MULTIPLE ZEROS OF POLYNOMIALS

by

CRAIG A. WOOD

Department of Mathematics
Oklahoma State University
Stillwater, Oklahoma

A National Aeronautics and Space Administration

Research Grant

Grant Number NASA NGR 37-002-084

MULTIPLE ZEROS OF POLYNOMIALS

by

CRAIG A. WOOD

Department of Mathematics
Oklahoma State University
Stillwater, Oklahoma

A National Aeronautics and Space Administration

Research Grant

Grant Number NASA NGR 37-002-084

Abstract: MULTIPLE ZEROS OF POLYNOMIALS

Grant: The National Aeronautics and Space Administration

Grant Number: NASA NGR 37-002-084

Grantee: Professor Craig A. Wood
Department of Mathematics
Oklahoma State University
Stillwater, Oklahoma

Various classical methods exist for extracting the zeros of a polynomial

$$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_{N+1}$$

where $a_1 \neq 0$ and $a_1, a_2, \ldots, a_{N+1}$ are complex numbers, when $N=1,2,3,4$. For polynomials of higher degree, iterative numerical methods must be used. In this material four iterative methods are presented for approximating the zeros of a polynomial using a digital computer. Newton's method and Muller's method are two well known iterative methods which are presented. They extract the zeros of a polynomial by generating a sequence of approximations converging to each zero. However, both of these methods are very unstable when used on a polynomial which has multiple zeros. That is, either they fail to converge to some or all of the zeros, or they converge to very bad approximations of the polynomial's zeros.

This material introduces two new methods, the greates common divisor (G.C.D.) method and the repeated greatest common divisor (repeated G.C.D.) method, which are superior methods for numerically approximating the zeros of a polynomial having multiple zeros.

The above methods were all programmed in FORTRAN IV and comparisons in time and accuracy are given. These programs were executed on the

IBM 360/50 computer as well as the UNIVAC 1108 and the CDC 6600 computer.

This material also contains complete documentations for six FORTRAN IV programs. Flow charts, program listings, definition of variables used in the program, and instructions for use of each program are included.

PREFACE

Four iterative methods for approximating the zeros of a polynomial using a digital computer are presented in this material. Chapter I is an introduction. Chapters II and III contain Newton's and Muller's methods, respectively. Chapters IV and V present two new methods which depend upon finding the greatest common divisor of two polynomials. Chapter VI contains a comparison of the four methods. Flow charts, FORTRAN IV programs, and complete program documentations for these four methods are presented in appendicies A through H.

I would like to express my appreciation to the National Aeronautics and Space Administration,specifically the Manned Spacecraft Center in Houston, Texas, for their financial support in making this work possible under grant number NASA NGR 37-002-084. I would also like to thank Randy Snider, a graduate assistant supported by this grant, for the great deal of work he put in on the FORTRAN programs. In particular, the material on Newton's and Muller's Methods included in this paper is part of his masters thesis at Oklahoma State University.

TABLE OF CONTENTS

Chapter Page

Chapter Page

CHAPTER I

INTRODUCTION

Frequently in scientific work it becomes necessary to find the zeros, real or complex, of the polynomial of degree N

$$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1}$$

where $a_1 \neq 0$ and the coefficients $a_1, a_2, \ldots, a_{N+1}$ are complex numbers. Various classical methods calculate the exact roots of polynomials of degree 1,2,3, or 4. For polynomials of higher degree, no such methods exist. Thus, to solve for the zeros of such polynomials, numerical methods of iteration based on successive approximations must be employed. In the following material four such methods are given which are particularly suited for modern high speed computers.

Newton's method is an iterative procedure which generates a sequence of successive approximations of a zero of P(X) by using the iteration formula

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$

An initial approximation to the zero is required to start the iterative process. Under certain conditions this sequence will converge quadratically to the desired root. It is, however, necessary to compute the value of the polynomial and its derivative for each step in the

iterative procedure. Once a zero of P(X) has been found, it is divided out of P(X), giving a deflated polynomial of lower degree. P(X) is replaced by the deflated polynomial and the iterative process is applied to extract another zero of P(X). This procedure is repeated until all zeros of P(X) have been found. The zeros may then be rechecked and their accuracy possibly improved by using them as initial approximations with Newton's process applied to the full (undeflated) polynomial.

Muller's method is also an iterative procedure generating a sequence $X_1, X_2, \ldots, X_n, \ldots$ of successive approximations of a root of P(X). This method converges almost quadratically near a zero and does not require the evaluation of the derivative of the polynomial. Muller's method requires three distinct approximations of a root to start the process of iteration. A quadratic equation is constructed through the three given points as an approximation of P(X). The root of the quadratic closest to X_n is taken as X_{n+1}, the next approximation to the zero. This process is then repeated on the last three points of the sequence. After a root of P(X) has been found, P(X) is deflated, and replaced in the above procedure by the deflated polynomial. After all zeros of P(X) are found from successive deflations, they are improved as in Newton's method.

The greatest common divisor method reduces the problem of finding all zeros (possibly multiple zeros) of P(X) to one of extracting the zeros of a polynomial $P_1(X) = P(X)/D(X)$, all of whose zeros are simple. D(X), the greatest common divisor of P(X) and its derivative, $P'(X)$, is obtained by repeated application of the division algorithm. Once $P_1(X)$ is obtained, some suitable method such as Newton's or Muller's method

is used to find the zeros of $P_1(X)$. By finding all the zeros of $P_1(X)$, all the zeros of P(X) are obtained. The multiplicity of each zero may then be determined.

The repeated greatest common divisor method repeatedly uses the greatest common divisor method to extract the zeros of P(X) and their multiplicities at the same time. That is, the repeated greatest common divisor method reduces the problem of finding the zeros of P(X), which possibly has multiple zeros, to one of finding the zeros of a polynomial which has only simple zeros and the zeros of this polynomial are all the zeros of P(X) of a given multiplicity. The repeated greatest common divisor method must also use a supporting method such as Newton's method or Muller's method.

Chapters II-V contain the examinations of these methods. Each examination includes a development of the method together with the conditions necessary for convergence of the method. Chapter VI contains a comparison of the methods giving advantages and disadvantages of each method.

A complete set of documentations is given for six FORTRAN IV programs in Appendices A-H. Flow charts, program listings, definition of variables used in the program, and instructions for use of each program are included.

It should also be noted that the expressions "zero of a polynomial" and "root of a polynomial" and the words "zero" and "root" are used interchangeably in this material.

CHAPTER II

NEWTON'S METHOD

1. Derivation of the Algorithm

Newton's method is probably the most popular iterative procedure for finding the zeros of a polynomial. This fact is due to the excellent results obtained, the simplicity of the computational routine, and the fast rate of convergence obtained provided the initial approximation of a zero is close enough. Also, the method can be applied to the extraction of complex as well as real zeros.

Consider the polynomial

$$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1} \qquad (2\text{-}1)$$

where $a_1 \neq 0$ and the coefficients $a_1, a_2, \ldots, a_{N+1}$ are complex. The algorithm for Newton's method can be derived by approximating $P(X)$ by a Taylor series expansion about an approximation, X_0, of a zero, α, of $P(X)$. Using only the first two terms of the expansion, the expression

$$P(X) \doteq P(X_0) + P'(X_0)(X - X_0)$$

is obtained. If this equation is solved for $P(X) = 0$, then

$$0 \doteq P(X_0) + P'(X_0)(X - X_0)$$

results. Rearranging terms produces

$$0 \doteq P(X_0) + P'(X_0)\ X - P'(X_0)\ X_0$$

followed by

$$P'(X_0)\ X_0 - P(X_0) \doteq P'(X_0)\ X$$

from which division by $P'(X_0)$ produces

$$X_0 - P(X_0)/P'(X_0) \doteq X$$

which is the basic formula for Newton's method. Thus, in general, we obtain the $(n+1)^{th}$ approximation, X_{n+1}, of α from the n^{th} approximation, X_n, by

$$X_{n+1} = X_n - P(X_n)/P'(X_n). \tag{2-2}$$

As a result of repeated use of this algorithm, we obtain the sequence

$$X_0, X_1, X_2, \ldots, X_n, \ldots \tag{2-3}$$

of successive approximations of the root, α. It should be noted that an initial approximation is necessary to start the iterative process for each new zero; that is, a polynomial of degree N may require N initial approximations.

In order to use equation (2-2), it is necessary to compute, for each X_n, the value of the polynomial, $P(X_n)$, and its derivative, $P'(X_n)$. The division algorithm states that if P(X) and G(X) are polynomials, then there exists polynomials H(X) and K(X) such that $P(X) = H(X)\ G(X) + K(X)$ where $K(X) = 0$ or deg. $K(X) <$ deg. $G(X)$. From this expression of P(X), the following remainder theorem is obtained:

Theorem 2.1. If $P(X)$ is a polynomial and c is a complex number, then the remainder obtained from dividing $P(X)$ by $(X - c)$ is $P(c)$.

The proof of Theorem 2.1 is given in [3, P. 102]. Thus, $P(X)$ can be written as $P(X) = (X - c)\,H(X) + R$ where $P(c) = R$. $P'(X)$ is then obtained by the following theorem, the proof of which can be found in [3, PP. 105-106].

Theorem 2.2. If $P(X)$ and $H(X)$ are polynomials and c is a complex number such that $P(X) = (X - c)\,H(X) + R$ where $P(c) = R$, then the remainder obtained from dividing $H(X)$ by $(X - c)$ is $P'(c)$.

From synthetic division, an algorithm known as Horner's Method is acquired for computing $P(X_n)$ and $P'(X_n)$.

Theorem 2.3. Let $P(X)$ be defined as in equation (2-1) and let d be a complex number. Define a sequence $b_1, b_2, \ldots, b_{N+1}$ by

$$b_1 = a_1$$

$$b_i = a_i + db_{i-1} \quad (i = 2,3,\ldots,N+1).$$

Define another sequence $c_1, c_2, \ldots, c_N$ by

$$c_1 = b_1$$

$$c_j = b_j + dc_{j-1} \quad (j = 2,3,\ldots,N).$$

Then $P(d) = b_{N+1}$ and $P'(d) = c_N$. The elements $b_1, b_2, \ldots, b_N$ are the coefficients of the polynomial $H(X)$ in Theorem 2.2 when $P(X)$ is divided by $(X - d)$.

These formulas are derived in [3, PP. 106-107]. Thus with equation (2-2) and the iteration formulas of the previous theorem, Newton's method can now be applied to generate the sequence (2-3) which will converge to the root, α, if the convergence conditions given in Theorem 2.4 are satisfied.

A criterion is needed to determine when to terminate the sequence (2-3); that is, when has a zero been found? For convergence of the sequence, there must exist a term in the sequence beyond which the difference between any two successive terms is arbitrarily small. Therefore, it is desirable to make the quotient $|X_n/X_{n+1}|$ sufficiently near 1. From equation (2-2)

$$1 = \left| \frac{X_n}{X_{n+1}} - \frac{\frac{P(X_n)}{P'(X_n)}}{X_{n+1}} \right|$$

$$\geq \left| \frac{X_n}{X_{n+1}} \right| - \left| \frac{\frac{P(X_n)}{P'(X_n)}}{X_{n+1}} \right| .$$

Thus

$$1 + \frac{\left| \frac{P(X_n)}{P'(X_n)} \right|}{\left| X_{n+1} \right|} \geq \left| \frac{X_n}{X_{n+1}} \right|$$

where $P'(X_n)$ and $X_{n+1} \neq 0$. Thus, iterations are continued until an X_n is obtained such that $|P(X_n)/P'(X_n)| \Big/ |X_{n+1}|$ is as small as desired.

After a zero, α, of $P(X)$ has been found, the term $(X - \alpha)$ is synthetically divided out of $P(X)$ by deflation using Theorem 2.3 obtaining

a polynomial, $P_1(X)$, of degree N-1. The root finding process is then repeated to extract a zero, α_1, of $P_1(X)$. P(X) can be written as

$$P(X) = (X - \alpha)\, P_1(X) + R$$

where $R = P(\alpha)$. But $P(\alpha) = 0$. Therefore, substitution produces

$$P(X) = (X - \alpha)\, P_1(X).$$

Now $P_1(\alpha_1) = 0$ implies that $P(\alpha_1) = 0$. Hence, α_1 is a zero of P(X).

By the process of root finding and successive deflations, zeros $\alpha_0, \alpha_1, \ldots, \alpha_{N-1}$ of the deflated polynomials

$$P(X) = P_0(X), P_1(X), \ldots, P_{N-1}(X),$$

respectively, are extracted. Each α_i $(i = 0,1,2,\ldots,N-1)$ is a zero of P(X) since each α_i is a zero of $P_{i-1}(X), P_{i-2}(X), \ldots, P_1(X), P(X)$.

After all zeros of P(X) have been found, it may be possible to improve their accuracy by using them as initial approximations with Newton's method applied to the full (undeflated) polynomial, P(X). This should correct any loss of accuracy which may have resulted from the successive deflations.

2. Convergence of Newton's Method

The following theorem from [2, PP. 79-81] gives sufficient conditions for the convergence of sequence (2-3).

Theorem 2.4. Let P(X) be a polynomial and let the following conditions be satisfied on the closed interval [a,b]:

1. $P(a)\,P(b) < 0$
2. $P'(X) \neq 0$, $X \in [a,b]$
3. $P''(X)$ is either ≥ 0 or ≤ 0 for all $X \in [a,b]$
4. If c denotes the endpoint of [a,b] at which $|P'(X)|$ is smaller, then $|P(c)/P'(c)| \leq b - a$.

Then Newton's method converges to the (only) solution, s, of $P(X) = 0$ for any choice of X_0 in [a,b].

When convergence is obtained, it is quadratic; that is,

$$e_{i+1} = \frac{1}{2} F''(\eta_i)\, e_i^2$$

where $F(X_i) = X_i - P(X_i)/P'(X_i)$, η_i is between X_i and the zero, α, and e_i is the error in X_i. This means that the error obtained in the $(i+1)^{th}$ iteration of Newton's algorithm is proportional to the square of the error obtained in the i^{th} iteration. A proof of quadratic convergence can be found in [1, PP. 31-33].

3. Procedure for Newton's Method

The general procedure for applying Newton's method is enumerated sequentially as follows, starting with initial approximation X_0:

1. Calculate a new approximation X_{n+1} by
$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$
2. Test for convergence; that is, test
$$|P(X_n)/P'(X_n)| \Big/ |X_{n+1}| < \varepsilon$$
for some ε chosen as small as desired.
3. If convergence is obtained, perform the following:

a. Save X_{n+1} as the desired approximation to a zero of P(X).

b. Deflate P(X) using X_{n+1}.

c. Replace P(X) by the deflated polynomial.

d. Return to step 1 with a new initial approximation.

4. If no convergence is obtained, increase n by 1 and return to step 1.

In order to prevent an unending iteration process in case the method does not produce convergence, a maximum number of iterations should be specified. If convergence is not obtained within this number of iterations, change the initial approximation and return to step 1 above.

4. Geometrical Interpretation of Newton's Method

A geometrical interpretation of Newton's method is given in Figure 2.1. X_i is an approximation to the zero, α. $P'(X_i)$ is the slope of the line tangent to P(X) at X_i. X_{i+1} is the intersection of the tangent line with the x axis.

5. Determining Multiple Roots

If P(X) has m distinct zeros, then P(X) can be written as

$$P(X) = a_1(X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \ldots (X - \alpha_m)^{e_m}, \quad (m \leq N)$$

where α_i is a zero of P(X) and e_i is the multiplicity of α_1 $(i = 1,2,\ldots,m)$. Consider the root α_j. Dividing out the term

$(X - \alpha_j)$ by deflating $P(X)$ gives $P_1(X)$ of degree N-1 which can be written as

$$P_1(X) = (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \ldots (X - \alpha_j)^{e_j - 1} \ldots (X - \alpha_m)^{e_m}.$$

Evaluating $P_1(X)$ at the zero, α_j, gives $P_1(\alpha_j) = 0$ if $e_j > 1$. Thus, after a zero, α, of $P(X)$ is determined by Newton's iterative process and the current polynomial is deflated giving $P_1(X)$, then $P_1(\alpha)$ is evaluated. If $P_1(\alpha) \leq \varepsilon$ for some small number ε, α is a root of $P_1(X)$ and thus has multiplicity at least equal to two. $P_1(X)$ is then deflated giving $P_2(X)$. If $P_2(\alpha) \leq \varepsilon$, α is of multiplicity at least three. This process is continued until a deflated polynomial $P_k(X)$ is encountered such that either deg. $P_k(X) = 0$ or $P_k(\alpha) > \varepsilon$. α is then a zero of multiplicity k+1.

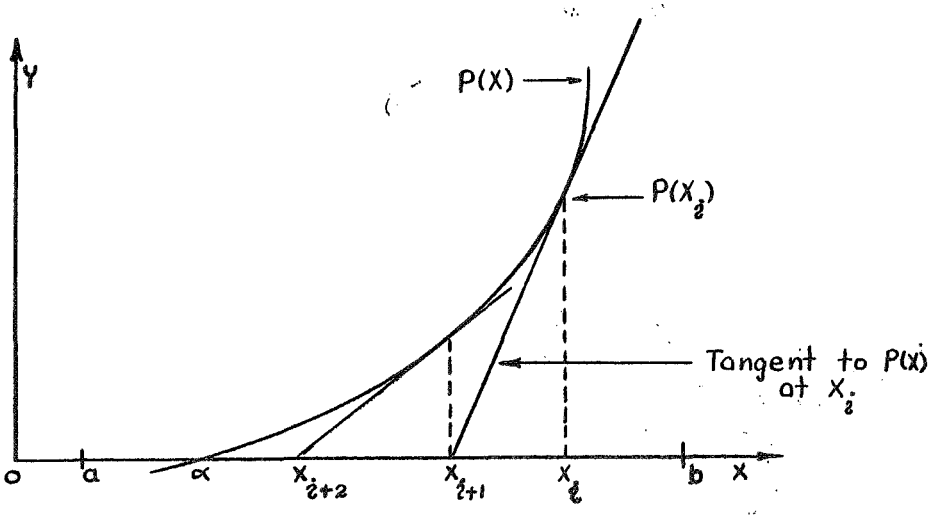


Figure 2.1. Geometrical Interpretation of Newton's Method

CHAPTER III

MULLER'S METHOD

1. Derivation of the Algorithm

Muller's method in [4] is an iterative procedure designed to find any prescribed number of zeros, real or complex, of a polynomial. The method does not require the evaluation of the derivative and near a zero the convergence is almost quadratic.

Consider the polynomial

$$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1} \tag{3-1}$$

with complex coefficients such that $a_1 \neq 0$. Given three distinct approximations, X_{n-2}, X_{n-1}, X_n, to a root, α, of $P(X)$, the problem is to determine X_{n+1} in such a way as to generate a sequence

$$X_1, X_2, X_3, \ldots, X_n, X_{n+1}, \ldots \tag{3-2}$$

of approximations converging to α. The points $(X_{n-2}, P(X_{n-2}))$, $(X_{n-1}, P(X_{n-1}))$, and $(X_n, P(X_n))$ determine a unique quadratic polynomial, $Q(X)$, approximating $P(X)$ in the vicinity of X_{n-2}, X_{n-1}, X_n. A general proof of this can be found in [2, PP. 133-134]. Thus, the zeros of $Q(X)$ will be approximations of the zeros of $P(X)$ in this region of approximation. From the general representation in [2, P. 184] of the Legrangian interpolating polynomial, the representation of $Q(X)$ is given by

$$Q(X) = \frac{(X - X_{n-1})(X - X_{n-2})}{(X_n - X_{n-1})(X_n - X_{n-2})} P(X_n)$$

$$+ \frac{(X - X_n)(X - X_{n-2})}{(X_{n-1} - X_n)(X_{n-1} - X_{n-2})} P(X_{n-1})$$

$$+ \frac{(X - X_n)(X - X_{n-1})}{(X_{n-2} - X_n)(X_{n-2} - X_{n-1})} P(X_{n-2})$$

which can be rewritten as

$$Q(X) = Q(X - X_n + X_n)$$

$$= \frac{(X - X_n + X_n - X_{n-1})(X - X_n + X_n - X_{n-1} + X_{n-1} - X_{n-2})}{(X_n - X_{n-1})(X_n - X_{n-1} + X_{n-1} - X_{n-2})} P(X_n)$$

$$- \frac{(X - X_n)(X - X_n + X_n - X_{n-1} + X_{n-1} - X_{n-2})}{(X_n - X_{n-1})(X_{n-1} - X_{n-2})} P(X_{n-1})$$

$$+ \frac{(X - X_n)(X - X_n + X_n - X_{n-1})}{(X_n - X_{n-1} + X_{n-1} - X_{n-2})(X_{n-1} - X_{n-2})} P(X_{n-2}).$$

In order to simplify this expression, introduce the quantities

$$h_n = X_n - X_{n-1}, \quad h = X - X_n.$$

Then

$$Q(X) = Q(X_n + h)$$

$$= \frac{(h + h_n)(h + h_n + h_{n-1})}{h_n(h_n + h_{n-1})} P(X_n)$$

$$- \frac{h(h + h_n + h_{n-1})}{h_n h_{n-1}} P(X_{n-1})$$

$$+ \frac{h(h + h_n)}{(h_n + h_{n-1})h_{n-1}} P(X_{n-2})$$

$$= \frac{h^2 + 2hh_n + hh_{n-1} + h_n^2 + h_n h_{n-1}}{h_n^2 + h_n h_{n-1}} P(X_n)$$

$$- \frac{h^2 + hh_n + hh_{n-1}}{h_n h_{n-1}} P(X_{n-1})$$

$$+ \frac{h^2 + hh_n}{h_n h_{n-1} + h_{n-1}^2} P(X_{n-2}).$$

Collecting terms containing like powers of h produces

$$Q(X) = Q(X_n + h)$$

$$= \left(\frac{P(X_n)}{h_n^2 + h_n h_{n-1}} - \frac{P(X_{n-1})}{h_n h_{n-1}} + \frac{P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h^2$$

$$+ \left(\frac{(2h_n + h_{n-1})\, P(X_n)}{h_n^2 + h_n h_{n-1}} - \frac{(h_n + h_{n-1})\, P(X_{n-1})}{h_n h_{n-1}} + \frac{h_n P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h$$

$$+ \frac{h_n(h_n + h_{n-1})\, P(X_n)}{h_n^2 + h_n h_{n-1}}$$

$$= \left(\frac{P(X_n)\, h_{n-1}}{h_n^2 h_{n-1} + h_n h_{n-1}^2} - \frac{P(X_{n-1})}{h_n h_{n-1}} + \frac{P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h^2$$

$$+ \left(\frac{(2h_n h_{n-1} + h_{n-1}^2)\, P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2} - \frac{(h_n + h_{n-1}) P(X_{n-1})}{h_n h_{n-1}} + \frac{h_n\, P(X_{n-2})}{h_n h_{n-1} + h_{n-1}^2} \right) h$$

$$+ \frac{(h_n^2 h_{n-1} + h_{n-1}^2 h_n)\, P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2} \quad .$$

Using the common denominator, $h_n^2 h_{n-1} + h_n h_{n-1}^2$, and combining terms yields

$$Q(X_n + h) = \left(\frac{P(X_n)\, h_{n-1} - P(X_{n-1})(h_n + h_{n-1}) + P(X_{n-2})\, h_n}{h_n^2 h_{n-1} + h_n h_{n-1}^2} \right) h^2$$

$$+ \left(\frac{(2h_n h_{n-1} + h_{n-1}^2)\, P(X_n) - (h_n + h_{n-1})^2\, P(X_{n-1}) + h_n^2 P(X_{n-2})}{h_n^2 h_{n-1} + h_n h_{n-1}^2} \right) h$$

$$+ \frac{(h_n^2 h_{n-1} + h_{n-1}^2 h_n)\, P(X_n)}{h_n^2 h_{n-1} + h_n h_{n-1}^2} \quad .$$

Multiplying by h_n / h_{n-1}^2 results in

$$Q(X_n + h) = \left[\frac{P(X_n) \frac{h_n}{h_{n-1}} - P(X_{n-1}) \left(\left(\frac{h_n}{h_{n-1}} \right)^2 + \frac{h_n}{h_{n-1}} \right) + P(X_{n-2}) \left(\frac{h_n}{h_{n-1}} \right)^2}{\frac{h_n^3}{h_{n-1}} + h_n^2} \right] h^2$$

$$+ \left[\frac{\left(2\frac{h_n^2}{h_{n-1}} + h_n \right) P(X_n) - h_n \left[\left(\frac{h_n}{h_{n-1}} \right) + \left(\frac{h_{n-1}}{h_{n-1}} \right) \right] P(X_{n-1}) + \frac{h_n^3}{h_{n-1}^2} P(X_{n-2})}{\frac{h_n^3}{h_{n-1}} + h_n^2} \right] h$$

$$+\left[\frac{\left(\frac{h_n^3}{h_{n-1}} + h_n^2\right) P(X_n)}{\frac{h_n^3}{h_{n-1}} + h_n^2}\right].$$

Let $q_n = \frac{h_n}{h_{n-1}}$ and $q = \frac{h}{h_n}$. Then

$$Q(X_n + h) = \left(\frac{P(X_n)\, q_n - P(X_{n-1})(q_n^2 + q_n) + P(X_{n-2})\, q_n^2}{q_n+1}\right) q^2$$

$$+ \left(\frac{(2\, q_n+1)\, P(X_n) - (q_n+1)^2\, P(X_{n-1}) + q_n^2\, P(X_{n-2})}{q_n+1}\right) q$$

$$+ \frac{(q_n+1)\, P(X_n)}{q_n+1}.$$

Now let

$$A_n = q_n\, P(X_n) - q_n(q_n+1)\, P(X_{n-1}) + q_n^2\, P(X_{n-2})$$

$$B_n = (2q_n+1)\, P(X_n) - (q_n+1)^2\, P(X_{n-1}) + q_n^2\, P(X_{n-2})$$

$$C_n = (q_n+1)\, P(X_n).$$

Then

$$Q(X_n + h) = Q(X_n + qh_n)$$

and

$$Q(X_n + qh_n) = \frac{A_n q^2 + B_n q + C_n}{q_n+1}.$$

Solving the quadratic equation $Q(X_n + qh_n) = 0$ and denoting the result by q_{n+1} gives:

$$q_{n+1} = \frac{-B_n \pm \sqrt{B_n^2 - 4A_nC_n}}{2A_n}$$

and the new approximation is found as follows:

$$q_{n+1} = \frac{h_{n+1}}{h_n} = \frac{X_{n+1} - X_n}{h_n} .$$

Thus

$$X_{n+1} = X_n + h_n q_{n+1}$$

In order to avoid loss of accuracy, q_{n+1} can be written in a better form as follows:

$$q_{n+1} = \frac{-B_n \pm \sqrt{B_n^2 - 4A_nC_n}}{2A_n} \cdot \frac{B_n \pm \sqrt{B_n^2 - 4A_nC_n}}{B_n \pm \sqrt{B_n^2 - 4A_nC_n}}$$

$$= \frac{-B_n^2 + B_n^2 - 4A_nC_n}{2A_n\,(B_n \pm \sqrt{B_n^2 - 4A_nC_n})}$$

$$q_{n+1} = \frac{-2C_n}{B_n \pm \sqrt{B_n^2 - 4A_nC_n}} . \qquad (3\text{-}3)$$

The sign in the denominator should be chosen such that the magnitude of the denominator is largest, thus causing $|q_{n+1}|$ to be smallest. This, in turn, will make X_{n+1} closest to X_n.

Note that each iteration of this process requires three approximations, X_{n-2}, X_{n-1}, X_n, in order to compute X_{n+1}. Thus, when X_{n+1} is found, X_{n-1}, X_n, X_{n+1} are used to compute X_{n+2}; that is, the last three terms of the generated sequence are used to compute the next term.

Convergence of the sequence (3-2) to a zero is obtained when the elements X_k and X_{k+1} of the sequence are found such that

$$\frac{|X_{k+1} - X_k|}{|X_{k+1}|} < \varepsilon, \quad X_{k+1} \neq 0;$$

that is, the ratio of the change in the approximation to the approximation itself is as small as desired.

In order to use the iterative formulas, it is necessary to compute the value, $P(X_j)$, of the polynomial $P(X)$ at the approximation X_j. The procedure for doing this is discussed in Chapter II, § 1. The iteration formulas are given in Theorem 2.3 of Chapter II.

After a zero, α, of $P(X)$ has been found, $P(X)$ is deflated as described in Chapter II, § 1, and the process repeated to extract a zero, α_1, of $P_1(X)$. By applying Muller's method to successively deflated polynomials, all the zeros of $P(X)$ are obtained. For more detailed discussion of this procedure see Chapter II, § 1, keeping in mind that Muller's instead of Newton's method is used.

Muller's method requires three initial approximations to a zero in order to start the iteration process. If three are not known, the values $X_1 = -1$, $X_2 = 1$, $X_3 = 0$ can be used.

Convergence of Muller's method is almost quadratic provided the three initial approximations are sufficiently close to a zero of $P(X)$. This is natural to expect since $P(X)$ is being approximated by a

quadratic polynomial. Quadratic convergence means that the error obtained in the $(n+1)^{th}$ step of the iterative process is proportional to the square of the error obtained in the n^{th} iteration. However, no general proof of convergence has been obtained for Muller's method. It has produced convergence in the majority of the cases tested.

In application of Muller's method, an alteration should be made to handle the case in which the denominator of equation (3-3) is zero (0). This occurs whenever $P(X_n) = P(X_{n-1}) = P(X_{n-2})$. If this happens, set $q_{n+1} = 1$.

Another alteration which should be made in actual practice is to compute the quantity $|P(X_{n+1})| / |P(X_n)|$ whenever the value $P(X_{n+1})$ is calculated. If the former quantity exceeds ten (10), q_{n+1} is halved and h_n, X_{n+1}, and $P(X_{n+1})$ are recomputed accordingly.

2. Procedure for Muller's Method

The basic steps performed by Muller's method are listed sequentially as follows, starting with initial approximations X_1, X_2, and X_3.

1. Compute h_n, q_n, D_n, B_n, C_n, q_{n+1} as defined previously.
2. Compute the next approximation X_{n+1} by
$$X_{n+1} = X_n + h_n q_{n+1}.$$
3. Test for convergence; that is, test
$$|X_{n+1} - X_n| / |X_{n+1}| < \varepsilon$$
for some suitably small number ε.
4. If the test fails, return to step 1 with the last three approximations X_{n+1}, X_n, X_{n-1}.

5. If the test passes, do the following:

   a. Save X_{n+1} as the desired approximation to a zero.

   b. Deflate the current polynomial using X_{n+1}.

   c. Replace the current polynomial by the deflated polynomial.

   d. Return to step 1 with a new set of initial approximations.

In order to avoid an unending iteration process in case the method does not produce convergence, a maximum number of iterations should be specified. If convergence is not obtained within this number of iterations, the initial approximations should be altered.

3. Geometrical Interpretation of Muller's Method

Figure 3.1. shows the geometrical interpretation of Muller's method for real roots of P(X) and the quadratic Q(X). The root of Q(X) closest to X_i is chosen as the next approximation X_{i+1}.

4. Determining Multiple Roots

For a discussion concerning multiple roots see Chapter II, § 5.

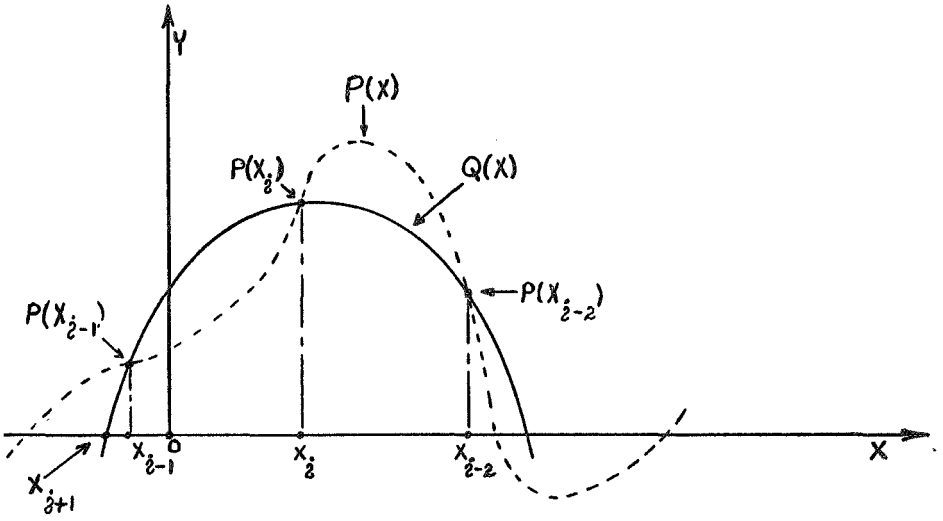


Figure 3.1. Geometrical Interpretation of Muller's Method

CHAPTER IV

GREATEST COMMON DIVISOR METHOD

1. Derivation of the Algorithm

The greatest common divisor (g.c.d.) method reduces the problem of finding all the zeros of a polynomial, possibly having multiple zeros, to one of solving for zeros of a polynomial all of whose zeros are simple.

Consider the N^{th} degree polynomial

$$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1}$$

where $a_1 \neq 0$ and $a_1,a_2,\ldots,a_{N+1}$ are complex numbers. If P(X) has m distinct zeros, $\alpha_1,\alpha_2,\ldots,\alpha_m$, then P(X) can be expressed in the form

$$P(X) = a_1(X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \ldots (X - \alpha_m)^{e_m} \qquad (4\text{-}1)$$

where e_i is the multiplicity of α_i, $i = 1,2,\ldots,m$. The derivative of P(X) is

$$P'(X) = N\, a_1X^{N-1} + (N-1)\, a_2X^{N-2} + \ldots + a_N$$

which can also be expressed as

$$P'(X) = a_1(X - \alpha_1)^{e_1-1}(X - \alpha_2)^{e_2-1} \cdots (X - \alpha_m)^{e_m-1} \sum_{i=1}^{m} e_i \prod_{\substack{j=1 \\ j\neq i}}^{m} (X - \alpha_j). \tag{4-2}$$

The greatest common divisor of $P(X)$ and $P'(X)$ is obtained from the following theorem.

Theorem 4.1. Let $P(X)$ be an N^{th} degree polynomial having m distinct zeros $\alpha_1,\alpha_2,\ldots,\alpha_m$ of multiplicity $e_1,e_2,\ldots,e_m$ respectively. Then the polynomial

$$D(X) = (X - \alpha_1)^{e_1-1}(X - \alpha_2)^{e_2-1} \cdots (X - \alpha_m)^{e_m-1}$$

is the unique monic greatest common divisor of $P(X)$ and its derivative $P'(X)$.

Proof. Since the set of all polynomials over the complex number field is a unique factorization domain and since each factor $X - \alpha_i$ is irreducible, it follows from (4-1) and (4-2) that $D(X)$ is the unique monic greatest common divisor of $P(X)$ and $P'(X)$.

It follows from Theorem 4.1 that each zero of $D(X)$ is also a zero of $P(X)$ and $P'(X)$. Hence we have the following result.

Theorem 4.2. If $P(X)$ is a polynomial, then $P(X)$ and $P'(X)$ are relatively prime if and only if $P(X)$ has no multiple zeros.

Consider the polynomial $H(X)$ obtained by dividing $P(X)$ by its monic g.c.d., $D(X)$.

$$H(X) = P(X)/D(X)$$

$$= a_1 \overbrace{\prod_{i=1}^{m}} (X - \alpha_i)^{e_i} \Big/ \overbrace{\prod_{i=1}^{m}} (X - \alpha_i)^{e_i - 1}$$

$$= a_1 \overbrace{\prod_{i=1}^{m}} (X - \alpha_i).$$

The zeros of H(X) are all simple zeros and are also all the distinct zeros of P(X). Use of the g.c.d. method involves computation of H(X) when given P(X).

In order to obtain H(X), a computational algorithm is necessary to find the g.c.d. of P(X) and $P'(X)$. The general method for computing the g.c.d. of two polynomials is as follows: Let $R_0(X)$ and $R_1(X)$ be two polynomials having degrees N_0 and N_1 respectively such that $N_1 \leq N_0$. The g.c.d. of $R_0(X)$ and $R_1(X)$ is desired. By the division algorithm, there exists polynomials $S_1(X)$ and $R_2(X)$ such that

$$R_0(X) = R_1(X)\ S_1(X) + R_2(X)$$

where either $R_2(X) = 0$ or deg. $R_2(X)$ < deg. $R_1(X)$. Similarly if $R_2(X) \neq 0$, there exists polynomials $S_2(X)$ and $R_3(X)$ such that

$$R_1(X) = S_2(X)\ R_2(X) + R_3(X)$$

where either $R_3(X) = 0$ or deg. $R_3(X)$ < deg. $R_2(X)$. Continuing in the above manner, suppose $R_i(X)$ and $R_{i+1}(X)$ have been found where deg. $R_{i+1}(X)$ < deg. $R_i(X)$. Then there exists polynomials $R_{i+2}(X)$ and $S_{i+1}(X)$ such that

$$R_i(X) = R_{i+1}(X)\ S_{i+1}(X) + R_{i+2}(X)$$

where either $R_{i+2}(X) = 0$ or deg. $R_{i+2}(X)$ < deg. $R_{i+1}(X)$. Then we obtain a sequence $R_0(X), R_1(X), \ldots, R_K(X), R_{K+1}(X)$ such that deg. $R_i(X)$ < deg. $R_{i-1}(X)$, $i = 1,2,\ldots,K+1$. Since a polynomial cannot have degree less than zero, the above process, in a finite number of steps (at most N_1), results in polynomials $R_{K-1}(X)$, $S_K(X)$ and $R_K(X)$ with deg. $R_K(X)$ < deg. $R_{K-1}(X)$ such that

$$R_{K-1}(X) = R_K(X)\ S_K(X) + R_{K+1}(X)$$

and $R_{K+1}(X) = 0$.

Theorem 4.3. Let the sequence $R_0(X), R_1(X), \ldots, R_K(X), R_{K+1}(X)$ be defined as above. Then $R_K(X)$ is the greatest common divisor of $R_0(X)$ and $R_1(X)$.

Proof. It is clear that $R_K(X)$ divides $R_{K-1}(X)$. If $R_K(X)$ divides $R_i(X)$ for $0 \leq j < i \leq k$, then $R_j(X) = R_{j+1}(X)\ S_{j+1}(X) + R_{j+2}(X)$. Thus, $R_K(X)$ divides $R_j(X)$ and it follows by induction that $R_K(X)$ divides both $R_0(X)$ and $R_1(X)$. By reversing the inductive argument given above, it is easy to see that if $L(X)$ divides $R_0(X)$ and $R_1(X)$, then $L(X)$ divides $R_i(X)$ for $i = 0,1,\ldots,K$. Therefore, $L(X)$ divides $R_K(X)$ which shows that $R_K(X)$ is the greatest common divisor of $R_0(X)$ and $R_1(X)$.

The above theorem tells how to obtain the greatest common divisor of two polynomials. A machine oriented method is now developed for computing the sequence of $R_j(X)$'s. Beginning the sequence with $R_0(X)$ and $R_1(X)$, the polynomial $R_{i+1}(X)$ of the sequence is derived from $R_i(X)$

and $R_{i-1}(X)$ as follows: Let $R_{i-1}(X)$ of degree N_{i-1} be given by

$$R_{i-1}(X) = r_{i-1,1} X^{N_{i-1}} + r_{i-1,2} X^{N_{i-1}-1} + \ldots + r_{i-1,N_{i-1}} X + r_{i-1,N_{i-1}+1}$$

and $R_i(X)$ of degree N_i be given by

$$R_i(X) = r_{i,1} X^{N_i} + r_{i,2} X^{N_i-1} + \ldots + r_{i,N_i} X + r_{i,N_i+1}$$

where $N_i \leq N_{i-1}$. Define $U_1(X)$ by

$$U_1(X) = (r_{i-1,1} / r_{i,1}) X^{N_{i-1}-N_i} .$$

Then define $T_1(X)$ by

$$\begin{aligned} T_1(X) &= R_{i-1}(X) - U_1(X) R_i(X) \\ &= [r_{i-1,1} - r_{i,1} (r_{i-1,1} / r_{i,1})] X^{N_{i-1}} \\ &\quad + [r_{i-1,2} - r_{i,2} (r_{i-1,1} / r_{i,1})] X^{N_{i-1}-1} \\ &\quad + \ldots \\ &\quad + [r_{i-1,N_{i-1}+1} - r_{i,N_{i-1}+1} (r_{i-1,1} / r_{i,1})] \end{aligned}$$

where $r_{i,j} = 0$ for $j > N_i+1$.

We consider three cases.

(1) If $T_1(X) = 0$, then $R_i(X) = R_K(X)$; that is, $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$.

(2) If $T_1(X) \neq 0$ and deg. $T_1(X) < N_i$, then $R_{i+1}(X) = T_1(X)$.

(3) If $T_1(X) \neq 0$ and deg. $T_1(X) = M_1 \geq N_i$, then define $U_2(X)$ by

$$U_2(X) = (t_{1,1} / r_{i,1}) X^{M_1 - N_i}$$

where

$$T_1(X) = t_{1,1} X^{M_1} + t_{1,2} X^{M_1 - 1} + \ldots + t_{1,M_1} X + t_{1,M_1+1}.$$

Define $T_2(X) = T_1(X) - U_2(X) R_i(X)$ which can be expressed by

$$\begin{aligned} T_2(X) = {} & [t_{1,1} - (t_{1,1} / r_{i,1}) r_{i,1}] X^{M_1} \\ & + [t_{1,2} - (t_{1,1} / r_{i,1}) r_{i,2}] X^{M_1 - 1} \\ & + \ldots \\ & + [t_{1,M_1+1} - (t_{1,1} / r_{i,1}) r_{i,M_1+1}] \end{aligned}$$

where $r_{i,j} = 0$ for $j > N_i + 1$. We again consider the following three cases.

(1) If $T_2(X) = 0$, then $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$.

(2) If $T_2(X) \neq 0$ and deg. $T_2(X) <$ deg. $R_i(X)$, then $R_{i+1}(X) = T_2(X)$.

(3) If $T_2(X) \neq 0$ and deg. $T_2(X) = M_2 \geq N_i$, then define $U_3(X)$ by

$$U_3(X) = (t_{2,1} / r_{i,1}) X^{M_2 - N_i}$$

where

$$T_2(X) = t_{2,1} X^{M_2} + t_{2,2} X^{M_2-1} + \ldots + t_{2,M_2} X + t_{2,M_2+1} .$$

Since deg. $T_{i+1}(X)$ < deg. $T_i(X)$, then this process is finite (not to exceed N_{i-1}) ending, for some integer S, in $T_S(X)$ such that

(1) $T_S(X) = 0$ and $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$ or

(2) $T_S(X) \neq 0$ but deg. $T_S(X)$ < deg. $R_i(X)$, in which case $T_S(X) = R_{i+1}(X)$.

Thus, using this algorithm and given $R_0(X)$ and $R_1(X)$, the sequence $R_0(X), R_1(X), R_2(X), \ldots, R_i(X), R_{i+1}(X)$ can be generated such that either

(1) $R_{i+1}(X) = 0$ and $R_i(X)$ is the g.c.d. of $R_0(X)$ and $R_1(X)$ or

(2) $R_{i+1}(X) \neq 0$ and $N_{i+1} < N_i$. In a finite number of iterations, $R_K(X)$, the g.c.d. of $R_0(X)$ and $R_1(X)$, can be obtained.

Recall that we wanted to obtain the polynomial $H(X) = P(X)/D(X)$ where D(X) is the g.c.d. of P(X) and $P'(X)$. Thus, after obtaining D(X) by the above algorithm, it is necessary to divide P(X) by D(X) obtaining H(X) all whose zeros are simple.

Once H(X) is obtained, an appropriate method such as Newton's method or Muller's method is applied to extract the zeros of H(X). This gives all the zeros of P(X).

As in Newton's or Muller's method, the zeros may be checked for accuracy and possibly improved by using them as initial approximations with the particular method applied to the full (undeflated) polynomial, P(X).

2. Determining Multiplicities

After all zeros of P(X) are found, the multiplicity of each zero can be determined by the process outlined in Chapter II, § 5.

3. Procedure for the G.C.D. Method

The basic steps performed by the greatest common divisor method are listed sequentially as follows:

1. Given a polynomial, P(X), in the form
   $P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1}$.
2. Calculate the derivative, $P'(X)$, of P(X) in the form
   $P'(X) = b_1X^{N-1} + b_2X^{N-2} + \ldots + b_N$ where $b_1 = Na_1$, $b_2 = (N-1)a_2, \ldots, b_N = a_N$.
3. Find D(X), the g.c.d. of P(X) and $P'(X)$, using the algorithms developed above.
4. Calculate H(X) = P(X)/D(X), the polynomial having only simple zeros.
5. Use some appropriate method to extract the zeros of H(X).
6. Determine the multiplicity of each of the zeros obtained in step 5.

CHAPTER V

REPEATED GREATEST COMMON DIVISOR METHOD

1. Derivation of the Algorithm

The repeated greatest common divisor (repeated g.c.d.) method makes repeated use of the g.c.d. method to extract the zeros and their multiplicities of a polynomial with complex coefficients. That is, the repeated g.c.d. method reduces the problem of finding the zeros of a polynomial, $P(X)$, which possibly has multiple zeros, to one of finding the zeros of a polynomial which has only simple zeros and the zeros of this polynomial are all the zeros of $P(X)$ of a given multiplicity.

Let

$$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1}$$
$$= a_1(X - \alpha_1)^{e_1}(X - \alpha_2)^{e_2} \ldots (X - \alpha_m)^{e_m}$$

where $a_1 \neq 0$, each a_i is a complex number, and $\alpha_1,\alpha_2,\ldots,\alpha_m$ are the distinct zeros of $P(X)$ having multiplicity $e_1,e_2,\ldots,e_m$, respectively. If $D_1(X)$ is the monic greatest common divisor of $P(X)$ and $P'(X)$, then Theorem 4.1 shows that

$$D_1(X) = (X - \alpha_1)^{e_1-1}(X - \alpha_2)^{e_2-1} \ldots (X - \alpha_m)^{e_m-1}$$

where we assume that if $e_j = 1$, then $X - \alpha_j$ does not appear in the

representation. Let $D_2(X)$ be the monic greatest common divisor of $D_1(X)$ and $D_1'(X)$. Then

$$D_2(X) = (X - \alpha_1)^{e_1-2} (X - \alpha_2)^{e_2-2} \ldots (X - \alpha_m)^{e_m-2}$$

where we assume that if $e_j \leq 2$, then $X - \alpha_j$ does not appear in the representation. From the above it is clear that the zeros of $D_1(X)$ are just the multiple zeros of $P(X)$ to one lower power. The zeros of $D_2(X)$ are just the multiple zeros of $D_1(X)$ to one lower power. Thus, the zeros of $D_2(X)$ are just the zeros of $P(X)$ which have multiplicity greater than two, and their multiplicity in $D_2(X)$ is reduced by two. Therefore, it follows that

$$G_1(X) = [P(X)/D_1(X)]/[D_1(X)/D_2(X)] = P(X)D_2(X)/[D_1(X)]^2$$

has only simple zeros and they are just the simple zeros of $P(X)$. In general if $D_j(X)$ has been defined for $1 \leq j \leq i$ and if $D_{i+1}(X)$ is the monic greatest common divisor of $D_i(X)$ and $D_i'(X)$, then the zeros of $D_{i+1}(X)$ are the multiple zeros of $D_i(X)$ to one lower power. Thus, the zeros of $D_{i+1}(X)$ are just the zeros of $P(X)$ which have multiplicity greater than $i+1$ and their multiplicity in $D_{i+1}(X)$ is reduced by $i+1$. It follows that

$$G_i(X) = [D_{i-1}(X)/D_i(X)]/[D_i(X)/D_{i+1}(X)]$$

$$= D_{i-1}(X)\ D_{i+1}(X)/[D_i(X)]^2$$

has simple zeros and they are just the zeros of $P(X)$ that have multiplicity i. Thus, we have proven the following theorem.

Theorem 5.1. Let $P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1}$ where $a_1 \neq 0$ and $a_1, a_2, \ldots, a_{N+1}$ are complex numbers. If $D_0(X) = P(X)$ and if $D_{i+1}(X)$ is the monic greatest common divisor of $D_i(X)$ and $D_i'(X)$ for $i \geq 0$, then

$$G_i(X) = D_{i-1}(X)\ D_{i+1}(X)/[D_i(X)]^2$$

has only simple zeros and they are just the zeros of $P(X)$ that have multiplicity i.

Thus, by the above theorem we can generate a sequence of polynomials $G_1(X), G_2(X), \ldots, G_K(X)$ where the set of zeros of $P(X)$ is the same as the set of zeros of this sequence and the multiplicity of each zero in $P(X)$ is given by the corresponding subscript on $G(X)$. Therefore, by using a method such as Newton's method or Muller's method to calculate the zeros of each $G_i(X)$, we will have the zeros of $P(X)$ along with their multiplicities.

2. Procedure for the Repeated G.C.D. Method

The basic steps performed by the greatest common divisor method are listed sequentially as follows:

1. Given a polynomial, $P(X)$, in the form
   $$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1}.$$
2. Set $D_0(X) = P(X)$.
3. Calculate the derivative, $D_0'(X)$, of $D_0(X)$ in the form
   $$D_0'(X) = b_1X^{M-1} + b_2X^{M-2} + \ldots + b_M$$

where deg. $D_0(X) = M$, $D_0(X) = d_1X^M + \ldots + d_{M+1}$, and $b_1 = Md_1$, $b_2 = (M-1)d_2$, $\ldots$, $b_M = d_M$.

4. Find $D_1(X)$, the g.c.d. of $D_0(X)$ and $D_0'(X)$ using the algorithms developed in Chapter IV.
5. Similar to 3., calculate $D_1'(X)$.
6. Find $D_2(X)$, the g.c.d. of $D_1(X)$ and $D_1'(X)$ using the algorithms developed in Chapter IV.
7. Calculate $G(X) = D_0(X)\ D_2(X)/[D_1(X)]^2$.
8. Use some appropriate method to extract the zeros of G(X) and assign these zeros the correct multiplicity as zeros of P(X).
9. Set $D_0(X) = D_1(X)$, $D_0'(X) = D_1'(X)$, and $D_1(X) = D_2(X)$. Then repeat 5.-8. above until all the zeros of P(X) are found.

CHAPTER VI

CONCLUSION

In order to compare Newton's, Muller's, the greatest common divisor, and the repeated greatest common divisor methods, we consider the polynomials as being divided into the following classes:

1. polynomials with all distinct zeros.
2. polynomials with multiple zeros.

The comparisons in the following material are results of tests made on the IBM 360/50 computer which has a 32 bit word. The programs were successfully run on the CDC 6600 and the UNIVAC 1108 which have a 60 bit word and a 36 bit word respectively. It was noted that the UNIVAC 1108 is about 15 times faster than the IBM 360/50. The CDC 6600 is faster than the UNIVAC 1108 but the difference is not as great as that between the UNIVAC 1108 and the IBM 360/50.

1. Polynomials With all Distinct Zeros

First we consider the class of polynomials having distinct zeros. Newton's method is particularly suited for this class of polynomials. Its quadratic convergence is very fast which can save time and money to the user. The accuracy obtained is excellent as shown in Exhibit 6.1 which presents the zeros of a 15^{th} degree polynomial in double precision. In most cases, the method produces convergence for almost any initial approximation given.

Muller's method also produces good results on this class of polynomials. The rate of convergence is, however, somewhat slower than Newton's method. This fact is especially significant when working with polynomials of high degree. The accuracy obtained by Muller's method is comparable to, but does not exceed that of Newton's method. In most cases, the accuracy of the two methods does not differ by more than one or two decimal places. Exhibit 6.2 shows results of Muller's method for the polynomial of Exhibit 6.1. As in Newton's method, convergence is produced for almost any initial approximation given.

The g.c.d. method, whether used with Newton's or Muller's method as a supporting method on this class of polynomials, is no better than Newton's or Muller's method alone. The reason for this is that the greatest common divisor of the polynomial, P(X), and its derivative is 1. Thus, H(X) = P(X)/g.c.d. P(X) = P(X); that is, the polynomial solved by the supporting method is the same as the original polynomial. Thus, in this case the g.c.d. method will not produce better results than the supporting method used alone. The above comments also hold for the repeated g.c.d. method.

Thus, this class of polynomials presents no difficulty to any of these four methods. Newton's method, because of its speed, is therefore recommended.

2. Polynomials With Multiple Zeros

Next consider the class of polynomials containing multiple zeros. Exhibits 6.3 - 6.26 illustrate output from six different programs using the methods described in Chapters II - V. Four polynomials are used where the zeros of these polynomials are listed below. The number in

parentheses indicates the multiplicity of that zero.

Polynomial #1	Polynomial #2	Polynomial #3	Polynomial #4
2+2i (3)	-2.33 (1)	2+2i (3)	1+i (6)
1+2i (2)	.003 (2)	1+2i (2)	1-i (6)
-1+.5i (1)	i (2)	-1+.5i (3)	
	1.5i (2)		
	-1.5i (2)		
	3i (3)		
	-1-i (3)		

Note the relationship between polynomials #1 and #3.

This class presents considerable difficulty for Newton's method, especially those polynomials containing zeros of high multiplicity or containing a considerable number of multiple zeros. The iteration formula for Newton's method is

$$X_{n+1} = X_n - P(X_n)/P'(X_n).$$

If c is a multiple zero then $P(c) = P'(c) = 0$. Hence, as $X_n \to c$, $P(X_n) \to 0$ and $P'(X_n) \to 0$ and the iteration formula may be unstable, resulting in no convergence or bad accuracy. As the number of multiple zeros increases, the polynomial becomes more ill-conditioned, convergence becomes more difficult, and accuracy is lost. Thus, the possibility of convergence decreases. This also holds true if the multiplicities of the zeros are increased. The rate of convergence of Newton's method is much slower for multiple zeros than for distinct zeros. Exhibit 6.3 shows a polynomial (#1) containing two multiple zeros solved in double precision. Note the following from Exhibit 6.3.

1. Roots #2 and #3 are greatly improved by iterating on the original polynomial. Distinct roots are usually improved in this manner.

2. The time taken to solve this 6^{th} degree equation with multiple roots is greater than the time taken by the same program to solve a 15^{th} degree polynomial with all distinct roots (Exhibit 6.1).
3. Root #2 did not pass the convergence test after 200 iterations even though it was improved. This is probably due to the fact that the polynomial from which root 2 was extracted had only one multiple root but the original polynomial from which it was extracted the second time had two multiple roots; that is, the original polynomial is more ill-conditioned.
4. The accuracy of the roots before the attempt to improve accuracy is very poor. Root #2 is accurate to only three decimal places as compared to the 15 decimal places in Exhibit 6.1 for distinct roots. Root #3 is especially bad, the imaginary part being accurate to only one decimal place.

Exhibit 6.4 uses polynomial #2. Note the poor results obtained before the attempt to improve accuracy and the improvement afterward. Also note that after the attempt to improve accuracy, one of the zeros, namely 3i, is lost and an extra zero, namely 1.5, is included in the list. (See Appendix A, § 4.) A convergence requirement of 10^{-5} was used on this polynomial to get it to converge to all of the zeros in a maximum number of 200 iterations.

In many cases, Newton's method fails to converge altogether. Polynomial #3 could not be solved using Newton's method with a maximum

number of 200 iterations and a convergence requirement of 10^{-9}. Exhibit 6.5 illustrates the bad results for a convergence requirement of 10^{-5} which was needed in order to get convergence. In Exhibit 6.6 a convergence requirement of 10^{-3} was needed in order to get convergence to the zeros of polynomial #4.

Muller's method also encounters difficulty, although to a lesser degree than Newton's method, on this class of polynomials. In most cases, Muller's method produces convergence even when Newton's method completely fails. Newton's method completely failed for polynomials #3 and #4 with a convergence requirement of 10^{-9} but convergence was obtained using Muller's method as shown in Exhibits 6.9 and 6.10. The accuracy obtained by Muller's method is not good but usually better than Newton's method using the same convergence requirement. The rate of convergence of Muller's method is considerably slower for multiple zeros than for distinct zeros. However, for multiple zeros, Muller's method is as fast or faster than Newton's.

The g.c.d. method is perfectly suited for polynomials with multiple zeros. All multiple zeros are removed leaving only a polynomial of class 1 (all distinct roots) to be solved. This indicates that best results should be obtained by using Newton's method as the supporting method, since Newton's method enjoys the advantage of speed over Muller's method for distinct zeros. This has indeed proved to be true. The accuracy of the zeros obtained decreases, somewhat, when the number of multiple zeros is increased. This is due to accuracy lost in computing the g.c.d. and the quotient polynomial and not as a result of the supporting method. It is easy to see that the accuracy of the g.c.d. method is best when the degree of the greatest common divisor of

$P(X)$ and $P'(X)$ is maximum. This is due to the fact that the error in the greatest common divisor is minimized in this case. The accuracy obtained using Newton's method and Muller's method as supporting methods is about the same. This is verified by Exhibits 6.11 - 6.14 (g.c.d. method with Newton) and Exhibits 6.15 - 6.18 (g.c.d. method with Muller).

Multiplicities are determined with excellent accuracy. The g.c.d. method is not as sensitive to zeros of high multiplicity or polynomials containing a large number of multiple zeros as are both Newton's and Muller's methods. A quick comparison of Exhibits 6.11 - 6.14 and 6.15 - 6.18 with Exhibits 6.3 - 6.6 and 6.7 - 6.10 show that the g.c.d. method with either supporting method is much more accurate than either Newton's or Muller's method. For example, Exhibits 6.5 and 6.9 show polynomial #3 for which Newton's method and Muller's method both gave poor convergence. But Exhibits 6.13 and 6.17 show very accurate results for polynomial #3.

The repeated g.c.d. method is also suited very well for polynomials with multiple zeros. Exhibits 6.19 - 6.22 and Exhibits 6.23 - 6.26 are results of the repeated g.c.d. method with Newton's method and Muller's method as supporting methods, respectively. However, the results of the repeated g.c.d. method are not as good as those obtained from the g.c.d. method. Since the repeated g.c.d. method repeated uses the g.c.d. algorithm, the error tends to build up in this method when a polynomial has several zeros of different multiplicities. This can be observed by comparing Exhibits 6.20 and 6.24 with Exhibits 6.12 and 6.16 on polynomial #2 and by comparing Exhibits 6.21 and 6.25 with Exhibits 6.13 and 6.17 on polynomial #3. As was the case of the g.c.d.

method, there is little difference between the repeated g.c.d. method with Newton's method or Muller's method as a supporting method. This can be observed by comparing Exhibits 6.19 - 6.22 (Newton) with Exhibits 6.23 - 6.26 (Muller). Even though the results of the repeated g.c.d. method are not quite as good as the results of the g.c.d. method, they are far superior to the results of both Newton's method and Muller's method.

Table 6.I gives a comparison of the execution times of the six methods for polynomials #1 - #4.

TABLE 6.I

METHOD	EXECUTION TIME*
Newton	104.16 seconds
Muller	96.79 seconds
G.C.D. with Newton	7.51 seconds
G.C.D. with Muller	8.91 seconds
Repeated G.C.D. with Newton	7.71 seconds
Repeated G.C.D. with Muller	15.16 seconds

It is clear from Table 6.I that the g.c.d. and the repeated g.c.d. methods are much faster than both Newton's and Muller's method on

*These times are from execution runs on the IBM 360/50 WATFOR system.

polynomials with multiple zeros. Therefore, for polynomials with multiple zeros, the order in which the methods are recommended is as follows.

1. G.C.D. with Newton.
2. G.C.D. with Muller.
3. Repeated G.C.D. with Newton.
4. Repeated G.C.D. with Muller.
5. Muller.
6. Newton.

```
NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER  7 OF DEGREE 15


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.30000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.17900000000000000D 02 + -0.00000000000000000D 00 I
  P( 3) =  0.20100000000000000D 02 + -0.65760000000000010D 02 I
  P( 4) =  0.17450000000000000D 03 +  0.28436800000000000D 03 I
  P( 5) = -0.75942000000000000D 03 +  0.31180800000000000D 03 I
  P( 6) =  0.82743600000000000D 03 + -0.30690400000000000D 04 I
  P( 7) =  0.13290840000000000D 04 +  0.47106240000000000D 04 I
  P( 8) = -0.56118600000000000D 04 + -0.16745760000000000D 04 I
  P( 9) =  0.72247560000000000D 04 + -0.15482880000000000D 04 I
  P(10) = -0.22769920000000000D 04 +  0.30463200000000000D 04 I
  P(11) = -0.12414720000000000D 04 + -0.40972800000000000D 04 I
  P(12) =  0.54028000000000010D 04 +  0.12634880000000000D 04 I
  P(13) = -0.64683360000000010D 04 +  0.12364800000000000D 04 I
  P(14) =  0.14674560000000000D 04 +  0.22720000000000000D 02 I
  P(15) = -0.10771200000000000D 03 + -0.54758400000000000D 03 I
  P(16) =  0.34560000000000000D 02 +  0.12672000000000000D 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.                0.10D-09
TEST FOR MULTIPLICITIES.             0.10D-01
RADIUS TO START SEARCH.              0.00D 00
RADIUS TO END SEARCH.                0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

 ROOTS OF P(X)                                                    MULTIPLICITIES          INITIAL APPROXIMATION

  ROOT( 1) =  0.30000000000000000D 00 +  0.16093581151665310-16 I        1        0.48296291156656279D 00 +  0.12940952844381870 00 I
  ROOT( 2) =  0.20000000000000000D 00 + -0.20000000000000000D 00 I        1        0.70710675530463460 00 +  0.70710680706845950 00 I
  ROOT( 3) = -0.20000000000000000D 00 +  0.20000000000000000D 00 I        1        0.38822847926540560 00 +  0.14488887631171930 01 I
  ROOT( 4) =  0.24850261096558030-17 +  0.10000000000000000D 01 I        1       -0.51763825519667240 00 +  0.19318516083687550 01 I
  ROOT( 5) = -0.10000000000000000D 01 + -0.53219537064599460-16 I        1       -0.17677671470807010 01 +  0.17677667588520150 01 I
  ROOT( 6) = -0.10000000000000000D 01 + -0.99999999999999980 00 I        1       -0.28977775830749900 01 +  0.77645674639870700 00 I
  ROOT( 7) = -0.31c.918759280349D-15 + -0.99999999999999940 00 I        1       -0.33807402483312290 01 + -0.90586719408161600 00 I
  ROOT( 8) = -0.20000000000000000D 01 + -0.30000000000000000D 01 I        1       -0.28284266071078960 01 + -0.28284276423843900 01 I
  ROOT( 9) =  0.10000000000000016D 01 + -0.10097649963521000-12 I        1       -0.11646848013998990 01 + -0.43466664598733680 01 I
  ROOT(10) =  0.20000000000000040D 01 + -0.10000000000000320 01 I        1        0.12940963450986450 01 + -0.48296288314530270 01 I
  ROOT(11) =  0.19999999999999800 01 +  0.13855078472222020-12 I        1        0.38890882929795090 01 + -0.38890863000722580 01 I
  ROOT(12) =  0.30000000000000010 01 + -0.14175491574492190-13 I        1        0.57955553935123030 01 + -0.15529126442689740 01 I
  ROOT(13) =  0.39999999999999980 01 +  0.40000000000000030 01 I        1        0.62785173577341250 01 +  0.16823257082477520 01 I
  ROOT(14) =  0.99999999999996060 00 +  0.10000000000000050 01 I        1        SOLVED BY DIRECT METHOD
  ROOT(15) = -0.33333333333333340 01 + -0.18503717077085940-15 I        1        SOLVED BY DIRECT METHOD
```

Exhibit 6.1.

```
AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

ROOTS OF P(X)                                                    MULTIPLICITIES        INITIAL APPROXIMATION

 ROOT( 1) =  0.3000000000000000D 00 +  0.3328943537705913D-16 I        1        0.4829629115656279D 00 +  0.1294095284438187D 00 I
 ROOT( 2) =  0.2000000000000000D 00 + -0.2000000000000000D 00 I        1        0.7071067553046346D 00 +  0.7071068070684595D 00 I
 ROOT( 3) = -0.2000000000000000D 00 +  0.2000000000000000D 00 I        1        0.3882284792654056D 00 +  0.1448888763117193D 01 I
 ROOT( 4) = -0.1126043043762753D-17 +  0.1000000000000000D 01 I        1       -0.5176382551966724D 00 +  0.1931851608368755D 01 I
 ROOT( 5) = -0.1000000000000000D 01 + -0.1882478030550796D-16 I        1       -0.1767767147080701D 01 +  0.1767766758852015D 01 I
 ROOT( 6) = -0.1000000000000000D 01 + -0.1000000000000000D 01 I        1       -0.2897777583074990D 01 +  0.7764567463987070D 00 I
 ROOT( 7) =  0.6131869319379140D-16 + -0.1000000000000000D 01 I        1       -0.3380740248331229D 01 + -0.9058671940816160D 00 I
 ROOT( 8) = -0.2000000000000000D 01 + -0.3000000000000000D 01 I        1       -0.2828426607107896D 01 + -0.2828427642384390D 01 I
 ROOT( 9) =  0.1000000000000000D 01 +  0.7287765480895064D-16 I        1       -0.1164684801399899D 01 + -0.4346666459873368D 01 I
 ROOT(10) =  0.2000000000000001D 01 + -0.9999999999999990D 00 I        1        0.1294096345098645D 01 + -0.4829628831453027D 01 I
 ROOT(11) =  0.2000000000000003D 01 + -0.3968039620811357D-14 I        1        0.3889088292979509D 01 + -0.3889086300072258D 01 I
 ROOT(12) =  0.2999999999999997D 01 +  0.2896889630437848D-14 I        1        0.5795555393512303D 01 + -0.1552912644268974D 01 I
 ROOT(13) =  0.4000000000000000D 01 +  0.4000000000000001D 01 I        1        0.6278517357734125D 01 +  0.1682325708247752D 01 I
 ROOT(14) =  0.9999999999999995D 00 +  0.1000000000000000D 01 I        1       SOLVED BY DIRECT METHOD
 ROOT(15) = -0.3333333333333340D 01 + -0.8998063670517638D-16 I        1       SOLVED BY DIRECT METHOD
```

Exhibit 6.1. Roots Are: $-1 - i$, $1 + i$, $-2 - 3i$, $2 - i$, 3, 2, i, $-i$, $-10/3$, $.3$, -1, 1, $4 + 4i$, $-.2 + .2i$, $.2 - .2i$.

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER  7 OF DEGREE 15


THE COEFFICIENTS OF P(X) ARE


  P( 1) =  0.30000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.17900000000000000D 02 + -0.00000000000000000D 00 I
  P( 3) =  0.20100000000000000D 02 + -0.65760000000000010D 02 I
  P( 4) =  0.17450000000000000D 03 +  0.28436800000000000D 03 I
  P( 5) = -0.75942000000000000D 03 +  0.31180800000000000D 03 I
  P( 6) =  0.82743600000000000D 03 + -0.30690400000000000D 04 I
  P( 7) =  0.13290840000000000D 04 +  0.47106240000000000D 04 I
  P( 8) = -0.56118600000000000D 04 + -0.16745760000000000D 04 I
  P( 9) =  0.72247560000000000D 04 + -0.15482880000000000D 04 I
  P(10) = -0.22769920000000000D 04 +  0.30463200000000000D 04 I
  P(11) = -0.12414720000000000D 04 + -0.40972800000000000D 04 I
  P(12) =  0.54028000000000010D 04 +  0.12634880000000000D 04 I
  P(13) = -0.64683360000000010D 04 +  0.12364800000000000D 04 I
  P(14) =  0.14674560000000000D 04 +  0.22720000000000000D 02 I
  P(15) = -0.10771200000000000D 03 + -0.54758400000000000D 03 I
  P(16) =  0.34560000000000000D 02 +  0.12672000000000000D 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY


ROOTS OF P(X)                                                    MULTIPLICITIES          INITIAL APPROXIMATION

 ROOT( 1) =  0.30000000000000000D 00 +  0.3383998751751866D-16 I          1          0.48296291156562790D 00 +  0.12940952844381870D 00 I
 ROOT( 2) =  0.20000000000000000D 00 + -0.20000000000000000D 00 I          1          0.70710675530463460D 00 +  0.70710680706845950D 00 I
 ROOT( 3) =  0.99999999999999950D 00 +  0.99999999999999970D 00 I          1          0.38822847926540560D 00 +  0.14488887631171930D 01 I
 ROOT( 4) = -0.23890824083133820D-16 +  0.10000000000000000D 01 I          1         -0.51763825519667240D 00 +  0.19318516083687550D 01 I
 ROOT( 5) = -0.10000000000000000D 01 +  0.25316292776176400D-15 I          1         -0.17677671470807010D 01 +  0.17677667588520150D 01 I
 ROOT( 6) = -0.33333333333333340D 01 + -0.21447226690877370D-15 I          1         -0.28977775830749900D 01 +  0.77645674639870700D 00 I
 ROOT( 7) = -0.10000000000000080D 01 + -0.99999999999996610D 00 I          1         -0.33807402483312290D 01 + -0.90586719408161600D 00 I
 ROOT( 8) = -0.20000000000000010D 01 + -0.30000000000000000D 01 I          1         -0.28284266071078960D 01 + -0.28284276423843900D 01 I
 ROOT( 9) = -0.11388167309366500D-12 + -0.10000000000001090D 01 I          1         -0.11646848013998990D 01 + -0.43466664598733680D 01 I
 ROOT(10) =  0.19999999999998790D 01 + -0.99999999999999540D 00 I          1          0.12940963450986450D 01 + -0.48296288314530270D 01 I
 ROOT(11) =  0.29999999999999460D 01 +  0.19218912044391240D-14 I          1          0.38890882929795090D 01 + -0.38890863000722580D 01 I
 ROOT(12) =  0.20000000000001220D 01 + -0.22541548795801700D-12 I          1          0.57955553935123030D 01 + -0.15529126442689740D 01 I
 ROOT(13) =  0.39999999999999990D 01 +  0.40000000000000050D 01 I          1          0.62785173577341250D 01 +  0.16823257082477520D 01 I
 ROOT(14) =  0.10000000000002080D 01 +  0.22974446419373370D-12 I          1         SOLVED BY DIRECT METHOD
 ROOT(15) = -0.20000000000000360D 00 +  0.20000000000005950D 00 I          1         SOLVED BY DIRECT METHOD
```

Exhibit 6.2.

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)		MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) =	0.30000000000000000D 00 + 0.33767394620832440-16 I	1	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) =	0.20000000000000000D 00 + -0.20000000000000000D 00 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
ROOT(3) =	0.99999999999999950 00 + 0.10000000000000000D 01 I	1	0.38822847926540560 00 + 0.14488887631171930 01 I
ROOT(4) =	0.1153775775153668D-16 + 0.10000000000000000D 01 I	1	-0.51763825519667240 00 + 0.19318516083687550 01 I
ROOT(5) =	-0.10000000000000000D 01 + -0.9935626030016848D-17 I	1	-0.17677671470807010 01 + 0.17677667588520150 01 I
ROOT(6) =	-0.33333333333333340 01 + -0.19191253971536640-15 I	1	-0.28977775830749900 01 + 0.77645674639870700 00 I
ROOT(7) =	-0.10000000000000000D 01 + -0.99999999999999980 00 I	1	-0.33807407483117290 01 + -0.90586719408161600 00 I
ROOT(8) =	-0.20000000000000000D 01 + -0.30000000000000000D 01 I	1	-0.28284266071078960 01 + -0.28284276423843900 01 I
ROOT(9) =	0.55291352338277240-16 + -0.10000000000000000D 01 I	1	-0.11646848013998990 01 + -0.43466664598733680 01 I
ROOT(10) =	0.20000000000000020 01 + -0.99999999999999860 00 I	1	0.12940963450986450 01 + -0.48296288314530270 01 I
ROOT(11) =	0.29999999999999960 01 + 0.24394773729111440-14 I	1	0.38890882929795090 01 + -0.38890863000722580 01 I
ROOT(12) =	0.20000000000000020 01 + -0.41020504900166570-14 I	1	0.57955553935123030 01 + -0.15529126442689740 01 I
ROOT(13) =	0.40000000000000000 01 + 0.40000000000000010 01 I	1	0.62785173577341250 01 + 0.16823257082477520 01 I
ROOT(14) =	0.10000000000000000D 01 + 0.7682438283857070D-16 I	1	SOLVED BY DIRECT METHOD
ROOT(15) =	-0.20000000000000000D 00 + 0.20000000000000000D 00 I	1	SOLVED BY DIRECT METHOD

Exhibit 6.2. Roots Are: $-1 - i$, $1 + i$, $-2 - 3i$, $2 - i$, 3, 2, i, $-i$, $-10/3$, $.3$, -1, 1, $4 + 4i$, $-.2 + .2i$, $.2 - .2i$.

