845 |
| -14.9631 | 0 | 12179.8673 |
| -15.7136 | 0 | 14567.2289 |
| -14.5567 | 0 | 15073.0130 |
| -14.6529 | .8741 | 154.4.432. |
| -14.8198 | 2.1122 | 17157.7494 |
| -15.1139 | 3.3266 | 2.256.1324 |
| -15.5107 | 4.5096 | 29254.5000 |
| -16.0086 | 5.6584 | 32446.5854 |
| -16.5716 | 6.7742 | 42291.9.3. |
| -17.1912 | 7.8508 | 55330.6627 |
| -17.8548 | 8.9196 | 72177.3729 |
| -18.5534 | 9.9555 | 93512.2188 |
| -19.2794 | 11.0727 | 120077.5672 |
| -20.1287 | 12.0736 | 152677.4298 |
| -21.7957 | 12.9614 | 192176.6.85 |
| -21.5775 | 13.9359 | 239499.6395 |
| -22.3727 | 14.9019 | 295630.4644 |
| -23.1781 | 15.8579 | 361612.1661 |

Table 4.2: List of Computed Points for System 15

# CHAPTER V

## APPLICATION OF ROOT98 TO SERIES COMPENSATION

### 5.1 Introduction

It seldom happens that the basic components of a control system perform according to specifications (i.e. system design specifications are seldom met) in terms of accuracy and transient response without additional design and/or adjustment. The measures taken to ensure that system performance conforms to the accuracy and/or stability specifications are what have come to be referred to (in classical control theory) as compensation.

The compensation design for a feedback control system could be achieved either by a series (cascade) compensation or a parallel (feedback) compensation. In series compensation, the compensating device (compensator) is inserted in the system in tandem with the forward path while in feedback compensation it is inserted along a feedback path. These conditions are illustrated in figures 5.1a, and 5.1b.

Feedback compensation is generally achieved by the introduction of rate or acceleration feedback signals. On the other hand, series compensation is accomplished by the insertion of a phase-lead, phase-lag or phase-lag-lead device. These may be realized electrically either as

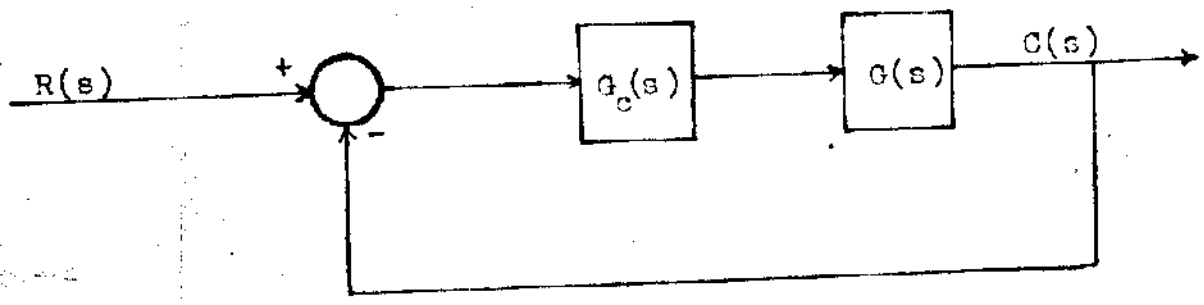



Fig. 5.1a : Series Compensation

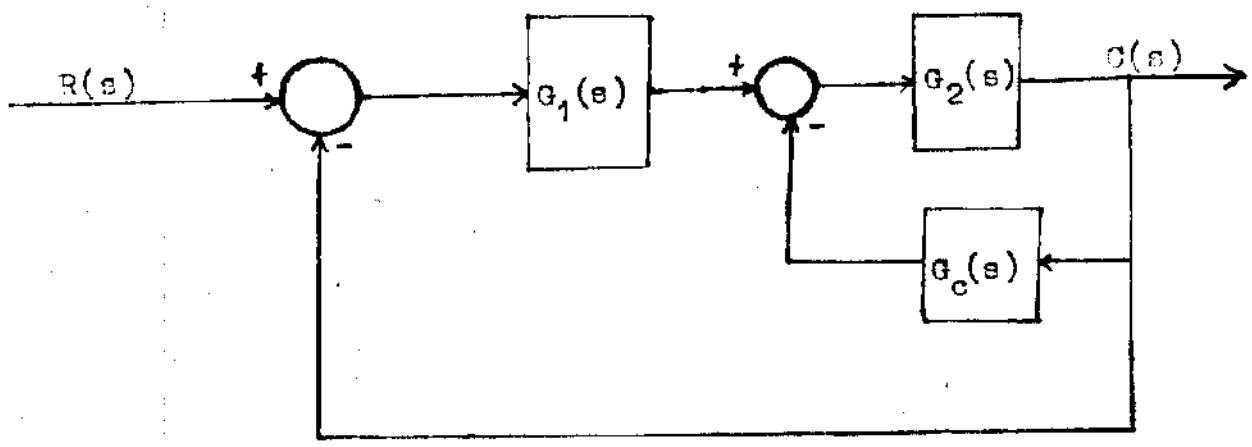



Fig 5.1b: Parallel Compensation

R C - filter circuits which introduce the desired poles and zeros in the system or as active-filter circuits that exhibit the desired response.

Figures 5.2 a and 5.2 b show standard R C lead-and lag-networks where the transfer function of the lead-network is

$$\frac{E_o}{E_i}(s) = \frac{s + 1/CR}{s + (R+R_1)/CR\ R_1} \quad \text{------} \quad 5.1$$

and that of the lag-network is

$$\frac{E_o}{E_i}(s) = \frac{R_1}{R + R_1} \cdot \frac{s + 1/C_1\ R_1}{s + 1/C_1(R+R_1)} \quad \text{------} \quad 5.2$$

Equations (5.1) and (5.2) are in the root locus form and they could be put in the Bode [2] form respectively, as

$$\frac{E_o}{E_1}(s) = \frac{R_1}{R + R_1} \cdot \frac{s\,CR + 1}{sCR\ R_1/(R+R_1) + 1} \quad \text{------} \quad 5.3$$

and

$$\frac{E_o}{E_1}(s) = \frac{s\ C_1R_1 + 1}{s\ C_1\ (R + R_1) + 1} \quad \text{------} \quad 5.4$$

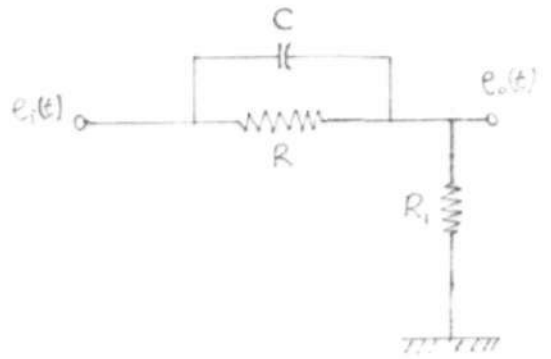



Fig. 5.[illegible]: Phase lead circuit

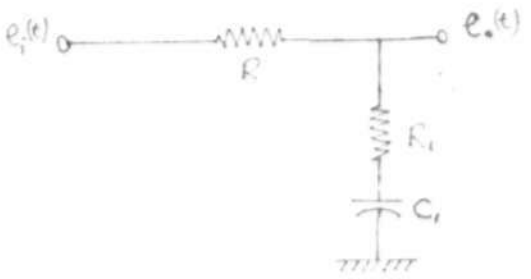



Fig. [illegible]: Phase lag circuit

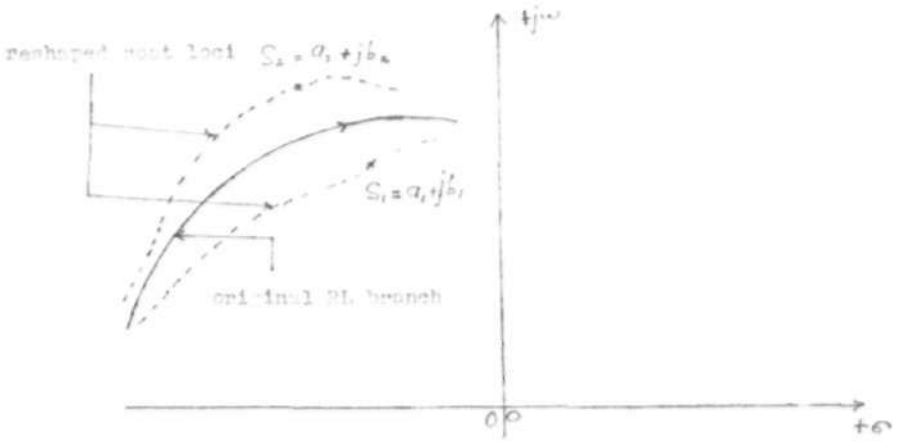



Fig. 5.3 : Reshaping root locl

Due to these two alternative forms of representation available for the transfer function of these networks, two methods exist for parameterizing the realization of a series compensation design using the root locus technique. The objective in this chapter is to consider the problem of designing a suitable series compensator for the plant

$$G(s) = \frac{K}{s(s+1)(s+3)} \qquad \text{------- 5.5}$$

using these root locus procedures for a root locus gain, K, value of 100. This exercise is instructive and provides an insi into how to implement design using the root loci technique.

## 5.2 Series Compensator Design Using Root Locus Method

### 5.2.1 Design Principles

The design of compensators using the root locus method is based on the fact that the root locus of a system can be reshaped[2] and, in so doing, forced to pass through any desired point in the complex frequency plane. This root locus reshaping technique is realized by introducing relevant control circuit elements (filters etc) and can be illustrated using figure 5.3. This figure represents a general root locus branch.

In chapter I, the root locus of a system was defined as the set of all points in the s-plane which satisfy the

ZARIA, NIGERIA

C E of the given system. That is the set of points for which

$$\text{Arg}\left\{F(s)\right\} - n\overline{\pi} = 0 \qquad \text{------} \qquad 5.6$$

where $n = \pm 1, \pm 3, \ldots$ , for regular root loci points, and

$$F(s) = \frac{GH(s)}{K} \qquad \text{-------} \qquad 5.7$$

is the gain- normalized open loop transfer function. Hence relative to the general root locus shown in figure 5.3, points in the s-plane such as

$$s_1 = a_1 + j\,b_1 \qquad \text{-------} \qquad 5.8$$

satisfy the inequality

$$\text{Arg}\left\{F(s_1)\right\} - n\overline{\pi} < 0 \qquad \text{-------} \qquad 5.9$$

while points such as

$$s_2 = a_2 + jb_2 \qquad \text{-------} \qquad 5.10$$

satisfy the inequality

$$\text{Arg}\left\{F(s_2)\right\} - \overline{\pi} > 0 \qquad \text{-------} \qquad 5.11$$

The reverse is the case if these points where in the lower half of the s-plane. This is so because of the symetric property of complex root loci branches.

If it is desired that the root locus of the general system shown passes through the point, $s_1$, then a

compensator will be added in series to the original systems such that

$$\text{Arg}\left\{G_c(s_1)\, G(s_1)\right\} - n\overrightarrow{11} = \text{Arg}\left\{\quad (s_1)\right\} - n\overrightarrow{11} + \underline{\triangle}\theta_1 \quad = 0$$

---- 5.12

$$\triangle\theta_1 \;=\; \text{Arg}\left\{G_c(s_1)\right\} \qquad \text{---- 5.13}$$

is the phase angle difference introduced by the compensator with a transfer function, $G_c(s)$.

Similarly a series compensator could be used to reshape the root locus of the original system to make it pass through the point, $s_2$. In this case,

$$\text{Arg}\left\{G_c(s_2)\, G(s_2)\right\} \;-\; n\overrightarrow{11} \;=\; \text{Arg}\left\{G(s_2)\right\} \;-\; n\overrightarrow{\quad} \;+\; \underline{\triangle}\theta_2 \;=\; 0$$

----- 5.14

Either of $\triangle\theta_1$ and $\underline{\triangle}\theta_2$ could be + ve or - ve depending on the value of n ( i.e. whether the original locus is a $-180^\circ$ locus or a $+180^\circ$ locus for instance).

If $\underline{\triangle}\theta_1 < 0$

then the compensating device has introduced a lagging angle and the compensator or compensation is the lag type. Otherwise the compensation is the lead type.

Similarly, if $\triangle\theta_2 < 0$, compensation is the lag type while the compensation is the lead type for + ve $\triangle\theta_2$.

However, if

$$\underline{\Delta}\theta_2 < 0 ,$$

then $\underline{\Delta}\theta_1 > 0$

and vice versa.

The introduction of a single pole in series with the forward path transfer function of a system produces the general effect of lag-compensation while that of a zero has the effect of a lead-compensation. Control elements introducing single poles and zeros are however, seldom used for compensation because with such elements both the root locus gain and static gain (hence accuracy) considerations cannot be met simultaneously.

It is possible to specify that a compensator be designed to meet some criteria on system parameters such as

(i) $M_p$ - the peak response of the closed loop frequency.

(ii) $\zeta$ - the damping ratio of the complex roots of the system.

(iii) $K_x$ - the static gain of the system and so on and so forth.

Under such circumstances, the inter-relationship between these system parameters is employed to determine actually the relevant point or points in the complex-plane. through which

the given specs would want the compensated root loci to pass As indicated earlier on, the compensator design for the system given in equation(5.5)in which restricti is placed only on the root locus gain is to serve as an introduction to the concept of parameterizing the root locus compensation design.

5.2.2. Compensator Transfer Function in Bode Form

The transfer functions of equations (5.3) and (5.4) can be parameterized respectively as follows:

Let

$$T = \frac{RR_1}{R + R_1} \cdot C \qquad \text{--------} \qquad 5.15$$

and

$$a = \frac{R + R_1}{R_1} \qquad \text{---------} \qquad 5.16$$

Then (5.3) for lead compensation becomes

$$\frac{E_o}{E_1}(s) = \frac{1}{a}\left(\frac{1 + aTs}{1 + Ts}\right) \qquad \text{--------} \qquad 5.17$$

Whereas if

$$T = C_1R_1 \qquad \text{--------} \qquad 5.18$$

and

$$a = \frac{R_1 + R}{R_1} \qquad \text{--------} \qquad 5.19$$

Equation (5.4) for lag compensation takes the form

$$\frac{E_o}{E_i}(s) = \frac{1 + Ts}{1 + aTs} \qquad \text{------} \qquad 5.20$$

Since $a > 1$ in both equations, the lead compensator introduces a net +ve phase and the lag compensator introduces a net negative phase into any given system. It is possible then to combine the equation for both lag and lead networks into one viz:

$$G_c(s) = \frac{1 + aTs}{1 + Ts} \qquad \text{-------} \qquad 5.21$$

where the network becomes the lag compensator if $a < 1$ and the lead compensator if $a > 1$.