```
      NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
      POLYNOMIAL NUMBER  1 OF DEGREE  6


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.70000000000000001D 01 + -0.10500000000000000D 02 I
  P( 3) = -0.28000000000000000D 02 +  0.58000000000000001D 02 I
  P( 4) =  0.17100000000000000D 03 +  0.15000000000000000D 01 I
  P( 5) = -0.73000000000000000D 02 + -0.25100000000000000D 03 I
  P( 6) = -0.22800000000000000D 03 +  0.10400000000000000D 03 I
  P( 7) =  0.72000000000000001D 02 +  0.10400000000000000D 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

 ROOTS OF P(X)                                                  MULTIPLICITIES              INITIAL APPROXIMATION

  ROOT( 1) =  0.99999988361250190 00 +  0.20000000521382840 01 I        2        0.48296291156562790 00 +  0.12940952844381870 00 I
  ROOT( 2) =  0.19967378102574860 01 +  0.19952538211436840 01 I        3        0.70710675530463460 00 +  0.70710680706845950 00 I
  ROOT( 3) = -0.99021319799746240 00 +  0.51423843229238120 00 I        1        SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 2) =  0.19967378102574860 01 +  0.19952538211436840 01 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

 ROOTS OF P(X)                                                  MULTIPLICITIES              INITIAL APPROXIMATION

  ROOT( 1) =  0.99999988361250190 00 +  0.20000000521382840 01 I        2        0.48296291156562790 00 +  0.12940952844381870 00 I
  ROOT( 2) =  0.19999929075033090 01 +  0.19999594740016890 01 I        3        0.70710675530463460 00 +  0.70710680706845950 00 I
  ROOT( 3) = -0.99999999999999980 00 +  0.50000000000000000 00 I        1        SOLVED BY DIRECT METHOD
```

Exhibit 6.3. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

```
 NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
 POLYNOMIAL NUMBER  2 OF DEGREE 15


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
  P( 2) =  0.25571200000000000D 03 + -0.38400000000000000D 03 I
  P( 3) = -0.73535568000000000D 02 + -0.21896960000000000D 04 I
  P( 4) = -0.38555656960000000D 04 + -0.69468514560000010 04 I
  P( 5) = -0.17333864648000000D 05 + -0.14206259728000000D 05 I
  P( 6) = -0.49679892704000010 05 + -0.17658574640000000D 05 I
  P( 7) = -0.10223945221300000D 06 + -0.60306642320000010 04 I
  P( 8) = -0.16427422005600000D 06 +  0.41373662304000000D 05 I
  P( 9) = -0.20366258884200000D 06 +  0.10938992276700000D 06 I
  P(10) = -0.18712557800100000D 06 +  0.19298654403300000D 06 I
  P(11) = -0.12749972985900000D 06 +  0.21713412274200000D 06 I
  P(12) = -0.28146927168000000D 05 +  0.19284897279600000D 06 I
  P(13) =  0.13294344348000000D 05 +  0.10381302265500000D 06 I
  P(14) =  0.30539007747000000D 05 +  0.29989891413000000D 05 I
  P(15) = -0.18358990200000000D 03 + -0.18276321600000000D 03 I
  P(16) =  0.27556200000000000D 00 +  0.27556200000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.10D-05
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                    MULTIPLICITIES          INITIAL APPROXIMATION

  ROOT( 1) =  0.30000007363981840-02 +  0.26093949065529810-08 I        2        0.48296291156562790 00 +  0.12940952844381870 00 I
  ROOT( 2) =  0.86266231240991390-04 +  0.10000801737229880 01 I        2        0.70710675530463460 00 +  0.70710680706845950 00 I
  ROOT( 3) =  0.57683426492756900-02 +  0.15081451743165700 01 I        1        0.38822847926540560 00 +  0.14488887631171930 01 I
  ROOT( 4) = -0.59266344710587620-02 +  0.14917229999229610 01 I        1       -0.51763825519667240 00 +  0.19318516083687550 01 I
  ROOT( 5) = -0.10149047544016010 01 + -0.10284209313843700 01 I        1       -0.17677671470807010 01 +  0.17677667588520150 01 I
  ROOT( 6) = -0.23333336178348630 01 +  0.21686244949452000-05 I        1       -0.28977775830749900 01 +  0.77645674639870700 00 I
  ROOT( 7) = -0.10185084356785580 01 + -0.97285956027561890 00 I        1       -0.33807402483312290 01 + -0.90586719408161600 00 I
  ROOT( 8) = -0.96654419892828350 00 + -0.99877747895279140 00 I        1       -0.28284266071078960 01 + -0.28284276423843900 01 I
  ROOT( 9) = -0.24814837099370320-02 + -0.14978798627107940 01 I        2       -0.11646848013998990 01 + -0.43466664598733680 01 I
  ROOT(10) =  0.15605144175373180 00 +  0.24952076852120910 01 I        1        0.12940963450986450 01 + -0.48296288314530270 01 I
  ROOT(11) =  0.33256756537607100 00 +  0.33659936274026380 01 I        1        SOLVED BY DIRECT METHOD
  ROOT(12) = -0.48371260812910090 00 +  0.31345856878908420 01 I        1        SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE
```

Exhibit 6.4.

ROOTS OF P(X)		MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) =	0.3000000368372262D-02 + 0.13040676163986510-08 I	2	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) =	0.6742035082463408D-06 + 0.1000000627692775D 01 I	2	0.70710675530463460 00 + 0.70710680706845950 00 I
ROOT(3) =	0.7624724932810605D-06 + 0.1500001030008833D 01 I	1	0.38822847926540560 00 + 0.14488887631171930 01 I
ROOT(4) =	-0.6671875662730541D-06 + 0.14999990111650930 01 I	1	-0.51763825519667240 00 + 0.19318516083687550 01 I
ROOT(5) =	-0.9999960111018027D 00 + -0.10000074938436410 01 I	1	-0.17677671470807010 01 + 0.17677667588520150 01 I
ROOT(6) =	-0.2333333333311710D 01 + 0.6499059058277495D-11 I	1	-0.28977775830749900 01 + 0.77645674639870700 00 I
ROOT(7) =	-0.1000007692970472D 01 + -0.99999836986193700 00 I	1	-0.33807402483312290 01 + -0.90586719408161600 00 I
ROOT(8) =	-0.9999951740158410D 00 + -0.99999502636776190 00 I	1	-0.28284266071078960 01 + -0.28284276423843900 01 I
ROOT(9) =	-0.5924878530578477D-06 + -0.14999994871877380 01 I	2	-0.11646848013998990 01 + -0.43466664598733680 01 I
ROOT(10) =	-0.6093654451589193D-06 + 0.15000010898249810 01 I	1	0.12940963450986450 01 + -0.48296288314530270 01 I
ROOT(11) =	0.2619599103083468D-04 + 0.30000078386717850 01 I	1	SOLVED BY DIRECT METHOD
ROOT(12) =	-0.9819291180928265D-05 + 0.29999741215895750 01 I	1	SOLVED BY DIRECT METHOD

Exhibit 6.4. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)

```
      NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
      POLYNOMIAL NUMBER  3 OF DEGREE  8


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.50000000000000010D 01 + -0.11500000000000000D 02 I
  P( 3) = -0.51750000000000010D 02 +  0.43000000000000010D 02 I
  P( 4) =  0.15725000000000000D 03 +  0.14462500000000000D 03 I
  P( 5) =  0.30750000000000000D 03 + -0.34750000000000000D 03 I
  P( 6) = -0.49525000000000000D 03 + -0.49487500000000000D 03 I
  P( 7) = -0.58575000000000010D 03 +  0.42475000000000010D 03 I
  P( 8) =  0.18100000000000000D 03 +  0.44200000000000010D 03 I
  P( 9) =  0.15800000000000000D 03 +  0.60000000000000010D 01 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.                0.10D-05
TEST FOR MULTIPLICITIES.             0.10D-01
RADIUS TO START SEARCH.              0.00D 00
RADIUS TO END SEARCH.                0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

 ROOTS OF P(X)                                                  MULTIPLICITIES              INITIAL APPROXIMATION

  ROOT( 1) =  0.99999924980447420D 00 +  0.19999984679040900D 01 I        2        0.48296291156562790D 00 +  0.12940952844381870D 00 I
  ROOT( 2) = -0.99216870167763280D 00 +  0.50273964052275640D 00 I        2        0.70710675530463460D 00 +  0.70710680706845950D 00 I
  ROOT( 3) =  0.19166896842724220D 01 +  0.19669597870445160D 01 I        1        0.38822847926540560D 00 +  0.14488887631171930D 01 I
  ROOT( 4) =  0.20137268061676120D 01 +  0.20860550121059220D 01 I        1       -0.51763825519667240D 00 +  0.19318516083687550D 01 I
  ROOT( 5) =  0.20694066410038210D 01 +  0.19469500958857110D 01 I        1        SOLVED BY DIRECT METHOD
  ROOT( 6) = -0.10154842276975370D 01 +  0.49455888811015920D 00 I        1        SOLVED BY DIRECT METHOD


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

 ROOTS OF P(X)                                                  MULTIPLICITIES              INITIAL APPROXIMATION

  ROOT( 1) =  0.99999962393979710D 00 +  0.19999992333486960D 01 I        2        0.48296291156562790D 00 +  0.12940952844381870D 00 I
  ROOT( 2) = -0.99999265391488270D 00 +  0.49999755579552720D 00 I        2        0.70710675530463460D 00 +  0.70710680706845950D 00 I
  ROOT( 3) =  0.19999562215780010D 01 +  0.20000208360361000D 01 I        1        0.38822847926540560D 00 +  0.14488887631171930D 01 I
  ROOT( 4) =  0.20000058293227980D 01 +  0.19999524995012820D 01 I        1       -0.51763825519667240D 00 +  0.19318516083687550D 01 I
  ROOT( 5) =  0.20000087385097060D 01 +  0.19999500538776400D 01 I        1        SOLVED BY DIRECT METHOD
  ROOT( 6) = -0.10000053739815870D 01 +  0.49999509201216870D 00 I        1        SOLVED BY DIRECT METHOD
```

Exhibit 6.5. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

```
   NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
   POLYNOMIAL NUMBER  4 OF DEGREE 12


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.12000000000000000D 02 + -0.00000000000000000D 00 I
  P( 3) =  0.72000000000000010D 02 +  0.00000000000000000D 00 I
  P( 4) = -0.28000000000000000D 03 + -0.00000000000000000D 00 I
  P( 5) =  0.78000000000000000D 03 +  0.00000000000000000D 00 I
  P( 6) = -0.16320000000000000D 04 + -0.00000000000000000D 00 I
  P( 7) =  0.26240000000000000D 04 +  0.00000000000000000D 00 I
  P( 8) = -0.32640000000000000D 04 + -0.00000000000000000D 00 I
  P( 9) =  0.31200000000000000D 04 +  0.00000000000000000D 00 I
  P(10) = -0.22400000000000000D 04 + -0.00000000000000000D 00 I
  P(11) =  0.11520000000000000D 04 +  0.00000000000000000D 00 I
  P(12) = -0.38400000000000000D 03 + -0.00000000000000000D 00 I
  P(13) =  0.64000000000000010D 02 +  0.00000000000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.              0.10D-03
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE


 ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

  ROOT( 1) =  0.99940798033691520 00 +  0.99513914896439410 00 I        5        0.48296291156562790 00 +  0.12940952844381870 00 I
  ROOT( 2) =  0.10200968856097200 01 + -0.54206461303878890 00 I        1        0.70710675530463460 00 +  0.70710680706845950 00 I
  ROOT( 3) =  0.10027211066816580 01 +  0.10233106615266340 01 I        1        0.38822847926540560 00 +  0.14488887631171930 01 I
  ROOT( 4) =  0.61899689593614000 00 + -0.79497020137361440 00 I        1       -0.51763825519667240 00 +  0.19318516083687550 01 I
  ROOT( 5) =  0.98514220474234140 00 + -0.13956979981330380 01 I        1       -0.17677671470807010 01 +  0.17677667588520150 01 I
  ROOT( 6) =  0.65158282151654210 00 + -0.12069876664549360 01 I        1       -0.28977775830749900 01 +  0.77645674639870700 00 I
  ROOT( 7) =  0.13937999844005080 01 + -0.82719243111768510 00 I        1         SOLVED BY DIRECT METHOD
  ROOT( 8) =  0.13306201994285140 01 + -0.12320934962305410 01 I        1         SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 2) =  0.10200968856097200 01 + -0.54206461303878890 00 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 5) =  0.98514220474234140 00 + -0.13956979981330380 01 I DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE
```

Exhibit 6.6.

ROOTS OF P(X)		MULTIPLICITIES	INITIAL APPROXIMATION
ROOT(1) =	0.99948113795146260 00 + 0.99560775860617460 00 I	5	0.48296291156562790 00 + 0.12940952844381870 00 I
ROOT(2) =	0.99592313317176770 00 + -0.10017819876754680 01 I	1	0.70710675530463460 00 + 0.70710680706845950 00 I
ROOT(3) =	0.99585780717884340 00 + 0.10031671344932110 01 I	1	0.38822847926540560 00 + 0.14488887631171930 01 I
ROOT(4) =	0.99991569960089460 00 + -0.99503023164852320 00 I	1	-0.51763825519667240 00 + 0.19318516083687550 01 I
ROOT(5) =	0.99391042189066210 00 + -0.99411359917024930 00 I	1	-0.17677671470807010 01 + 0.17677667588520150 01 I
ROOT(6) =	0.99605385885658690 00 + -0.10027599236645720 01 I	1	-0.28977775830749900 01 + 0.77645674639870700 00 I
ROOT(7) =	0.10037198385540710 01 + -0.99684656745825120 00 I	1	SOLVED BY DIRECT METHOD
ROOT(8) =	0.10001682256999940 01 + -0.99537592890632930 00 I	1	SOLVED BY DIRECT METHOD

Exhibit 6.6. Roots Are: 1+i(6), 1-i(6)

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER  1 OF DEGREE  6

THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.70000000000000010D 01 + -0.10500000000000000D 02 I
  P( 3) = -0.28000000000000000D 02 +  0.58000000000000010D 02 I
  P( 4) =  0.17100000000000000D 03 +  0.15000000000000000D 01 I
  P( 5) = -0.73000000000000000D 02 + -0.25100000000000000D 03 I
  P( 6) = -0.22800000000000000D 03 +  0.10400000000000000D 03 I
  P( 7) =  0.72000000000000010D 02 +  0.10400000000000000D 03 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.     0
MAXIMUM NUMBER OF ITERATIONS.             200
TEST FOR CONVERGENCE.                0.10D-09
TEST FOR MULTIPLICITIES.             0.10D-01
RADIUS TO START SEARCH.              0.00D 00
RADIUS TO END SEARCH.                0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)                                                    MULTIPLICITIES        INITIAL APPROXIMATION

 ROOT( 1) =  0.19999548887498100 01 +  0.20000177333639120 01 I        3        0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.10135026276728690 01 +  0.19961630831608820 01 I        1        0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.98663481225078640 00 +  0.20038068823255210 01 I        1        SOLVED BY DIRECT METHOD
 ROOT( 4) = -0.10000021061730840 01 +  0.49997683442186050 00 I        1        SOLVED BY DIRECT METHOD


AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)                                                    MULTIPLICITIES        INITIAL APPROXIMATION

 ROOT( 1) =  0.19999548354197870 01 +  0.20000176278985030 01 I        3        0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.10000001597792780 01 +  0.19999999796000450 01 I        1        0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.99999983067112130 00 +  0.20000000716850890 01 I        1        SOLVED BY DIRECT METHOD
 ROOT( 4) = -0.99999999999999980 00 +  0.50000000000000000 00 I        1        SOLVED BY DIRECT METHOD
```

Exhibit 6.7. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER  2 OF DEGREE 15

THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
  P( 2) =  0.25571200000000000D 03 + -0.38400000000000000D 03 I
  P( 3) = -0.73535568000000000D 02 + -0.21896960000000000D 04 I
  P( 4) = -0.38555656960000000D 04 + -0.69468514560000010 04 I
  P( 5) = -0.17333864648000000D 05 + -0.14206259728000000D 05 I
  P( 6) = -0.49679892704000010 05 + -0.17658574640000000D 05 I
  P( 7) = -0.10223945221300000D 06 + -0.60306642320000010 04 I
  P( 8) = -0.16427422005600000D 06 +  0.41373662304000000D 05 I
  P( 9) = -0.20366258884200000D 06 +  0.10938992276700000D 06 I
  P(10) = -0.18712557800100000D 06 +  0.19298654403300000D 06 I
  P(11) = -0.12749972985900000D 06 +  0.21713412274200000D 06 I
  P(12) = -0.28146927168000000D 05 +  0.19284897279600000D 06 I
  P(13) =  0.13294344348000000D 05 +  0.10381302265500000D 06 I
  P(14) =  0.30539007747000000D 05 +  0.29989891413000000D 05 I
  P(15) = -0.18358990200000000D 03 + -0.18276321600000000D 03 I
  P(16) =  0.27556200000000000D 00 +  0.27556200000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)                                                    MULTIPLICITIES          INITIAL APPROXIMATION

 ROOT( 1) =  0.30000000562593720-02 +  0.82779319321193550-12 I        2        0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.16864729610578890-04 +  0.10000018357040090 01 I        1        0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.12802271618969960-04 +  0.14999987299639190 01 I        1        0.38822847926540560 00 +  0.14488887631171930 01 I
 ROOT( 4) =  0.60196787821467610-04 +  0.29999366616581900 01 I        1       -0.51763825519667240 00 +  0.19318516083687550 01 I
 ROOT( 5) = -0.16864853931179430-04 +  0.99999816357761600 00 I        1       -0.17677671470807010 01 +  0.17677667588520150 01 I
 ROOT( 6) = -0.23333333333336490 01 +  0.32494094552714940-12 I        1       -0.28977775830749900 01 +  0.77645674639870700 00 I
 ROOT( 7) = -0.10001637690223590 01 + -0.10001737714599140 01 I        2       -0.33807402483312290 01 + -0.90586719408161600 00 I
 ROOT( 8) = -0.33349050543089760-03 + -0.15001333733984800 01 I        1       -0.28284266071078960 01 + -0.28284276423843900 01 I
 ROOT( 9) = -0.99967247835329040 00 + -0.99965213993255630 00 I        1       -0.11646848013998990 01 + -0.43466664598733680 01 I
 ROOT(10) =  0.33348119140084980-03 + -0.14998669162486160 01 I        1        0.12940963450986450 01 + -0.48296288314530270 01 I
 ROOT(11) = -0.17152209734821560-03 +  0.29999902162099160 01 I        1        0.38890882929795090 01 + -0.38890863000722580 01 I
 ROOT(12) =  0.11131368982465770-03 +  0.30000731498786090 01 I        1        SOLVED BY DIRECT METHOD
 ROOT(13) = -0.12764927758354440-04 +  0.15000012155052380 01 I        1        SOLVED BY DIRECT METHOD


AFTER THE ATTEMPT TO IMPROVE ACCURACY
```

Exhibit 6.8.

```
ROOTS OF P(X)                                                    MULTIPLICITIES          INITIAL APPROXIMATION

ROOT( 1) =  0.30000000562581190-02 +  0.82793704892536880-12 I        2          0.48296291156562790 00 +  0.12940952844381870 00 I
ROOT( 2) =  0.30010861721562150-07 +  0.10000000438390740 01 I        1          0.70710675530463460 00 +  0.70710680706845950 00 I
ROOT( 3) =  0.61890537903191320-07 +  0.15000001407680000 01 I        1          0.38822847926540560 00 +  0.14488887631171930 01 I
ROOT( 4) =  0.23973274100738230-04 +  0.30000087372252940 01 I        1         -0.51763825519667240 00 +  0.19318516083687550 01 I
ROOT( 5) =  0.39739845404284450-07 +  0.10000000424546840 01 I        1         -0.17677671470807010 01 +  0.17677667588520150 01 I
ROOT( 6) = -0.23333333333333340 01 + -0.30279561954291830-15 I        1         -0.28977775830749900 01 +  0.77645674639870700 00 I
ROOT( 7) = -0.10000069904049780 01 + -0.99999889122472430 00 I        2         -0.33807402483312290 01 + -0.90586719408161600 00 I
ROOT( 8) =  0.92844225443363060-08 + -0.15000000131286020 01 I        1         -0.28284266071078960 01 + -0.28284276423843900 01 I
ROOT( 9) = -0.10000062510065850 01 + -0.10000015374992320 01 I        1         -0.11646848013998990 01 + -0.43466664598733680 01 I
ROOT(10) =  0.16717142793063800-07 + -0.15000000185263290 01 I        1          0.12940963450986450 01 + -0.48296288314530270 01 I
ROOT(11) = -0.25587686230373010-04 +  0.30000216537917870 01 I        1          0.38890882929795090 01 + -0.38890863000722580 01 I
ROOT(12) =  0.21109261089828320-04 +  0.30000149073606530 01 I        1          SOLVED BY DIRECT METHOD
ROOT(13) = -0.54537993846922740-07 +  0.14999998637957010 01 I        1          SOLVED BY DIRECT METHOD
```

Exhibit 6.8. Roots Are: -2.33, .003 (2), i(2), 1.5i (2), -1.5i (2) 3i (3), -1-i(3)

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER  3 OF DEGREE  8

THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.50000000000000010D 01 + -0.11500000000000000D 02 I
  P( 3) = -0.51750000000000010D 02 +  0.43000000000000010D 02 I
  P( 4) =  0.15725000000000000D 03 +  0.14462500000000000D 03 I
  P( 5) =  0.30750000000000000D 03 + -0.34750000000000000D 03 I
  P( 6) = -0.49525000000000000D 03 + -0.49487500000000000D 03 I
  P( 7) = -0.58575000000000010D 03 +  0.42475000000000010D 03 I
  P( 8) =  0.18100000000000000D 03 +  0.44200000000000010D 03 I
  P( 9) =  0.15800000000000000D 03 +  0.60000000000000010D 01 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.000 00


BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)                                                    MULTIPLICITIES        INITIAL APPROXIMATION

 ROOT( 1) =  0.20000422028730180D 01 +  0.20000287439982940D 01 I        3          0.48296291156562790D 00 +  0.12940952844381870D 00 I
 ROOT( 2) =  0.10005638069089880D 01 +  0.19806389525682160D 01 I        1          0.70710675530463460D 00 +  0.70710680706845950D 00 I
 ROOT( 3) = -0.10635284187498440D 01 +  0.49423723791991200D 00 I        1          0.38822847926540560D 00 +  0.14488887631171930D 01 I
 ROOT( 4) = -0.97221225168698240D 00 +  0.56037974579731250D 00 I        1         -0.51763825519667240D 00 +  0.19318516083687550D 01 I
 ROOT( 5) =  0.99943706805064100D 00 +  0.20190942869286920D 01 I        1          SOLVED BY DIRECT METHOD
 ROOT( 6) = -0.96438681314185650D 00 +  0.44556354479098650D 00 I        1          SOLVED BY DIRECT METHOD


AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)                                                    MULTIPLICITIES        INITIAL APPROXIMATION

 ROOT( 1) =  0.20000421758895910D 01 +  0.20000287507138950D 01 I        3          0.48296291156562790D 00 +  0.12940952844381870D 00 I
 ROOT( 2) =  0.10000000231652550D 01 +  0.19999998057777000D 01 I        1          0.70710675530463460D 00 +  0.70710680706845950D 00 I
 ROOT( 3) = -0.10000065941223140D 01 +  0.49999528211260520D 00 I        1          0.38822847926540560D 00 +  0.14488887631171930D 01 I
 ROOT( 4) = -0.10000009725718650D 01 +  0.50000841757266510D 00 I        1         -0.51763825519667240D 00 +  0.19318516083687550D 01 I
 ROOT( 5) =  0.99999991684094000D 00 +  0.20000001001326590D 01 I        1          SOLVED BY DIRECT METHOD
 ROOT( 6) = -0.99999257381295570D 00 +  0.49999628183377030D 00 I        1          SOLVED BY DIRECT METHOD
```

Exhibit 6.9. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

```
MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER  4 OF DEGREE 12

THE COEFFICIENTS OF P(X) ARE

 P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P( 2) = -0.12000000000000000D 02 + -0.00000000000000000D 00 I
 P( 3) =  0.72000000000000010D 02 +  0.00000000000000000D 00 I
 P( 4) = -0.28000000000000000D 03 + -0.00000000000000000D 00 I
 P( 5) =  0.78000000000000000D 03 +  0.00000000000000000D 00 I
 P( 6) = -0.16320000000000000D 04 + -0.00000000000000000D 00 I
 P( 7) =  0.26240000000000000D 04 +  0.00000000000000000D 00 I
 P( 8) = -0.32640000000000000D 04 + -0.00000000000000000D 00 I
 P( 9) =  0.31200000000000000D 04 +  0.00000000000000000D 00 I
 P(10) = -0.22400000000000000D 04 + -0.00000000000000000D 00 I
 P(11) =  0.11520000000000000D 04 +  0.00000000000000000D 00 I
 P(12) = -0.38400000000000000D 03 + -0.00000000000000000D 00 I
 P(13) =  0.64000000000000010D 02 +  0.00000000000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.              0.10D-09
TEST FOR MULTIPLICITIES.           0.10D-01
RADIUS TO START SEARCH.            0.00D 00
RADIUS TO END SEARCH.              0.00D 00


BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)                                                    MULTIPLICITIES        INITIAL APPROXIMATION

 ROOT( 1) =  0.10044681929487390 01 +  0.10021861175327510 01 I        4        0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.98589784284950020 00 +  0.10068045189960200 01 I        1        0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.99622622484739470 00 +  0.98443823578951680 00 I        1        0.38822847926540560 00 +  0.14488887631171930 01 I
 ROOT( 4) =  0.10070694905203640 01 + -0.82784262477528560 00 I        2       -0.51763825519667240 00 +  0.19318516083687550 01 I
 ROOT( 5) =  0.75845181074234980 00 + -0.94103202050523650 00 I        1       -0.17677671470807010 01 +  0.17677667588520150 01 I
 ROOT( 6) =  0.84390135441641700 00 + -0.12151582772901180 01 I        1       -0.28977775830749900 01 +  0.77645674639870700 00 I
 ROOT( 7) =  0.12454795861515510 01 + -0.96097310967704480 00 I        1        SOLVED BY DIRECT METHOD
 ROOT( 8) =  0.11380314281571030 01 + -0.12271385678935710 01 I        1        SOLVED BY DIRECT METHOD


AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)                                                    MULTIPLICITIES        INITIAL APPROXIMATION

 ROOT( 1) =  0.10035767401254570 01 +  0.10017426347906060 01 I        4        0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.99905247886814380 00 +  0.10045121143842710 01 I        1        0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.10035967935676750 01 +  0.99656003170413060 00 I        1        0.38822847926540560 00 +  0.14488887631171930 01 I
```

Exhibit 6.10.

```
ROOT( 4) =  0.10007324697602040 01 + -0.99564295721310670 00 I        2        -0.51763825519667240 00 +  0.19318516083687550 01 I
ROOT( 5) =  0.10047654833534890 01 + -0.10006957183589740 01 I        1        -0.17677671470807010 01 +  0.17677667588520150 01 I
ROOT( 6) =  0.99947825662743310 00 + -0.99607660789158700 00 I        1        -0.28977775830749900 01 +  0.77645674639870700 00 I
ROOT( 7) =  0.99541663095228990 00 + -0.10001898299218090 01 I        1          SOLVED BY DIRECT METHOD
ROOT( 8) =  0.10000745905941600 01 + -0.99512853288497680 00 I        1          SOLVED BY DIRECT METHOD
```

Exhibit 6.10. Roots Are: 1+i (6), 1-i (6)

```
        GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER  1


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.     0.10D-02
TEST FOR CONVERGENCE.                0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.    0.10D-19
TEST FOR MULTIPLICITIES.             0.10D-01
RADIUS TO START SEARCH.              0.00D 00
RADIUS TO END SEARCH.                0.00D 00


THE DEGREE OF P(X) IS  6  THE COEFFICIENTS ARE


 P(7 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(6 ) = -0.70000000000000001D 01 + -0.10500000000000000D 02 I
 P(5 ) = -0.28000000000000000D 02 +  0.58000000000000010D 02 I
 P(4 ) =  0.17100000000000000D 03 +  0.15000000000000000D 01 I
 P(3 ) = -0.73000000000000000D 02 + -0.25100000000000000D 03 I
 P(2 ) = -0.22800000000000000D 03 +  0.10400000000000000D 03 I
 P(1 ) =  0.72000000000000010D 02 +  0.10400000000000000D 03 I


Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X).
THE DEGREE OF Q(X) IS  3  THE COEFFICIENTS ARE


 Q(4 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 Q(3 ) = -0.20000000000001500D 01 + -0.45000000000001180D 01 I
 Q(2 ) = -0.70000000000004230D 01 +  0.35000000000000600D 01 I
 Q(1 ) =  0.99999999999977620D 00 +  0.69999999999998120D 01 I


ROOTS OF Q(X)                                                          INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999975650D 00 +  0.19999999999995740D 01 I      0.48296291156562790D 00 +  0.12940952844381870D 00 I
 ROOT( 2) =  0.20000000000003470D 01 +  0.20000000000005290D 01 I        RESULTS OF SUBROUTINE QUAD
 ROOT( 3) = -0.99999999999995390D 00 +  0.50000000000001490D 00 I        RESULTS OF SUBROUTINE QUAD


ROOTS OF P(X)                                          MULTIPLICITIES  INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999975650D 00 +  0.19999999999995740D 01 I      2  0.48296291156562790D 00 +  0.12940952844381870D 00 I
 ROOT( 2) =  0.20000000000003470D 01 +  0.20000000000005300D 01 I      3    RESULTS OF SUBROUTINE QUAD
 ROOT( 3) = -0.99999999999995410D 00 +  0.50000000000001470D 00 I      1    RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.11. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

```
          GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER  2


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.     0.10D-02
TEST FOR CONVERGENCE.                0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.    0.10D-19
TEST FOR MULTIPLICITIES.             0.10D-01
RADIUS TO START SEARCH.              0.000 00
RADIUS TO END SEARCH.                0.00D 00


THE DEGREE OF P(X) IS 15  THE COEFFICIENTS ARE


 P(16) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
 P(15) =  0.25571200000000000D 03 + -0.38400000000000000D 03 I
 P(14) = -0.73535568000000000D 02 + -0.21896960000000000D 04 I
 P(13) = -0.38555656960000000D 04 + -0.69468514560000010D 04 I
 P(12) = -0.17333864648000000D 05 + -0.14206259728000000D 05 I
 P(11) = -0.49679892704000010D 05 + -0.17658574640000000D 05 I
 P(10) = -0.10223945221300000D 06 + -0.60306642320000010D 04 I
 P(9 ) = -0.16427422005600000D 06 +  0.41373662304000000D 05 I
 P(8 ) = -0.20366258884200000D 06 +  0.10938992276700000D 06 I
 P(7 ) = -0.18712557800100000D 06 +  0.19298654403300000D 06 I
 P(6 ) = -0.12749972985900000D 06 +  0.21713412274200000D 06 I
 P(5 ) = -0.28146927168000000D 05 +  0.19284897279600000D 06 I
 P(4 ) =  0.13294344348000000D 05 +  0.10381302265500000D 06 I
 P(3 ) =  0.30539007747000000D 05 +  0.29989891413000000D 05 I
 P(2 ) = -0.18358990200000000D 03 + -0.18276321600000000D 03 I
 P(1 ) =  0.27556200000000000D 00 +  0.27556200000000000D 00 I


Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X).
THE DEGREE OF Q(X) IS  7  THE COEFFICIENTS ARE


 Q(8 ) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
 Q(7 ) =  0.15985599999959083D 03 + -0.14400000000018463D 03 I
 Q(6 ) =  0.26751999999866287D 03 + -0.52756800000682380 03 I
 Q(5 ) =  0.32719599999781904D 03 + -0.91441600000759420 03 I
 Q(4 ) =  0.23015999960445530 02 + -0.15212520000037680 04 I
 Q(3 ) = -0.72072000047996880 02 + -0.13274279999966590 04 I
 Q(2 ) = -0.75578400005415630 03 + -0.75200400003332210 03 I
 Q(1 ) =  0.22680001335502310 01 +  0.22679999258286330 01 I


ROOTS OF Q(X)                                                    INITIAL APPROXIMATION

 ROOT( 1) =  0.30000000386459970-02 + -0.13705431766384130-09 I    0.482962911565627900 +  0.129409528443818700 I
 ROOT( 2) =  0.14666355435090540-09 +  0.10000000003025560 01 I    0.707106755304634600 +  0.707106807068459500 I
 ROOT( 3) = -0.16262060880959170-09 +  0.14999999998070670 01 I    0.388228479265405600 +  0.144888876311719300 I
 ROOT( 4) = -0.11677672021943630-10 + -0.15000000000067630 01 I   -0.517638255196672400 +  0.193185160836875500 I
```

Exhibit 6.12.

```
ROOT( 5) = -0.2333333333340885D 01 +  0.42923013959255554D-11 I          -0.17677671470807010 01 +  0.17677667588520150 01 I
ROOT( 6) =  0.1140452177139650D-09 +  0.30000000000539440 01 I             RESULTS OF SUBROUTINE QUAD
ROOT( 7) = -0.1000000000032260D 01 + -0.99999999998557810 00 I             RESULTS OF SUBROUTINE QUAD
```

ROOTS OF P(X)		MULTIPLICITIES	INITIAL APPROXIMATION	
ROOT(1) =	0.3000000038645997D-02 + -0.1370543176638413D-09 I	2	0.48296291156562790 00 +	0.12940952844381870 00 I
ROOT(2) =	0.1466637065231162D-09 + 0.10000000003025560 01 I	2	0.70710675530463460 00 +	0.70710680706845950 00 I
ROOT(3) =	-0.1626208545466292D-09 + 0.14999999998070660 01 I	2	0.38822847926540560 00 +	0.14488887631171930 01 I
ROOT(4) =	-0.1167769247572452D-10 + -0.1500000000006763D 01 I	2	-0.51763825519667240 00 +	0.19318516083687550 01 I
ROOT(5) =	-0.2333333333340885D 01 + 0.42915512506970990-11 I	1	-0.17677671470807010 01 +	0.17677667588520150 01 I
ROOT(6) =	0.1140455675609210D-09 + 0.30000000000539440 01 I	3	RESULTS OF SUBROUTINE QUAD	
ROOT(7) =	-0.1000000000032260D 01 + -0.99999999998557790 00 I	3	RESULTS OF SUBROUTINE QUAD	

Exhibit 6.12. Roots Are: -2.33, .003 (2), i(2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)

```
GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER  3


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS  8  THE COEFFICIENTS ARE


 P( 9 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P( 8 ) = -0.50000000000000001D 01 + -0.11500000000000000D 02 I
 P( 7 ) = -0.51750000000000001D 02 +  0.43000000000000001D 02 I
 P( 6 ) =  0.15725000000000000D 03 +  0.14462500000000000D 03 I
 P( 5 ) =  0.30750000000000000D 03 + -0.34750000000000000D 03 I
 P( 4 ) = -0.49525000000000000D 03 + -0.49487500000000000D 03 I
 P( 3 ) = -0.58575000000000001D 03 +  0.42475000000000001D 03 I
 P( 2 ) =  0.18100000000000000D 03 +  0.44200000000000001D 03 I
 P( 1 ) =  0.15800000000000000D 03 +  0.60000000000000001D 01 I


Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X).
THE DEGREE OF Q(X) IS  3  THE COEFFICIENTS ARE


 Q( 4 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 Q( 3 ) = -0.20000000000000273D 01 + -0.45000000000000275D 01 I
 Q( 2 ) = -0.70000000000000935D 01 +  0.35000000000000469D 01 I
 Q( 1 ) =  0.99999999999994173D 00 +  0.70000000000000498D 01 I


ROOTS OF Q(X)                                                          INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999995483D 00 +  0.19999999999999755D 01 I        0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.20000000000000665D 01 +  0.20000000000000539D 01 I          RESULTS OF SUBROUTINE QUAD
 ROOT( 3) = -0.99999999999999408D 00 +  0.49999999999999814D 00 I          RESULTS OF SUBROUTINE QUAD


ROOTS OF P(X)                                          MULTIPLICITIES      INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999995483D 00 +  0.19999999999999755D 01 I        2  0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.20000000000000665D 01 +  0.20000000000000539D 01 I        3    RESULTS OF SUBROUTINE QUAD
 ROOT( 3) = -0.99999999999999408D 00 +  0.49999999999999812D 00 I        3    RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.13. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

```
GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER  4


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.     0.10D-02
TEST FOR CONVERGENCE.                0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.    0.10D-19
TEST FOR MULTIPLICITIES.             0.10D-01
RADIUS TO START SEARCH.              0.000 00
RADIUS TO END SEARCH.                0.000 00


THE DEGREE OF P(X) IS 12  THE COEFFICIENTS ARE

 P(13) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(12) = -0.12000000000000000D 02 + -0.00000000000000000D 00 I
 P(11) =  0.72000000000000010D 02 +  0.00000000000000000D 00 I
 P(10) = -0.28000000000000000D 03 + -0.00000000000000000D 00 I
 P(9 ) =  0.78000000000000000D 03 +  0.00000000000000000D 00 I
 P(8 ) = -0.16320000000000000D 04 + -0.00000000000000000D 00 I
 P(7 ) =  0.26240000000000000D 04 +  0.00000000000000000D 00 I
 P(6 ) = -0.32640000000000000D 04 + -0.00000000000000000D 00 I
 P(5 ) =  0.31200000000000000D 04 +  0.00000000000000000D 00 I
 P(4 ) = -0.22400000000000000D 04 + -0.00000000000000000D 00 I
 P(3 ) =  0.11520000000000000D 04 +  0.00000000000000000D 00 I
 P(2 ) = -0.38400000000000000D 03 + -0.00000000000000000D 00 I
 P(1 ) =  0.64000000000000010D 02 +  0.00000000000000000D 00 I


Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X).
THE DEGREE OF Q(X) IS  2  THE COEFFICIENTS ARE

 Q(3 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 Q(2 ) = -0.20000000000000015D 01 + -0.00000000000000000D 00 I
 Q(1 ) =  0.19999999999998300D 01 +  0.00000000000000000D 00 I


ROOTS OF P(X)                                                MULTIPLICITIES

 ROOT( 1) =  0.10000000000000070D 01 +  0.99999999999990740 00 I        6          RESULTS OF SUBROUTINE QUAD
 ROOT( 2) =  0.10000000000000070D 01 + -0.99999999999990740 00 I        6          RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.14. Roots Are: 1+i (6), 1-i (6)

```
          GREATEST COMMON DIVISOR METHOD USED WITH MULLERS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER  1


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X)-IS  6  THE COEFFICIENTS ARE


 P(7 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(6 ) = -0.70000000000000001D 01 + -0.10500000000000000D 02 I
 P(5 ) = -0.28000000000000000D 02 +  0.58000000000000001D 02 I
 P(4 ) =  0.17100000000000000D 03 +  0.15000000000000000D 01 I
 P(3 ) = -0.73000000000000000D 02 + -0.25100000000000000D 03 I
 P(2 ) = -0.22800000000000000D 03 +  0.10400000000000000D 03 I
 P(1 ) =  0.72000000000000001D 02 +  0.10400000000000000D 03 I


Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X).
THE DEGREE OF Q(X) IS  3  THE COEFFICIENTS ARE


 Q(4 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 Q(3 ) = -0.20000000000000150D 01 + -0.45000000000000118D 01 I
 Q(2 ) = -0.70000000000000423D 01 +  0.35000000000000060D 01 I
 Q(1 ) =  0.99999999999997762D 00 +  0.69999999999999812D 01 I


ROOTS OF Q(X)                                                          INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999997565D 00 +  0.19999999999999574D 01 I      0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.20000000000000347D 01 +  0.20000000000000529D 01 I      SOLVED BY DIRECT METHOD
 ROOT( 3) = -0.99999999999999539D 00 +  0.50000000000000149D 00 I      SOLVED BY DIRECT METHOD


ROOTS OF P(X)                                          MULTIPLICITIES  INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999997568D 00 +  0.19999999999999574D 01 I        2  0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.20000000000000347D 01 +  0.20000000000000530D 01 I        3  RESULTS OF SUBROUTINE QUAD
 ROOT( 3) = -0.99999999999999541D 00 +  0.50000000000000148D 00 I        1  RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.15. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

```
        GREATEST COMMON DIVISOR METHOD USED WITH MULLERS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER  2


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS 15  THE COEFFICIENTS ARE


 P(16) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
 P(15) =  0.25571200000000000D 03 + -0.38400000000000000D 03 I
 P(14) = -0.73535568000000000D 02 + -0.21896960000000000D 04 I
 P(13) = -0.38555656960000000D 04 + -0.69468514560000010D 04 I
 P(12) = -0.17333864648000000D 05 + -0.14206259728000000D 05 I
 P(11) = -0.49679892704000010D 05 + -0.17658574640000000D 05 I
 P(10) = -0.10223945221300000D 06 + -0.60306642320000010D 04 I
 P(9 ) = -0.16427422005600000D 06 +  0.41373662304000000D 05 I
 P(8 ) = -0.20366258884200000D 06 +  0.10938992276700000D 06 I
 P(7 ) = -0.18712557800100000D 06 +  0.19298654403300000D 06 I
 P(6 ) = -0.12749972985900000D 06 +  0.21713412274200000D 06 I
 P(5 ) = -0.28146927168000000D 05 +  0.19284897279600000D 06 I
 P(4 ) =  0.13294344348000000D 05 +  0.10381302265500000D 06 I
 P(3 ) =  0.30539007747000000D 05 +  0.29989891413000000D 05 I
 P(2 ) = -0.18358990200000000D 03 + -0.18276321600000000D 03 I
 P(1 ) =  0.27556200000000000D 00 +  0.27556200000000000D 00 I


Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X).
THE DEGREE OF Q(X) IS  7  THE COEFFICIENTS ARE


 Q(8 ) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
 Q(7 ) =  0.15985599999959083D 03 + -0.14400000000184663D 03 I
 Q(6 ) =  0.26751999998662870D 03 + -0.52756800000682380D 03 I
 Q(5 ) =  0.32719599997819040D 03 + -0.91441600007594200D 03 I
 Q(4 ) =  0.23015999960445530D 02 + -0.15212520000037680D 04 I
 Q(3 ) = -0.72072000047996880D 02 + -0.13274279999966590D 04 I
 Q(2 ) = -0.75578400005415630D 03 + -0.75200400003332210D 03 I
 Q(1 ) =  0.22680001335502310D 01 +  0.22679999258286330D 01 I


ROOTS OF Q(X)                                                    INITIAL APPROXIMATION

 ROOT( 1) =  0.30000000386459970D-02 + -0.13705431778101480D-09 I    0.48296291156562790D 00 +  0.12940952844381870D 00 I
 ROOT( 2) =  0.14666372491173310D-09 +  0.10000000003025560D 01 I    0.70710675530463460D 00 +  0.70710680706845950D 00 I
 ROOT( 3) = -0.16262101465322160D-09 +  0.14999999998070670D 01 I    0.38822847926540560D 00 +  0.14488887631171930D 01 I
 ROOT( 4) =  0.11404545919875120D-09 +  0.30000000000539440D 01 I   -0.51763825519667240D 00 +  0.19318516083687550D 01 I
```

Exhibit 6.16.

```
ROOT( 5) = -0.23333333333408840 01 +  0.4292669751093708D-11 I          -0.17677671470807010 01 +  0.17677667588520150 01 I
ROOT( 6) = -0.11677251758139090-10 + -0.15000000000067630 01 I           SOLVED BY DIRECT METHOD
ROOT( 7) = -0.10000000000322610 01 + -0.99999999998557870 00 I           SOLVED BY DIRECT METHOD

ROOTS OF P(X)                                          MULTIPLICITIES     INITIAL APPROXIMATION

ROOT( 1) =  0.30000000386459970-02 + -0.13705431778187450-09 I    2      0.48296291156562790 00 +  0.12940952844381870 00 I
ROOT( 2) =  0.14666360907717090-09 +  0.10000000003025560 01 I    2      0.70710675530463460 00 +  0.70710680706845950 00 I
ROOT( 3) = -0.16262091348697400-09 +  0.14999999998070660 01 I    2      0.38822847926540560 00 +  0.14488887631171930 01 I
ROOT( 4) =  0.11404556580843080-09 +  0.30000000000539440 01 I    3     -0.51763825519667240 00 +  0.19318516083687550 01 I
ROOT( 5) = -0.23333333333408850 01 +  0.4291402714880148D-11 I    1     -0.17677671470807010 01 +  0.17677667588520150 01 I
ROOT( 6) = -0.11677735717066360-10 + -0.15000000000067630 01 I    2      RESULTS OF SUBROUTINE QUAD
ROOT( 7) = -0.10000000000322600 01 + -0.99999999998557790 00 I    3      RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.16. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)

```
          GREATEST COMMON DIVISOR METHOD USED WITH MULLERS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER  3


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS  8  THE COEFFICIENTS ARE


 P(9 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(8 ) = -0.50000000000000001D 01 + -0.11500000000000000D 02 I
 P(7 ) = -0.51750000000000001D 02 +  0.43000000000000001D 02 I
 P(6 ) =  0.15725000000000000D 03 +  0.14462500000000000D 03 I
 P(5 ) =  0.30750000000000000D 03 + -0.34750000000000000D 03 I
 P(4 ) = -0.49525000000000000D 03 + -0.49487500000000000D 03 I
 P(3 ) = -0.58575000000000001D 03 +  0.42475000000000001D 03 I
 P(2 ) =  0.18100000000000000D 03 +  0.44200000000000001D 03 I
 P(1 ) =  0.15800000000000000D 03 +  0.60000000000000001D 01 I


Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X).
THE DEGREE OF Q(X) IS  3  THE COEFFICIENTS ARE


 Q(4 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 Q(3 ) = -0.20000000000000273D 01 + -0.45000000000000275D 01 I
 Q(2 ) = -0.70000000000000935D 01 +  0.35000000000000469D 01 I
 Q(1 ) =  0.99999999999941730D 00 +  0.70000000000004980 01 I


ROOTS OF Q(X)                                                          INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999954830D 00 +  0.19999999999999755D 01 I      0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.20000000000000665D 01 +  0.20000000000000539D 01 I      SOLVED BY DIRECT METHOD
 ROOT( 3) = -0.99999999999994080 00 +  0.49999999999999140 00 I       SOLVED BY DIRECT METHOD


ROOTS OF P(X)                                          MULTIPLICITIES  INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999954830D 00 +  0.19999999999999755D 01 I      2  0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.20000000000000665D 01 +  0.20000000000000539D 01 I      3  RESULTS OF SUBROUTINE QUAD
 ROOT( 3) = -0.99999999999994100 00 +  0.49999999999999120 00 I       3  RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.17. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

```
          GREATEST COMMON DIVISOR METHOD USED WITH MULLERS METHOD TO FIND ZEROS OF POLYNOMIALS
          POLYNOMIAL NUMBER  4


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
TEST FOR MULTIPLICITIES.            0.10D-01
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS 12  THE COEFFICIENTS ARE


 P(13) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(12) = -0.12000000000000000D 02 + -0.00000000000000000D 00 I
 P(11) =  0.72000000000000001D 02 +  0.00000000000000000D 00 I
 P(10) = -0.28000000000000000D 03 + -0.00000000000000000D 00 I
 P(9 ) =  0.78000000000000000D 03 +  0.00000000000000000D 00 I
 P(8 ) = -0.16320000000000000D 04 + -0.00000000000000000D 00 I
 P(7 ) =  0.26240000000000000D 04 +  0.00000000000000000D 00 I
 P(6 ) = -0.32640000000000000D 04 + -0.00000000000000000D 00 I
 P(5 ) =  0.31200000000000000D 04 +  0.00000000000000000D 00 I
 P(4 ) = -0.22400000000000000D 04 + -0.00000000000000000D 00 I
 P(3 ) =  0.11520000000000000D 04 +  0.00000000000000000D 00 I
 P(2 ) = -0.38400000000000000D 03 + -0.00000000000000000D 00 I
 P(1 ) =  0.64000000000000001D 02 +  0.00000000000000000D 00 I


Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X).
THE DEGREE OF Q(X) IS  2  THE COEFFICIENTS ARE


 Q(3 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 Q(2 ) = -0.20000000000000015D 01 + -0.00000000000000000D 00 I
 Q(1 ) =  0.19999999999998300 01 +  0.00000000000000000D 00 I


ROOTS OF P(X)                                                    MULTIPLICITIES

 ROOT( 1) =  0.10000000000000070 01 +  0.99999999999990740 00 I       6          RESULTS OF SUBROUTINE QUAD
 ROOT( 2) =  0.10000000000000070 01 + -0.99999999999990740 00 I       6          RESULTS OF SUBROUTINE QUAD
```

Exhibit 6.18. Roots Are: 1+i(6), 1-i (6)

```
          REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS  METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
          POLYNOMIAL NUMBER   1


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS  6  THE COEFFICIENTS ARE


 P( 7 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P( 6 ) = -0.70000000000000001D 01 + -0.10500000000000000D 02 I
 P( 5 ) = -0.28000000000000000D 02 +  0.58000000000000001D 02 I
 P( 4 ) =  0.17100000000000000D 03 +  0.15000000000000000D 01 I
 P( 3 ) = -0.73000000000000000D 02 + -0.25100000000000000D 03 I
 P( 2 ) = -0.22800000000000000D 03 +  0.10400000000000000D 03 I
 P( 1 ) =  0.72000000000000001D 02 +  0.10400000000000000D 03 I



**********************************************************************************************************************************


THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  1


 G( 2 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G( 1 ) =  0.99999999999973500D 00 + -0.50000000000175420D 00 I


ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

 ROOT( 1) = -0.99999999999973500D 00 +  0.50000000000175420D 00 I        1                 NO INITIAL APPROXIMATIONS



**********************************************************************************************************************************


THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  2


 G( 2 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G( 1 ) = -0.99999999999991790 00 + -0.19999999999968450 01 I
```

Exhibit 6.19.

```
ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999991790 00 +  0.19999999999968450 01 I        2                 NO INITIAL APPROXIMATIONS

*****************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  3

    2 ) =  0.10000000000000000 01 +  0.00000000000000000 00 I
    1 ) = -0.19999999999999670 01 + -0.20000000000015190 01 I

ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

 ROOT( 3) =  0.19999999999999670 01 +  0.20000000000015190 01 I        3                 NO INITIAL APPROXIMATIONS
```

Exhibit 6.19. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

```
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS  METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
POLYNOMIAL NUMBER  2


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS 15  THE COEFFICIENTS ARE


 P(16) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
 P(15) =  0.25571200000000000D 03 + -0.38400000000000000D 03 I
 P(14) = -0.73535568000000000D 02 + -0.21896960000000000D 04 I
 P(13) = -0.38555656960000000D 04 + -0.69468514560000010D 04 I
 P(12) = -0.17333864648000000D 05 + -0.14206259728000000D 05 I
 P(11) = -0.49679892704000010D 05 + -0.17658574640000000D 05 I
 P(10) = -0.10223945221300000D 06 + -0.60306642320000010D 04 I
 P(9 ) = -0.16427422005600000D 06 +  0.41373662304000000D 05 I
 P(8 ) = -0.20366258884200000D 06 +  0.10938992276700000D 06 I
 P(7 ) = -0.18712557800100000D 06 +  0.19298654403300000D 06 I
 P(6 ) = -0.12749972985900000D 06 +  0.21713412274200000D 06 I
 P(5 ) = -0.28146927168000000D 05 +  0.19284897279600000D 06 I
 P(4 ) =  0.13294344348000000D 05 +  0.10381302265500000D 06 I
 P(3 ) =  0.30539007747000000D 05 +  0.29989891413000000D 05 I
 P(2 ) = -0.18358990200000000D 03 + -0.18276321600000000D 03 I
 P(1 ) =  0.27556200000000000D 00 +  0.27556200000000000D 00 I


**********************************************************************************************************************************


THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  1


 G(2 ) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
 G(1 ) =  0.11199999928960310 03 + -0.82749886587407680-06 I


ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

 ROOT( 1) = -0.23333333185333970 01 +  0.17239559705709930-07 I        1                NO INITIAL APPROXIMATIONS
```

Exhibit 6.20.

```
**********************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  2

 G( 5 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G( 4 ) = -0.29999706558634380-02 + -0.99999996563627250 00 I
 G( 3 ) =  0.22499999311788850 01 +  0.30000585981173610-02 I
 G( 2 ) = -0.67501099024602780-02 + -0.22500001290676650 01 I
 G( 1 ) =  0.46434794653826830-06 +  0.67496039821701720-02 I


ROOTS OF G(X)                                                                    INITIAL APPROXIMATION

 ROOT( 1) =  0.2999823203921417D-02 + -0.2056988264688649D-06 I                  0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.4767581845696651D-06 +  0.10000005558501840 01 I                  0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.2331721616933697D-07 + -0.14999999724317250 01 I                      RESULTS OF SUBROUTINE QUAD
 ROOT( 4) = -0.3526234587180479D-06 +  0.14999995879166400 01 I                      RESULTS OF SUBROUTINE QUAD


ROOTS OF P(X)                                                  MULTIPLICITIES          INITIAL APPROXIMATION

 ROOT( 1) =  0.2999823203921417D-02 + -0.2056988264688661D-06 I        2          0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.4767581845685002D-06 +  0.10000005558501840 01 I        2          0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.2340917191042791D-07 + -0.14999999724317250 01 I        2                NO INITIAL APPROXIMATIONS
 ROOT( 4) = -0.3527154144585488D-06 +  0.14999995879166400 01 I        2                NO INITIAL APPROXIMATIONS


**********************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  3

 G( 3 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G( 2 ) =  0.99999998537055410D 00 + -0.20000000171626320 01 I
 G( 1 ) =  0.30000000025333350 01 + -0.30000000317915140 01 I


ROOTS OF P(X)                                                  MULTIPLICITIES          INITIAL APPROXIMATION

 ROOT( 1) =  0.1462930432349907D-07 +  0.30000000171626330 01 I        3                NO INITIAL APPROXIMATIONS
 ROOT( 2) = -0.9999999999998584D 00 + -0.10000000000000020 01 I        3                NO INITIAL APPROXIMATIONS
```

Exhibit 6.20. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2), 3i (3), -1-i (3)

```
      REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS  METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
      POLYNOMIAL NUMBER  3


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS  8  THE COEFFICIENTS ARE


 P(9 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(8 ) = -0.50000000000000001D 01 + -0.11500000000000000D 02 I
 P(7 ) = -0.51750000000000001D 02 +  0.43000000000000001D 02 I
 P(6 ) =  0.15725000000000000D 03 +  0.14462500000000000D 03 I
 P(5 ) =  0.30750000000000000D 03 + -0.34750000000000000D 03 I
 P(4 ) = -0.49525000000000000D 03 + -0.49487500000000000D 03 I
 P(3 ) = -0.58575000000000001D 03 +  0.42475000000000001D 03 I
 P(2 ) =  0.18100000000000000D 03 +  0.44200000000000001D 03 I
 P(1 ) =  0.15800000000000000D 03 +  0.60000000000000001D 01 I



**********************************************************************************************************************************


NO ROOTS OF MULTIPLICITY  1



**********************************************************************************************************************************


THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  2

 G(2 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G(1 ) = -0.99999999999464980D 00 + -0.19999999999964680D 01 I

ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999464980D 00 +  0.19999999999964680D 01 I        2                NO INITIAL APPROXIMATIONS
```

Exhibit 6.21.

```
**********************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  3

 G(3 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G(2 ) = -0.10000000000025390D 01 + -0.25000000000016290D 01 I
 G(1 ) = -0.30000000000032960D 01 + -0.10000000000004210D 01 I


ROOTS OF P(X)                                                  MULTIPLICITIES              INITIAL APPROXIMATION

 ROOT( 1) =  0.20000000000025320D 01 +  0.20000000000016530D 01 I        3                NO INITIAL APPROXIMATIONS
 ROOT( 2) = -0.99999999999999260D 00 +  0.49999999999997590D 00 I        3                NO INITIAL APPROXIMATIONS
```

Exhibit 6.21. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

```
          REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS  METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
          POLYNOMIAL NUMBER  4


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS 12  THE COEFFICIENTS ARE

 P(13) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(12) = -0.12000000000000000D 02 + -0.00000000000000000D 00 I
 P(11) =  0.72000000000000010D 02 +  0.00000000000000000D 00 I
 P(10) = -0.28000000000000000D 03 + -0.00000000000000000D 00 I
 P(9 ) =  0.78000000000000000D 03 +  0.00000000000000000D 00 I
 P(8 ) = -0.16320000000000000D 04 + -0.00000000000000000D 00 I
 P(7 ) =  0.26240000000000000D 04 +  0.00000000000000000D 00 I
 P(6 ) = -0.32640000000000000D 04 + -0.00000000000000000D 00 I
 P(5 ) =  0.31200000000000000D 04 +  0.00000000000000000D 00 I
 P(4 ) = -0.22400000000000000D 04 + -0.00000000000000000D 00 I
 P(3 ) =  0.11520000000000000D 04 +  0.00000000000000000D 00 I
 P(2 ) = -0.38400000000000000D 03 + -0.00000000000000000D 00 I
 P(1 ) =  0.64000000000000010D 02 +  0.00000000000000000D 00 I



**********************************************************************************************************************************************************


NO ROOTS OF MULTIPLICITY  1



**********************************************************************************************************************************************************


NO ROOTS OF MULTIPLICITY  2
```

Exhibit 6.22.

```
**********************************************************************************************************************************

NO ROOTS OF MULTIPLICITY  3

**********************************************************************************************************************************

NO ROOTS OF MULTIPLICITY  4

**********************************************************************************************************************************

NO ROOTS OF MULTIPLICITY  5

**********************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  6

 G(3 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G(2 ) = -0.20000000000000066D 01 + -0.00000000000000000D 00 I
 G(1 ) =  0.20000000000000070 01 +  0.00000000000000000D 00 I

ROOTS OF P(X)                                                    MULTIPLICITIES          INITIAL APPROXIMATION

 ROOT( 1) =  0.10000000000000330 01 +  0.99999999999997060 00 I        6              NO INITIAL APPROXIMATIONS
 ROOT( 2) =  0.10000000000000330 01 + -0.99999999999997060 00 I        6              NO INITIAL APPROXIMATIONS
```

Exhibit 6.22. Roots Are: 1+i (6), 1-i (6)

```
      REPEATED USE OF THE GREATEST COMMON DIVISOR AND MULLERS  METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
      POLYNOMIAL NUMBER  1


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS  6  THE COEFFICIENTS ARE

 P(7 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(6 ) = -0.70000000000000010D 01 + -0.10500000000000000D 02 I
 P(5 ) = -0.28000000000000000D 02 +  0.58000000000000010D 02 I
 P(4 ) =  0.17100000000000000D 03 +  0.15000000000000000D 01 I
 P(3 ) = -0.73000000000000000D 02 + -0.25100000000000000D 03 I
 P(2 ) = -0.22800000000000000D 03 +  0.10400000000000000D 03 I
 P(1 ) =  0.72000000000000010D 02 +  0.10400000000000000D 03 I

**********************************************************************************************************************************


THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  1

 G(2 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G(1 ) =  0.99999999999973500 00 + -0.50000000000175420 00 I


ROOTS OF G(X)                                                                    INITIAL APPROXIMATION

 ROOT( 1) = -0.99999999999973500 00 +  0.50000000000175410 00 I                0.48296291156562790 00 +  0.12940952844381870 00 I


ROOTS OF P(X)                                          MULTIPLICITIES            INITIAL APPROXIMATION

 ROOT( 1) = -0.99999999999973500 00 +  0.50000000000175410 00 I        1       0.48296291156562790 00 +  0.12940952844381870 00 I


**********************************************************************************************************************************
```

Exhibit 6.23.

```
THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  2

 G(2 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G(1 ) = -0.99999999999991790D 00 + -0.19999999999968450D 01 I


ROOTS OF G(X)                                                                    INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999991790D 00 +  0.19999999999968450D 01 I                 0.48296291156562790D 00 +  0.12940952844381870D 00 I

ROOTS OF P(X)                                          MULTIPLICITIES            INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999991790D 00 +  0.19999999999968450D 01 I        2        0.48296291156562790D 00 +  0.12940952844381870D 00 I

**********************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  3

    2 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
    1 ) = -0.19999999999999670D 01 + -0.20000000000015190D 01 I

ROOTS OF P(X)                                          MULTIPLICITIES            INITIAL APPROXIMATION

 ROOT( 3) =  0.19999999999999670D 01 +  0.20000000000015190D 01 I        3        NO INITIAL APPROXIMATIONS
```

Exhibit 6.23. Roots Are: 2+2i (3), 1+2i (2), -1+.5i

```
          REPEATED USE OF THE GREATEST COMMON DIVISOR AND MULLERS  METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
          POLYNOMIAL NUMBER  2


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS 15  THE COEFFICIENTS ARE


 P(16) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
 P(15) =  0.25571200000000000D 03 + -0.38400000000000000D 03 I
 P(14) = -0.73535568000000000D 02 + -0.21896960000000000D 04 I
 P(13) = -0.38555656960000000D 04 + -0.69468514560000010D 04 I
 P(12) = -0.17333864648000000D 05 + -0.14206259728000000D 05 I
 P(11) = -0.49679892704000010D 05 + -0.17658574640000000D 05 I
 P(10) = -0.10223945221300000D 06 + -0.60306642320000010D 04 I
 P(9 ) = -0.16427422005600000D 06 +  0.41373662304000000D 05 I
 P(8 ) = -0.20366258884200000D 06 +  0.10938992276700000D 06 I
 P(7 ) = -0.18712557800100000D 06 +  0.19298654403300000D 06 I
 P(6 ) = -0.12749972985900000D 06 +  0.21713412274200000D 06 I
 P(5 ) = -0.28146927168000000D 05 +  0.19284897279600000D 06 I
 P(4 ) =  0.13294344348000000D 05 +  0.10381302265500000D 06 I
 P(3 ) =  0.30539007747000000D 05 +  0.29989891413000000D 05 I
 P(2 ) = -0.18358990200000000D 03 + -0.18276321600000000D 03 I
 P(1 ) =  0.27556200000000000D 00 +  0.27556200000000000D 00 I



**********************************************************************************************************************************


THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  1


 G(2 ) =  0.48000000000000000D 02 +  0.00000000000000000D 00 I
 G(1 ) =  0.11199999992896031D 03 + -0.82749886587407680-06 I


ROOTS OF G(X)                                                                    INITIAL APPROXIMATION

 ROOT( 1) = -0.23333333185333970 01 +  0.17239559705709930-07 I                  0.48296291156562790 00 +  0.12940952844381870 00 I


ROOTS OF P(X)                                          MULTIPLICITIES            INITIAL APPROXIMATION
```

Exhibit 6.24.

```
ROOT( 1) = -0.2333333318533397D 01 +  0.1723955970570993D-07 I          1          0.48296291156562790 00 +  0.12940952844381870 00 I

*****************************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  2

 G(5 ) =  0.10000000000000000 01 +  0.00000000000000000 00 I
 G(4 ) = -0.29999706558634380-02 + -0.99999996563627250 00 I
 G(3 ) =  0.22499999311788850 01 +  0.30000585981173610-02 I
 G(2 ) = -0.67501099024602780-02 + -0.22500001290676650 01 I
 G(1 ) =  0.46434794653826830-06 +  0.67496039821701720-02 I

ROOTS OF G(X)                                                                  INITIAL APPROXIMATION

 ROOT( 1) =  0.29998232039214170-02 + -0.20569882646886600-06 I                0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.47675818456954060-06 +  0.10000005558501840 01 I                0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.23317216169372120-07 + -0.14999999972431720 01 I                SOLVED BY DIRECT METHOD
 ROOT( 4) = -0.35262345871801270-06 +  0.14999995879166400 01 I                SOLVED BY DIRECT METHOD

ROOTS OF P(X)                                                 MULTIPLICITIES   INITIAL APPROXIMATION

 ROOT( 1) =  0.29998232039214170-02 + -0.20569882646886500-06 I          2     0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.47675818456831650-06 +  0.10000005558501840 01 I          2     0.70710675530463460 00 +  0.70710680706845950 00 I
 ROOT( 3) =  0.23409171910410740-07 + -0.14999999972431720 01 I          2     NO INITIAL APPROXIMATIONS
 ROOT( 4) = -0.35271541445867050-06 +  0.14999995879166400 01 I          2     NO INITIAL APPROXIMATIONS

*****************************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  3

 G(3 ) =  0.10000000000000000 01 +  0.00000000000000000 00 I
 G(2 ) =  0.99999998537055410 00 + -0.20000000171626320 01 I
 G(1 ) =  0.30000000025333350 01 + -0.30000000317915140 01 I

ROOTS OF G(X)                                                                  INITIAL APPROXIMATION
```

Exhibit 6.24.

```
ROOT( 1) = -0.9999999999985870D 00 + -0.1000000000000002D 01 I          0.48296291156562790 00 +  0.12940952844381870 00 I
ROOT( 2) =  0.1462930461493261D-07 +  0.3000000017162634D 01 I          SOLVED BY DIRECT METHOD

ROOTS OF P(X)                                                MULTIPLICITIES      INITIAL APPROXIMATION

ROOT( 1) = -0.9999999999985860 00 + -0.1000000000000002D 01 I        3          0.48296291156562790 00 +  0.12940952844381870 00 I
ROOT( 2) =  0.1462930456963442D-07 +  0.3000000017162634D 01 I        3          NO INITIAL APPROXIMATIONS
```

Exhibit 6.24. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)

```
        REPEATED USE OF THE GREATEST COMMON DIVISOR AND MULLERS  METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
        POLYNOMIAL NUMBER  3


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS  8  THE COEFFICIENTS ARE


 P(9 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(8 ) = -0.50000000000000001D 01 + -0.11500000000000000D 02 I
 P(7 ) = -0.51750000000000001D 02 +  0.43000000000000001D 02 I
 P(6 ) =  0.15725000000000000D 03 +  0.14462500000000000D 03 I
 P(5 ) =  0.30750000000000000D 03 + -0.34750000000000000D 03 I
 P(4 ) = -0.49525000000000000D 03 + -0.49487500000000000D 03 I
 P(3 ) = -0.58575000000000001D 03 +  0.42475000000000001D 03 I
 P(2 ) =  0.18100000000000000D 03 +  0.44200000000000001D 03 I
 P(1 ) =  0.15800000000000000D 03 +  0.60000000000000001D 01 I



**********************************************************************************************************************************


NO ROOTS OF MULTIPLICITY  1




**********************************************************************************************************************************


THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  2


 G(2 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G(1 ) = -0.99999999999464980D 00 + -0.19999999999964680D 01 I


ROOTS OF G(X)                                                                    INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999464980D 00 +  0.19999999999964680D 01 I                0.48296291156562790D 00 +  0.12940952844381870D 00 I
```

Exhibit 6.25.

```
ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

 ROOT( 1) =  0.99999999999464980 00 +  0.19999999999964680 01 I         2          0.48296291156562790 00 +  0.12940952844381870 00 I


**********************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  3

 G(3 ) =  0.10000000000000000 01 +  0.00000000000000000 00 I
 G(2 ) = -0.10000000000025390 01 + -0.25000000000016290 01 I
 G(1 ) = -0.30000000000032960 01 + -0.10000000000004210 01 I

ROOTS OF G(X)                                                                                INITIAL APPROXIMATION

 ROOT( 1) = -0.99999999999999280 00 +  0.49999999999997590 00 I                0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.20000000000025320 01 +  0.20000000000016530 01 I                SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT( 2) =  0.20000000000025320 01 +  0.20000000000016530 01 I
DID NOT CONVERGE AFTER 200 ITERATIONS
THE PRESENT APPROXIMATION IS  0.20019999742533280 01 +  0.20019999742524480 01 I

ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

 ROOT( 1) = -0.99999999999999300 00 +  0.49999999999997590 00 I         3          0.48296291156562790 00 +  0.12940952844381870 00 I

NOT ALL ROOTS OF THE ABOVE POLYNOMIAL,G, WERE FOUND
```

Exhibit 6.25. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)

```
        REPEATED USE OF THE GREATEST COMMON DIVISOR AND MULLERS METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
        POLYNOMIAL NUMBER  4


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   0
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR ZERO IN SUBROUTINE GCD.    0.10D-02
TEST FOR CONVERGENCE.               0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD.   0.10D-19
RADIUS TO START SEARCH.             0.00D 00
RADIUS TO END SEARCH.               0.00D 00


THE DEGREE OF P(X) IS 12  THE COEFFICIENTS ARE


 P(13) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 P(12) = -0.12000000000000000D 02 + -0.00000000000000000D 00 I
 P(11) =  0.72000000000000010D 02 +  0.00000000000000000D 00 I
 P(10) = -0.28000000000000000D 03 + -0.00000000000000000D 00 I
 P(9 ) =  0.78000000000000000D 03 +  0.00000000000000000D 00 I
 P(8 ) = -0.16320000000000000D 04 + -0.00000000000000000D 00 I
 P(7 ) =  0.26240000000000000D 04 +  0.00000000000000000D 00 I
 P(6 ) = -0.32640000000000000D 04 + -0.00000000000000000D 00 I
 P(5 ) =  0.31200000000000000D 04 +  0.00000000000000000D 00 I
 P(4 ) = -0.22400000000000000D 04 + -0.00000000000000000D 00 I
 P(3 ) =  0.11520000000000000D 04 +  0.00000000000000000D 00 I
 P(2 ) = -0.38400000000000000D 03 + -0.00000000000000000D 00 I
 P(1 ) =  0.64000000000000010D 02 +  0.00000000000000000D 00 I



**********************************************************************************************************************************************************


NO ROOTS OF MULTIPLICITY  1



**********************************************************************************************************************************************************


NO ROOTS OF MULTIPLICITY  2
```

Exhibit 6.26.

```
**********************************************************************************************************************************

NO ROOTS OF MULTIPLICITY  3

**********************************************************************************************************************************

NO ROOTS OF MULTIPLICITY  4

**********************************************************************************************************************************

NO ROOTS OF MULTIPLICITY  5

**********************************************************************************************************************************

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY  6

 G( 3 ) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
 G( 2 ) = -0.20000000000000066D 01 + -0.00000000000000000D 00 I
 G( 1 ) =  0.20000000000000070D 01 +  0.00000000000000000D 00 I

ROOTS OF G(X)                                                                    INITIAL APPROXIMATION

 ROOT( 1) =  0.10000000000000330 01 +  0.99999999999997070 00 I                  0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.10000000000000330 01 + -0.99999999999997070 00 I                  SOLVED BY DIRECT METHOD

ROOTS OF P(X)                                                 MULTIPLICITIES     INITIAL APPROXIMATION

 ROOT( 1) =  0.10000000000000330 01 +  0.99999999999997070 00 I        6         0.48296291156562790 00 +  0.12940952844381870 00 I
 ROOT( 2) =  0.10000000000000330 01 + -0.99999999999997070 00 I        6         NO INITIAL APPROXIMATIONS
```

Exhibit 6.26. Roots Are: 1+i (6), 1-i (6)

REFERENCES

1. S. D. Conte, Elementary Numerical Analysis, McGraw Hill, New York, 1965.

2. Peter Henrici, Elements of Numerical Analysis, John Wiley and Sons, Inc., New York, 1964.

3. Thomas R. McCalla, Introduction to Numerical Methods and FORTRAN Programming, John Wiley and Sons, Inc., New York, 1967.

4. David E. Muller, A method for solving algebraic equations using an automatic computer, Math. Tables and Aids to Comp., 10 (1956), 208-215.

APPENDIX A

SPECIAL FEATURES OF NEWTON'S AND MULLER'S PROGRAMS

Several special features have been provided in each program as an aid to the user and to improve accuracy of the results. These are explained and illustrated below.*

1. Generating Approximations

If the user does not have initial approximations available, subroutine GENAPP can systematically generate, for an N^{th} degree polynomial, N initial approximations of increasing magnitude, beginning with the magnitude specified by XSTART. If XSTART is 0., XSTART is automatically initialized to 0.5 to avoid the approximation 0. + 0.i. The approximations are generated according to the formula:

$$X_K = (XSTART + 0.5K)\ (Cos\ \beta + i\ Sin\ \beta)$$

where

$$\beta = \frac{\Pi}{12} + K\frac{\Pi}{6}\ ,\ K = 0,1,2,...$$

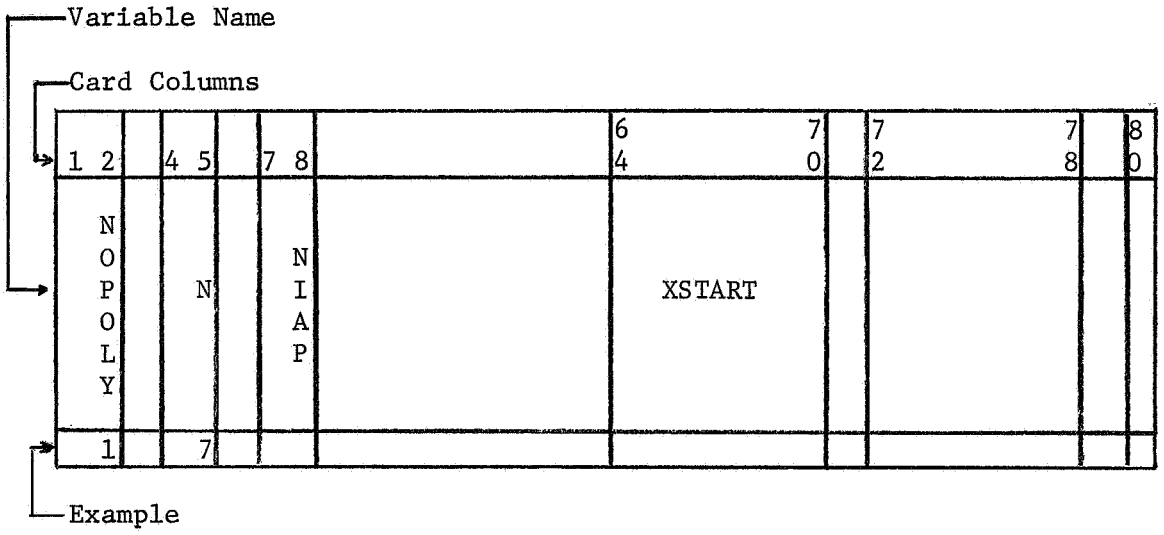
To accomplish this, the user defined the number of initial approximations to be read (NIAP) on the control card to be zero (0) or these

---

*These illustrations are representative of Newton's method in double precision. The control cards for Muller's method are similarly prepared.

columns (7-8) may be left blank. If XSTART is left blank, it is interpreted as 0.

For example, a portion of a control card which generates initial approximations beginning at the origin for a seventh degree polynomial is shown in Example A.1.



Example A.1

The approximations are generated in a spiral configuration as illustrated in Figure A.1. Exhibit 6.1 is an example of output resulting from generated approximations.

Example A.2 shows a portion of a control card which generated initial approximations beginning at a magnitude of 25.0 for a sixth degree polynomial.

1 2		4 5		7 8		64 70		72 78		80
NOPOLY		N		NIAP		XSTART				
1		6				2.5D+01				

Example A.2

Note that if the approximations are generated beginning at the origin, the order in which the roots are found will probably be of increasing magnitude. Roots obtained in this way are usually more accurate.

2. Altering Approximations

If an initial approximation, X_0, does not produce convergence to a zero within the maximum number of iterations, it is systematically altered a maximum of five times until convergence is possibly obtained according to the following formulas:

If the number of the alteration is odd: $(j = 1,3)$

$$X_{j+1} = |X_0| \ (\text{Cos } \beta + i \ \text{Sin } \beta) \text{ where}$$

$$\beta = \text{Tan}^{-1} \frac{\text{Im } X_0}{\text{Re } X_0} + K \frac{\Pi}{3} \ ; \quad K = 1 \text{ if } j = 1, \ 2 \text{ if } j = 3.$$

If the number of the alteration is even: $(j = 0,2,4)$

$$X_{j+1} = -X_j.$$

Each altered approximation is then taken as a starting approximation. Each initial or altered approximation which does not produce convergence is printed as in Exhibit A.1. If none of the six starting approximations produce convergence, the next initial approximation is taken, and the process repeated. The six approximations are spaced 60 degrees apart on a circle of radius $|X_0|$ centered at the origin as illustrated in Figure A.2.

3. Searching the Complex Plane

By use of initial approximations and the altering technique, any region of the complex plane in the form of an annulus centered at the origin can be searched for roots. This procedure can be accomplished in two ways.

The first way is more versatile but requires more effort on the part of the user. Specifically selected initial approximations can be used to define particular regions to be searched. For example, if the roots of a particular polynomial are known to have magnitudes between 20 and 40, an annulus of inner radius 20 and outer radius 40 could be searched by using the initial approximations 20. + i, 23. + i, 26. + i, 29. + i, 32. + i, 35. + i, 38. + i, 40. + i.

By generating initial approximations internally, the program can search an annulus centered at the origin of inner radius XSTART and outer radius XEND. Values for XSTART and XEND are supplied on the control card by the user. Example A.3 shows a portion of a control card to search the above annulus of inner radius 20.0 and outer radius 40.0.

1 2		4 5		7 8		64 70		72 78		80
NOPOLY		N		NIAP		XSTART		XEND		
1		7				2.0D+01		4.0D+01		

Example A.3

Note that since not less than N initial approximations can be generated at one time, the outer radius of the annulus actually searched may be greater than XEND but not greater than XEND + .5N.

Example A.4 shows a control card to search a circle of radius 15.

1 2		4 5		7 8		64 70		72 78		80
NOPOLY		N		NIAP		XSTART		XEND		
2		7						1.5D+01		

Example A.4

Figure A.3 shows the distribution of initial and altered approximations for an annulus of width 2 and inner radius a.

4. Improving Zeros Found

After the zeros of a polynomial are found, they are printed under the heading "Before the Attempt to Improve Accuracy." They are then used as initial approximations with Newton's (Muller's) method applied each time to the full (undeflated) polynomial. In most cases, zeros that have lost accuracy due to roundoff error in the deflation process are improved. The improved zeros are then printed under the heading "After the Attempt to Improve Accuracy." Since each root is used as an approximation to the original (undeflated) polynomial, it is possible that the root may converge to an entirely different root. This is especially true where several zeros are close together. Therefore, the user should check both lists of zeros to determine whether or not this has occurred. See Exhibit 6.4.

5. Solving Quadratic Polynomial

After N-2 roots of an N^{th} degree polynomial have been extracted, the remaining quadratic, $aX^2 + bX + c$, is solved using the quadratic formula

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

for the two remaining roots. These are indicated by the words "Solved By Direct Method" in the initial approximation column. If only a polynomial of degree 1 is to be solved, the solution is found directly as $(X - C) = 0$ implies $X - C$.

6. Missing Roots

If not all N roots of an N^{th} degree polynomial are found, the coefficients of the remaining deflated polynomial are printed under the heading "Coefficients of Deflated Polynomial For Which No Zeros Were Found." The user may then work with this polynomial in an attempt to find the remaining roots. The coefficient of the highest degree term will be printed first (Exhibit A.2).

7. Miscellaneous

By using various combinations of values for NIAP, XSTART, and XEND, the user has several options available as illustrated below.

Example A.5 shows the control card for a seventh degree polynomial. Three initial approximations are supplied by the user. At most three distinct roots will be found and the remaining deflated polynomial will be printed (Exhibit A.2).

1 2		4 5		7 8		64 70		72 78		80
NOPOLY		N		NIAP		XSTART		XEND		
1		7		3						

Example A.5

Note that if several roots are known to the user, they may be "divided out" of the original polynomial by using this procedure.

Example A.6 indicates that 2 initial approximations are supplied by the user to a 7^{th} degree polynomial. After these approximations are used the circle of radius 15 will be searched for the remaining roots.

1 2	4 5	7 8	64 70	72 78	80
NOPOLY	N	NIAP	XSTART	XEND	
1	7	2		1.5D+01	

Example A.6

By defining XSTART between 0. and 15. an annulus instead of the circle will be searched (Exhibit A.3).

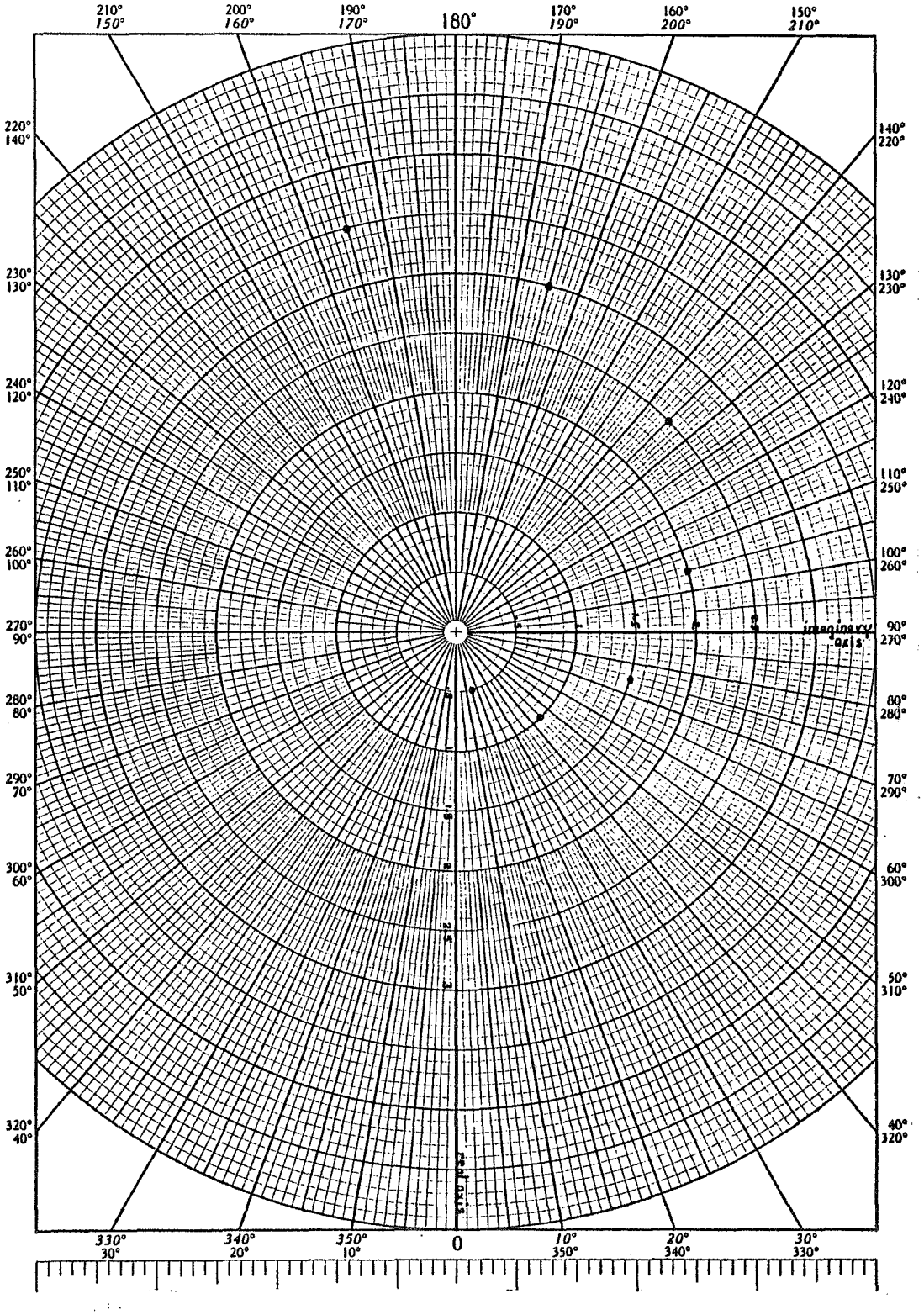


Figure A.1. Generating Initial Approximations

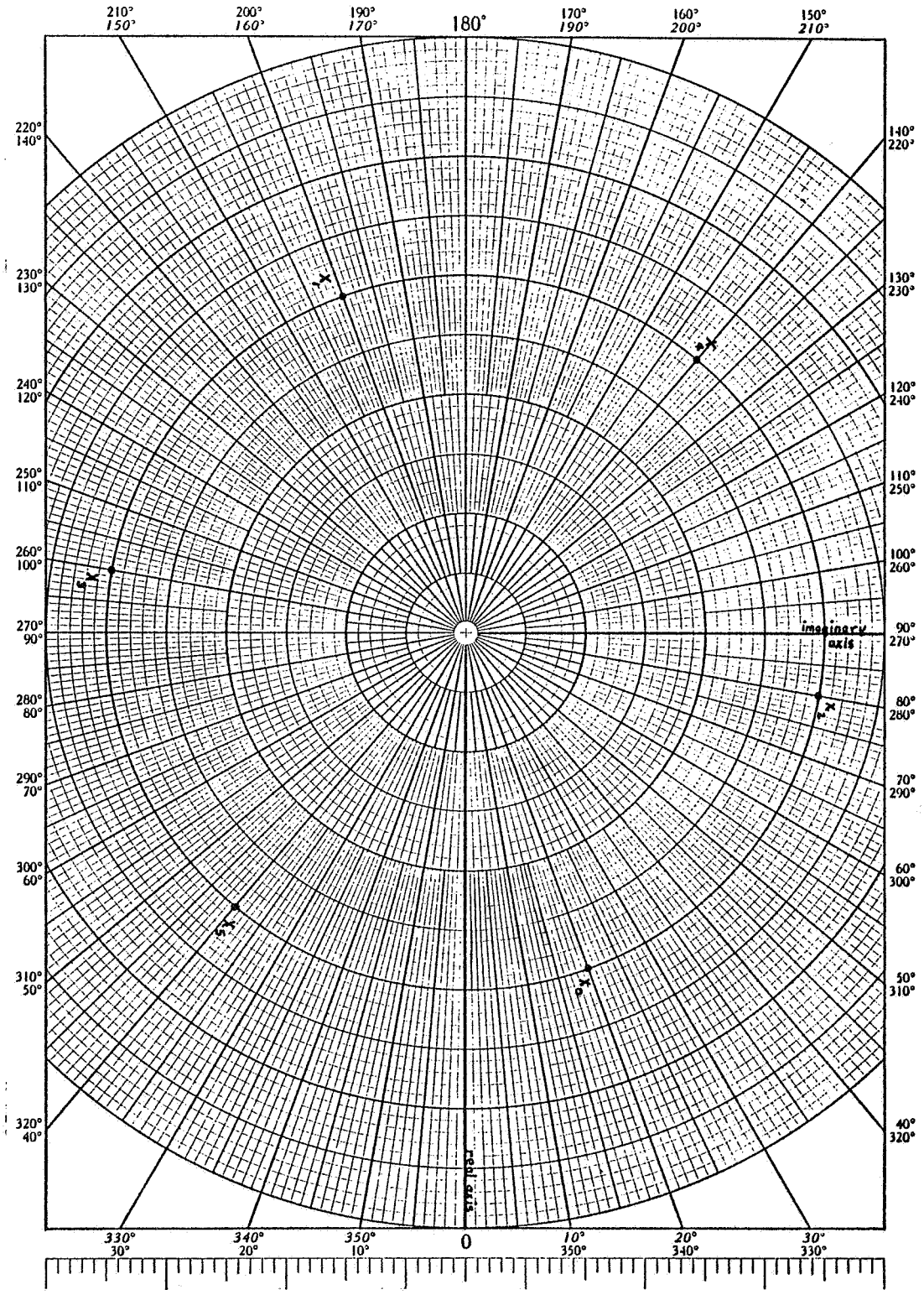


Figure A.2. Altering Approximations

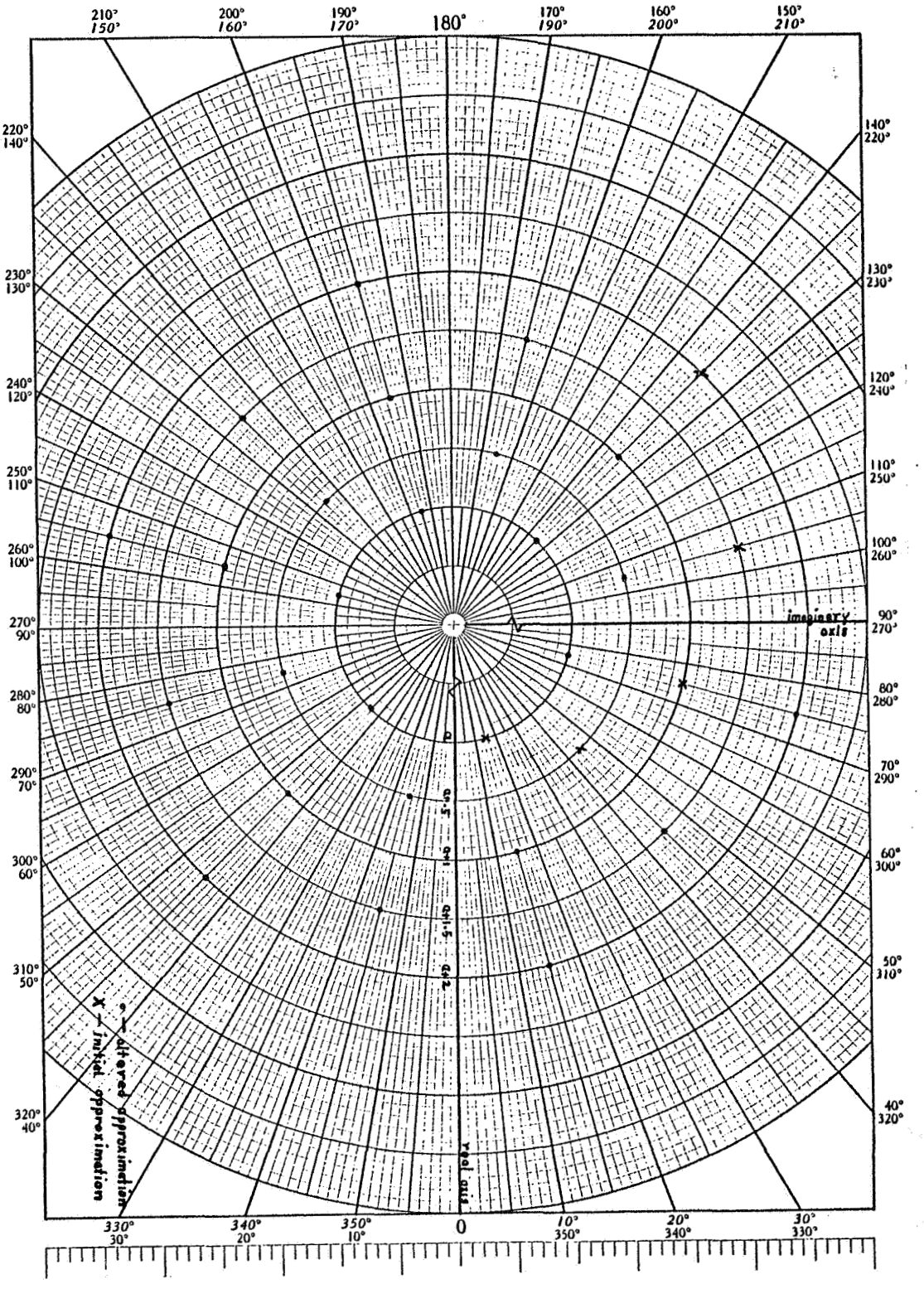


Figure A.3. Distribution of Approximations

```
      NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
      POLYNOMIAL NUMBER  2 OF DEGREE  3


THE COEFFICIENTS OF P(X) ARE

   P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
   P( 2) =  0.20000000000000000D 01 +  0.00000000000000000D 00 I
   P( 3) = -0.10000000000000000D 01 + -0.00000000000000000D 00 I
   P( 4) = -0.20000000000000000D 01 + -0.00000000000000000D 00 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    0
MAXIMUM NUMBER OF ITERATIONS.              3
TEST FOR CONVERGENCE.                  0.10D-03
TEST FOR MULTIPLICITIES.               0.10D-01
RADIUS TO START SEARCH.                0.00D 00
RADIUS TO END SEARCH.                  0.00D 00


NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER   3 ITERATIONS.


 0.48296291156562790 00 +  0.12940952844381870 00 I          INITIAL APPROXIMATION
-0.48296291156562790 00 + -0.12940952844381870 00 I          ALTERED APPROXIMATION
 0.12940949308846860 00 +  0.48296292103906440 00 I          ALTERED APPROXIMATION
-0.12940949308846860 00 + -0.48296292103906440 00 I          ALTERED APPROXIMATION
-0.35355342941614020 00 +  0.35355335177040300 00 I          ALTERED APPROXIMATION
 0.35355342941614020 00 + -0.35355335177040300 00 I          ALTERED APPROXIMATION

 0.70710675530463460 00 +  0.70710680706845950 00 I          INITIAL APPROXIMATION
-0.70710675530463460 00 + -0.70710680706845950 00 I          ALTERED APPROXIMATION
-0.25881912759833590 00 +  0.96592580418437740 00 I          ALTERED APPROXIMATION
 0.25881912759833590 00 + -0.96592580418437740 00 I          ALTERED APPROXIMATION
-0.96592586102499680 00 +  0.25881891546623570 00 I          ALTERED APPROXIMATION
 0.96592586102499680 00 + -0.25881891546623570 00 I          ALTERED APPROXIMATION

 0.38822847926540560 00 +  0.14488887631171930 01 I          INITIAL APPROXIMATION
-0.38822847926540560 00 + -0.14488887631171930 01 I          ALTERED APPROXIMATION
-0.10606602882484210 01 +  0.10606600553112090 01 I          ALTERED APPROXIMATION
 0.10606602882484210 01 + -0.10606600553112090 01 I          ALTERED APPROXIMATION
-0.14488886778562400 01 + -0.38822879746355020 00 I          ALTERED APPROXIMATION
 0.14488886778562400 01 +  0.38822879746355020 00 I          ALTERED APPROXIMATION


COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

   D( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
   D( 2) =  0.20000000000000000D 01 +  0.00000000000000000D 00 I
   D( 3) = -0.10000000000000000D 01 + -0.00000000000000000D 00 I
   D( 4) = -0.20000000000000000D 01 + -0.00000000000000000D 00 I
```

Exhibit A.1.

```
        NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
        POLYNOMIAL NUMBER  1 OF DEGREE  7


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.10000000000000000D 01 +  0.11000000000000000D 02 I
  P( 3) = -0.59000000000000010D 02 + -0.29000000000000000D 02 I
  P( 4) =  0.19500000000000000D 03 + -0.16900000000000000D 03 I
  P( 5) =  0.70000000000000010D 02 +  0.72300000000000000D 03 I
  P( 6) = -0.16240000000000000D 04 + -0.69600000000000010D 03 I
  P( 7) =  0.19220000000000000D 04 + -0.18320000000000000D 04 I
  P( 8) =  0.15960000000000000D 04 +  0.16920000000000000D 04 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.   3
MAXIMUM NUMBER OF ITERATIONS.           200
TEST FOR CONVERGENCE.              0.100-09
TEST FOR MULTIPLICITIES.           0.100-01
RADIUS TO START SEARCH.            0.000 00
RADIUS TO END SEARCH.              0.000 00


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

 ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

  ROOT( 1) = -0.29999999999999970 01 + -0.30000000000000020 01 I        1       -0.35000000000000000 01 + -0.35000000000000000 01 I
  ROOT( 2) =  0.20000000000000000 01 +  0.20000000000000000 01 I        1        0.25000000000000000 01 +  0.25000000000000000 01 I
  ROOT( 3) = -0.99999999999999620 00 + -0.39999999999999940 01 I        1       -0.15000000000000000 01 + -0.45000000000000010 01 I


AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

 ROOTS OF P(X)                                                    MULTIPLICITIES              INITIAL APPROXIMATION

  ROOT( 1) = -0.29999999999999960 01 + -0.30000000000000010 01 I        1       -0.35000000000000000 01 + -0.35000000000000000 01 I
  ROOT( 2) =  0.20000000000000000 01 +  0.20000000000000000 01 I        1        0.25000000000000000 01 +  0.25000000000000000 01 I
  ROOT( 3) = -0.99999999999999740 00 + -0.39999999999999960 01 I        1       -0.15000000000000000 01 + -0.45000000000000010 01 I


COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

  D( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  D( 2) = -0.29999999999999930D 01 +  0.60000000000000050D 01 I
  D( 3) = -0.20000000000000050D 02 + -0.18999999999999990 02 I
  D( 4) =  0.41000000000000030D 02 + -0.22000000000000080 02 I
  D( 5) =  0.23000000000000000D 02 +  0.41000000000000090 02 I
```

Exhibit A.2. Roots Are: -1 - 4i, -2 - 3i, -3 - 3i, -1 - i, 2 + 2i, 4 - i, 2 - i.

```
      NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS
      POLYNOMIAL NUMBER  1 OF DEGREE  7


THE COEFFICIENTS OF P(X) ARE

  P( 1) =  0.10000000000000000D 01 +  0.00000000000000000D 00 I
  P( 2) = -0.10000000000000000D 01 +  0.11000000000000000D 02 I
  P( 3) = -0.59000000000000001D 02 + -0.29000000000000000D 02 I
  P( 4) =  0.19500000000000000D 03 + -0.16900000000000000D 03 I
  P( 5) =  0.70000000000000001D 02 +  0.72300000000000000D 03 I
  P( 6) = -0.16240000000000000D 04 + -0.69600000000000001D 03 I
  P( 7) =  0.19220000000000000D 04 + -0.18320000000000000D 04 I
  P( 8) =  0.15960000000000000D 04 +  0.16920000000000000D 04 I


NUMBER OF INITIAL APPROXIMATIONS GIVEN.    2
MAXIMUM NUMBER OF ITERATIONS.            200
TEST FOR CONVERGENCE.                0.10D-09
TEST FOR MULTIPLICITIES.             0.10D-01
RADIUS TO START SEARCH.              0.70D 01
RADIUS TO END SEARCH.                0.15D 02


BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

 ROOTS OF P(X)                                                    MULTIPLICITIES          INITIAL APPROXIMATION

  ROOT( 1) = -0.29999999999999997D 01 + -0.30000000000000002D 01 I       1        -0.35000000000000000D 01 + -0.35000000000000000D 01 I
  ROOT( 2) =  0.20000000000000000D 01 +  0.20000000000000000D 01 I       1         0.25000000000000000D 01 +  0.25000000000000000D 01 I
  ROOT( 3) =  0.40000000000000001D 01 + -0.10000000000000001D 01 I       1         0.67614807619187791D 01 +  0.18117333982134620D 01 I
  ROOT( 4) =  0.19999999999999997D 01 + -0.99999999999999976D 00 I       1         0.53033006647847600D 01 +  0.53033010530134470D 01 I
  ROOT( 5) = -0.99999999999999998D 00 + -0.10000000000000004D 01 I       1         0.20705518894154970D 01 +  0.77274067366250320D 01 I
  ROOT( 6) = -0.99999999999999976D 00 + -0.39999999999999950D 01 I       1         SOLVED BY DIRECT METHOD
  ROOT( 7) = -0.20000000000000004D 01 + -0.30000000000000010D 01 I       1         SOLVED BY DIRECT METHOD

AFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE

 ROOTS OF P(X)                                                    MULTIPLICITIES          INITIAL APPROXIMATION

  ROOT( 1) = -0.29999999999999996D 01 + -0.30000000000000001D 01 I       1        -0.35000000000000000D 01 + -0.35000000000000000D 01 I
  ROOT( 2) =  0.20000000000000000D 01 +  0.20000000000000000 01 I        1         0.25000000000000000D 01 +  0.25000000000000000D 01 I
  ROOT( 3) =  0.40000000000000001D 01 + -0.10000000000000001D 01 I       1         0.67614807619187791D 01 +  0.18117333982134620D 01 I
  ROOT( 4) =  0.20000000000000000D 01 + -0.99999999999999980 00 I        1         0.53033006647847600D 01 +  0.53033010530134470D 01 I
  ROOT( 5) = -0.99999999999999997D 00 + -0.10000000000000000D 01 I       1         0.20705518894154970D 01 +  0.77274067366250320D 01 I
  ROOT( 6) = -0.99999999999999982D 00 + -0.40000000000000000D 01 I       1         SOLVED BY DIRECT METHOD
  ROOT( 7) = -0.20000000000000003D 01 + -0.30000000000000070 01 I        1         SOLVED BY DIRECT METHOD
```

Exhibit A.3. Roots Are: $-1 - 4i$, $-2 - 3i$, $-3 - 3i$, $-1 - i$, $2 + 2i$, $4 - i$, $2 - i$.

APPENDIX B

NEWTON'S METHOD

1. Use of the Program

A double precision FORTRAN IV program using Newton's method is presented here. Flow charts for this program are given in Figure B.6 while Table B.VIII gives a FORTRAN IV listing of this program. Single precision variables are listed in some of the tables. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from the appropriate tables.

The program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N, where N > 25, certain array dimensions must be changed. These are listed in Table B.I for the main program and subprograms in double precision.

TABLE B.I

PROGRAM CHANGES FOR SOLVING POLYNOMIALS
OF DEGREE GREATER THAN 25
BY NEWTON'S METHOD

Double Precision

Main Program

RA(N+1), VA(N+1)
RB(N+1), VB(N+1)
RC(N+1), VC(N+1)
RD(N+1), VD(N+1)
RCOEF(N+1), VCOEF(N+1)
MULT(N)
RXZERO(N), VXZERO(N)
RX(N), VX(N)
RXINIT(N), VXINIT(N)

Subroutine HORNER

RA(N+1), VA(N+1)
RB(N+1), VB(N+1)
RC(N+1), VC(N+1)

Subroutine BETTER

RXZERO(N), VXZERO(N)
RX(N), VX(N)
RA(N+1), VA(N+1)
RCOEF(N+1), VCOEF(N+1)
RC(N+1), VC(N+1)
RB(N+1), VB(N+1)

Subroutine GENAPP

APPR(N), APPI(N)

Subroutine QUAD

UA(N+1), VA(N+1)
UROOT(N), VROOT(N)
MULTI(N)

Table B.II lists the system functions used in the program of Newton's method. In the table "d" denotes a double precision variable name.

TABLE B.II

SYSTEM FUNCTIONS USED IN NEWTON'S METHOD

Double Precision

DABS(d)	- obtain absolute value
DCOS(d)	- obtain cosine of angle
DSIN(d)	- obtain sine of angle
DATAN2(d_1,d_2)	- arctangent of d_1/d_2
DSQRT(d)	- square root

2. Input Data for Newton's Method

The input data for Newton's method is grouped into polynomial data sets. Each polynomial data set consists of the data for one and only one polynomial. As many polynomials as the user desires may be solved by placing the polynomial data sets one behind the other. Each polynomial data set consists of three kinds of information placed in the following order:

1. Control information.
2. Coefficients of the polynomial.
3. Initial approximations. These may be omitted as described in Appendix A, § 1.

An end card follows the entire collection of data sets. It indicates that there is no more data to follow and terminates execution of the program. This information is displayed in Figure B.1 and described below. For the double precision data, the D-type specification should

be used. All data should be right justified. The recommendations given in Table B.III are those found to give best results on the IBM 360/50 computer which has a 32 bit word.

Control Information

The control card is the first card of the polynomial data set and contains the information given in Table B.III. See Figure B.2.

TABLE B.III

CONTROL DATA FOR NEWTON'S METHOD

Variable Name	Card Columns	Description
NOPOLY	c.c. 1-2	Number of the polynomial. Integer. Right justified.
N	c.c. 4-5	Degree of the polynomial. Integer. Right justified.
NIAP	c.c. 7-8	Number of initial approximations to be read. Integer. If no approximations are given, this should be left blank.
MAX	c.c. 19-21	Maximum number of iterations. Integer. Right justified. 200 is recommended.
EPSCNV	c.c. 30-35	Convergence requirement. Double precision. 1.D-10 is recommended.

TABLE B.III (Continued)

Variable Name	Card Columns	Description
EPSQ	c.c. 37-42	Tolerance check for zero (0) in subroutine QUAD. Double precision. Right justify. 1.D-20 is recommended.
EPSMUL	c.c. 44-49	Multiplicity requirement. Double precision. Right justify. 1.D-02 is recommended.
XSTART	c.c. 64-70	Magnitude at which to begin generating initial approximations. Double precision. Right justify. This is a special feature of the program and may be omitted.
XEND	c.c. 72-78	Magnitude at which to end the generating of initial approximations. Double precision. Right justify. This is a special feature of the program and may be omitted.
KCHECK	c.c. 80	This should be left blank.

Coefficients of the Polynomial

The coefficient cards follow the control card. For an N^{th} degree polynomial, N+1 coefficients must be entered one per card. The coefficient of the highest degree term is entered first. For example, if the polynomial $X^5 + 3X^4 + 2X + 5$ were to be solved, the order in which the coefficients would be entered is: 1, 3, 0, 0, 2, 5. Each

coefficient is entered, one per card, as described in Table B.IV and illustrated in Figure B.3.

TABLE B.IV

COEFFICIENT DATA FOR NEWTON'S METHOD

Variable Name	Card Columns	Description
RA (A in single precision)	c.c. 1-30	Real part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0D00.
VA (A in single precision)	c.c. 31-60	Imaginary part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0D00.

Initial Approximations

The initial approximation cards follow the set of coefficient cards. The number of initial approximations read must be the number specified on the control card and are entered, one per card, as given in Table B.V and illustrated in Figure B.4.

TABLE B.V

INITIAL APPROXIMATION DATA FOR NEWTON'S METHOD

Variable Name	Card Columns	Description
RXZERO (XZERO in single precision)	c.c. 1-30	Real part of complex number. Double precision. Right justify. If none, leave blank or enter 0.0D00.
VXZERO (XZERO in single precision)	c.c. 31-60	Imaginary part of complex number. Double precision. Right justify. If none, leave blank or enter 0.0D00.

End Card

The end card is the last card of the input data to the program. It indicates that there is no more data to be read. When this card is read, program execution is terminated. This card is described in Table B.VI and illustrated in Figure B.5.

TABLE B.VI

DATA TO END EXECUTION OF NEWTON'S METHOD

Variable Name	Card Columns	Description
KCHECK	c.c. 80	Must contain the number 1. Integer.

3. Variables Used in Newton's Method

The definitions of the major variables used in Newton's method are given in Table B.VII. The symbols used to indicate type are:

R - real variable

I - integer variable

C - complex variable

D - double precision

L - logical variable

A - alphanumeric variable

When two variables are listed, the one on the left is the real part of the corresponding single precision complex variable; the one on the right is the imaginary part. The symbols used to indicate disposition are:

E - entered

R - returned

ECR - entered, changed, and returned

C - variable in common

4. Description of Program Output

The output from Newton's method programs consist of the following information.

The number and degree of the polynomial are printed in the heading (Exhibit 6.1).

The coefficients are printed under the heading "THE COEFFICIENTS OF P(X) ARE." The coefficient of the highest degree term is listed first (Exhibit 6.1).

As an aid to ensure the control information is correct, the number of initial approximations given, maximum number of iterations, test for convergence, test for multiplicities, radius to start search, and radius to end search are printed as read from the control card (Exhibit 6.1).

The zeros found before and after the attempt to improve accuracy are printed. See Appendix A, § 4 for further explanation (Exhibit 6.1).

If not all zeros of the polynomial are found, the coefficients of the remaining unsolved polynomial will be printed, with coefficient of highest degree term first, under the heading "COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." See Appendix A, § 6. This is illustrated in Exhibit A.2.

The multiplicity of each zero is given under the title "MULTIPLICITIES" (Exhibit 6.1).

The initial approximation producing convergence to a root is printed to the right of the corresponding root and headed by "INITIAL APPROXIMATION." The initial approximations may be those supplied by the user, or generated by the program, or a combination of both (Exhibit A.3). See Appendix A, § 1 and § 2 for discussion of approximations. The message "SOLVED BY DIRECT METHOD" indicates that the corresponding root or roots was obtained by Subroutine QUAD. See Appendix A, § 5.

If an approximation does not produce convergence within the maximum number of iterations, it is printed under the heading "NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER XXX ITERATIONS." XXX is replaced by the maximum number of iterations. The type of the approximation, that is, initial approximation or altered approximations is given (Exhibit A.1). See Appendix A, § 1 and § 2 for discussion of approximations.

5. Informative and Error Messages

The output may contain informative or error messages. These are intended as an aid to the user and are described as follows:

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(X) = YYY DID NOT CONVERGE THE PRESENT APPROXIMATION AFTER ZZZ ITERATIONS IS PRINTED BELOW." X is the number of the zero, YYY is the value of the zero before the attempt to improve accuracy, ZZZ is the maximum number of iterations. This message indicates that a zero found before attempting to improve accuracy did not converge sufficiently when being used as an initial approximation on the full (undeflated) polynomial. The current approximation is printed in the list of improved zeros. In many cases, this failure to converge is a result of an ill-conditioned polynomial and this current approximation of the root may be better than its approximation before the attempt to improve accuracy. In most cases, the polynomial from which this root was first extracted had fewer multiple roots, due to deflations, than the original polynomial.

"THE VALUE OF THE DERIVATIVE AT XO = XXX IS ZERO."

This message is printed as a result of the value of the derivative of the original polynomial at an approximation, XXX, being zero (0). It occurred in the attempt to improve the accuracy of a zero. The previous message is then printed.

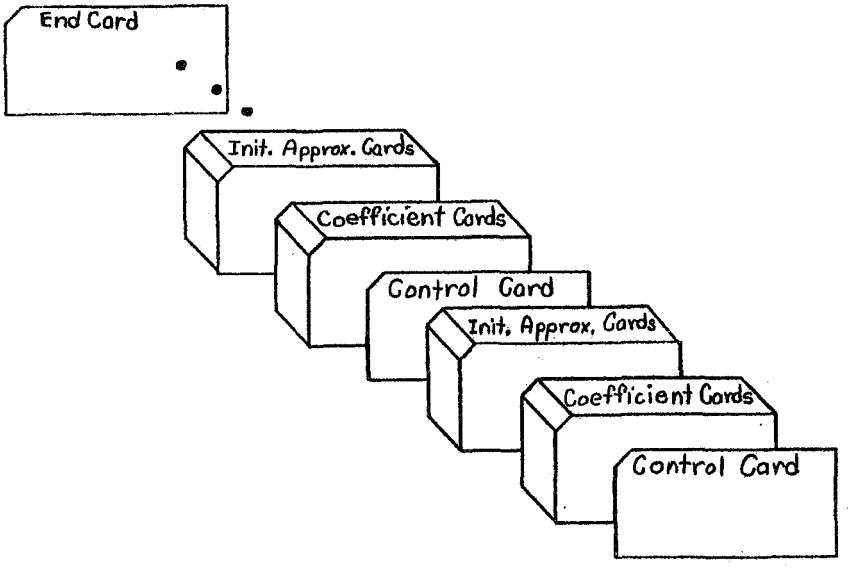


Figure B.1. Sequence of Input Data for Newton's Method

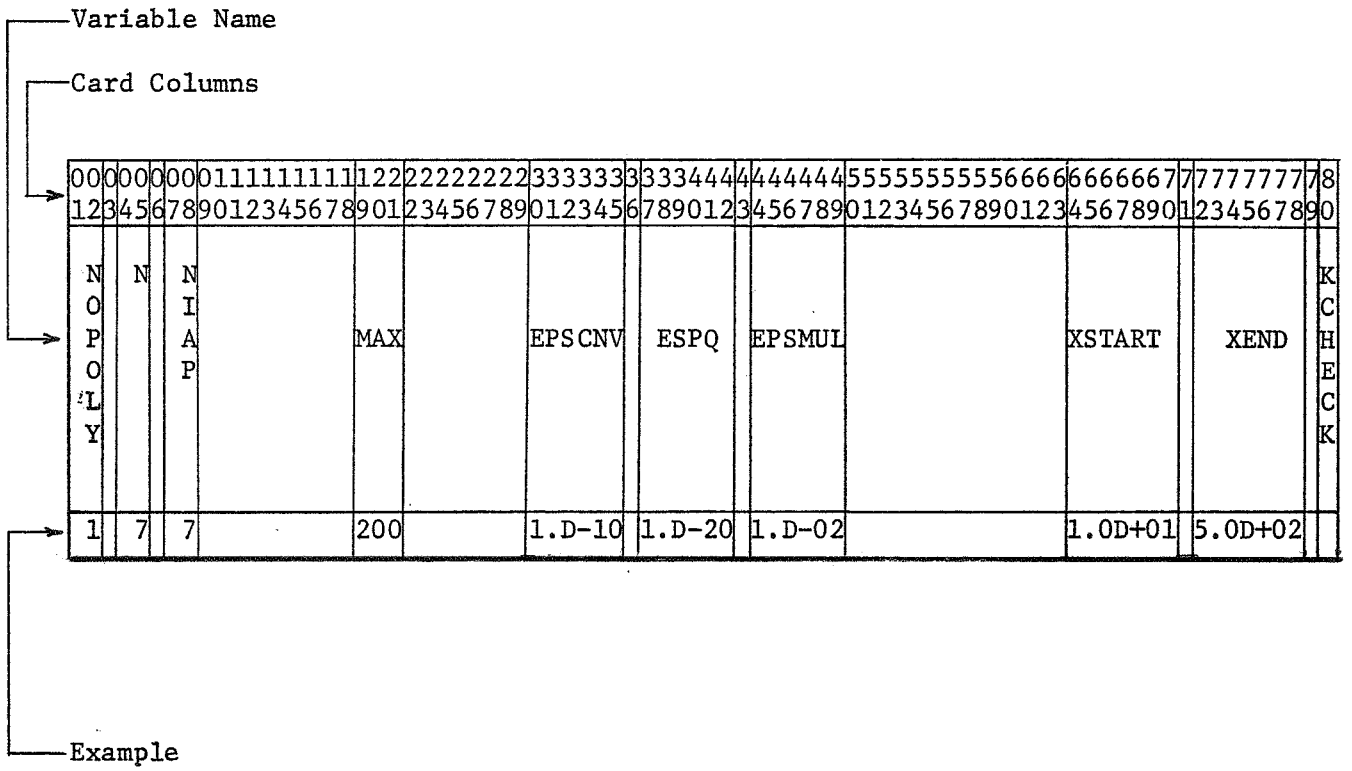


Figure B.2. Control Card for Newton's Method

00000000011111111112222222222333333333444444444455555555566666666677777777778 12345678901234567890123456789012345678901234567890123456789012345678901234567890		
A (RA)	A (VA)	
0.621735D+01	-0.132714D-02	

Figure B.3. Coefficient Card for Newton's Method

00000000011111111112222222223 12345678901234567890123456789	33333333344444444455555555556 12345678901234567890123456789	66666666677777777778 1234567890123456789
XZERO (RXZERO)	XZERO (VXZERO)	
0.15D+01	-0.25D-00	

Figure B.4. Initial Approximation Card for Newton's Method

000000000111111111122222222223333333333444444444455555555556666666666777777777 123456789012345678901234567890123456789012345678901234567890123456789012345678	8 9 0
	KCHECK 1

Figure B.5. End Card for Newton's Method

TABLE B. VII

VARIABLES USED IN NEWTÓN'S METHOD

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Main Program
NOPOLY	I	NOPOLY	I		Number of the polynomial
N	I	N	I		Degree of the polynomial
NIAP	I	NIAP	I		Number of initial approximations to be read
MAX	I	MAX	I		Maximum number of iterations to be performed
EPSCNV	R	EPSCNV	D		Tolerance check for convergence
EPSMUL	R	EPSMUL	D		Tolerance check for multiplicities
EPSQ	R	EPSQ	D		Tolerance check for zero in subroutine QUAD
XSTART	R	XSTART	D		Magnitude from which to begin the search for zeros
XEND	R	XEND	D		Magnitude to end the search for zeros
KCHECK	I	KCHECK	I		Program Control. When KCHECK = 1, program will terminate execution.
NA	I	NA	I		Number of coefficients or original polynomial
A	C	RA,VA	D		Array containing the coefficients of original polynomial P(X)
NDEF	I	NDEF	I		Degree of current deflated polynomial
L	I	L	I		Counter for number of initial approximations used
ITER	I	ITER	I		Counter for number of iterations
NROOT	I	NROOT	I		Counter for number of roots found (counting multiplicities)
IALTER	I	IALTER	I		Counter for number of alterations of each initial approximation
ITIME	I	ITIME	I		Program control
K	I	K	I		Counter for number of distinct roots found
ND	I	ND	I		Program control & number of coefficient of deflated polynomial for which no zeros were found

TABLE B. VII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
XO	C	RXO,VXO	D		Current approximation (X_n) to root
COEF	C	RCOEF,VCOEF	D		Working array containing coefficients of current deflated polynomial
DPX	C	RDPX,VDPX	D		Derivative of P(X) at some value X
PX	C	RPX,VPX	D		Value of P(X) at some point X
XZERO	C	RXZERO, VXZERO	D		Array containing the initial approximations
XNEW	C	RXNEW,VXNEW	D		New approximation (X_{n+1}) obtained from old approximation (X_n) by Newton's Algorithm
KANS	I	KANS	I		KANS = 1 implies convergence, KANS = 0 implies no convergence
MULT	I	MULT	I		Array containing the number of multiplicities of each root
X	C	RX,VX	D		Array containing the zeros of P(X)
XINIT	C	RXINIT, VXINIT	D		Array containing the initial or altered approximations which produced convergence to each root
NUM	I	NUM	I		Number of coefficients of current deflated polynomial
B	C	RB,VB	D		Array containing the coefficients of newly deflated polynomial
IROOT	I	IROOT	I		Number of distinct roots found by Newton's method, i.e. not solved for directly by subroutine QUAD
D	C	RD,VD	D		Array containing the coefficients of deflated polynomial for which no zeros were found
I01	I	I01	I		Unit number of input device
I02	I	I02	I		Unit number of output device
C	C	RC,VC	D		Array containing sequence of values leading to the derivative
EPSCHK	R	EPSCHK	D		Current tolerance for checking convergence or multiplicity

TABLE B. VII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Subroutine HORNER
A	C	RA,VA	D	E	Array of coefficients of polynomial
B	C	RB,VB	D	R	Array of coefficients of deflated polynomial
NDEF	I	NDEF	I	E	Degree of polynomial
NUM	I	NUM	I		Number of coefficients of polynomial
XO	C	RXO,VXO	D	E	Point (X_n) at which to evaluate the polynomial and its derivative. Also current approximation (X_{n+1}) used to deflate the polynomial
PX	C	RPX,VPX	D	R	Value of polynomial at X_n
DPX	C	RDPX,VDPX	D	R	Value of the derivative of polynomial at X_n
C	C	RC,VC	D	R	Array of containing sequence of values leading to the derivative
					Subroutine NEWTON
PX	C	RPX,VPX	D	E	Value of polynomial at X_n
DPX	C	RDPX,VDPX	D	E	Derivative of polynomial at X_n
XO	C	RXO,VXO	D	E	Current approximation (X_n) to root
XNEW	C	RXNEW,VXNEW	D	R	New approximation (X_{n+1}) to root
					Subroutine CHECK
EPSLON	R	EPS	D	C	Tolerance for convergence or multiplicity check
PX	C	RPX,VPX	D	E	Value of P(X) at X_n
DPX	C	RDPX,VDPX	D	E	Derivative of P(X) at X_n
XO	C	RXO,VXO	D	E	Current approximations (X_{n+1}) to root
IO2	I	IO2	I	C	Unit number of output device
KANS	I	KANS	I	R	KANS = 1 implies convergence, KANS = 0 implies no convergence

TABLE B. VII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Subroutine BETTER
I02	I	I02	I	C	Unit number of output device
XZERO	C	RXZERO, VXZERO	D	E	Array of approximations
X	C	RX,VX	D	ECR	Array of roots
A	C	RA,VA	D	E	Coefficients of original (undeflated) polynomial, P(X)
COEF	C	RCOEF,VCOEF	D	E	Working array for coefficients of polynomial
NA	I	NA	I	E	Number of coefficients of original polynomial
XO	C	RXO,VXO	D		Current approximation (X_n) to root
DPX	C	RDPX,VDPX	D		Derivative of P(X) at X_n
PX	C	RPX,VPX	D		Value of P(X) at X_n
KANS	I	KANS	I		KANS = 1 implies convergence; KANS = 0 implies no convergence
ITER	I	ITER	I		Counter for number of iterations
XNEW	C	RXNEW,VXNEW	D		New approximation (X_{n+1}) to root
NN	I	NN	I		Degree of polynomial
C	C	RC,VC	D	E	Array containing the sequence of values leading to the derivative
K	I	K	I	E	Number of distinct roots of P(X) found
N	I	N	I	E	Degree of polynomial P(X)
B	C	RB,VB	D	E	Array of coefficients of deflated polynomial
MAX	I	MAX	I	C	Maximum number of iterations permitted
EPSCHK	R	EPS	D	C	Tolerance for checking convergence
					Subroutine GENAPP
APP	C	APPR,APPI	D	R	Array containing initial approximations
NAPP	I	NAPP	I	E	Number of initial approximations to be generated

TABLE B. VII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
XSTART	R	XSTART	D	ECR	Magnitude at which to begin generating approximations; also magnitude of the approximation being generated
BETA	R	BETA	D		Argument of the complex approximation being generated
U	R	APPR(I)	D		Real part of complex approximation
V	R	APPI(I)	D		Imaginary part. of complex approximation
				Subroutine ALTER	
XOLD	C	XOLDR,XOLDI	D	ECR	Old approximation to be altered to new approximation
NALTER	I	NALTER	I	ECR	Number of alterations performed on an initial approximation
ITIME	I	ITIME	I	E	Program control
MAX	I	MAX	I	C	Maximum number of iterations permitted
Y	R	XOLDI	D		Imaginary part of original initial approximation (unaltered)
X	R	XOLDR	D		Real part of original unaltered initial approximation
R	R	R	D		Magnitude of original unaltered initial approximation
BETA	R	BETA	D		Argument of new approximation
XOLDR	R	XOLDR	D		Real part of new approximation
XOLDI	R	XOLDI	D		Imaginary part of new approximation
I02	I	I02	I	C	Unit number of output device
				Subroutine QUAD	
A	C	UA,VA	D	E	Coefficients of polynomial to be solved
NA	I	NA	I	E	Degree of polynomial
ROOT	C	UROOT,VROOT	D	ECR	Array of roots of P(X) (original polynomial)
NROOT	I	NROOT	I	ECR	Number of distinct roots of P(X) (the original polynomial)

TABLE B. VII (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
MULTI	I	MULTI	I	ECR	Array containing multiplicities of each root
EPST	R	EPST	D	E	Tolerance check for the number zero
DISC	C	UDISC,VDISC	D		Value of the discriminate ($b^2 - 4ac$) of Quadratic
				Subroutine COMSQT	
		UX,VX	D	E	Complex number for which the square root is desired
		UY,VY	D	R	Square root of the complex number

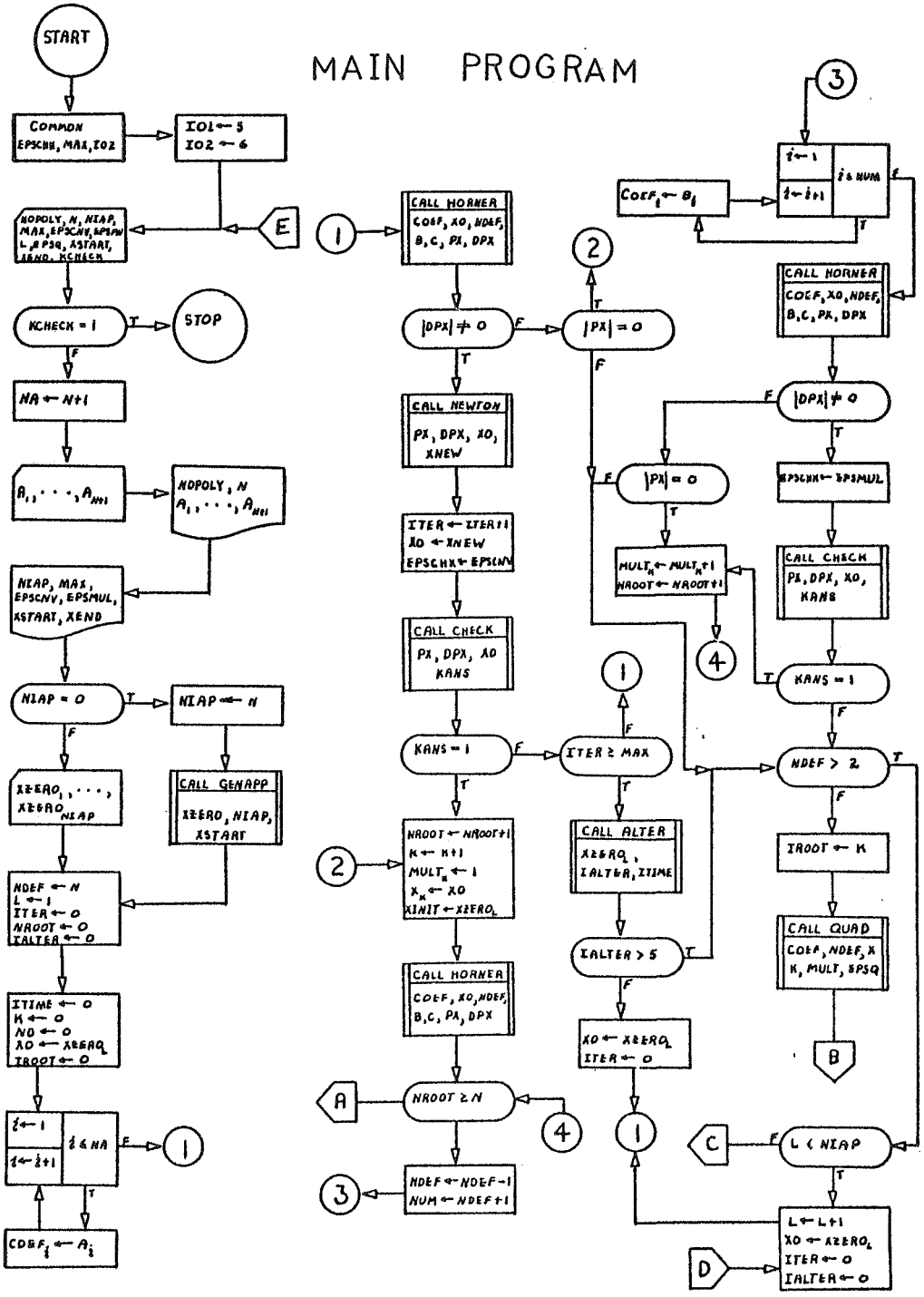


Figure B.6. Flow Charts for Newton's Method

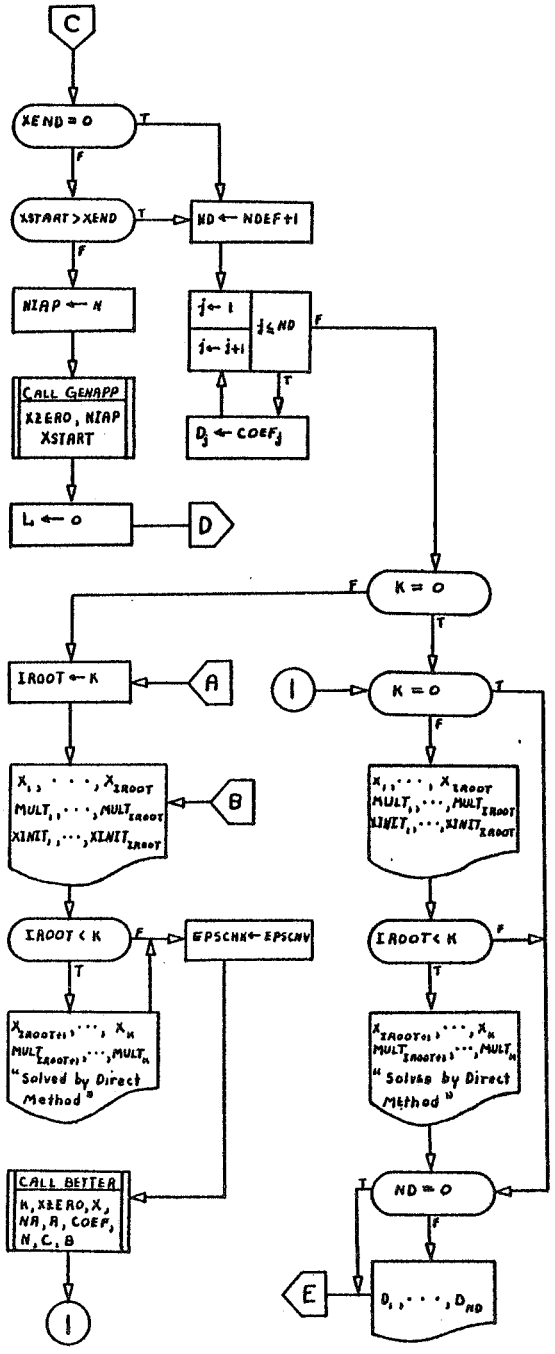


Figure B.6. (Continued)

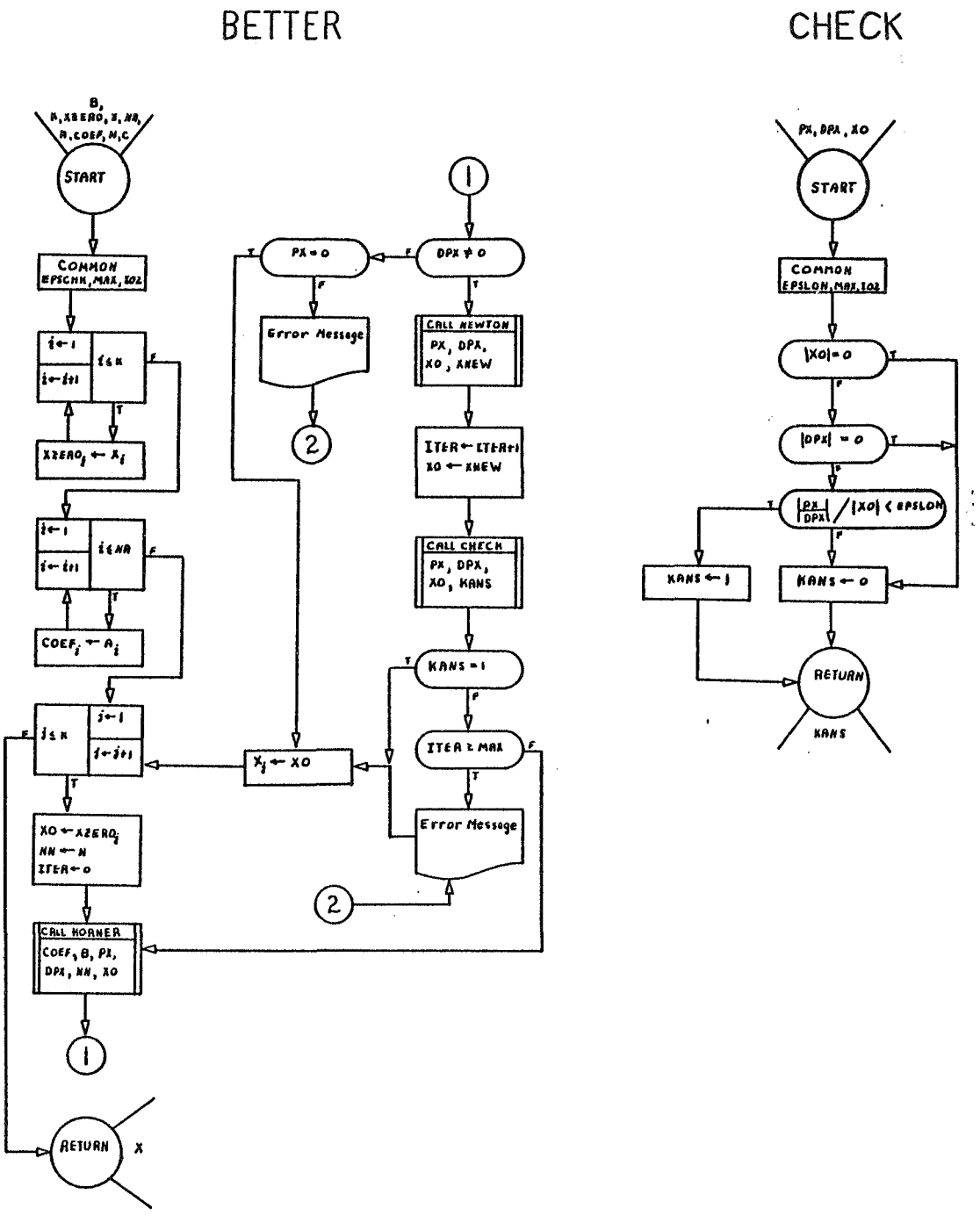


Figure B.6. (Continued)

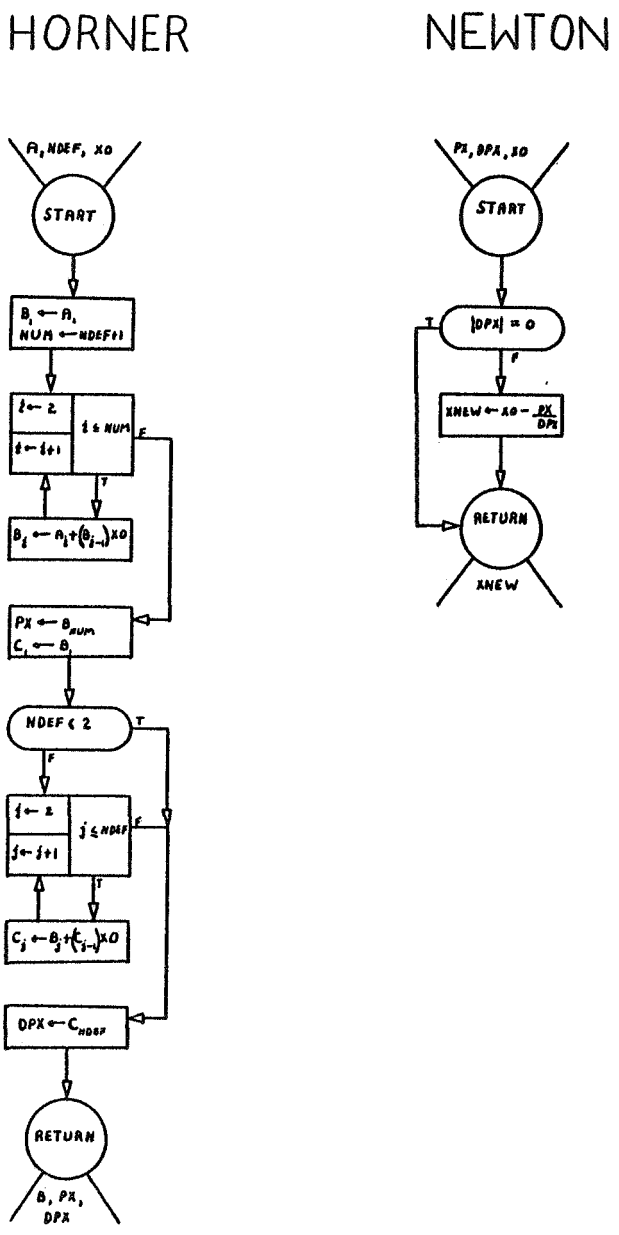


Figure B.6. (Continued)

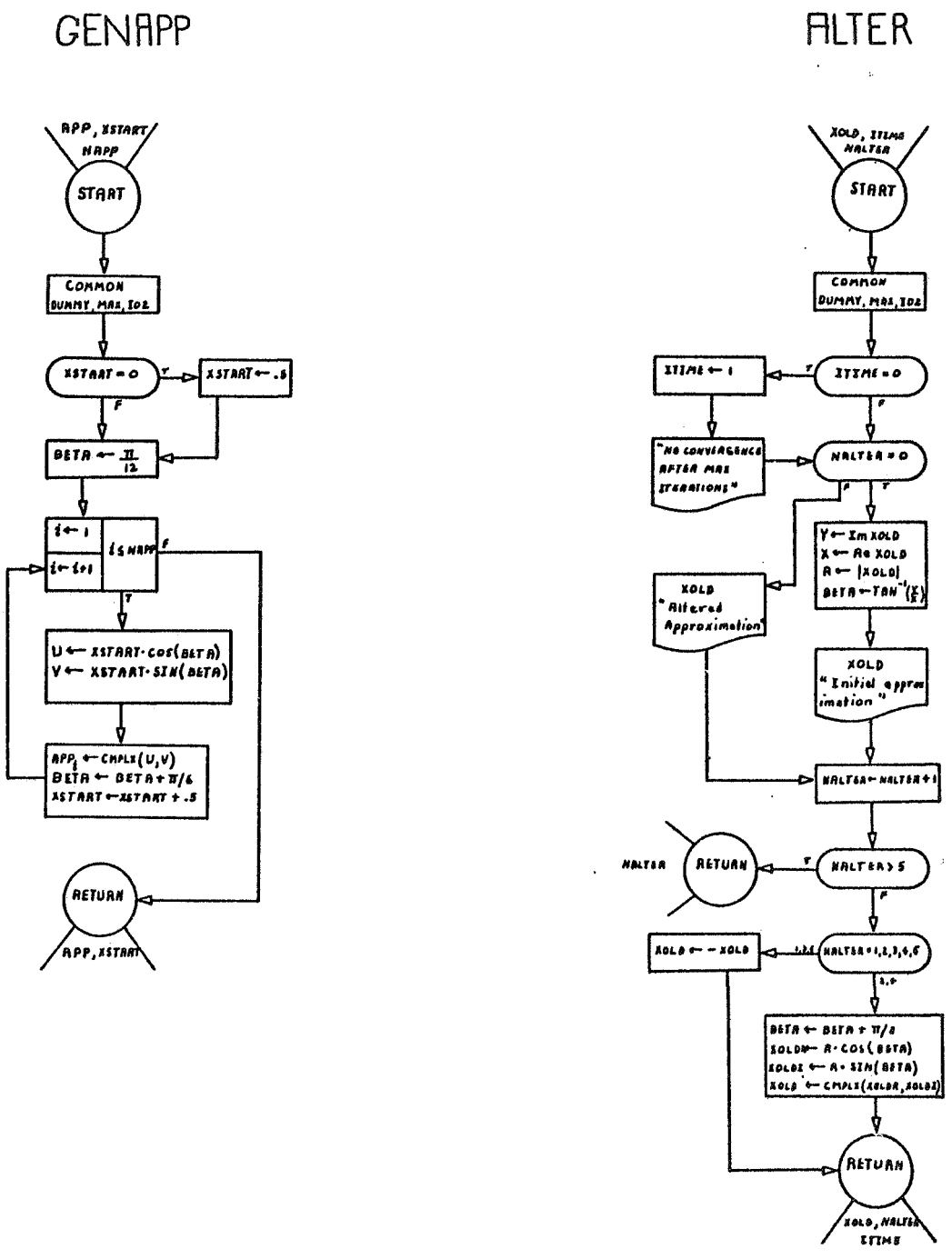


Figure B.6. (Continued)

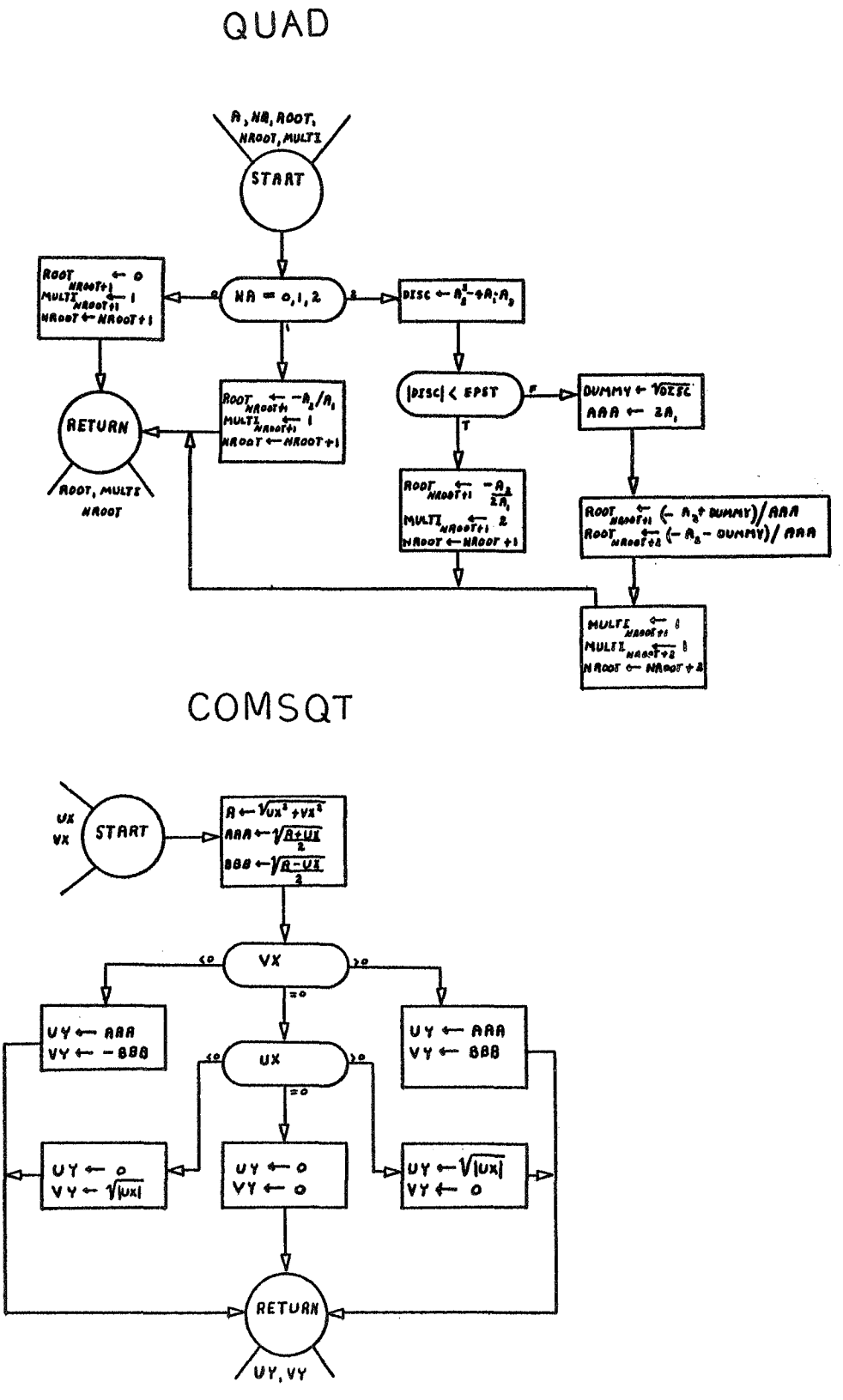


Figure B.6. (Continued)

TABLE B. VIII

PROGRAM FOR NEWTON'S METHOD

```
C   ************************************************************************
C   *                                                                      *
C   * DOUBLE PRECISION PROGRAM FOR NEWTON'S METHOD                         *
C   *                                                                      *
C   *                                                                      *
C   * NEWTONS METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A      *
C   * POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-   *
C   * IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION   *
C   * FORMULA                                                              *
C   *              X(N+1) = X(N)-P(X(N))/P'(X(N)).                         *
C   *                                                                      *
C   ************************************************************************
0001        DOUBLE PRECISION RA,VA,RXZERO,VXZERO,RB,VB,RCOEF,VCOEF,RX,VX,RXINI
           1T,VXINIT,RC,VC,RD,VD,RPX,VPX,RDPX,VDPX,RXNEW,VXNEW,RXO,VXO,EPSCHK,
           2EPSCNV,EPSQ,EPSMUL,XSTART,XEND,ABPX,ABDPX
0002        DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26),RD(26),VD(26),
           1RCOEF(26),VCOEF(26),MULT(25),RXZERO(25),VXZERO(25),RX(25),VX(25),R
           2XINIT(25),VXINIT(25)
0003        COMMON EPSCHK,MAX,IO2
0004        IO1=5                                                          112
0005        IO2=6                                                          116
0006      1 READ(IO1,1000) NOPOLY,N,NIAP,MAX,EPSCNV,EPSQ,EPSMUL,XSTART,XEND,KC
           1HECK
0007        IF(KCHECK.EQ.1) STOP
0008        NA=N+1                                                         130
0009        READ(IO1,1010) (RA(I),VA(I),I=1,NA)                            132
0010        WRITE(IO2,1030) NOPOLY,N
0011        WRITE(IO2,1040) (I,RA(I),VA(I),I=1,NA)
0012        WRITE(IO2,2060)
0013        WRITE(IO2,2000) NIAP
0014        WRITE(IO2,2010) MAX
0015        WRITE(IO2,2020) EPSCNV
0016        WRITE(IO2,2030) EPSMUL
0017        WRITE(IO2,2040) XSTART
0018        WRITE(IO2,2050) XEND
0019        IF(NIAP.NE.0) GO TO 3
0020        NIAP=N
0021        CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
0022        GO TO 4
0023      3 READ(IO1,1020) (RXZERO(I),VXZERO(I),I=1,NIAP)
0024      4 NDEF=N
0025        L=1                                                            152
0026        ITER=0                                                         154
0027        NROOT=0                                                        160
0028        IROOT=0
0029        ITIME=0
0030        ND=0
0031        IALTER=0                                                       164
0032        K=0                                                            168
0033        RXO=RXZERO(L)                                                  180
0034        VXO=VXZERO(L)                                                  181
0035        DO 5 I=1,NA                                                    188
0036        RCOEF(I)=RA(I)                                                 190
0037      5 VCOEF(I)=VA(I)                                                 191
0038     10 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
           1)
0039        ABPX=DSQRT(RPX*RPX+VPX*VPX)
0040        ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
```

TABLE B. VIII (Continued)

```
0041            IF(ABDPX.NE.0.0) GO TO 20
0042            IF(ABPX.EQ.0.0) GO TO 70
0043            GO TO 110
0044         20 CALL NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)
0045            ITER=ITER+1                                                  200
0046            RXO=RXNEW                                                    204
0047            VXO=VXNEW                                                    205
0048            EPSCHK=EPSCNV
0049            CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
0050            IF(KANS.EQ.1) GO TO 70                                       194
0051            IF(ITER.GE.MAX) GO TO 40
0052            GO TO 10                                                     208
0053         40 CALL ALTER(RXZERO(L),VXZERO(L),IALTER,ITIME)
0054            IF(IALTER.GT.5) GO TO 110
0055            RXO=RXZERO(L)
0056            VXO=VXZERO(L)
0057            ITER=0                                                       244
0058            GO TO 10                                                     248
0059         60 ND=NDEF+1
0060            DO 65 J=1,ND
0061            RD(J)=RCOEF(J)
0062         65 VD(J)=VCOEF(J)
0063            GO TO 140
0064         70 NROOT=NROOT+1                                                268
0065            K=K+1                                                        272
0066            MULT(K)=1                                                    276
0067            RX(K)=RXO                                                    280
0068            VX(K)=VXO                                                    281
0069            RXINIT(K)=RXZERO(L)                                          288
0070            VXINIT(K)=VXZERO(L)                                          289
0071            CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
               1)
0072         80 IF(NROOT.GE.N) GO TO 147
0073            NDEF=NDEF-1
0074            NUM=NDEF+1
0075            DO 105 I=1,NUM                                               294
0076            RCOEF(I)=RB(I)                                               296
0077        105 VCOEF(I)=VB(I)                                               297
0078            CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
               1)
0079            ABPX=DSQRT(RPX*RPX+VPX*VPX)
0080            ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
0081            IF(ABDPX.NE.0.0) GO TO 107
0082            IF(ABPX.EQ.0.0) GO TO 130
0083            GO TO 110
0084        107 CONTINUE
0085            EPSCHK=EPSMUL
0086            CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
0087            IF(KANS.EQ.1) GO TO 130                                      300
0088        110 IF(NDEF.GT.2) GO TO 113
0089            IROOT=K
0090            CALL QUAD(RCOEF,VCOEF,NDEF,RX,VX,K,MULT,EPSQ)
0091            GO TO 150
0092        113 IF(L.LT.NIAP) GO TO 115
0093            IF(XEND.EQ.0.0) GO TO 60
0094            IF(XSTART.GT.XEND) GO TO 60
0095            NIAP=N
0096            CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
```

TABLE B. VIII (Continued)

```
0097            L=0
0098        115 L=L+1
0099            RXO=RXZERO(L)                                                  312
0100            VXO=VXZERO(L)                                                  313
0101            ITER=0                                                         316
0102            IALTER=0                                                       320
0103            GO TO 10                                                       324
0104        130 MULT(K)=MULT(K)+1                                              328
0105            NROOT=NROOT+1                                                  332
0106            GO TO 80                                                       336
0107        140 IF(K.EQ.0) GO TO 160                                           338
0108        147 IROOT=K
0109        150 WRITE(IO2,1025)
0110            WRITE(IO2,1050)
0111            WRITE(IO2,1060) (I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,IRO
               1OT)
0112            KKK=IROOT+1
0113            IF(IROOT.LT.K) WRITE(IO2,1062) (I,RX(I),VX(I),MULT(I),I=KKK,K)
0114            EPSCHK=EPSCNV
0115            CALL BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCOEF,N,RC,VC,RB,
               1VB)
0116        160 IF(K.EQ.0) GO TO 170
0117            WRITE(IO2,1065)
0118            WRITE(IO2,1050)
0119            WRITE(IO2,1060) (I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,IRO
               1OT)
0120            KKK=IROOT+1
0121            IF(IROOT.LT.K) WRITE(IO2,1062) (I,RX(I),VX(I),MULT(I),I=KKK,K)
0122        170 IF(ND.EQ.0) GO TO 1
0123            WRITE(IO2,1070)                                                396
0124            WRITE(IO2,1075) (J,RD(J),VD(J),J=1,ND)
0125            GO TO 1                                                        444
0126       1000 FORMAT(3(I2,1X),9X,I3,8X,3(D6.0,1X),13X,2(D7.0,1X),I1)
0127       1010 FORMAT(2D30.0)
0128       1030 FORMAT(1H1,8X,43HNEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS/9X,18
               1HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2,////1X,28HTHE COEFFICIENT
               2S OF P(X) ARE/)
0129       1040 FORMAT(3X,2HP(,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
0130       1020 FORMAT(2D30.0)
0131       1025 FORMAT(///1X,61HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
               1OF P(X) ARE)
0132       1050 FORMAT(///2X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
               1 APPROXIMATION//)
0133       1060 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,7X,D23.
               116,3H + ,D23.16,2H I)
0134       1062 FORMAT(3X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,8X,23HS
               1OLVED BY DIRECT METHOD)
0135       1065 FORMAT(///1X,61HAFTER  THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
               1OF P(X) ARE)
0136       1070 FORMAT(///1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO Z
               1EROS WERE FOUND/)
0137       1075 FORMAT(3X,2HD(,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
0138       2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
0139       2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
0140       2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
0141       2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
0142       2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
0143       2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
0144       2060 FORMAT(//1X)
0145            END                                                            450
```

TABLE B. VIII (Continued)

```
0001            SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
                C  ***************************************************************************
                C  *                                                                         *
                C  * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE     *
                C  * DEGREE OF THE ORIGINAL POLYNOMIAL.                                      *
                C  *                                                                         *
                C  ***************************************************************************
0002            DOUBLE PRECISION APPR,APPI,XSTART,DUMMY,BETA
0003            DIMENSION APPR(25),APPI(25)
0004            COMMON DUMMY,MAX,IO2
0005            IF(XSTART.EQ.0.0) XSTART=0.5
0006            BETA=0.2617994
0007            DO 10 I=1,NAPP
0008            APPR(I)=XSTART*DCOS(BETA)
0009            APPI(I)=XSTART*DSIN(BETA)
0010            BETA=BETA+0.5235988
0011          10 XSTART=XSTART+0.5
0012            RETURN
0013            END



0001            SUBROUTINE ALTER(XOLDR,XOLDI,NALTER,ITIME)
                C  ***************************************************************************
                C  *                                                                         *
                C  * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO      *
                C  * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
                C  *                                                                         *
                C  ***************************************************************************
0002            DOUBLE PRECISION XOLDR,XOLDI,DUMMY,ABXOLD,BETA
0003            COMMON DUMMY,MAX,IO2
0004            IF(ITIME.NE.0) GO TO 5
0005            ITIME =1
0006            WRITE(IO2,1010) MAX
0007          5 IF(NALTER.EQ.0) GO TO 10
0008            WRITE(IO2,1000) XOLDR,XOLDI
0009            GO TO 20
0010         10 ABXOLD=DSQRT(XOLDR*XOLDR+XOLDI*XOLDI)
0011            BETA=DATAN2(XOLDI,XOLDR)
0012            WRITE(IO2,1020) XOLDR,XOLDI
0013         20 NALTER=NALTER+1
0014            IF(NALTER.GT.5) RETURN
0015            GO TO (30,40,30,40,30),NALTER
0016         30 XOLDR=-XOLDR
0017            XOLDI=-XOLDI
0018            GO TO 50
0019         40 BETA=BETA+1.0471976
0020            XOLDR=ABXOLD*DCOS(BETA)
0021            XOLDI=ABXOLD*DSIN(BETA)
0022         50 RETURN
0023       1000 FORMAT(1X,D23.16,3H + ,D23.16,2H I,10X,21HALTERED APPROXIMATION)
0024       1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
                  ITER ,I3,12H ITERATIONS.//)
0025       1020 FORMAT(/1X,D23.16,3H + ,D23.16,2H I,10X,21HINITIAL APPROXIMATION)
0026            END
```

TABLE B. VIII (Continued)

```
0001            SUBROUTINE QUAD(UA,VA,NA,UROOT,VROOT,NROOT,MULTI,EPST)
                C   ***************************************************************************
                C   *                                                                         *
                C   * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES  *
                C   * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE   *
                C   * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                          *
                C   *                                                                         *
                C   ***************************************************************************
0002            DOUBLE PRECISION UA,VA,UROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
                1Y,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
0003            DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULTI(25)
0004            IF(NA.EQ.2) GO TO 7
0005            IF(NA.EQ.1) GO TO 5
0006            UROOT(NROOT+1)=0.0
0007            VROOT(NROOT+1)=0.0
0008            MULTI(NROOT+1)=1
0009            NROOT=NROOT+1
0010            GO TO 50
0011          5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
0012            UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
0013            VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
0014            MULTI(NROOT+1)=1
0015            NROOT=NROOT+1
0016            GO TO 50
0017          7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(1)*UA(3)-VA(1)*VA(3)))
0018            VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-(4.0*(VA(1)*UA(3)+UA(1)*VA(3)))
0019            BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
0020            IF(BBB.LT.EPST) GO TO 10
0021            CALL COMSQT(UDISC,VDISC,UDUMMY,VDUMMY)
0022            UBBB=-UA(2)+UDUMMY
0023            VBBB=-VA(2)+VDUMMY
0024            RDUMMY=-UA(2)-UDUMMY
0025            SDUMMY=-VA(2)-VDUMMY
0026            UAAA=2.0*UA(1)
0027            VAAA=2.0*VA(1)
0028            BBB=UAAA*UAAA+VAAA*VAAA
0029            UROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
0030            VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
0031            UROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
0032            VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
0033            MULTI(NROOT+1)=1
0034            MULTI(NROOT+2)=1
0035            NROOT=NROOT+2
0036            GO TO 50
0037         10 UAAA=2.0*UA(1)
0038            VAAA=2.0*VA(1)
0039            BBB=UAAA*UAAA+VAAA*VAAA
0040            UROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
0041            VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB
0042            MULTI(NROOT+1)=2
0043            NROOT=NROOT+1
0044         50 RETURN
0045            END
```

TABLE B. VIII (Continued)

```
0001            SUBROUTINE COMSQT(UX,VX,UY,VY)
                C  ***************************************************************************
                C  *                                                                         *
                C  * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.           *
                C  *                                                                         *
                C  ***************************************************************************
0002            DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
0003            R=DSQRT(UX*UX+VX*VX)
0004            AAA=DSQRT(DABS((R+UX)/2.0))
0005            BBB=DSQRT(DABS((R-UX)/2.0))
0006            IF(VX) 10,20,30
0007          10 UY=AAA
0008            VY=-1.0*BBB
0009            GO TO 100
0010          20 IF(UX) 40,50,60
0011          30 UY=AAA
0012            VY=BBB
0013            GO TO 100
0014          40 DUMMY=DABS(UX)
0015            UY=0.0
0016            VY=DSQRT(DUMMY)
0017            GO TO 100
0018          50 UY=0.0
0019            VY=0.0
0020            GO TO 100
0021          60 DUMMY=DABS(UX)
0022            UY=DSQRT(DUMMY)
0023            VY=0.0
0024         100 RETURN
0025            END
```

TABLE B. VIII (Continued)

```
0001            SUBROUTINE HORNER(RA,VA,RXO,VXO,NDEF,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX
              C ********************************************************************
              C *                                                                  *
              C * HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL P(X) AT A POINT D AND *
              C * ITS DERIVATIVE AT D.  SYNTHETIC DIVISION IS USED TO DEFLATE THE  *
              C * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).                     *
              C *                                                                  *
              C ********************************************************************
               1)
0002            DOUBLE PRECISION VDPX,RXO,VXO,RB,VB,RC,VC,RPX,VPX,RDPX,RA,VA
0003            DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26)
0004            RB(1)=RA(1)                                                    516
0005            VB(1)=VA(1)                                                    517
0006            NUM=NDEF+1                                                     520
0007            DO 10 I=2,NUM                                                  524
0008            RB(I)=RA(I)+(RB(I-1)*RXO-VB(I-1)*VXO)
0009         10 VB(I)=VA(I)+(VB(I-1)*RXO+RB(I-1)*VXO)
0010            RPX=RB(NUM)                                                    532
0011            VPX=VB(NUM)                                                    533
0012            RC(1)=RB(1)                                                    540
0013            VC(1)=VB(1)                                                    541
0014            IF(NDEF.LT.2) GO TO 25
0015            DO 20 J=2,NDEF                                                 544
0016            RC(J)=RB(J)+(RC(J-1)*RXO-VC(J-1)*VXO)
0017         20 VC(J)=VB(J)+(VC(J-1)*RXO+RC(J-1)*VXO)
0018         25 RDPX=RC(NDEF)
0019            VDPX=VC(NDEF)                                                  553
0020            RETURN                                                         572
0021            END                                                            580
```

```
0001            SUBROUTINE NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)       600
              C ********************************************************************
              C *                                                                  *
              C * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX- *
              C * IMATION BY USING THE ITERATION FORMULA                           *
              C *              X(N+1) = X(N)-P(X(N))/P'(X(N)).                     *
              C *                                                                  *
              C ********************************************************************
0002            DOUBLE PRECISION RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW,ARG
0003            DOUBLE PRECISION DDD
0004            ARG=RDPX*RDPX+VDPX*VDPX
0005            DDD=DSQRT(ARG)
0006            IF(DDD.EQ.0.0) RETURN
0007            RXNEW=RXO-((RPX*RDPX+VPX*VDPX)/ARG)
0008            VXNEW=VXO-((VPX*RDPX-RPX*VDPX)/ARG)
0009            RETURN                                                         616
0010            END                                                            620
```

TABLE B. VIII (Continued)

```
0001            SUBROUTINE CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
              C ********************************************************************
              C *                                                                  *
              C * THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX- *
              C * IMATIONS BY TESTING THE EXPRESSION                               *
              C * ABSOLUTE VALUE OF (P(X(N))/P'(X(N)))/ABSOLUTE VALUE OF X(N+1).   *
              C * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.        *
              C *                                                                  *
              C ********************************************************************
0002            DOUBLE PRECISION RPX,VPX,RDPX,VDPX,RXO,VXO,ABSXO,ABSQUO,RDUMMY,VDU      749
               1MMY,EPS                                                                 750
0003            DOUBLE PRECISION ARG
0004            DOUBLE PRECISION DDD
0005            COMMON EPS,MAX,IO2
0006            ABSXO=DSQRT(RXO*RXO+VXO*VXO)
0007            IF(ABSXO.EQ.0.) GO TO 25
0008            ARG=RDPX*RDPX+VDPX*VDPX
0009            DDD=DSQRT(ARG)
0010            IF(DDD.EQ.0.0) GO TO 25
0011            RDUMMY=(RPX*RDPX+VPX*VDPX)/ARG
0012            VDUMMY=(VPX*RDPX-RPX*VDPX)/ARG
0013            ABSQUO=DSQRT(RDUMMY*RDUMMY+VDUMMY*VDUMMY)
0014            IF(ABSQUO/ABSXO.LT.EPS) GO TO 10
0015            KANS=0                                                                   760
0016            RETURN                                                                   764
0017         10 KANS=1                                                                   768
0018            RETURN                                                                   772
0019         25 KANS=0
0020            RETURN
0021            END                                                                      780
```

TABLE B. VIII (Continued)

```
0001              SUBROUTINE BETTER(K,RXZERO,VXZERO,RX,VX,NA,RA,VA,RCOEF,VCOEF,N,RC,
                 1VC,RB,VB)
                C  ***************************************************************************
                C  *                                                                         *
                C  * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND   *
                C  * BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD APPLIED TO *
                C  * THE FULL, UNDEFLATED POLYNOMIAL.                                       *
                C  *                                                                         *
                C  ***************************************************************************
0002              DOUBLE PRECISION RXZERO,VXZERO,RX,VX,RA,VA,RCOEF,VCOEF,RC,VC,RB,VB      805
                 1,RXO,VXO,RPX,VPX,RDPX,VDPX,RXNEW,VXNEW,EPS
0003              DIMENSION RXZERO(25),VXZERO(25),RX(25),VX(25),RA(26),VA(26),RCOEF(
                 126),VCOEF(26),RC(26),VC(26),RB(26),VB(26)
0004              DOUBLE PRECISION ABPX,ABDPX
0005              COMMON EPS,MAX,IO2
0006              DO 10 I=1,K                                                             812
0007              RXZERO(I)=RX(I)                                                         815
0008            10 VXZERO(I)=VX(I)                                                        816
0009              DO 20 I=1,NA
0010              RCOEF(I)=RA(I)                                                          824
0011            20 VCOEF(I)=VA(I)                                                         825
0012              DO 50 J=1,K                                                             828
0013              RXO=RXZERO(J)                                                           832
0014              VXO=VXZERO(J)                                                           833
0015              NN=N                                                                    834
0016              ITER=0                                                                  836
0017            30 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NN,RB,VB,RC,VC,RPX,VPX,RDPX,VDPX)
0018              ABPX=DSQRT(RPX*RPX+VPX*VPX)
0019              ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
0020              IF(ABDPX.NE.0.0) GO TO 33
0021              IF(ABPX.EQ.0.0) GO TO 40
0022              GO TO 34
0023            33 CALL NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)
0024              ITER=ITER+1                                                             856
0025              RXO=RXNEW                                                               860
0026              VXO=VXNEW                                                               861
0027              CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
0028              IF(KANS.EQ.1) GO TO 40                                                  844
0029              IF(ITER.GE.MAX) GO TO 35
0030              GO TO 30                                                                864
0031            34 WRITE(IO2,1112) RXO,VXO
0032            35 WRITE(IO2,100) J,RXZERO(J),VXZERO(J)
0033              WRITE(IO2,200) MAX
0034            40 RX(J)=RXO                                                              870
0035              VX(J)=VXO                                                               871
0036            50 CONTINUE                                                               872
0037              RETURN                                                                  876
0038          1112 FORMAT(1HO,36HTHE VALUE OF THE DERIVATIVE AT XO = ,D23.16,3H + ,D2
                 13.16,2H I,10H  IS ZERO.)
0039           100 FORMAT(42HOIN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,D23
                 1.16,3H + ,D23.16,2H I,18H DID NOT CONVERGE.)
0040           200 FORMAT(33H THE PRESENT APPROXIMATION AFTER ,I3,29H ITERATIONS IS P
                 1RINTED BELOW.)
0041              END                                                                     880
```

APPENDIX C

MULLER'S METHOD

2. Use of the Program

A double precision FORTRAN IV program using Muller's method is presented in this appendix. Flow charts for this program are given in Figure C.1 while Table C.V gives a FORTRAN IV listing of this program. Single precision variables are listed in some of the tables. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from the appropriate tables.