For the given plant with CE:

$$G(s) = \frac{100}{s(s+1)(s+3)} = -1 \qquad \text{------} \qquad 5.22$$

the compensated system transfer function is given by

$$G_c(s)\, G(s) = \frac{100\,(1 + aTs)}{s(s+1)(s+3)(s+Ts)} \qquad \text{-----} \qquad 5.23$$

The root locus of the plant computed and plotted by Root9e is shown in figure 5.4. The three roots for the given K values are $r_1 = 1.078 + j3.866$, $r_2 = 1.078 - j3.866$, $r_3 = -6.154 + j0$

The CE of the compensated system is

$$\frac{100\,(1 + aTs)}{s(s+1)(s+3)(1+Ts)} = -1 \qquad \text{-------} \qquad 5.24$$

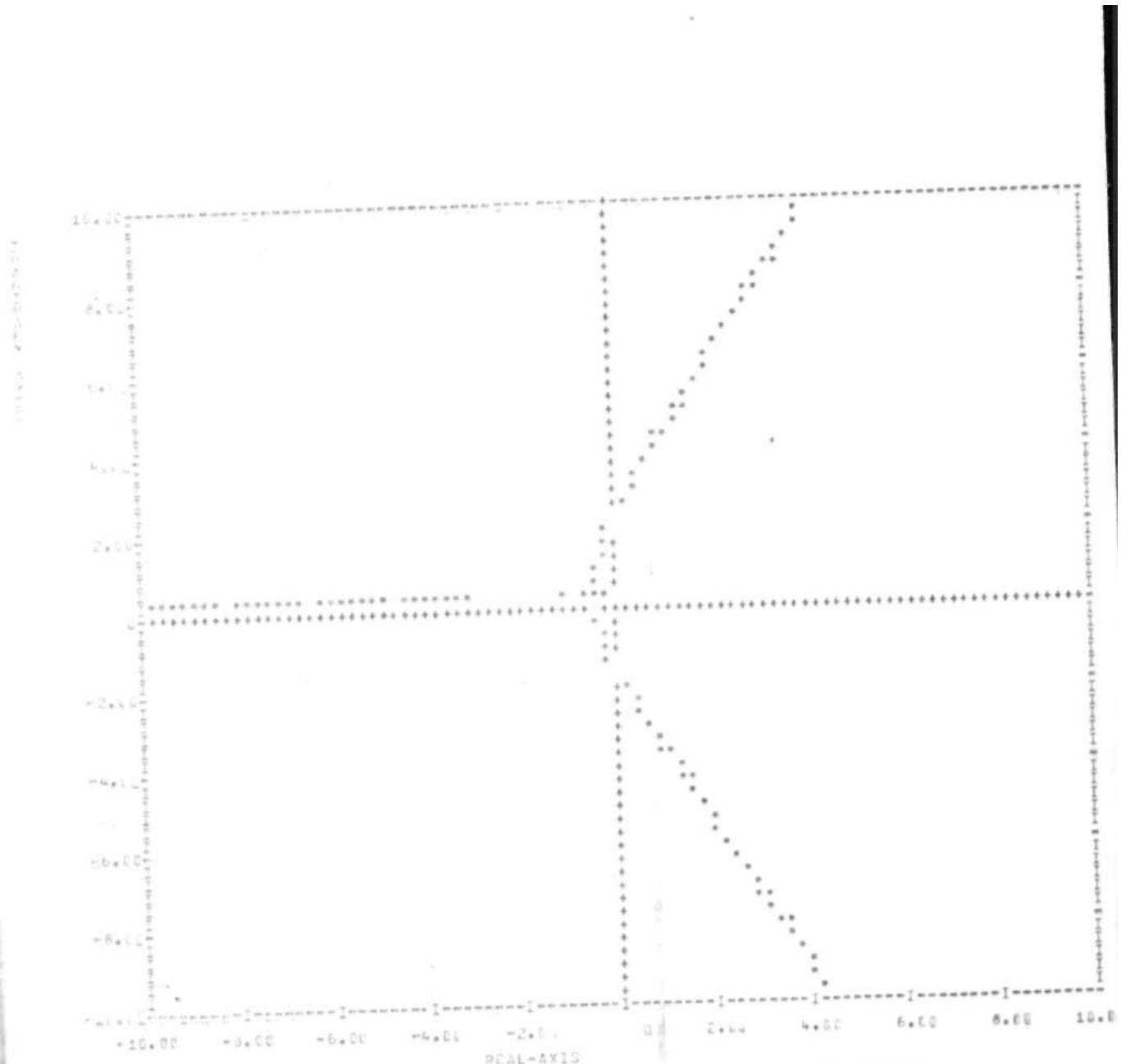



Fig. 5.4 : REGULAR ROOT LOCI PLOT FOR THE UNCOMPENSATED SYSTEM

$$G(s) = \frac{K}{s(s+1)(s+3)}$$

For $a = 0$, this reduces to

$$\frac{100}{s(s+1)(s+3)(1+Ts)} = -1 \qquad \text{------ 5.25}$$

This equation can be rearranged viz:

$$\frac{Ts^2(s+1)(s+3)}{s(s+1)(s+3)+100} = -1 \qquad \text{------ 5.26}$$

The denominator of (5.26) can be equated to zero to obtain

$$s(s+1)(s+3)+100 = 0$$

$$\text{i.e.} \quad \frac{100}{s(s+1)(s+3)} = -1 \qquad \text{------ 5.27}$$

Since (5.27) is the CE of the compensated system which has the roots $r_1$, $r_2$ and $r_3$ given above, (5.26) can then be written as:

$$\frac{Ts^2(s+1)(s+3)}{(s+r_1)(s+r_2)(s+r_3)} = -1$$

$$\text{i.e.} \quad \frac{Ts^2(s+1)(s+3)}{(s-1.078-j3.886)(s-1.078+j3.886)(s+6.154)} = -1 \qquad \text{------- 5.28}$$

Hence the modified root locus of the compensated system for $a = 0$ and with $T$ $(0 \leq T \leq \infty)$ as the root locus parameter is as shown in figure 5.5.

It is possible to invert equation (5.28) and obtain,

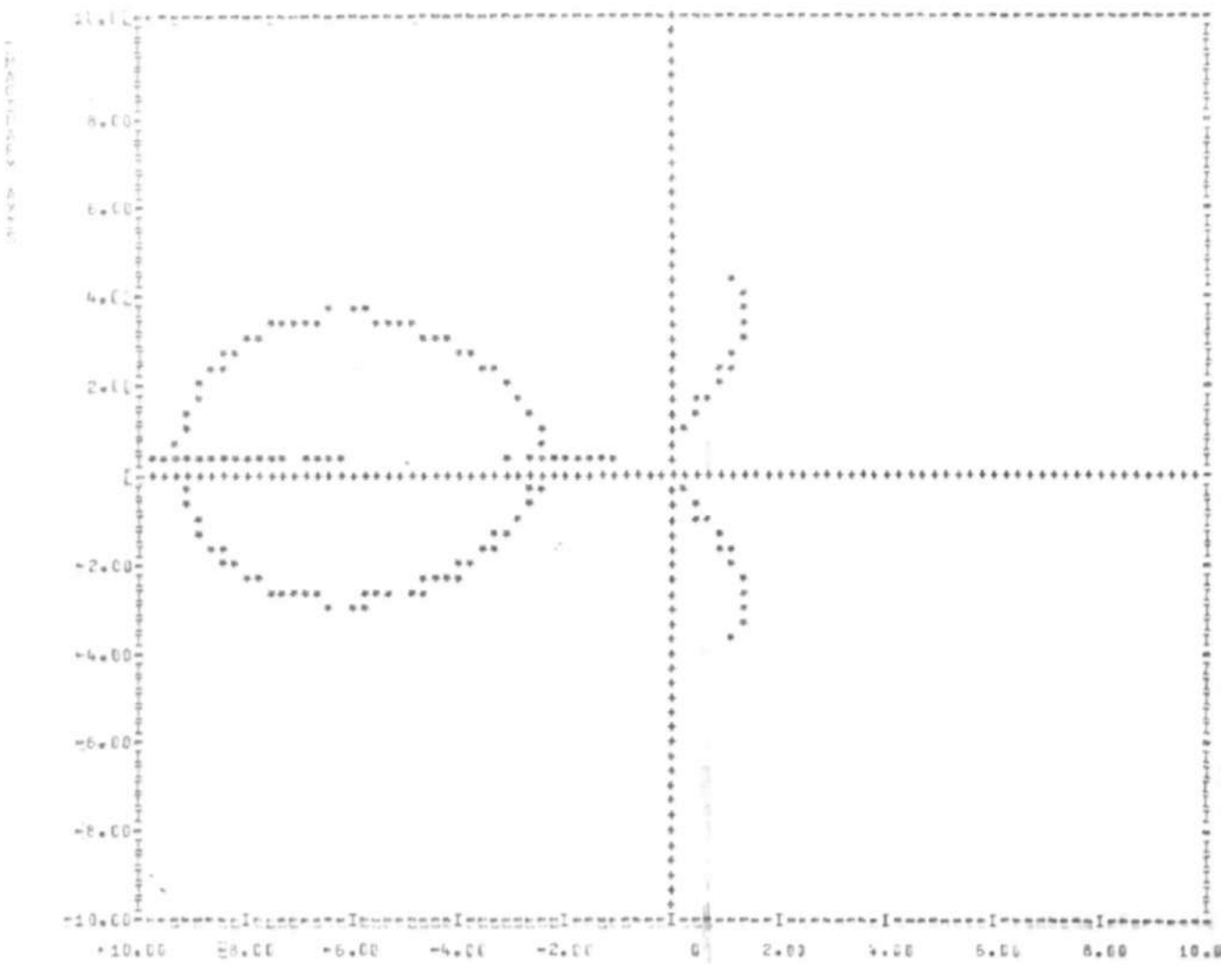



Fig. 5.5 : ROOT LOCI PLOT FOR VARIABLE PARAMETER T WITH A=0

$$1/T \; \frac{(s+6.154)(s-1.078+j3.886)(s-1.078+j3.886)}{s^2(s+1)(s+3)} = -1 \qquad \text{-------} \quad 5.29$$

The modified root loci is once more shown for $0 \leq 1/T \leq \infty$ in figure 5.5. Both loci are same save for the fact that the poles and zeros have been interchanged and either could be used to realize the design.

If a, $T \neq 0$, then the C E of the compensated system can be rearranged as

$$\frac{100\,aTs}{s(s+1)(s+3)(1+Ts)+100} = -1 \qquad \text{-------} \quad 5.30$$

Since the C E of the compensated system for $a = 0$, is the same as the denominator of equation (5.30), the root contour of the compensated system can be obtained by chosing fixed values of T to determine the roots $r_4$, $r_5$, $r_6$ and $r_7$ which can then be used in the equation

$$\frac{100\,aTs}{(s+r_4)(s+r_5)(s+r_6)(s+r_7)} = -1 \qquad \text{-------} \quad 5.31$$

The different values of T chosen and the corresponding values of the roots are as tabulated below (Table 5.1). The root loci for the various values of T are as shown in figures 5.6 through 5.8. Figure 5.9 shows the root contour for these values of T.

| Parameter, T | Real Part | Imaginary Part | Root Locus Gain, K | Root |
|---|---|---|---|---|
| 0.193 | 1.369 | -3.114 | 100 | $r_4$ |
| " | 1.369 | 3.114 | " | $r_5$ |
| " | -5.968 | 3.041 | " | $r_6$ |
| " | -5.968 | -3.041 | " | $r_7$ |

| Parameter, T | Real Part | Imaginary Part | Root Locus Gain, K | Root |
|---|---|---|---|---|
| 0.052 | 1.144 | -3.815 | 100 | $r_4$ |
| " | 1.144 | 3.815 | " | $r_5$ |
| " | -4.853 | 0.0 | " | $r_6$ |
| " | -19.05 | 0.0 | " | $r_7$ |

| Parameter, T | Real Part | Imaginary Part | Root Locus Gain, K | Root |
|---|---|---|---|---|
| 0.472 | -1.286 | -2.573 | 100 | $r_4$ |
| " | -1.286 | 2.573 | " | $r_5$ |
| " | -4.348 | 2.582 | " | $r_6$ |
| " | -4.348 | 2.582 | " | $r_7$ |