The program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree N, where N > 25, certain array dimensions must be changed. These are listed in Table C.I for the main program and subprograms in double precision.

TABLE C.I

PROGRAM CHANGES FOR SOLVING POLYNOMIALS OF
DEGREE GREATER THAN 25
BY MULLER'S METHOD

Double Precision

Main Program

UROOT(N),VROOT(N)
MULT(N)
UAPP(N,3),VAPP(N,3)
UWORK(N+1),VWORK(N+1)
UB(N+1),VB(N+1)
UA(N+1),VA(N+1)
URAPP(N,3),VRAPP(N,3)

Subroutine BETTER

UROOT(N),VROOT(N)
UA(N+1),VA(N+1)
UBAPP(N,3),VBAPP(N,3)
UB(N+1),VB(N+1)
UROOTS(N),VROOTS(N)
URAPP(N,3),VRAPP(N,3)
MULT(N)

Subroutine GENAPP

APPR(N,3),APPI(N,3)

Subroutine HORNER

UA(N+1),VA(N+1)
UB(N+1),VB(N+1)

Subroutine QUAD

UA(N+1),VA(N+1)
UROOT(N),VROOT(N)
MULTI(N)

Table C.II lists the system functions used in the program of Muller's method. In the table "d" denotes a double precision variable name.

TABLE C.II

SYSTEM FUNCTIONS USED IN MULLER'S METHOD

Double Precision

DABS(d)	- obtain absolute value
DATAN2(d_1,d_2)	- arctangent of d_1/d_2
DSQRT(d)	- square root
DCOS(d)	- cosine of angle
DSIN(d)	- sine of angle
DSQRT(d)	- square root

2. Input Data for Muller's Method

The input data for Muller's method is identical to the input data for Newton's method as described in Appendix B, § 2 except for the variable names. The correspondence of input variable names is given in Table C.III. Only one (not three) initial approximation, X_0, is given for each root. The other two required by Muller's method are constructed within the program and are $.9X_0$ and $1.1X_0$.

3. Variables Used in Muller's Method

The definitions of the major variables used in Muller's method are given in Table C.IV. For definitions of variables not listed in this table see the definitions of variables for the corresponding subroutine in Table B.VII. The notation and symbols used here are the same as for Table B.VII and are described in Appendix B, § 3.

TABLE C.III

CORRESPONDENCE OF NEWTON'S AND MULLER'S
INPUT DATA VARIABLES

Newton's Method	Muller's Method
Control Card	
NOPOLY	NOPOLY
N	NP
NIAP	NAPP
MAX	MAX
EPSCNV	EPS
EPSQ	EPSQ
EPSMUL	EPSM
XSTART	XSTART
XEND	XEND
KCHECK	KCHECK
Coefficient Card	
A (RA)	A (UA)
A (VA)	A (VA)
Initial Approximation Card	
XZERO (RXZERO)	APP (UAPP)
XZERO (VXZERO)	APP (VAPP)
End Card	
KCHECK	KCHECK

4. Description of Program Output

The output from Muller's method is the same as that for Newton's method as described in Apptendix B, § 4. Only one initial approximation, Z, (not three) is printed for each root. It is either that supplied by the user or generated by the program. The other two approximations used were 0.9Z and 1.1Z.

5. Informative and Error Messages

The output may contain informative messages printed as an aid to the user. These are:

"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX."

XX is the number of the polynomial. This message is printed if no roots of the polynomial were found.

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(X) = YYY
DID NOT CONVERGE AFTER ZZZ ITERATIONS
THE PRESENT APPROXIMATION IS AAA"

X is the number of the root before the attempt to improve accuracy, YYY is the value of the root before attempt to improve accuracy, ZZZ is the maximum number of iterations, and AAA is the current approximation after the maximum number of iterations. This message has the same meaning as the corresponding message in Appendix B, § 5.

TABLE C. IV

VARIABLES USED IN MULLER'S METHOD

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Main Program
NP	I	NP	I		Degree of polynomial P(X)
NROOT	I	NROOT	I		Number of distinct roots found
NOMULT	I	NOMULT	I		Number of roots (counting multiplicities)
ROOT	C	UROOT,VROOT	D		Array containing the roots
NAPP	I	NAPP	I		Number of initial approximations to be read in
APP	C	UAPP,VAPP	D		Array of initial approximations
WORK	C	UWORK,VWORK	D		Working array containing coefficients of current polynomial
B	C	UB,VB	D		Array containing coefficients of deflated polynomial
A	C	UA,VA	D		Array containing coefficients of original polynomial, P(X)
RAPP	C	URAPP,VRAPP	D		Array of initial or altered approximations for which convergence was obtained
X1	C	UX1,VX1	D		One of three current approximations to a root
X2	C	UX2,VX2	D		One of three current approximations to a root
X3	C	UX3,VX3	D		One of three current approximations to a root
PX1	C	UPX1,VPX1	D		Value of polynomial P(X) at X1
PX2	C	UPX2,VPX2	D		Value of polynomial P(X) at X2
PX3	C	UPX3,VPX3	D		Value of polynomial P(X) at X3
X4	C	UX4,VX4	D		Newest approximation (X_{n+1}) to root
PX4	C	UPX4,VPX4	D		Value of polynomial P(X) at X4
MULT	I	MULT	I		Array containing the multiplicities of each root found
ITER	I	ITER	I		Counter for iterations
I01	I	I01	I		Unit number of input device
I02	I	I02	I		Unit number of output device

TABLE C.IV. (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
EPSRT	R	EPSRT	D		Number used in subroutine BETTER to generate two approximations from the one given
NOPOLY	I	NOPOLY	I		Number of the polynomial
MAX	I	MAX	I		Maximum number of iterations
EPS	R	EPS	D		Tolerance check for convergence
EPSO	R	EPSO	D		Tolerance check for zero (0)
EPSM	R	EPSM	D		Tolerance check for multiplicities
KCHECK	I	KCHECK	I		Program control, KCHECK = 1 stops execution of program
XSTART	R	XSTART	D		Magnitude at which to start generating initial approximations
XEND	R	XEND	D		Magnitude at which to end generating initial approximations
NWORK	I	NWORK	I		Degree of current deflated polynomial whose coefficients are in WORK
ITIME	I	ITIME	I		Program control
NALTER	I	NALTER	I		Number of alterations which have been performed on an initial approximation
IAPP	I	IAPP	I		Counter for number of initial approximations used
CONV	L	CONV	L		When CONV is true, convergence has been obtained
IROOT	I	IROOT	I		Number of distinct roots solved by Muller's method, i.e. not solved directly by subroutine QUAD
				Subroutine HORNER	
A	C	UA,VA	D	E	Array of current polynomial coefficients (to be deflated or evaluated)
NA	I	NA	I	E	Degree of polynomial to be deflated or evaluated
X	C	UX,VX	D	E	Approximation at which to evaluate or deflate the polynomial

TABLE C. IV (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
B	C	UB,VB	D	R	Array containing the coefficients of the deflated polynomial
PX	C	UPX,VPX	D	R	Value of the polynomial at X
NUM	I	NUM	I		Number of coefficients of polynomial to be deflated
				Subroutine TEST	
X3	C	UX3,VX3	D	E	Approximation to Root (old) (X_n)
X4	C	UX4,VX4	D	E	New approximation to root (X_{n+1})
CONV	L	CONV	L	R	CONV = true implies convergence has been obtained
EPS	R	EPS	D	C	Tolerance for convergence test
EPSO	R	EPSO	D	C	Tolerance check for zero (0)
DENOM	R	DENOM	D		Magnitude of new approximation, (X_{n+1})
				Subroutine BETTER	
MULT	I	MULT	I	ECR	Array of multiplicities of each root
A	C	UA,VA	D	E	Array of coefficients of original undeflated polynomial
NP	I	NP	I	E	Degree of original polynomial
ROOT	C	UROOT,VROOT	D	ECR	Array of roots
NROOT	I	NROOT	I	ECR	Number of roots stored in root
BAPP	C	UBAPP,VBAPP	D	E	Array of initial approximations (old roots)
IROOT	I	IROOT	I	ECR	Number of roots solved by the iterative process (Not QUAD)
ROOTS	C	UROOTS,VROOTS	D		Temporary storage for new (better) roots
L	I	L	I		Number of roots found by BETTER
EPSRT	R	EPSRT	D	C	A small number used to generate two of the three approximations when given one
ITER	I	ITER	I	C	Counter for number of iterations

TABLE C.IV (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
B	C	UB,VB	D		Array containing coefficients of deflated polynomial
X1	C	UX1,VX1	D		One of three approximations to the root
X2	C	UX2,VX2	D		One of three approximations to the root
X3	C	UX3,VX3	D		One of three approximations to the root
PX1	C	UPX1,VPX1	D		Value of polynomial (P(X)) at X1
PX2	C	UPX2,VPX2	D		Value of polynomial (P(X)) at X2
PX3	C	UPX3,VPX3	D		Value of polynomial (P(X)) at X3
CONV	L	CONV	L		CONV = true implies convergence has been obtained
X4	C	UX4,VX4	D		Newest approximation to root
J	I	J	I		Program control - counts the number of roots used as initial approximations
MAX	I	MAX	I	C	Maximum number of iterations permitted
I02	I	I02	I	C	Unit number of output device
				Subroutine ALTER	
X1	C	X1R,X1I	D	ECR	One of the three approximations to be altered
X2	C	X2R,X2I	D	ECR	One of the three approximations to be altered
X3	C	X3R,X3I	D	ECR	One of the three approximations to be altered
X2R	R	X2R	D		Real part of complex approximation
X2I	R	X2I	D		Imaginary part of complex approximation
				Subroutine QUAD	
EPST	R	EPST	D	E	Tolerance check for zero (0)
				Subroutine CALC	

These variables are dummy variables used for temporary storage and thus, are not defined.

MAIN PROGRAM

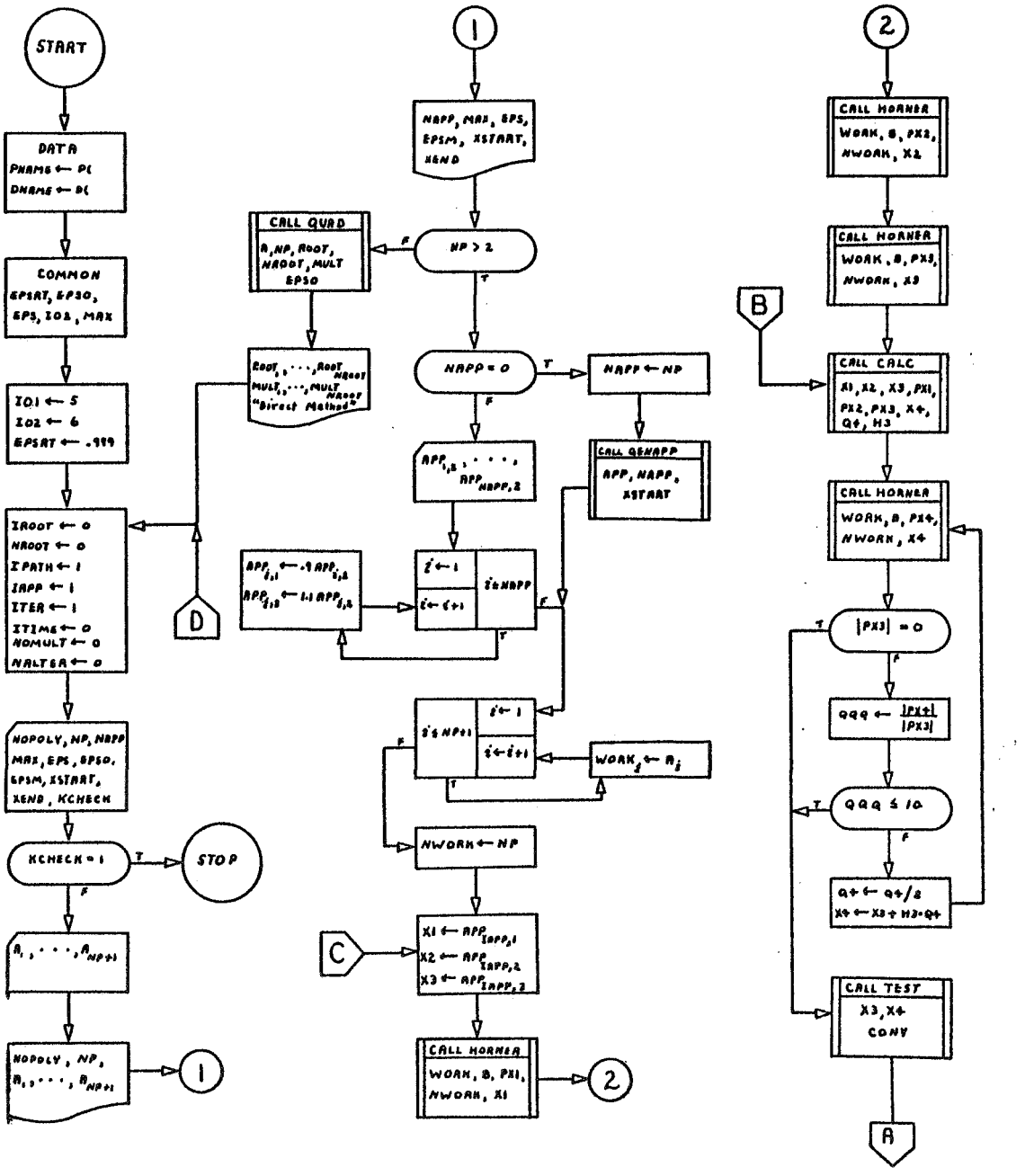


Figure C.1. Flow Charts for Muller's Method

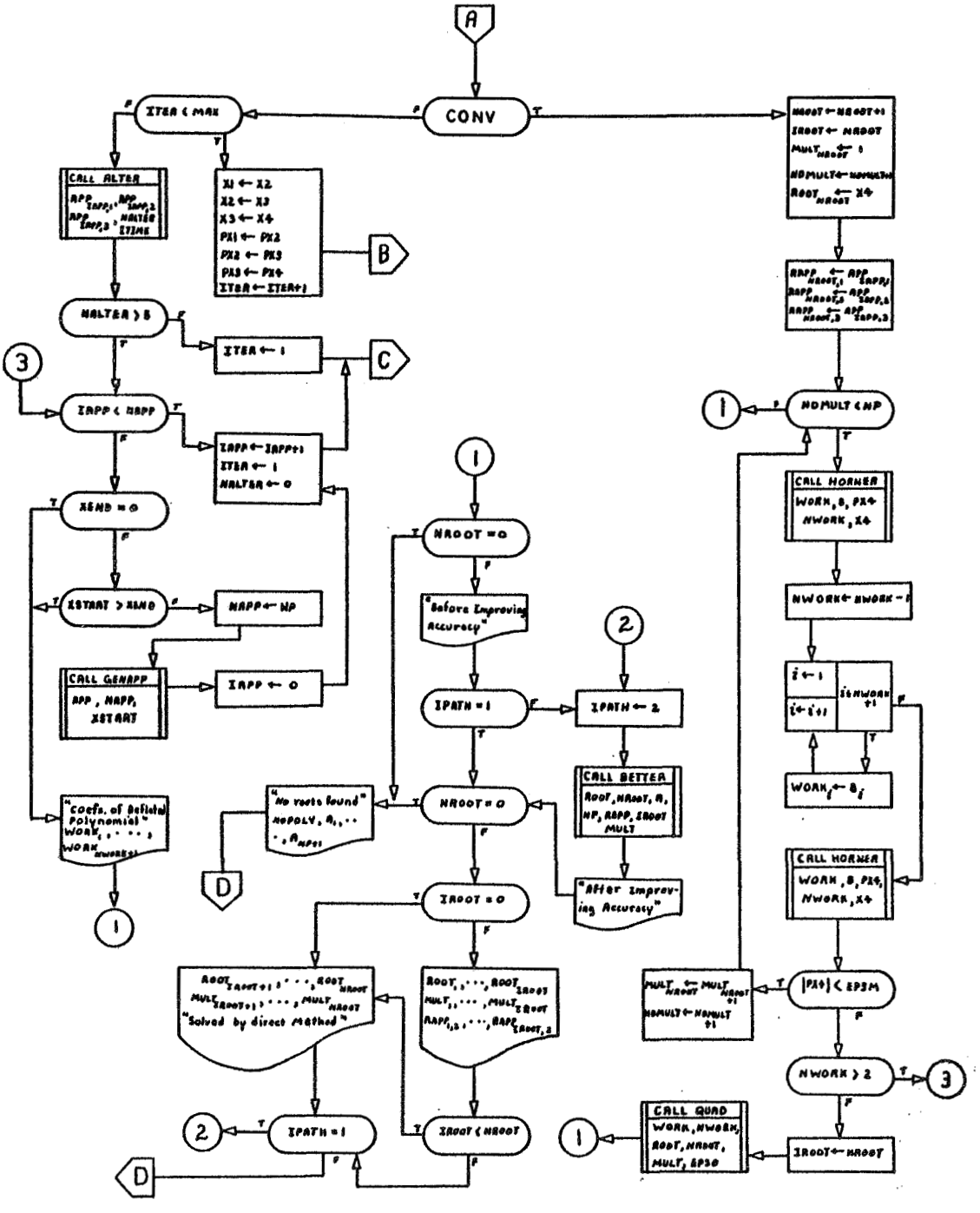


Figure C.1. (Continued)

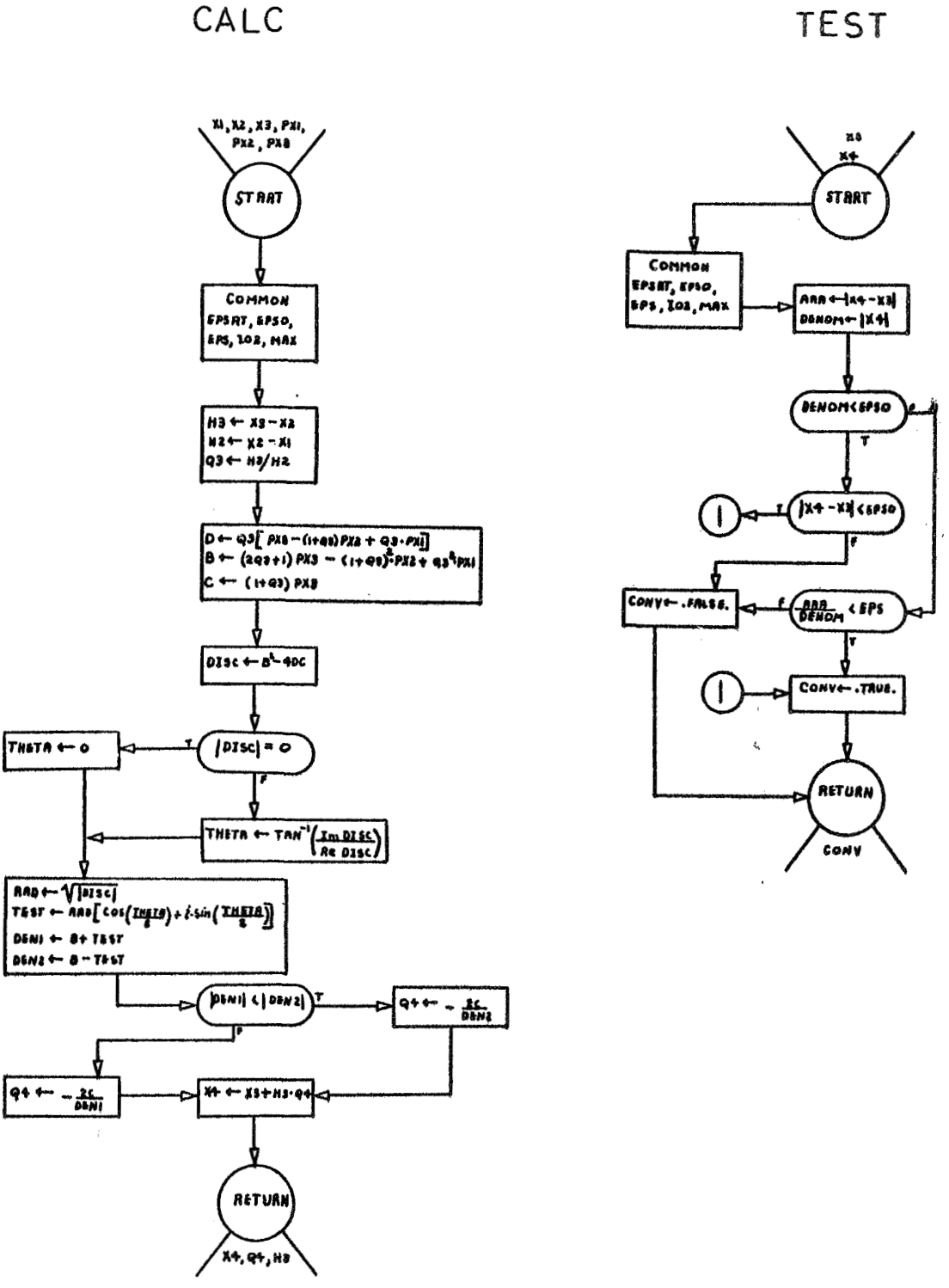


Figure C.1. (Continued)

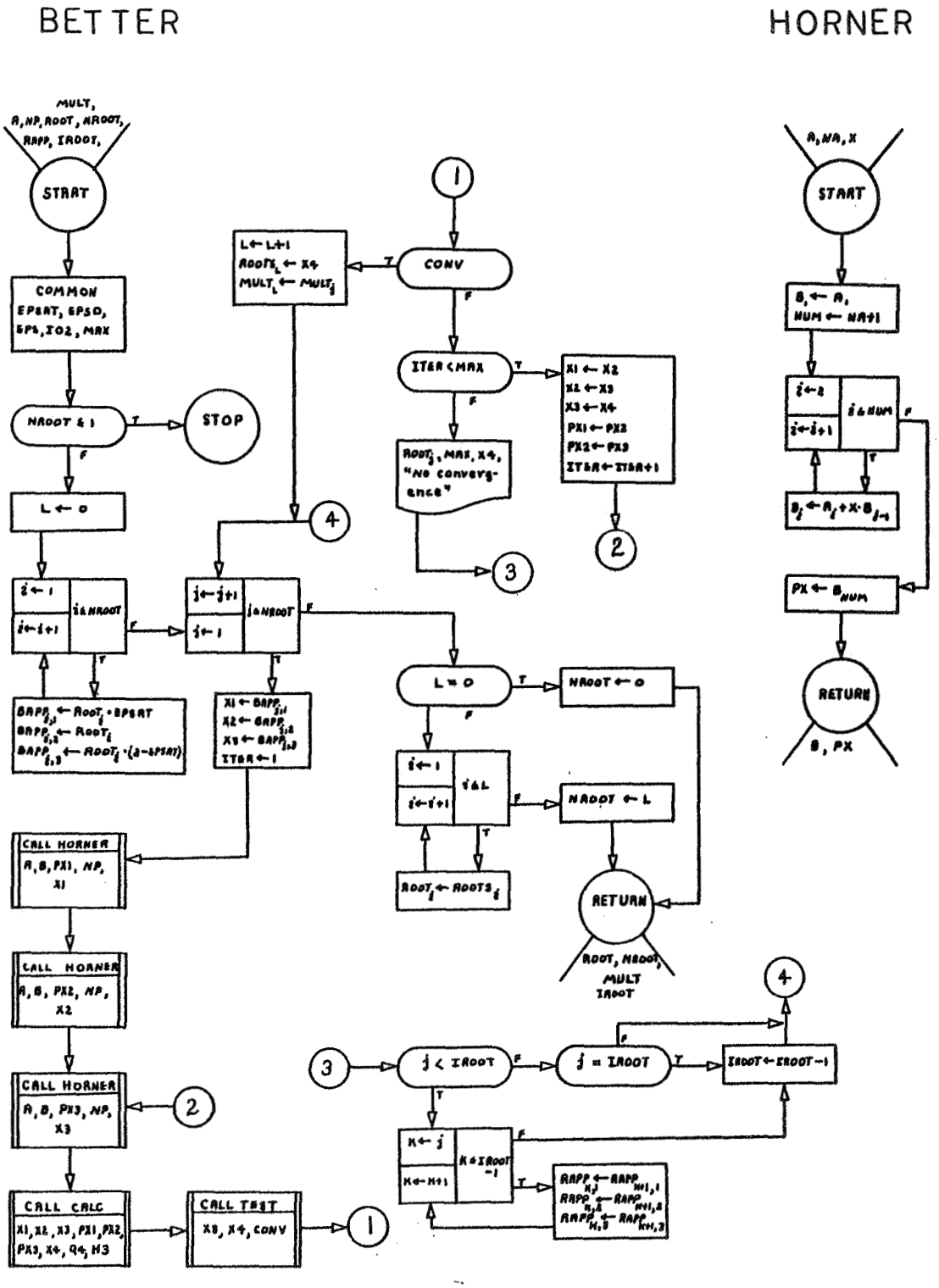


Figure C.1. (Continued)

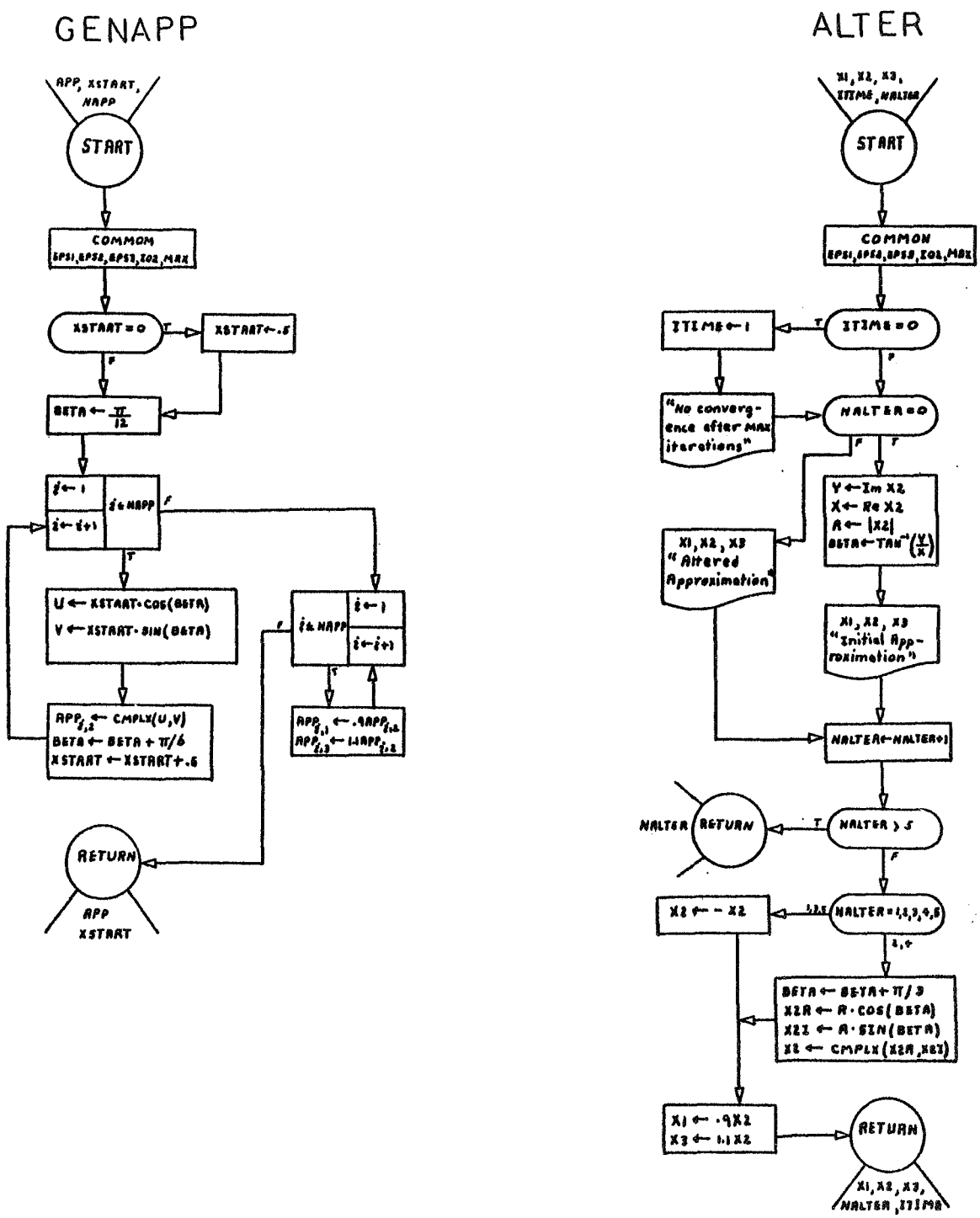
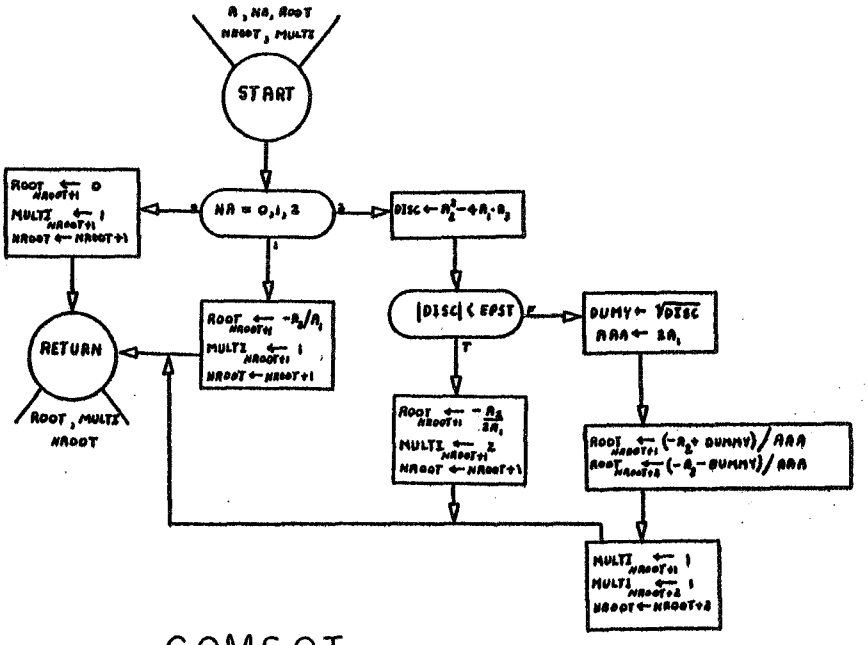


Figure C.1. (Continued)

QUAD



COMSQT

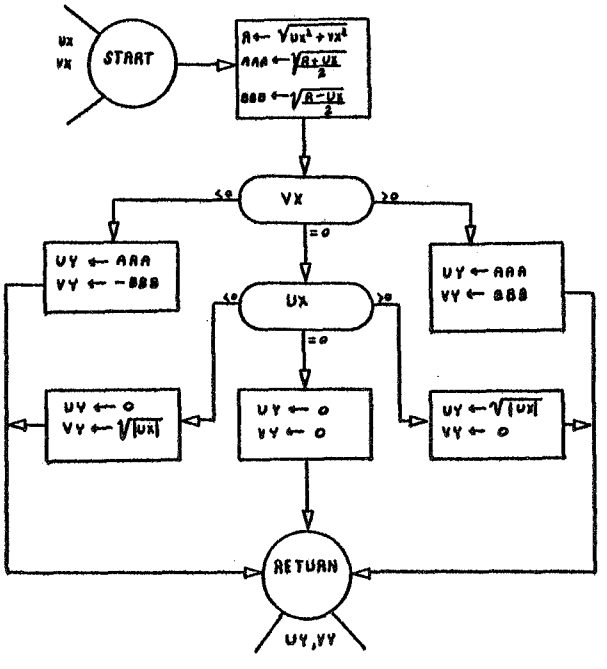


Figure C.1. (Continued)

TABLE C.V

PROGRAM FOR MULLER'S METHOD

```
C   ************************************************************************
C   *                                                                      *
C   * DOUBLE PRECISION PROGRAM FOR MULLER'S METHOD                         *
C   *                                                                      *
C   *                                                                      *
C   * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A     *
C   * POLYNOMIAL OF MAXIMUM DEGREE 25.  THROUGH THREE GIVEN POINTS THE     *
C   * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC.  THE ZERO OF THE QUADRATIC *
C   * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.  *
C   * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.          *
C   *                                                                      *
C   ************************************************************************
0001          DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,VX1,UAPP,VAPP
             1,UX2,VX2,UWORK,VWORK,UX3,VX3,UB,VB,UX4,VX4,UA,VA,UPX1,VPX1,URAPP,V
             2RAPP,UPX4,VPX4,EPSRT,EPSO,EPS,CCC,EPSM,UH3,VH3,UQ4,VQ4,ABPX4,ABPX3
             3,QQQ,XSTART,XEND
0002          DIMENSION UROOT(25),VROOT(25),MULT(25),UAPP(25,3),VAPP(25,3),UWORK
             1(26),VWORK(26),UB(26),VB(26),UA(26),VA(26),URAPP(25,3),VRAPP(25,3)
0003          DATA PNAME,DNAME/2HP(,2HD(/
0004          LOGICAL CONV
0005          COMMON EPSRT,EPSO,EPS,IO2,MAX
0006          IO1=5
0007          IO2=6
0008          EPSRT=0.999
0009       10 NROOT=0
0010          IROOT=0
0011          IPATH=1
0012          NOMULT=0
0013          NALTER=0
0014          ITIME=0
0015          IAPP=1
0016          ITER=1
0017          READ(IO1,1000) NOPOLY,NP,NAPP,MAX,EPS,EPSO,EPSM,XSTART,XEND,KCHECK
0018          IF(KCHECK.EQ.1) STOP
0019          KKK=NP+1
0020          READ(IO1,1010) (UA(I),VA(I),I=1,KKK)
0021          WRITE(IO2,1020) NOPOLY,NP
0022          WRITE(IO2,1035) (PNAME,I,UA(I),VA(I),I=1,KKK)
0023          WRITE(IO2,2060)
0024          WRITE(IO2,2000) NAPP
0025          WRITE(IO2,2010) MAX
0026          WRITE(IO2,2020) EPS
0027          WRITE(IO2,2030) EPSM
0028          WRITE(IO2,2040) XSTART
0029          WRITE(IO2,2050) XEND
0030          IF(NP.GT.2) GO TO 15
0031          CALL QUAD(UA,VA,NP,UROOT,VROOT,NROOT,MULT,EPSO)
0032          WRITE(IO2,1037)
0033          WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULT(I),I=1,NROOT)
0034          GO TO 10
0035       15 IF(NAPP.NE.0) GO TO 20
0036          NAPP=NP
0037          CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0038          GO TO 27
0039       20 READ(IO1,1030) (UAPP(I,2),VAPP(I,2),I=1,NAPP)
0040          DO 25 I=1,NAPP
0041          UAPP(I,1)=0.9*UAPP(I,2)
0042          VAPP(I,1)=0.9*VAPP(I,2)
```

TABLE C.V (Continued)

```
0043            UAPP(I,3)=1.1*UAPP(I,2)
0044          25 VAPP(I,3)=1.1*VAPP(I,2)
0045          27 KKK=NP+1
0046             DO 30 I=1,KKK
0047             UWORK(I)=UA(I)
0048          30 VWORK(I)=VA(I)
0049             NWORK=NP
0050          40 UX1=UAPP(IAPP,1)
0051             VX1=VAPP(IAPP,1)
0052             UX2=UAPP(IAPP,2)
0053             VX2=VAPP(IAPP,2)
0054             UX3=UAPP(IAPP,3)
0055             VX3=VAPP(IAPP,3)
0056             CALL HORNER(NWORK,UWORK,VWORK,UX1,VX1,UB,VB,UPX1,VPX1)
0057             CALL HORNER(NWORK,UWORK,VWORK,UX2,VX2,UB,VB,UPX2,VPX2)
0058             CALL HORNER(NWORK,UWORK,VWORK,UX3,VX3,UB,VB,UPX3,VPX3)
0059          50 CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
                14,VX4,UQ4,VQ4,UH3,VH3)
0060          60 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0061             ABPX4=DSQRT(UPX4*UPX4+VPX4*VPX4)
0062             ABPX3=DSQRT(UPX3*UPX3+VPX3*VPX3)
0063             IF(ABPX3.EQ.0.0) GO TO 70
0064             QQQ=ABPX4/ABPX3
0065             IF(QQQ.LE.10.) GO TO 70
0066             UQ4=0.5*UQ4
0067             VQ4=0.5*VQ4
0068             UX4=UX3+(UH3*UQ4-VH3*VQ4)
0069             VX4=VX3+(VH3*UQ4+UH3*VQ4)
0070             GO TO 60
0071          70 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0072             IF(CONV) GO TO 120
0073             IF(ITER.LT.MAX) GO TO 110
0074             CALL ALTER(UAPP(IAPP,1),VAPP(IAPP,1),UAPP(IAPP,2),VAPP(IAPP,2),UAP
                1P(IAPP,3),VAPP(IAPP,3),NALTER,ITIME)
0075             IF(NALTER.GT.5) GO TO 75
0076             ITER=1
0077             GO TO 40
0078          75 IF(IAPP.LT.NAPP) GO TO 100
0079             IF(XEND.EQ.0.0) GO TO 77
0080             IF(XSTART.GT.XEND) GO TO 77
0081             NAPP=NP
0082             CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0083             IAPP=0
0084             GO TO 100
0085          77 WRITE(IO2,1090)
0086             KKK=NWORK+1
0087             WRITE(IO2,1035) (ONAME,J,UWORK(J),VWORK(J),J=1,KKK)
0088          80 IF(NROOT.EQ.0) GO TO 90
0089             WRITE(IO2,1060)
0090             IF(IPATH.EQ.1) GO TO 82
0091          81 IPATH=2
0092             CALL BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MULT)
0093             WRITE(IO2,1200)
0094          82 IF(NROOT.EQ.0)GO TO 90
0095             IF(IROOT.EQ.0) GO TO 85
0096             WRITE(IO2,1080)
0097             DO 55 I=1,IROOT
0098          55 WRITE(IO2,1085) I,UROOT(I),VROOT(I),MULT(I),URAPP(I,2),VRAPP(I,2)
```

TABLE C.V (Continued)

```
0099              IF(IROOT.LT.NROOT) GO TO 85
0100              GO TO 87
0101           85 KKK=IROOT+1
0102              WRITE(IO2,1086) (I,UROOT(I),VROOT(I),MULT(I),I=KKK,NROOT)
0103           87 IF(IPATH.EQ.1) GO TO 81
0104              GO TO 10
0105           90 WRITE(IO2,1070) NOPOLY
0106              GO TO 10
0107          100 IAPP=IAPP+1
0108              ITER=1
0109              NALTER=0
0110              GO TO 40
0111          120 NROOT=NROOT+1
0112              IROOT=NROOT
0113              MULT(NROOT)=1
0114              NOMULT=NOMULT+1
0115              UROOT(NROOT)=UX4
0116              VROOT(NROOT)=VX4
0117              URAPP(NROOT,1)=UAPP(IAPP,1)
0118              VRAPP(NROOT,1)=VAPP(IAPP,1)
0119              URAPP(NROOT,2)=UAPP(IAPP,2)
0120              VRAPP(NROOT,2)=VAPP(IAPP,2)
0121              URAPP(NROOT,3)=UAPP(IAPP,3)
0122              VRAPP(NROOT,3)=VAPP(IAPP,3)
0123          125 IF(NOMULT.LT.NP) GO TO 130
0124              GO TO 80
0125          130 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0126              NWORK=NWORK-1
0127              KKK=NWORK+1
0128              DO 140 I=1,KKK
0129              UWORK(I)=UB(I)
0130          140 VWORK(I)=VB(I)
0131              CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0132              CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
0133              IF(CCC.LT.EPSM) GO TO 150
0134              IF(NWORK.GT.2) GO TO 75
0135              IROOT=NROOT
0136              CALL QUAD(UWORK,VWORK,NWORK,UROOT,VROOT,NROOT,MULT,EPSO)
0137              GO TO 80
0138          150 MULT(NROOT)=MULT(NROOT)+1
0139              NOMULT=NOMULT+1
0140              GO TO 125
0141          110 UX1=UX2
0142              VX1=VX2
0143              UX2=UX3
0144              VX2=VX3
0145              UX3=UX4
0146              VX3=VX4
0147              UPX1=UPX2
0148              VPX1=VPX2
0149              UPX2=UPX3
0150              VPX2=VPX3
0151              UPX3=UPX4
0152              VPX3=VPX4
0153              ITER=ITER+1
0154              GO TO 50
0155         1010 FORMAT(2D30.0)
0156         1020 FORMAT(1H1,1X,52HMULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOM
```

TABLE C.V (Continued)

```
     1IAL/1H ,1X,18HPOLYNOMIAL NUMBER ,I2,11H OF DEGREE ,I2///1H ,1X,28H
     2THE COEFFICIENTS OF P(X) ARE//)
0157 1030 FORMAT(2D30.0)
0158 1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
     1ZEROS WERE FOUND//)
0159 1080 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
     1 APPROXIMATION//)
0160 1070 FORMAT(//,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
0161 1086 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,9X,23HS
     1OLVED BY DIRECT METHOD)
0162 1037 FORMAT(///,1X,13HZEROS OF P(X),51X,14HMULTIPLICITIES//)
0163 1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
0164 1085 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,8X,D23.
     116,3H + ,D23.16,2H I)
0165 1000 FORMAT(3(I2,1X),9X,I3,8X,3(D6.0,1X),13X,2(D7.0,1X),I1)
0166 1060 FORMAT(///35H BEFORE ATTEMPT TO IMPROVE ACCURACY)
0167 1200 FORMAT(///1X,37HAFTER THE ATTEMPT TO IMPROVE ACCURACY)
0168 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
0169 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
0170 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
0171 2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
0172 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
0173 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
0174 2060 FORMAT(//1X)
0175       END
```

TABLE C.V (Continued)

```
0001            SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)
              C   ************************************************************************
              C   *                                                                      *
              C   * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO   *
              C   * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.*
              C   *                                                                      *
              C   ************************************************************************
0002            DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA
0003            COMMON EPS1,EPS2,EPS3,IO2,MAX
0004            IF(ITIME.NE.0) GO TO 5
0005            ITIME=1
0006            WRITE(IO2,1010) MAX
0007          5 IF(NALTER.EQ.0) GO TO 10
0008            WRITE(IO2,1000) X1R,X1I,X2R,X2I,X3R,X3I
0009            GO TO 20
0010         10 R=DSQRT(X2R*X2R+X2I*X2I)
0011            BETA=DATAN2(X2I,X2R)
0012            WRITE(IO2,1020) X1R,X1I,X2R,X2I,X3R,X3I
0013         20 NALTER=NALTER+1
0014            IF(NALTER.GT.5) RETURN
0015            GO TO (30,40,30,40,30),NALTER
0016         30 X2R=-X2R
0017            X2I=-X2I
0018            GO TO 50
0019         40 BETA=BETA+1.0471976
0020            X2R=R*DCOS(BETA)
0021            X2I=R*DSIN(BETA)
0022         50 X1R=0.9*X2R
0023            X1I=0.9*X2I
0024            X3R=1.1*X2R
0025            X3I=1.1*X2I
0026            RETURN
0027       1000 FORMAT(1X,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HALTERED APPROXIM
                1ATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
                2,D23.16,2H I/)
0028       1020 FORMAT(1HO,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HINITIAL APPROXI
                1MATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
                2 ,D23.16,2H I/)
0029       1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
                1TER ,I3,12H ITERATIONS.//)
0030            END
```

TABLE C.V (Continued)

```
0001            SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
                C  ***************************************************************************
                C  *                                                                         *
                C  * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE    *
                C  * DEGREE OF THE ORIGINAL POLYNOMIAL.                                      *
                C  *                                                                         *
                C  ***************************************************************************
0002            DOUBLE PRECISION APPR,APPI,XSTART,EPS1,EPS2,EPS3,BETA
0003            DIMENSION APPR(25,3),APPI(25,3)
0004            COMMON EPS1,EPS2,EPS3,IO2,MAX
0005            IF(XSTART.EQ.0.0) XSTART=0.5
0006            BETA=0.2617994
0007            DO 10 I=1,NAPP
0008            APPR(I,2)=XSTART*DCOS(BETA)
0009            APPI(I,2)=XSTART*DSIN(BETA)
0010            BETA=BETA+0.5235988
0011         10 XSTART=XSTART+0.5
0012            DO 20 I=1,NAPP
0013            APPR(I,1)=0.9*APPR(I,2)
0014            APPI(I,1)=0.9*APPI(I,2)
0015            APPR(I,3)=1.1*APPR(I,2)
0016         20 APPI(I,3)=1.1*APPI(I,2)
0017            RETURN
0018            END
```

TABLE C.V (Continued)

```
0001            SUBROUTINE BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MUL
               1T)
          C    ***************************************************************************
          C    *                                                                         *
          C    * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND   *
          C    * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO  *
          C    * THE FULL, UNDEFLATED POLYNOMIAL.                                        *
          C    *                                                                         *
          C    ***************************************************************************
0002            DOUBLE PRECISION UROOT,VROOT,UA,VA,UBAPP,VBAPP,UX1,VX1,UX2,VX2,UX3
               1,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UB,VB,UROOTS,VROOTS,EPSRT,UX4,V
               2X4,URAPP,VRAPP,EPSO,EPS,UQ4,VQ4,UH3,VH3
0003            LOGICAL CONV
0004            DIMENSION UROOT(25),VROOT(25),UA(26),VA(26),UBAPP(25,3),VBAPP(25,3
               1),UB(26),VB(26),UROOTS(25),VROOTS(25),URAPP(25,3),VRAPP(25,3),MULT
               3(25)
0005            COMMON EPSRT,EPSO,EPS,IO2,MAX
0006            IF(NROOT.LE.1) RETURN
0007            L=0
0008            DO 10 I=1,NROOT
0009            UBAPP(I,1)=UROOT(I)*EPSRT
0010            VBAPP(I,1)=VROOT(I)*EPSRT
0011            UBAPP(I,2)=UROOT(I)
0012            VBAPP(I,2)=VROOT(I)
0013            UBAPP(I,3)=UROOT(I)*(2.0-EPSRT)
0014         10 VBAPP(I,3)=VROOT(I)*(2.0-EPSRT)
0015            DO 100 J=1,NROOT
0016            UX1=UBAPP(J,1)
0017            VX1=VBAPP(J,1)
0018            UX2=UBAPP(J,2)
0019            VX2=VBAPP(J,2)
0020            UX3=UBAPP(J,3)
0021            VX3=VBAPP(J,3)
0022            ITER=1
0023            CALL HORNER(NP,UA,VA,UX1,VX1,UB,VB,UPX1,VPX1)
0024            CALL HORNER(NP,UA,VA,UX2,VX2,UB,VB,UPX2,VPX2)
0025         20 CALL HORNER(NP,UA,VA,UX3,VX3,UB,VB,UPX3,VPX3)
0026            CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
               14,VX4,UQ4,VQ4,UH3,VH3)
0027         30 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0028            IF(CONV) GO TO 50
0029            IF(ITER.LT.MAX) GO TO 40
0030            WRITE(IO2,1000) J,UROOT(J),VROOT(J),MAX
0031            WRITE(IO2,1010) UX4,VX4
0032            IF(J.LT.IROOT) GO TO 33
0033            IF(J.EQ.IROOT) GO TO 35
0034            GO TO 100
0035         33 KKK=IROOT-1
0036            DO 34 K=J,KKK
0037            URAPP(K,1)=URAPP(K+1,1)
0038            VRAPP(K,1)=VRAPP(K+1,1)
0039            URAPP(K,2)=URAPP(K+1,2)
0040            VRAPP(K,2)=VRAPP(K+1,2)
0041            URAPP(K,3)=URAPP(K+1,3)
0042         34 VRAPP(K,3)=VRAPP(K+1,3)
0043         35 IROOT=IROOT-1
0044            GO TO 100
0045         40 UX1=UX2
```

TABLE C.V (Continued)

```
0046              VX1=VX2
0047              UX2=UX3
0048              VX2=VX3
0049              UX3=UX4
0050              VX3=VX4
0051              UPX1=UPX2
0052              VPX1=VPX2
0053              UPX2=UPX3
0054              VPX2=VPX3
0055              ITER=ITER+1
0056              GO TO 20
0057           50 L=L+1
0058              UROOTS(L)=UX4
0059              VROOTS(L)=VX4
0060              MULT(L)=MULT(J)
0061          100 CONTINUE
0062              IF(L.EQ.0) GO TO 120
0063              DO 110 I=1,L
0064              UROOT(I)=UROOTS(I)
0065          110 VROOT(I)=VROOTS(I)
0066              NROOT=L
0067              RETURN
0068          120 NROOT=0
0069              RETURN
0070         1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
                 1D23.16,3H + ,D23.16,2H I/24H DID NOT CONVERGE AFTER ,I3,11H ITERAT
                 2IONS)
0071         1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,D23.16,3H + ,D23.16,2H I/
                 1/)
0072              END
```

TABLE C.V (Continued)

```
0001              SUBROUTINE CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,V
                 1PX3,UX4,VX4,UQ4,VQ4,UH3,VH3)
                C ********************************************************************************
                C *                                                                              *
                C * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC         *
                C * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF        *
                C * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION            *
                C * X(N+1) TO THE ZERO OF THE POLYNOMIAL.                                       *
                C *                                                                              *
                C ********************************************************************************
0002              DOUBLE PRECISION ARG1,ARG2
0003              DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1,
                 1VPX1,UH3,VH3,UH2,VH2,UQ3,VQ3,UD,VD,UB,VB,UC,VC,UDISC,VDISC,UCCC,VC
                 2CC,UDEN1,VDEN1,UDEN2,VDEN2,UQ4,VQ4,UX4,VX4,EPSRT,EPSO,EPS,UDDD,VDD
                 3D,AAA,BBB,RAD,UAAA,VAAA,UBBB,VBBB
0004              DOUBLE PRECISION THETA,ANGLE,UTEST,VTEST
0005              COMMON EPSRT,EPSO,EPS,IO2,MAX
0006              UH3=UX3-UX2
0007              VH3=VX3-VX2
0008              UH2=UX2-UX1
0009              VH2=VX2-VX1
0010              BBB=UH2*UH2+VH2*VH2
0011              UQ3=(UH3*UH2+VH3*VH2)/BBB
0012              VQ3=(VH3*UH2-UH3*VH2)/BBB
0013              UDDD=1.0+UQ3
0014              VDDD=VQ3
0015              UD=(UPX3-(UDDD*UPX2-VDDD*VPX2))+(UQ3*UPX1-VQ3*VPX1)
0016              VD=(VPX3-(VDDD*UPX2+UDDD*VPX2))+(VQ3*UPX1+UQ3*VPX1)
0017              UAAA=2.0*UQ3
0018              VAAA=2.0*VQ3
0019              UAAA=UAAA+1.0
0020              UBBB=UDDD*UDDD-VDDD*VDDD
0021              VBBB=VDDD*UDDD+UDDD*VDDD
0022              UCCC=UQ3*UQ3-VQ3*VQ3
0023              VCCC=VQ3*UQ3+UQ3*VQ3
0024              UB=((UAAA*UPX3-VAAA*VPX3)-(UBBB*UPX2-VBBB*VPX2))+(UCCC*UPX1-VCCC*V
                 1PX1)
0025              VB=((VAAA*UPX3+UAAA*VPX3)-(VBBB*UPX2+UBBB*VPX2))+(VCCC*UPX1+UCCC*V
                 1PX1)
0026              UC=UDDD*UPX3-VDDD*VPX3
0027              VC=VDDD*UPX3+UDDD*VPX3
0028              UDISC=(UB*UB-VB*VB)-(4.0*(UD*UC-VD*VC))
0029              VDISC=(2.0*(VB*UB))-(4.0*(VD*UC+UD*VC))
0030              AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
0031              IF(AAA.EQ.0.0) GO TO 5
0032              GO TO 7
0033            5 THETA=0.0
0034              GO TO 9
0035            7 THETA=DATAN2(VDISC,UDISC)
0036            9 RAD=DSQRT(AAA)
0037              ANGLE=THETA/2.0
0038              UTEST=RAD*DCOS(ANGLE)
0039              VTEST=RAD*DSIN(ANGLE)
0040              UDEN1=UB+UTEST
0041              VDEN1=VB+VTEST
0042              UDEN2=UB-UTEST
0043              VDEN2=VB-VTEST
0044              ARG1=UDEN1*UDEN1+VDEN1*VDEN1
```

TABLE C.V (Continued)

```
0045              ARG2=UDEN2*UDEN2+VDEN2*VDEN2
0046              AAA=DSQRT(ARG1)
0047              BBB=DSQRT(ARG2)
0048              IF(AAA.LT.BBB) GO TO 10
0049              IF(AAA.EQ.0.0) GO TO 60
0050              UAAA=-2.0*UC
0051              VAAA=-2.0*VC
0052              UQ4=(UAAA*UDEN1+VAAA*VDEN1)/ARG1
0053              VQ4=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
0054              GO TO 50
0055           10 IF(BBB.EQ.0.0) GO TO 60
0056              UAAA=-2.0*UC
0057              VAAA=-2.0*VC
0058              UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2
0059              VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2
0060              GO TO 50
0061           50 UX4=UX3+(UH3*UQ4-VH3*VQ4)
0062              VX4=VX3+(VH3*UQ4+UH3*VQ4)
0063              RETURN
0064           60 UQ4=1.0
0065              VQ4=0.0
0066              GO TO 50
0067              END
```

TABLE C.V (Continued)

```
0001            SUBROUTINE TEST(UX3,VX3,UX4,VX4,CONV)
              C ********************************************************************************
              C *                                                                              *
              C * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-            *
              C * IMATIONS BY TESTING THE EXPRESSION                                           *
              C * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).                   *
              C * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.                    *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UX3,VX3,UX4,VX4,EPSRT,EPSO,EPS,AAA,UDUMMY,VDUMMY,
               1DENOM
0003            LOGICAL CONV
0004            COMMON EPSRT,EPSO,EPS,IO2,MAX
0005            UDUMMY=UX4-UX3
0006            VDUMMY=VX4-VX3
0007            AAA=DSQRT(UDUMMY*UDUMMY+VDUMMY*VDUMMY)
0008            DENOM=DSQRT(UX4*UX4+VX4*VX4)
0009            IF(DENOM.LT.EPSO) GO TO 20
0010            IF(AAA/DENOM.LT.EPS) GO TO 10
0011          5 CONV=.FALSE.
0012            GO TO 100
0013         10 CONV=.TRUE.
0014            GO TO 100
0015         20 IF(AAA.LT.EPSO) GO TO 10
0016            GO TO 5
0017        100 RETURN
0018            END
```

```
0001            SUBROUTINE HORNER(NA,UA,VA,UX,VX,UB,VB,UPX,VPX)
              C ********************************************************************************
              C *                                                                              *
              C * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D.      *
              C * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE     *
              C * FACTOR (X-D).                                                               *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UX,VX,UPX,VPX,UB,VB,UA,VA
0003            DIMENSION UA(26),VA(26),UB(26),VB(26)
0004            UB(1)=UA(1)
0005            VB(1)=VA(1)
0006            NUM=NA+1
0007            DO 10 I=2,NUM
0008            UB(I)=UA(I)+(UB(I-1)*UX-VB(I-1)*VX)
0009         10 VB(I)=VA(I)+(VB(I-1)*UX+UB(I-1)*VX)
0010            UPX=UB(NUM)
0011            VPX=VB(NUM)
0012            RETURN
0013            END
```

TABLE C.V (Continued)

```
0001            SUBROUTINE QUAD(UA,VA,NA,UROOT,VROOT,NROOT,MULTI,EPST)
              C ********************************************************************************
              C *                                                                              *
              C * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES       *
              C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE        *
              C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                               *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UA,VA,UROOT,VROOT,BBB,UAAA,VAAA,UDISC,VDISC,UDUMM
               1Y,VDUMMY,RDUMMY,SDUMMY,EPST,UBBB,VBBB
0003            DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULTI(25)
0004            IF(NA.EQ.2) GO TO 7
0005            IF(NA.EQ.1) GO TO 5
0006            UROOT(NROOT+1)=0.0
0007            VROOT(NROOT+1)=0.0
0008            MULTI(NROOT+1)=1
0009            NROOT=NROOT+1
0010            GO TO 50
0011          5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
0012            UROOT(NROOT+1)=(-UA(2)*UA(1)-VA(2)*VA(1))/BBB
0013            VROOT(NROOT+1)=(-VA(2)*UA(1)+UA(2)*VA(1))/BBB
0014            MULTI(NROOT+1)=1
0015            NROOT=NROOT+1
0016            GO TO 50
0017          7 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(1)*UA(3)-VA(1)*VA(3)))
0018            VDISC=(VA(2)*UA(2)+UA(2)*VA(2))-(4.0*(VA(1)*UA(3)+UA(1)*VA(3)))
0019            BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
0020            IF(BBB.LT.EPST) GO TO 10
0021            CALL COMSQT(UDISC,VDISC,UDUMMY,VDUMMY)
0022            UBBB=-UA(2)+UDUMMY
0023            VBBB=-VA(2)+VDUMMY
0024            RDUMMY=-UA(2)-UDUMMY
0025            SDUMMY=-VA(2)-VDUMMY
0026            UAAA=2.0*UA(1)
0027            VAAA=2.0*VA(1)
0028            BBB=UAAA*UAAA+VAAA*VAAA
0029            UROOT(NROOT+1)=(UBBB*UAAA+VBBB*VAAA)/BBB
0030            VROOT(NROOT+1)=(VBBB*UAAA-UBBB*VAAA)/BBB
0031            UROOT(NROOT+2)=(RDUMMY*UAAA+SDUMMY*VAAA)/BBB
0032            VROOT(NROOT+2)=(SDUMMY*UAAA-RDUMMY*VAAA)/BBB
0033            MULTI(NROOT+1)=1
0034            MULTI(NROOT+2)=1
0035            NROOT=NROOT+2
0036            GO TO 50
0037         10 UAAA=2.0*UA(1)
0038            VAAA=2.0*VA(1)
0039            BBB=UAAA*UAAA+VAAA*VAAA
0040            UROOT(NROOT+1)=(-UA(2)*UAAA-VA(2)*VAAA)/BBB
0041            VROOT(NROOT+1)=(-VA(2)*UAAA+UA(2)*VAAA)/BBB
0042            MULTI(NROOT+1)=2
0043            NROOT=NROOT+1
0044         50 RETURN
0045            END
```

TABLE C.V (Cnntinued)

```
0001            SUBROUTINE COMSQT(UX,VX,UY,VY)
                C  ***************************************************************************
                C  *                                                                         *
                C  * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.           *
                C  *                                                                         *
                C  ***************************************************************************
0002            DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
0003            R=DSQRT(UX*UX+VX*VX)
0004            AAA=DSQRT(DABS((R+UX)/2.0))
0005            BBB=DSQRT(DABS((R-UX)/2.0))
0006            IF(VX) 10,20,30
0007          10 UY=AAA
0008             VY=-1.0*BBB
0009             GO TO 100
0010          20 IF(UX) 40,50,60
0011          30 UY=AAA
0012             VY=BBB
0013             GO TO 100
0014          40 DUMMY=DABS(UX)
0015             UY=0.0
0016             VY=DSQRT(DUMMY)
0017             GO TO 100
0018          50 UY=0.0
0019             VY=0.0
0020             GO TO 100
0021          60 DUMMY=DABS(UX)
0022             UY=DSQRT(DUMMY)
0023             VY=0.0
0024         100 RETURN
0025             END
```

APPENDIX D

SPECIAL FEATURES OF THE G.C.D. AND THE REPEATED G.C.D. PROGRAMS

Several special features have been provided in each program as an aid to the user and to improve accuracy of the results. These are explained and illustrated below.*

1. Generating Approximations

If the user does not have initial approximations available, subroutine GENAPP can systematically generate, for an N^{th} degree polynomial, N initial approximations of increasing magnitude, beginning with the magnitude specified by XSTART. If XSTART is 0., XSTART is automatically initialized to 0.5 to avoid the approximation 0. + 0.i. The approximations are generated according to the formula:

$$X_K = (XSTART + 0.5K)\ (\cos \beta + i \sin \beta)$$

where

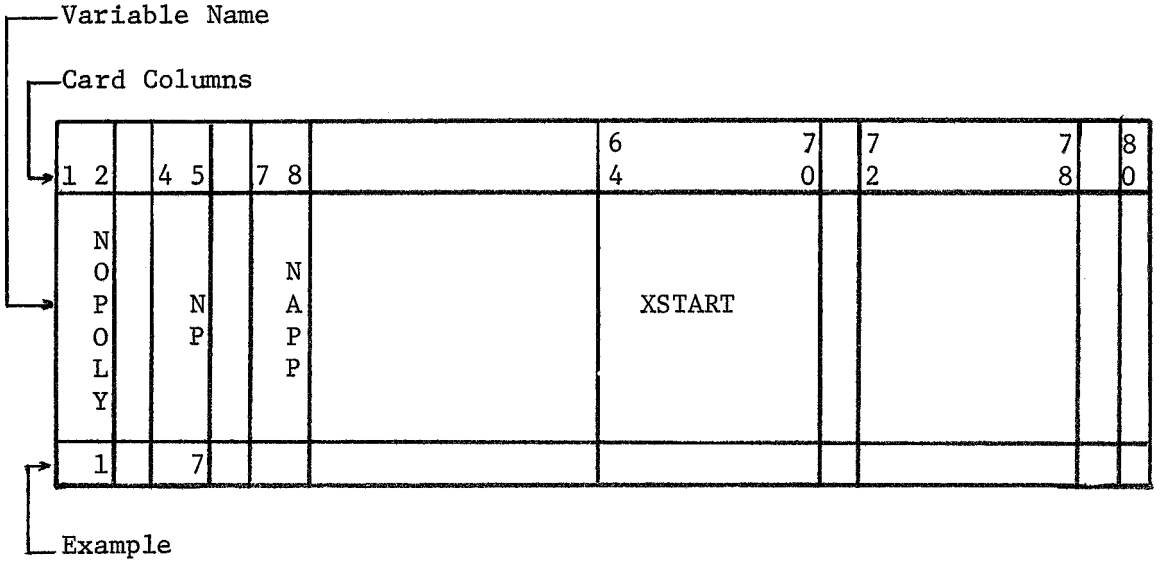
$$\beta = \frac{\pi}{12} + K\,\frac{\pi}{6}\ ,\ K = 0,1,2,\ldots$$

To accomplish this, the user defines the number of initial approximations to be read (NAPP) on the control card to be zero (0) or these columns

*These illustrations are representative of G.C.D.-Newton's method in double precision. Control cards for other methods should be prepared accordingly.

(7-8) may be left blank. If XSTART is left blank, it is interpreted as 0.

For example, a portion of a control card which generates initial approximations beginning at the origin for a seventh degree polynomial is shown in Example D.1.



Example D.1

The approximations are generated in a spiral configuration as illustrated in Figure A.1.

Example D.2 shows a portion of a control card which generates initial approximations beginning at a magnitude of 25.0 for a sixth degree polynomial.

1 2		4 5		7 8		6 4 7 0		7 2 7 8		8 0
NOPOLY		NP		NAPP		XSTART				
2		6				2.5D+01				

Example D.2

Note that if the approximations are generated beginning at the origin, the order in which the roots are found will probably be of increasing magnitude. Roots obtained in this way are usually more accurate.

2. Altering Approximations

If an initial approximation, X_0, does not produce convergence to a root within the maximum number of iterations, it is systematically altered a maximum of five times until convergence is possibly obtained according to the following formulas:

If the number of the alteration is odd: $(j = 1,3)$

$$X_{j+1} = |X_0| \ (\text{Cos } \beta + i \text{ Sin } \beta) \text{ where}$$

$$\beta = \text{Tan}^{-1} \frac{\text{Im } X_0}{\text{Re } X_0} + K \frac{\pi}{3} \ ; \ \begin{matrix} K = 1 \text{ if } j = 1 \\ K = 2 \text{ if } j = 3. \end{matrix}$$

If the number of the alteration is even: $(j = 0,2,4)$

$$X_{j+1} = -X_j .$$

Each altered approximation is then taken as a starting approximation. If none of the six starting approximations produce convergence, the next initial approximation is taken, and the process repeated. The six approximations are spaced 60 degrees apart on a circle of radius $|X_0|$ centered at the origin as illustrated in Figure A.2.

3. Searching the Complex Plane

By use of initial approximations and the altering technique, any region of the complex plane in the form of an annulus centered at the origin can be searched for roots. This procedure can be accomplished in two ways.

The first way is more versatile but requires more effort on the part of the user. Specifically selected initial approximation can be used to define particular regions to be searched. For example, if the roots of a particular polynomial are known to have magnitudes between 20 and 40 an annulus of inner radius 20 and outer radius 40 could be searched by using the initial approximations 20. + i, 23. + i, 26. + i, 29. + i, 32. + i, 35. + i, 38. + i, 40. + i.

By generating initial approximations internally, the program can search an annulus centered at the origin of inner radius XSTART and outer radius XEND. Values for XSTART and XEND are supplied on the control card by the user. Example D.3 shows a portion of a control card to search the above annulus of inner radius 20.0 and outer radius 40.0.

1 2		4 5		7 8		64 70		72 78		80
NOPOLY		NP		NAPP		XSTART		XEND		
1		7				2.0D+01		4.0D+01		

Example D.3

Note that since not less than N initial approximations can be generated at one time, the outer radius of the annulus actually searched may be greater than XEND but not greater than XEND + .5N.

Example D.4 shows a control card to search a circle of radius 15.

1 2		4 5		7 8		64 70		72 78		80
NOPOLY		NP		NAPP		XSTART		XEND		
1		7						1.5D+01		

Example D.4

Figure A.3 shows the distribution of initial and altered approximations for an annulus of width 2 and inner radius a.

4. Improving Zeros Found

After the zeros of a polynomial are found, they are printed under the heading "Roots of Q(X)." They are then used as initial approximations with Newton's (Muller's) method applied each time to the full (undeflated) polynomial Q(X), which contains only distinct roots. In most cases, zeros that have lost accuracy due to roundoff error in the deflation process are improved. The improved zeros are then printed under the heading "Roots of P(X)." Since each root is used as an approximation to the original (undeflated) polynomial Q(X), it is possible that the root may converge to an entirely different root. This is especially true where several zeros are close together. Therefore, the user should check both lists of zeros to determine whether or not this has occurred.

5. Solving Quadratic Polynomial

After N-2 roots of an N^{th} degree polynomial have been extracted, the remaining quadratic, $aX^2 + bX + c$, is solved using the quadratic formula

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

for the two remaining roots. These are indicated by the words "Results of Subroutine QUAD" in the initial approximation column. If only a polynomial of degree 1 is to be solved, the solution is found directly as $(X - C) = 0$ implies $X = C$.

6. Missing Roots

If not all N roots of an N^{th} degree polynomial are found, the coefficients of the remaining deflated polynomial are printed under the heading "Coefficients of Deflated Polynomial For Which No Zeros Were Found." The user may then work with this polynomial in an attempt to find the remaining roots. The leading coefficient (coefficient of the highest degree term) will be printed first (Exhibit 6.11)

7. Miscellaneous

By using various combinations of values for NAPP, XSTART, and XEND, the user has several options available as illustrated below.

Example D.5 shows the control card for a seventh degree polynomial. Three initial approximations are supplied by the user. At most three roots will be found and the coefficients of the remaining deflated polynomial will be printed.

1 2		4 5		7 8		64 70		72 78		80
NOPOLY		NP		NAPP		XSTART		XEND		
1		7		3						

Example D.5

Note that if several roots are known to the user, they may be "divided out" of the original polynomial by using this procedure.

Example D.6 indicates that 2 initial approximations are supplied by the user to a 7^{th} degree polynomial. After these approximations are used the circle of radius 15 will be searched for the remaining roots.

1 2	4 5	7 8		64 70	72 78	80
NOPOLY	NP	NAPP		XSTART	XEND	
1	7	2			1.5D+01	

Example D.6

By defining XSTART between 0. and 15. an annulus instead of the circle will be searched.

APPENDIX E

G.C.D. - NEWTON'S METHOD

1. Use of the Program

A double precision FORTRAN IV program using the G.C.D. method with Newton's method as a supporting method is presented here. Flow charts for this program are given in Figure E.6 while Table E.VII gives a FORTRAN IV listing of this program. Single precision variables are listed in some of the tables. The simple precision variables are used in the flow charts and the corresponding double precision variables can be obtained from the appropriate tables.

This program is designed to solve polynomials having degree less than or equal to 25. In order to solve polynomials of degree N where N > 25, the data statement and array dimensions given in Talbe E.I must be changed.

In this program both the leading coefficient and the constant coefficient are assumed to be non-zero.

TABLE E.I

PROGRAM CHANGES NECESSARY TO SOLVE POLYNOMIALS OF DEGREE GREATER THAN 25 BY G.C.D. - NEWTON'S METHOD

Main Program

Data Entry/1H1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1
UP(N+1), VP(N+1)
UAPP(N), VAPP(N)
UROOT(N), VROOT(N)
MULT(N)
UDP(N+1), VDP(N+1)
UD(N+1), VD(N+1)
UQ(N+1), VQ(N+1)
UQQ(N+1), VQQ(N+1)
UAP(N), VAP(N)
UQD(N+1), VQD(N+1)
ENTRY(N+1)
UROOTS(N), VROOTS(N)

Subroutine GENAPP

APPR(N), APPI(N)

Subroutine GCD

UR(N+1), VR(N+1)
US(N+1), VS(N+1)
USS(N+1), VSS(N+1)
URR(N+1), VRR(N+1)
UT(N+1), VT(N+1)

Subroutine QUAD

UA(N+1), VA(N+1)
UROOT(N), VROOT(N)
MULT(N)

Subroutine NEWTON

UP(N+1), VP(N+1)
UB(N+1), VB(N+1)

Subroutine DIVIDE

UP(N+1), VP(N+1)
UD(N+1), VD(N+1)
UQ(N+1), VQ(N+1)

TABLE E.I (Continued)

Subroutine HORNER

UP(N+1), VP(N+1)
UB(N+1), VB(N+1)

Subroutine DERIV

UP(N+1), VP(N+1)
UA(N+1), VA(N+1)

Subroutine MULTI

UP(N+1), VP(N+1)
UROOT(N), VROOT(N)
UA(N+1), VA(N+1)
UB(N+1), VB(N+1)
MULT(N)

2. Input Data for G.C.D. - Newton's Method

The input data for G.C.D. - Newton's method is grouped into polynomial data sets. Each polynomial data set consists of the data for one and only one polynomial. As many polynomials as the user desires may be solved by placing the polynomial data sets one behind the other. Each polynomial data set consists of three kinds of information placed in the following order:

1. Control information.
2. Coefficients of the polynomial.
3. Initial approximations. These may be omitted as described in Appendix D, § 1.

An end card follows the entire collection of data sets. It indicates that there is no more data to follow and terminates execution of the

program. This information is displayed in Figure E.1 and described below. All data should be right justified and the D-type specification should be used. The recommendations given in Table E.II are those found to give best results on the IBM 360/50 computer which has a 32 bit word.

Control Information

The control card is the first card of the polynomial data set and contains the information given in Table E.II. See Figure E.2.

TABLE E.II

CONTROL DATA FOR G.C.D. - NEWTON'S METHOD

Variable Name	Card Columns	Description
NOPOLY	c.c. 1-2	Number of the polynomial. Integer. Right justified.
NP	c.c. 4-5	Degree of the polynomial. Integer. Right justified.
NAPP	c.c. 7-8	Number of initial approximations to be read. Integer. Right justified. If no initial approximations are given, leave blank.
MAX	c.c. 19-21	Maximum number of iterations. Integer. Right justified. 200 is recommended.
EPS1	c.c. 23-28	Test for zero in subroutine GCD. Double precision. Right justify. 1.D-03 is recommended.

TABLE E.II (Continued)

Variable Name	Card Columns	Description
EPS2	c.c. 30-35	Convergence requirement. Double precision. Right justify. 1.D-10 is recommended.
EPS3	c.c. 37-42	Test for zero in subroutine QUAD. Double precision. Right justify. 1.D-20 is recommended.
EPS4	c.c. 44-49	Multiplicity requirement. Double precision. Right justify. 1.D-02 is recommended.
XSTART	c.c. 64-70	Magnitude at which to begin generating initial approximations. Double precision. Right justify. This is a special feature of the program and may be omitted.
XEND	c.c. 72-78	Magnitude at which to end the generating of initial approximations. Double precision. Right justify. This is a special feature of the program and may be omitted.
KCHECK	c.c. 80	This should be left blank.

Coefficients of the Polynomial

The coefficient cards follow the control card. For an N^{th} degree polynomial, N+1 coefficients must be entered one per card. The coefficient of the highest degree term is entered first; that is, the leading coefficient is entered first. For example, if the polynomial $X^5 + 3X^4 + 2X + 5$ were to be solved for its zeros, the order in which

the coefficients would be entered is: 1, 3, 0, 0, 2, 5. Each real or complex coefficient is entered, one per card, as described in Table E.III and illustrated in Figure E.3.

TABLE E.III

COEFFICIENT DATA FOR G.C.D. - NEWTON'S METHOD

Variable Name	Card Columns	Description
UP (P in single precision)	c.c. 1-30	Real part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0D00.
VP (P in single precision)	c.c. 31-60	Imaginary part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0D00.

Initial Approximations

The initial approximation cards follow the set of coefficient cards. The number of initial approximations read must be the number specified on the control card and are entered, one per card, as given in Table E.IV and illustrated in Figure E.4.

TABLE E.IV

INITIAL APPROXIMATION DATA FOR G.C.D. - NEWTON'S METHOD

Variable Name	Card Columns	Description
UAPP (APP in single precision)	c.c. 1-30	Real part of complex number. Double precision. Right justify. If none, leave blank or enter 0.0D00.
VAPP (APP in single precision)	c.c. 31-60	Imaginary part of complex number. Double precision. Right justify. If none, leave blank or enter 0.0D00.

End Card

The end card is the last card of the input data to the program. It indicates that there is no more data to be read. When this card is read, program execution is terminated. This card is described in Table E.V and illustrated in Figure E.5.

TABLE E.V

DATA TO END EXECUTION OF G.C.D. - NEWTON'S METHOD

Variable Name	Card Column	Description
KCHECK	c.c. 80	Must contain the number 1. Integer.

3. Variables Used in G.C.D. - Newton's Method

The definitions of the major variables used in G.C.D. - Newton's method are given in Table E.VI. The symbols used to indicate type are:

R - real variable

I - integer variable

D - double precision

C - complex variable

L - logical variable

A - alphanumeric variable

When two variables are listed, the one on the left is the real part of the corresponding single precision complex variable; the one on the right is the imaginary part. The symbols used to indicate disposition are:

E - entered

R - returned

ECR - entered, changed, and returned

C - variable in common

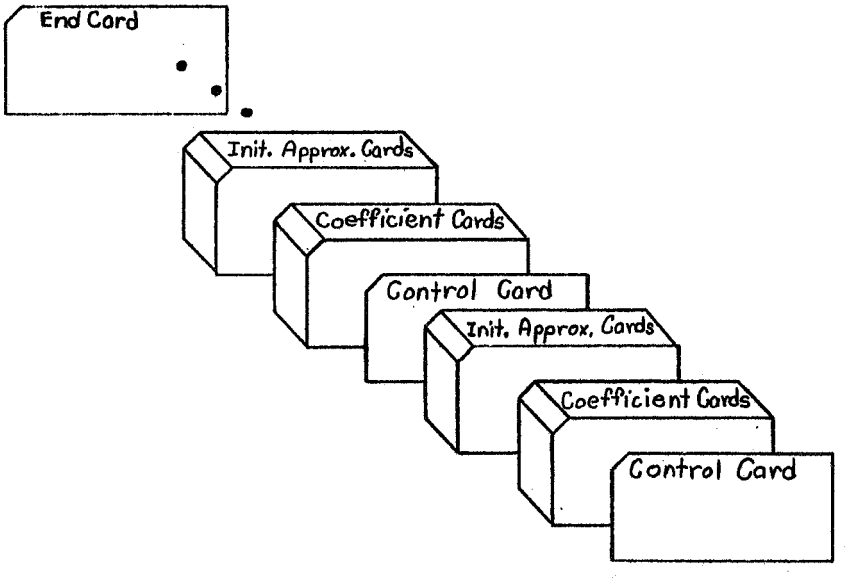


Figure E.1. Sequence of Input Data for G.C.D.-Newton's Method

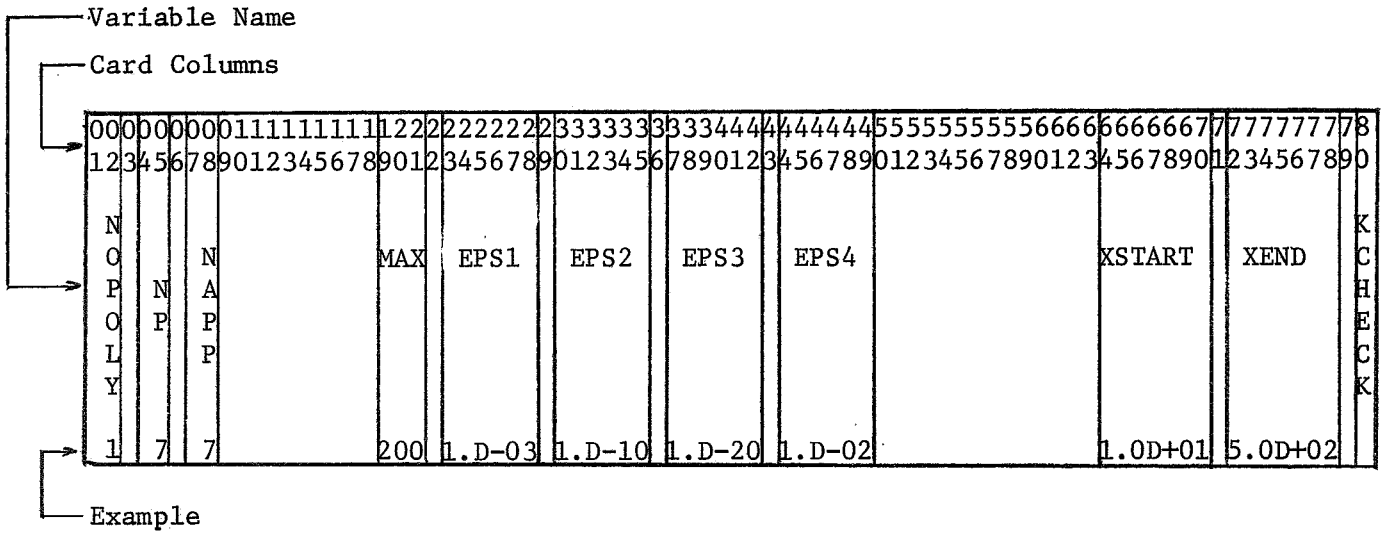


Figure E.2. Control Card for G.C.D. - Newton's Method

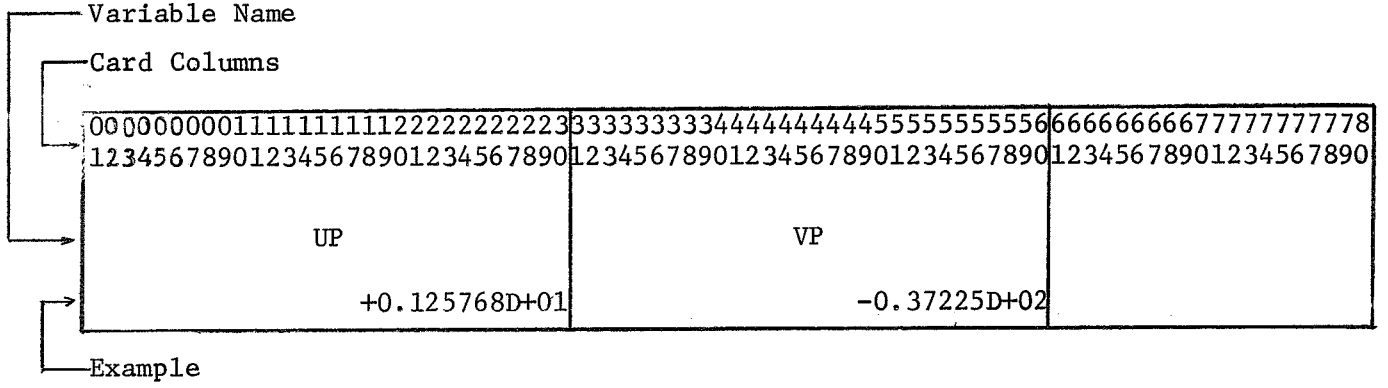


Figure E.3. Coefficient Card for G.C.D. - Newton's Method

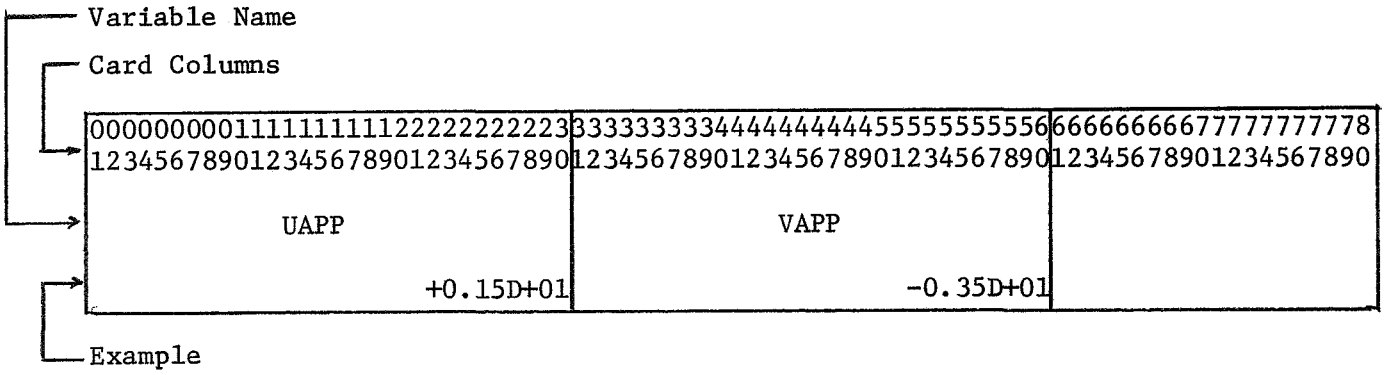


Figure E.4. Initial Approximation Card for G.C.D. - Newton's Method

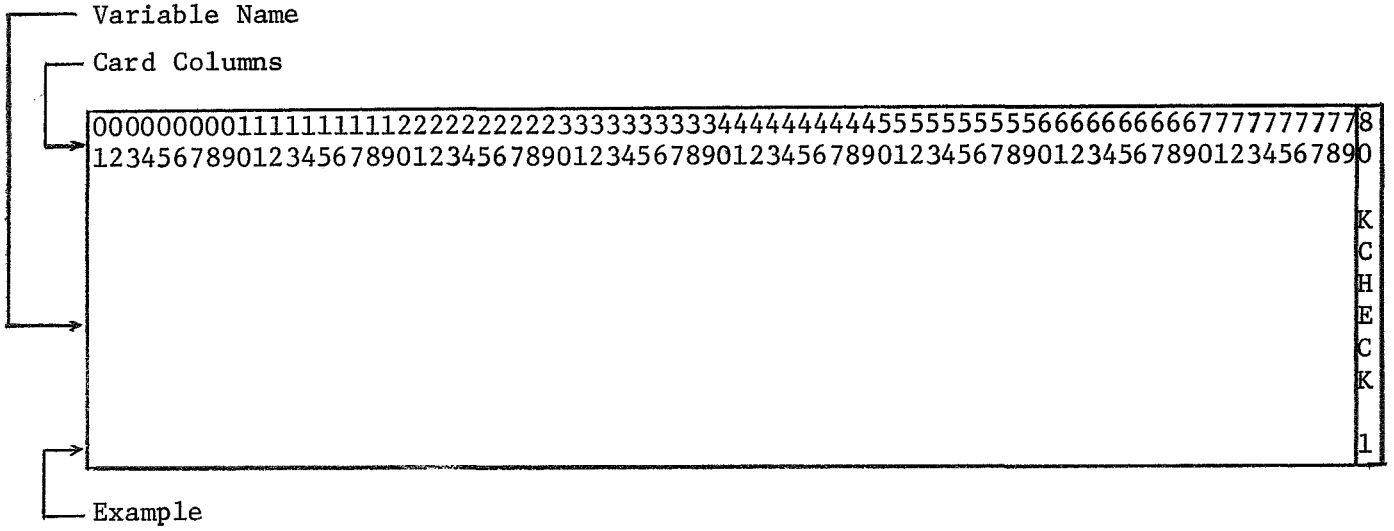


Figure E.5. End Card for G.C.D. - Newton's Method

TABLE E.VI

VARIABLES USED IN G.C.D. - NEWTON'S METHOD

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Main Program
J	I	J	I		Number of distinct roots found
ITIME	I	ITIME	I		Program control
NOPOLY	I	NOPOLY	I		Number of the polynomial
NP	I	NP	I		Degree of the original polynomial
P	C	UP,VP	D		Array of coefficients of original polynomial, P(X)
NAPP	I	NAPP	I		Number of initial approximation to be read
EPS1	R	EPS1	D		Tolerance check for zero (0) in Subroutine GCD
EPS2	R	EPS2	D		Tolerance check for convergence
EPS3	R	EPS3	D		Tolerance check for zero (0) in Subroutine QUAD
MAX	I	MAX	I		Maximum number of iterations permitted
I01	I	I01	I		Unit number of input device
I02	I	I02	I		Unit number of output device
KCHECK	I	KCHECK	I		Program control, KCHECK = 1 implies stop execution
APP	C	UAPP,VAPP	D		Array of initial approximations
XSTART	R	XSTART	D		Magnitude at which to start search for roots
XEND	R	XEND	D		Magnitude at which to end search for roots
ANAME	A	ANAME	A		Contains name of method used "NEWTONS"
ROOT	C	UROOT,VROOT	D		Array of roots found
MULT	I	MULT	I		Array of multiplicities
DP	C	UDP,VDP	D		Array containing coefficients of the derivative, (P'(X)), of P(X)
NDP	I	NDP	I		Degree of the derivative of original polynomial
D	C	UD,VD	D		Array of coefficients of the greatest common divisor of P(X) and P'(X)
ND	I	ND	I		Degree of g.c.d. of P(X) and P'(X)
Q	C	UQ,VQ	D		Array of coefficients of quotient polynomial P(X)/g.c.d.

TABLE E.VI (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
NQ	I	NQ	I		Degree of quotient polynomial Q(X)
ZRO	C	UZRO,VZRO	D		Value at which to evaluate or deflate polynomial
DUMMY	C	UDUMMY,VDUMMY	D		Dummy variable
QQ	C	UQQ,VQQ	D		Working array of coefficients of current polynomial
NQQ	I	NQQ	I		Degree of current polynomial, QQ(X)
IALTER	I	IALTER	I		Number of alterations of an initial approximation
CONV	L	CONV	L		CONV = TRUE implies convergence to a root
EPS4	R	EPS4	D		Tolerance for checking multiplicities
AP	C	UAP,VAP	D		Array of approximations (initial or altered) producing convergence
QD	C	UQD,VQD	D		Array of coefficients of newly deflated polynomial
JAP	I	JAP	I		Number of distinct roots found by iterative process i.e. not as a result of Subroutine QUAD
J1	I	J1	I		Number of distinct roots found in the attempt to improve roots
ROOTS	C	UROOTS,VROOTS	D		Array of improved roots
NEWT	L	NEWT	L		Program control. NEWT = TRUE implies that Newton's method was used instead of Subroutine QUAD
				Subroutine NEWTON	
X	C	UX,VX	D	E	Starting approximation (initial or altered)
N	I	N	I	E	Degree of current polynomial
P	C	UP,VP	D	E	Array of coefficients of current polynomial
MAX	I	MAX	I	C	Maximum number of iterations
EPSLON	R	EPSLON	D	C	Tolerance for checking convergence
XO	C	UXO,VXO	D	R	Current approximation to root
B	C	UB,VB	D		Array of coefficients of newly deflated polynomial
DPXO	C	UDPXO,VDPXO	D		Derivative of the polynomial at XO

TABLE E.VI (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
DIFF	C	UDIFF,VDIFF	D		PX0/DPX0
PX0	C	UPX0,VPX0	D		Value of polynomial at X0
CONV	L	CONV	L	R	CONV = TRUE implies convergence to root
					Subroutine HORNER
X	C	UX,VX	D	E	Value at which to evaluate or deflate polynomial
N	I	N	I	E	Degree of polynomial
P	C	UP,VP	D		Array of coefficients of polynomial
C	C	UC,VC	D	R	Updated at each iteration to yield derivative of polynomial at X
B	C	UB,VB	D		Array of coefficients of newly deflated polynomial
					Subroutine QUAD
N	I	N	I	E	Degree of polynomial to be solved
A	C	UA,VA	D	E	Array of coefficients of polynomial to be solved
J	I	J	I	ECR	Number of distinct roots found of original polynomial (J = -1 implies original polynomial is of degree 2 or 1)
ROOT	C	UROOT,VROOT	D	ECR	Array of roots found
MULT	I	MULT	I	ECR	Array of multiplicities
DISC	C	UDISC,VDISC	D		Discriminate of quadratic
TEMP	C	UTEMP,VTEMP	D		$\sqrt{DISC}$
EPSLON	R	EPSLON	D	C	Tolerance for zero (0)
D	C	UD,VD	D		Twice leading coefficient of quadratic

TABLE E.VI (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Subroutine GCD
R	C	UR,VR	D	E	Array of coefficients of original polynomial, P(X)
S	C	US,VS	D	E	Array of coefficients of derivative polynomial, P'(X)
N	I	N	I	E	Degree of original polynomial, P(X)
M	I	M	I	E	Degree of derivative polynomial, P'(X)
RR	C	URR,VRR	D		Array of coefficients of dividend polynomial
SS	C	USS,VSS	D	R	Array of coefficients of divisor polynomial also array of coefficients of g.c.d. of P(X) and P'(X) when returned
N1	I	N1	I		Degree of dividend polynomial, RR(X)
M1	I	M1	I	R	Degree of divisor polynomial, SS(X), also degree of g.c.d. of P(X) and P'(X) when returned
D	C	UD,VD	D		Quotient RR_{N1+1}/SS_{M1+1}
T	C	UT,VT	D		Array of coefficients of difference polynomial(RR - D(SS))
K	I	K	I		Degree of difference polynomial T(X)
EPSLON	R	EPSLON	D	C	Tolerance check for zero (0)
					Subroutine MULTI
N	I	N	I	E	Degree of original polynomial, P(X)
P	C	UP,VP	D	E	Array of coefficients of original polynomial, P(X)
J	I	J	I	E	Number of distinct roots of P(X)
ROOT	C	UROOT,VROOT	D	E	Array of distinct roots of P(X)
A	C	UA,VA	D		Working array of coefficients of current polynomial
M	I	M	I		Degree of current polynomial, A(X)
MULT	I	MULT	I	R	Array of multiplicities of the roots
IO2	I	IO2	I	C	Unit number of output device
B	C	UB,VB	D		Array of coefficients of newly deflated polynomial
C	C	UC,VC	D		Derivative of polynomial at $ROOT_i$
EPSLON	R	EPSLON	D	C	Tolerance for checking multiplicities

TABLE E.VI (Continued)

Single Precision Variable	Type	Doubel Precision Variable	Type	Disposition of Argument	Description
					Subroutine DERIV
N	I	N	I	E	Degree of polynomial, P(X)
P	C	UP,VP	D	E	Array of coefficients of polynomial, P(X)
A	C	UA,VA	D	R	Array of coefficients of derivative, P'(X)
M	I	M	I	R	Degree of derivative polynomial, P'(X)
					Subroutine DIVIDE
P	C	UP,VP	D	E	Array of coefficients of dividend polynomial
N	I	N	I	E	Degree of dividend polynomial
D	C	UD,VD	D	E	Array of coefficients of divisor polynomial
M	I	M	I	E	Degree of divisor polynomial
Q	C	UQ,VQ	D	R	Array of coefficients of quotient polynomial P(X)/D(X)
K	I	K	I	R	Degree of quotient polynomial, Q(X)
J	I	J	I		Counter
TERM	C	UTERM,VTERM	D		Dummy variable used for temporary storage of products
KK	I	KK	I		Number of coefficients of quotient polynomial, Q(X)
					Subroutine GENAPP
APP	C	APPR,APPI	D	R	Array containing initial approximations
NAPP	I	NAPP	I	E	Number of initial approximations to be generated
XSTART	R	XSTART	D	ECR	Magnitude at which to begin generating approximations; also magnitude of the approximation being generated
BETA	R	BETA	D		Argument of complex approximation being generated
U	R	APPR(I)	D		Real part of complex approximation
V	R	APPI(I)	D		Imaginary part of complex approximation

TABLE E.VI (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Subroutine ALTER
XOLD	C	XOLDR,XOLDI	D	ECR	Old approximation to be altered to new approximation
NALTER	I	NALTER	I	ECR	Number of alterations performed on an initial approximation
ITIME	I	ITIME	I	E	Program control
MAX	I	MAX	I	C	Maximum number of iterations permitted
Y	R	XOLDI	D		Imaginary part of original initial approximation (unaltered)
X	R	XOLDR	D		Real part of original, unaltered initial approximation
R	R	ABXOLD	D		Magnitude of original unaltered initial approximation
BETA	R	BETA	D		Argument of new approximation
XOLDR	R	XOLDR	D		Real part of new approximation
XOLDI	R	XOLDI	D		Imaginary part of new approximation
I02	I	I02	I	C	Unit number of output device
					Subroutine COMSQT
		UX,VX	D	E	Complex number for which the square root is desired
		UY,VY	D	R	Square root of the complex number

4. Description of Program Output

The output from G.C.D. - Newton's method consists of the following information.

The heading is "GREATEST COMMON DIVISOR METHOD USED WITH NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS NUMBER XX." XX represents the number of the polynomial.

As an aid to ensure that the control information is correct, the number of initial approximations given, maximum number of iterations, test for zero in subroutine GCD, test for convergence, test for zero in subroutine QUAD, test for multiplicities, radius to start search, and radius to end search are printed as read from the control card.

The coefficients of the polynomial are printed under the heading "THE DEGREE OF P(X) IS XX THE COEFFICIENTS ARE." XX represents the degree of the polynomial. The coefficient of the highest degree term is printed first.

The polynomial obtained after dividing the original polynomial, P(X), by the greatest common divisor of P(X) and its derivative, P'(X), is printed under the heading "Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X). THE DEGREE OF Q(X) IS XX THE COEFFICIENTS ARE." XX represents the degree of this polynomial. This polynomial contains all distinct roots and is solved by Newton's method. The coefficient of the highest degree term is printed first; that is, the leading coefficient is printed first.

The zeros found before the attempt to improve accuracy are printed under the heading "ROOTS OF Q(X)."

The initial approximation producing convergence to a root is

printed to the right of the corresponding root and headed by "INITIAL APPROXIMATION." The initial approximations may be those supplied by the user, or generated by the program or a combination of both. The message "RESULTS OF SUBROUTINE QUAD" indicates that the corresponding root was obtained by subroutine QUAD. See Appendix D, § 5.

The zeros found after the attempt to improve accuracy are printed under the heading "ROOTS OF P(X)." The corresponding initial approximation producing convergence is printed as described above.

The multiplicity of each zero is given under the title "MULTIPLICITIES."

5. Informative Messages and Error Messages

The output may contain informative or error messages. These are intended as an aid to the user and are described as follows.

If not all roots of a polynomial were found before the attempt to improve accuracy, the remaining unsolved polynomial will be printed, with the leading coefficient first, under the heading "COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." See Appendix D, § 6.

"NO ROOTS FOR INITIAL APPROXIMATION ROOT XX = YYY." This message is printed if a root fails to produce convergence when trying to improve accuracy. XX represents the number of the root and YYY represents the value of the root before the attempt to improve accuracy.

"NO ROOTS FOR THE POLYNOMIAL Q(X) OF DEGREE XX WITH GENERATED INITIAL APPROXIMATIONS." XX represents the degree of the polynomial Q(X). This message is printed if none of the roots produce convergence in the attempt to improve accuracy.

"THE EPSILON (XXX) CHECK IN SUBROUTINE MULTI INDICATES THAT ROOT YY = ZZZ IS NOT CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIPLICITY 0." XXX represents the multiplicity requirement (EPS4 on the control card), YY represents the number of the root, and ZZZ represents the value of the root after the attempt to improve accuracy. The message indicates that this root does not meet the requirement for multiplicities. It is, however, usually a good approximation to the true root since convergence was obtained both before and after the attempt to improve accuracy.

MAIN PROGRAM

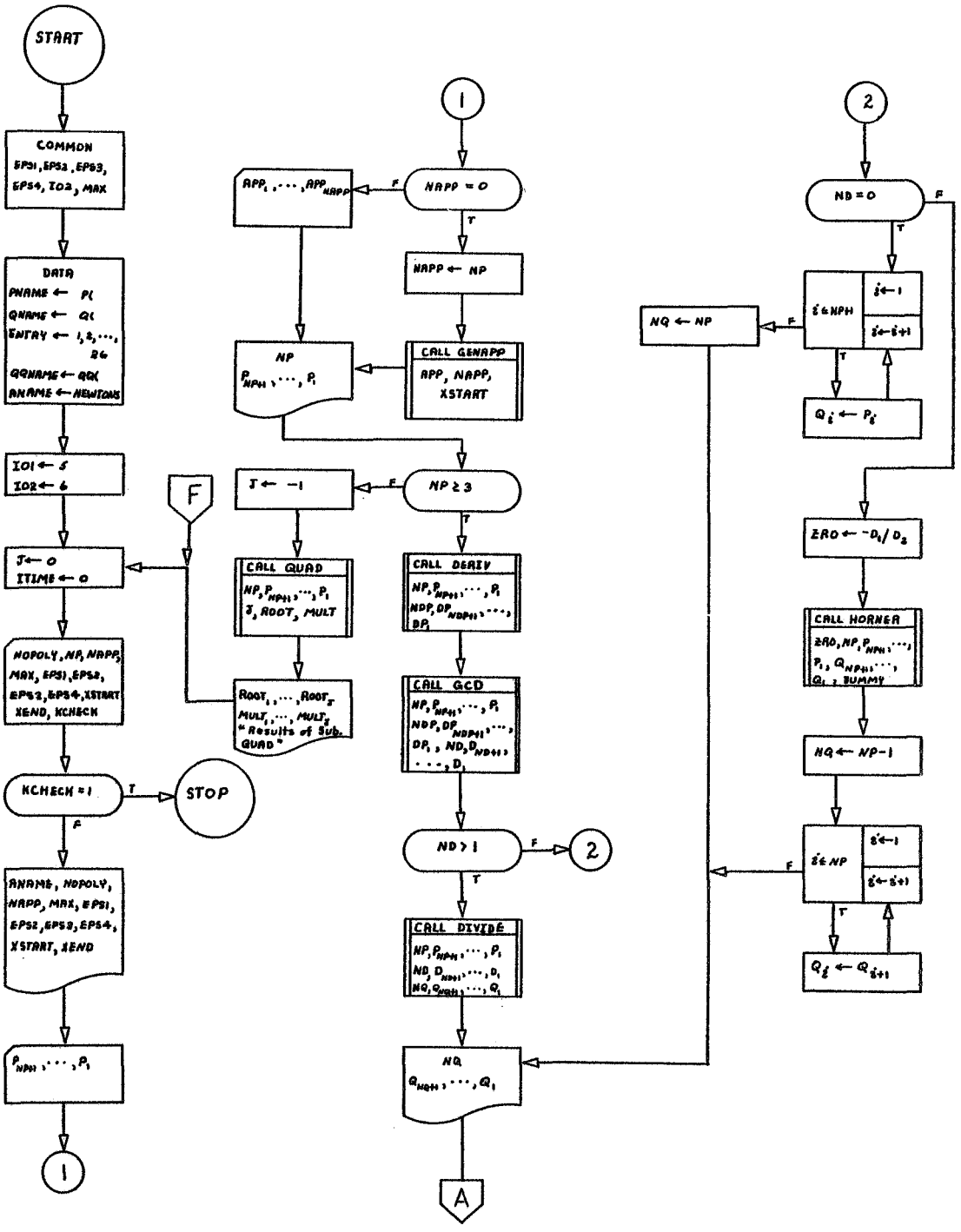


Figure E.6. Flow Charts for G.C.D.-Newton's Method

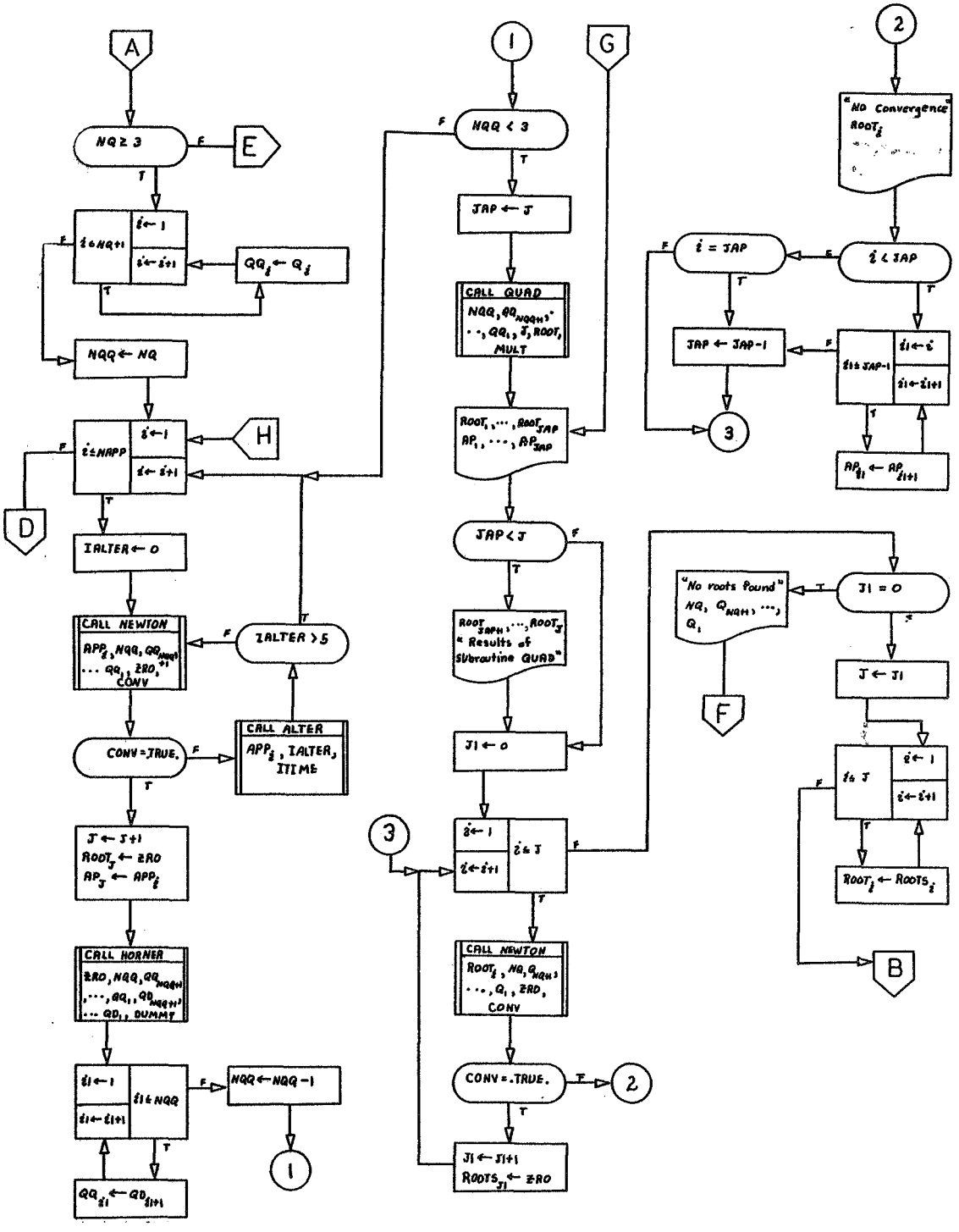


Figure E.6. (Continued)

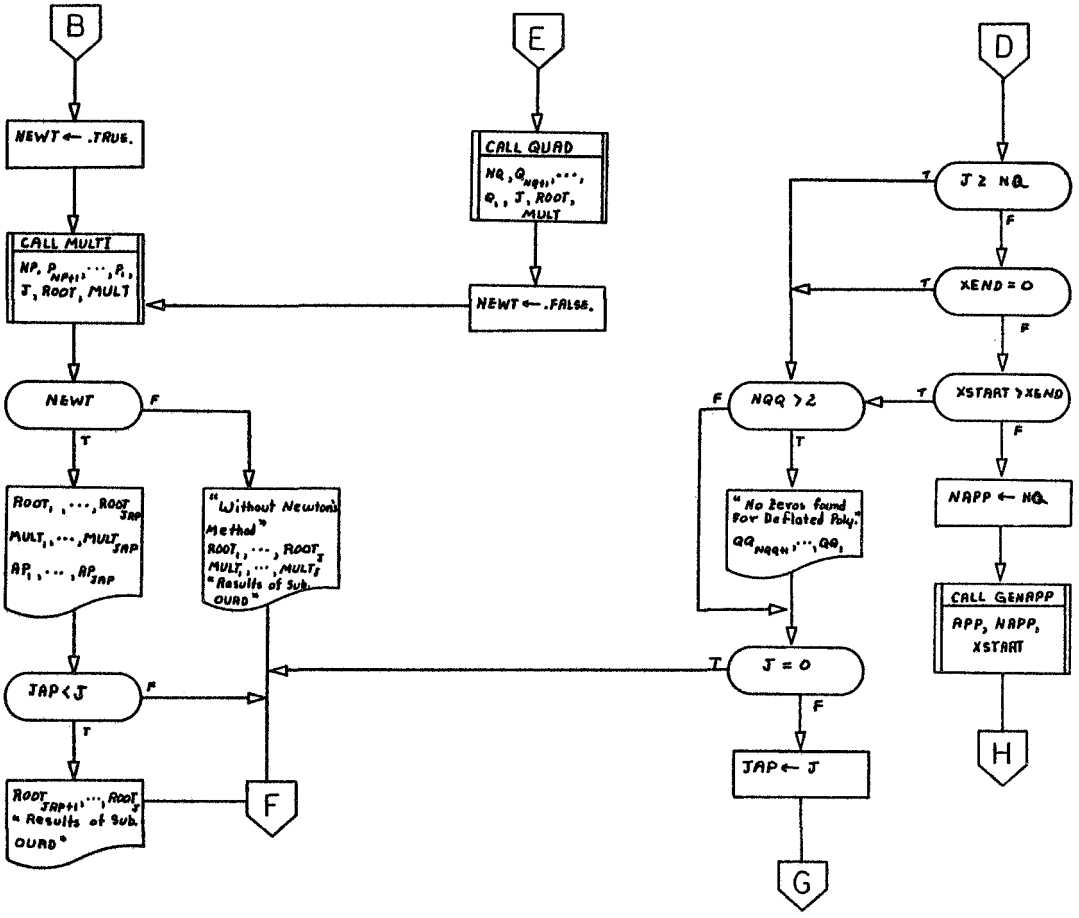


Figure E.6. (Continued)

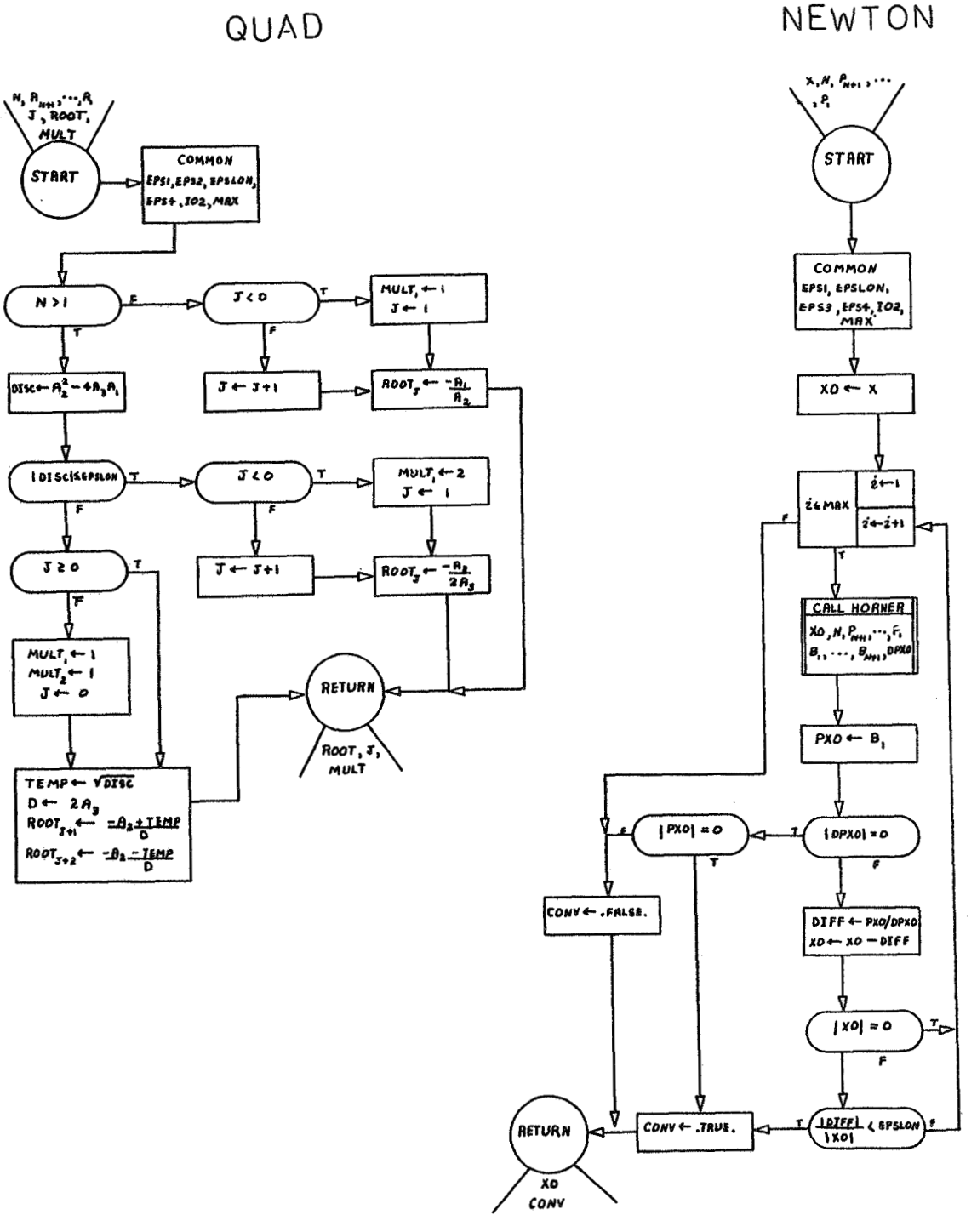


Figure E.6. (Continued)

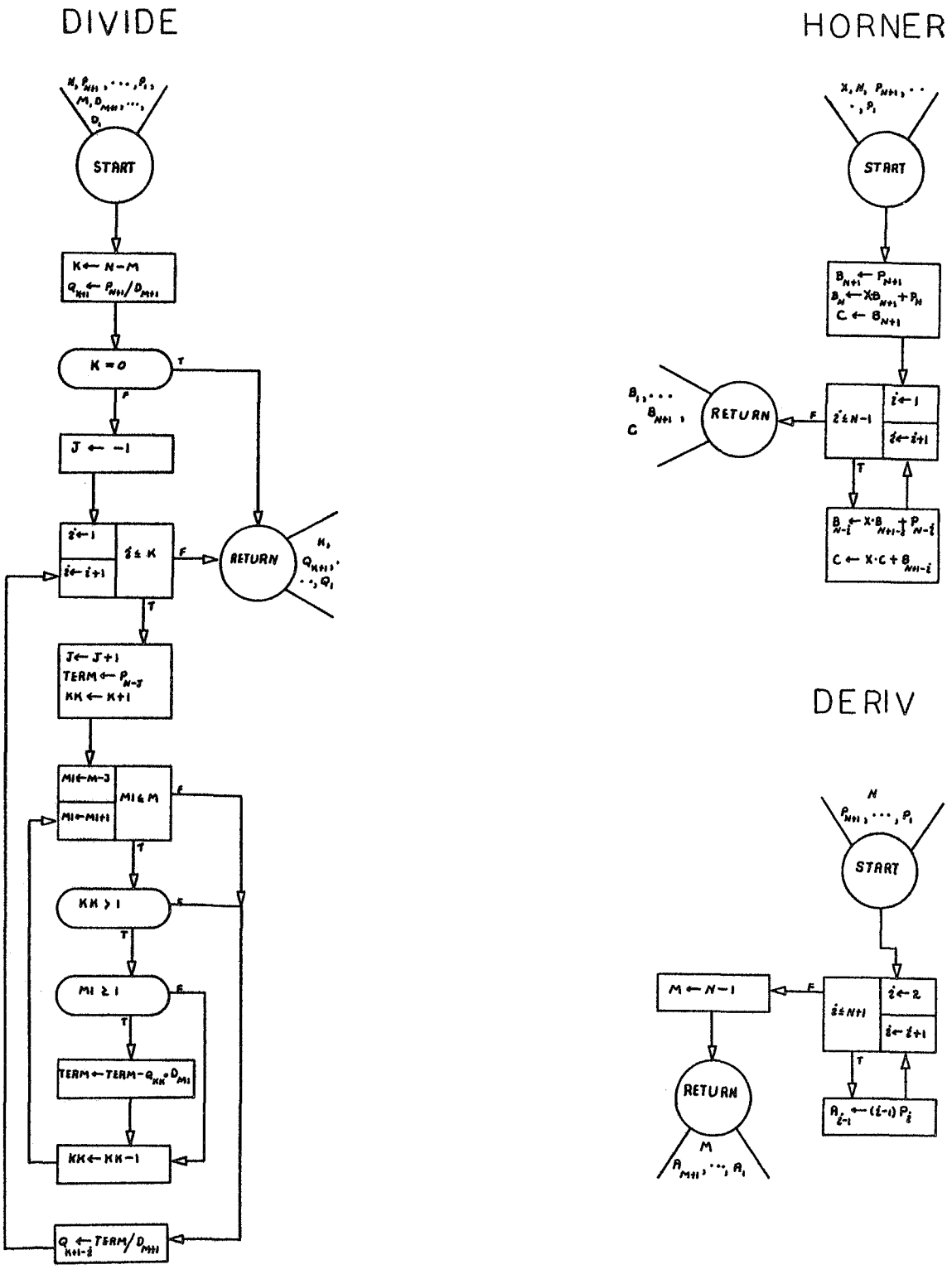


Figure E.6. (Continued)

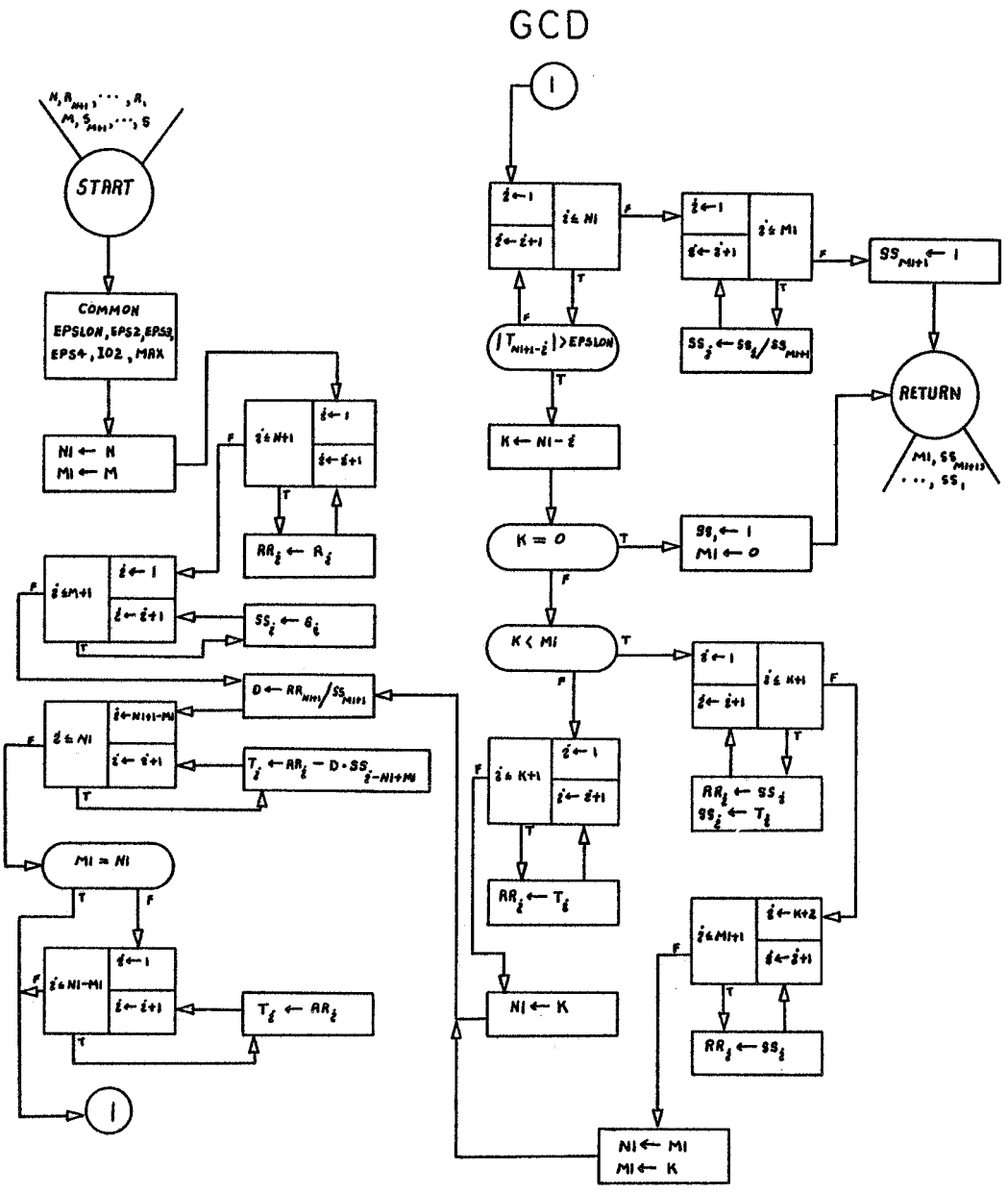


Figure E.6. (Continued)

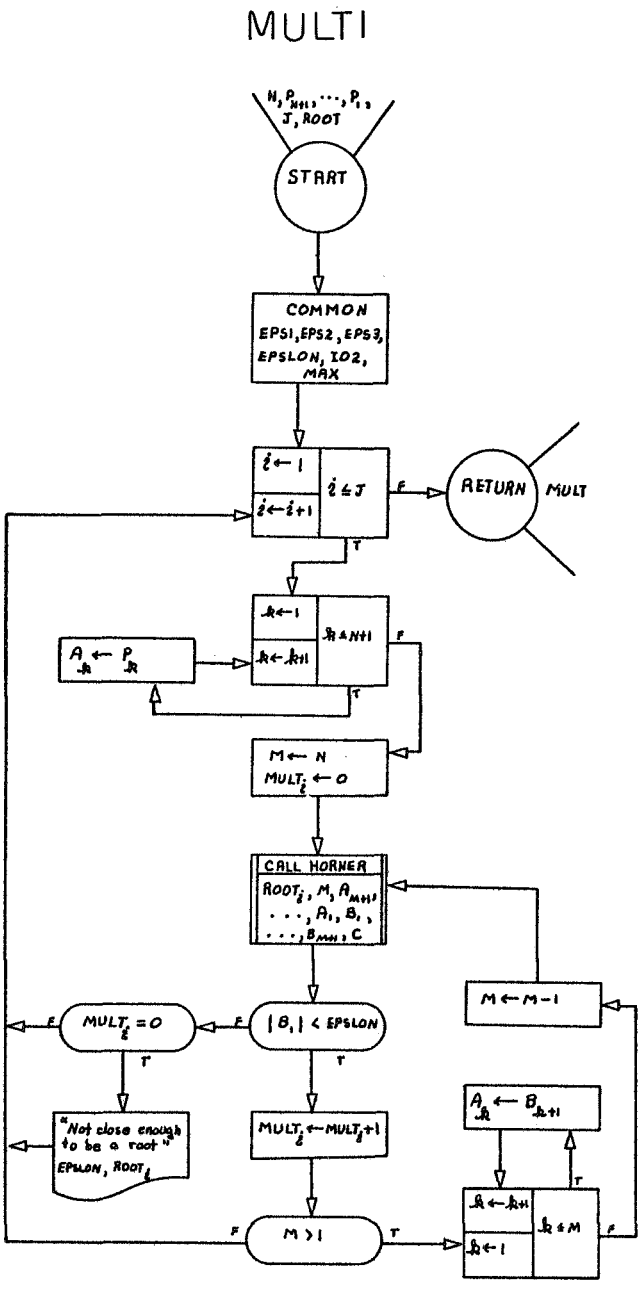


Figure E.6. (Continued)

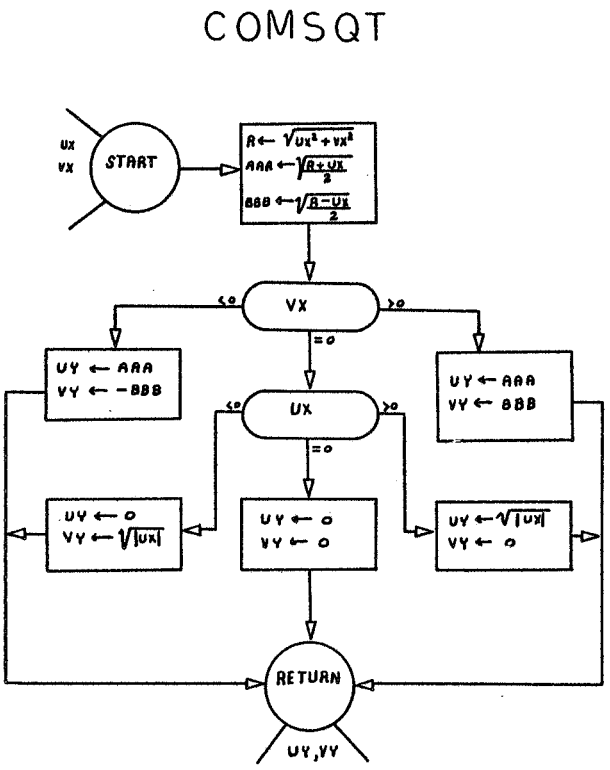


Figure E.6. (Continued)

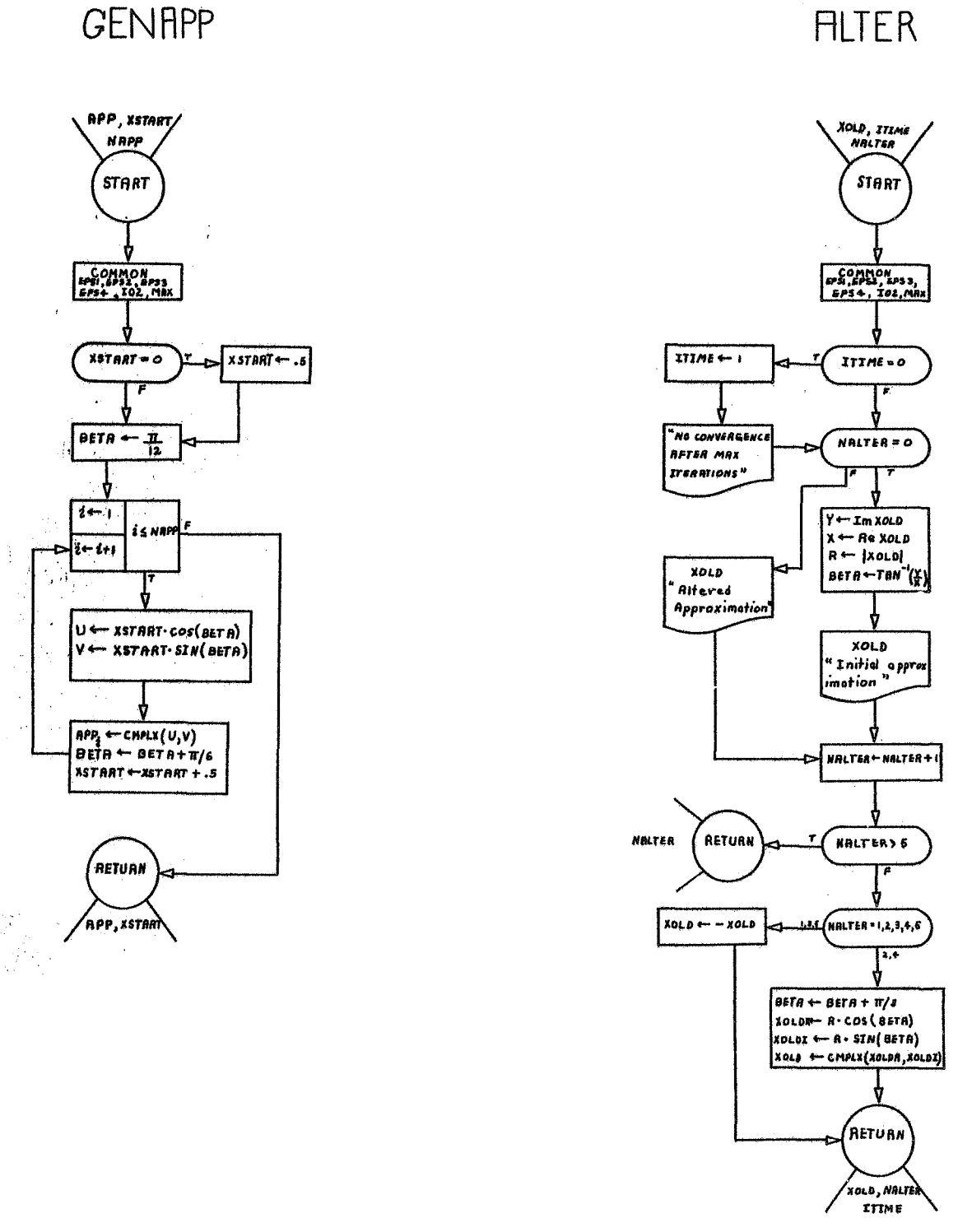


Figure E.6. (Continued)

TABLE E.VII

PROGRAM FOR G.C.D.-NEWTON'S METHOD

```
C   ************************************************************************
C   *                                                                      *
C   * DOUBLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD                *
C   *                                                                      *
C   *                                                                      *
C   * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A   *
C   * POLYNOMIAL OF MAXIMUM DEGREE 25.  ALL MULTIPLE ROOTS ARE REMOVED BY  *
C   * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL *
C   * AND ITS DERIVATIVE.  THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED *
C   * AND THEIR MULTIPLICITIES DETERMINED.                                 *
C   *                                                                      *
C   ************************************************************************
0001          DOUBLE PRECISION UP,VP,UAPP,VAPP,UROOT,VROOT,UDP,VDP,UD,VD,UZRO,VZ
             1RO,UQ,VQ,UDUMMY,VDUMMY,UQQ,VQQ,UAP,VAP,UQD,VQD,UROOTS,VROOTS,EPS1,
             2EPS2,EPS3,EPS4
0002          DOUBLE PRECISION XSTART
0003          DOUBLE PRECISION XEND
0004          DIMENSION UP(26),VP(26),UAPP(25),VAPP(25),UROOT(25),VROOT(25),MULT
             1(25),UDP(26),VDP(26),UD(26),VD(26),UQ(26),VQ(26),UQQ(26),VQQ(26),U
             2AP(25),VAP(25),UQD(26),VQD(26),ANAME(2),ENTRY(26),UROOTS(25),VROOT
             3S(25)
0005          COMMON EPS1,EPS2,EPS3,EPS4,IO2,MAX
0006          LOGICAL NEWT,CONV
0007          DATA PNAME,QNAME,QQNAME/2HP(,2HQ(,3HQQ(/
0008          DATA ENTRY/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,2H12,2H13
             1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
0009          DATA ANAME(1),ANAME(2)/4HNEWT,4HONS /
0010          IO1=5
0011          IO2=6
0012       10 J=0
0013          ITIME=0
0014          READ(IO1,1000) NOPOLY,NP,NAPP,MAX,EPS1,EPS2,EPS3,EPS4,XSTART,XEND,
             1KCHECK
0015          IF(KCHECK.EQ.1) STOP
0016          WRITE(IO2,1020) ANAME(1),ANAME(2),NOPOLY
0017          WRITE(IO2,2000) NAPP
0018          WRITE(IO2,2010) MAX
0019          WRITE(IO2,2070) EPS1
0020          WRITE(IO2,2020) EPS2
0021          WRITE(IO2,2080) EPS3
0022          WRITE(IO2,2030) EPS4
0023          WRITE(IO2,2040) XSTART
0024          WRITE(IO2,2050) XEND
0025          WRITE(IO2,2060)
0026          KKK=NP+1
0027          NNN=KKK+1
0028          DO 20 I=1,KKK
0029          JJJ=NNN-I
0030       20 READ(IO1,1010) UP(JJJ),VP(JJJ)
0031          IF(NAPP.NE.0) GO TO 22
0032          NAPP=NP
0033          CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0034          GO TO 23
0035       22 READ(IO1,1015) (UAPP(I),VAPP(I),I=1,NAPP)
0036       23 WRITE(IO2,1030) NP
0037          KKK=NP+1
0038          NNN=KKK+1
0039          DO 25 I=1,KKK
```

TABLE E.VII (Continued)

```
0040            JJJ=NNN-I
0041         25 WRITE(IO2,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
0042            IF(NP.GE.3) GO TO 30
0043            J=-1
0044            CALL QUAD(NP,UP,VP,J,UROOT,VROOT,MULT)
0045            WRITE(IO2,1070)
0046            WRITE(IO2,1165) (I,UROOT(I),VROOT(I),MULT(I),I=1,J)
0047            GO TO 10
0048         30 CALL DERIV(NP,UP,VP,NDP,UDP,VDP)
0049            CALL GCD(NP,UP,VP,NDP,UDP,VDP,ND,UD,VD)
0050            IF(ND.GT.1) GO TO 70
0051            IF(ND.EQ.0) GO TO 65
0052            UDUMMY=(UD(2)*UD(2))+(VD(2)*VD(2))
0053            UZRO=(-(UD(1)*UD(2))-(VD(1)*VD(2)))/UDUMMY
0054            VZRO=(-(UD(2)*VD(1))+(UD(1)*VD(2)))/UDUMMY
0055            CALL HORNER(UZRO,VZRO,NP,UP,VP,UQ,VQ,UDUMMY,VDUMMY)
0056            NQ=NP-1
0057            DO 60 I=1,NP
0058            UQ(I)=UQ(I+1)
0059         60 VQ(I)=VQ(I+1)
0060            GO TO 80
0061         65 KKK=NP+1
0062            DO 66 I=1,KKK
0063            UQ(I)=UP(I)
0064         66 VQ(I)=VP(I)
0065            NQ=NP
0066            GO TO 80
0067         70 CALL DIVIDE(NP,UP,VP,ND,UD,VD,NQ,UQ,VQ)
0068         80 WRITE(IO2,1120) NQ
0069            KKK=NQ+1
0070            NNN=KKK+1
0071            DO 83 I=1,KKK
0072            JJJ=NNN-I
0073         83 WRITE(IO2,1040) QNAME,ENTRY(JJJ),UQ(JJJ),VQ(JJJ)
0074            IF(NQ.GE.3) GO TO 85
0075            GO TO 110
0076         85 KKK=NQ+1
0077            DO 90 I=1,KKK
0078            UQQ(I)=UQ(I)
0079         90 VQQ(I)=VQ(I)
0080            NQQ=NQ
0081            GO TO 120
0082        110 CALL QUAD(NQ,UQ,VQ,J,UROOT,VROOT,MULT)
0083            NEWT=.FALSE.
0084            GO TO 310
0085        120 DO 200 I=1,NAPP
0086            IALTER=0
0087        130 CALL NEWTON(UAPP(I),VAPP(I),NQQ,UQQ,VQQ,UZRO,VZRO,CONV)
0088            IF(CONV) GO TO 160
0089            CALL ALTER(UAPP(I),VAPP(I),IALTER,ITIME)
0090            IF(IALTER.GT.5) GO TO 200
0091            GO TO 130
0092        160 J=J+1
0093            UROOT(J)=UZRO
0094            VROOT(J)=VZRO
0095            UAP(J)=UAPP(I)
0096            VAP(J)=VAPP(I)
0097            CALL HORNER(UZRO,VZRO,NQQ,UQQ,VQQ,UQD,VQD,UDUMMY,VDUMMY)
```

TABLE E.VII (Continued)

```
0098              DO 180 I1=1,NQQ
0099              UQQ(I1)=UQD(I1+1)
0100          180 VQQ(I1)=VQD(I1+1)
0101              NQQ=NQQ-1
0102              IF(NQQ.LT.3) GO TO 220
0103          200 CONTINUE
0104              IF(J.GE.NQ) GO TO 205
0105              IF(XEND.EQ.0.0) GO TO 205
0106              IF(XSTART.GT.XEND) GO TO 205
0107              NAPP=NQ
0108              CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0109              GO TO 120
0110          205 IF(NQQ.LE.2) GO TO 210
0111              WRITE(IO2,1200)
0112              KKK=NQQ+1
0113              NNN=KKK+1
0114              DO 157 L=1,KKK
0115              JJJ=NNN-L
0116          157 WRITE(IO2,1100) QQNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0117          210 IF(J.EQ.0) GO TO 10
0118              JAP=J
0119              GO TO 230
0120          220 JAP=J
0121              CALL QUAD(NQQ,UQQ,VQQ,J,UROOT,VROOT,MULT)
0122          230 WRITE(IO2,1132)
0123              WRITE(IO2,1133) (I,UROOT(I),VROOT(I),UAP(I),VAP(I),I=1,JAP)
0124              IF(JAP.LT.J) GO TO 235
0125              GO TO 240
0126          235 KKK=JAP+1
0127              WRITE(IO2,1134) (I,UROOT(I),VROOT(I),I=KKK,J)
0128          240 J1=0
0129              DO 300 I=1,J
0130              CALL NEWTON(UROOT(I),VROOT(I),NQ,UQ,VQ,UZRO,VZRO,CONV)
0131              IF(CONV) GO TO 280
0132              WRITE(IO2,1140) I,UROOT(I),VROOT(I)
0133              IF(I.LT.JAP) GO TO 241
0134              IF(I.EQ.JAP) GO TO 250
0135              GO TO 300
0136          241 KKK=JAP-1
0137              DO 245 I1=I,KKK
0138              UAP(I1)=UAP(I1+1)
0139          245 VAP(I1)=VAP(I1+1)
0140          250 JAP=JAP-1
0141              GO TO 300
0142          280 J1=J1+1
0143              UROOTS(J1)=UZRO
0144              VROOTS(J1)=VZRO
0145          300 CONTINUE
0146              IF(J1.EQ.0) GO TO 305
0147              J=J1
0148              DO 303 I=1,J
0149              UROOT(I)=UROOTS(I)
0150          303 VROOT(I)=VROOTS(I)
0151              GO TO 307
0152          305 WRITE(IO2,1150) NQ
0153              KKK=NQ+1
0154              NNN=KKK+1
0155              DO 306 L=1,KKK
```

TABLE E.VII (Continued)

```
0156            JJJ=NNN-L
0157          306 WRITE(IO2,1040) QNAME,ENTRY(JJJ),UQ(JJJ),VQ(JJJ)
0158            GO TO 10
0159          307 NEWT=.TRUE.
0160          310 CALL MULTI(NP,UP,VP,J,UROOT,VROOT,MULT)
0161            IF(NEWT) GO TO 330
0162            WRITE(IO2,1070)
0163            WRITE(IO2,1165) (L,UROOT(L),VROOT(L),MULT(L),L=1,J)
0164            GO TO 10
0165          330 WRITE(IO2,1180)
0166            WRITE(IO2,1190) (L,UROOT(L),VROOT(L),MULT(L),UAP(L),VAP(L),L=1,JAP
              1)
0167            KKK=JAP+1
0168            IF(JAP.LT.J) WRITE(IO2,1165) (L,UROOT(L),VROOT(L),MULT(L),L=KKK,J)
0169            GO TO 10
0170         1000 FORMAT(3(I2,1X),9X,I3,1X,4(D6.0,1X),13X,2(D7.0,1X),I1)
0171         1010 FORMAT(2D30.0)
0172         1015 FORMAT(2D30.0)
0173         1020 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
                 135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11X,18HPOLYNOMIAL NUMBER ,I
                 22///)
0174         1030 FORMAT(1X,22HTHE DEGREE OF P(X) IS ,I2,22H  THE COEFFICIENTS ARE//
                 1)
0175         1040 FORMAT(2X,A2,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
0176         1070 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES//)
0177         1080 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,10X,I2)
0178         1100 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
0179         1120 FORMAT(///1X,73HQ(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE
                 1DISTINCT ROOTS OF P(X)./1X,22HTHE DEGREE OF Q(X) IS ,I2,22H  THE C
                 2OEFFICIENTS ARE//)
0180         1200 FORMAT(///1X,70HCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH
                 1NO ZEROS WERE FOUND.//)
0181         1132 FORMAT(///1X,13HROOTS OF Q(X),84X,21HINITIAL APPROXIMATION//)
0182         1133 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,17X,D23.16,3H
                 1 + ,D23.16,2H I)
0183         1134 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,20X,26HRESULT
                 1S OF SUBROUTINE QUAD)
0184         1140 FORMAT(///,1X,40HNO ROOTS FOR INITIAL APPROXIMATION ROOT(,I2,4H) =
                 1 ,D23.16,3H + ,D23.16,2H I)
0185         1150 FORMAT(///,1X,45HNO ROOTS FOR THE POLYNOMIAL Q(X) OF DEGREE = ,I2,
                 138H WITH GENERATED INITIAL APPROXIMATIONS//)
0186         1165 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,10X,26H
                 1RESULTS OF SUBROUTINE QUAD)
0187         1180 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
                 1 APPROXIMATION//)
0188         1190 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,7X,D23.
                 116,3H + ,D23.16,2H I)
0189         2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
0190         2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
0191         2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
0192         2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
0193         2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
0194         2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
0195         2060 FORMAT(//1X)
0196         2070 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE GCD.  ,D9.2)
0197         2080 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE QUAD. ,D9.2)
0198            END
```

TABLE E.VII (Continued)

```
0001              SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
                C ***************************************************************************
                C *                                                                         *
                C * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE     *
                C * DEGREE OF THE ORIGINAL POLYNOMIAL.                                      *
                C *                                                                         *
                C ***************************************************************************
0002              DOUBLE PRECISION APPR,APPI,XSTART,BETA,    EPS1,EPS2,EPS3,EPS4
0003              DIMENSION APPR(25),APPI(25)
0004              COMMON EPS1,EPS2,EPS3,EPS4,IO2,MAX
0005              IF(XSTART.EQ.0.0) XSTART=0.5
0006              BETA=0.2617994
0007              DO 10 I=1,NAPP
0008              APPR(I)=XSTART*DCOS(BETA)
0009              APPI(I)=XSTART*DSIN(BETA)
0010              BETA=BETA+0.5235988
0011           10 XSTART=XSTART+0.5
0012              RETURN
0013              END
```

TABLE E.VII (Continued)

```
0001              SUBROUTINE ALTER(XOLDR,XOLDI,NALTER,ITIME)
                C ********************************************************************************
                C *                                                                              *
                C * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO          *
                C * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.      *
                C *                                                                              *
                C ********************************************************************************
0002              DOUBLE PRECISION XOLDR,XOLDI,ABXOLD,BETA,EPS1,EPS2,EPS3,EPS4
0003              COMMON EPS1,EPS2,EPS3,EPS4,IO2,MAX
0004              IF(ITIME.NE.0) GO TO 5
0005              ITIME =1
0006              WRITE(IO2,1010) MAX
0007            5 IF(NALTER.EQ.0) GO TO 10
0008              WRITE(IO2,1000) XOLDR,XOLDI
0009              GO TO 20
0010           10 ABXOLD=DSQRT((XOLDR*XOLDR)+(XOLDI*XOLDI))
0011              BETA=DATAN2(XOLDI,XOLDR)
0012              WRITE(IO2,1020) XOLDR,XOLDI
0013           20 NALTER=NALTER+1
0014              IF(NALTER.GT.5) RETURN
0015              GO TO (30,40,30,40,30),NALTER
0016           30 XOLDR=-XOLDR
0017              XOLDI=-XOLDI
0018              GO TO 50
0019           40 BETA=BETA+1.0471976
0020              XOLDR=ABXOLD*DCOS(BETA)
0021              XOLDI=ABXOLD*DSIN(BETA)
0022           50 RETURN
0023         1000 FORMAT(1X,D23.16,3H + ,D23.16,2H I,10X,21HALTERED APPROXIMATION)
0024         1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
                    1TER ,I3,12H ITERATIONS.//)
0025         1020 FORMAT(/1X,D23.16,3H + ,D23.16,2H I,10X,21HINITIAL APPROXIMATION)
0026              END
```

TABLE E.VII (Continued)

```
0001            SUBROUTINE GCD(N,UR,VR,M,US,VS,M1,USS,VSS)
                C   ************************************************************************
                C   *                                                                      *
                C   * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.  *
                C   * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND *
                C   * DP(X).                                                               *
                C   *                                                                      *
                C   ************************************************************************
0002            DOUBLE PRECISION USSSSS,VSSSSS
0003            DOUBLE PRECISION UR,VR,US,VS,USS,VSS,URR,VRR,UD,VD,UT,VT,EPSLON,EP
                1S2,EPS3,EPS4,BBB
0004            DIMENSION UR(26),VR(26),US(26),VS(26),USS(26),VSS(26),URR(26),VRR(
                126),UT(26),VT(26)
0005            COMMON EPSLON,EPS2,EPS3,EPS4,IO2,MAX
0006            N1=N
0007            M1=M
0008            KKK=N+1
0009            DO 20 I=1,KKK
0010            URR(I)=UR(I)
0011         20 VRR(I)=VR(I)
0012            KKK=M+1
0013            DO 25 I=1,KKK
0014            USS(I)=US(I)
0015         25 VSS(I)=VS(I)
0016         30 BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0017            UD=(URR(N1+1)*USS(M1+1)+VRR(N1+1)*VSS(M1+1))/BBB
0018            VD=(USS(M1+1)*VRR(N1+1)-URR(N1+1)*VSS(M1+1))/BBB
0019            KKK=N1+1-M1
0020            DO 40 I=KKK,N1
0021            UT(I)=URR(I)-(UD*USS(I-N1+M1)-VD*VSS(I-N1+M1))
0022         40 VT(I)=VRR(I)-(UD*VSS(I-N1+M1)+VD*USS(I-N1+M1))
0023            IF(M1.EQ.N1) GO TO 70
0024            KKK=N1-M1
0025            DO 60 I=1,KKK
0026            UT(I)=URR(I)
0027         60 VT(I)=VRR(I)
0028         70 DO 90 I=1,N1
0029            BBB=DSQRT(UT(N1+1-I)*UT(N1+1-I)+VT(N1+1-I)*VT(N1+1-I))
0030            IF(BBB.GT.EPSLON) GO TO 100
0031         90 CONTINUE
0032            DO 95 I=1,M1
0033            BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0034            USSSSS=(USS(I)*USS(M1+1)+VSS(I)*VSS(M1+1))/BBB
0035            VSSSSS=(VSS(I)*USS(M1+1)-USS(I)*VSS(M1+1))/BBB
0036            USS(I)=USSSSS
0037         95 VSS(I)=VSSSSS
0038            USS(M1+1)=1.0
0039            VSS(N1+1)=0.0
0040            GO TO 200
0041        100 K=N1-I
0042            IF(K.EQ.0) GO TO 170
0043            IF(K.LT.M1) GO TO 140
0044            KKK=K+1
0045            DO 130 J=1,KKK
0046            URR(J)=UT(J)
0047        130 VRR(J)=VT(J)
0048            N1=K
0049            GO TO 30
```

TABLE E.VII (Continued)

```
0050          140 KKK=K+1
0051              DO 150 J=1,KKK
0052              URR(J)=USS(J)
0053              VRR(J)=VSS(J)
0054              USS(J)=UT(J)
0055          150 VSS(J)=VT(J)
0056              KKK=K+2
0057              NNN=M1+1
0058              DO 160 J=KKK,NNN
0059              URR(J)=USS(J)
0060          160 VRR(J)=VSS(J)
0061              N1=M1
0062              M1=K
0063              GO TO 30
0064          170 USS(1)=1.0
0065              VSS(1)=0.0
0066              M1=0
0067          200 RETURN
0068              END
```

TABLE E.VII (Continued)

```
0001            SUBROUTINE QUAD(N,UA,VA,J,UROOT,VROOT,MULT)
              C ********************************************************************************
              C *                                                                              *
              C * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES       *
              C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE        *
              C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                               *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UA,VA,UROOT,VROOT,UDISC,VDISC,UTEMP,VTEMP,UD,VD,E
               1PS1,EPS2,EPS4,EPSLON,BBB
0003            DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULT(25)
0004            COMMON EPS1,EPS2,EPSLON,EPS4,IO2,MAX
0005            IF(N.GT.1) GO TO 60
0006            IF(J.LT.0) GO TO 40
0007            J=J+1
0008            GO TO 50
0009         40 MULT(1)=1
0010            J=1
0011         50 BBB=UA(2)*UA(2)+VA(2)*VA(2)
0012            UROOT(J)=-(UA(1)*UA(2)+VA(1)*VA(2))/BBB
0013            VROOT(J)=-(VA(1)*UA(2)-UA(1)*VA(2))/BBB
0014            GO TO 200
0015         60 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(3)*UA(1)-VA(3)*VA(1)))
0016            VDISC=(2.0*UA(2)*VA(2))-(4.0*(UA(3)*VA(1)+VA(3)*UA(1)))
0017            BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
0018            IF(BBB.LE.EPSLON) GO TO 100
0019            IF(J.GE.0) GO TO 80
0020            MULT(1)=1
0021            MULT(2)=1
0022            J=0
0023         80 CALL COMSQT(UDISC,VDISC,UTEMP,VTEMP)
0024            UD=2.0*UA(3)
0025            VD=2.0*VA(3)
0026            BBB=UD*UD+VD*VD
0027            UROOT(J+1)=((-UA(2)+UTEMP)*UD+(-VA(2)+VTEMP)*VD)/BBB
0028            VROOT(J+1)=((-VA(2)+VTEMP)*UD-(-UA(2)+UTEMP)*VD)/BBB
0029            UROOT(J+2)=((-UA(2)-UTEMP)*UD+(-VA(2)-VTEMP)*VD)/BBB
0030            VROOT(J+2)=((-VA(2)-VTEMP)*UD-(-UA(2)-UTEMP)*VD)/BBB
0031            J=J+2
0032            GO TO 200
0033        100 IF(J.LT.0) GO TO 110
0034            J=J+1
0035            GO TO 130
0036        110 MULT(1)=2
0037            J=1
0038        130 UD=2.0*UA(3)
0039            VD=2.0*VA(3)
0040            BBB=UD*UD+VD*VD
0041            UROOT(J)=(-UA(2)*UD-VA(2)*VD)/BBB
0042            VROOT(J)=(-VA(2)*UD+UA(2)*VD)/BBB
0043        200 RETURN
0044            END
```

TABLE E.VII (Continued)

```
0001            SUBROUTINE NEWTON(UX,VX,N,UP,VP,UXO,VXO,CONV)
                C   ************************************************************************
                C   *                                                                      *
                C   * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-  *
                C   * IMATION BY USING THE ITERATION FORMULA                               *
                C   *                X(N+1) = X(N)-P(X(N))/P'(X(N)).                       *
                C   *                                                                      *
                C   ************************************************************************
0002            DOUBLE PRECISION UX,VX,UP,VP,UXO,VXO,UB,VB,UDPXO,VDPXO,UPXO,VPXO,U
               1DIFF,VDIFF,EPS1,EPSLON,EPS3,EPS4,AAA,BBB
0003            DOUBLE PRECISION DDD
0004            DOUBLE PRECISION ABPXO
0005            DIMENSION UP(26),VP(26),UB(26),VB(26)
0006            COMMON EPS1,EPSLON,EPS3,EPS4,IO2,MAX
0007            LOGICAL CONV
0008            UXO=UX
0009            VXO=VX
0010            DO 10 I=1,MAX
0011            CALL HORNER(UXO,VXO,N,UP,VP,UB,VB,UDPXO,VDPXO)
0012            UPXO=UB(1)
0013            VPXO=VB(1)
0014            DDD=DSQRT(UDPXO*UDPXO+VDPXO*VDPXO)
0015            IF(DDD.NE.0.0) GO TO 5
0016            ABPXO=DSQRT(UPXO*UPXO+VPXO*VPXO)
0017            IF(ABPXO.EQ.0.0) GO TO 20
0018            GO TO 15
0019          5 BBB=UDPXO*UDPXO+VDPXO*VDPXO
0020            UDIFF=(UPXO*UDPXO+VPXO*VDPXO)/BBB
0021            VDIFF=(VPXO*UDPXO-UPXO*VDPXO)/BBB
0022            UXO=UXO-UDIFF
0023            VXO=VXO-VDIFF
0024            AAA=DSQRT(UDIFF*UDIFF+VDIFF*VDIFF)
0025            BBB=DSQRT(UXO*UXO+VXO*VXO)
0026            IF(BBB.EQ.0.0) GO TO 10
0027            IF(AAA/BBB.LT.EPSLON) GO TO 20
0028         10 CONTINUE
0029         15 CONV=.FALSE.
0030            RETURN
0031         20 CONV=.TRUE.
0032            RETURN
0033            END
```

TABLE E.VII (Continued)

```
0001            SUBROUTINE DIVIDE(N,UP,VP,M,UD,VD,K,UQ,VQ)
              C  ********************************************************************************
              C  *                                                                              *
              C  * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE          *
              C  * QUOTIENT POLYNOMIAL  H(X) = F(X)/G(X).                                       *
              C  *                                                                              *
              C  ********************************************************************************
0002            DOUBLE PRECISION UP,VP,UD,VD,UQ,VQ,UTERM,VTERM,UDUMMY
0003            DIMENSION UP(26),VP(26),UD(26),VD(26),UQ(26),VQ(26)
0004            K=N-M
0005            UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0006            UQ(K+1)=(UP(N+1)*UD(M+1)+VP(N+1)*VD(M+1))/UDUMMY
0007            VQ(K+1)=(VP(N+1)*UD(M+1)-UP(N+1)*VD(M+1))/UDUMMY
0008            IF(K.EQ.0) GO TO 100
0009            J=-1
0010            DO 50 I=1,K
0011            J=J+1
0012            UTERM=UP(N-J)
0013            VTERM=VP(N-J)
0014            KK=K+1
0015            NNN=M-J
0016            DO 40 M1=NNN,M
0017            IF(KK.GT.1) GO TO 10
0018            GO TO 45
0019           10 IF(M1.GE.1) GO TO 20
0020            GO TO 40
0021           20 UTERM=UTERM-(UQ(KK)*UD(M1)-VQ(KK)*VD(M1))
0022            VTERM=VTERM-(UQ(KK)*VD(M1)+VQ(KK)*UD(M1))
0023           40 KK=KK-1
0024           45 UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0025            UQ(K+1-I)=(UTERM*UD(M+1)+VTERM*VD(M+1))/UDUMMY
0026           50 VQ(K+1-I)=(VTERM*UD(M+1)-UTERM*VD(M+1))/UDUMMY
0027          100 RETURN
0028            END
```

TABLE E.VII (Continued)

```
0001            SUBROUTINE HORNER(UX,VX,N,UP,VP,UB,VB,UC,VC)
              C ********************************************************************************
              C *                                                                              *
              C * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A               *
              C * POINT D AND ITS DERIVATIVE AT D.  SYNTHETIC DIVISION IS USED TO              *
              C * DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X - D).                   *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UX,VX,UP,VP,UB,VB,UC,VC
0003            DOUBLE PRECISION UDUMMY,VDUMMY
0004            DIMENSION UP(26),VP(26),UB(26),VB(26)
0005            UB(N+1)=UP(N+1)
0006            VB(N+1)=VP(N+1)
0007            UB(N)=(UX*UB(N+1)-VX*VB(N+1))+UP(N)
0008            VB(N)=(UX*VB(N+1)+VX*UB(N+1))+VP(N)
0009            UC=UB(N+1)
0010            VC=VB(N+1)
0011            KKK=N-1
0012            DO 10 I=1,KKK
0013            UB(KKK+1-I)=(UX*UB(KKK+2-I)-VX*VB(KKK+2-I))+UP(KKK+1-I)
0014            VB(KKK+1-I)=(UX*VB(KKK+2-I)+VX*UB(KKK+2-I))+VP(KKK+1-I)
0015            UDUMMY=UX*UC-VX*VC
0016            VDUMMY=UX*VC+VX*UC
0017            UC=UDUMMY+UB(KKK+2-I)
0018         10 VC=VDUMMY+VB(KKK+2-I)
0019            RETURN
0020            END
```

```
0001            SUBROUTINE DERIV(N,UP,VP,M,UA,VA)
              C ********************************************************************************
              C *                                                                              *
              C * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF       *
              C * ITS DERIVATIVE P'(X).                                                        *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UP,VP,UA,VA,AAA
0003            DIMENSION UP(26),VP(26),UA(26),VA(26)
0004            KKK=N+1
0005            DO 10 I=2,KKK
0006            AAA=I-1
0007            UA(I-1)=AAA*UP(I)
0008         10 VA(I-1)=AAA*VP(I)
0009            M=N-1
0010            RETURN
0011            END
```

TABLE E.VII (Continued)

```
0001            SUBROUTINE MULTI(N,UP,VP,J,UROOT,VROOT,MULT)
                C   ********************************************************************************
                C   *                                                                              *
                C   * GIVEN N ZEROS OF A POLYNOMIAL, SUBROUTINE MULTI COMPUTES THEIR               *
                C   * MULTIPLICITIES.                                                              *
                C   *                                                                              *
                C   ********************************************************************************
0002            DOUBLE PRECISION UP,VP,UROOT,VROOT,UA,VA,UB,VB,UC,VC,EPS1,EPS2,EPS
                1LON,EPS3,BBB
0003            DIMENSION UP(26),VP(26),UROOT(25),VROOT(25),UA(26),VA(26),UB(26),V
                1B(26),MULT(25)
0004            COMMON EPS1,EPS2,EPS3,EPSLON,IO2,MAX
0005            DO 100 I=1,J
0006            KKK=N+1
0007            DO 10 K=1,KKK
0008            UA(K)=UP(K)
0009          10 VA(K)=VP(K)
0010            M=N
0011            MULT(I)=0
0012          20 CALL HORNER(UROOT(I),VROOT(I),M,UA,VA,UB,VB,UC,VC)
0013            BBB=DSQRT(UB(1)*UB(1)+VB(1)*VB(1))
0014            IF(BBB.LT.EPSLON) GO TO 50
0015            IF(MULT(I).EQ.0) GO TO 40
0016            GO TO 100
0017          40 WRITE(IO2,1000) EPSLON,I,UROOT(I),VROOT(I)
0018            GO TO 100
0019          50 MULT(I)=MULT(I)+1
0020            IF(M.GT.1) GO TO 60
0021            GO TO 100
0022          60 DO 70 K=1,M
0023            UA(K)=UB(K+1)
0024          70 VA(K)=VB(K+1)
0025            M=M-1
0026            GO TO 20
0027         100 CONTINUE
0028            RETURN
0029        1000 FORMAT(///15H THE EPSILON  (,D10.3,48H) CHECK IN SUBROUTINE MULTI
                1INDICATES THAT ROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,/80H IS NO
                2T CLOSE ENOUGH TO BE A TRUE ROOT.  IT IS PRINTED BELOW WITH MULTIP
                3LICITY O//)
0030            END
```

TABLE E.VII (Continued)

```
0001            SUBROUTINE CONSQT(UX,VX,UY,VY)
                C   ********************************************************************************
                C   *                                                                              *
                C   * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.                *
                C   *                                                                              *
                C   ********************************************************************************
0002            DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
0003            R=DSQRT(UX*UX+VX*VX)
0004            AAA=DSQRT(DABS((R+UX)/2.0))
0005            BBB=DSQRT(DABS((R-UX)/2.0))
0006            IF(VX) 10,20,30
0007          10 UY=AAA
0008             VY=-1.0*BBB
0009             GO TO 100
0010          20 IF(UX) 40,50,60
0011          30 UY=AAA
0012             VY=BBB
0013             GO TO 100
0014          40 DUMMY=DABS(UX)
0015             UY=0.0
0016             VY=DSQRT(DUMMY)
0017             GO TO 100
0018          50 UY=0.0
0019             VY=0.0
0020             GO TO 100
0021          60 DUMMY=DABS(UX)
0022             UY=DSQRT(DUMMY)
0023             VY=0.0
0024         100 RETURN
0025             END
```

APPENDIX F

G.C.D. - MULLER'S METHOD

1. Use of the Program

A double precision FORTRAN IV program using the G.C.D. method with Muller's method as a supporting method is presented here. Flow charts for this program are given in Figure F.1 while Table F.III gives a FORTRAN IV listing of this program. Single precision variables are listed in Table F.II. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from Table F.II.