Table 5.1 showing values of T and the roots

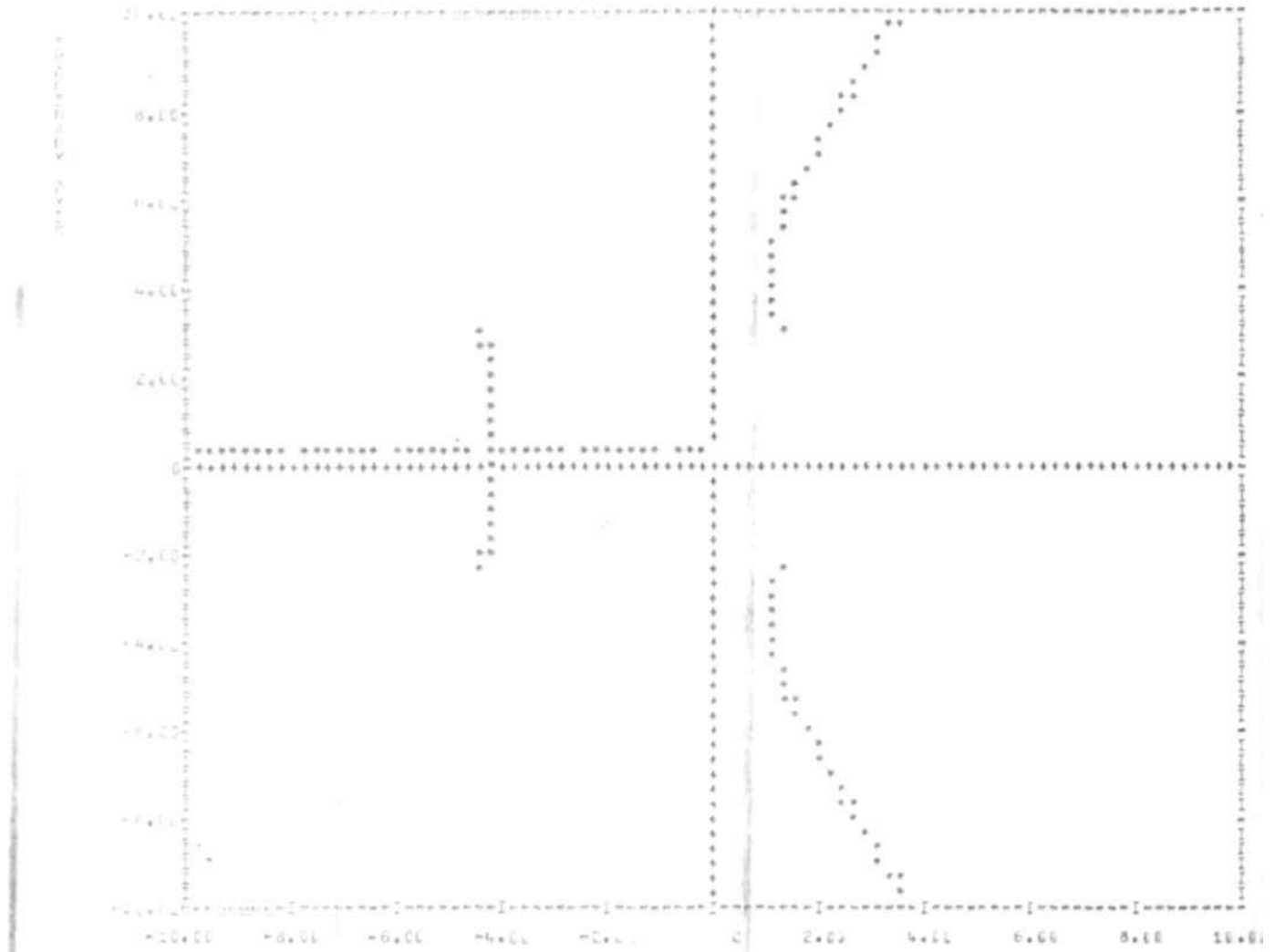



Fig. 5.6: ROOT LOCI PLOT FOR T=1.472 WITH A AS THE VARIABLE PARAMETER

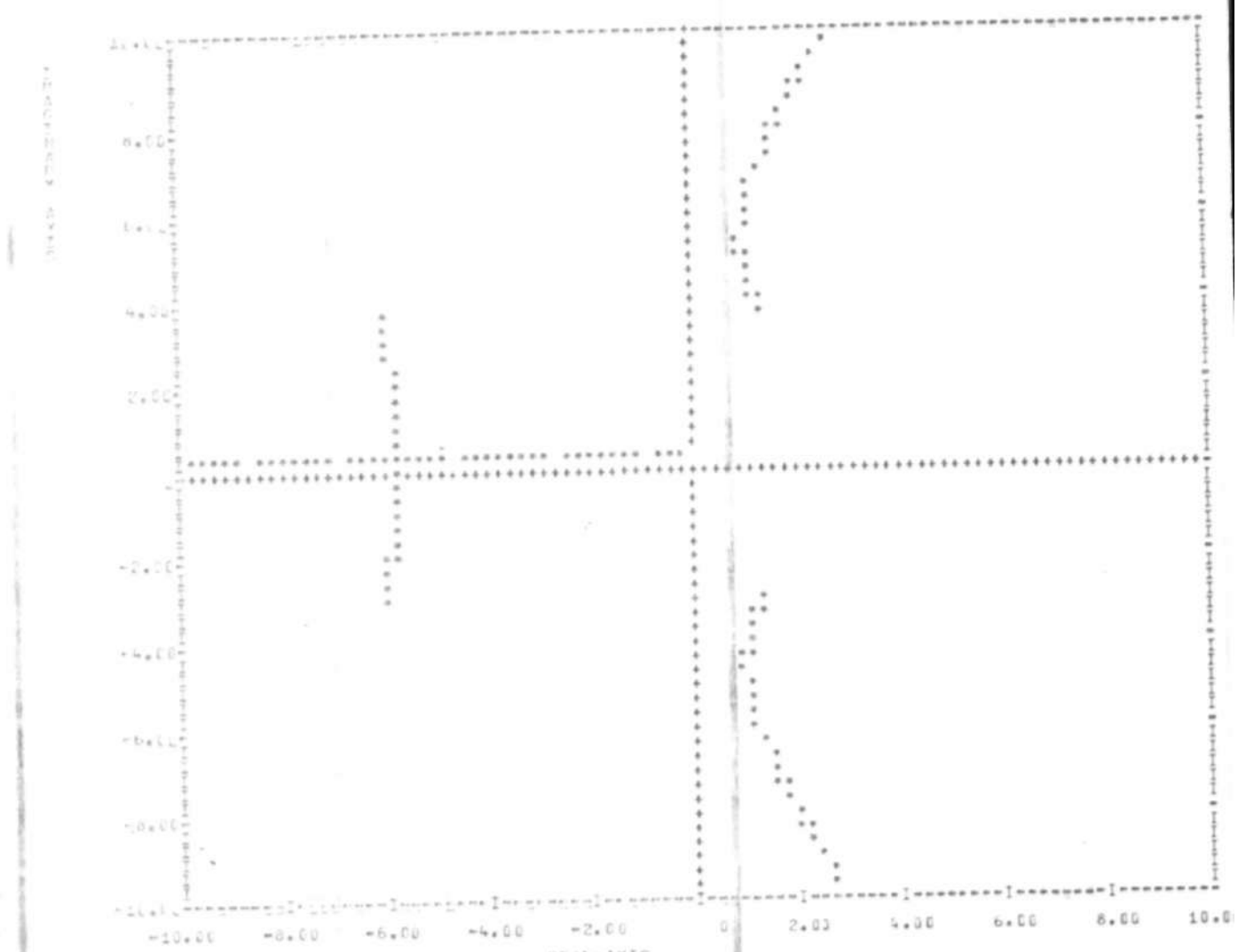



Fig 5.7: ROOT LOCI PLOT FOR T=0.193 WITH A AS THE VARIABLE PARAMETER

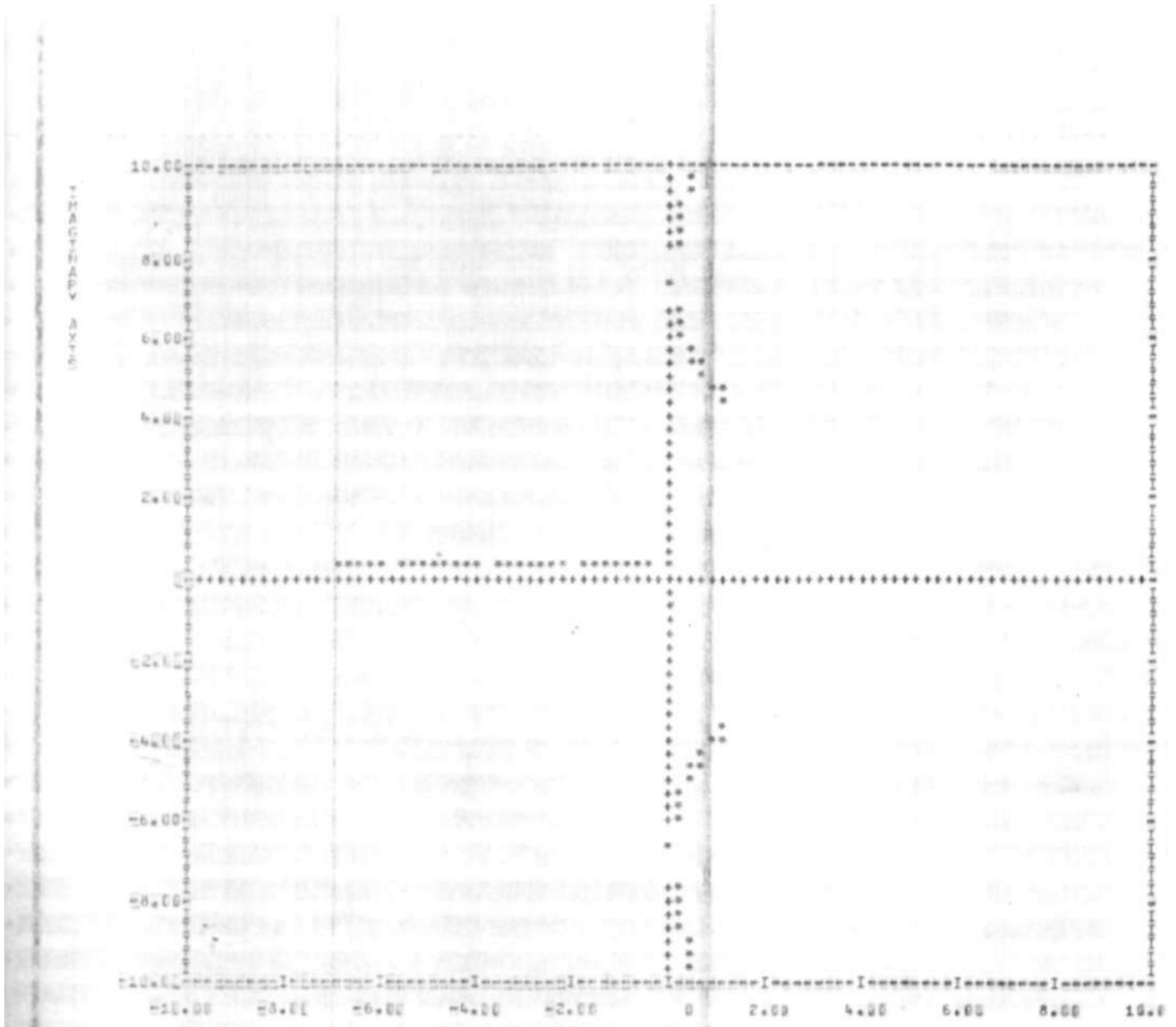



Fig. 5.8: ROOT LOCI PLOT FOR T=0.152 WITH A AS THE VARIABLE PARAMETER

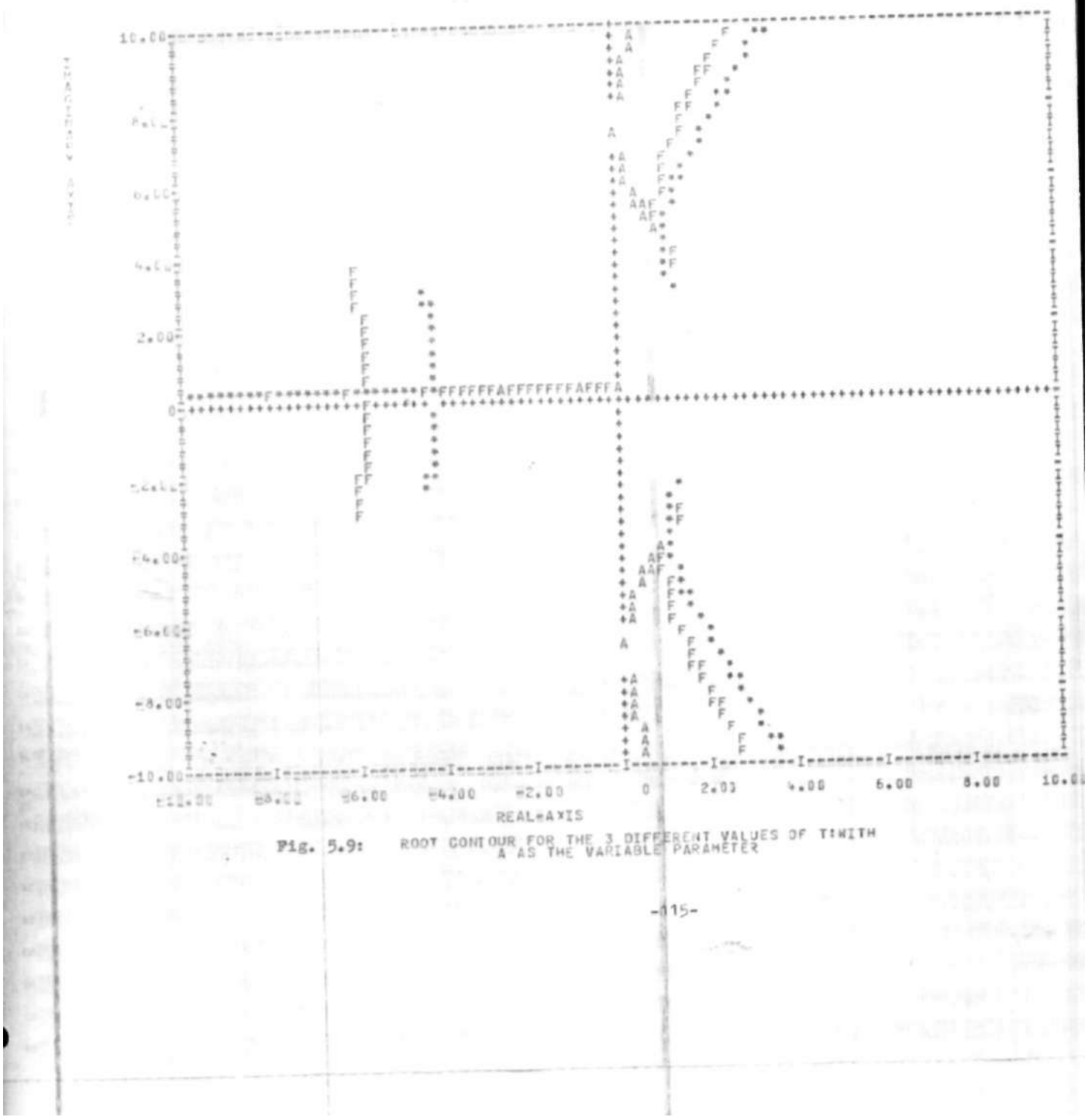



Fig. 5.9: ROOT CONTOUR FOR THE 3 DIFFERENT VALUES OF T1WITH A AS THE VARIABLE PARAMETER

It is seen from these figures that for values of $T > 0.052$, no part of the complex root loci from the RHP complex poles crosses the imaginary axis. Whereas for values of $T \leq 0.052$, parts of the complex root loci from the RHP complex poles could cross or be on the imaginary axis.

This observation shows that the given system cannot be compensated (at least in terms of stability) for any values of $T > 0.052$. If $T < 0.052$, it is possible to determine some point in the s-plane (at some a values) for which the system could be compensated.

However, since the same design is tackled here by two different methods, the problem of determining some fixed values of the parameters which could be adequate for the compensator design is deferred till the next section.

It should be noticed that the parameter, T , is a time constant of the compensator. Hence this method is recommended for compensator design when direct information on the settling time or transient response of the system is required. As will be seen in the next section the alternative method deals directly with the pole-zero locations for the compensator. This method of course serves the same purpose but indirectly. In fact, the values of the pole and zero are the inverses of the time constants of the compensator.

5.2.3. Compensator Transfer Function in Root Locus Form

In this method, an alternative representation is used for the transfer function.

With

$$z = \frac{1}{CR} \qquad \text{--------} \qquad 5.32$$

and

$$p = \frac{R + R_1}{C R R_1} \qquad \text{--------} \qquad 5.33$$

The transfer function for the lead compensator of equation (5.1) becomes

$$\frac{E_o}{E_i}(s) = \frac{s + z}{s + p} \qquad \text{------} \qquad 5.34$$

While with

$$z = \frac{1}{C_1 R_1} \qquad \text{------} \qquad 5.35$$

and

$$p = \frac{1}{C_1 (R + R_1)} \qquad \text{------} \qquad 5.36$$

The transfer function for the lag compensator of equation (5.2) becomes

$$\frac{E_o}{E_i}(s) = \frac{p}{z} \left(\frac{s + z}{s + p}\right) \qquad \text{------} \qquad 5.37$$

The two equations (5.34) and (5.37) can be combined into single compensator transfer function

$$G_c(s) = \frac{1}{K_c} \left(\frac{s + z}{s + p}\right) \qquad \text{-----} \qquad 5.38$$

where it is obvious that for lag compensation, the constant $K_c$ is given by

$$K_c = \frac{z}{p} \qquad \text{------} \qquad 5.39$$

while for lead compensation

$$K_c = 1 \qquad \text{------} \qquad 5.40$$

It is necessary to put the transfer function in the general form of (5.38) in order that the lag compensator does not affect the velocity constant, $K_v$, and hence the steady state accuracy of the given plant.

If $z > p$, the compensator of (5.38) is a lead compensator while if $z < p$, the compensator is a lag type.

The transfer function of the uncompensated system

$$G(s) = \frac{K}{s(s + 1)(s + 3)} \; ; \quad K = 100$$

is still as shown in figure 5.4. However, using the compensator transfer function of equation (5.38), with $K_c = z/p$ (i.e. for lag compensation), the transfer function of the compensated system now takes the form:

$$G_c(s)\ G(s) = \frac{K(s+z)}{K_c\ s(s+1)(s+3)(s+p)} \quad \text{----} \quad 5.41$$

where

$$K = 100$$

Using equation (5.40), (5.41) could be expressed viz:

$$G_c(s)\ G(s) = \frac{p(\ s(s+1)(s+3) + 100)}{s(s^3 + 4s^2 + 3s + \frac{100}{K_c}\ )} \quad \text{------} \quad 5.42$$

and so the CE for the compensated system becomes

$$\frac{p\ (s(s+1)(s+3) + 100\ )}{s(s^3 + s^2 + 3\ s + \frac{100}{K_c}\ )} = -1$$

If the roots of the uncompensated system are once more assumed to be $r_1$, $r_2$ and $r_3$, then the equation takes the form:

$$\frac{p\ (s + r_1)(s + r_2)(s + r_3)}{s(s^3 + s^2 + 3s + 100/K_c\ )} = -1 \quad \text{-------} \quad 5.43$$

where (Cf Section 5.2.2.)

$$r_1 = 1.077 - j3.886 \quad \text{-------} \quad 5.44$$

$$r_2 = 1.077 + j3.886 \quad \text{-------} \quad 5.45$$

and

$$r_3 = -6.154 + j\,0 \quad \text{-------} \quad 5.46$$

The term $s^3 + 4s^2 + 3s + 100/K_c$ can be factored viz:-

$$\frac{3\,(s + 100/3K_c)}{s^2(s+4)} = -1 \qquad \text{------} \quad 5.47$$

in which case its solution is the same with the solution of the uncompensated system with

$$k = \frac{100}{K_c} \qquad \text{------} \quad 5.48$$

Hence for each value of $K_c$, $0 \leqslant K_c \leqslant \infty$, a set of roots $r_4$, $r_5$ and $r_6$ could be determined such that the composite C E of the compensated system becomes

$$\frac{p(s + r_1)\,(s + r_2)\,(s + r_3)}{s(s + r_4)\,(s + r_5)\,(s + r_6)} = -1 \qquad \text{-------} \quad 5.49$$

The root loci with $p$ as the variable parameter will then depend on the particular value of $K_c$. For values of $0 \leqslant K_c < 1$, the locus will be on the high gain side since $K = 100/K_c$ and system becomes less stable. While for $K_c > 1$, the locus is on the low gain side.

Since values of $K_c$ in the range $(0, 1)$ exclussive, will result in lead compensation, it is apparent that the given system cannot be compensated using a lead compensator (since this has a destabilizing effect on the system.

The values $K_c$ used and the corresponding values of the roots $r_4$, $r_5$, and $r_6$ are given in table 5.2.

| $K_c$ | Real Part | Imaginary Part | K | Roots |
|---|---|---|---|---|
| 0.5 | 1.675 | - 4.988 | 100 | $r_4$ |
| " | 1.675 | 4.988 | " | $r_5$ |
| " | -7.35 | 0.0 | " | $r_6$ |
| | | | | |
| 5 | 0.182 | - 2.140 | 100 | $r_4$ |
| " | 0.182 | 2.140 | " | $r_5$ |
| " | -4.380 | 0.0 | " | $r_6$ |
| | | | | |
| 50 | -0.372 | 0.696 | 100 | $r_4$ |
| " | -0.372 | - 0.696 | " | $r_5$ |
| " | -3.26 | 0.0 | " | $r_6$ |

Table 5.2 showing values of $K_c$ and the Roots Used

The modified root loci plots for these values of $K_c$ with p as the variable parameter are as shown in Figures 5.10 through 5.12.

It is obvious from the three root loci plots that a lag compensator with $K_c = 50$ could be used to compensate the given system. Since

$$K_c = z/p \ ,$$

it implies that

$$z = 50p$$

Hence the compensator T.F. becomes

$$G_c(s) = \frac{1}{50} \left(\frac{s + 50p}{s + p}\right) \qquad \text{------ 5.50}$$