This program is designed to solve polynomials having degree less than or equal to 25. In order to solve polynomials of degree N where N > 25, the data statement and array dimensions given in Table F.I must be changed.

In this program both the leading coefficient and the constant coefficient are assumed to be non-zero.

TABLE F.I

PROGRAM CHANGES NECESSARY TO SOLVE POLYNOMIALS OF DEGREE GREATER THAN 25 BY G.C.D. - MULLER'S METHOD

Main Program

Data Entry/1H1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1
URAPP(N,3), VRAPP(N,3)
UAPP(N,3), VAPP(N,3)
UP(N+1), VP(N+1)
UROOT(N), VROOT(N)
MULT(N)
UDP(N+1), VDP(N+1)
UD(N+1), VD(N+1)
UQ(N+1), VQ(N+1)
UQQ(N+1), VQQ(N+1)
UB(N+1), VB(N+1)
ENTRY(N+1)

Subroutines MULTI, DIVIDE, DERIV, GCD, and QUAD

See corresponding subroutines in Table E.I.

Subroutine MULLER

UROOT(N), VROOT(N)
MULT(N)
UAPP(N,3), VAPP(n,3)
UWORK(N+1), VWORK(N+1)
UB(N+1), VB(N+1)
UA(N+1), VA(N+1)
URAPP(N,3), VRAPP(N,3)

Subroutine BETTER

UROOT(N), VROOT(N)
UA(N+1), VA(N+1)
UBAPP(N,3), VBAPP(N,3)
UB(N+1), VB(N+1)
UROOTS(N), VROOTS(N)
URAPP(N,3), VRAPP(N,3)
MULT(N)

Subroutine GENAPP

APPR(N,3) APPI(N,3)

Subroutine HORNER

UA(N+1), VA(N+1)
UB(N+1), VB(N+1)

2. Input Data for G.C.D. - Muller's Method

The input data for G.C.D. - Muller's method is prepared exactly as described in Appendix E, § 2 for G.C.D. - Newton's method.

3. Variables Used in G.C.D. - Muller's Method

The main variables used in G.C.D. - Muller's method are given in Table F.II. The symbols used to indicate type and disposition are described in Appendix E, § 3. For variables not listed in Table F.II, see the main program or corresponding subprogram of Table E.VI.

4. Description of Program Output

The output from G.C.D. - Muller's method is identical to that for G.C.D. - Newton's method as described in Apptendix E, § 4, keeping in mind that Muller's instead of Newton's method is used. The expression "SOLVED BY DIRECT METHOD" is equivalent to "RESULTS OF SUBROUTINE QUAD." Only one initial approximation, X_0, (not three) is printed. The other two required by Muller's method were $.9X_0$ and $1.1X_0$.

5. Informative Messages and Error Messages

The informative messages and error messages in this program are described as follows. For other messages not listed here, see Appendix E, § 5.

"THE EPSILON (XXX) CHECK IN SUBROUTINE MULTI INDICATES THAT ROOT YY = ZZZ IS NOT CLOSE ENOUGHT TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIPLICITY 0." This message is described in Appendix E, § 5.

"COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." This message is described in Apptendix E, § 5.

"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX." XX represents the number of the polynomial for which no zeros were extracted.

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT XX = YYY DID NOT CONVERGE AFTER ZZZ ITERATIONS." This message indicates that a root did not produce convergence during the attempt to improve accuracy. XX represents the number of the root before the attempt to improve accuracy, YYY represents its value, and ZZZ represents the maximum number of iterations. The following message then follows. "THE PRESENT APPROXIMATION IS AAA." AAA represents the present approximation to the root after the maximum number of iterations.

TABLE F.II

VARIABLES USED IN G.C.D. - MULLER'S METHOD

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Subroutine MULLER
NP	I	NP	I	E	Degree of polynomial P(X)
NROOT	I	NROOT	I	R	Number of distinst roots found
NOMULT	I	NOMULT	I		Number of roots (counting multiplicities)
ROOT	C	UROOT,VROOT	D	R	Array containing the roots
NAPP	I	NAPP	I	E	Number of initial approximations to be read in
APP	C	UAPP,VAPP	D	E	Array of initial approximations
WORK	C	UWORK,VWORK	D		Working array containing coefficients of current polynomial
B	C	UB,VB	D		Array containing coefficients of deflated polynomial
A	C	UA,VA	D	E	Array containing coefficients of original polynomial, P(X)
RAPP	C	URAPP,VRAPP	D	R	Array of initial or altered approximation for which convergence was obtained
X1	C	UX1,VX1	D		One of three current approximations to a root
X2	C	UX2,VX2	D		One of three current approximations to a root
X3	C	UX3,VX3	D		One of three current approximations to a root
PX1	C	UPX1,VPX1	D		Value of polynomial P(X) at X1
PX2	C	UPX2,VPX2	D		Value of polynomial P(X) at X2
PX3	C	UPX3,VPX3	D		Value of polynomial P(X) at X3
X4	C	UX4,VX4	D		Newest approximation (X_{n+1}) to root
PX4	C	UPX4,VPX4	D		Value of polynomial P(X) at X4
MULT	I	MULT	I		Array containing the multiplicities of each root found
ITER	I	ITER	I		Counter for iterations
I01	I	I01	I		Unit number of input device
I02	I	I02	I	C	Unit number of output device
EPSRT	R	EPSRT	D	C	Number used in subroutine BETTER to generate two approximations from the one given
NOPOLY	I	NOPOLY	I	E	Number of the polynomial

TABLE F.II (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
MAX	I	MAX	I	C	Maximum number of iterations
EPS	R	EPS	D	C	Tolerance check for convergence
EPSO	R	EPSO	D	C	Tolerance check for zero (0)
EPSM	R	EPSM	D	C	Tolerance check for multiplicities
KCHECK	I	KCHECK	I		Program control, KCHECK = 1 stops execution of program
XSTART	R	XSTART	D	E	Magnitude at which to start generating initial approximations
XEND	R	XEND	D	E	Magnitude at which to end generating initial approximations
NWORK	I	NWORK	I		Degree of current deflated polynomial whose coefficients are in WORK
ITIME	I	ITIME	I		Program control
NALTER	I	NALTER	I		Number of alterations which have been performed on an initial approximation
IAPP	I	IAPP	I		Counter for number of initial approximations used
CONV	L	CONV	L		When CONV is true, convergence has been obtained
IROOT	I	IROOT	I	R	Number of distinct roots solved by Muller's method, i.e. not solved directly by subroutine QUAD
					Subroutine HORNER
A	C	UA,VA	D	E	Array of current polynomial coefficients (to be deflated or evaluated)
NA	I	NA	I	E	Degree of polynomial to be deflated or evaluated
X	C	UX,VX	D	E	Approximation at which to evaluate or deflate the polynomial
B	C	UB,VB	D	R	Array containing the coefficients of the deflated polynomial
PX	C	UPX,VPX	D	R	Value of the polynomial at X
NUM	I	NUM	I		Number of coefficients of polynomial to be deflated

TABLE F.II (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Subroutine TEST
X3	C	UX3,VX3	D	E	Approximation to root (old) (X_n)
X4	C	UX4,VX4	D	E	New approximation to root (X_{n+1})
CONV	L	CONV	L	R	CONV = True implies convergence has been obtained
EPS	R	EPS	D	C	Tolerance for convergence test
EPSO	R	EPSO	D	C	Tolerance check for zero (0)
DENOM	R	DENOM	D		Magnitude of new approximation, (X_{n+1})
					Subroutine BETTER
MULT	I	MULT	I	ECR	Array of multiplicities of each root
A	C	UA,VA	D	E	Array of coefficients of original undeflated polynomial
NP	I	NP	I	E	Degree of original polynomial
ROOT	C	UROOT,VROOT	D	ECR	Array of ROOTS
NROOT	I	NROOT	I	ECR	Number of roots stored in ROOT
BAPP	C	UBAPP,VBAPP	D	E	Array of initial approximations (old roots)
IROOT	I	IROOT	I	ECR	Number of roots solved by the iterative process (Not QUAD)
ROOTS	C	UROOTS,VROOTS	D		Temporary storage for new (better) roots
L	I	L	I		Number of roots found by BETTER
EPSRT	R	EPSRT	D	C	A small number used to generate two of the three approximations when given one
ITER	I	ITER	I	C	Counter for number of iterations
B	C	UB,VB	D		Array containing coefficients of deflated polynomial
X1	C	UX1,VX1	D		One of three approximations to the root
X2	C	UX2,VX2	D		One of three approximations to the root
X3	C	UX3,VX3	D		One of three approximations to the root
PX1	C	UPX1,VPX1	D		Value of polynomial (P(X)) at X1
PX2	C	UPX2,VPX2	D		Value of polynomial (P(X)) at X2
PX3	C	UPX3,VPX3	D		Value of polynomial (P(X)) at X3

TABLE F.II (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
CONV	L	CONV	L		CONV = true implies convergence has been obtained
X4	C	UX4,VX4	D		Newest approximation to root
J	I	J	I		Program control - counts the number of roots used as initial approximations
MAX	I	MAX	I	C	Maximum number of iterations permitted
I02	I	I02	I	C	Unit number of output device
					Subroutine ALTER
X1	C	X1R,X1I	D	ECR	One of the three approximations to be altered
X2	C	X2R,X2I	D	ECR	One of the three approximations to be altered
X3	C	X3R,X3I	D	ECR	One of the three approximations to be altered
X2R	R	X2R	D		Real part of complex approximation
X2I	R	X2I	D		Imaginary part of complex approximation
					Subroutine CALC

These variables are dummy variables used for temporary storage and thus, are not defined.

MAIN PROGRAM

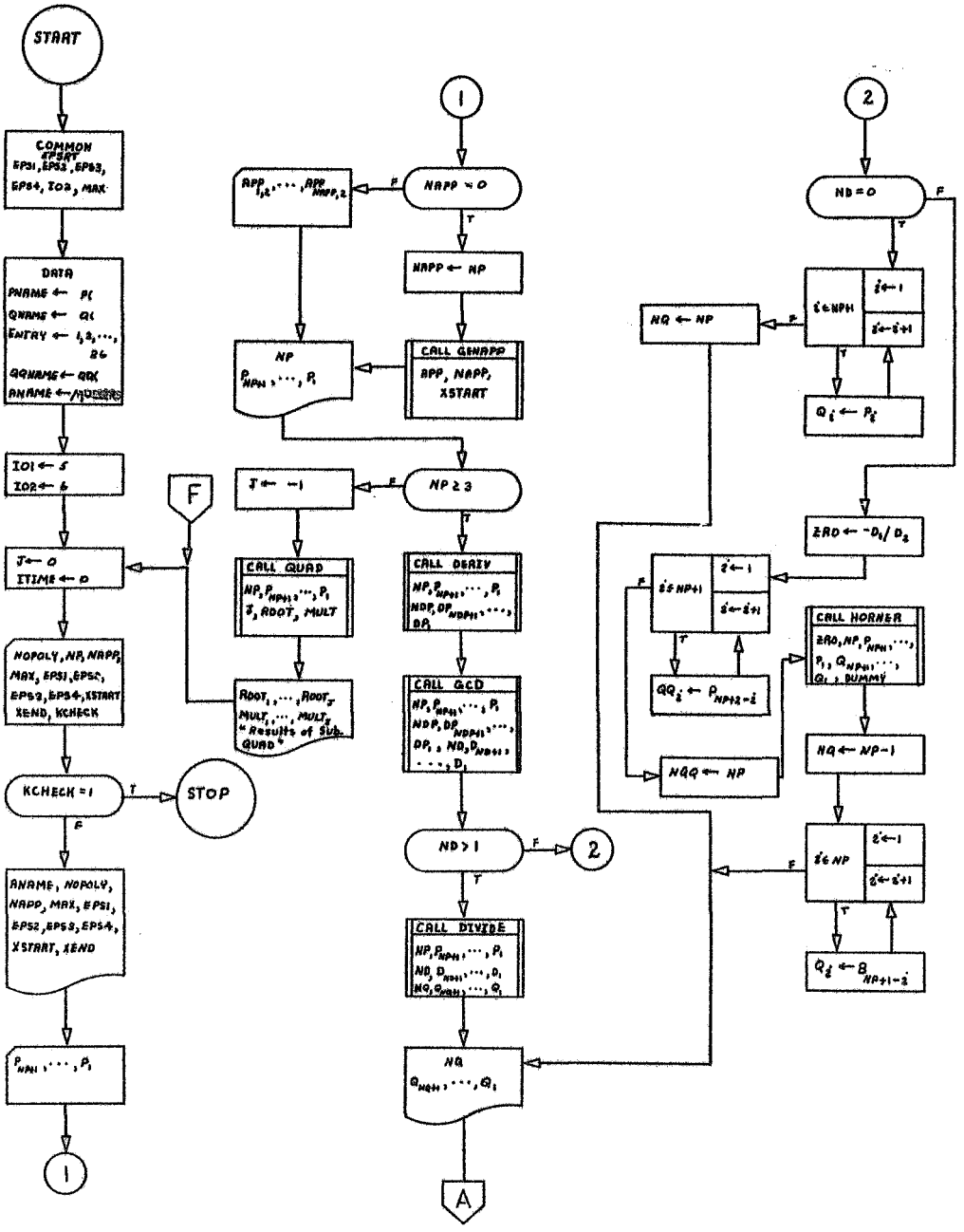


Figure F.1. Flow Charts for G.C.D.-Muller's Method

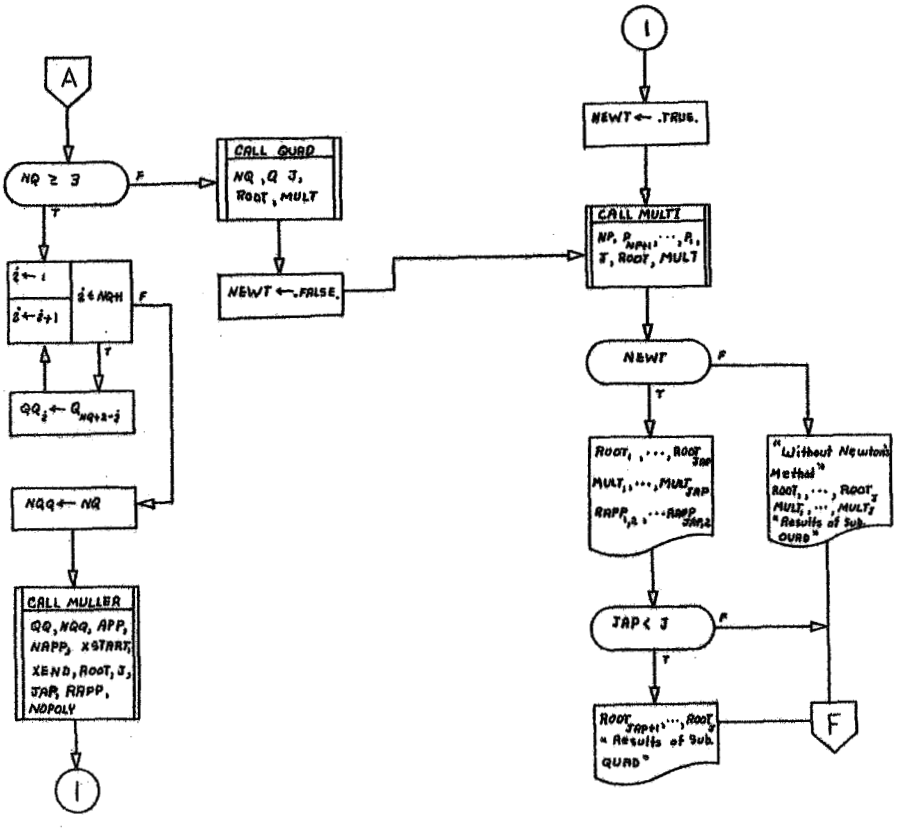


Figure F.1. (Continued)

MULLER

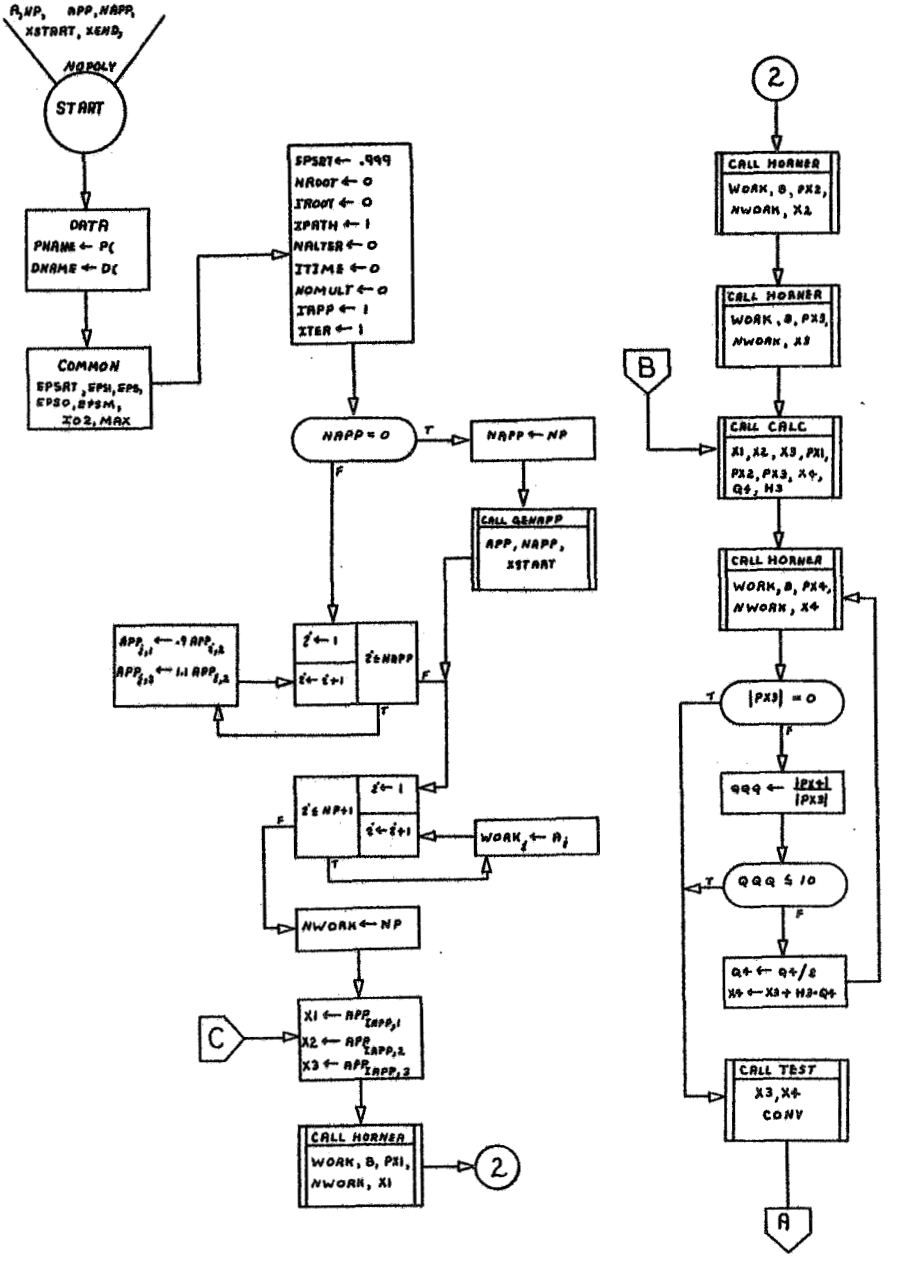


Figure F.1. (Continued)

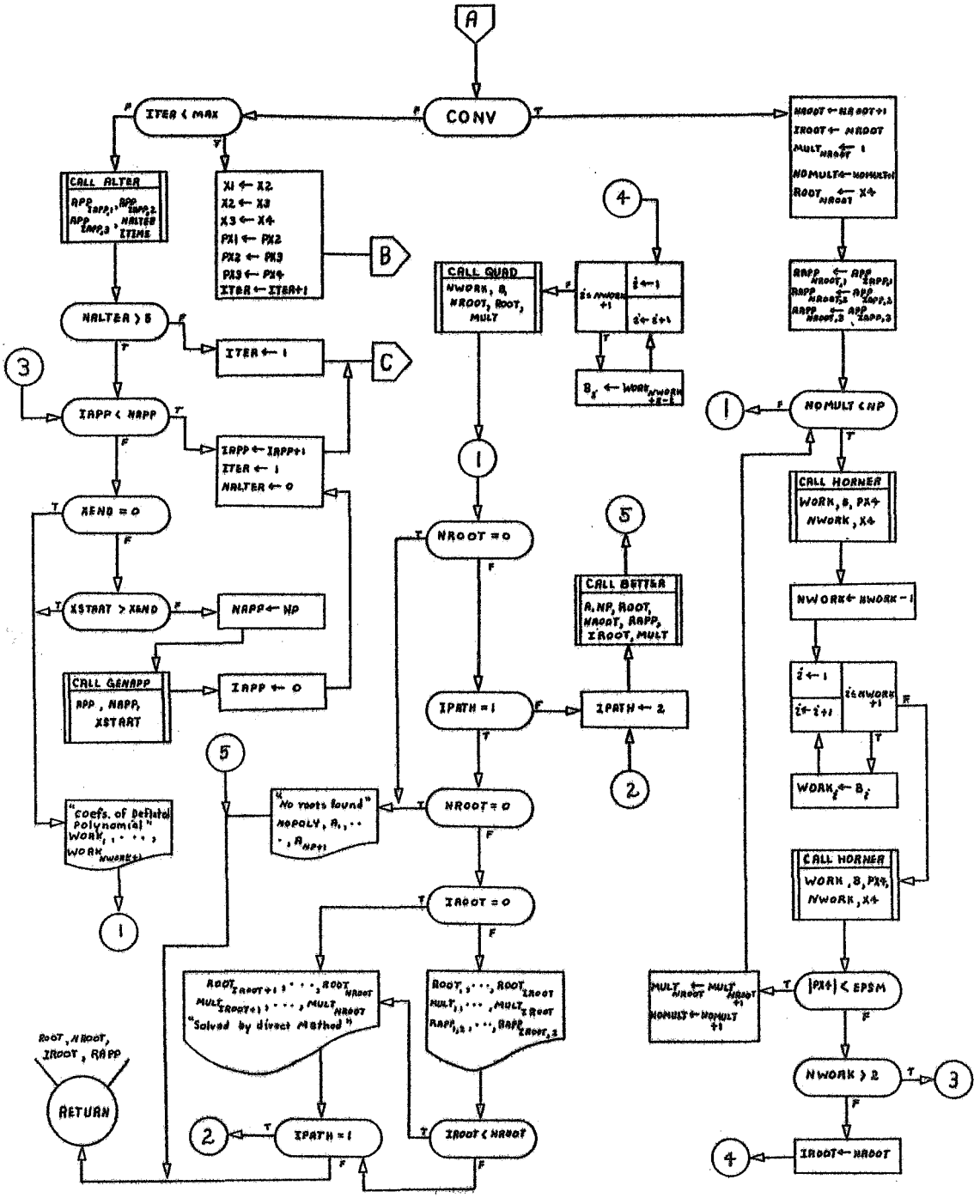


Figure F.1. (Continued)

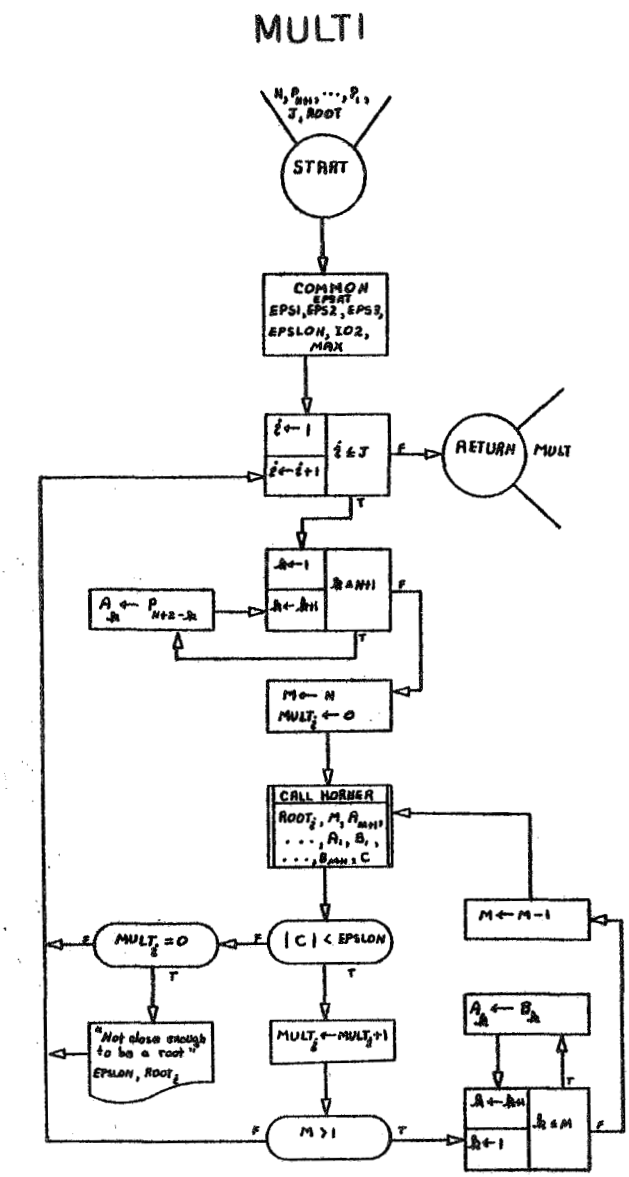
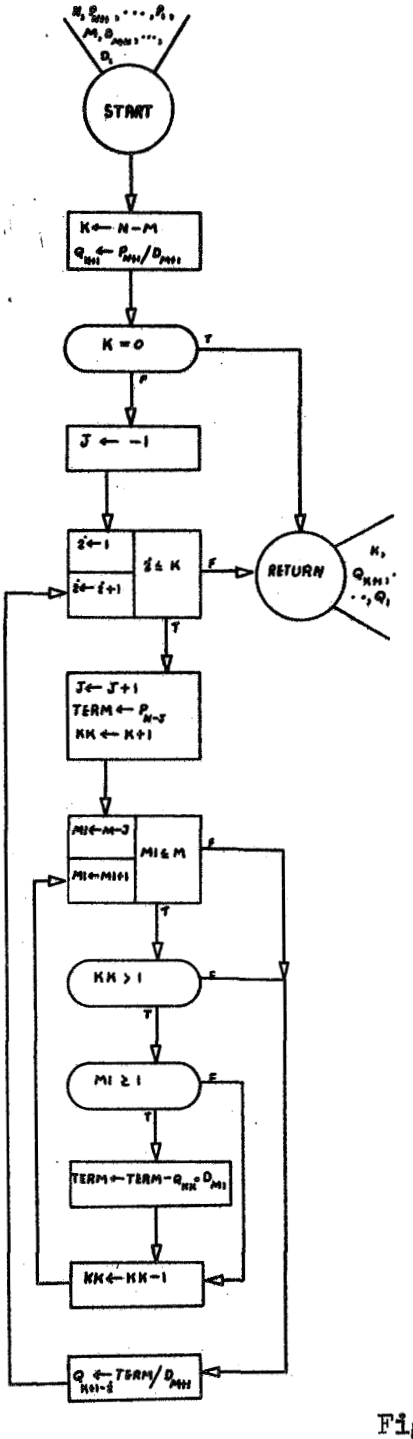


Figure F.1. (Continued)

DIVIDE



DERIV

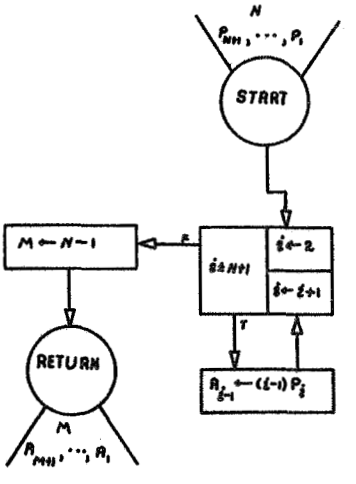


Figure F.1. (Continued)

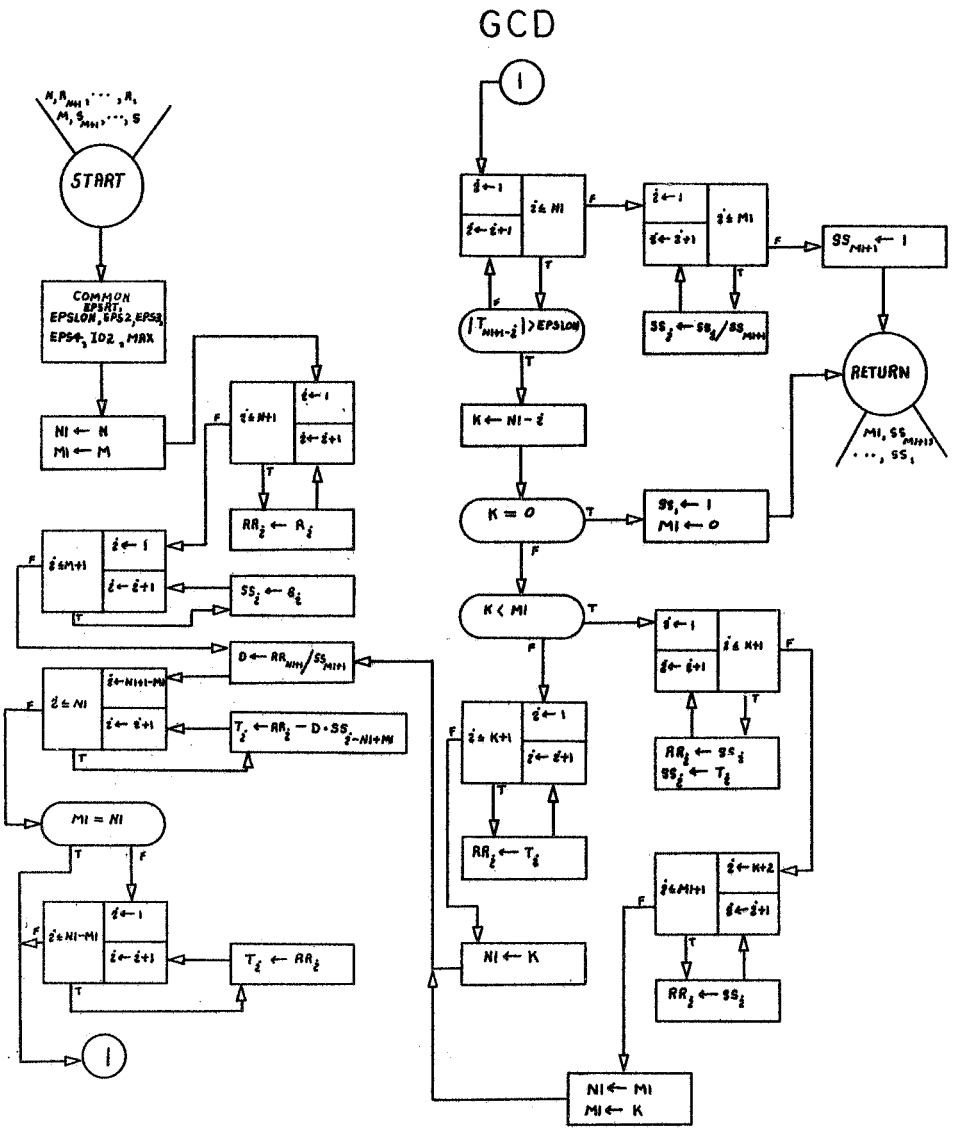


Figure F.1. (Continued)

QUAD

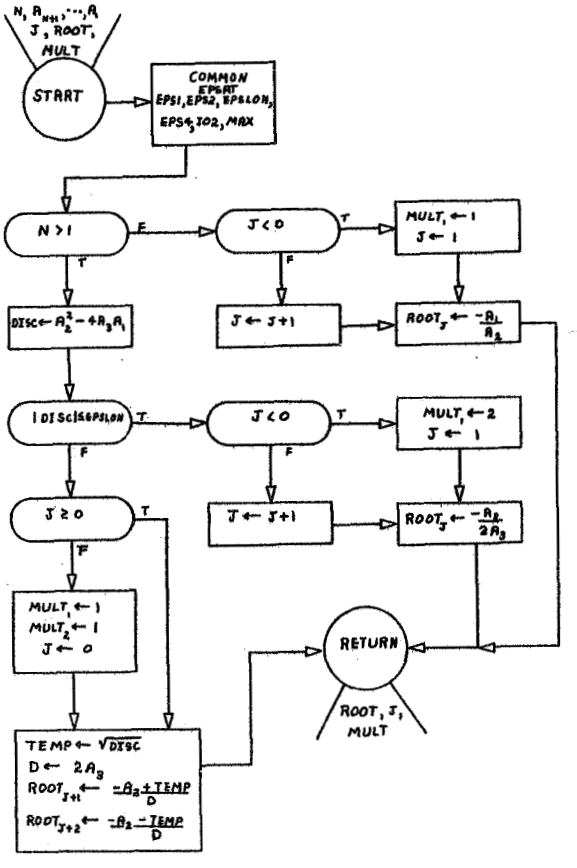


Figure F.1. (Continued)

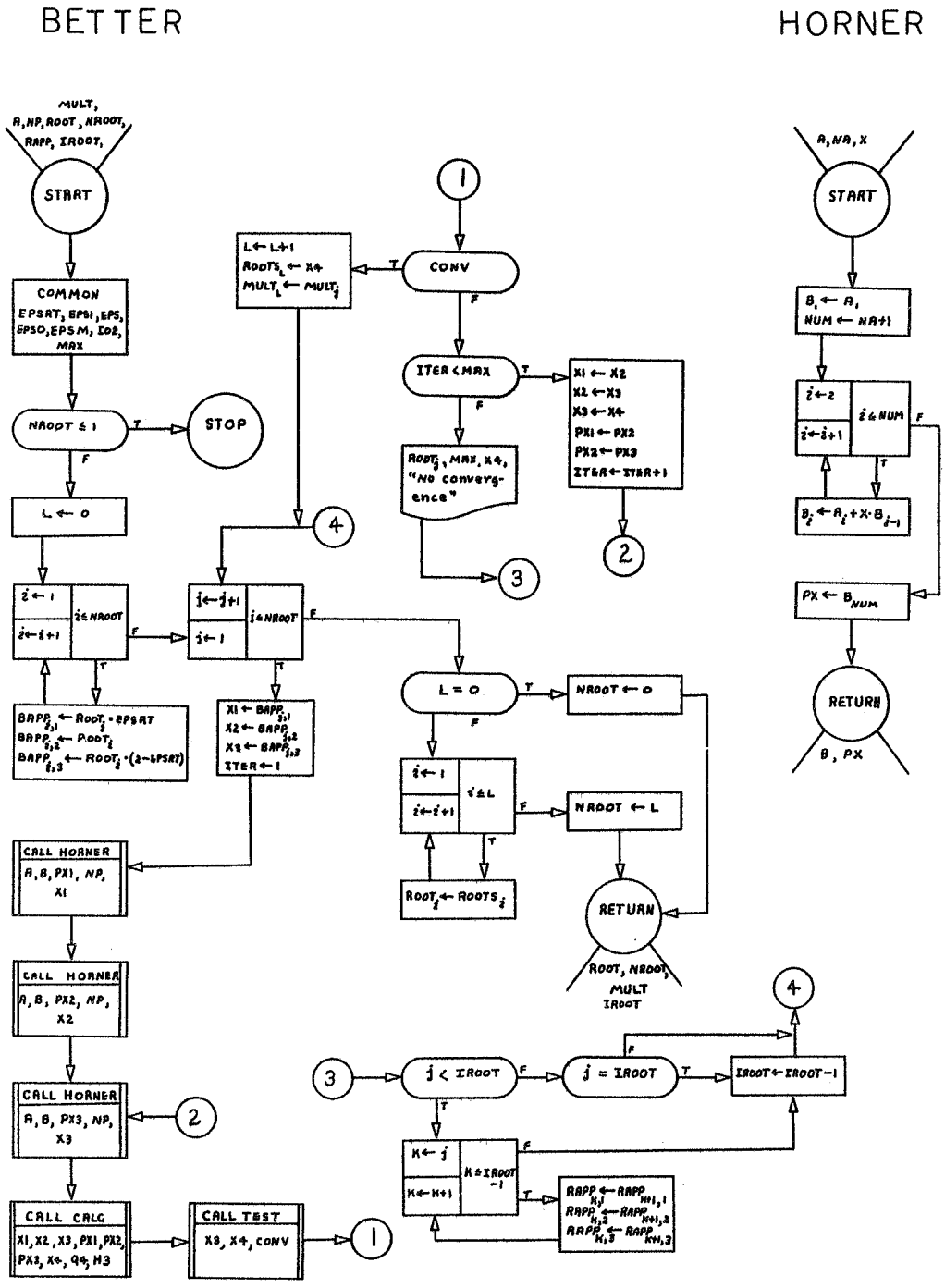


Figure F.1. (Continued)

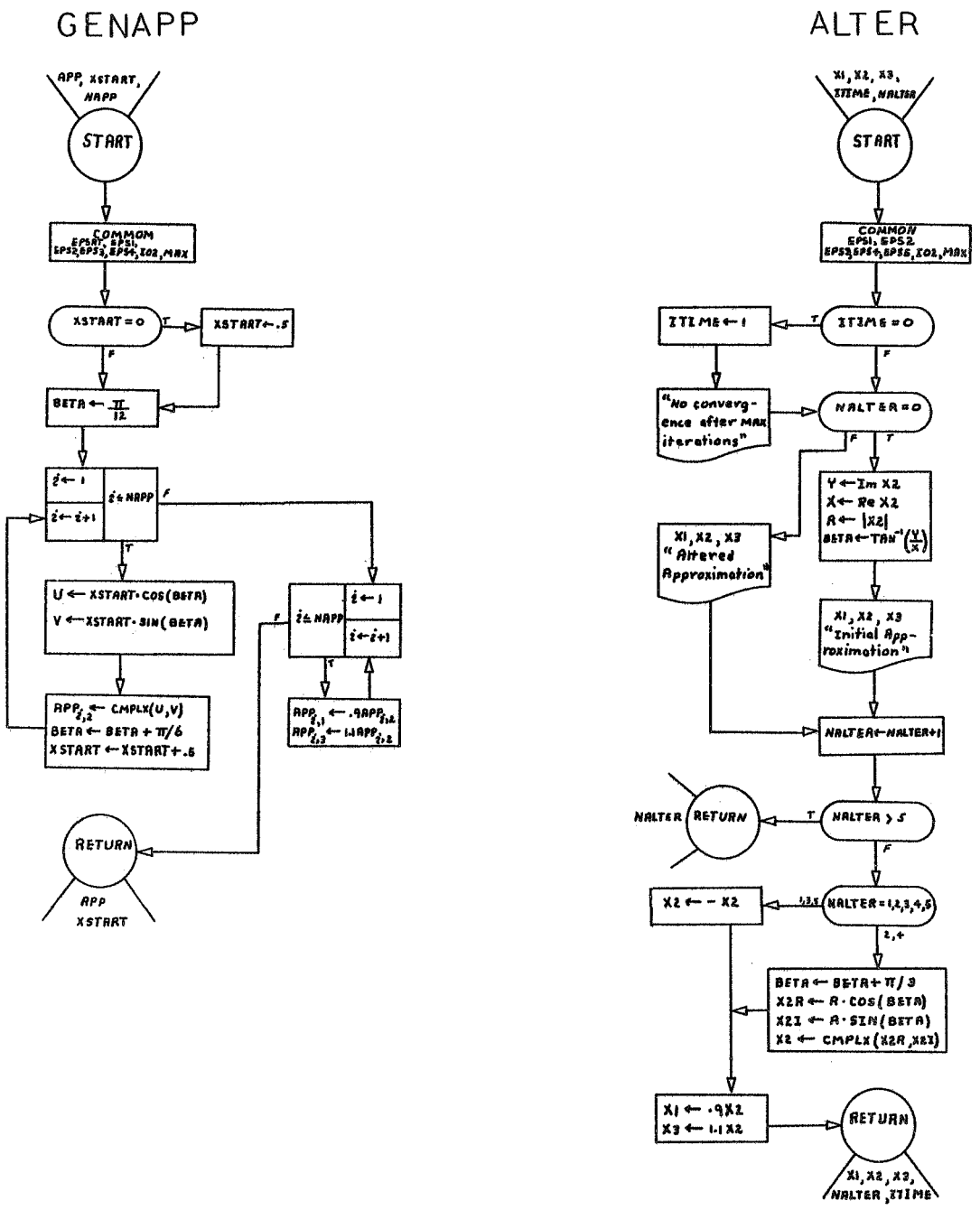


Figure F.1. (Continued)

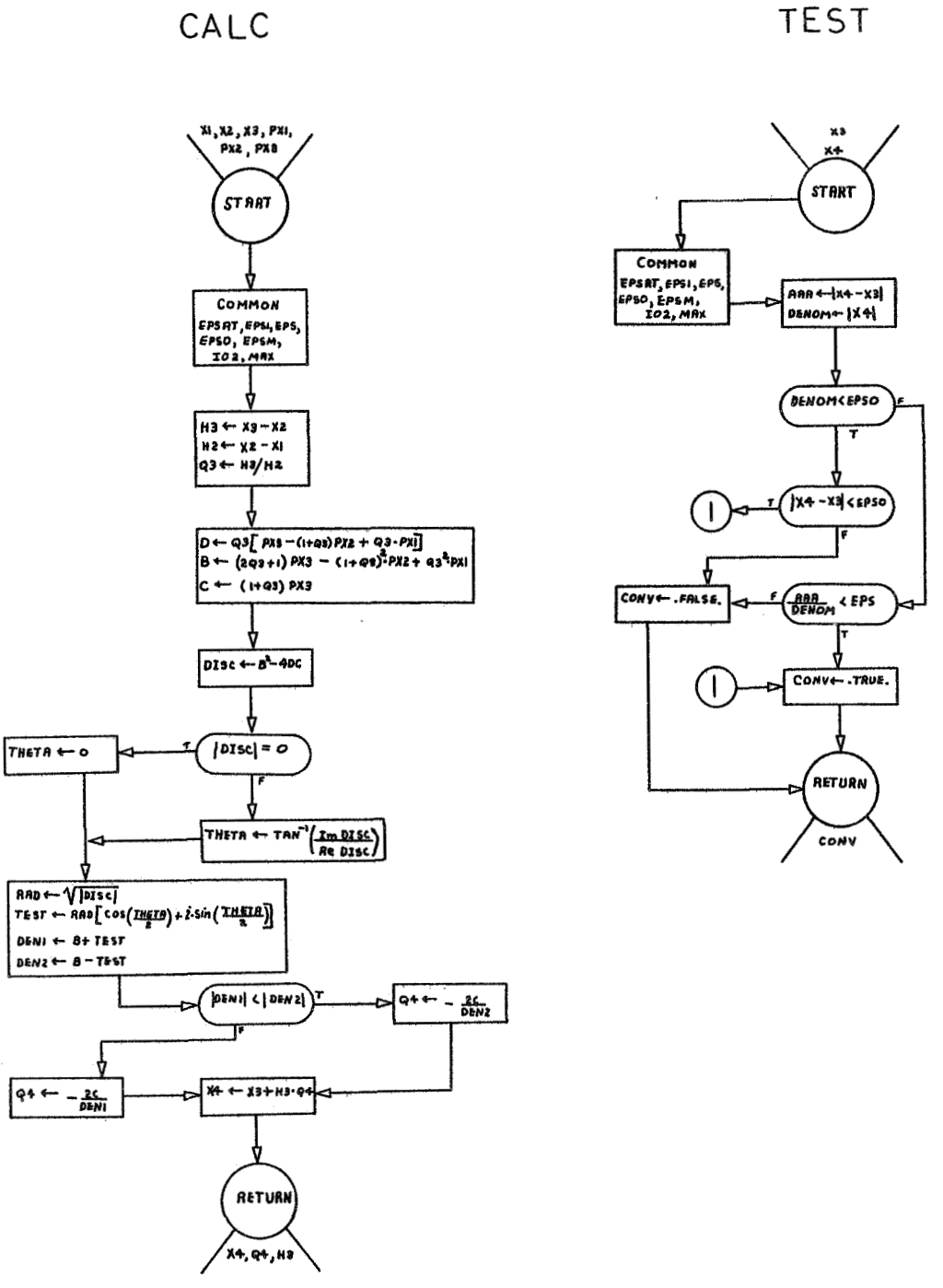


Figure F.1. (Continued)

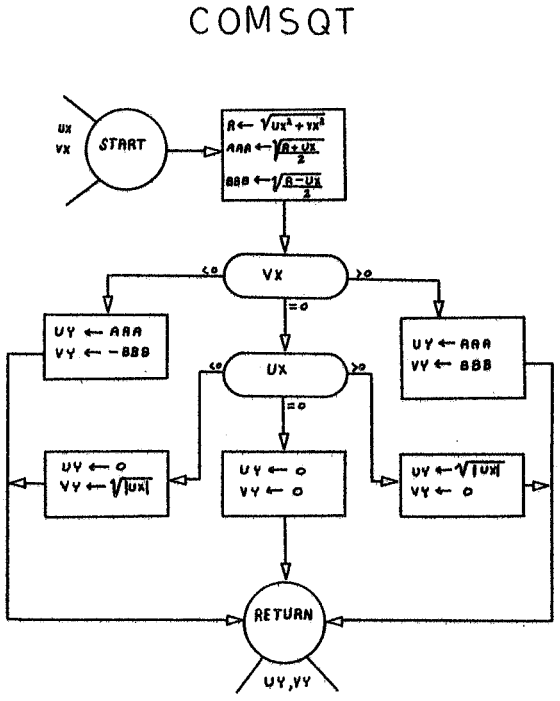


Figure F.1. (Continued)

TABLE F.III

PROGRAM FOR G.C.D.-MULLER'S METHOD

```
C   ************************************************************************
C   *                                                                      *
C   * DOUBLE PRECISION PROGRAM FOR G.C.D. - MULLER'S METHOD                *
C   *                                                                      *
C   *                                                                      *
C   * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A   *
C   * POLYNOMIAL OF MAXIMUM DEGREE 25.  ALL MULTIPLE ROOTS ARE REMOVED BY  *
C   * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL *
C   * AND ITS DERIVATIVE.  THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED *
C   * AND THEIR MULTIPLICITIES DETERMINED.                                 *
C   *                                                                      *
C   ************************************************************************
0001          DOUBLE PRECISION URAPP,VRAPP
0002          DOUBLE PRECISION UP,VP,UAPP,VAPP,UROOT,VROOT,UDP,VDP,UD,VD,UZRO,VZ
             1RO,UQ,VQ,UDUMMY,VDUMMY,UQQ,VQQ,UB,VB,EPS1,EPS2,EPS3,EPS4
0003          DIMENSION URAPP(25,3),VRAPP(25,3),UAPP(25,3),VAPP(25,3)
0004          DIMENSION UP(26),VP(26),UROOT(25),VROOT(25),MULT(25),UDP(26),VDP(2
             16),UD(26),VD(26),UQ(26),VQ(26),UQQ(26),VQQ(26),UB(26),VB(26),ANAME
             2(2),ENTRY(26)
0005          DOUBLE PRECISION XSTART
0006          DOUBLE PRECISION XEND
0007          DOUBLE PRECISION EPSRT
0008          COMMON EPSRT,EPS1,EPS2,EPS3,EPS4,IO2,MAX
0009          DATA PNAME,QNAME,QQNAME/2HP(,2HQ(,3HQQ(/
0010          DATA ENTRY/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,2H12,2H13
             1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
0011          DATA ANAME(1),ANAME(2)/4HMULL,4HERS /
0012          LOGICAL NEWT
0013          IO1=5
0014          IO2=6
0015       10 J=0
0016          ITIME=0
0017          READ(IO1,1000) NOPOLY,NP,NAPP,MAX,EPS1,EPS2,EPS3,EPS4,XSTART,XEND,
             1KCHECK
0018          IF(KCHECK.EQ.1) STOP
0019          WRITE(IO2,1020) ANAME(1),ANAME(2),NOPOLY
0020          WRITE(IO2,2000) NAPP
0021          WRITE(IO2,2010) MAX
0022          WRITE(IO2,2070) EPS1
0023          WRITE(IO2,2020) EPS2
0024          WRITE(IO2,2080) EPS3
0025          WRITE(IO2,2030) EPS4
0026          WRITE(IO2,2040) XSTART
0027          WRITE(IO2,2050) XEND
0028          WRITE(IO2,2060)
0029          KKK=NP+1
0030          NNN=KKK+1
0031          DO 20 I=1,KKK
0032          JJJ=NNN-I
0033       20 READ(IO1,1010) UP(JJJ),VP(JJJ)
0034          IF(NAPP.NE.0) GO TO 22
0035          NAPP=NP
0036          CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0037          GO TO 23
0038       22 READ(IO1,1015) (UAPP(I,2),VAPP(I,2),I=1,NAPP)
0039       23 WRITE(IO2,1030) NP
0040          KKK=NP+1
0041          NNN=KKK+1
```

TABLE F.III (Continued)

```
0042              DO 25 I=1,KKK
0043              JJJ=NNN-I
0044           25 WRITE(IO2,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
0045              IF(NP.GE.3) GO TO 30
0046              J=-1
0047              CALL QUAD(NP,UP,VP,J,UROOT,VROOT,MULT)
0048              WRITE(IO2,1070)
0049              WRITE(IO2,1165) (I,UROOT(I),VROOT(I),MULT(I),I=1,J)
0050              GO TO 10
0051           30 CALL DERIV(NP,UP,VP,NDP,UDP,VDP)
0052              CALL GCD(NP,UP,VP,NDP,UDP,VDP,ND,UD,VD)
0053              IF(ND.GT.1) GO TO 70
0054              IF(ND.EQ.0) GO TO 65
0055              UDUMMY=UD(2)*UD(2)+VD(2)*VD(2)
0056              UZRO=-(UD(1)*UD(2)+VD(1)*VD(2))/UDUMMY
0057              VZRO=-(UD(2)*VD(1)-UD(1)*VD(2))/UDUMMY
0058              KKK=NP+1
0059              DO 55 I=1,KKK
0060              UQQ(I)=UP(KKK+1-I)
0061           55 VQQ(I)=VP(KKK+1-I)
0062              NQQ=NP
0063              CALL HORNER(NQQ,UQQ,VQQ,UZRO,VZRO,UB,VB,UDUMMY,VDUMMY)
0064              NQ=NP-1
0065              DO 60 I=1,NP
0066              UQ(I)=UB(NP+1-I)
0067           60 VQ(I)=VB(NP+1-I)
0068              GO TO 80
0069           65 KKK=NP+1
0070              DO 66 I=1,KKK
0071              UQ(I)=UP(I)
0072           66 VQ(I)=VP(I)
0073              NQ=NP
0074              GO TO 80
0075           70 CALL DIVIDE(NP,UP,VP,ND,UD,VD,NQ,UQ,VQ)
0076           80 WRITE(IO2,1120) NQ
0077              KKK=NQ+1
0078              NNN=KKK+1
0079              DO 83 I=1,KKK
0080              JJJ=NNN-I
0081           83 WRITE(IO2,1040) QNAME,ENTRY(JJJ),UQ(JJJ),VQ(JJJ)
0082              IF(NQ.GE.3) GO TO 85
0083              GO TO 110
0084           85 KKK=NQ+1
0085              DO 90 I=1,KKK
0086              UQQ(I)=UQ(KKK+1-I)
0087           90 VQQ(I)=VQ(KKK+1-I)
0088              NQQ=NQ
0089              GO TO 120
0090          110 CALL QUAD(NQ,UQ,VQ,J,UROOT,VROOT,MULT)
0091              NEWT=.FALSE.
0092              GO TO 310
0093          120 CALL MULLER(UQQ,VQQ,NQQ,UAPP,VAPP,NAPP,XSTART,XEND,UROOT,VROOT,J,J
                 1AP,URAPP,VRAPP,NOPOLY)
0094              NEWT=.TRUE.
0095          310 CALL MULTI(NP,UP,VP,J,UROOT,VROOT,MULT)
0096              IF(NEWT) GO TO 330
0097              WRITE(IO2,1070)
0098              WRITE(IO2,1165) (L,UROOT(L),VROOT(L),MULT(L),L=1,J)
```

TABLE F.III (Continued)

```
0099              GO TO 10
0100           330 WRITE(IO2,1180)
0101              DO 350 L=1,JAP
0102           350 WRITE(IO2,1190) L,UROOT(L),VROOT(L),MULT(L),URAPP(L,2),VRAPP(L,2)
0103              KKK=JAP+1
0104              IF(JAP.LT.J) WRITE(IO2,1165) (L,UROOT(L),VROOT(L),MULT(L),L=KKK,J)
0105              GO TO 10
0106          1000 FORMAT(3(I2,1X),9X,I3,1X,4(D6.0,1X),13X,2(D7.0,1X),I1)
0107          1010 FORMAT(2D30.0)
0108          1015 FORMAT(2D30.0)
0109          1020 FORMAT(1H1,10X,41HGREATEST COMMON DIVISOR METHOD USED WITH ,2(A4),
                  135HMETHOD TO FIND ZEROS OF POLYNOMIALS/11X,18HPOLYNOMIAL NUMBER ,I
                  22///)
0110          1030 FORMAT(1X,22HTHE DEGREE OF P(X) IS ,I2,22H  THE COEFFICIENTS ARE//
                  1)
0111          1040 FORMAT(2X,A2,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
0112          1070 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES//)
0113          1080 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,10X,I2)
0114          1100 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
0115          1120 FORMAT(///1X,73HQ(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE
                  1DISTINCT ROOTS OF P(X)./1X,22HTHE DEGREE OF Q(X) IS ,I2,22H  THE C
                  2OEFFICIENTS ARE//)
0116          1165 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,10X,26H
                  1RESULTS OF SUBROUTINE QUAD)
0117          1180 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
                  1 APPROXIMATION//)
0118          1190 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,9X,D23.
                  116,3H + ,D23.16,2H I)
0119          2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
0120          2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
0121          2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
0122          2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES.,10X,D9.2)
0123          2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
0124          2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
0125          2060 FORMAT(//1X)
0126          2070 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE GCD.  ,D9.2)
0127          2080 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE QUAD. ,D9.2)
0128              END
```

TABLE F.III (Continued)

```
0001            SUBROUTINE MULTI(N,UP,VP,J,UROOT,VROOT,MULT)
              C   ********************************************************************************
              C   *                                                                              *
              C   * GIVEN N ZEROS OF A POLYNOMIAL, SUBROUTINE MULTI COMPUTES THEIR               *
              C   * MULTIPLICITIES.                                                              *
              C   *                                                                              *
              C   ********************************************************************************
0002            DOUBLE PRECISION UP,VP,UROOT,VROOT,UA,VA,UB,VB,UC,VC,EPS1,EPS2,EPS
               1LON,EPS3,BBB
0003            DIMENSION UP(26),VP(26),UROOT(25),VROOT(25),UA(26),VA(26),UB(26),V
               1B(26),MULT(25)
0004            DOUBLE PRECISION EPSRT
0005            COMMON EPSRT,EPS1,EPS2,EPS3,EPSLON,IO2,MAX
0006            DO 100 I=1,J
0007            KKK=N+1
0008            DO 10 K=1,KKK
0009            UA(K)=UP(KKK+1-K)
0010          10 VA(K)=VP(KKK+1-K)
0011            M=N
0012            MULT(I)=0
0013          20 CALL HORNER(M,UA,VA,UROOT(I),VROOT(I),UB,VB,UC,VC)
0014            BBB=DSQRT(UC*UC+VC*VC)
0015            IF(BBB.LT.EPSLON) GO TO 50
0016            IF(MULT(I).EQ.0) GO TO 40
0017            GO TO 100
0018          40 WRITE(IO2,1000) EPSLON,I,UROOT(I),VROOT(I)
0019            GO TO 100
0020          50 MULT(I)=MULT(I)+1
0021            IF(M.GT.1) GO TO 60
0022            GO TO 100
0023          60 DO 70 K=1,M
0024            UA(K)=UB(K)
0025          70 VA(K)=VB(K)
0026            M=M-1
0027            GO TO 20
0028         100 CONTINUE
0029            RETURN
0030        1000 FORMAT(///15H THE EPSILON  (,D10.3,48H) CHECK IN SUBROUTINE MULTI
               1INDICATES THAT ROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,/80H IS NO
               2T CLOSE ENOUGH TO BE A TRUE ROOT.  IT IS PRINTED BELOW WITH MULTIP
               3LICITY 0//)
0031            END
```

TABLE F.III (Continued)

```
0001              SUBROUTINE DIVIDE(N,UP,VP,M,UD,VD,K,UQ,VQ)
                C  ********************************************************************************
                C  *                                                                              *
                C  * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE          *
                C  * QUOTIENT POLYNOMIAL  H(X) = F(X)/G(X).                                       *
                C  *                                                                              *
                C  ********************************************************************************
0002              DOUBLE PRECISION UP,VP,UD,VD,UQ,VQ,UTERM,VTERM,UDUMMY
0003              DIMENSION UP(26),VP(26),UD(26),VD(26),UQ(26),VQ(26)
0004              K=N-M
0005              UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0006              UQ(K+1)=(UP(N+1)*UD(M+1)+VP(N+1)*VD(M+1))/UDUMMY
0007              VQ(K+1)=(VP(N+1)*UD(M+1)-UP(N+1)*VD(M+1))/UDUMMY
0008              IF(K.EQ.0) GO TO 100
0009              J=-1
0010              DO 50 I=1,K
0011              J=J+1
0012              UTERM=UP(N-J)
0013              VTERM=VP(N-J)
0014              KK=K+1
0015              NNN=M-J
0016              DO 40 M1=NNN,M
0017              IF(KK.GT.1) GO TO 10
0018              GO TO 45
0019             10 IF(M1.GE.1) GO TO 20
0020              GO TO 40
0021             20 UTERM=UTERM-(UQ(KK)*UD(M1)-VQ(KK)*VD(M1))
0022              VTERM=VTERM-(UQ(KK)*VD(M1)+VQ(KK)*UD(M1))
0023             40 KK=KK-1
0024             45 UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0025              UQ(K+1-I)=(UTERM*UD(M+1)+VTERM*VD(M+1))/UDUMMY
0026             50 VQ(K+1-I)=(VTERM*UD(M+1)-UTERM*VD(M+1))/UDUMMY
0027            100 RETURN
0028              END
```

```
0001              SUBROUTINE DERIV(N,UP,VP,M,UA,VA)
                C  ********************************************************************************
                C  *                                                                              *
                C  * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF       *
                C  * ITS DERIVATIVE P'(X).                                                        *
                C  *                                                                              *
                C  ********************************************************************************
0002              DOUBLE PRECISION UP,VP,UA,VA,AAA
0003              DIMENSION UP(26),VP(26),UA(26),VA(26)
0004              KKK=N+1
0005              DO 10 I=2,KKK
0006              AAA=I-1
0007              UA(I-1)=AAA*UP(I)
0008             10 VA(I-1)=AAA*VP(I)
0009              M=N-1
0010              RETURN
0011              END
```

TABLE F.III (Continued)

```
0001            SUBROUTINE GCD(N,UR,VR,M,US,VS,M1,USS,VSS)
              C  ********************************************************************************
              C  *                                                                              *
              C  * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.          *
              C  * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND        *
              C  * DP(X).                                                                      *
              C  *                                                                              *
              C  ********************************************************************************
0002            DOUBLE PRECISION EPSRT
0003            DOUBLE PRECISION USSSSS,VSSSSS
0004            DOUBLE PRECISION UR,VR,US,VS,USS,VSS,URR,VRR,UD,VD,UT,VT,EPSLON,EP
               1S2,EPS3,EPS4,BBB
0005            DIMENSION UR(26),VR(26),US(26),VS(26),USS(26),VSS(26),URR(26),VRR(
               126),UT(26),VT(26)
0006            COMMON EPSRT,EPSLON,EPS2,EPS3,EPS4,IO2,MAX
0007            N1=N
0008            M1=M
0009            KKK=N+1
0010            DO 20 I=1,KKK
0011            URR(I)=UR(I)
0012         20 VRR(I)=VR(I)
0013            KKK=M+1
0014            DO 25 I=1,KKK
0015            USS(I)=US(I)
0016         25 VSS(I)=VS(I)
0017         30 BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0018            UD=(URR(N1+1)*USS(M1+1)+VRR(N1+1)*VSS(M1+1))/BBB
0019            VD=(USS(M1+1)*VRR(N1+1)-URR(N1+1)*VSS(M1+1))/BBB
0020            KKK=N1+1-M1
0021            DO 40 I=KKK,N1
0022            UT(I)=URR(I)-(UD*USS(I-N1+M1)-VD*VSS(I-N1+M1))
0023         40 VT(I)=VRR(I)-(UD*VSS(I-N1+M1)+VD*USS(I-N1+M1))
0024            IF(M1.EQ.N1) GO TO 70
0025            KKK=N1-M1
0026            DO 60 I=1,KKK
0027            UT(I)=URR(I)
0028         60 VT(I)=VRR(I)
0029         70 DO 90 I=1,N1
0030            BBB=DSQRT(UT(N1+1-I)*UT(N1+1-I)+VT(N1+1-I)*VT(N1+1-I))
0031            IF(BBB.GT.EPSLON) GO TO 100
0032         90 CONTINUE
0033            DO 95 I=1,M1
0034            BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0035            USSSSS=(USS(I)*USS(M1+1)+VSS(I)*VSS(M1+1))/BBB
0036            VSSSSS=(VSS(I)*USS(M1+1)-USS(I)*VSS(M1+1))/BBB
0037            USS(I)=USSSSS
0038         95 VSS(I)=VSSSSS
0039            USS(M1+1)=1.0
0040            VSS(M1+1)=0.0
0041            GO TO 200
0042        100 K=N1-I
0043            IF(K.EQ.0) GO TO 170
0044            IF(K.LT.M1) GO TO 140
0045            KKK=K+1
0046            DO 130 J=1,KKK
0047            URR(J)=UT(J)
0048        130 VRR(J)=VT(J)
0049            N1=K
```

TABLE F.III (Continued)

```
0050              GO TO 30
0051          140 KKK=K+1
0052              DO 150 J=1,KKK
0053              URR(J)=USS(J)
0054              VRR(J)=VSS(J)
0055              USS(J)=UT(J)
0056          150 VSS(J)=VT(J)
0057              KKK=K+2
0058              NNN=M1+1
0059              DO 160 J=KKK,NNN
0060              URR(J)=USS(J)
0061          160 VRR(J)=VSS(J)
0062              N1=M1
0063              M1=K
0064              GO TO 30
0065          170 USS(1)=1.0
0066              VSS(1)=0.0
0067              M1=0
0068          200 RETURN
0069              END
```

TABLE F.III (Continued)

```
0001            SUBROUTINE QUAD(N,UA,VA,J,UROOT,VROOT,MULT)
              C ********************************************************************************
              C *                                                                              *
              C * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES       *
              C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE        *
              C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                               *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION EPSRT
0003            DOUBLE PRECISION UA,VA,UROOT,VROOT,UDISC,VDISC,UTEMP,VTEMP,UD,VD,E
               1PSI,EPS2,EPS4,EPSLON,BBB
0004            DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULT(25)
0005            COMMON EPSRT,EPS1,EPS2,EPSLON,EPS4,IO2,MAX
0006            IF(N.GT.1) GO TO 60
0007            IF(J.LT.0) GO TO 40
0008            J=J+1
0009            GO TO 50
0010          40 MULT(1)=1
0011            J=1
0012          50 BBB=UA(2)*UA(2)+VA(2)*VA(2)
0013            UROOT(J)=-(UA(1)*UA(2)+VA(1)*VA(2))/BBB
0014            VROOT(J)=-(VA(1)*UA(2)-UA(1)*VA(2))/BBB
0015            GO TO 200
0016          60 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(3)*UA(1)-VA(3)*VA(1)))
0017            VDISC=(2.0*UA(2)*VA(2))-(4.0*(UA(3)*VA(1)+VA(3)*UA(1)))
0018            BBB=DSQRT(UDISC*UDISC+VDISC*VDISC)
0019            IF(BBB.LE.EPSLON) GO TO 100
0020            IF(J.GE.0) GO TO 80
0021            MULT(1)=1
0022            MULT(2)=1
0023            J=0
0024          80 CALL COMSQT(UDISC,VDISC,UTEMP,VTEMP)
0025            UD=2.0*UA(3)
0026            VD=2.0*VA(3)
0027            BBB=UD*UD+VD*VD
0028            UROOT(J+1)=((-UA(2)+UTEMP)*UD+(-VA(2)+VTEMP)*VD)/BBB
0029            VROOT(J+1)=((-VA(2)+VTEMP)*UD-(-UA(2)+UTEMP)*VD)/BBB
0030            UROOT(J+2)=((-UA(2)-UTEMP)*UD+(-VA(2)-VTEMP)*VD)/BBB
0031            VROOT(J+2)=((-VA(2)-VTEMP)*UD-(-UA(2)-UTEMP)*VD)/BBB
0032            J=J+2
0033            GO TO 200
0034         100 IF(J.LT.0) GO TO 110
0035            J=J+1
0036            GO TO 130
0037         110 MULT(1)=2
0038            J=1
0039         130 UD=2.0*UA(3)
0040            VD=2.0*VA(3)
0041            BBB=UD*UD+VD*VD
0042            UROOT(J)=(-UA(2)*UD-VA(2)*VD)/BBB
0043            VROOT(J)=(-VA(2)*UD+UA(2)*VD)/BBB
0044         200 RETURN
0045            END
```

TABLE F.III (Continued)

```
0001            SUBROUTINE MULLER(UA,VA,NP,UAPP,VAPP,NAPP,XSTART,XEND,UROOT,VROOT,
               1NROOT,IROOT,URAPP,VRAPP,NOPOLY)
          C    ************************************************************************
          C    *                                                                      *
          C    * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A     *
          C    * POLYNOMIAL OF MAXIMUM DEGREE 25.  THROUGH THREE GIVEN POINTS THE     *
          C    * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC.  THE ZERO OF THE QUADRATIC *
          C    * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.  *
          C    * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.          *
          C    *                                                                      *
          C    ************************************************************************
0002            DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,VX1,UAPP,VAPP
               1,UX2,VX2,UWORK,VWORK,UX3,VX3,UB,VB,UX4,VX4,UA,VA,UPX1,VPX1,URAPP,V
               2RAPP,UPX4,VPX4,EPSRT,EPSO,EPS,CCC,EPSM,UH3,VH3,UQ4,VQ4,ABPX4,ABPX3
               3,QQQ,XSTART,XEND
0003            DIMENSION UROOT(25),VROOT(25),NULT(25),UAPP(25,3),VAPP(25,3),UWORK
               1(26),VWORK(26),UB(26),VB(26),UA(26),VA(26),URAPP(25,3),VRAPP(25,3)
0004            LOGICAL CONV
0005            DOUBLE PRECISION EPS1
0006            COMMON EPSRT,EPS1,EPS,EPSO,EPSM,IO2,MAX
0007            DATA PNAME,DNAME/2HP(,2HD(/
0008            EPSRT=0.999
0009            NROOT=0
0010            IROOT=0
0011            IPATH=1
0012            NOMULT=0
0013            NALTER=0
0014            ITIME=0
0015            IAPP=1
0016            ITER=1
0017            IF(NAPP.NE.0) GO TO 18
0018            NAPP=NP
0019            CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0020            GO TO 27
0021         18 DO 25 I=1,NAPP
0022            UAPP(I,1)=0.9*UAPP(I,2)
0023            VAPP(I,1)=0.9*VAPP(I,2)
0024            UAPP(I,3)=1.1*UAPP(I,2)
0025         25 VAPP(I,3)=1.1*VAPP(I,2)
0026         27 KKK=NP+1
0027            DO 30 I=1,KKK
0028            UWORK(I)=UA(I)
0029         30 VWORK(I)=VA(I)
0030            NWORK=NP
0031         40 UX1=UAPP(IAPP,1)
0032            VX1=VAPP(IAPP,1)
0033            UX2=UAPP(IAPP,2)
0034            VX2=VAPP(IAPP,2)
0035            UX3=UAPP(IAPP,3)
0036            VX3=VAPP(IAPP,3)
0037            CALL HORNER(NWORK,UWORK,VWORK,UX1,VX1,UB,VB,UPX1,VPX1)
0038            CALL HORNER(NWORK,UWORK,VWORK,UX2,VX2,UB,VB,UPX2,VPX2)
0039            CALL HORNER(NWORK,UWORK,VWORK,UX3,VX3,UB,VB,UPX3,VPX3)
0040         50 CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
               14,VX4,UQ4,VQ4,UH3,VH3)
0041         60 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0042            ABPX4=DSQRT(UPX4*UPX4+VPX4*VPX4)
0043            ABPX3=DSQRT(UPX3*UPX3+VPX3*VPX3)
```

TABLE F.III (Continued)

```
0044              IF(ABPX3.EQ.0.0) GO TO 70
0045              QQQ=ABPX4/ABPX3
0046              IF(QQQ.LE.10.) GO TO 70
0047              UQ4=0.5*UQ4
0048              VQ4=0.5*VQ4
0049              UX4=UX3+(UH3*UQ4-VH3*VQ4)
0050              VX4=VX3+(VH3*UQ4+UH3*VQ4)
0051              GO TO 60
0052           70 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0053              IF(CONV) GO TO 120
0054              IF(ITER.LT.MAX) GO TO 110
0055              CALL ALTER(UAPP(IAPP,1),VAPP(IAPP,1),UAPP(IAPP,2),VAPP(IAPP,2),UAP
                 1P(IAPP,3),VAPP(IAPP,3),NALTER,ITIME)
0056              IF(NALTER.GT.5) GO TO 75
0057              ITER=1
0058              GO TO 40
0059           75 IF(IAPP.LT.NAPP) GO TO 100
0060              IF(XEND.EQ.0.0) GO TO 77
0061              IF(XSTART.GT.XEND) GO TO 77
0062              NAPP=NP
0063              CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0064              IAPP=0
0065              GO TO 100
0066           77 WRITE(IO2,1090)
0067              KKK=NWORK+1
0068              WRITE(IO2,1035) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0069           80 IF(NROOT.EQ.0) GO TO 90
0070              IF(IPATH.EQ.1) GO TO 82
0071           81 IPATH=2
0072              CALL BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MULT)
0073              RETURN
0074           82 IF(NROOT.EQ.0)GO TO 90
0075              IF(IROOT.EQ.0) GO TO 85
0076              WRITE(IO2,1080)
0077              DO 55 I=1,IROOT
0078           55 WRITE(IO2,1085) I,UROOT(I),VROOT(I),URAPP(I,2),VRAPP(I,2)
0079              IF(IROOT.LT.NROOT) GO TO 85
0080              GO TO 87
0081           85 KKK=IROOT+1
0082              WRITE(IO2,1086) (I,UROOT(I),VROOT(I),I=KKK,NROOT)
0083           87 IF(IPATH.EQ.1) GO TO 81
0084              RETURN
0085           90 WRITE(IO2,1070) NOPOLY
0086              RETURN
0087          100 IAPP=IAPP+1
0088              ITER=1
0089              NALTER=0
0090              GO TO 40
0091          120 NROOT=NROOT+1
0092              IROOT=NROOT
0093              MULT(NROOT)=1
0094              NOMULT=NOMULT+1
0095              UROOT(NROOT)=UX4
0096              VROOT(NROOT)=VX4
0097              URAPP(NROOT,1)=UAPP(IAPP,1)
0098              VRAPP(NROOT,1)=VAPP(IAPP,1)
0099              URAPP(NROOT,2)=UAPP(IAPP,2)
0100              VRAPP(NROOT,2)=VAPP(IAPP,2)
```

TABLE F.III (Continued)

```
0101              URAPP(NROOT,3)=UAPP(IAPP,3)
0102              VRAPP(NROOT,3)=VAPP(IAPP,3)
0103            125 IF(NOMULT.LT.NP) GO TO 130
0104              GO TO 80
0105            130 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0106              NWORK=NWORK-1
0107              KKK=NWORK+1
0108              DO 140 I=1,KKK
0109              UWORK(I)=UB(I)
0110            140 VWORK(I)=VB(I)
0111              CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0112              CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
0113              IF(CCC.LT.EPSM) GO TO 150
0114              IF(NWORK.GT.2) GO TO 75
0115              IROOT=NROOT
0116              KKK=NWORK+1
0117              DO 145 I=1,KKK
0118              UB(I)=UWORK(KKK+1-I)
0119            145 VB(I)=VWORK(KKK+1-I)
0120              CALL QUAD(NWORK,UB,VB,NROOT,UROOT,VROOT,MULT)
0121              GO TO 80
0122            150 MULT(NROOT)=MULT(NROOT)+1
0123              NOMULT=NOMULT+1
0124              GO TO 125
0125            110 UX1=UX2
0126              VX1=VX2
0127              UX2=UX3
0128              VX2=VX3
0129              UX3=UX4
0130              VX3=VX4
0131              UPX1=UPX2
0132              VPX1=VPX2
0133              UPX2=UPX3
0134              VPX2=VPX3
0135              UPX3=UPX4
0136              VPX3=VPX4
0137              ITER=ITER+1
0138              GO TO 50
0139           1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
                  1ZEROS WERE FOUND//)
0140           1080 FORMAT(///1X,13HROOTS OF Q(X),83X,21HINITIAL APPROXIMATION//)
0141           1070 FORMAT(///,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
0142           1086 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,19X,23HSOLVED
                  1 BY DIRECT METHOD)
0143           1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
0144           1050 FORMAT(82X,D23.16,3H + ,D23.16,2H I/82X,D23.16,3H + ,D23.16,2H I/)
0145           1085 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,18X,D23.16,3H
                  1 + ,D23.16,2H I)
0146              END
```

TABLE F.III (Continued)

```
0001              SUBROUTINE BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MUL
                 1LT)
                C  ***************************************************************************
                C  *                                                                         *
                C  * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND   *
                C  * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO  *
                C  * THE FULL, UNDEFLATED POLYNOMIAL.                                       *
                C  *                                                                         *
                C  ***************************************************************************
0002              DOUBLE PRECISION UROOT,VROOT,UA,VA,UBAPP,VBAPP,UX1,VX1,UX2,VX2,UX3
                 1,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UB,VB,UROOTS,VROOTS,EPSRT,UX4,V
                 2X4,URAPP,VRAPP,EPSO,EPS,UQ4,VQ4,UH3,VH3
0003              LOGICAL CONV
0004              DIMENSION UROOT(25),VROOT(25),UA(26),VA(26),UBAPP(25,3),VBAPP(25,3
                 1),UB(26),VB(26),UROOTS(25),VROOTS(25),URAPP(25,3),VRAPP(25,3),MULT
                 3(25)
0005              DOUBLE PRECISION EPS1,EPSM
0006              COMMON EPSRT,EPS1,EPS,EPSO,EPSM,IO2,MAX
0007              IF(NROOT.LE.1) RETURN
0008              L=0
0009              DO 10 I=1,NROOT
0010              UBAPP(I,1)=UROOT(I)*EPSRT
0011              VBAPP(I,1)=VROOT(I)*EPSRT
0012              UBAPP(I,2)=UROOT(I)
0013              VBAPP(I,2)=VROOT(I)
0014              UBAPP(I,3)=UROOT(I)*(2.0-EPSRT)
0015             10 VBAPP(I,3)=VROOT(I)*(2.0-EPSRT)
0016              DO 100 J=1,NROOT
0017              UX1=UBAPP(J,1)
0018              VX1=VBAPP(J,1)
0019              UX2=UBAPP(J,2)
0020              VX2=VBAPP(J,2)
0021              UX3=UBAPP(J,3)
0022              VX3=VBAPP(J,3)
0023              ITER=1
0024              CALL HORNER(NP,UA,VA,UX1,VX1,UB,VB,UPX1,VPX1)
0025              CALL HORNER(NP,UA,VA,UX2,VX2,UB,VB,UPX2,VPX2)
0026             20 CALL HORNER(NP,UA,VA,UX3,VX3,UB,VB,UPX3,VPX3)
0027              CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
                 14,VX4,UQ4,VQ4,UH3,VH3)
0028             30 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0029              IF(CONV) GO TO 50
0030              IF(ITER.LT.MAX) GO TO 40
0031              WRITE(IO2,1000) J,UROOT(J),VROOT(J),MAX
0032              WRITE(IO2,1010) UX4,VX4
0033              IF(J.LT.IROOT) GO TO 33
0034              IF(J.EQ.IROOT) GO TO 35
0035              GO TO 100
0036             33 KKK=IROOT-1
0037              DO 34 K=J,KKK
0038              URAPP(K,1)=URAPP(K+1,1)
0039              VRAPP(K,1)=VRAPP(K+1,1)
0040              URAPP(K,2)=URAPP(K+1,2)
0041              VRAPP(K,2)=VRAPP(K+1,2)
0042              URAPP(K,3)=URAPP(K+1,3)
0043             34 VRAPP(K,3)=VRAPP(K+1,3)
0044             35 IROOT=IROOT-1
0045              GO TO 100
```

TABLE F.III (Continued)

```
0046          40 UX1=UX2
0047             VX1=VX2
0048             UX2=UX3
0049             VX2=VX3
0050             UX3=UX4
0051             VX3=VX4
0052             UPX1=UPX2
0053             VPX1=VPX2
0054             UPX2=UPX3
0055             VPX2=VPX3
0056             ITER=ITER+1
0057             GO TO 20
0058          50 L=L+1
0059             UROOTS(L)=UX4
0060             VROOTS(L)=VX4
0061         100 CONTINUE
0062             IF(L.EQ.0) GO TO 120
0063             DO 110 I=1,L
0064             UROOT(I)=UROOTS(I)
0065         110 VROOT(I)=VROOTS(I)
0066             NROOT=L
0067             RETURN
0068         120 NROOT=0
0069             RETURN
0070        1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
                1D23.16,3H + ,D23.16,2H I/24H DID NOT CONVERGE AFTER ,I3,11H ITERAT
                2IONS)
0071        1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,D23.16,3H + ,D23.16,2H I/
                1/)
0072             END
```

TABLE F.III (Continued)

```
0001            SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)
              C  ************************************************************************
              C  *                                                                      *
              C  * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO  *
              C  * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
              C  *                                                                      *
              C  ************************************************************************
0002            DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA
0003            DOUBLE PRECISION EPS4,EPS5
0004            COMMON EPS1,EPS2,EPS3,EPS4,EPS5,IO2,MAX
0005            IF(ITIME.NE.0) GO TO 5
0006            ITIME=1
0007            WRITE(IO2,1010) MAX
0008          5 IF(NALTER.EQ.0) GO TO 10
0009            WRITE(IO2,1000) X1R,X1I,X2R,X2I,X3R,X3I
0010            GO TO 20
0011         10 R=DSQRT(X2R*X2R+X2I*X2I)
0012            BETA=DATAN2(X2I,X2R)
0013            WRITE(IO2,1020) X1R,X1I,X2R,X2I,X3R,X3I
0014         20 NALTER=NALTER+1
0015            IF(NALTER.GT.5) RETURN
0016            GO TO (30,40,30,40,30),NALTER
0017         30 X2R=-X2R
0018            X2I=-X2I
0019            GO TO 50
0020         40 BETA=BETA+1.0471976
0021            X2R=R*DCOS(BETA)
0022            X2I=R*DSIN(BETA)
0023         50 X1R=0.9*X2R
0024            X1I=0.9*X2I
0025            X3R=1.1*X2R
0026            X3I=1.1*X2I
0027            RETURN
0028       1000 FORMAT(1X,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HALTERED APPROXIM
                1ATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
                2,D23.16,2H I/)
0029       1020 FORMAT(1H0,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HINITIAL APPROXI
                1MATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
                2 ,D23.16,2H I/)
0030       1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
                1TER ,I3,12H ITERATIONS.//)
0031            END
```

TABLE F.III (Continued)

```
0001              SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
                C ********************************************************************************
                C *                                                                              *
                C * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE         *
                C * DEGREE OF THE ORIGINAL POLYNOMIAL.                                           *
                C *                                                                              *
                C ********************************************************************************
0002              DOUBLE PRECISION APPR,APPI,XSTART,EPS1,EPS2,EPS3,BETA
0003              DOUBLE PRECISION EPSRT,EPS4
0004              DIMENSION APPR(25,3),APPI(25,3)
0005              COMMON EPSRT,EPS1,EPS2,EPS3,EPS4,IO2,MAX
0006              IF(XSTART.EQ.0.0) XSTART=0.5
0007              BETA=0.2617994
0008              DO 10 I=1,NAPP
0009              APPR(I,2)=XSTART*DCOS(BETA)
0010              APPI(I,2)=XSTART*DSIN(BETA)
0011              BETA=BETA+0.5235988
0012           10 XSTART=XSTART+0.5
0013              DO 20 I=1,NAPP
0014              APPR(I,1)=0.9*APPR(I,2)
0015              APPI(I,1)=0.9*APPI(I,2)
0016              APPR(I,3)=1.1*APPR(I,2)
0017           20 APPI(I,3)=1.1*APPI(I,2)
0018              RETURN
0019              END
```

TABLE F.III (Continued)

```
0001              SUBROUTINE TEST(UX3,VX3,UX4,VX4,CONV)
                C ***************************************************************************
                C *                                                                         *
                C * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-       *
                C * IMATIONS BY TESTING THE EXPRESSION                                      *
                C * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).              *
                C * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.               *
                C *                                                                         *
                C ***************************************************************************
0002              DOUBLE PRECISION UX3,VX3,UX4,VX4,EPSRT,EPSO,EPS,AAA,UDUMMY,VDUMMY,
                 1DENOM
0003              LOGICAL CONV
0004              DOUBLE PRECISION EPS1,EPSM
0005              COMMON EPSRT,EPS1,EPS,EPSO,EPSM,IO2,MAX
0006              UDUMMY=UX4-UX3
0007              VDUMMY=VX4-VX3
0008              AAA=DSQRT(UDUMMY*UDUMMY+VDUMMY*VDUMMY)
0009              DENOM=DSQRT(UX4*UX4+VX4*VX4)
0010              IF(DENOM.LT.EPSO) GO TO 20
0011              IF(AAA/DENOM.LT.EPS) GO TO 10
0012            5 CONV=.FALSE.
0013              GO TO 100
0014           10 CONV=.TRUE.
0015              GO TO 100
0016           20 IF(AAA.LT.EPSO) GO TO 10
0017              GO TO 5
0018          100 RETURN
0019              END
```

```
0001              SUBROUTINE HORNER(NA,UA,VA,UX,VX,UB,VB,UPX,VPX)
                C ***************************************************************************
                C *                                                                         *
                C * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D. *
                C * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE *
                C * FACTOR (X-D).                                                           *
                C *                                                                         *
                C ***************************************************************************
0002              DOUBLE PRECISION UX,VX,UPX,VPX,UB,VB,UA,VA
0003              DIMENSION UA(26),VA(26),UB(26),VB(26)
0004              UB(1)=UA(1)
0005              VB(1)=VA(1)
0006              NUM=NA+1
0007              DO 10 I=2,NUM
0008              UB(I)=UA(I)+(UB(I-1)*UX-VB(I-1)*VX)
0009           10 VB(I)=VA(I)+(VB(I-1)*UX+UB(I-1)*VX)
0010              UPX=UB(NUM)
0011              VPX=VB(NUM)
0012              RETURN
0013              END
```

TABLE F.III (Continued)

```
0001              SUBROUTINE CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,V
                C  ********************************************************************************
                C  *                                                                              *
                C  * GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC         *
                C  * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF        *
                C  * THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION           *
                C  * X(N+1) TO THE ZERO OF THE POLYNOMIAL.                                       *
                C  *                                                                              *
                C  ********************************************************************************
                 1PX3,UX4,VX4,UQ4,VQ4,UH3,VH3)
0002              DOUBLE PRECISION ARG1,ARG2
0003              DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1,
                 1VPX1,UH3,VH3,UH2,VH2,UQ3,VQ3,UD,VD,UB,VB,UC,VC,UDISC,VDISC,UCCC,VC
                 2CC,UDEN1,VDEN1,UDEN2,VDEN2,UQ4,VQ4,UX4,VX4,EPSRT,EPSO,EPS,UDDD,VDD
                 3D,AAA,BBB,RAD,UAAA,VAAA,UBBB,VBBB
0004              DOUBLE PRECISION THETA,ANGLE,UTEST,VTEST
0005              DOUBLE PRECISION EPS1,EPSM
0006              COMMON EPSRT,EPS1,EPS,EPSO,EPSM,IO2,MAX
0007              UH3=UX3-UX2
0008              VH3=VX3-VX2
0009              UH2=UX2-UX1
0010              VH2=VX2-VX1
0011              BBB=UH2*UH2+VH2*VH2
0012              UQ3=(UH3*UH2+VH3*VH2)/BBB
0013              VQ3=(VH3*UH2-UH3*VH2)/BBB
0014              UDDD=1.0+UQ3
0015              VDDD=VQ3
0016              UD=(UPX3-(UDDD*UPX2-VDDD*VPX2))+(UQ3*UPX1-VQ3*VPX1)
0017              VD=(VPX3-(VDDD*UPX2+UDDD*VPX2))+(VQ3*UPX1+UQ3*VPX1)
0018              UAAA=2.0*UQ3
0019              VAAA=2.0*VQ3
0020              UAAA=UAAA+1.0
0021              UBBB=UDDD*UDDD-VDDD*VDDD
0022              VBBB=VDDD*UDDD+UDDD*VDDD
0023              UCCC=UQ3*UQ3-VQ3*VQ3
0024              VCCC=VQ3*UQ3+UQ3*VQ3
0025              UB=((UAAA*UPX3-VAAA*VPX3)-(UBBB*UPX2-VBBB*VPX2))+(UCCC*UPX1-VCCC*V
                 1PX1)
0026              VB=((VAAA*UPX3+UAAA*VPX3)-(VBBB*UPX2+UBBB*VPX2))+(VCCC*UPX1+UCCC*V
                 1PX1)
0027              UC=UDDD*UPX3-VDDD*VPX3
0028              VC=VDDD*UPX3+UDDD*VPX3
0029              UDISC=(UB*UB-VB*VB)-(4.0*(UD*UC-VD*VC))
0030              VDISC=(2.0*(VB*UB))-(4.0*(VD*UC+UD*VC))
0031              AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
0032              IF(AAA.EQ.0.0) GO TO 5
0033              GO TO 7
0034             5 THETA=0.0
0035              GO TO 9
0036             7 THETA=DATAN2(VDISC,UDISC)
0037             9 RAD=DSQRT(AAA)
0038              ANGLE=THETA/2.0
0039              UTEST=RAD*DCOS(ANGLE)
0040              VTEST=RAD*DSIN(ANGLE)
0041              UDEN1=UB+UTEST
0042              VDEN1=VB+VTEST
0043              UDEN2=UB-UTEST
0044              VDEN2=VB-VTEST
```

TABLE F.III (Continued)

```
0045              ARG1=UDEN1*UDEN1+VDEN1*VDEN1
0046              ARG2=UDEN2*UDEN2+VDEN2*VDEN2
0047              AAA=DSQRT(ARG1)
0048              BBB=DSQRT(ARG2)
0049              IF(AAA.LT.BBB) GO TO 10
0050              IF(AAA.EQ.0.0) GO TO 60
0051              UAAA=-2.0*UC
0052              VAAA=-2.0*VC
0053              UQ4=(UAAA*UDEN1+VAAA*VDEN1)/ARG1
0054              VQ4=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
0055              GO TO 50
0056           10 IF(BBB.EQ.0.0) GO TO 60
0057              UAAA=-2.0*UC
0058              VAAA=-2.0*VC
0059              UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2
0060              VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2
0061              GO TO 50
0062           50 UX4=UX3+(UH3*UQ4-VH3*VQ4)
0063              VX4=VX3+(VH3*UQ4+UH3*VQ4)
0064              RETURN
0065           60 UQ4=1.0
0066              VQ4=0.0
0067              GO TO 50
0068              END
```

TABLE F.III (Continued)

```
0001            SUBROUTINE COMSQT(UX,VX,UY,VY)
                C  ***************************************************************************
                C  *                                                                         *
                C  * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.           *
                C  *                                                                         *
                C  ***************************************************************************
0002            DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
0003            R=DSQRT(UX*UX+VX*VX)
0004            AAA=DSQRT(DABS((R+UX)/2.0))
0005            BBB=DSQRT(DABS((R-UX)/2.0))
0006            IF(VX) 10,20,30
0007          10 UY=AAA
0008             VY=-1.0*BBB
0009             GO TO 100
0010          20 IF(UX) 40,50,60
0011          30 UY=AAA
0012             VY=BBB
0013             GO TO 100
0014          40 DUMMY=DABS(UX)
0015             UY=0.0
0016             VY=DSQRT(DUMMY)
0017             GO TO 100
0018          50 UY=0.0
0019             VY=0.0
0020             GO TO 100
0021          60 DUMMY=DABS(UX)
0022             UY=DSQRT(DUMMY)
0023             VY=0.0
0024         100 RETURN
0025             END
```

APPENDIX G

REPEATED G.C.D. - NEWTON'S METHOD

1. Use of the Program

A double precision FORTRAN IV program using the repeated G.C.D. method with Newton's method as a supporting method is presented here. Flow charts for this program are given in Figure G.2 while Table G.III gives a FORTRAN IV listing of this program. Single precision variables are listed in Table G.II. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from Table G.II.