The cross-over value of p is about 0.018. Hence plant can be compensated using $p = 0.01$ in which case

$$G_c(s) = \frac{1}{50} \ \frac{s + 0.5}{s + 0.01} \qquad \text{------ 5.51}$$

and the closed loop transfer function of the system becomes

$$G_c(s)\ G(s) = \frac{2(s + 0.5)}{s(s+1)(s+3)(s+0.01)}$$

——— 5.52

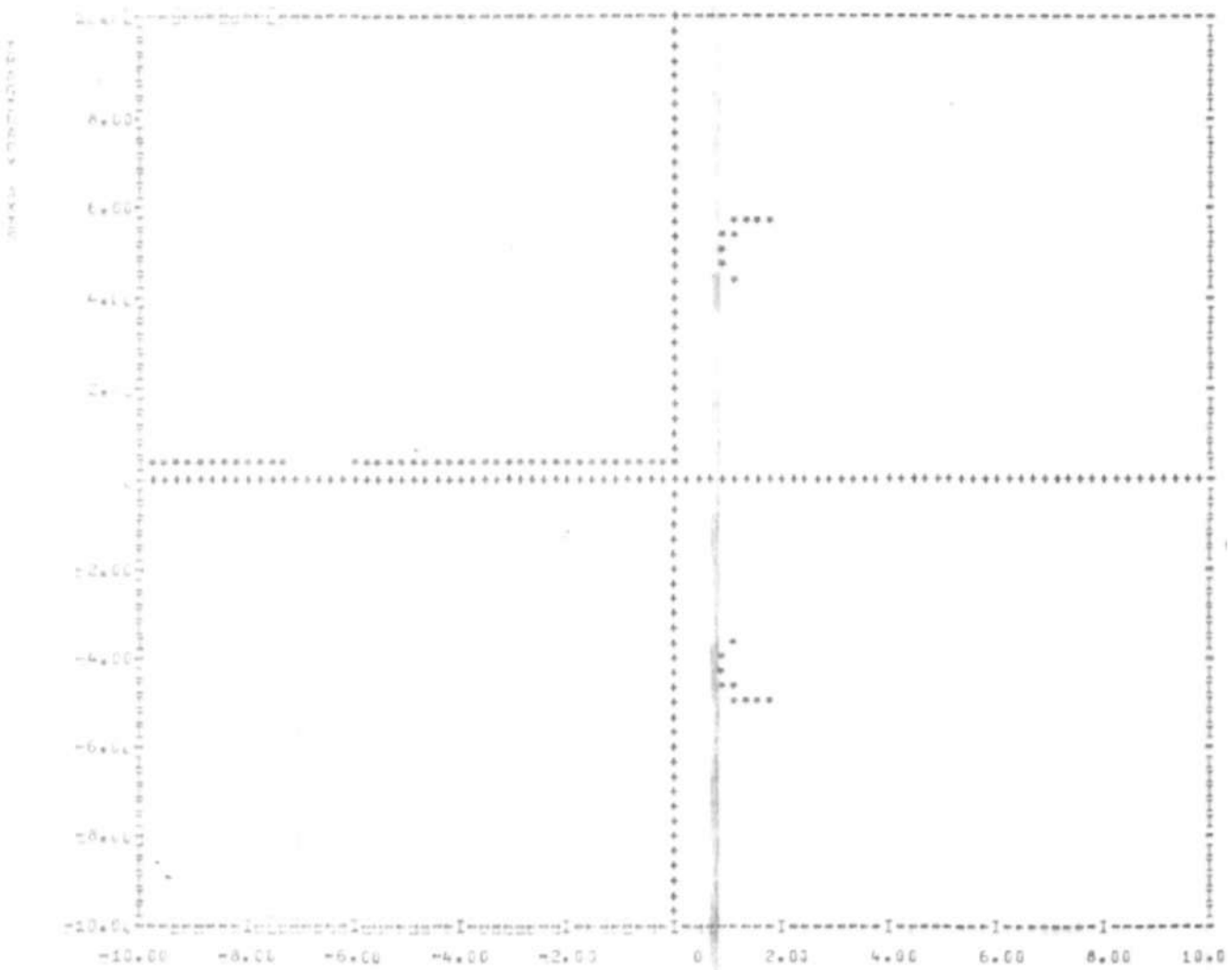



Fig. 5.10: ROOT LOCI PLOT FOR KC=0.5 WITH P AS THE VARIABLE PARAMETER

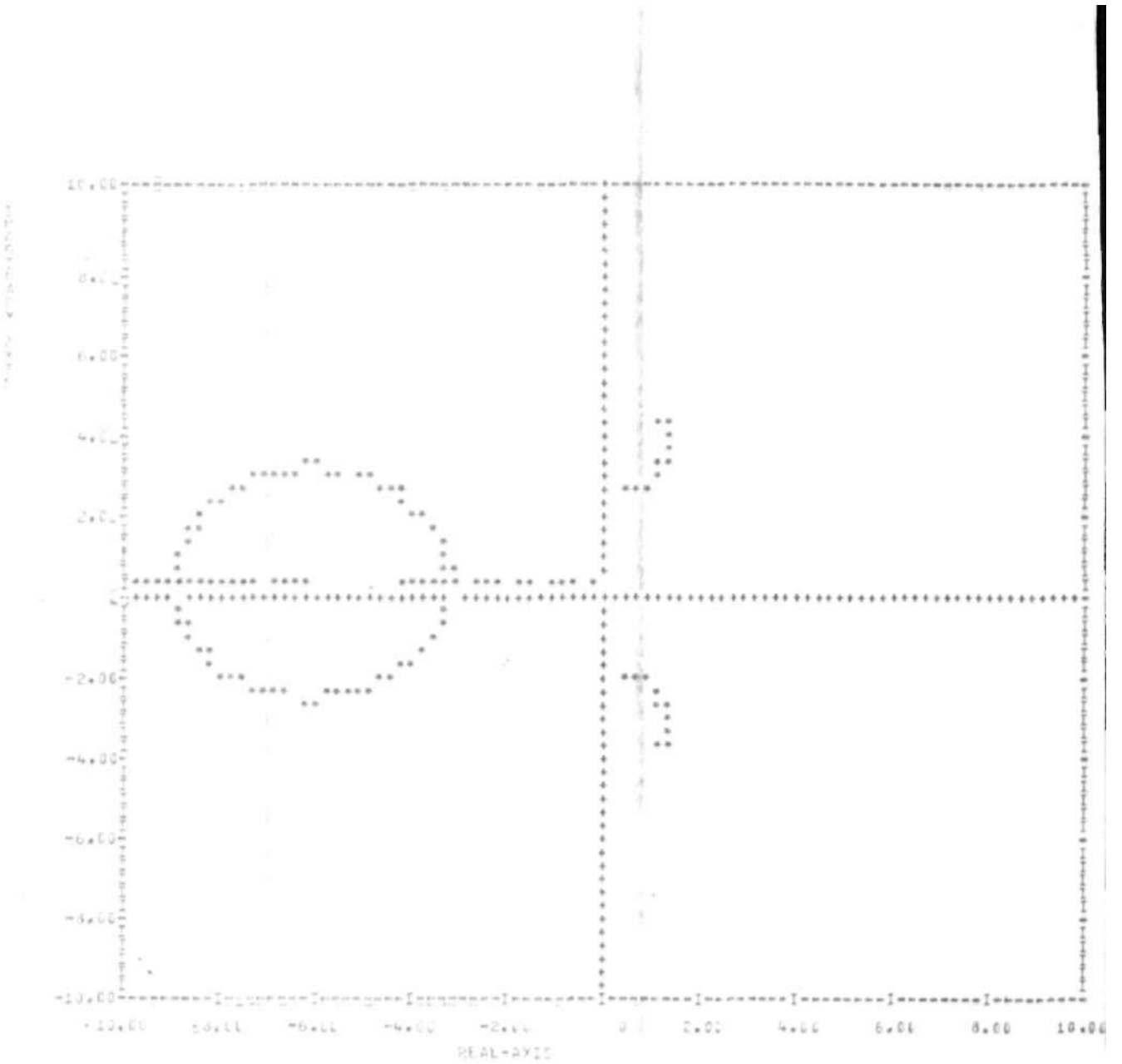



Fig 5.11: ROOT LOCI PLOT FOR KC=SMITH P AS THE VARIABLE PARAMETER

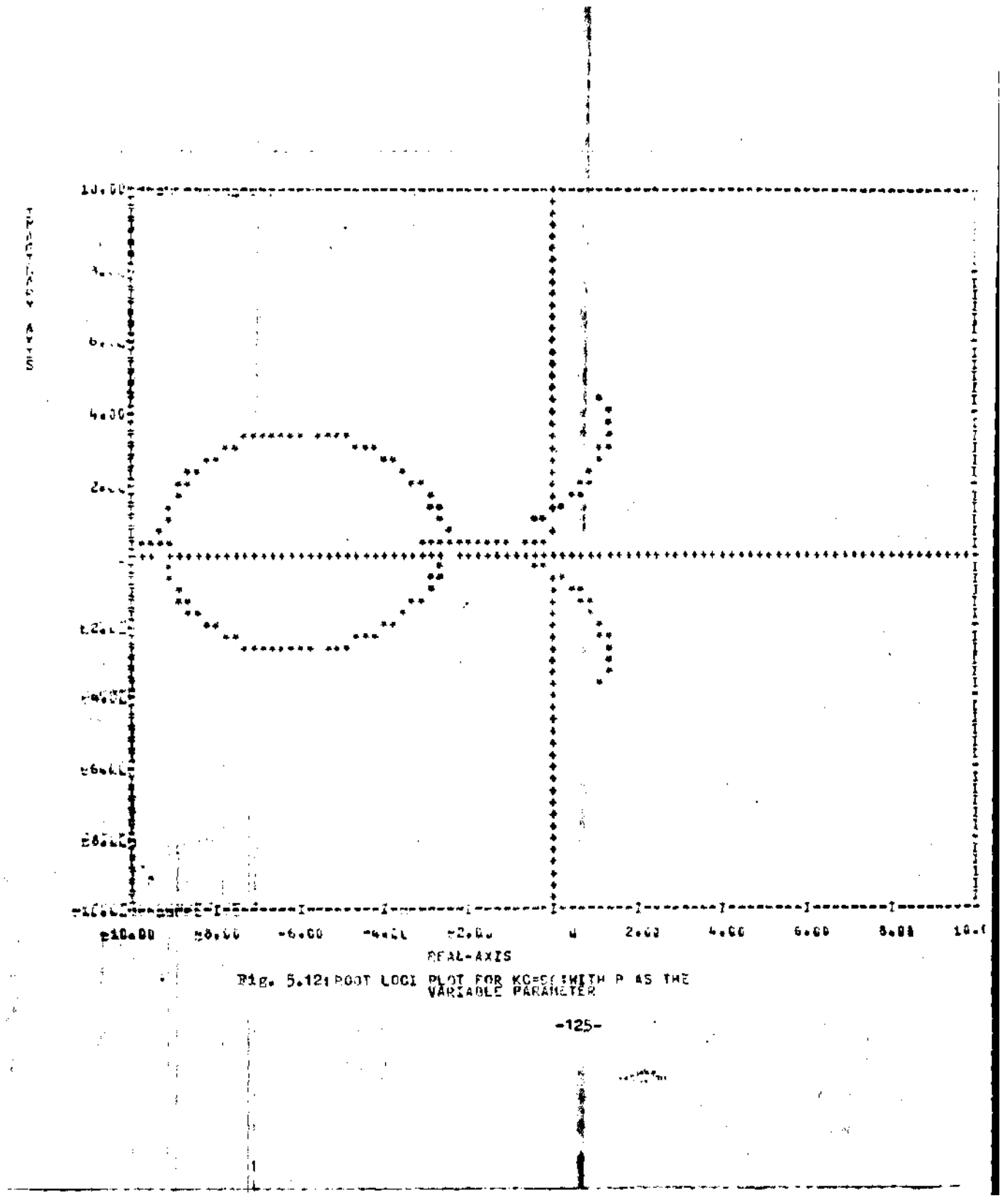



Fig. 5.12: ROOT LOCI PLOT FOR KC=5( WITH P AS THE VARIABLE PARAMETER

The values of the components for the hardware realization could be obtained by solving the following equations

$$\frac{R_1}{R + R_1} = \frac{1}{K_c} = \frac{1}{50} \quad \text{------} \quad 5.53$$

$$\frac{1}{C_1 R_1} = z = 0.5 \quad \text{------} \quad 5.54$$

and

$$\frac{1}{C_1(R + R_1)} = p = 0.01 \quad \text{------} \quad 5.55$$

These equations can be rearranged viz:

$$R = 49 R_1 \quad \text{------} \quad 5.56$$

and

$$C_1 = 2/R_1 \quad \text{------} \quad 5.57$$

The possible combinations of the component values are as listed in table 5.3.

| RESISTOR R1 (OHMS) | RESISTOR R (OHMS) | CAPACITOR C1 (FARADS) |
|---|---|---|
| 10. | 90. | .2[illegible] |
| 20. | 98. | .1[illegible] |
| 30. | 1470. | .666666666667E-01 |
| 40. | .9[illegible] | .5[illegible]E-01 |
| 50. | 2450. | .400000000000E-01 |
| 60. | 294. | .333333333333E-01 |
| 70. | 3430. | .285714285714E-01 |
| 80. | 392. | .250000000000E-01 |
| 90. | 441. | .222222222222E-01 |
| 100. | 490. | .200000000000E-01 |
| 110. | 539. | .181818181818E-01 |
| 120. | 588. | .166666666667E-01 |
| 130. | 637. | .153846153846E-01 |
| 140. | 686. | .142857142857E-01 |
| 150. | 735. | .133333333333E-01 |
| 160. | 784. | .125000000000E-01 |
| 170. | 833. | .117647058824E-01 |
| 180. | 882. | .111111111111E-01 |
| 190. | 931. | .105263157895E-01 |
| 200. | 980. | .100000000000E-01 |
| 210. | 1029. | .952380952381E-02 |
| 220. | 1078. | .[illegible]E-02 |
| 230. | 1127. | .869565217391E-02 |
| 240. | 1176. | .833333333333E-02 |
| 250. | 1225. | .800000000000E-02 |
| 260. | 1274. | .769230769231E-02 |
| 270. | 1323. | .740740740741E-02 |
| 280. | 1372. | .714285714286E-02 |
| 290. | 1421. | .689655172414E-02 |
| 300. | 1470. | .666666666667E-02 |
| 310. | 1519. | .[illegible] |
| 320. | 1568. | .[illegible] |
| 330. | 1617. | .[illegible] |
| 340. | 1666. | .[illegible] |
| 350. | 1715. | .[illegible] |
| 360. | 1764. | .[illegible] |
| 370. | 1813. | .[illegible] |
| 380. | 1862. | .[illegible] |
| 390. | 1911. | .[illegible] |
| 400. | 1960. | .[illegible] |
| 410. | 2009. | .[illegible] |
| 420. | 2058. | .[illegible] |
| 430. | 2107. | .[illegible] |
| 440. | 2156. | .[illegible] |
| 450. | 2205. | .[illegible] |
| 460. | 2254. | .[illegible] |
| 470. | 2303. | .[illegible] |
| 480. | 2352. | .[illegible] |
| 490. | 2401. | .[illegible] |
| 500. | 2450. | .[illegible] |

Table 5.3: Realistic Component Values

# CHAPTER VI

## CONCLUSION AND RECOMMENDATION FOR FURTHER WORK

### 6.1 Introduction

In every scientific and engineering research, there is always a set objective which is sought, by the researcher, to realize systematically. At times it is possible to bring the research to a conclusive end within some reasonable time lapse. In some cases, the period of research may drag to unexpected, or rather, unenvisaged lengths while in yet, some, it may not even be possible to bring the research to any reasonable conclusion.

It is desired, in this chapter, to reappraise and also give summary of what was intended and what has been achieved. The attendant problems are also discussed and suggestions for improvements offered.

### 6.2 Conclusion

With regards to the major task of this work, the implementation on the CDC Cyber 72 (installed here in A.B.U.) a computer program for the generation of the root loci points for any given system, there is no gain saying that this thesis has to a greater extent succeeded. The plotting of the computed points is realized using the subroutine plot z , the

output of which can be obtained either on the visual display terminals or on paper via the line printer. Hence the objective of the work serving as a root locus teaching aid is also met to a reasonable extent.

However, to serve as a proper design/analysis tool, it is essential that the root loci of a system be continuous rather than descrete as it is the case with the output of plot z. Designing with such plotted outputs as is obtained requires extrapolations here and there. Hence the series compensation design performed in chapter V is, as was rightly pointed out, rather indicative of the method for realizing series compensation design using the root locus technique.

As is always the case, whenever there exists more than one methods for realizing a particular objective, each of these alternatives must necessarily, have its own merits and demerits. Because of the fact that the algorithm employed in this thesis determines the root loci points without due consideration to the gain values, the values of the gains at the computed points are therefore purely unsupervised.(18) Since the two alternative methods described in chapter II can generate root loci points by selecting appropriate step increments for the gain, their gain values

are therefore supervised and it will be possible, for instance, to obtain a locus point corresponding exactly to, say, a gain of 100. The nearest Root98 can approach the locus point with a gain of 100 depends on the step size, $\delta$ . If a value of 0.125 is used for the step size, the nearest gain value to 100 generated will be 102.558. However, if a value of 0.001 is used the nearest gain value to 100 generated will become 100.049.

Presently, the execution of the program is a two-stage process - the first stage generates the root loci points while the second stage plots the graph for the generated points. The program may be ran more efficiently if the execution is combined in one. The need for the peace meal procedure arose initially due to storage problems with the computer.

Despite the shortcomings enumerated above, it is pictured in mind that this thesis will go a long way to providing a broad base for any work in the department in which analysis and/or design using the root locus technique is relevant.

## 6.3 Suggestions for Further Work

In order to improve the quality of the plotted output for the purpose of design and analysis, it is mandatory to realize the plotting of the computed points using the calcomp

plotter. Despite the fact that the calcomp plotter is an incremental digital plotter, it is fascinatingly vasertile and has the capabilities of producing very smooth curves as far as the plotted points are not very far apart. With such plotted outputs, it will be possible to perform very accurate design/analysis work.

To increase the ability of Root98 to generate a root locus point whose gain value is very close to a predetermined gain, it is suggested that the program be ran several times using smaller and smaller values of $\delta$ until the required gain value is determined to an acceptable accuracy.

Finally, it is possible to reduce the storage problem by using overlays. Such an exercise could be accomplished as follows:-

(a) The main program becomes the main overlay.