This program is designed to solve polynomials having degree less than or equal to 25. In order to solve polynomials of degree N where $N > 25$, the data statement and array dimensions given in Table G.I must be changed.

In this program both the leading coefficient and the constant coefficient are assumed to be non-zero.

TABLE G.I

PROGRAM CHANGES NECESSARY TO SOLVE POLYNOMIALS OF DEGREE GREATER THAN 25 BY THE REPEATED G.C.D. - NEWTON'S METHOD

Main Program

Data Entry/1H1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1
UP(N+1), VP(N+1)
UAPP(N), VAPP(N)
UD0(N+1), VD0(N+1)
UDD0(N+1), VDD0(N+1)
UD1(N+1), VD1(N+1)
UD2(N+1), VD2(N+1)
UDD1(N+1), VDD1(N+1)
UG(N+1), VG(N+1)
UD3(2N+1), VD3(2N+1)
UD4(2N+1), VD4(2N+1)
UZROS(N), VZROS(N)
UAP(N), VAP(N)
UROOT(N), VROOT(N)
NULT(N)
ENTRY(N+1)

Subroutine PROD

UH(2N+1), VH(2N+1)
UF(N+1), VF(N+1)
UG(N+1), VG(N+1)

Subroutine ZROS

UAPP(N), VAPP(N)
UROOT(N), VROOT(N)
UQ(N+1), VQ(N+1)
UQQ(N+1), VQQ(N+1)
UAP(N), VAP(N)
UQD(N+1), VQD(N+1)
ENTRY(N+1)
UROOTS(N), VROOTS(N)

Subroutines GENAPP, GCD, NEWTON, DIVIDE, HORNER, and DERIV

See corresponding subroutine in Table E.I.

Subroutine QUAD

UROOT(N), VROOT(N)
UA(N+1), VA(N+1)

2. Input Data for Repeated G.C.D. - Newton's Method

The input data for repeated G.C.D. - Newton's method is prepared as described for G.C.D. - Newton's method in Appendix E, § 2 except that the item EPS4 on the control card (Figure E.2) is omitted. An example control card for the repeated G.C.D. - Newton's method is given in Figure G.1.

3. Variables Used in Repeated G.C.D. - Newton's Method

The definitions of variables used in repeated G.C.D. - Newton's method are given in Table G.II. For definitions of variables not listed in this table, see the main program or corresponding subprogram of Table E.VI. The notation and symbols used are defined in Appendix E, § 3.

4. Description of Program Output

The number of the polynomial, control data, degree and coefficients of the polynomial are printed as described in Appendix E, § 4.

All roots of multiplicity one are extracted first. Following the first row of asterixes, the message "THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY 1." This is followed by the coefficients of G(X) with the leading coefficient listed first. If there are no roots of multiplicity one, then the message "NO ROOTS OF MULTIPLICITY ONE" is printed.

The roots of G(X) are printed under the heading "ROOTS OF G(X)." These are the roots obtained before the attempt to improve accuracy. The initial approximations producing convergence to the corresponding root are printed under the heading "INITIAL APPROXIMATION." The

message "RESULTS OF SUBROUTINE QUAD" means that the corresponding root was obtained from subroutine QUAD.

The roots found as a result of attempting to improve accuracy are printed under the heading "ROOTS OF P(X)." Their multiplicity is given under the heading "MULTIPLICITIES." The initial approximation is printed above where "NO INITIAL APPROXIMATION" means the same as "RESULTS OF SUBROUTINE QUAD."

A line of asterixes is then printed. This procedure is then repeated for the roots of multiplicity 2,3,4, etc. until all roots have been found.

5. Informative Messages and Error Messages

The informative messages and error messages for repeated G.C.D. - Newton's method are given below. For those not listed, see Appendix E, § 5.

"NOT ALL ROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUND." This message indicates that some of the roots of the polynomial G(X) were not extracted.

"QUAD FOUND XXX TO BE A MULTIPLE ROOT." XXX represents the value of the root found as a multiple root by Subroutine QUAD.

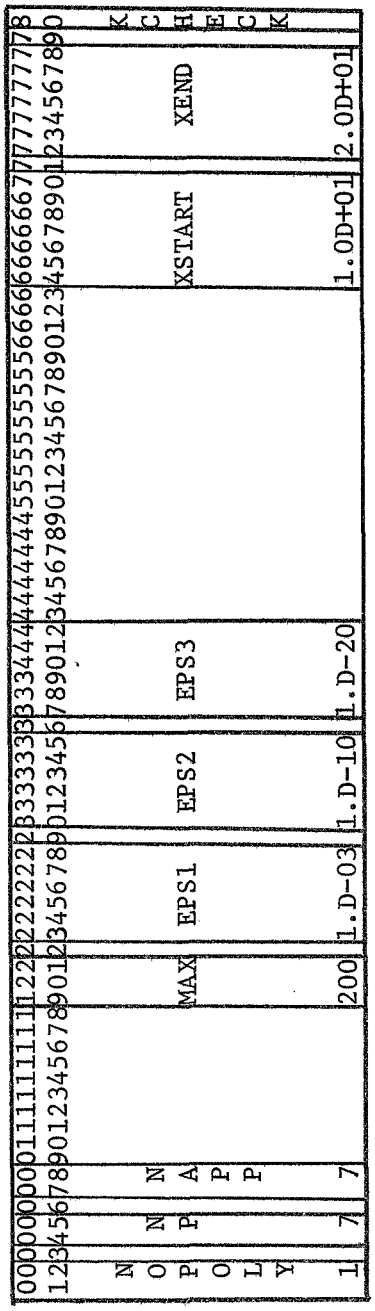


Figure G.1 Control Card for Repeated G.C.D. - Newton's Method

TABLE G.II

REPEATED GCD - NEWTON'S METHOD

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument		Description
				Main Program		
KD	I	KD	I			Number of distinct roots found
K	I	K	I			Number of roots found
J1	I	J1	I			Multiplicity of given root
D0	C	UD0,VD0	D			Array of coefficients of original polynomial
ND0	I	ND0	I			Degree of original polynomial
DD0	C	UDD0,VDD0	D			Array of coefficients of derivative of D0(X) i.e. D0'(X)
NDD0	I	NDD0	I			Degree of DD0(X)
D1	C	UD1,VD1	D			Array of coefficients of g.c.d. of D0(X) and DD0(X)
ND1	I	ND1	I			Degree of D1(X)
DD1	C	UDD1,VDD1	D			Array of coefficients of derivative of D1(X) i.e. D1'(X)
NDD1	I	NDD1	I			Degree of DD1(X)
D2	C	UD2,VD2	D			Array of coefficients of g.c.d. of D1(X) and DD1(X)
ND2	I	ND2	I			Degree of D2(X)
D3	C	UD3,VD3	D			Array of coefficients of the product of D0(X) and D2(X)
ND3	I	ND3	I			Degree of D3(X)
D4	C	UD4,VD4	D			Array of coefficients of the square of D1(X)
ND4	I	ND4	I			Degree of D4(X)
G	C	UG,VG	D			Array of coefficients of the quotient D3(X)/D4(X)
NG	I	NG	I			Degree of G(X)
ZROS	C	UZROS,VZROS	D			Array of roots of G(X)
				Subroutine ZROS		
APROX	C	UAPROX,VAPROX	D		R	Starting approximation (initial or altered)

TABLE G.II (Continued)

Single Precision Variable	Type	Double Precision Variable	Type	Disposition of Argument	Description
					Subroutine PROD
M	I	M	I	E	Degree of polynomial to be multiplied
F	C	UF,VF	D	E	Array of coefficients of polynomial to be multiplied
N	I	N	I	E	Degree of polynomial to be multiplied
G	C	UG,VG	D	E	Array of coefficients of polynomial to be multiplied
MN	I	MN	I	R	Degree of product polynomial H(X)
H	C	UH,VH	D	R	Array of coefficients of product polynomial
LIMIT	I	LIMIT	I		Number of coefficients of polynomial F(X)
K	I	K	I		Counter

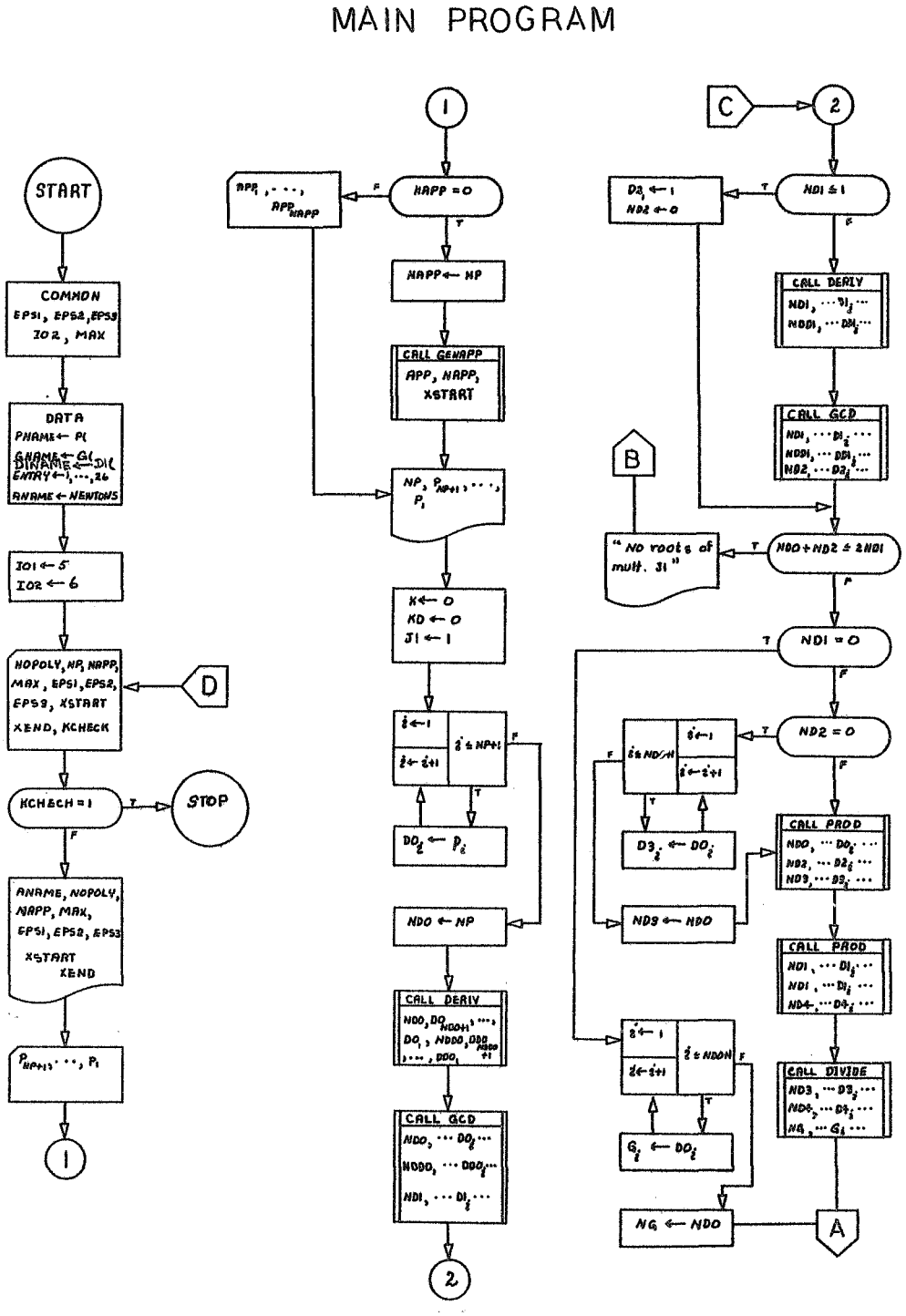


Figure G.2. Flow Charts for Repeated G.C.D.-Newton's Method

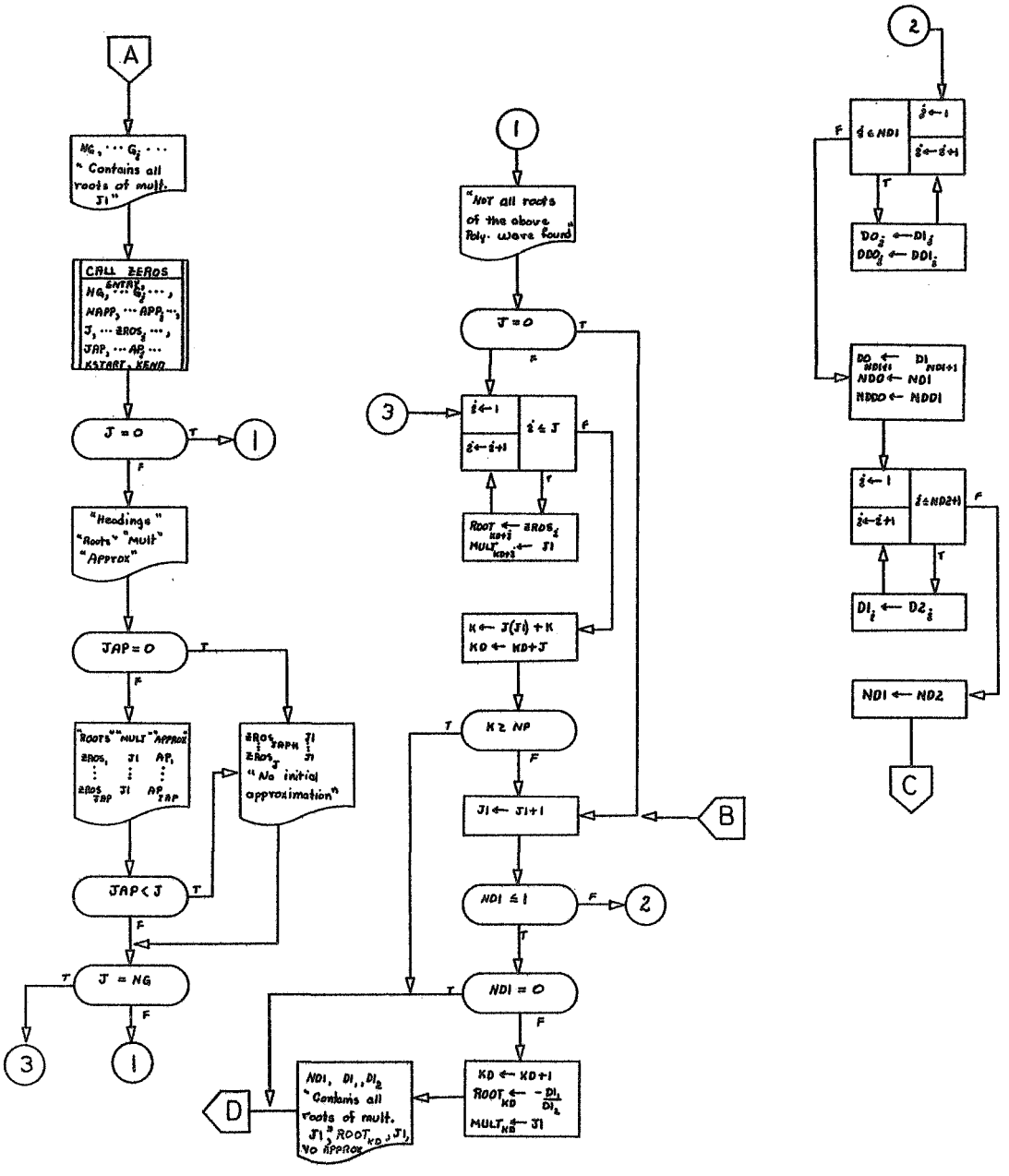


Figure G.2. (Continued)

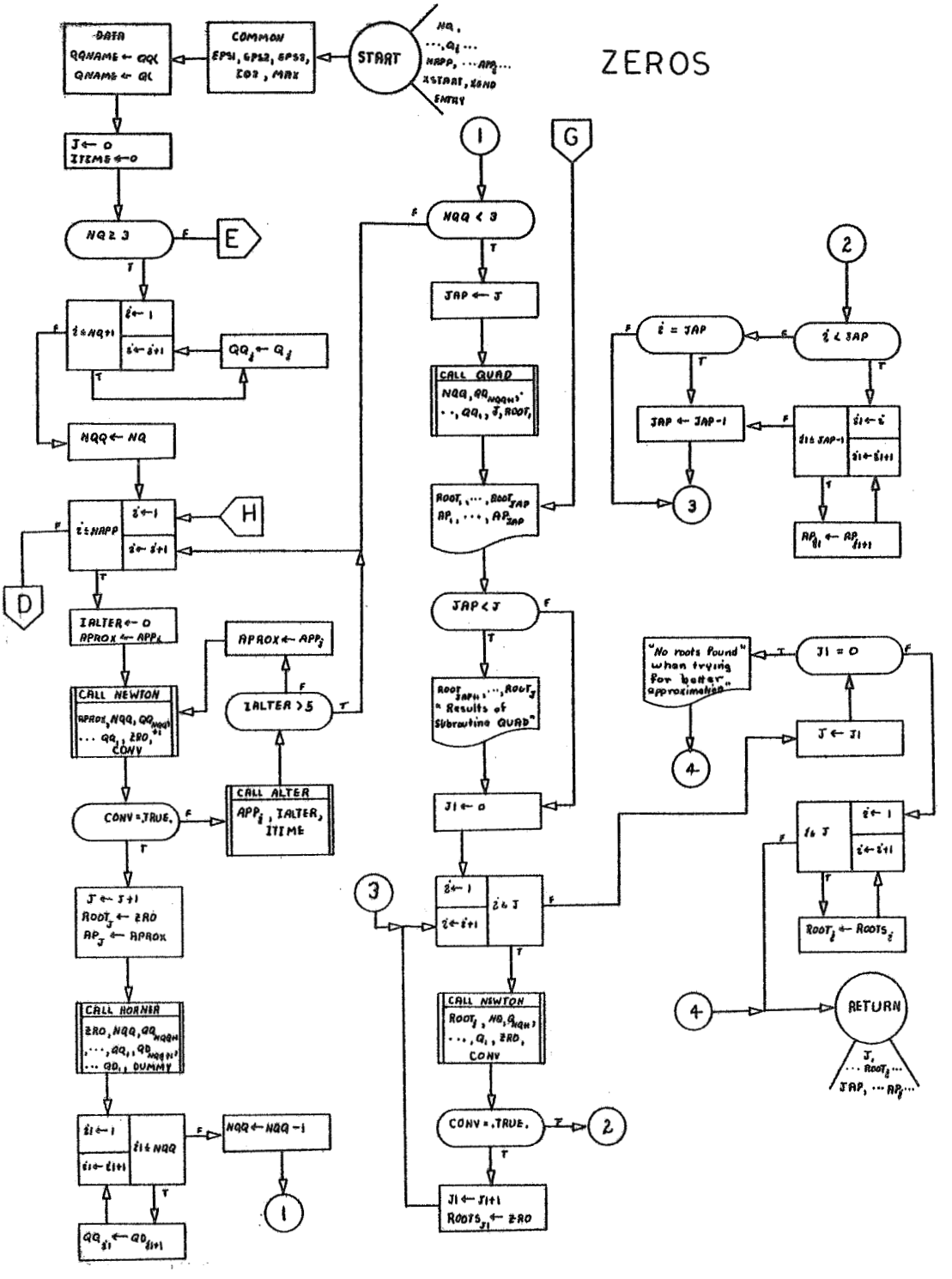


Figure G.2. (Continued)

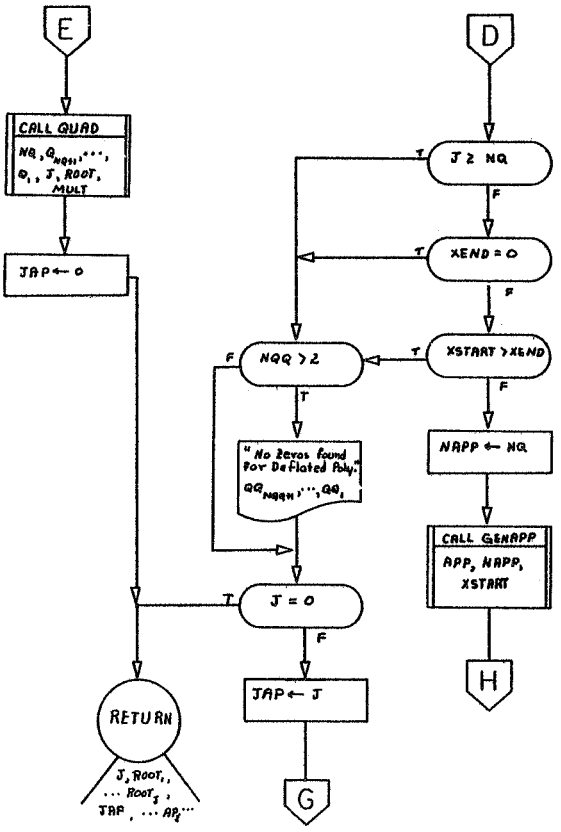
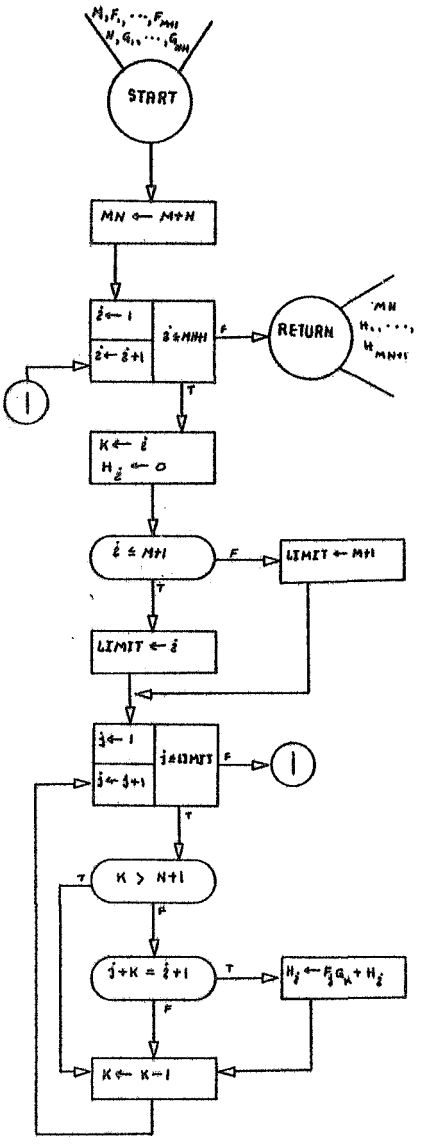


Figure G.2. (Continued)

PROD



QUAD

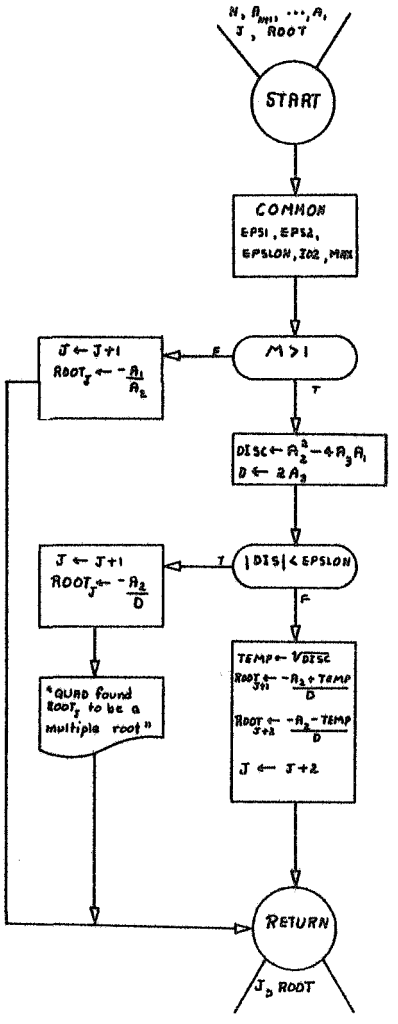


Figure G.2. (Continued)

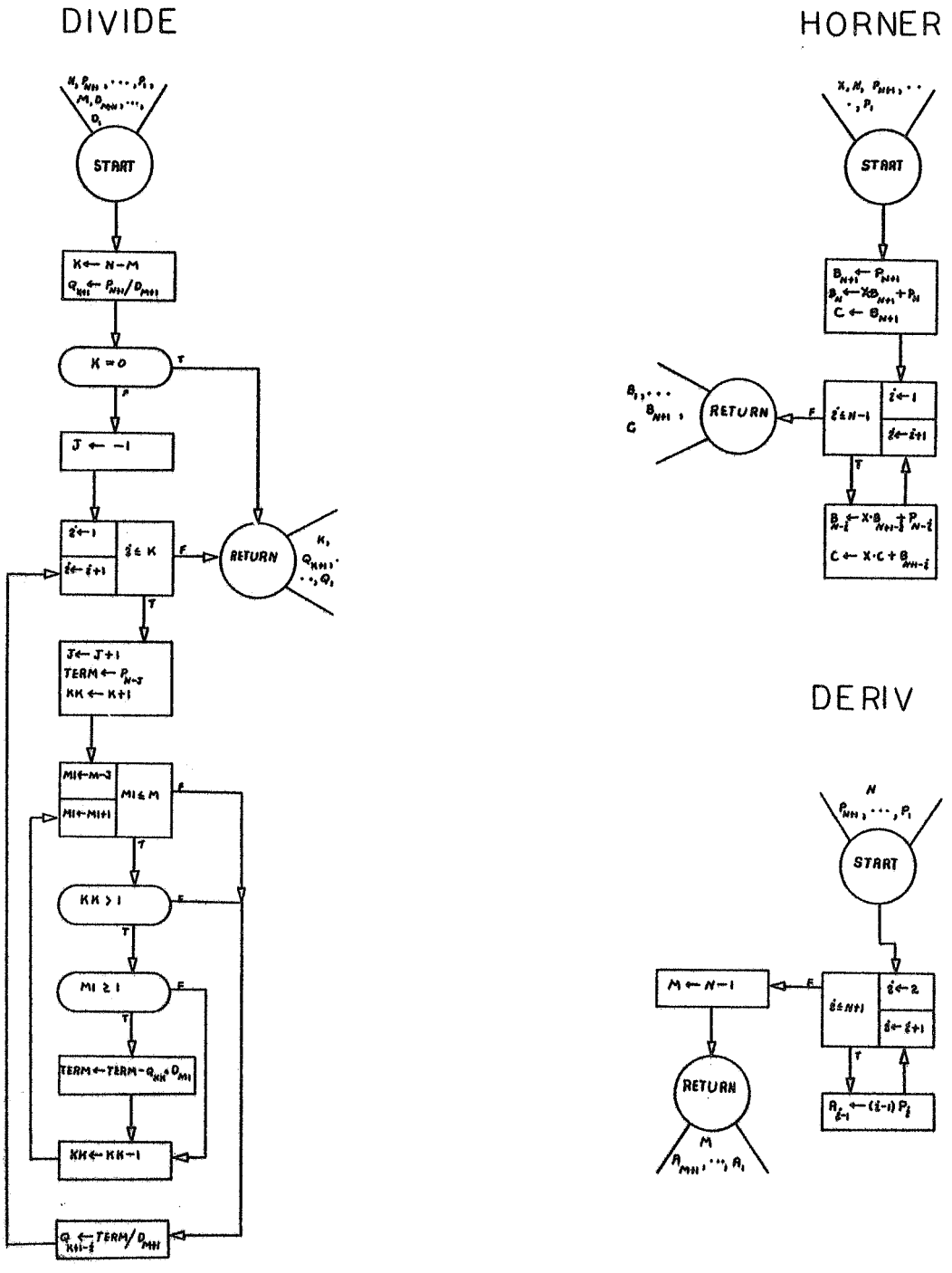
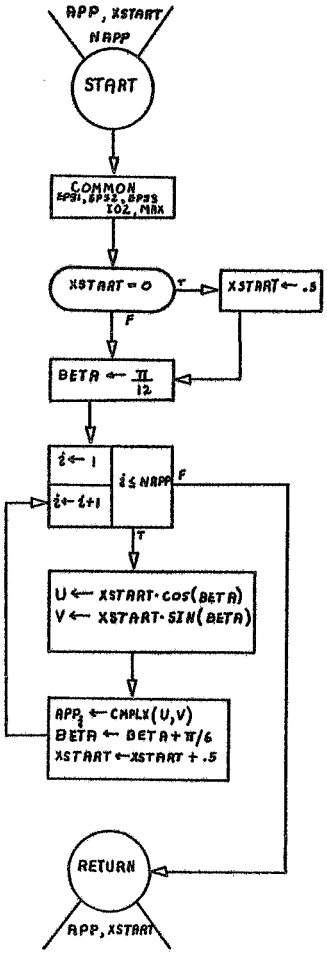


Figure G.2. (Continued)

GENAPP



ALTER

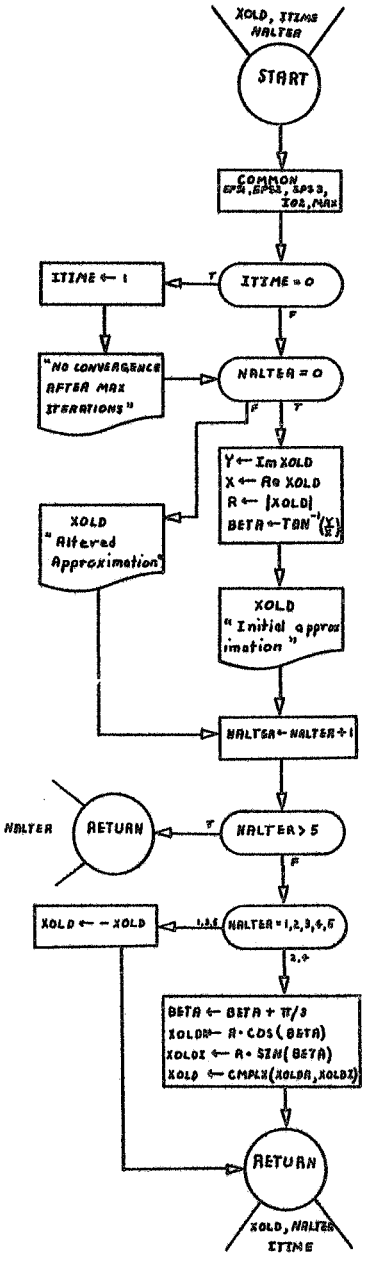


Figure G.2. (Continued)

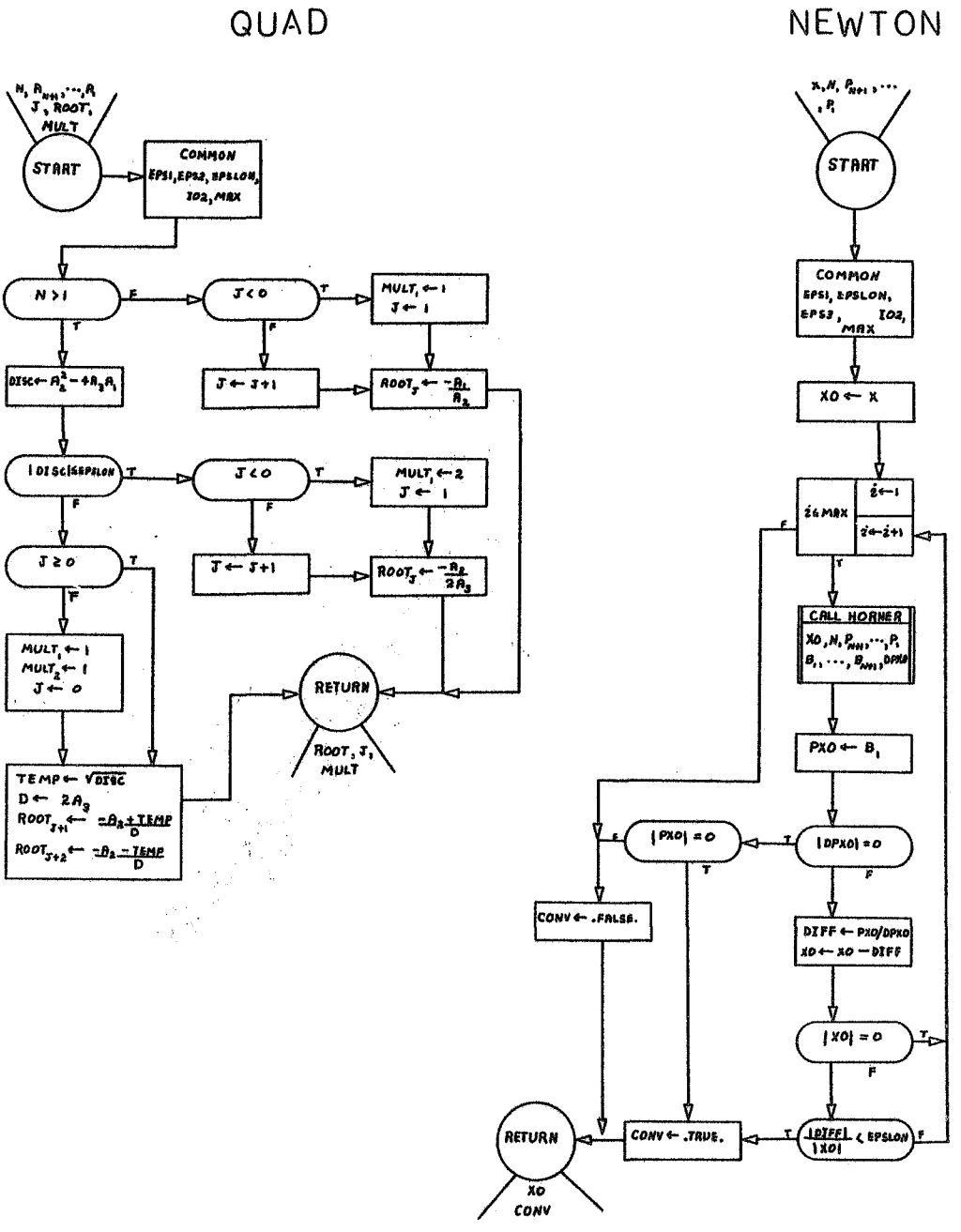


Figure G.2. (Continued)

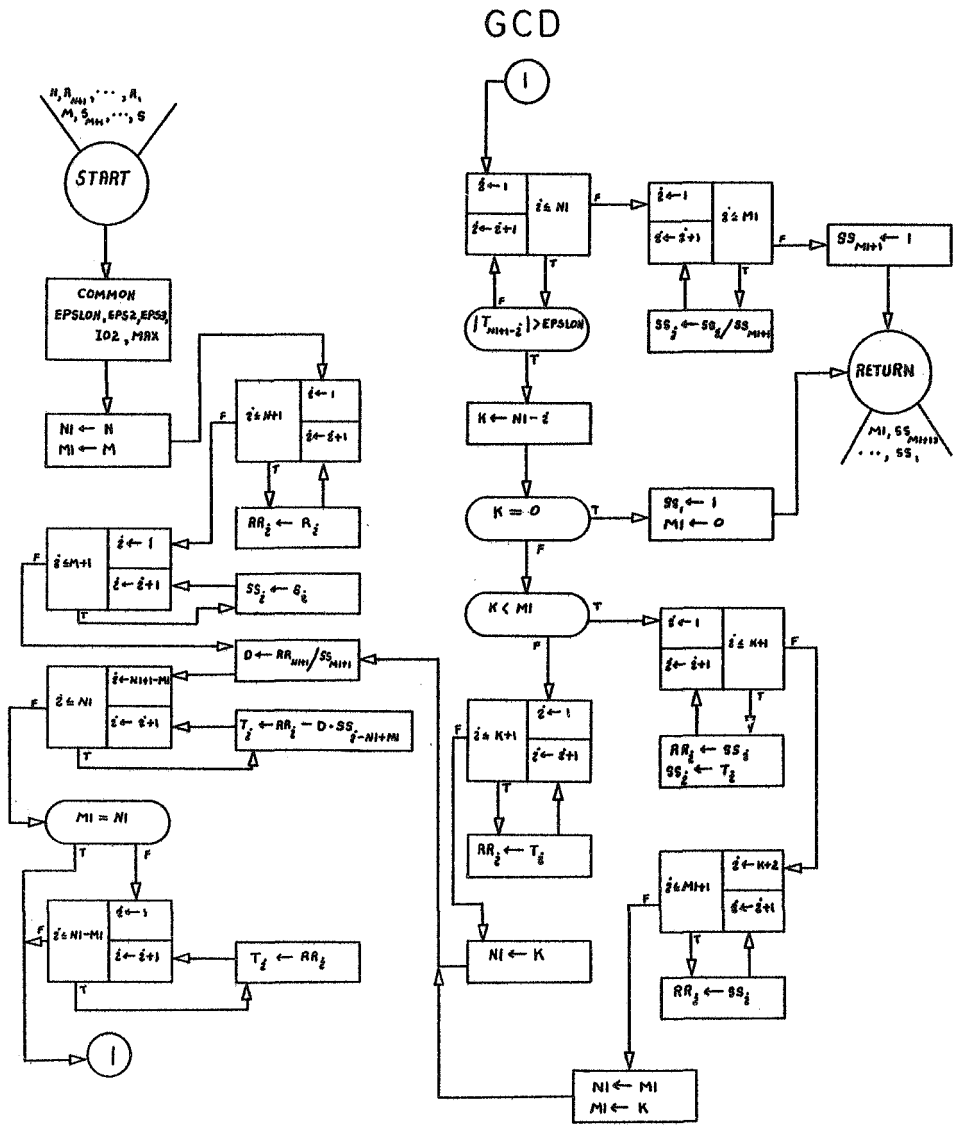


Figure G.2. (Continued)

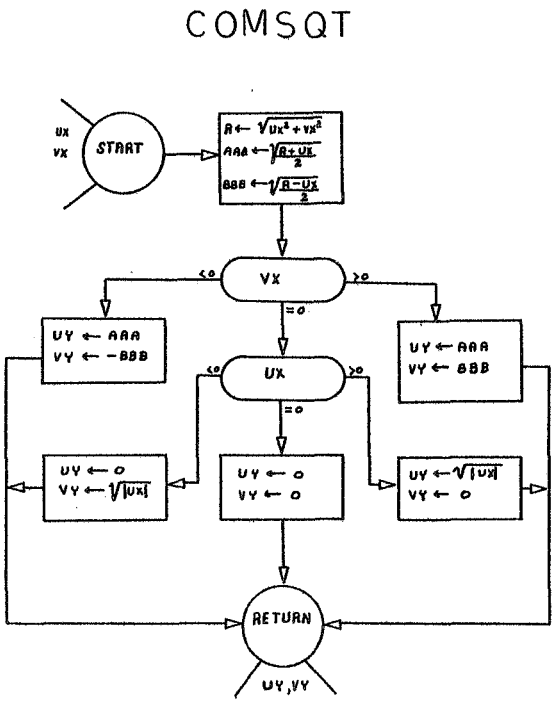


Figure G.2. (Continued)

TABLE G.III

PROGRAM FOR REPEATED G.C.D.-NEWTON'S METHOD

```
C    ************************************************************************
C    *                                                                      *
C    * DOUBLE PRECISION PROGRAM FOR THE REPEATED G.C.D. - NEWTON'S METHOD   *
C    *                                                                      *
C    *                                                                      *
C    * THIS METHOD REPEATEDLY FINDS THE GREATEST COMMON DIVISOR OF TWO       *
C    * POLYNOMIALS IN ORDER TO EXTRACT THE ZEROS IN GROUPS ACCORDING TO     *
C    * MULTIPLICITY USING NEWTON'S METHOD.  ALL ZEROS OF MULTIPLICITY 1     *
C    * ARE EXTRACTED FOLLOWED BY THOSE OF MULTIPLICITY 2, ETC.              *
C    *                                                                      *
C    ************************************************************************
0001          DOUBLE PRECISION EPS1,EPS2,EPS3,UP,VP,UAPP,VAPP,UDO,VDO,UDDO,VDDO,
             1UD1,VD1,UD2,VD2,UDD1,VDD1,UG,VG,UD3,VD3,UD4,VD4,UZROS,VZROS,UAP,VA
             2P,UROOT,VROOT,DENOM
0002          DOUBLE PRECISION XSTART
0003          DOUBLE PRECISION XEND
0004          DIMENSION ANAME(2),UP(26),VP(26),UAPP(25),VAPP(25),UDO(26),VDO(26)
             1,UDDO(26),VDDO(26),UD1(26),VD1(26),UD2(26),VD2(26),UDD1(26),VDD1(2
             26),UG(26),VG(26),UD3(51),VD3(51),UD4(51),VD4(51),UZROS(25),VZROS(2
             35),UAP(25),VAP(25),UROOT(25),VROOT(25),MULT(25),ENTRY(26)
0005          COMMON EPS1,EPS2,EPS3,IO2,MAX
0006          DATA ASTER/4H****/
0007          DATA PNAME,GNAME/2HP(,2HG(/, DINAME/3HD1(/
0008          DATA ENTRY/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,2H12,2H13
             1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
0009          DATA ANAME(1),ANAME(2)/4HNEWT,4HONS /
0010          IO1=5
0011          IO2=6
0012        1 READ(IO1,1000) NOPOLY,NP,NAPP,MAX,EPS1,EPS2,EPS3,XSTART,XEND,KCHEC
             1K
0013          IF(KCHECK.EQ.1) STOP
0014          WRITE(IO2,1020) ANAME(1),ANAME(2),NOPOLY
0015          WRITE(IO2,2000) NAPP
0016          WRITE(IO2,2010) MAX
0017          WRITE(IO2,2070) EPS1
0018          WRITE(IO2,2020) EPS2
0019          WRITE(IO2,2080) EPS3
0020          WRITE(IO2,2040) XSTART
0021          WRITE(IO2,2050) XEND
0022          WRITE(IO2,2060)
0023          KKK=NP+1
0024          NNN=KKK+1
0025          DO 5 I=1,KKK
0026          JJJ=NNN-I
0027        5 READ(IO1,1010) UP(JJJ),VP(JJJ)
0028          IF(NAPP.NE.0) GO TO 22
0029          NAPP=NP
0030          CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0031          GO TO 23
0032       22 READ(IO1,1015) (UAPP(I),VAPP(I),I=1,NAPP)
0033       23 WRITE(IO2,1030) NP
0034          KKK=NP+1
0035          NNN=KKK+1
0036          DO 8 I=1,KKK
0037          JJJ=NNN-I
0038        8 WRITE(IO2,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
0039          K=0
0040          KD=0
```

TABLE G.III (Continued)

```
0041              J1=1
0042              KKK=NP+1
0043              DO 10 I=1,KKK
0044              UDO(I)=UP(I)
0045           10 VDO(I)=VP(I)
0046              NDO=NP
0047              CALL DERIV(NDO,UDO,VDO,NDDO,UDDO,VDDO)
0048              CALL GCD(NDO,UDO,VDO,NDDO,UDDO,VDDO,ND1,UD1,VD1)
0049           20 WRITE(IO2,3000) (ASTER,I=1,33)
0050              IF(ND1.LE.1) GO TO 30
0051              GO TO 40
0052           30 UD2(1)=1.0
0053              VD2(1)=0.0
0054              ND2=0
0055              GO TO 50
0056           40 CALL DERIV(ND1,UD1,VD1,NDD1,UDD1,VDD1)
0057              CALL GCD(ND1,UD1,VD1,NDD1,UDD1,VDD1,ND2,UD2,VD2)
0058           50 IF(NDO+ND2.LE.2*ND1) GO TO 60
0059              GO TO 70
0060           60 WRITE(IO2,1025) J1
0061              GO TO 170
0062           70 IF(ND1.EQ.0) GO TO 80
0063              GO TO 90
0064           80 KKK=NDO+1
0065              DO 85 I=1,KKK
0066              UG(I)=UDO(I)
0067           85 VG(I)=VDO(I)
0068              NG=NDO
0069              GO TO 110
0070           90 IF(ND2.EQ.0) GO TO 115
0071              CALL PROD(NDO,UDO,VDO,ND2,UD2,VD2,ND3,UD3,VD3)
0072          100 CALL PROD(ND1,UD1,VD1,ND1,UD1,VD1,ND4,UD4,VD4)
0073              CALL DIVIDE(ND3,UD3,VD3,ND4,UD4,VD4,NG,UG,VG)
0074          110 WRITE(IO2,1035) J1
0075              KKK=NG+1
0076              NNN=KKK+1
0077              DO 112 I=1,KKK
0078              JJJ=NNN-I
0079          112 WRITE(IO2,1040) GNAME,ENTRY(JJJ),UG(JJJ),VG(JJJ)
0080              CALL ZEROS(NG,UG,VG,NAPP,UAPP,VAPP,J,UZROS,VZROS,JAP,UAP,VAP,ENTRY
                 1,XSTART,XEND)
0081              IF(J.EQ.0) GO TO 150
0082              WRITE(IO2,1180)
0083              IF(JAP.EQ.0) GO TO 120
0084              GO TO 130
0085          115 KKK=NDO+1
0086              DO 116 I=1,KKK
0087              UD3(I)=UDO(I)
0088          116 VD3(I)=VDO(I)
0089              ND3=NDO
0090              GO TO 100
0091          120 KKK=JAP+1
0092              WRITE(IO2,1085) (I,UZROS(I),VZROS(I),J1,I=KKK,J)
0093              GO TO 140
0094          130 WRITE(IO2,1190) (I,UZROS(I),VZROS(I),J1,UAP(I),VAP(I),I=1,JAP)
0095              IF(JAP.LT.J) GO TO 120
0096          140 IF(J.EQ.NG) GO TO 155
0097          150 WRITE(IO2,1095)
```

TABLE G.III (Continued)

```
0098            IF(J.EQ.0) GO TO 170
0099          155 DO 160 I=1,J
0100            UROOT(KD+I)=UZROS(I)
0101            VROOT(KD+I)=VZROS(I)
0102          160 MULT(KD+I)=J1
0103            K=(J*J1)+K
0104            KD=KD+J
0105            IF(K.GE.NP) GO TO 1
0106          170 J1=J1+1
0107            IF(ND1.LE.1) GO TO 200
0108            DO 180 I=1,ND1
0109            UDO(I)=UD1(I)
0110            VDO(I)=VD1(I)
0111            UDDO(I)=UDD1(I)
0112          180 VDDO(I)=VDD1(I)
0113            UDO(ND1+1)=UD1(ND1+1)
0114            VDO(ND1+1)=VD1(ND1+1)
0115            NDO=ND1
0116            NDDO=NDD1
0117            KKK=ND2+1
0118            DO 190 I=1,KKK
0119            UD1(I)=UD2(I)
0120          190 VD1(I)=VD2(I)
0121            ND1=ND2
0122            GO TO 20
0123          200 IF(ND1.EQ.0) GO TO 1
0124            KD=KD+1
0125            DENOM=UD1(2)*UD1(2)+VD1(2)*VD1(2)
0126            UROOT(KD)=(-UD1(1)*UD1(2)-VD1(1)*VD1(2))/DENOM
0127            VROOT(KD)=(-VD1(1)*UD1(2)+UD1(1)*VD1(2))/DENOM
0128            MULT(KD)=J1
0129            WRITE(IO2,3000) (ASTER,I=1,33)
0130            WRITE(IO2,1035) J1
0131            KKK=ND1+1
0132            NNN=KKK+1
0133            DO 210 I=1,KKK
0134            JJJ=NNN-I
0135          210 WRITE(IO2,1100) DINAME,ENTRY(JJJ),UD1(JJJ),VD1(JJJ)
0136            WRITE(IO2,1180)
0137            WRITE(IO2,1085) KD,UROOT(KD),VROOT(KD),J1
0138            GO TO 1
0139         1020 FORMAT(1H1,10X,48HREPEATED USE OF THE GREATEST COMMON DIVISOR AND
                1,A4,A4,58H METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIA
                2LS/14X,18HPOLYNOMIAL NUMBER ,I2///)
0140         1025 FORMAT(///1X,25HNO ROOTS OF MULTIPLICITY ,I2//)
0141         1035 FORMAT(///1X,87HTHE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE R
                1OOTS OF P(X) WHICH HAVE MULTIPLICITY ,I2//)
0142         1085 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,18X,25H
                1NO INITIAL APPROXIMATIONS)
0143         1095 FORMAT(///1X,51HNOT ALL ROOTS OF THE ABOVE POLYNOMIAL,G, WERE FOUN
                1D//)
0144         1000 FORMAT(3(I2,1X),9X,I3,1X,3(D6.0,1X),20X,2(D7.0,1X),I1)
0145         1010 FORMAT(2D30.0)
0146         1015 FORMAT(2D30.0)
0147         1030 FORMAT(1X,22HTHE DEGREE OF P(X) IS ,I2,22H  THE COEFFICIENTS ARE//
                1)
0148         1040 FORMAT(2X,A2,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
0149         1100 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
```

TABLE G.III (Continued)

```
0150          1180 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
                  1 APPROXIMATION//)
0151          1190 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,7X,I2,7X,D23.
                  116,3H + ,D23.16,2H I)
0152          2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
0153          2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
0154          2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
0155          2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
0156          2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
0157          2060 FORMAT(//1X)
0158          2070 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE GCD.  ,D9.2)
0159          2080 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE QUAD. ,D9.2)
0160          3000 FORMAT(/////1X,A3,32A4)
0161               END
```

TABLE G.III (Continued)

```
0001            SUBROUTINE PROD(M,UF,VF,N,UG,VG,MN,UH,VH)
              C ********************************************************************************
              C *                                                                              *
              C * GIVEN POLYNOMIALS R(X) AND S(X), THIS SUBROUTINE COMPUTES THE                *
              C * COEFFICIENTS OF THE PRODUCT POLYNOMIAL T(X) = R(X).S(X).                     *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UH,VH,UF,VF,UG,VG
0003            DIMENSION UH(51),VH(51),UF(26),VF(26),UG(26),VG(26)
0004            MN=M+N
0005            KKK=MN+1
0006            DO 100 I=1,KKK
0007            K=I
0008            UH(I)=0.0
0009            VH(I)=0.0
0010            IF(I.LE.M+1) GO TO 10
0011            LIMIT=M+1
0012            GO TO 20
0013           10 LIMIT=I
0014           20 DO 50 J=1,LIMIT
0015            IF(K.GT.N+1) GO TO 50
0016            IF(J+K.EQ.I+1) GO TO 40
0017            GO TO 50
0018           40 UH(I)=UH(I)+(UF(J)*UG(K)-VF(J)*VG(K))
0019            VH(I)=VH(I)+(VF(J)*UG(K)+UF(J)*VG(K))
0020           50 K=K-1
0021          100 CONTINUE
0022            RETURN
0023            END
```

```
0001            SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
              C ********************************************************************************
              C *                                                                              *
              C * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE         *
              C * DEGREE OF THE ORIGINAL POLYNOMIAL.                                           *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION APPR,APPI,XSTART,BETA,    EPS1,EPS2,EPS3
0003            DIMENSION APPR(25),APPI(25)
0004            COMMON EPS1,EPS2,EPS3,IO2,MAX
0005            IF(XSTART.EQ.0.0) XSTART=0.5
0006            BETA=0.2617994
0007            DO 10 I=1,NAPP
0008            APPR(I)=XSTART*DCOS(BETA)
0009            APPI(I)=XSTART*DSIN(BETA)
0010            BETA=BETA+0.5235988
0011           10 XSTART=XSTART+0.5
0012            RETURN
0013            END
```

TABLE G.III (Continued)

```
0001            SUBROUTINE ALTER(XOLDR,XOLDI,NALTER,ITIME)
              C   ***************************************************************************
              C   *                                                                         *
              C   * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO      *
              C   * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.  *
              C   *                                                                         *
              C   ***************************************************************************
0002            DOUBLE PRECISION XOLDR,XOLDI,ABXOLD,BETA,EPS1,EPS2,EPS3
0003            COMMON EPS1,EPS2,EPS3,IO2,MAX
0004            IF(ITIME.NE.0) GO TO 5
0005            ITIME =1
0006            WRITE(IO2,1010) MAX
0007          5 IF(NALTER.EQ.0) GO TO 10
0008            WRITE(IO2,1000) XOLDR,XOLDI
0009            GO TO 20
0010         10 ABXOLD=DSQRT((XOLDR*XOLDR)+(XOLDI*XOLDI))
0011            BETA=DATAN2(XOLDI,XOLDR)
0012            WRITE(IO2,1020) XOLDR,XOLDI
0013         20 NALTER=NALTER+1
0014            IF(NALTER.GT.5) RETURN
0015            GO TO (30,40,30,40,30),NALTER
0016         30 XOLDR=-XOLDR
0017            XOLDI=-XOLDI
0018            GO TO 50
0019         40 BETA=BETA+1.0471976
0020            XOLDR=ABXOLD*DCOS(BETA)
0021            XOLDI=ABXOLD*DSIN(BETA)
0022         50 RETURN
0023       1000 FORMAT(1X,D23.16,3H + ,D23.16,2H I,10X,21HALTERED APPROXIMATION)
0024       1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
                 1TER ,I3,12H ITERATIONS.//)
0025       1020 FORMAT(/1X,D23.16,3H + ,D23.16,2H I,10X,21HINITIAL APPROXIMATION)
0026            END
```

TABLE G.III (Continued)

```
0001              SUBROUTINE ZEROS(NQ,UQ,VQ,NAPP,UAPP,VAPP,J,UROOT,VROOT,JAP,UAP,VAP
                 1,ENTRY,XSTART,XEND)
                C   ************************************************************************
                C   *                                                                      *
                C   * NEWTONS METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A      *
                C   * POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-   *
                C   * IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION   *
                C   * FORMULA                                                              *
                C   *                X(N+1) = X(N)-P(X(N))/P'(X(N)).                       *
                C   *                                                                      *
                C   ************************************************************************
0002              DOUBLE PRECISION UAPP,VAPP,UROOT,VROOT,UZRO,VZRO,UQ,VQ,UDUMMY,VDUM
                 1MY,UQQ,VQQ,UAP,VAP,UQD,VQD,UROOTS,VROOTS,EPS1,EPS2,EPS3,UAPROX,VAP
                 2ROX
0003              DOUBLE PRECISION XEND,XSTART
0004              DIMENSION UAPP(25),VAPP(25),UROOT(25),VROOT(25),UQ(26),VQ(26),UQQ(
                 126),VQQ(26),UAP(25),VAP(25),UQD(26),VQD(26),ENTRY(26),UROOTS(25),V
                 2ROOTS(25)
0005              COMMON EPS1,EPS2,EPS3,IO2,MAX
0006              DATA QQNAME,QNAME/3HQQ(,2HQ(/
0007              LOGICAL CONV
0008              J=0
0009              ITIME=0
0010              IF(NQ.GE.3) GO TO 85
0011              GO TO 110
0012             85 KKK=NQ+1
0013              DO 90 I=1,KKK
0014              UQQ(I)=UQ(I)
0015             90 VQQ(I)=VQ(I)
0016              NQQ=NQ
0017              GO TO 120
0018            110 CALL QUAD(NQ,UQ,VQ,J,UROOT,VROOT)
0019              JAP=0
0020              GO TO 310
0021            120 DO 200 I=1,NAPP
0022              IALTER=0
0023              UAPROX=UAPP(I)
0024              VAPROX=VAPP(I)
0025            130 CALL NEWTON(UAPROX,VAPROX,NQQ,UQQ,VQQ,UZRO,VZRO,CONV)
0026              IF(CONV) GO TO 160
0027              CALL ALTER(UAPP(I),VAPP(I),IALTER,ITIME)
0028              IF(IALTER.GT.5) GO TO 200
0029              UAPROX=UAPP(I)
0030              VAPROX=VAPP(I)
0031              GO TO 130
0032            160 J=J+1
0033              UROOT(J)=UZRO
0034              VROOT(J)=VZRO
0035              UAP(J)=UAPROX
0036              VAP(J)=VAPROX
0037              CALL HORNER(UZRO,VZRO,NQQ,UQQ,VQQ,UQD,VQD,UDUMMY,VDUMMY)
0038              DO 180 I1=1,NQQ
0039              UQQ(I1)=UQD(I1+1)
0040            180 VQQ(I1)=VQD(I1+1)
0041              NQQ=NQQ-1
0042              IF(NQQ.GE.3) GO TO 200
0043              JAP=J
0044              GO TO 220
```

TABLE G.III (Continued)

```
0045          200 CONTINUE
0046              IF(J.GE.NQ) GO TO 205
0047              IF(XEND.EQ.0.0) GO TO 205
0048              IF(XSTART.GT.XEND) GO TO 205
0049              NAPP=NQ
0050              CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0051              GO TO 120
0052          205 IF(NQQ.LE.2) GO TO 210
0053              WRITE(IO2,1200)
0054              KKK=NQQ+1
0055              NNN=KKK+1
0056              DO 157 L=1,KKK
0057              JJJ=NNN-L
0058          157 WRITE(IO2,1100) QQNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0059          210 IF(J.EQ.0) GO TO 310
0060              JAP=J
0061              GO TO 230
0062          220 CALL QUAD(NQQ,UQQ,VQQ,J,UROOT,VROOT)
0063          230 WRITE(IO2,1132)
0064              WRITE(IO2,1133) (I,UROOT(I),VROOT(I),UAP(I),VAP(I),I=1,JAP)
0065              IF(JAP.LT.J) GO TO 235
0066              GO TO 240
0067          235 KKK=JAP+1
0068              WRITE(IO2,1134) (I,UROOT(I),VROOT(I),I=KKK,J)
0069          240 J1=0
0070              DO 300 I=1,J
0071              CALL NEWTON(UROOT(I),VROOT(I),NQ,UQ,VQ,UZRO,VZRO,CONV)
0072              IF(CONV) GO TO 280
0073              WRITE(IO2,1140) I,UROOT(I),VROOT(I),MAX,NQ
0074              KKK=NQ+1
0075              NNN=KKK+1
0076              DO 242 L=1,KKK
0077              JJJ=NNN-L
0078          242 WRITE(IO2,1040) QNAME,ENTRY(JJJ),UQ(JJJ),VQ(JJJ)
0079              IF(I.LT.JAP) GO TO 241
0080              IF(I.EQ.JAP) GO TO 250
0081              GO TO 300
0082          241 KKK=JAP-1
0083              DO 245 I1=I,KKK
0084              UAP(I1)=UAP(I1+1)
0085          245 VAP(I1)=VAP(I1+1)
0086          250 JAP=JAP-1
0087              GO TO 300
0088          280 J1=J1+1
0089              UROOTS(J1)=UZRO
0090              VROOTS(J1)=VZRO
0091          300 CONTINUE
0092              J=J1
0093              IF(J.EQ.0) GO TO 305
0094              DO 303 I=1,J
0095              UROOT(I)=UROOTS(I)
0096          303 VROOT(I)=VROOTS(I)
0097              GO TO 310
0098          305 WRITE(IO2,1150) NQ
0099              KKK=NQ+1
0100              NNN=KKK+1
0101              DO 306 L=1,KKK
0102              JJJ=NNN-L
```

TABLE G.III (Continued)

```
0103          306 WRITE(IO2,1040) QNAME,ENTRY(JJJ),UQ(JJJ),VQ(JJJ)
0104          310 RETURN
0105         1200 FORMAT(///1X,70HCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH
                1NO ZEROS WERE FOUND.//)
0106         1132 FORMAT(///1X,13HROOTS OF G(X),84X,21HINITIAL APPROXIMATION//)
0107         1133 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,17X,D23.16,3H
                1 + ,D23.16,2H I)
0108         1134 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,22X,26HRESULT
                1S OF SUBROUTINE QUAD)
0109         1140 FORMAT(///,1X,40HNO ROOTS FOR INITIAL APPROXIMATION ROOT(,I2,4H) =
                1 ,D23.16,3H + ,D23.16,2H I/6H  AND ,I3,40H ITERATIONS ON THE POLYN
                2OMIAL OF DEGREE ,I2,18H WITH COEFFICIENTS//)
0110         1150 FORMAT(///,1X,45HNO ROOTS FOR THE POLYNOMIAL Q(X) OF DEGREE = ,I2,
                138H WITH GENERATED INITIAL APPROXIMATIONS//)
0111         1040 FORMAT(2X,A2,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
0112         1100 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
0113              END
```

TABLE G.III (Continued)

```
0001              SUBROUTINE GCD(N,UR,VR,M,US,VS,M1,USS,VSS)
                C ***************************************************************************
                C *                                                                         *
                C * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.     *
                C * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND   *
                C * DP(X).                                                                 *
                C *                                                                         *
                C ***************************************************************************
0002              DOUBLE PRECISION USSSSS,VSSSSS
0003              DOUBLE PRECISION UR,VR,US,VS,USS,VSS,URR,VRR,UD,VD,UT,VT,EPSLON,EP
                 1S2,EPS3,BBB
0004              DIMENSION UR(26),VR(26),US(26),VS(26),USS(26),VSS(26),URR(26),VRR(
                 126),UT(26),VT(26)
0005              COMMON EPSLON,EPS2,EPS3,IO2,MAX
0006              N1=N
0007              M1=M
0008              KKK=N+1
0009              DO 20 I=1,KKK
0010              URR(I)=UR(I)
0011           20 VRR(I)=VR(I)
0012              KKK=M+1
0013              DO 25 I=1,KKK
0014              USS(I)=US(I)
0015           25 VSS(I)=VS(I)
0016           30 BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0017              UD=(URR(N1+1)*USS(M1+1)+VRR(N1+1)*VSS(M1+1))/BBB
0018              VD=(USS(M1+1)*VRR(N1+1)-URR(N1+1)*VSS(M1+1))/BBB
0019              KKK=N1+1-M1
0020              DO 40 I=KKK,N1
0021              UT(I)=URR(I)-(UD*USS(I-N1+M1)-VD*VSS(I-N1+M1))
0022           40 VT(I)=VRR(I)-(UD*VSS(I-N1+M1)+VD*USS(I-N1+M1))
0023              IF(M1.EQ.N1) GO TO 70
0024              KKK=N1-M1
0025              DO 60 I=1,KKK
0026              UT(I)=URR(I)
0027           60 VT(I)=VRR(I)
0028           70 DO 90 I=1,N1
0029              BBB=DSQRT(UT(N1+1-I)*UT(N1+1-I)+VT(N1+1-I)*VT(N1+1-I))
0030              IF(BBB.GT.EPSLON) GO TO 100
0031           90 CONTINUE
0032              DO 95 I=1,M1
0033              BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0034              USSSSS=(USS(I)*USS(M1+1)+VSS(I)*VSS(M1+1))/BBB
0035              VSSSSS=(VSS(I)*USS(M1+1)-USS(I)*VSS(M1+1))/BBB
0036              USS(I)=USSSSS
0037           95 VSS(I)=VSSSSS
0038              USS(M1+1)=1.0
0039              VSS(M1+1)=0.0
0040              GO TO 200
0041          100 K=N1-I
0042              IF(K.EQ.0) GO TO 170
0043              IF(K.LT.M1) GO TO 140
0044              KKK=K+1
0045              DO 130 J=1,KKK
0046              URR(J)=UT(J)
0047          130 VRR(J)=VT(J)
0048              N1=K
0049              GO TO 30
```

TABLE G.III (Continued)

```
0050          140 KKK=K+1
0051              DO 150 J=1,KKK
0052              URR(J)=USS(J)
0053              VRR(J)=VSS(J)
0054              USS(J)=UT(J)
0055          150 VSS(J)=VT(J)
0056              KKK=K+2
0057              NNN=M1+1
0058              DO 160 J=KKK,NNN
0059              URR(J)=USS(J)
0060          160 VRR(J)=VSS(J)
0061              N1=M1
0062              M1=K
0063              GO TO 30
0064          170 USS(1)=1.0
0065              VSS(1)=0.0
0066              M1=0
0067          200 RETURN
0068              END
```

TABLE G.III (Continued)

```
0001            SUBROUTINE NEWTON(UX,VX,N,UP,VP,UXO,VXO,CONV)
              C ********************************************************************************
              C *                                                                              *
              C * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-          *
              C * IMATION BY USING THE ITERATION FORMULA                                       *
              C *                X(N+1) = X(N)-P(X(N))/P'(X(N)).                               *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UX,VX,UP,VP,UXO,VXO,UB,VB,UDPXO,VDPXO,UPXO,VPXO,U
               1DIFF,VDIFF,EPS1,EPSLON,EPS3,AAA,BBB
0003            DOUBLE PRECISION DDD
0004            DOUBLE PRECISION ABPXO
0005            DIMENSION UP(26),VP(26),UB(26),VB(26)
0006            COMMON EPS1,EPSLON,EPS3,IO2,MAX
0007            LOGICAL CONV
0008            UXO=UX
0009            VXO=VX
0010            DO 10 I=1,MAX
0011            CALL HORNER(UXO,VXO,N,UP,VP,UB,VB,UDPXO,VDPXO)
0012            UPXO=UB(1)
0013            VPXO=VB(1)
0014            DDD=DSQRT(UDPXO*UDPXO+VDPXO*VDPXO)
0015            IF(DDD.NE.0.0) GO TO 5
0016            ABPXO=DSQRT(UPXO*UPXO+VPXO*VPXO)
0017            IF(ABPXO.EQ.0.0) GO TO 20
0018            GO TO 15
0019          5 BBB=UDPXO*UDPXO+VDPXO*VDPXO
0020            UDIFF=(UPXO*UDPXO+VPXO*VDPXO)/BBB
0021            VDIFF=(VPXO*UDPXO-UPXO*VDPXO)/BBB
0022            UXO=UXO-UDIFF
0023            VXO=VXO-VDIFF
0024            AAA=DSQRT(UDIFF*UDIFF+VDIFF*VDIFF)
0025            BBB=DSQRT(UXO*UXO+VXO*VXO)
0026            IF(BBB.EQ.0.0) GO TO 10
0027            IF(AAA/BBB.LT.EPSLON) GO TO 20
0028           10 CONTINUE
0029           15 CONV=.FALSE.
0030            RETURN
0031           20 CONV=.TRUE.
0032            RETURN
0033            END
```

TABLE G.III (Continued)

```
0001            SUBROUTINE DIVIDE(N,UP,VP,M,UD,VD,K,UQ,VQ)
              C ***************************************************************************
              C *                                                                         *
              C * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE     *
              C * QUOTIENT POLYNOMIAL  H(X) = F(X)/G(X).                                  *
              C *                                                                         *
              C ***************************************************************************
0002            DOUBLE PRECISION UP,VP,UD,VD,UQ,VQ,UTERM,VTERM,UDUMMY
0003            DIMENSION UP(26),VP(26),UD(26),VD(26),UQ(26),VQ(26)
0004            K=N-M
0005            UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0006            UQ(K+1)=(UP(N+1)*UD(M+1)+VP(N+1)*VD(M+1))/UDUMMY
0007            VQ(K+1)=(VP(N+1)*UD(M+1)-UP(N+1)*VD(M+1))/UDUMMY
0008            IF(K.EQ.0) GO TO 100
0009            J=-1
0010            DO 50 I=1,K
0011            J=J+1
0012            UTERM=UP(N-J)
0013            VTERM=VP(N-J)
0014            KK=K+1
0015            NNN=M-J
0016            DO 40 M1=NNN,M
0017            IF(KK.GT.1) GO TO 10
0018            GO TO 45
0019           10 IF(M1.GE.1) GO TO 20
0020            GO TO 40
0021           20 UTERM=UTERM-(UQ(KK)*UD(M1)-VQ(KK)*VD(M1))
0022            VTERM=VTERM-(UQ(KK)*VD(M1)+VQ(KK)*UD(M1))
0023           40 KK=KK-1
0024           45 UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0025            UQ(K+1-I)=(UTERM*UD(M+1)+VTERM*VD(M+1))/UDUMMY
0026           50 VQ(K+1-I)=(VTERM*UD(M+1)-UTERM*VD(M+1))/UDUMMY
0027          100 RETURN
0028            END
```

TABLE G.III (Continued)

```
0001            SUBROUTINE HORNER(UX,VX,N,UP,VP,UB,VB,UC,VC)
              C ********************************************************************************
              C *                                                                              *
              C * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A               *
              C * POINT D AND ITS DERIVATIVE AT D.  SYNTHETIC DIVISION IS USED TO              *
              C * DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X - D).                  *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UX,VX,UP,VP,UB,VB,UC,VC
0003            DOUBLE PRECISION UDUMMY,VDUMMY
0004            DIMENSION UP(26),VP(26),UB(26),VB(26)
0005            UB(N+1)=UP(N+1)
0006            VB(N+1)=VP(N+1)
0007            UB(N)=(UX*UB(N+1)-VX*VB(N+1))+UP(N)
0008            VB(N)=(UX*VB(N+1)+VX*UB(N+1))+VP(N)
0009            UC=UB(N+1)
0010            VC=VB(N+1)
0011            KKK=N-1
0012            DO 10 I=1,KKK
0013            UB(KKK+1-I)=(UX*UB(KKK+2-I)-VX*VB(KKK+2-I))+UP(KKK+1-I)
0014            VB(KKK+1-I)=(UX*VB(KKK+2-I)+VX*UB(KKK+2-I))+VP(KKK+1-I)
0015            UDUMMY=UX*UC-VX*VC
0016            VDUMMY=UX*VC+VX*UC
0017            UC=UDUMMY+UB(KKK+2-I)
0018         10 VC=VDUMMY+VB(KKK+2-I)
0019            RETURN
0020            END
```

TABLE G.III (Continued)

```
0001              SUBROUTINE QUAD(N,UA,VA,J,UROOT,VROOT)
                  C   ***************************************************************************
                  C   *                                                                         *
                  C   * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES  *
                  C   * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE   *
                  C   * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                          *
                  C   *                                                                         *
                  C   ***************************************************************************
0002                  DOUBLE PRECISION EPS1,EPS2,EPSLON,UROOT,VROOT,UA,VA,UDISC,VDISC,UD
                     1,VD,DDD,UTEMP,VTEMP,BBB
0003                  DIMENSION UROOT(25),VROOT(25),UA(26),VA(26)
0004                  COMMON EPS1,EPS2,EPSLON,IO2,MAX
0005                  IF(N.GT.1) GO TO 10
0006                  J=J+1
0007                  BBB=UA(2)*UA(2)+VA(2)*VA(2)
0008                  UROOT(J)=-(UA(1)*UA(2)+VA(1)*VA(2))/BBB
0009                  VROOT(J)=-(VA(1)*UA(2)-UA(1)*VA(2))/BBB
0010                  GO TO 100
0011               10 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(3)*UA(1)-VA(3)*VA(1)))
0012                  VDISC=(2.0*UA(2)*VA(2))-(4.0*(UA(3)*VA(1)+VA(3)*UA(1)))
0013                  UD=2.0*UA(3)
0014                  VD=2.0*VA(3)
0015                  DDD=DSQRT(UDISC*UDISC+VDISC*VDISC)
0016                  IF(DDD.LT.EPSLON) GO TO 20
0017                  CALL COMSQT(UDISC,VDISC,UTEMP,VTEMP)
0018                  BBB=UD*UD+VD*VD
0019                  UROOT(J+1)=((-UA(2)+UTEMP)*UD+(-VA(2)+VTEMP)*VD)/BBB
0020                  VROOT(J+1)=((-VA(2)+VTEMP)*UD-(-UA(2)+UTEMP)*VD)/BBB
0021                  UROOT(J+2)=((-UA(2)-UTEMP)*UD+(-VA(2)-VTEMP)*VD)/BBB
0022                  VROOT(J+2)=((-VA(2)-VTEMP)*UD-(-UA(2)-UTEMP)*VD)/BBB
0023                  J=J+2
0024                  GO TO 100
0025               20 J=J+1
0026                  BBB=UD*UD+VD*VD
0027                  UROOT(J)=(-UA(2)*UD-VA(2)*VD)/BBB
0028                  VROOT(J)=(-VA(2)*UD+UA(2)*VD)/BBB
0029                  WRITE(IO2,1000) UROOT(J),VROOT(J)
0030             1000 FORMAT(///1X,11HQUAD FOUND ,D23.16,3H + ,D23.16,2H I,22H TO BE A M
                     1ULTIPLE ROOT//)
0031              100 RETURN
0032                  END
```

TABLE G.III (Continued)

```
0001            SUBROUTINE DERIV(N,UP,VP,M,UA,VA)
              C ********************************************************************************
              C *                                                                              *
              C * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF       *
              C * ITS DERIVATIVE P'(X).                                                        *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UP,VP,UA,VA,AAA
0003            DIMENSION UP(26),VP(26),UA(26),VA(26)
0004            KKK=N+1
0005            DO 10 I=2,KKK
0006            AAA=I-1
0007            UA(I-1)=AAA*UP(I)
0008           10 VA(I-1)=AAA*VP(I)
0009            M=N-1
0010            RETURN
0011            END
```

```
0001            SUBROUTINE COMSQT(UX,VX,UY,VY)
              C ********************************************************************************
              C *                                                                              *
              C * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.                *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
0003            R=DSQRT(UX*UX+VX*VX)
0004            AAA=DSQRT(DABS((R+UX)/2.0))
0005            BBB=DSQRT(DABS((R-UX)/2.0))
0006            IF(VX) 10,20,30
0007           10 UY=AAA
0008            VY=-1.0*BBB
0009            GO TO 100
0010           20 IF(UX) 40,50,60
0011           30 UY=AAA
0012            VY=BBB
0013            GO TO 100
0014           40 DUMMY=DABS(UX)
0015            UY=0.0
0016            VY=DSQRT(DUMMY)
0017            GO TO 100
0018           50 UY=0.0
0019            VY=0.0
0020            GO TO 100
0021           60 DUMMY=DABS(UX)
0022            UY=DSQRT(DUMMY)
0023            VY=0.0
0024          100 RETURN
0025            END
```

APPENDIX H

REPEATED G.C.D. - MULLER'S METHOD

1. Use of the Program

A double precision FORTRAN IV program using the repeated G.C.D. method with Muller's method as a supporting method is presented here. Flow charts for this program are given in Figure H.1 while Table H.III gives a FORTRAN IV listing of this program.