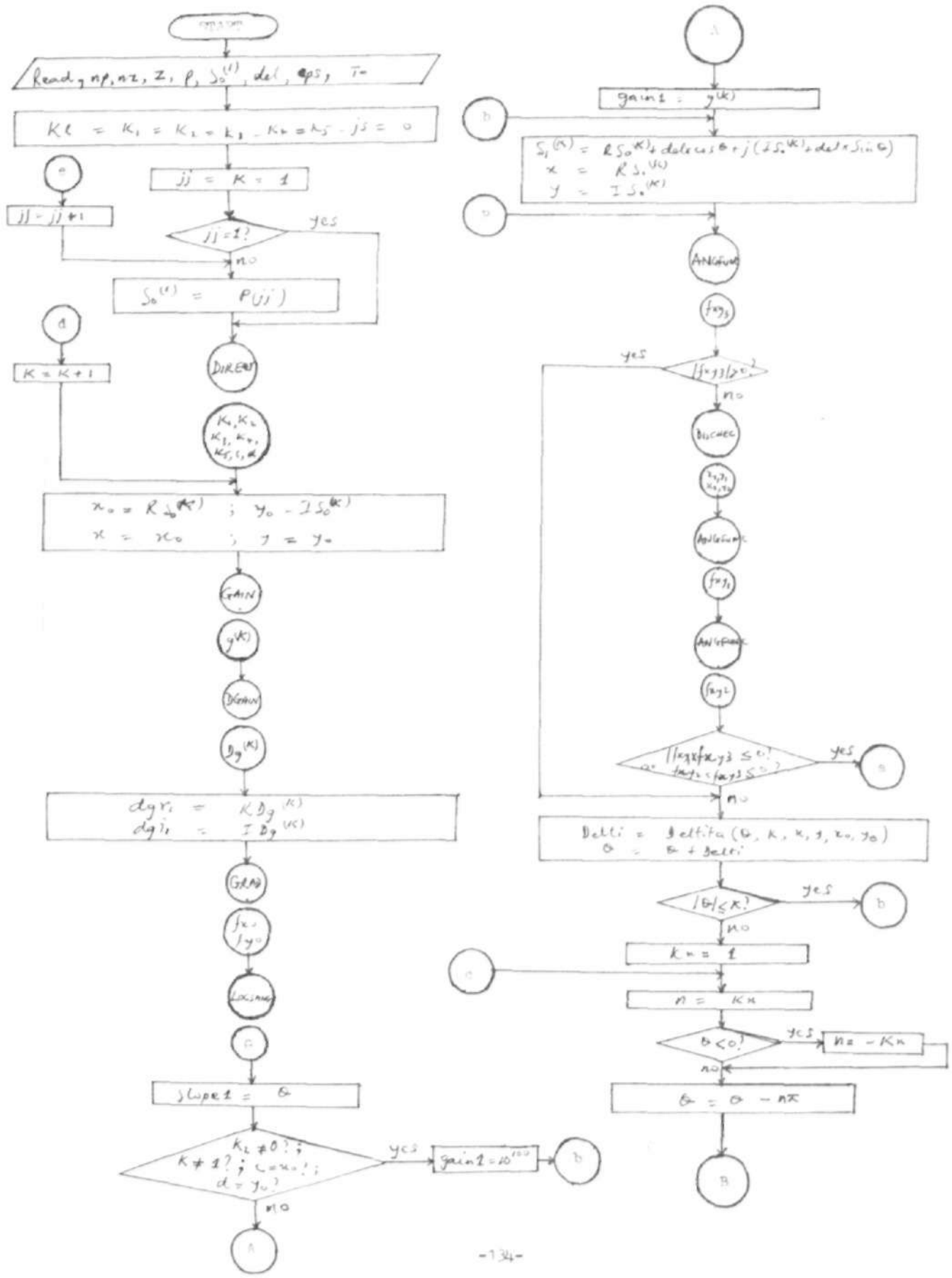
(b) All the subroutines become primary overlays.

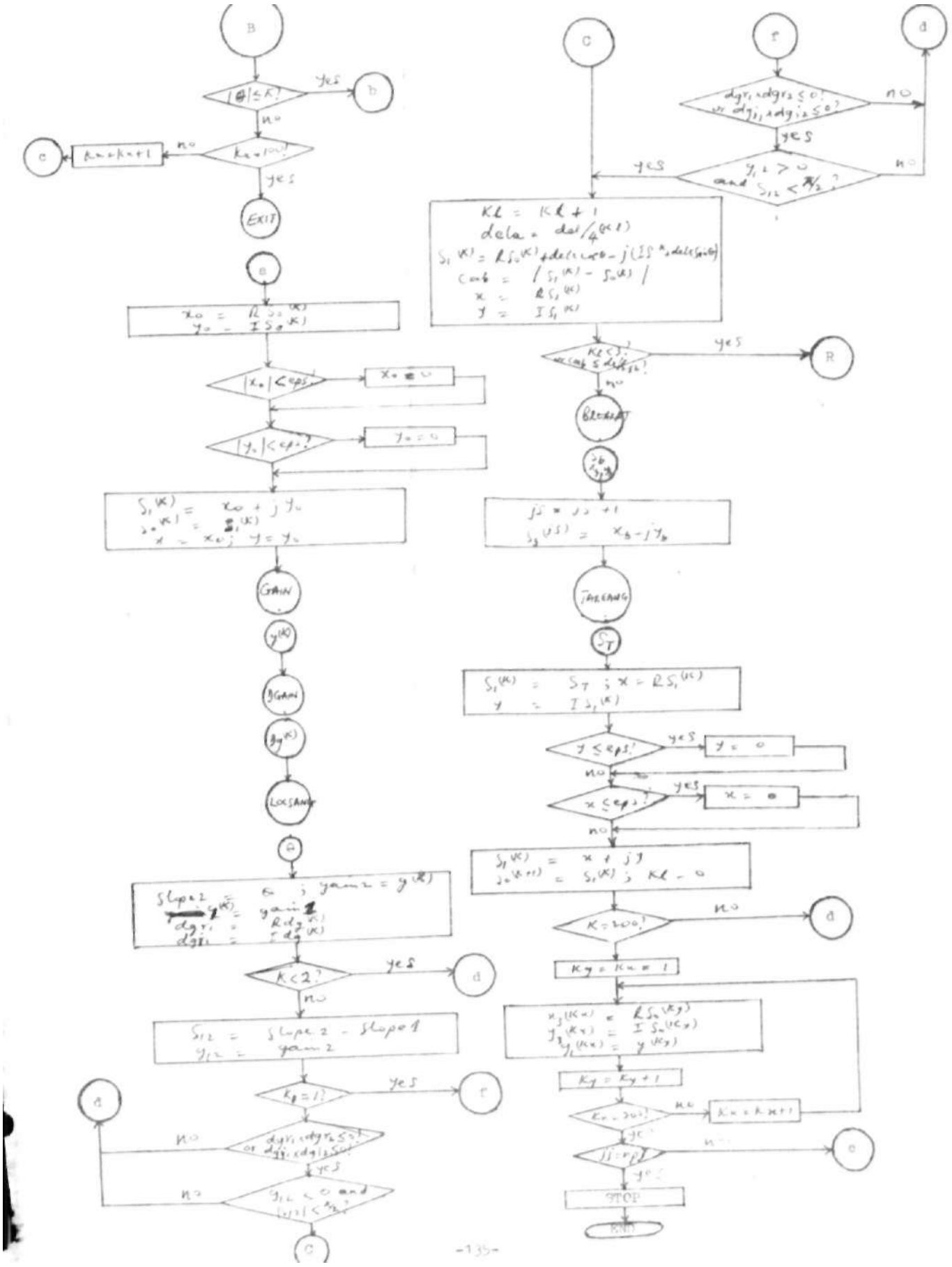
(c) The plotter becomes a primary overlay with the subroutine order as its secondary overlay.

It will infact be more economical to run the program with such an arrangement.

It is hoped that with the suggested improvements, Root98 will become a better design/analysis tool than as at now and rewardingly enough, practical design and/or analysis work using the root locus method can be implemented much easier.

# A P P E N D I X  A


Read, np, nz, Z, P, $S_0^{(i)}$, del, eps, T=
Kl = K1 = K2 = k1 = K4 = K5 = js = 0
jj = K = 1
jj = jj + 1
jj = 1?
yes
no
$S_0^{(i)}$ = P(jj)
K = K + 1
DIRETT
$K_0, K_2, K_3, K_4, K_5, i, \alpha$
$x_0 = R\,S_0^{(K)}$ ; $y_0 = I\,S_0^{(K)}$
x = x0 ; y = y0
GRAIN
g(K)
$dgr_i = R\,Dg^{(K)}$
$dgi_i = I\,Dg^{(K)}$
GRAD
fx0
fy0
Lacing
Slope1 = B
$K_2 \neq 0$ ?; K ≠ 1 ?; c = x0 ?; d = y0 ?
yes
gain1 = $10^{100}$
no
gain1 = g(K)
$S_1^{(K)} = R\,S_0^{(K)} + del\cos\theta + j(I\,S_0^{(K)} + del \times \sin\theta)$
x = R S0(K)
y = I S0(K)
ANGFUNC
fxy3
$|fxy3|^2 < \varepsilon^2$ ?
yes
no
DIRCHEC
x1,y1 x2,y2
AGGIUNTA
fxy1
fxy2
ANGFUNC
$|fxy3 \times fxy1| \leq \varepsilon^2$ ?
$tan\alpha = fxy3 \leq \varepsilon^2$ ?
yes
no
Delti = Deltita (B, K, K, 1, x0, y0)
B = B + Delti
|B| ≤ K ?
yes
no
kn = 1
n = Kn
B < 0 ?
yes
n = - Kn
no
B = B - nK


B
|Θ| ≤ K?
yes
b
no
c
k = k + 1
k = 100?
yes
EXIT
a
x0 = R S0(K)
y0 = I S0(K)
|x0| < eps?
x0 = 0
|y0| < eps?
y0 = 0
S1(K) = x0 + j y0
z0(K) = S1(K)
x = x0; y = y0
GAIN
y(K)
yGAIN
yy(K)
LOCSAND
a
slope2 = ... ; yam2 = y(K)
gain2
dyr2 = R dy(K)
dyi2 = I dy(K)
K < 2?
yes
d
no
S12 = slope2 - slope1
y12 = yam2
Kg = 1?
yes
f
no
dyr1 · dyr2 ≤ 0?
or dyi1 · dyi2 ≤ 0?
yes
no
y12 < 0 and
|x12| < ...
c
C
KL = KL + 1
dela = dal/4(KL)
S1(K) = R S0(K) + dela(cos Θ − j(I S0(K) · dela(sin Θ)
comb = |S1(K) − S0(K)|
x = R S1(K)
y = I S1(K)
KL < 3?
or comb ≤ dist(KL)?
yes
R
no
BLOCKAT
xb
yb
jj = jj + 1
yy(jj) = xb − j yb
TALKANG
ST
S1(K) = ST ; x = R S1(K)
y = I S1(K)
y ≤ eps1?
yes
y = 0
no
x ≤ eps2?
yes
x = 0
no
S1(K) = x + j y
z0(K+1) = S1(K) ; KL = 0
K = 200?
no
d
Ky = Kx = 1
x3(Kx) = R S0(Ky)
y3(Kx) = I S0(Ky)
y1(Kx) = y(Ky)
Ky = Ky + 1
Ky = 200?
no
Kx = Kx + 1
yes
jj = ...?
no
e
yes
STOP
END
f
d
e

LEGEND

np is number of finite open loop poles

nz is number of finite open loop zeros

$z_i$, i = 1...nz is open loop zeros

$p_i$, i = 1...np is open loop poles

del is step size

eps is error tolerance

$T_o$ = time delay

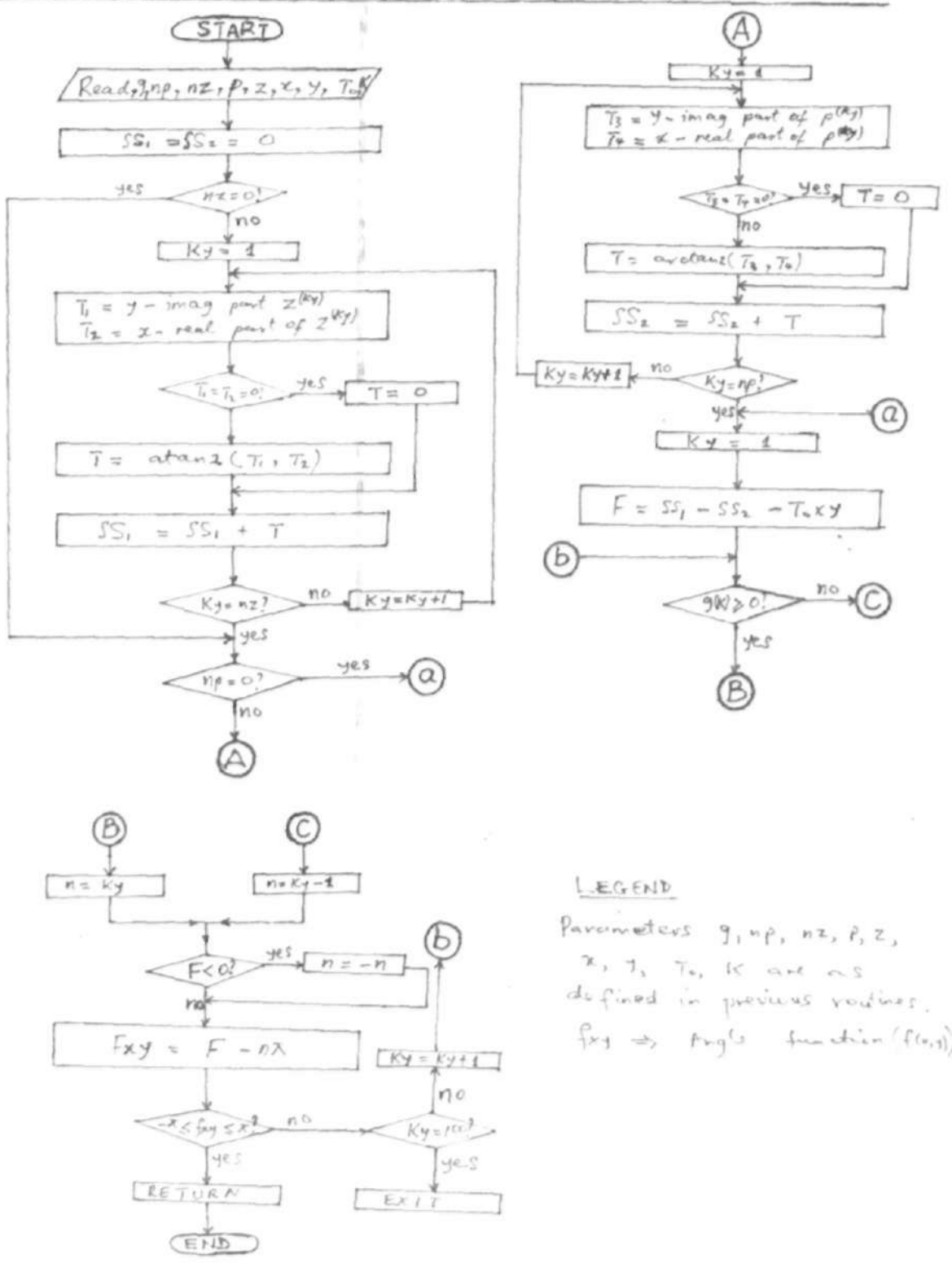
$K_1$ is counter for step cutting procedure

$K_1$, $K_2$, $K_3$, $K_4$, $K_5$ are the starting and direction indices

js is a break-away point counter

The subroutines are as defined in table 3.1

# FLOW CHART FOR ROUTINE ANGFUNG

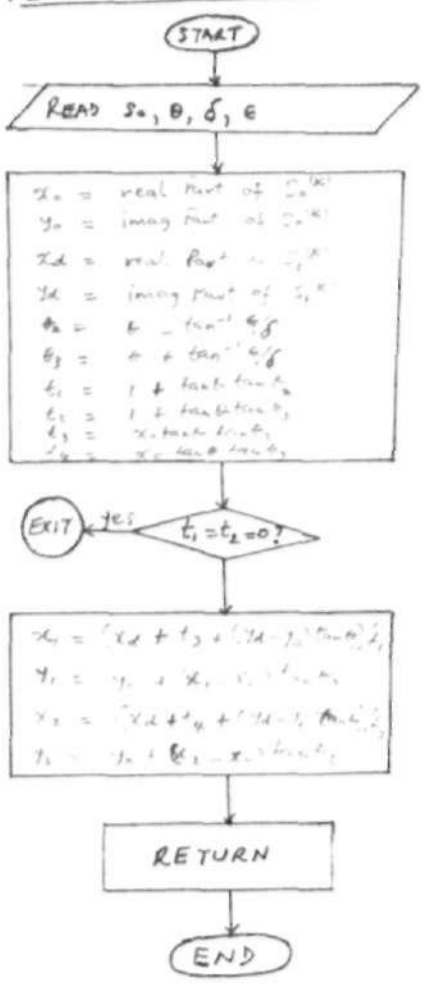



LEGEND

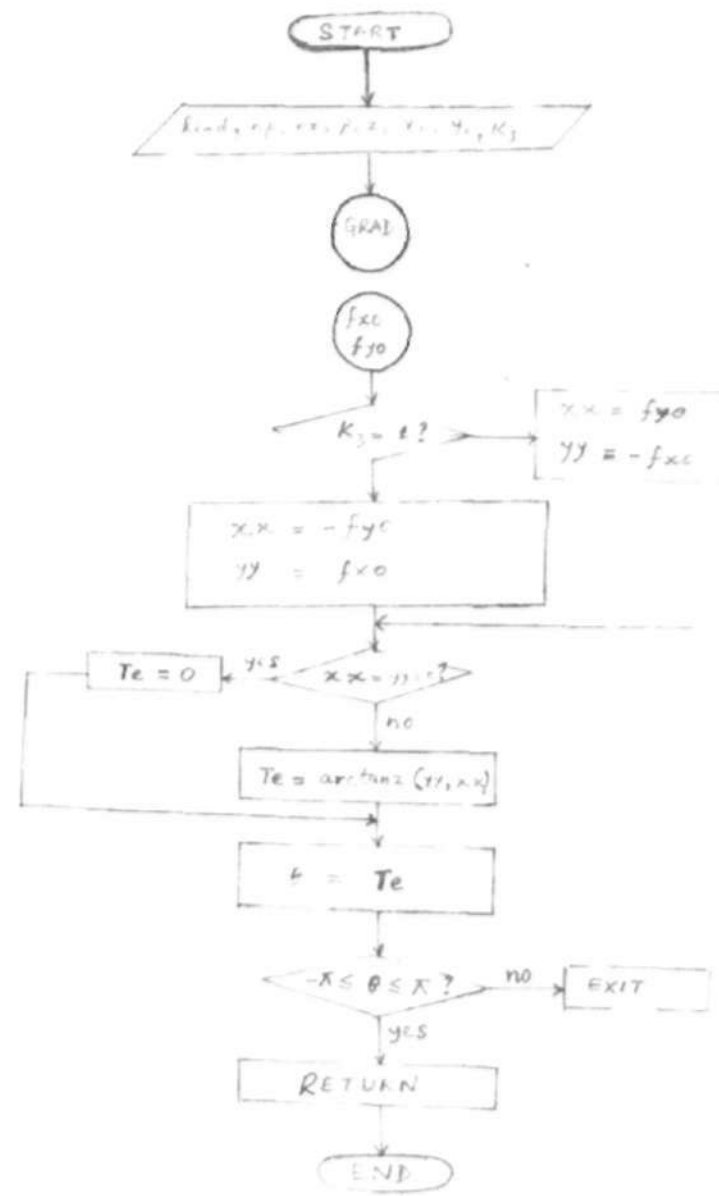
Parameters g, np, nz, P, z, x, y, Tol, K are as defined in previous routines.

fxy ⇒ Arg's function (f(x,y))

FLOW CHART FOR ROUTINE DISCHRG




FLOW CHART FOR ROUTINE LOCSANG



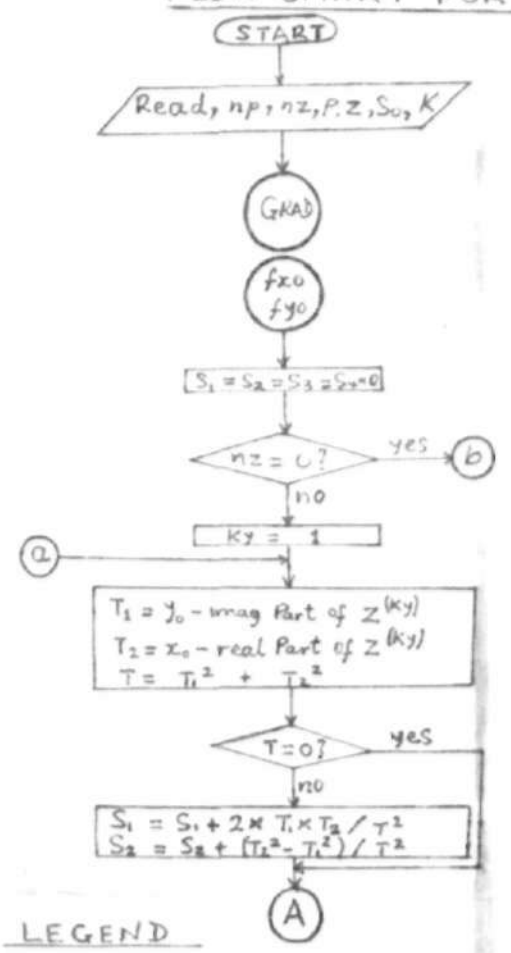

LEGEND

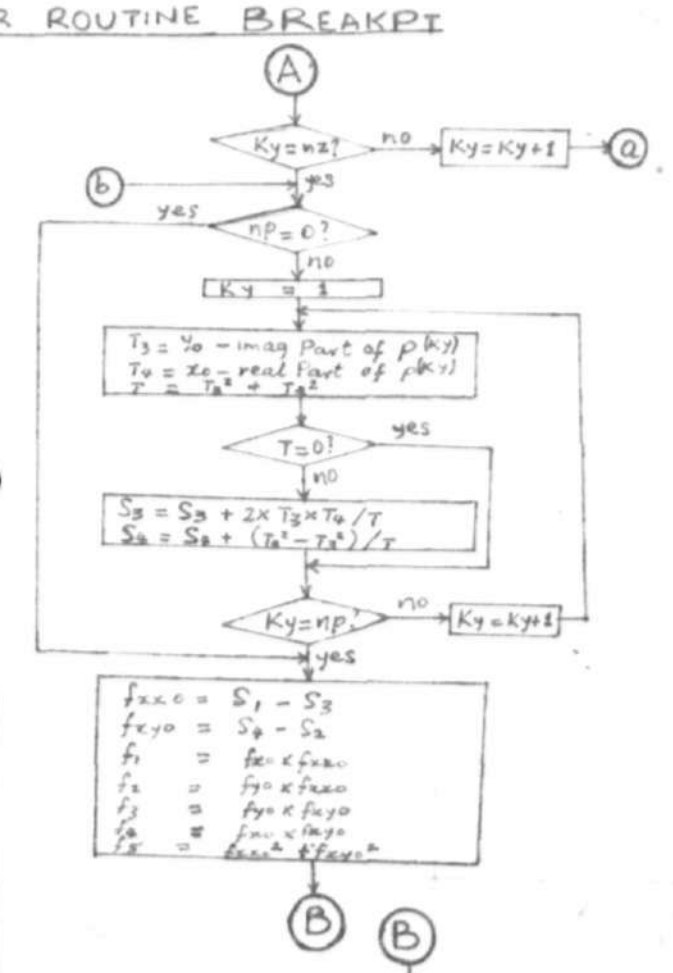
$\theta$ is locus slope angle

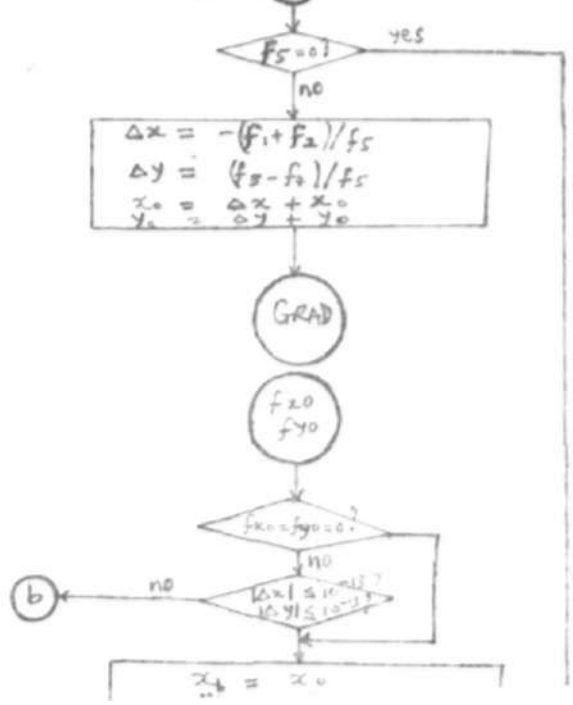
$x_1, y_1$; $x_2, y_2$ are the two differentiation points.

Other parameters are as previously defined.

# FLOW CHART FOR ROUTINE BREAKPI




## LEGEND

GRAD ⇒ SUBROUTINE GRAD

$f_{x0}$ ⇒ $f_x^*(x_0, y_0)$

$f_{y0}$ ⇒ $f_y^*(x_0, y_0)$

$f_{xx0}$ ⇒ $f_{xx}^*(x_0, y_0)$

$f_{xy0}$ ⇒ $f_{xy}^*(x_0, y_0)$

np ⇒ no of finite open loop poles

nz ⇒ no of finite open loop zeros

$p^{(Ky)}$, Ky = 1, ... np ⇒ pole positions

$Z^{(Ky)}$, Ky = 1, ... nz ⇒ zero positions

$S_0 = x_0 + jy_0$ ⇒ starting point

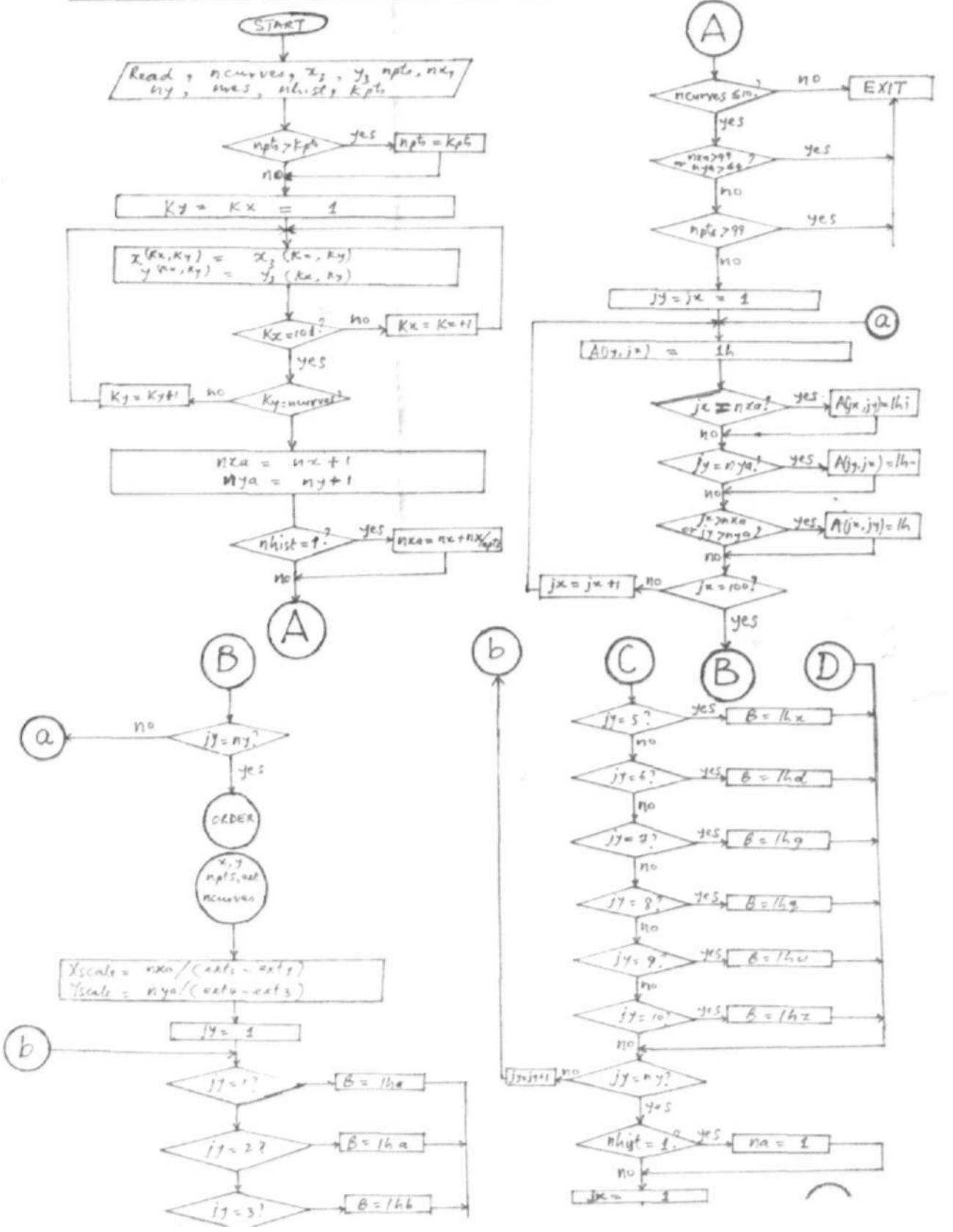
$S_b = x_b + jy_b$ ⇒ Breakaway/in point

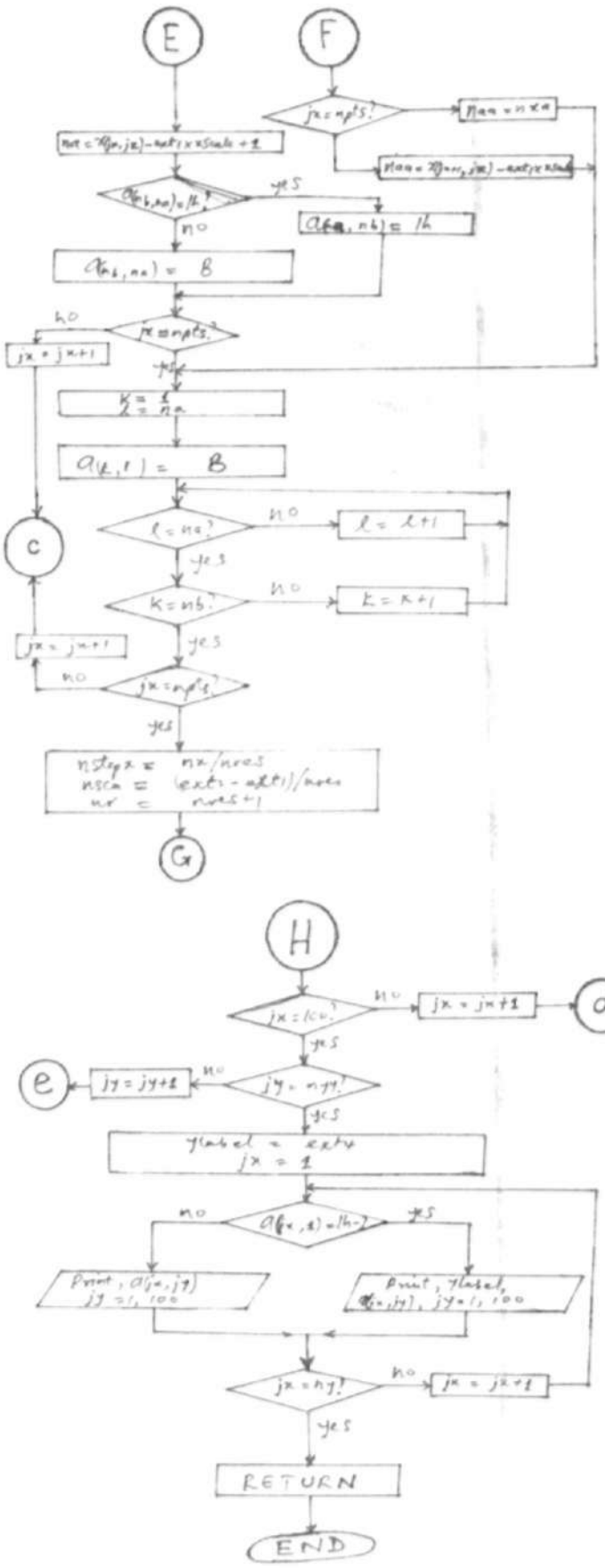
K ⇒ main program loop counter

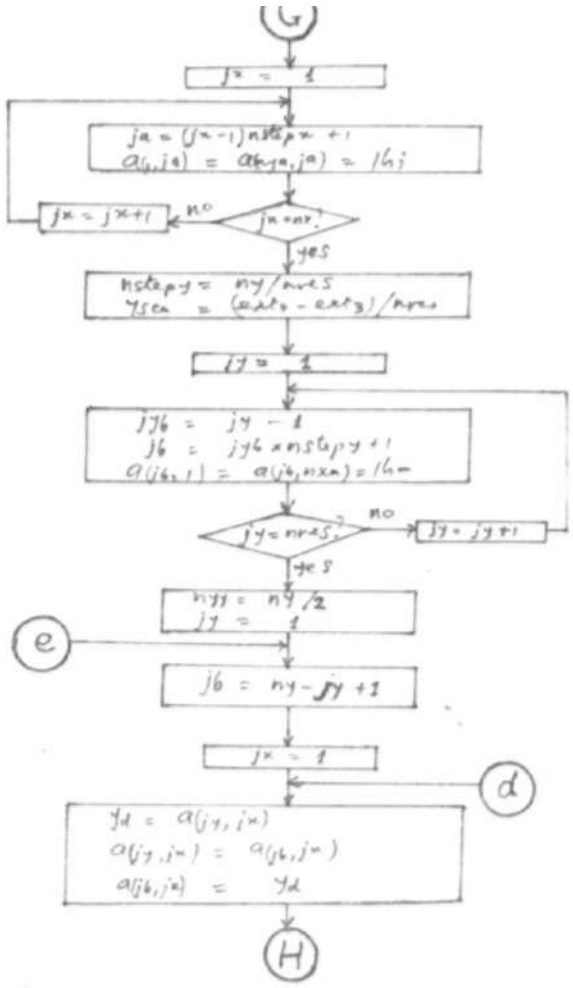
Ky ⇒ Subroutine loop index.

Note : The definitions of the above parameters remain same in subsequent subroutine flow charts.

# FLOW CHART FOR ROUTINE PLOTZ









## LEGEND

Ncurves ⇒ no of curves ≤ 15

$x_3$ ⇒ real part of a locus point

$y_3$ ⇒ imag part of a locus point

npts ⇒ no of points for each curve ≤ 99

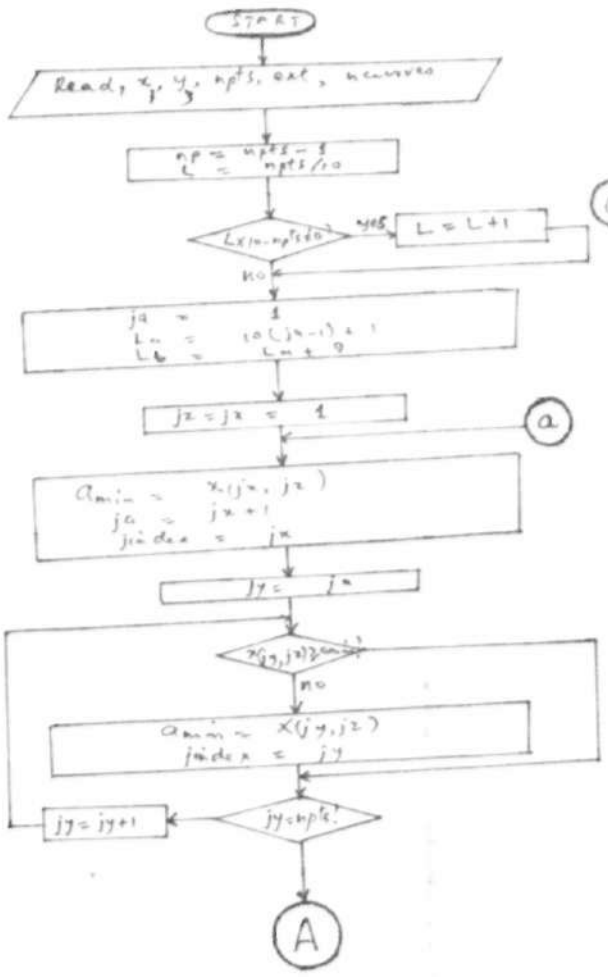
nx ⇒ no of columns ≤ 99

ny ⇒ no of rows ≤ 63

nres ⇒ resolution = 10

nhist indicates whether an ordinary curve or a histogram is wanted. nhist = 0 for root locus plots.

# FLOW CHART FOR ROUTINE ORDER

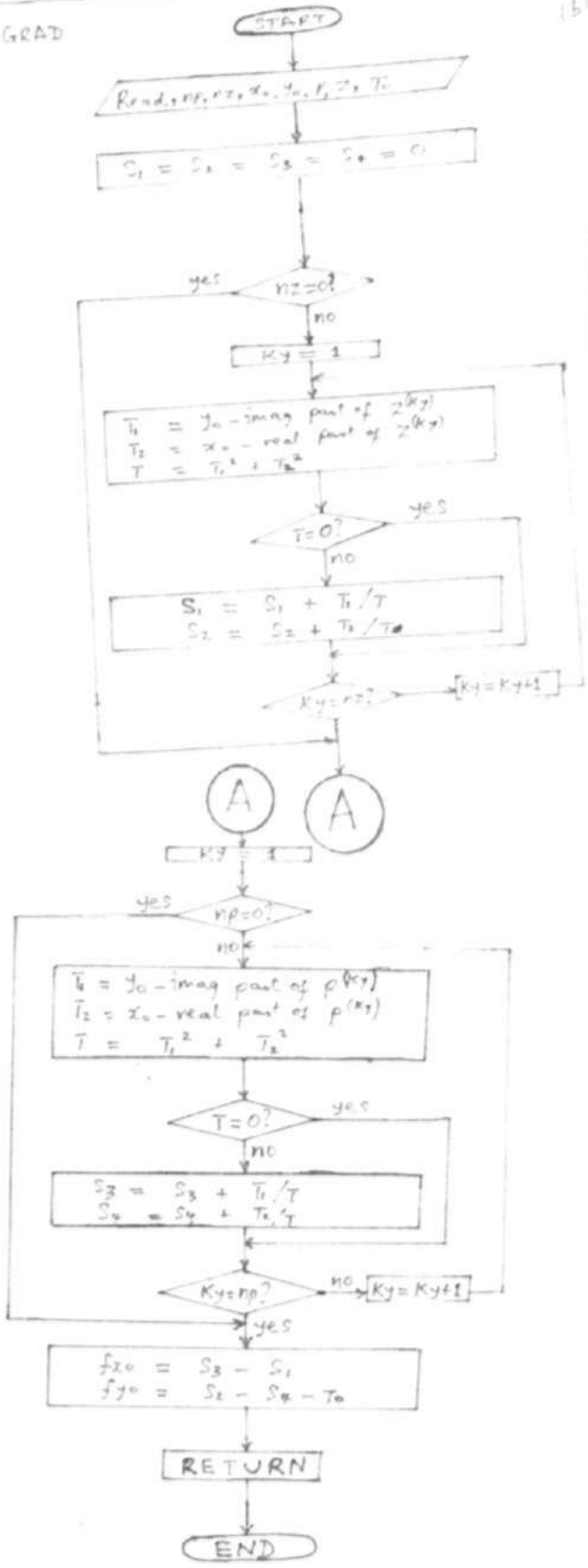



LEGEND

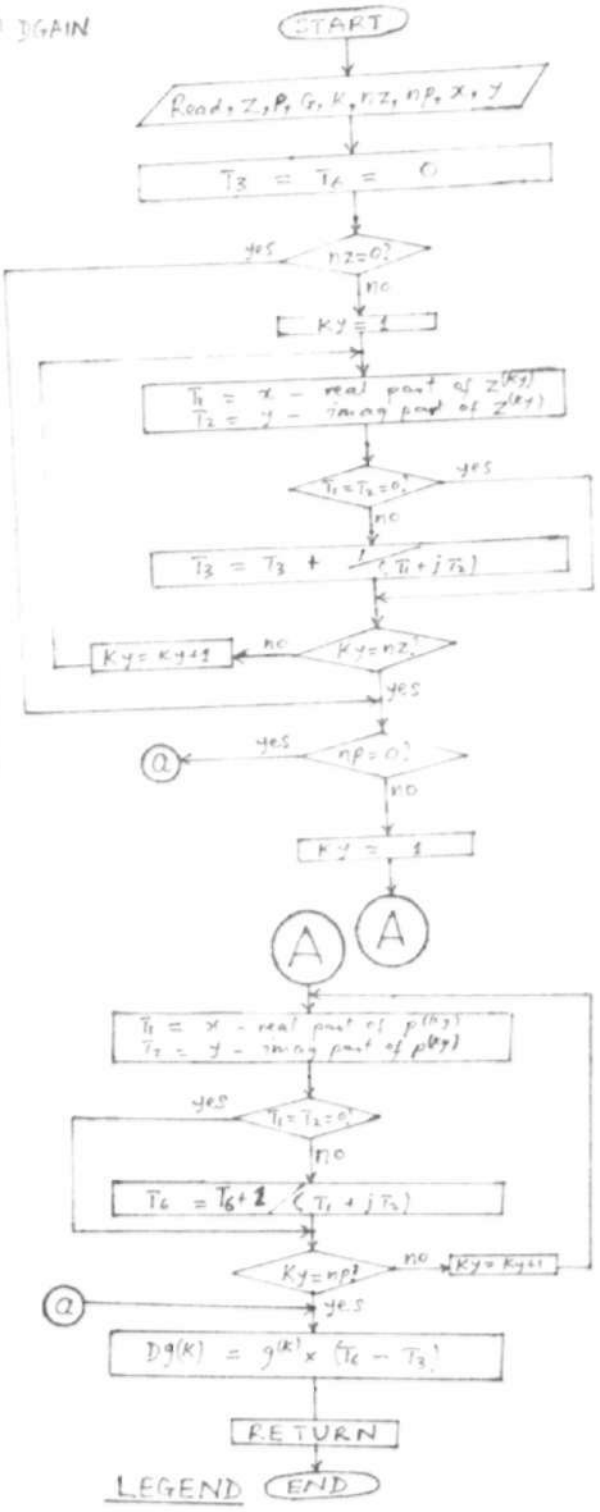
$Ext_1$, $Ext_2$, $Ext_3$, $Ext_4$ are the maximum and minimum values of $y_j$ and $x_j$ respectively. Other parameters are as defined in routine plotz

FLOW CHARTS FOR ROUTINES GRAD AND DGAIN

(a) GRAD




(b) DGAIN

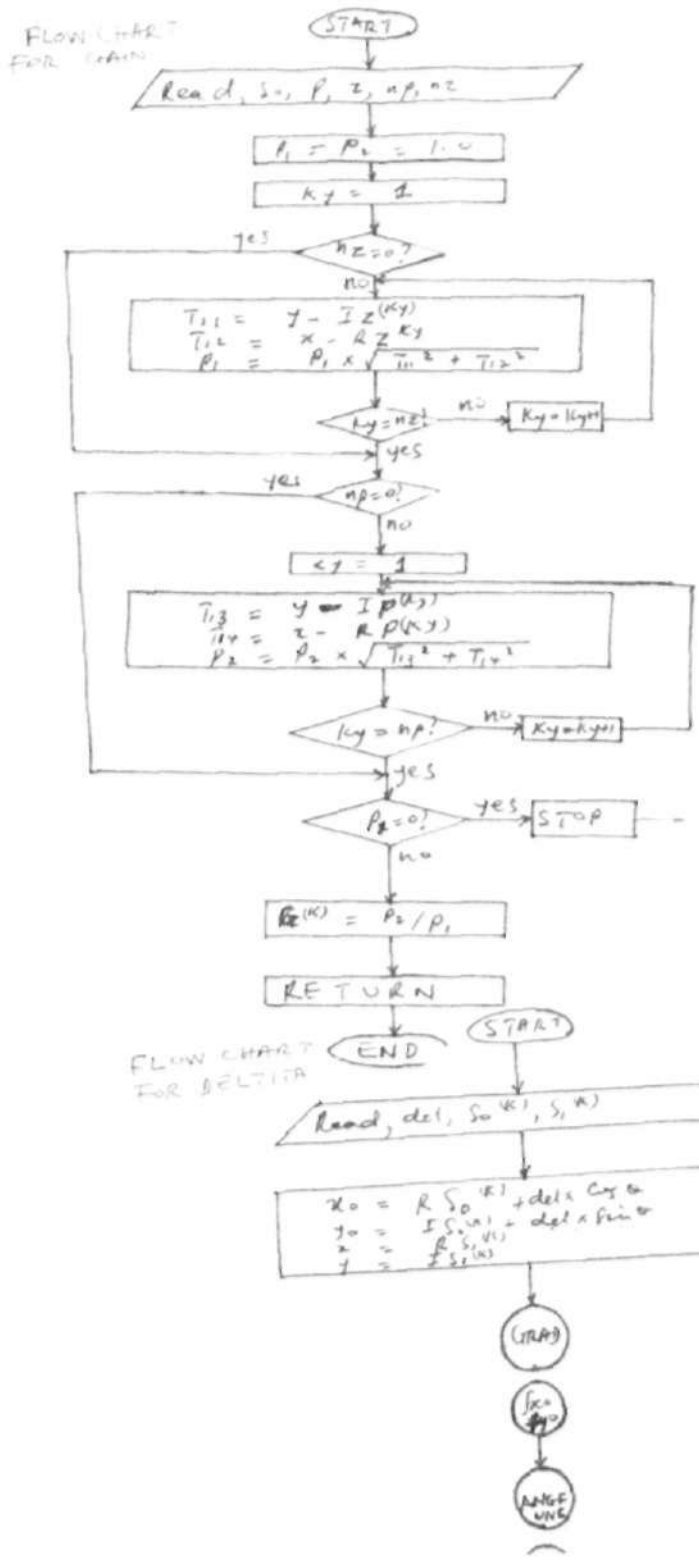



LEGEND

np, nz, P, Z, K are as previously defined.
x, y are the real and imaginary parts of any test point.
x0, y0 are the real and imaginary parts of any starting point.
g ⇒ Root locus gain.
Dg ⇒ first derivative of the complex gain.
T0 ⇒ Time delay.

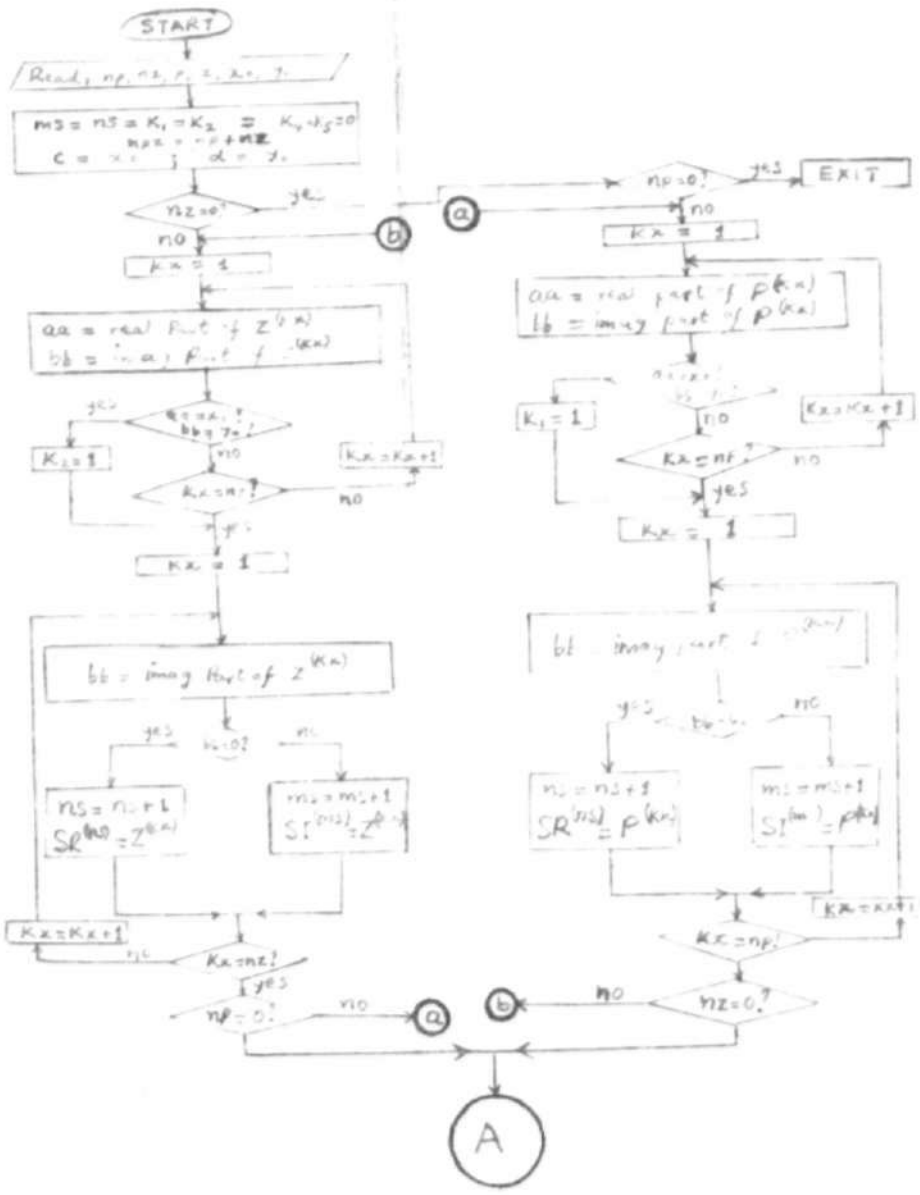
FLOW CHART FOR ROUTINES GAIN AND DELTIA

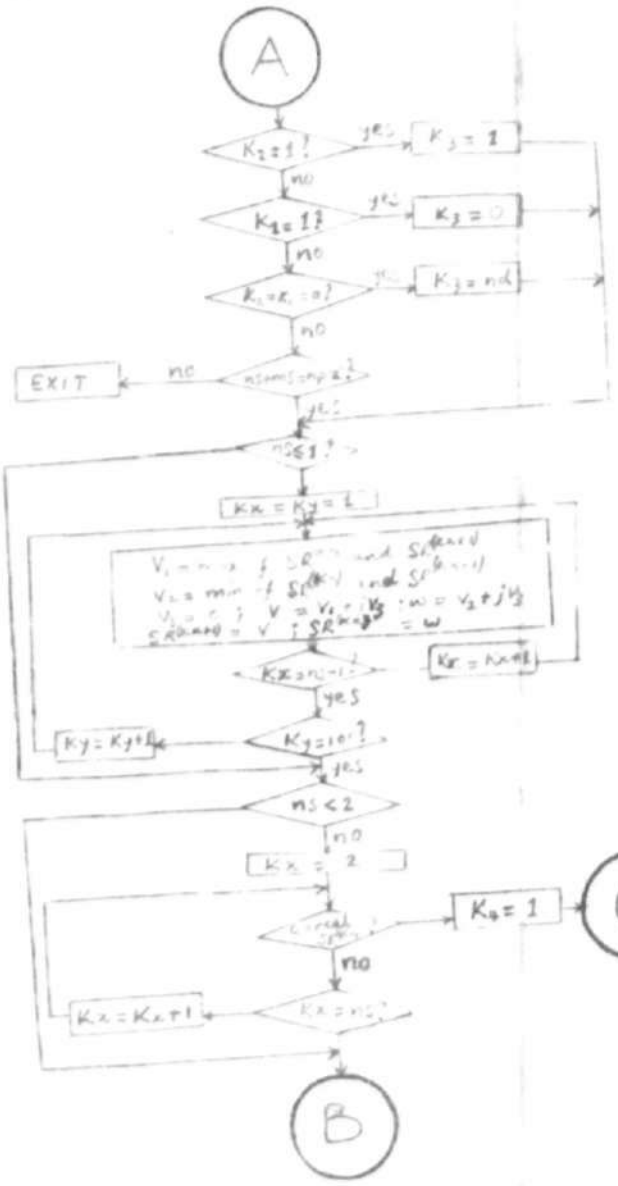
FLOW CHART FOR GAIN

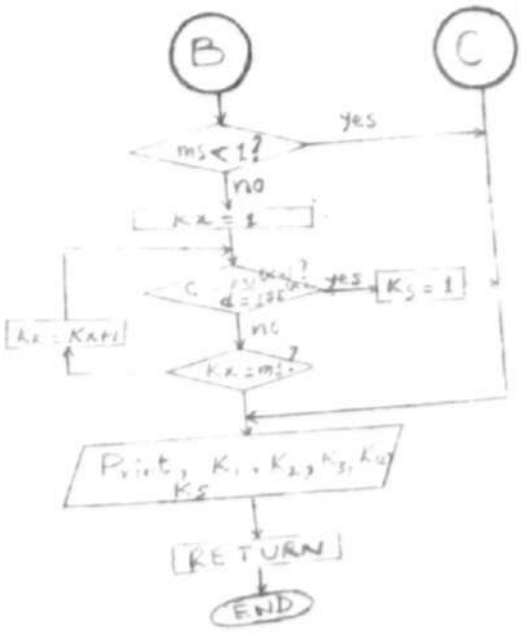



FLOW CHART FOR DELTIA

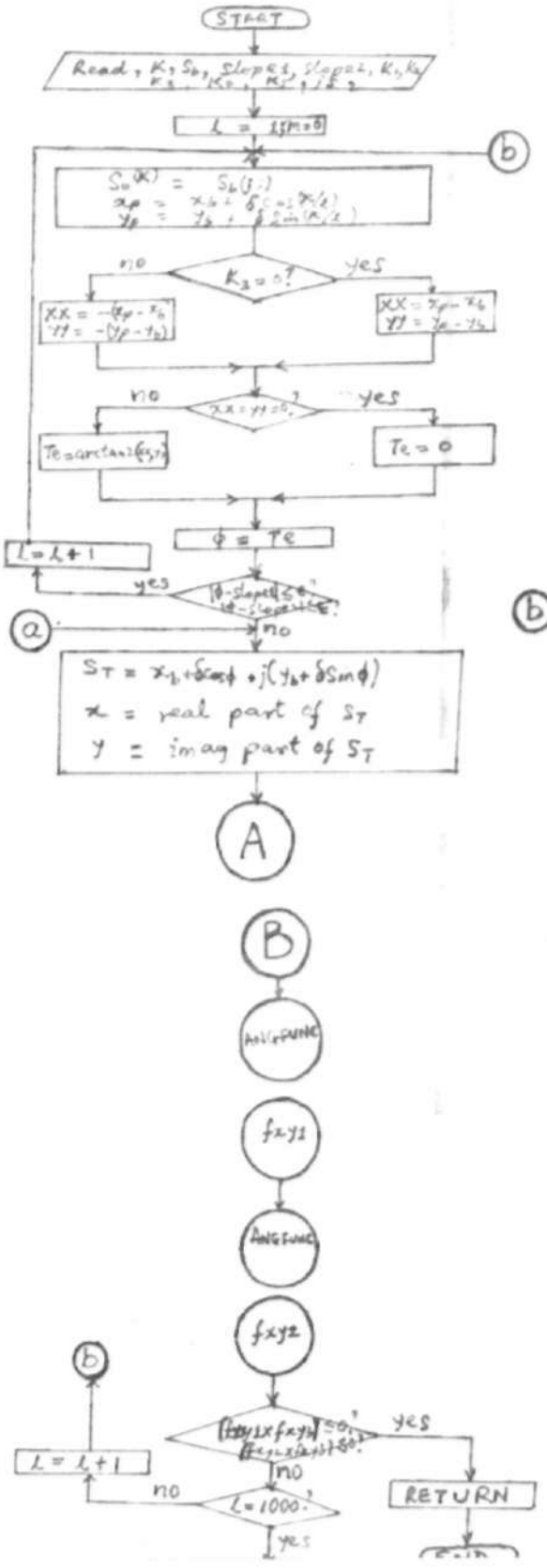
FLOW CHART FOR ROUTINE DIREGI

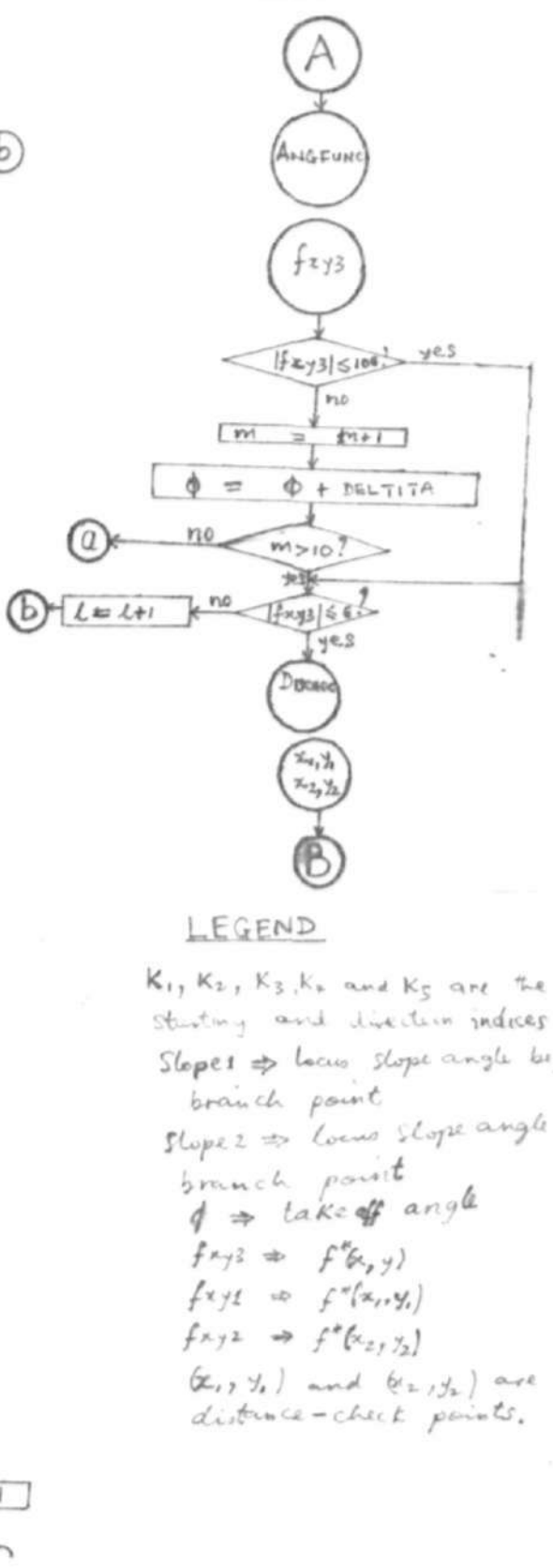



A
$K_2 = 1$?
yes
$K_3 = 1$
no
$K_1 = 1$?
yes
$K_3 = 0$
no
$K_1 = K_2$ ?
yes
$K_3 = n_1$
no
EXIT
no
$n_1 = n_2$ ?
yes
$n_1 \leq 1$ ?
$K_x = K_y = 1$
$K_x = n_1 - 1$ ?
yes
$K_y = n_2 - 1$ ?
yes
$K_y = K_y + 1$
$n_2 < 2$
no
$K_x = 2$
$K_3 = 1$
C
no
$K_x = K_x + 1$
B


B
C
$n_1 < 1$?
yes
no
$K_x = 1$
$K_3 = 1$
yes
$K_x = K_x + 1$
$K_x = n_1$ ?
$P_i, t_i, K_1, K_2, K_3, K_4, K_5$
RETURN
END

# FLOW CHART FOR ROUTINE TAKEANG







## LEGEND

$K_1, K_2, K_3, K_4$ and $K_5$ are the starting and direction indices

Slope1 ⇒ locus slope angle before a branch point

Slope2 ⇒ locus slope angle after a branch point

$\phi$ ⇒ take off angle

$fxy3 \Rightarrow f''(x, y)$

$fxy1 \Rightarrow f''(x_1, y_1)$

$fxy2 \Rightarrow f''(x_2, y_2)$

$(x_1, y_1)$ and $(x_2, y_2)$ are the distance-check points.

# A P P E N D I X  B

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| -1.1261 | 2.0152 | 2446.7656 |
| -1.1261 | 2.0152 | 2446.7656 |
| -1.1261 | 2.0152 | 2446.7656 |

BRANCH NUMBER 2

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| -4.4688 | 1 | 27.5781 |
| -3.9000 | 1 | 55.6254 |
| -3.7500 | 1 | 65.9694 |
| -3.[illegible] | 1 | [illegible] |
| -3.2500 | 1 | 89.7656 |
| -3.[illegible] | 1 | 116.8754 |
| -3.0938 | 0 | 263.8469 |
| -2.[illegible] | 1 | 417.[illegible] |
| -2.[illegible] | 1 | 619.4931 |
| -2.[illegible] | 1 | 913.[illegible] |
| -1.[illegible] | 1 | 1272.7344 |
| -1.[illegible] | 1 | 1740.0000 |
| -1.2500 | 1 | 2316.6406 |
| -1.[illegible] | 0 | 3014.3750 |
| -0.[illegible] | 1 | 3844.9219 |
| -0.[illegible] | 1 | 4820.0000 |
| -0.[illegible] | 0 | 5951.3281 |
| -0.[illegible] | 1 | 7251.8438 |
| -0.2500 | 0 | 8729.6094 |

BRANCH NUMBER 2

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| -5.[illegible] | 4.[illegible] | [illegible] |
| -4.1372 | 3.0992 | 46.1113 |
| -3.9766 | 2.[illegible] | 68.4451 |
| -2.0025 | 3.6028 | 101.8563 |
| -1.2253 | 4.5816 | 169.7861 |
| -0.5161 | 5.6114 | 284.6640 |
| -.1035 | 6.6595 | 458.2652 |
| .8107 | 7.719[illegible] | 7[illegible].3141 |
| 1.4661 | 8.7847 | 1028.5288 |
| 2.1209 | 9.8543 | 1448.6265 |
| [illegible].7933 | [illegible].9268 | 1974.3540 |
| 3.4685 | [illegible].0268 | 2617.3411 |
| [illegible].[illegible]32 | [illegible]3.[illegible]765 | 3389.3939 |
| 4.6862 | [illegible]4.1532 | 4302.2113 |
| 5.3137 | [illegible]5.2301 | 5367.4626 |
| 5.9466 | [illegible]6.3091 | 6590.9533 |
| 6.5771 | [illegible]7.3876 | 8[illegible].3349 |
| 7.[illegible]84 | [illegible]8.4671 | 9595.3448 |
| 7.8382 | [illegible]9.5467 | 11387.7020 |
| 8.4677 | [illegible]0.6266 | 12391.[illegible]48 |

List of computed points for System 91

BRANCH NUMBER 2

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| [illegible] | [illegible] | [illegible] |
| -4.3401 | 1.9348 | 29.6065 |
| -3.5987 | .9377 | 38.7559 |
| -2.6481 | -.4426 | 42.7471 |
| -1.9735 | 2.4928 | 61.1872 |
| -1.3301 | 3.5644 | 106.2391 |
| -.6954 | 4.6413 | 189.6206 |
| -.1644 | 5.7284 | 323.1511 |
| .5946 | 6.8305 | 518.2461 |
| 1.1925 | 7.8814 | 786.9274 |
| 1.8198 | 8.9626 | 1140.8126 |
| 2.4465 | 10.0441 | 1591.6204 |
| 3.[illegible]73 | [illegible]1.1258 | 2151.6876 |
| 3.6998 | [illegible]2.2078 | 2830.8789 |
| 4.3256 | [illegible]3.2897 | 3642.7671 |
| 4.9518 | [illegible]4.37[illegible]8 | 4598.453[illegible] |
| 5.5766 | [illegible]5.4539 | 5709.6552 |
| 6.2022 | [illegible]6.5361 | 6988.1925 |
| 6.8277 | [illegible]7.6183 | 8445.4836 |
| 7.4532 | [illegible]8.7005 | 10093.5474 |

BRANCH NUMBER 2

| REAL PART | IMAG PART | GAIN |
|---|---|---|
| -5.[illegible] | -3.[illegible] | [illegible] |
| -4.340[illegible] | -[illegible].9348 | 29.6[illegible]5 |
| -3.5987 | -.9378 | 38.7559 |
| -2.6481 | -1.4426 | 42.7471 |
| -1.9735 | -2.4928 | 61.1872 |
| -1.3301 | -3.5644 | 106.2391 |
| -.6954 | -4.6413 | 189.6206 |
| -.1644 | -5.7284 | 323.1511 |
| .5946 | -6.8[illegible]5 | 518.2461 |
| 1.1925 | -7.8814 | 786.9274 |
| 1.8198 | -8.9626 | 1140.8126 |
| 2.4465 | [illegible]0.0441 | 1591.6204 |
| 3.[illegible]73 | [illegible]1.1258 | 2151.6876 |
| 3.6998 | [illegible]2.2078 | 2830.8789 |
| 4.3256 | [illegible]3.2897 | 3642.7671 |
| 4.9518 | [illegible]4.37[illegible]8 | 4598.453[illegible] |
| 5.5766 | [illegible]5.4539 | 5709.6552 |
| 6.2022 | [illegible]6.5361 | 6988.1925 |
| 6.8277 | [illegible]7.6183 | 8445.4836 |
| 7.4532 | [illegible]8.7005 | 10093.5474 |

List of computed points for System 92

# B I B L I O G R A P H Y

## Bibliography


1. Truxal, J.G., Control System Synthesis, New York, McGraw-Hill, 1955.

2. Thaler, G.J., Design of Feedback Systems, Pensylvania, DH and R, 1973.

3. Kuo, B.C., Automatic Control Systems, New Jersey, Prentice Hall, 1967.

4. Thaler, G.G. and Brown, R.B., Analysis and Design of Feedback Systems, New York, McGraw-Hill, 1960.

5. Shinners, S.M., Modern Control System Theory and Applications, New York, Addison Wesley, 1972.

6. Carlson, A. and Fredrick, B., Linear Systems in Communication and Control, New York, John Wiley, 1971.

7. Eveleigh, V.W., Introduction to Control System Design, New York, McGraw-Hill, 1972.

8. Witts, C.H., Principles of Feedback Control, Reading, Addison-Wesley, 1960.

9. Monro, D.M., Computing with Fortran, London, Edward Arnolds, 1977

10. Fenner, R.T., Computing for Engineers, London, Macmillan, 1974.

11. Ash, R.H. and Ash, G.R., "Numerical Computation of Root Loci Using Newton Raphson techniques", IEET on AC, Oct. 1968, pp. 576 - 586.

12. Power, H.M., "Root Loci Having a Total of Four Poles and Zeros", IEET on AC, Oct. 1971, pp. 484 - 486.

13. Mitchell, J.R. and McDaniel, W.L. (Jr), "A generalized Root Locus Following Technique", IEET on AC, Aug. 1970, pp. 483 - 486.

14. Pan, C.T. and Chao, K.S. "A Computer Aided Root Locus Method", IEET on AC, Vol. 23, No.5, Oct. 1978, pp. 856 - 860.

15. Power, H.M., "Application of Bilinear Transformation to Root Locus Plotting", IEET on AC, Dec. 1970 pp. 693 - 694.

16. Koziol, J.S. (Jr), "An Approximate Root Locus Method in the s - plane for Sampled Data Systems", IEET on Ac, Feb. 1971, pp. 101 - 102.

17. Bollinger, K.E. and Mathur, J.C., "To Compute Zeros of Large Systems", IEET on AC, Feb. 1971, pp. 95 - 96.

18. Kocourek, C.J. and Northouse, R.A., "Unsupervised Root Locus Gain Selection", Int. J. Control, 1974, Vol. 19, No.3, pp. 517 - 527.

19. Fortescue, P.W., "The Generalized Root Locus (GRL) method (For Similar Systems with Antisymmetric Coupling)", Int. J. Control, 1976, Vol.24, No.4, pp. 477 - 492.

20. Ralston, A., A First Course in Numerical Analysis, New York, McGraw-Hill, 1965.

21. Hamming, R.W., Numerical Methods for Scientists and Engineers, N.Y., McGraw-Hill, 1962.

22. Krishnan, V., "Semi-Analytic Approach to Root Locus", IEET on AC, 1966, Vol. AC - 11, No.1, pp. 101 - 108.

23. Steilitz, K., "An Analytical Approach to Root Loci", IRET on AC, Sept. 1961, pp. 326 - 340.

*24. Bhaattacharyya, B.P., "Root Locus Equation of Fourth Degree".

* Could not identify the year of publication nor Journal that published the article from the photostat copies sent to me.