This program is designed to solve polynomials having degree less than or equal to 25. In order to solve polynomials of degree N where N > 25, the data statement and array dimensions given in Table H.I must be changed.

In this program both the leading coefficient and the constant coefficient are assumed to be non-zero.

TABLE H.I

PROGRAM CHANGES NECESSARY TO SOLVE POLYNOMIALS OF DEGREE GREATER THAN 25 BY THE REPEATED G.C.D. - MULLER'S METHOD

Main Program

Data Entry/1H1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1
UAPP(N,3), VAPP(N,3)
URAPP(N,3), URAPP(N,3)
UP(N+1), VP(N+1)
MULT(N)
UDD0(N+1), VDD0(N+1)
UD1(N+1), VD1(N+1)
UDD1(N+1), VDD1(N+1)
UD2(N+1), VD2(N+1)
UG(N+1), VG(N+1)
UD3(2N+1), VD3(2N+1)
UD4(2N+1), VD4(2N+1)
UAP(N+1), VAP(N+1)
UZROS(N), VZROS(N)
UROOT(N), VROOT(N)
UD0(N+1), VD0(N+1)
ENTRY(N+1)

Subroutines PROD, QUAD

See corresponding subroutine in Table G.I.

Subroutines DERIV, GCD, and DIVIDE

See corresponding subroutine in Table E.I.

Subroutines MULLER, GENAPP, BETTER and HORNER

See corresponding subroutine in Table F.I.

2. Input Data for Repeated G.C.D. - Muller's Method

The input data to the repeated G.C.D. - Muller's method is the same as for the repeated G.C.D. - Newton's method as described in Appendix G, § 2.

3. Variables Used in Repeated G.C.D. - Muller's Method

The variables used in this program are referenced in Table H.II. The notation and symbols used in the referenced tables are described in Appendix E, § 3.

TABLE H.II

VARIABLES USED IN REPEATED G.C.D. - MULLER'S METHOD

Main Program and Subroutine PROD

See Table G.II.

Subroutines QUAD, DERIV, GCD, DIVIDE, and COMSQT

See corresponding subroutine in Table E.VI.

Subroutines CALC, MULLER, GENAPP, ALTER, BETTER, TEST, and HORNER

See corresponding subroutine in Table F.II.

4. Description of Program Output

The output for this program is the same as that for repeated G.C.D. - Newton's method as described in Appendix G, § 4. Only one initial approximation, X_0, (not three) is printed. The other two required by Muller's method are $.9X_0$ and $1.1X_0$. The message "SOLVED BY DIRECT METHOD" means that the corresponding root was obtained by Subroutine QUAD.

5. Informative Messages and Error Messages

Descriptions of the informative messages and error messages printed by this program can be found either in Appendix E, § 5, Appendix F, § 5, or Appendix G, § 5.

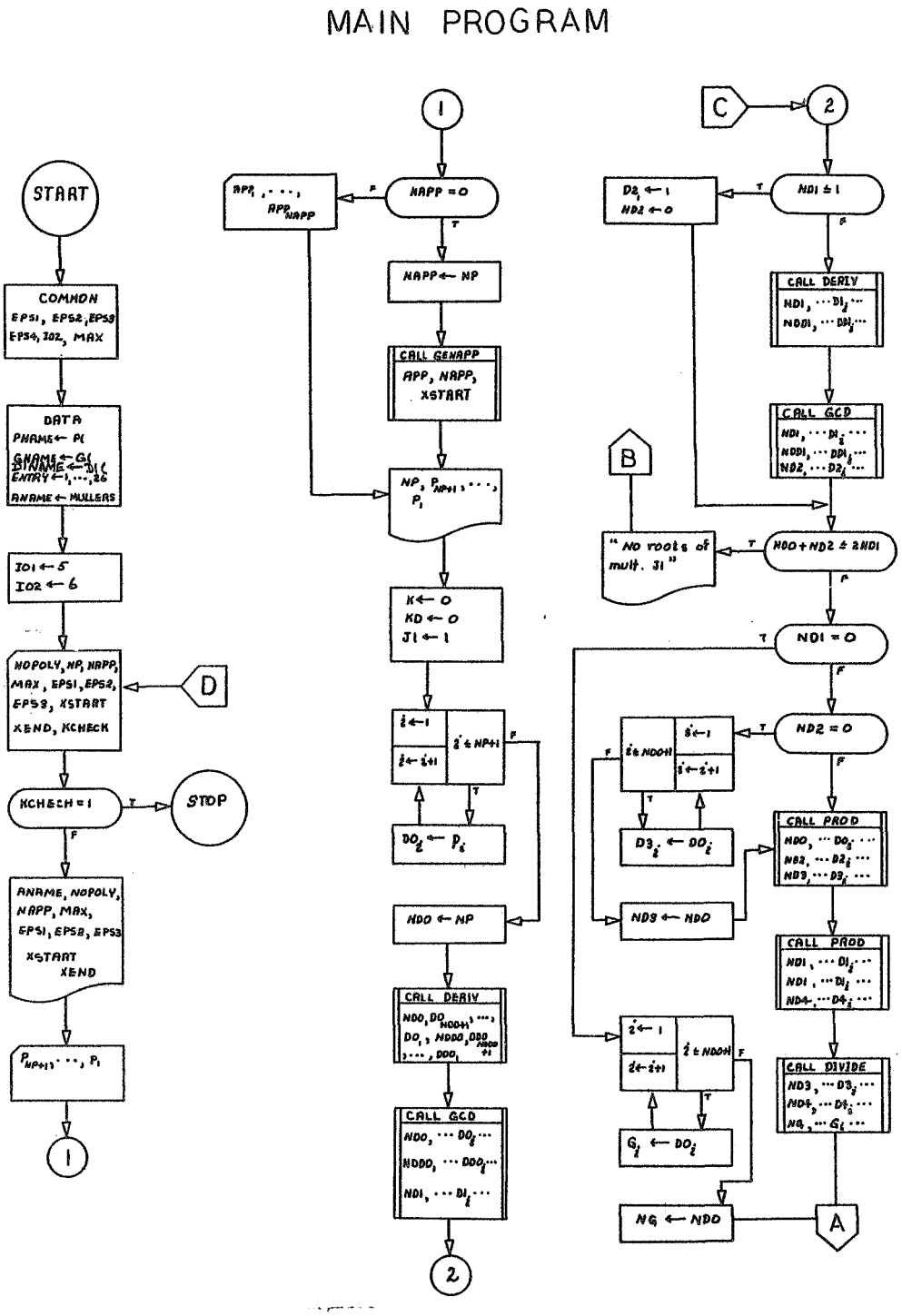


Figure H.1. Flow Charts for Repeated G.C.D.-Muller's Method

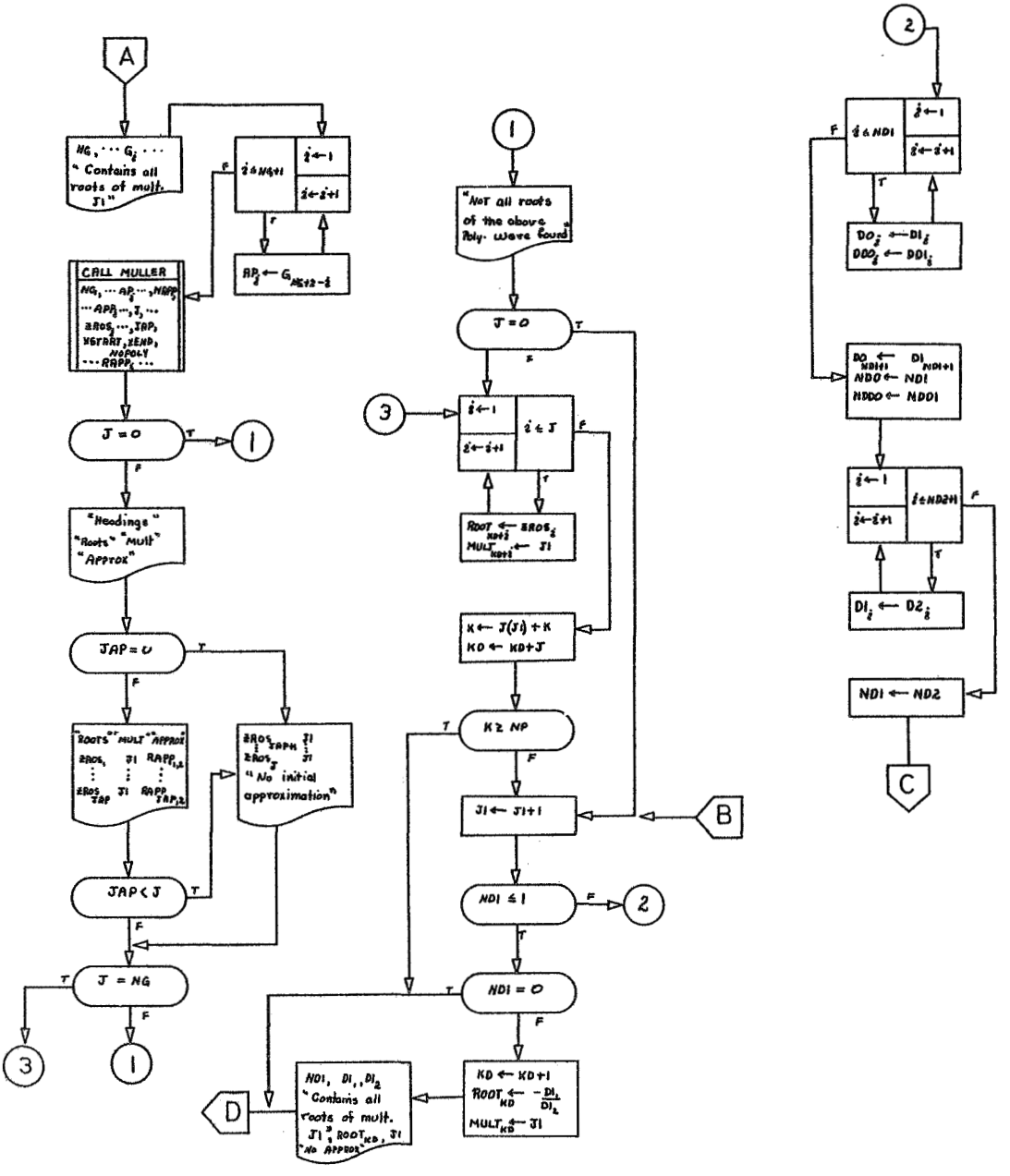


Figure H.1. (Continued)

MULLER

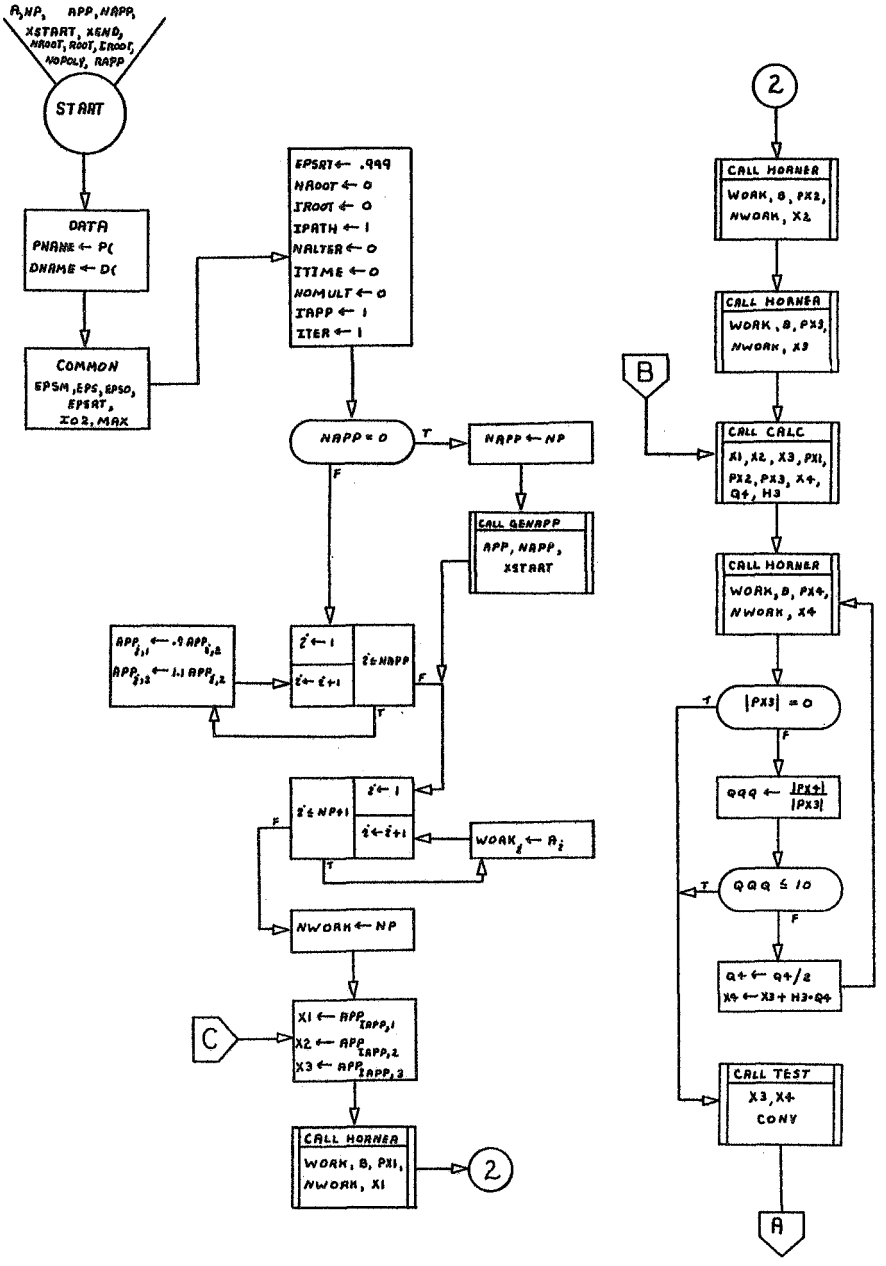


Figure H.1. (Continued)

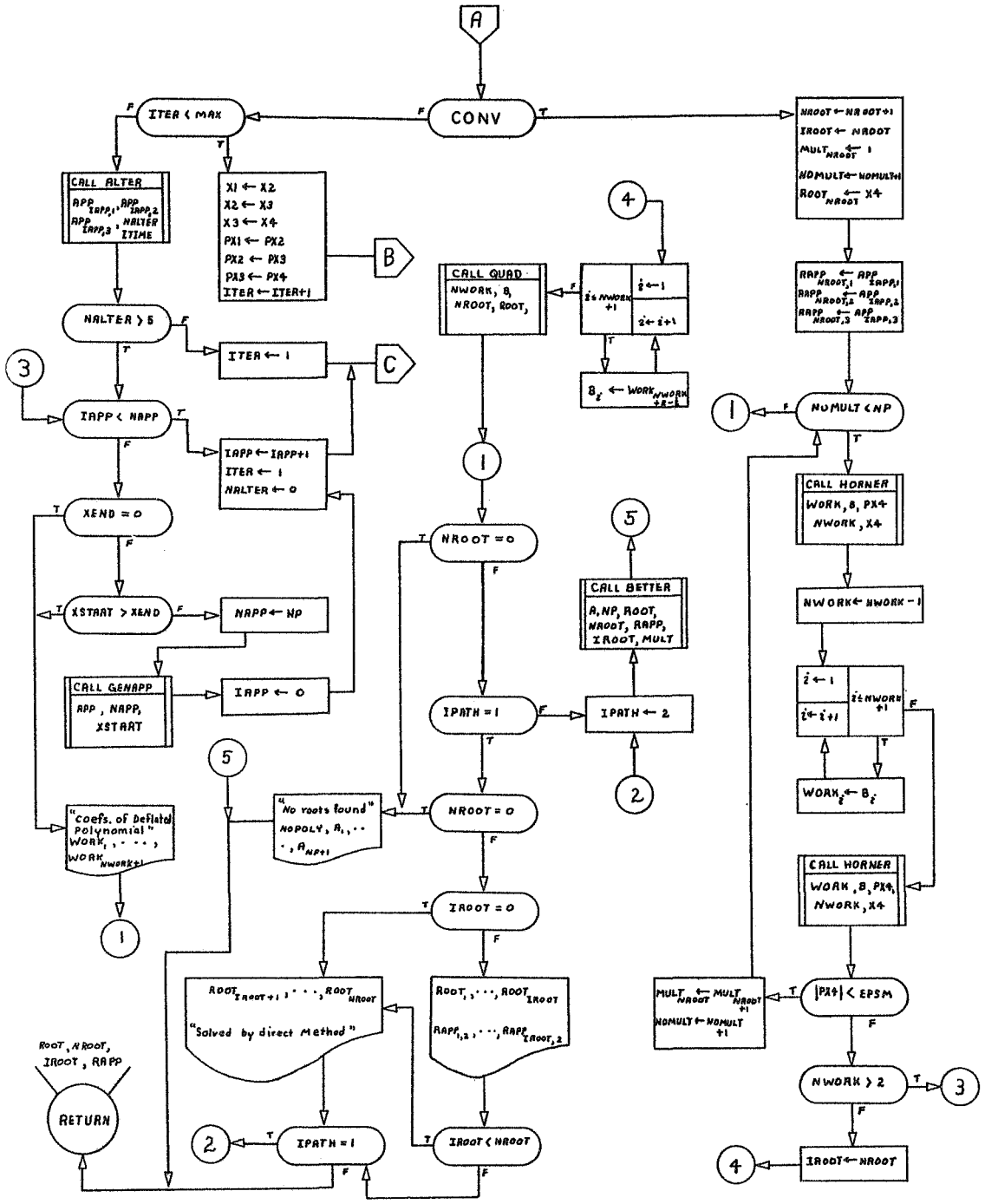


Figure H.1. (Continued)

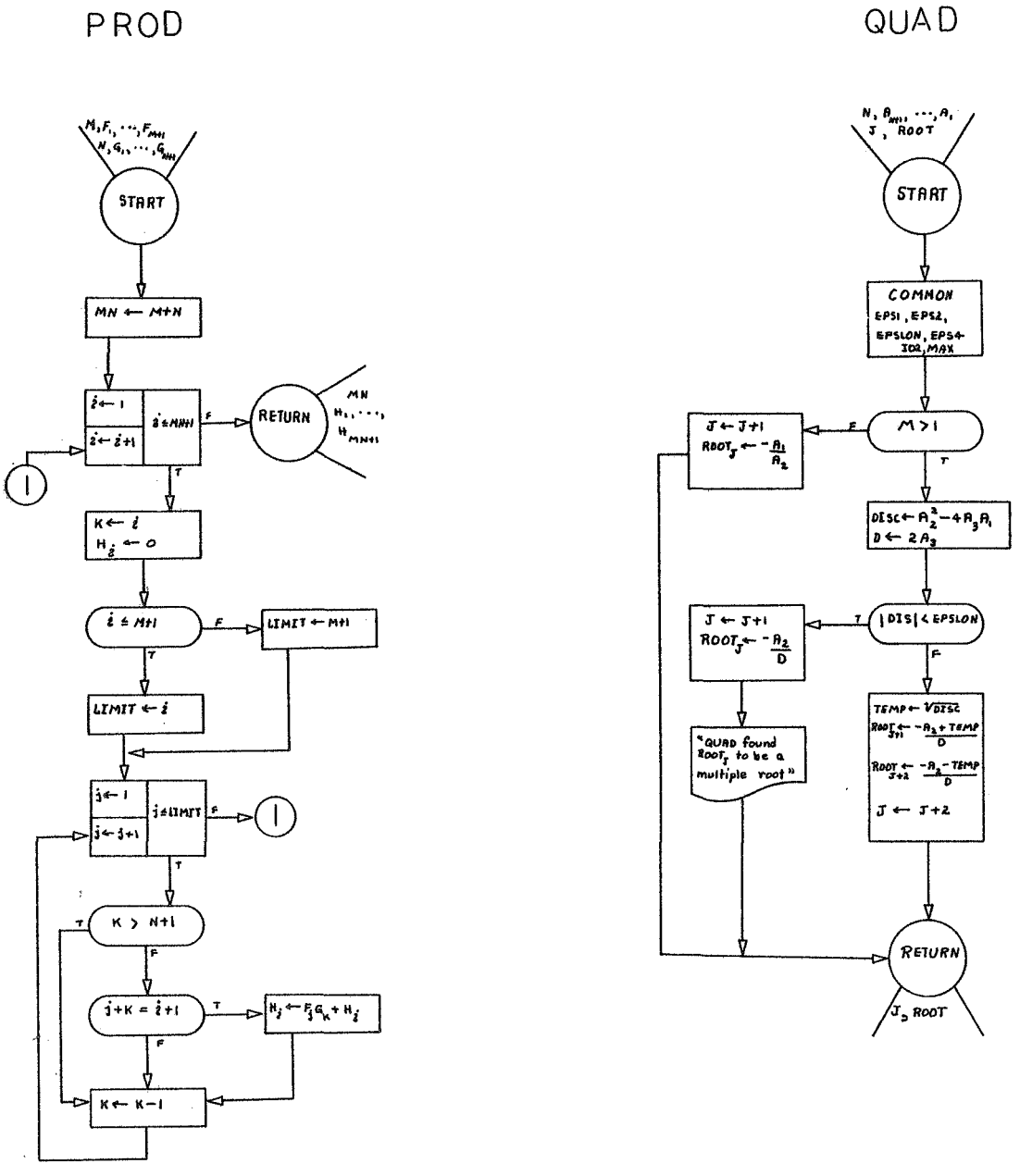
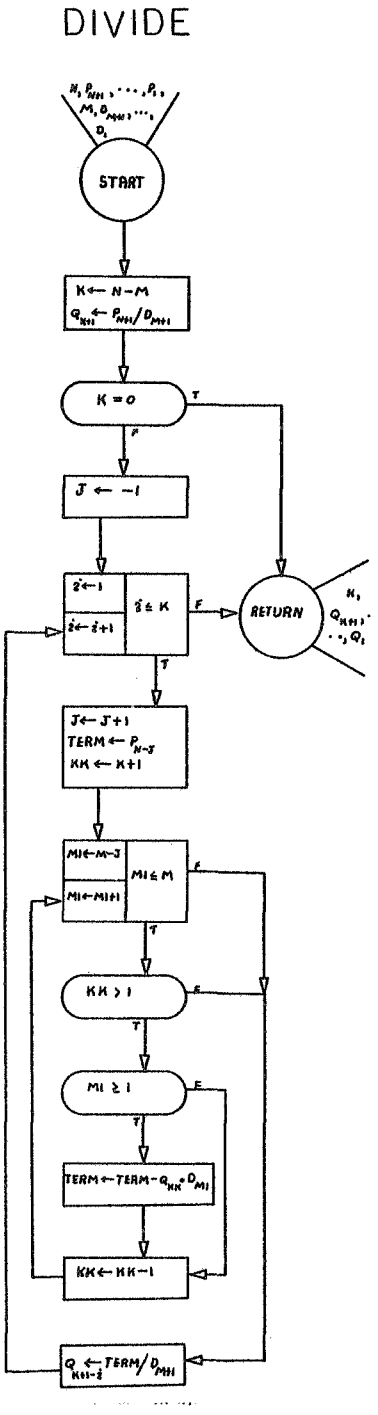


Figure H.1. (Continued)



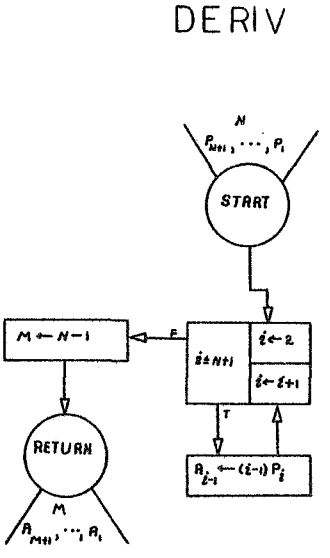


Figure H.1. (Continued)

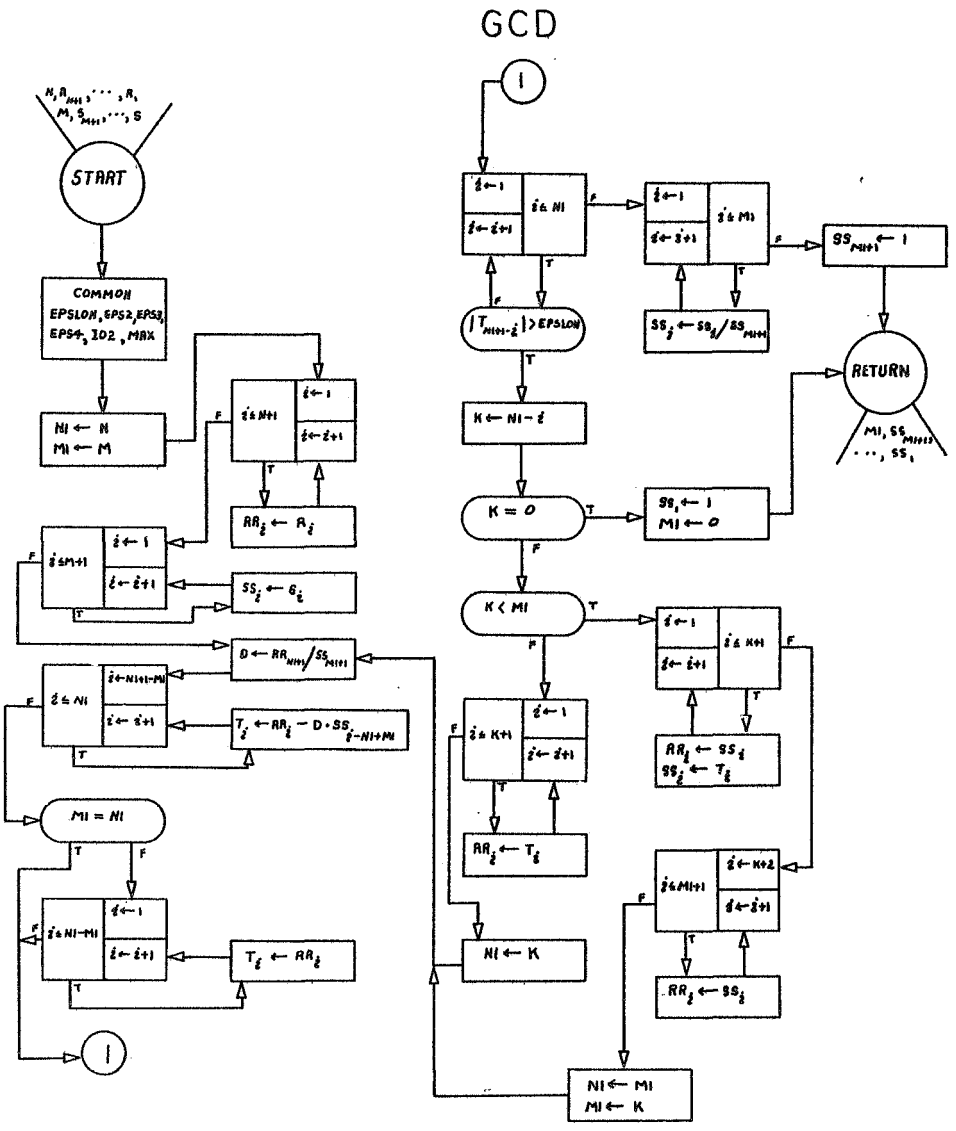


Figure H.1. (Continued)

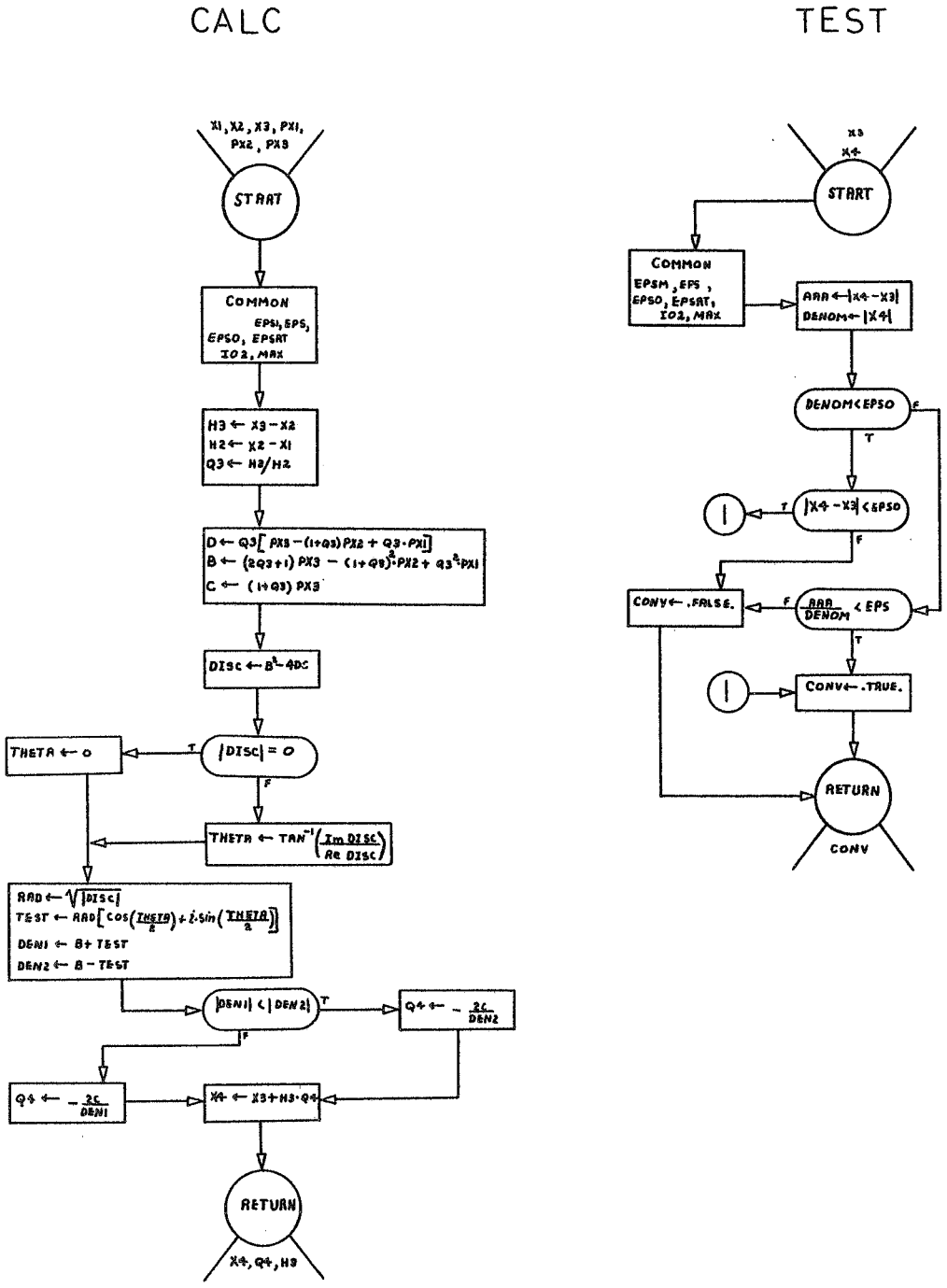


Figure H.1. (Continued)

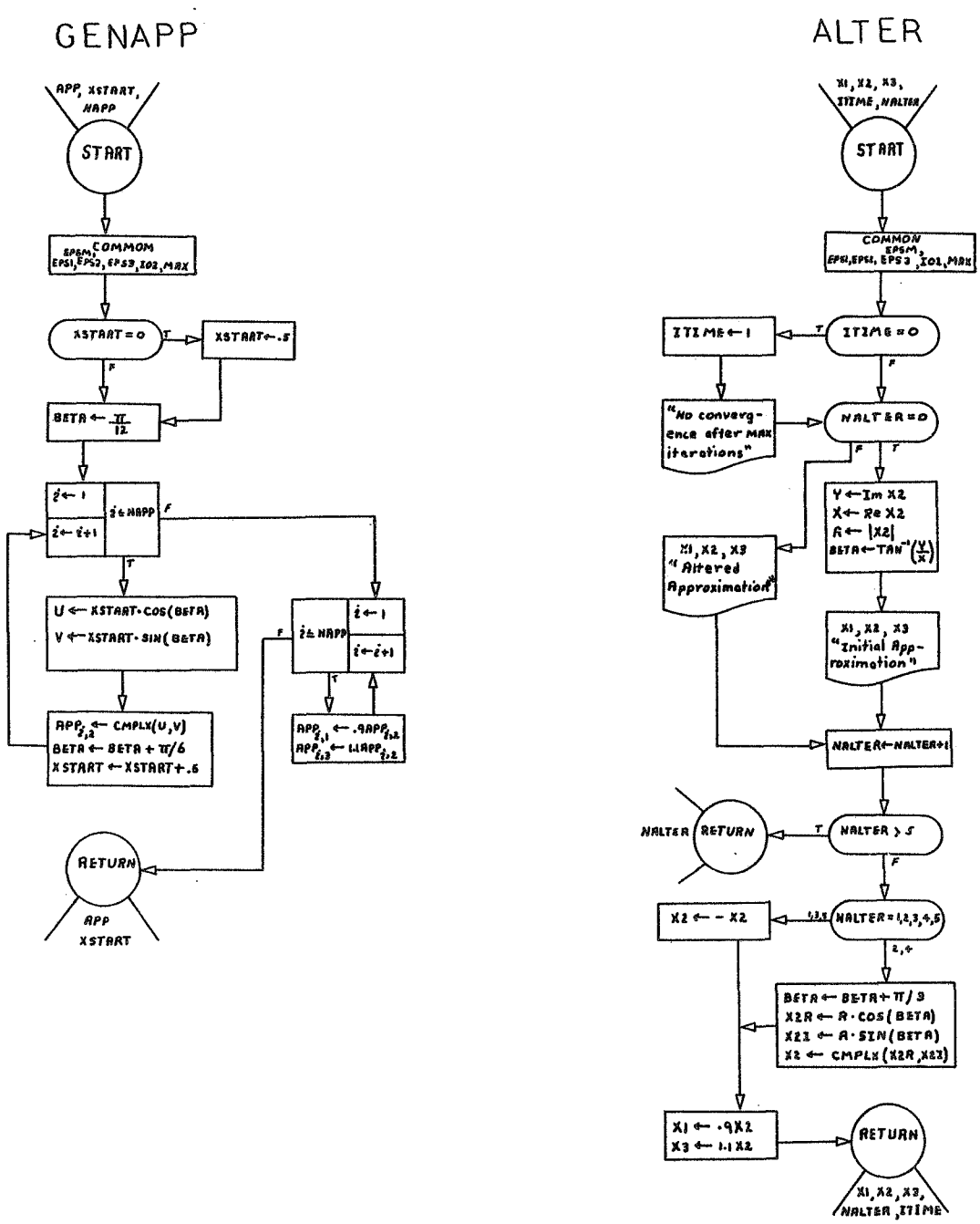


Figure H.1. (Continued)

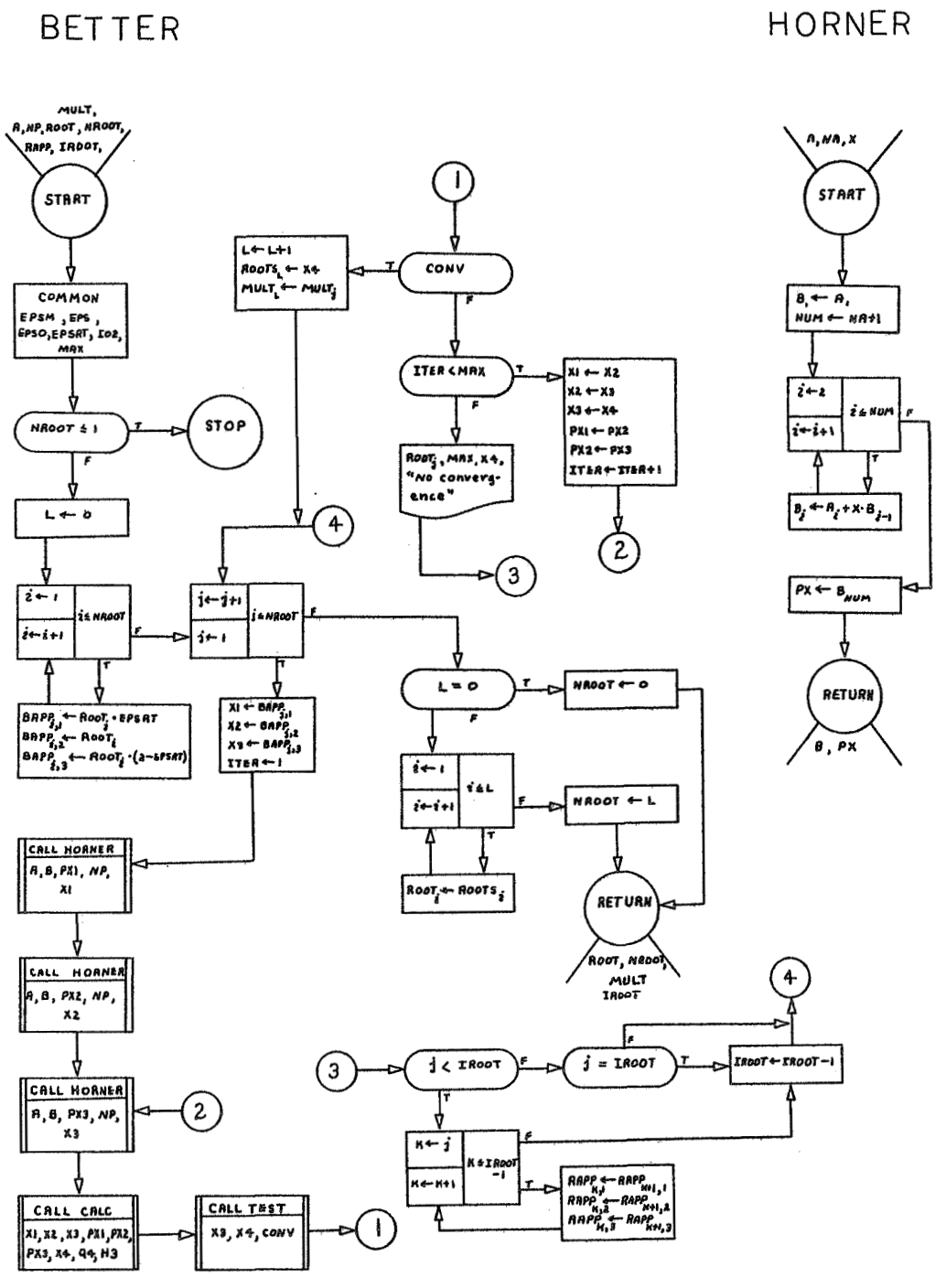
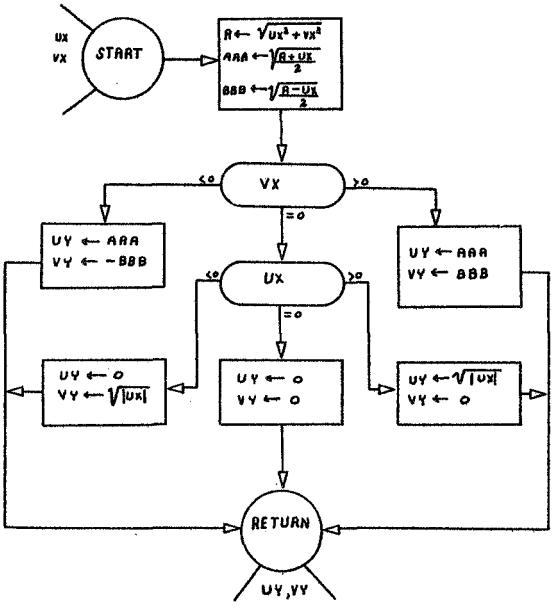


Figure H.1. (Continued)

COMSQT



Figue H.1. (Continued)

TABLE H.III

PROGRAM FOR REPEATED G.C.D.-MULLER'S METHOD

```
C   ************************************************************************
C   *                                                                      *
C   * DOUBLE PRECISION PROGRAM FOR THE REPEATED G.C.D. - MULLER'S METHOD    *
C   *                                                                      *
C   *                                                                      *
C   * THIS METHOD REPEATEDLY FINDS THE GREATEST COMMON DIVISOR OF TWO       *
C   * POLYNOMIALS IN ORDER TO EXTRACT THE ZEROS IN GROUPS ACCORDING TO     *
C   * MULTIPLICITY USING NEWTON'S METHOD.  ALL ZEROS OF MULTIPLICITY 1     *
C   * ARE EXTRACTED FOLLOWED BY THOSE OF MULTIPLICITY 2, ETC.              *
C   *                                                                      *
C   ************************************************************************
0001          DOUBLE PRECISION EPS1,EPS2,EPS3,UP,VP,UAPP,VAPP,UDO,VDO,UDDO,VDDO,
             1UD1,VD1,UD2,VD2,UDD1,VDD1,UG,VG,UD3,VD3,UD4,VD4,UZROS,VZROS,UAP,VA
             2P,UROOT,VROOT,DENOM
0002          DOUBLE PRECISION XSTART
0003          DOUBLE PRECISION XEND
0004          DOUBLE PRECISION URAPP,VRAPP
0005          DOUBLE PRECISION EPS4
0006          DIMENSION UAPP(25,3),VAPP(25,3),URAPP(25,3),VRAPP(25,3)
0007          DIMENSION UP(26),VP(26),MULT(25),UDDO(26),VDDO(26),UD1(26),VD1(26)
             1,UDD1(26),VDD1(26),UD2(26),VD2(26),UG(26),VG(26),UD3(51),VD3(51),U
             2D4(51),VD4(51),UAP(26),VAP(26),UZROS(25),VZROS(25),UROOT(25),VROOT
             3(25),ANAME(2),UDO(26),VDO(26),ENTRY(26)
0008          COMMON EPS1,EPS2,EPS3,EPS4,IO2,MAX
0009          DATA PNAME,GNAME/2HP(,2HG(/, DINAME/3HD1(/
0010          DATA ASTER/4H****/
0011          DATA ENTRY/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,2H12,2H13
             1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
0012          DATA ANAME(1),ANAME(2)/4HMULL,4HERS /
0013          IO1=5
0014          IO2=6
0015        1 READ(IO1,1000) NOPOLY,NP,NAPP,MAX,EPS1,EPS2,EPS3,XSTART,XEND,KCHEC
             1K
0016          IF(KCHECK.EQ.1) STOP
0017          WRITE(IO2,1020) ANAME(1),ANAME(2),NOPOLY
0018          WRITE(IO2,2000) NAPP
0019          WRITE(IO2,2010) MAX
0020          WRITE(IO2,2070) EPS1
0021          WRITE(IO2,2020) EPS2
0022          WRITE(IO2,2080) EPS3
0023          WRITE(IO2,2040) XSTART
0024          WRITE(IO2,2050) XEND
0025          WRITE(IO2,2060)
0026          KKK=NP+1
0027          NNN=KKK+1
0028          DO 5 I=1,KKK
0029          JJJ=NNN-I
0030        5 READ(IO1,1010) UP(JJJ),VP(JJJ)
0031          IF(NAPP.NE.0) GO TO 22
0032          NAPP=NP
0033          CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0034          GO TO 23
0035       22 READ(IO1,1015) (UAPP(I,2),VAPP(I,2),I=1,NAPP)
0036       23 WRITE(IO2,1030) NP
0037          KKK=NP+1
0038          NNN=KKK+1
0039          DO 8 I=1,KKK
0040          JJJ=NNN-I
```

TABLE H.III (Continued)

```
0041            8 WRITE(IO2,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
0042              K=0
0043              KD=0
0044              J1=1
0045              KKK=NP+1
0046              DO 10 I=1,KKK
0047              UDO(I)=UP(I)
0048             10 VDO(I)=VP(I)
0049              NDO=NP
0050              CALL DERIV(NDO,UDO,VDO,NDDO,UDDO,VDDO)
0051              CALL GCD(NDO,UDO,VDO,NDDO,UDDO,VDDO,ND1,UD1,VD1)
0052             20 WRITE(IO2,3000) (ASTER,I=1,33)
0053              IF(ND1.LE.1) GO TO 30
0054              GO TO 40
0055             30 UD2(1)=1.0
0056              VD2(1)=0.0
0057              ND2=0
0058              GO TO 50
0059             40 CALL DERIV(ND1,UD1,VD1,NDD1,UDD1,VDD1)
0060              CALL GCD(ND1,UD1,VD1,NDD1,UDD1,VDD1,ND2,UD2,VD2)
0061             50 IF(NDO+ND2.LE.2*ND1) GO TO 60
0062              GO TO 70
0063             60 WRITE(IO2,1025) J1
0064              GO TO 170
0065             70 IF(ND1.EQ.0) GO TO 80
0066              GO TO 90
0067             80 KKK=NDO+1
0068              DO 85 I=1,KKK
0069              UG(I)=UDO(I)
0070             85 VG(I)=VDO(I)
0071              NG=NDO
0072              GO TO 110
0073             90 IF(ND2.EQ.0) GO TO 115
0074              CALL PROD(NDO,UDO,VDO,ND2,UD2,VD2,ND3,UD3,VD3)
0075            100 CALL PROD(ND1,UD1,VD1,ND1,UD1,VD1,ND4,UD4,VD4)
0076              CALL DIVIDE(ND3,UD3,VD3,ND4,UD4,VD4,NG,UG,VG)
0077            110 WRITE(IO2,1035) J1
0078              KKK=NG+1
0079              NNN=KKK+1
0080              DO 112 I=1,KKK
0081              JJJ=NNN-I
0082            112 WRITE(IO2,1040) GNAME,ENTRY(JJJ),UG(JJJ),VG(JJJ)
0083              KKK=NG+1
0084              DO 113 I=1,KKK
0085              UAP(I)=UG(KKK+1-I)
0086            113 VAP(I)=VG(KKK+1-I)
0087              CALL MULLER(NG,UAP,VAP,NAPP,UAPP,VAPP,J,UZROS,VZROS,JAP,XSTART,XEN
                 1D,NOPOLY,URAPP,VRAPP)
0088              IF(J.EQ.0) GO TO 150
0089              WRITE(IO2,1180)
0090              IF(JAP.EQ.0) GO TO 120
0091              GO TO 130
0092            115 KKK=NDO+1
0093              DO 116 I=1,KKK
0094              UD3(I)=UDO(I)
0095            116 VD3(I)=VDO(I)
0096              ND3=NDO
0097              GO TO 100
```

TABLE H.III (Continued)

```
0098          120 KKK=JAP+1
0099              WRITE(IO2,1085) (I,UZROS(I),VZROS(I),J1,I=KKK,J)
0100              GO TO 140
0101          130 DO 135 I=1,JAP
0102          135 WRITE(IO2,1190) I,UZROS(I),VZROS(I),J1,URAPP(I,2),VRAPP(I,2)
0103              IF(JAP.LT.J) GO TO 120
0104          140 IF(J.EQ.NG) GO TO 155
0105          150 WRITE(IO2,1095)
0106              IF(J.EQ.0) GO TO 170
0107          155 DO 160 I=1,J
0108              UROOT(KD+I)=UZROS(I)
0109              VROOT(KD+I)=VZROS(I)
0110          160 MULT(KD+I)=J1
0111              K=(J*J1)+K
0112              KD=KD+J
0113              IF(K.GE.NP) GO TO 1
0114          170 J1=J1+1
0115              IF(ND1.LE.1) GO TO 200
0116              DO 180 I=1,ND1
0117              UDO(I)=UD1(I)
0118              VDO(I)=VD1(I)
0119              UDDO(I)=UDD1(I)
0120          180 VDDO(I)=VDD1(I)
0121              UDO(ND1+1)=UD1(ND1+1)
0122              VDO(ND1+1)=VD1(ND1+1)
0123              NDO=ND1
0124              NDDO=NDD1
0125              KKK=ND2+1
0126              DO 190 I=1,KKK
0127              UD1(I)=UD2(I)
0128          190 VD1(I)=VD2(I)
0129              ND1=ND2
0130              GO TO 20
0131          200 IF(ND1.EQ.0) GO TO 1
0132              KD=KD+1
0133              DENOM=UD1(2)*UD1(2)+VD1(2)*VD1(2)
0134              UROOT(KD)=(-UD1(1)*UD1(2)-VD1(1)*VD1(2))/DENOM
0135              VROOT(KD)=(-VD1(1)*UD1(2)+UD1(1)*VD1(2))/DENOM
0136              MULT(KD)=J1
0137              WRITE(IO2,3000) (ASTER,I=1,33)
0138              WRITE(IO2,1035) J1
0139              KKK=ND1+1
0140              NNN=KKK+1
0141              DO 210 I=1,KKK
0142              JJJ=NNN-I
0143          210 WRITE(IO2,1100) DINAME,ENTRY(JJJ),UD1(JJJ),VD1(JJJ)
0144              WRITE(IO2,1180)
0145              WRITE(IO2,1085) KD,UROOT(KD),VROOT(KD),J1
0146              GO TO 1
0147         1020 FORMAT(1H1,10X,48HREPEATED USE OF THE GREATEST COMMON DIVISOR AND
                 1,A4,A4,58H METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIA
                 2LS/11X,18HPOLYNOMIAL NUMBER ,I2///)
0148         1025 FORMAT(///1X,25HNO ROOTS OF MULTIPLICITY ,I2//)
0149         1035 FORMAT(///1X,87HTHE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE R
                 1OOTS OF P(X) WHICH HAVE MULTIPLICITY ,I2//)
0150         1085 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,9X,25HN
                 1O INITIAL APPROXIMATIONS)
0151         1095 FORMAT(///1X,51HNOT ALL ROOTS OF THE ABOVE POLYNOMIAL,G, WERE FOUN
```

TABLE H.III (Continued)

```
     1D//)
0152 1000 FORMAT(3(I2,1X),9X,I3,1X,3(D6.0,1X),20X,2(D7.0,1X),I1)
0153 1010 FORMAT(2D30.0)
0154 1015 FORMAT(2D30.0)
0155 1030 FORMAT(1X,22HTHE DEGREE OF P(X) IS ,I2,22H  THE COEFFICIENTS ARE//
     1)
0156 1040 FORMAT(2X,A2,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
0157 1100 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H I)
0158 1180 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
     1 APPROXIMATION//)
0159 1190 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,8X,I2,8X,D23.
     116,3H + ,D23.16,2H I)
0160 2000 FORMAT(1X,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.  ,I2)
0161 2010 FORMAT(1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,I3)
0162 2020 FORMAT(1X,21HTEST FOR CONVERGENCE.,13X,D9.2)
0163 2040 FORMAT(1X,23HRADIUS TO START SEARCH.,11X,D9.2)
0164 2050 FORMAT(1X,21HRADIUS TO END SEARCH.,13X,D9.2)
0165 2060 FORMAT(//1X)
0166 2070 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE GCD.  ,D9.2)
0167 2080 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE QUAD. ,D9.2)
0168 3000 FORMAT(/////1X,A3,32A4)
0169       END
```

TABLE H.III (Continued)

```
0001              SUBROUTINE PROD(M,UF,VF,N,UG,VG,MN,UH,VH)
                C ********************************************************************************
                C *                                                                              *
                C * GIVEN POLYNOMIALS R(X) AND S(X), THIS SUBROUTINE COMPUTES THE                *
                C * COEFFICIENTS OF THE PRODUCT POLYNOMIAL T(X) = R(X).S(X).                     *
                C *                                                                              *
                C ********************************************************************************
0002              DOUBLE PRECISION UH,VH,UF,VF,UG,VG
0003              DIMENSION UH(51),VH(51),UF(26),VF(26),UG(26),VG(26)
0004              MN=M+N
0005              KKK=MN+1
0006              DO 100 I=1,KKK
0007              K=I
0008              UH(I)=0.0
0009              VH(I)=0.0
0010              IF(I.LE.M+1) GO TO 10
0011              LIMIT=M+1
0012              GO TO 20
0013             10 LIMIT=I
0014             20 DO 50 J=1,LIMIT
0015              IF(K.GT.N+1) GO TO 50
0016              IF(J+K.EQ.I+1) GO TO 40
0017              GO TO 50
0018             40 UH(I)=UH(I)+(UF(J)*UG(K)-VF(J)*VG(K))
0019              VH(I)=VH(I)+(VF(J)*UG(K)+UF(J)*VG(K))
0020             50 K=K-1
0021            100 CONTINUE
0022              RETURN
0023              END
```

TABLE H.III. (Continued)

```
0001              SUBROUTINE QUAD(N,UA,VA,J,UROOT,VROOT)
                C   ************************************************************************
                C   *                                                                      *
                C   * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
                C   * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR.  SOLUTION OF THE  *
                C   * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                        *
                C   *                                                                      *
                C   ************************************************************************
0002              DOUBLE PRECISION EPS1,EPS2,EPSLON,UROOT,VROOT,UA,VA,UDISC,VDISC,UD
                 1,VD,DDD,UTEMP,VTEMP,BBB
0003              DOUBLE PRECISION EPS4
0004              DIMENSION UROOT(25),VROOT(25),UA(26),VA(26)
0005              COMMON EPS1,EPS2,EPSLON,EPS4,IO2,MAX
0006              IF(N.GT.1) GO TO 10
0007              J=J+1
0008              BBB=UA(2)*UA(2)+VA(2)*VA(2)
0009              UROOT(J)=-(UA(1)*UA(2)+VA(1)*VA(2))/BBB
0010              VROOT(J)=-(VA(1)*UA(2)-UA(1)*VA(2))/BBB
0011              GO TO 100
0012           10 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(3)*UA(1)-VA(3)*VA(1)))
0013              VDISC=(2.0*UA(2)*VA(2))-(4.0*(UA(3)*VA(1)+VA(3)*UA(1)))
0014              UD=2.0*UA(3)
0015              VD=2.0*VA(3)
0016              DDD=DSQRT(UDISC*UDISC+VDISC*VDISC)
0017              IF(DDD.LT.EPSLON) GO TO 20
0018              CALL COMSQT(UDISC,VDISC,UTEMP,VTEMP)
0019              BBB=UD*UD+VD*VD
0020              UROOT(J+1)=((-UA(2)+UTEMP)*UD+(-VA(2)+VTEMP)*VD)/BBB
0021              VROOT(J+1)=((-VA(2)+VTEMP)*UD-(-UA(2)+UTEMP)*VD)/BBB
0022              UROOT(J+2)=((-UA(2)-UTEMP)*UD+(-VA(2)-VTEMP)*VD)/BBB
0023              VROOT(J+2)=((-VA(2)-VTEMP)*UD-(-UA(2)-UTEMP)*VD)/BBB
0024              J=J+2
0025              GO TO 100
0026           20 J=J+1
0027              BBB=UD*UD+VD*VD
0028              UROOT(J)=(-UA(2)*UD-VA(2)*VD)/BBB
0029              VROOT(J)=(-VA(2)*UD+UA(2)*VD)/BBB
0030              WRITE(IO2,1000) UROOT(J),VROOT(J)
0031         1000 FORMAT(///1X,11HQUAD FOUND ,D23.16,3H + ,D23.16,2H I,22H TO BE A M
                 1ULTIPLE ROOT//)
0032          100 RETURN
0033              END
```

TABLE H.III (Continued)

```
0001            SUBROUTINE DERIV(N,UP,VP,M,UA,VA)
              C ********************************************************************************
              C *                                                                              *
              C * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF       *
              C * ITS DERIVATIVE P'(X).                                                        *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UP,VP,UA,VA,AAA
0003            DIMENSION UP(26),VP(26),UA(26),VA(26)
0004            KKK=N+1
0005            DO 10 I=2,KKK
0006            AAA=I-1
0007            UA(I-1)=AAA*UP(I)
0008          10 VA(I-1)=AAA*VP(I)
0009            M=N-1
0010            RETURN
0011            END
```

TABLE H.III (Continued)

```
0001            SUBROUTINE GCD(N,UR,VR,M,US,VS,M1,USS,VSS)
                C  ***************************************************************************
                C  *                                                                         *
                C  * GIVEN POLYNOMIALS P(X) AND DP(X) WHERE DEG. DP(X) IS LESS THAN DEG.     *
                C  * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND   *
                C  * DP(X).                                                                 *
                C  *                                                                         *
                C  ***************************************************************************
0002            DOUBLE PRECISION USSSSS,VSSSSS
0003            DOUBLE PRECISION UR,VR,US,VS,USS,VSS,URR,VRR,UD,VD,UT,VT,EPSLON,EP
               1S2,EPS3,EPS4,BBB
0004            DIMENSION UR(26),VR(26),US(26),VS(26),USS(26),VSS(26),URR(26),VRR(
               126),UT(26),VT(26)
0005            COMMON EPSLON,EPS2,EPS3,EPS4,IO2,MAX
0006            N1=N
0007            M1=M
0008            KKK=N+1
0009            DO 20 I=1,KKK
0010            URR(I)=UR(I)
0011         20 VRR(I)=VR(I)
0012            KKK=M+1
0013            DO 25 I=1,KKK
0014            USS(I)=US(I)
0015         25 VSS(I)=VS(I)
0016         30 BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0017            UD=(URR(N1+1)*USS(M1+1)+VRR(N1+1)*VSS(M1+1))/BBB
0018            VD=(USS(M1+1)*VRR(N1+1)-URR(N1+1)*VSS(M1+1))/BBB
0019            KKK=N1+1-M1
0020            DO 40 I=KKK,N1
0021            UT(I)=URR(I)-(UD*USS(I-N1+M1)-VD*VSS(I-N1+M1))
0022         40 VT(I)=VRR(I)-(UD*VSS(I-N1+M1)+VD*USS(I-N1+M1))
0023            IF(M1.EQ.N1) GO TO 70
0024            KKK=N1-M1
0025            DO 60 I=1,KKK
0026            UT(I)=URR(I)
0027         60 VT(I)=VRR(I)
0028         70 DO 90 I=1,N1
0029            BBB=DSQRT(UT(N1+1-I)*UT(N1+1-I)+VT(N1+1-I)*VT(N1+1-I))
0030            IF(BBB.GT.EPSLON) GO TO 100
0031         90 CONTINUE
0032            DO 95 I=1,M1
0033            BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)
0034            USSSSS=(USS(I)*USS(M1+1)+VSS(I)*VSS(M1+1))/BBB
0035            VSSSSS=(VSS(I)*USS(M1+1)-USS(I)*VSS(M1+1))/BBB
0036            USS(I)=USSSSS
0037         95 VSS(I)=VSSSSS
0038            USS(M1+1)=1.0
0039            VSS(M1+1)=0.0
0040            GO TO 200
0041        100 K=N1-I
0042            IF(K.EQ.0) GO TO 170
0043            IF(K.LT.M1) GO TO 140
0044            KKK=K+1
0045            DO 130 J=1,KKK
0046            URR(J)=UT(J)
0047        130 VRR(J)=VT(J)
0048            N1=K
0049            GO TO 30
```

TABLE H.III (Continued)

```
0050          140 KKK=K+1
0051              DO 150 J=1,KKK
0052              URR(J)=USS(J)
0053              VRR(J)=VSS(J)
0054              USS(J)=UT(J)
0055          150 VSS(J)=VT(J)
0056              KKK=K+2
0057              NNN=M1+1
0058              DO 160 J=KKK,NNN
0059              URR(J)=USS(J)
0060          160 VRR(J)=VSS(J)
0061              N1=M1
0062              M1=K
0063              GO TO 30
0064          170 USS(1)=1.0
0065              VSS(1)=0.0
0066              M1=0
0067          200 RETURN
0068              END
```

TABLE H.III (Continued)

```
0001            SUBROUTINE DIVIDE(N,UP,VP,M,UD,VD,K,UQ,VQ)
              C ********************************************************************************
              C *                                                                              *
              C * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE          *
              C * QUOTIENT POLYNOMIAL  H(X) = F(X)/G(X).                                       *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UP,VP,UD,VD,UQ,VQ,UTERM,VTERM,UDUMMY
0003            DIMENSION UP(26),VP(26),UD(26),VD(26),UQ(26),VQ(26)
0004            K=N-M
0005            UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0006            UQ(K+1)=(UP(N+1)*UD(M+1)+VP(N+1)*VD(M+1))/UDUMMY
0007            VQ(K+1)=(VP(N+1)*UD(M+1)-UP(N+1)*VD(M+1))/UDUMMY
0008            IF(K.EQ.0) GO TO 100
0009            J=-1
0010            DO 50 I=1,K
0011            J=J+1
0012            UTERM=UP(N-J)
0013            VTERM=VP(N-J)
0014            KK=K+1
0015            NNN=M-J
0016            DO 40 M1=NNN,M
0017            IF(KK.GT.1) GO TO 10
0018            GO TO 45
0019           10 IF(M1.GE.1) GO TO 20
0020            GO TO 40
0021           20 UTERM=UTERM-(UQ(KK)*UD(M1)-VQ(KK)*VD(M1))
0022            VTERM=VTERM-(UQ(KK)*VD(M1)+VQ(KK)*UD(M1))
0023           40 KK=KK-1
0024           45 UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0025            UQ(K+1-I)=(UTERM*UD(M+1)+VTERM*VD(M+1))/UDUMMY
0026           50 VQ(K+1-I)=(VTERM*UD(M+1)-UTERM*VD(M+1))/UDUMMY
0027          100 RETURN
0028            END
```

TABLE H.III (Continued)

```
0001            SUBROUTINE CONSQT(UX,VX,UY,VY)
              C ********************************************************************************
              C *                                                                              *
              C * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.                *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
0003            R=DSQRT(UX*UX+VX*VX)
0004            AAA=DSQRT(DABS((R+UX)/2.0))
0005            BBB=DSQRT(DABS((R-UX)/2.0))
0006            IF(VX) 10,20,30
0007           10 UY=AAA
0008            VY=-1.0*BBB
0009            GO TO 100
0010           20 IF(UX) 40,50,60
0011           30 UY=AAA
0012            VY=BBB
0013            GO TO 100
0014           40 DUMMY=DABS(UX)
0015            UY=0.0
0016            VY=DSQRT(DUMMY)
0017            GO TO 100
0018           50 UY=0.0
0019            VY=0.0
0020            GO TO 100
0021           60 DUMMY=DABS(UX)
0022            UY=DSQRT(DUMMY)
0023            VY=0.0
0024          100 RETURN
0025            END
```

TABLE H.III (Continued)

```
0001              SUBROUTINE CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,V
                 1PX3,UX4,VX4,UQ4,VQ4,UH3,VH3)
                C*************************************************************************
                C*                                                                       *
                C* GIVEN THREE APPROXIMATIONS X(N-2), X(N-1), AND X(N), SUBROUTINE CALC  *
                C* APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF *
                C* THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION     *
                C* X(N+1) TO THE ZERO OF THE POLYNOMIAL.                                 *
                C*                                                                       *
                C*************************************************************************
0002              DOUBLE PRECISION ARG1,ARG2
0003              DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1,
                 1VPX1,UH3,VH3,UH2,VH2,UQ3,VQ3,UD,VD,UB,VB,UC,VC,UDISC,VDISC,UCCC,VC
                 2CC,UDEN1,VDEN1,UDEN2,VDEN2,UQ4,VQ4,UX4,VX4,EPSRT,EPSO,EPS,UDDD,VDD
                 3D,AAA,BBB,RAD,UAAA,VAAA,UBBB,VBBB
0004              DOUBLE PRECISION THETA,ANGLE,UTEST,VTEST
0005              DOUBLE PRECISION EPS1
0006              COMMON EPS1,EPS,EPSO,EPSRT,IO2,MAX
0007              UH3=UX3-UX2
0008              VH3=VX3-VX2
0009              UH2=UX2-UX1
0010              VH2=VX2-VX1
0011              BBB=UH2*UH2+VH2*VH2
0012              UQ3=(UH3*UH2+VH3*VH2)/BBB
0013              VQ3=(VH3*UH2-UH3*VH2)/BBB
0014              UDDD=1.0+UQ3
0015              VDDD=VQ3
0016              UD=(UPX3-(UDDD*UPX2-VDDD*VPX2))+(UQ3*UPX1-VQ3*VPX1)
0017              VD=(VPX3-(VDDD*UPX2+UDDD*VPX2))+(VQ3*UPX1+UQ3*VPX1)
0018              UAAA=2.0*UQ3
0019              VAAA=2.0*VQ3
0020              UAAA=UAAA+1.0
0021              UBBB=UDDD*UDDD-VDDD*VDDD
0022              VBBB=VDDD*UDDD+UDDD*VDDD
0023              UCCC=UQ3*UQ3-VQ3*VQ3
0024              VCCC=VQ3*UQ3+UQ3*VQ3
0025              UB=((UAAA*UPX3-VAAA*VPX3)-(UBBB*UPX2-VBBB*VPX2))+(UCCC*UPX1-VCCC*V
                 1PX1)
0026              VB=((VAAA*UPX3+UAAA*VPX3)-(VBBB*UPX2+UBBB*VPX2))+(VCCC*UPX1+UCCC*V
                 1PX1)
0027              UC=UDDD*UPX3-VDDD*VPX3
0028              VC=VDDD*UPX3+UDDD*VPX3
0029              UDISC=(UB*UB-VB*VB)-(4.0*(UD*UC-VD*VC))
0030              VDISC=(2.0*(VB*UB))-(4.0*(VD*UC+UD*VC))
0031              AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
0032              IF(AAA.EQ.0.0) GO TO 5
0033              GO TO 7
0034            5 THETA=0.0
0035              GO TO 9
0036            7 THETA=DATAN2(VDISC,UDISC)
0037            9 RAD=DSQRT(AAA)
0038              ANGLE=THETA/2.0
0039              UTEST=RAD*DCOS(ANGLE)
0040              VTEST=RAD*DSIN(ANGLE)
0041              UDEN1=UB+UTEST
0042              VDEN1=VB+VTEST
0043              UDEN2=UB-UTEST
0044              VDEN2=VB-VTEST
```

TABLE H.III (Continued)

```
0045              ARG1=UDEN1*UDEN1+VDEN1*VDEN1
0046              ARG2=UDEN2*UDEN2+VDEN2*VDEN2
0047              AAA=DSQRT(ARG1)
0048              BBB=DSQRT(ARG2)
0049              IF(AAA.LT.BBB) GO TO 10
0050              IF(AAA.EQ.0.0) GO TO 60
0051              UAAA=-2.0*UC
0052              VAAA=-2.0*VC
0053              UQ4=(UAAA*UDEN1+VAAA*VDEN1)/ARG1
0054              VQ4=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
0055              GO TO 50
0056           10 IF(BBB.EQ.0.0) GO TO 60
0057              UAAA=-2.0*UC
0058              VAAA=-2.0*VC
0059              UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2
0060              VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2
0061              GO TO 50
0062           50 UX4=UX3+(UH3*UQ4-VH3*VQ4)
0063              VX4=VX3+(VH3*UQ4+UH3*VQ4)
0064              RETURN
0065           60 UQ4=1.0
0066              VQ4=0.0
0067              GO TO 50
0068              END
```

TABLE H.III (Continued)

```
0001            SUBROUTINE MULLER(NP,UA,VA,NAPP,UAPP,VAPP,NROOT,UROOT,VROOT,IROOT,
               1XSTART,XEND,NOPOLY,URAPP,VRAPP)
          C    ************************************************************************
          C    *                                                                      *
          C    * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A     *
          C    * POLYNOMIAL OF MAXIMUM DEGREE 25.  THROUGH THREE GIVEN POINTS THE     *
          C    * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC.  THE ZERO OF THE QUADRATIC *
          C    * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.  *
          C    * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.          *
          C    *                                                                      *
          C    ************************************************************************
0002            DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,VX1,UAPP,VAPP
               1,UX2,VX2,UWORK,VWORK,UX3,VX3,UB,VB,UX4,VX4,UA,VA,UPX1,VPX1,URAPP,V
               2RAPP,UPX4,VPX4,EPSRT,EPSO,EPS,CCC,EPSM,UH3,VH3,UQ4,VQ4,ABPX4,ABPX3
               3,QQQ,XSTART,XEND
0003            DIMENSION UROOT(25),VROOT(25),MULT(25),UAPP(25,3),VAPP(25,3),UWORK
               1(26),VWORK(26),UB(26),VB(26),UA(26),VA(26),URAPP(25,3),VRAPP(25,3)
0004            LOGICAL CONV
0005            COMMON EPSM,EPS,EPSO,EPSRT,IO2,MAX
0006            DATA PNAME,DNAME/2HP(,2HD(/
0007            EPSM=0.0D00
0008            EPSRT=0.999
0009            NROOT=0
0010            IROOT=0
0011            IPATH=1
0012            NOMULT=0
0013            NALTER=0
0014            ITIME=0
0015            IAPP=1
0016            ITER=1
0017            IF(NAPP.NE.0) GO TO 18
0018            NAPP=NP
0019            CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0020            GO TO 27
0021         18 DO 25 I=1,NAPP
0022            UAPP(I,1)=0.9*UAPP(I,2)
0023            VAPP(I,1)=0.9*VAPP(I,2)
0024            UAPP(I,3)=1.1*UAPP(I,2)
0025         25 VAPP(I,3)=1.1*VAPP(I,2)
0026         27 KKK=NP+1
0027            DO 30 I=1,KKK
0028            UWORK(I)=UA(I)
0029         30 VWORK(I)=VA(I)
0030            NWORK=NP
0031         40 UX1=UAPP(IAPP,1)
0032            VX1=VAPP(IAPP,1)
0033            UX2=UAPP(IAPP,2)
0034            VX2=VAPP(IAPP,2)
0035            UX3=UAPP(IAPP,3)
0036            VX3=VAPP(IAPP,3)
0037            CALL HORNER(NWORK,UWORK,VWORK,UX1,VX1,UB,VB,UPX1,VPX1)
0038            CALL HORNER(NWORK,UWORK,VWORK,UX2,VX2,UB,VB,UPX2,VPX2)
0039            CALL HORNER(NWORK,UWORK,VWORK,UX3,VX3,UB,VB,UPX3,VPX3)
0040         50 CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
               14,VX4,UQ4,VQ4,UH3,VH3)
0041         60 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0042            ABPX4=DSQRT(UPX4*UPX4+VPX4*VPX4)
0043            ABPX3=DSQRT(UPX3*UPX3+VPX3*VPX3)
```

TABLE H.III (Continued)

```
0044              IF(ABPX3.EQ.0.0) GO TO 70
0045              QQQ=ABPX4/ABPX3
0046              IF(QQQ.LE.10.) GO TO 70
0047              UQ4=0.5*UQ4
0048              VQ4=0.5*VQ4
0049              UX4=UX3+(UH3*UQ4-VH3*VQ4)
0050              VX4=VX3+(VH3*UQ4+UH3*VQ4)
0051              GO TO 60
0052           70 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0053              IF(CONV) GO TO 120
0054              IF(ITER.LT.MAX) GO TO 110
0055              CALL ALTER(UAPP(IAPP,1),VAPP(IAPP,1),UAPP(IAPP,2),VAPP(IAPP,2),UAP
                 1P(IAPP,3),VAPP(IAPP,3),NALTER,ITIME)
0056              IF(NALTER.GT.5) GO TO 75
0057              ITER=1
0058              GO TO 40
0059           75 IF(IAPP.LT.NAPP) GO TO 100
0060              IF(XEND.EQ.0.0) GO TO 77
0061              IF(XSTART.GT.XEND) GO TO 77
0062              NAPP=NP
0063              CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0064              IAPP=0
0065              GO TO 100
0066           77 WRITE(IO2,1090)
0067              KKK=NWORK+1
0068              WRITE(IO2,1035) (ONAME,J,UWORK(J),VWORK(J),J=1,KKK)
0069           80 IF(NROOT.EQ.0) GO TO 90
0070              IF(IPATH.EQ.1) GO TO 82
0071           81 IPATH=2
0072              CALL BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MULT)
0073              RETURN
0074           82 IF(NROOT.EQ.0)GO TO 90
0075              IF(IROOT.EQ.0) GO TO 85
0076              WRITE(IO2,1080)
0077              DO 55 I=1,IROOT
0078           55 WRITE(IO2,1085) I,UROOT(I),VROOT(I),URAPP(I,2),VRAPP(I,2)
0079              IF(IROOT.LT.NROOT) GO TO 85
0080              GO TO 87
0081           85 KKK=IROOT+1
0082              WRITE(IO2,1086) (I,UROOT(I),VROOT(I),I=KKK,NROOT)
0083           87 IF(IPATH.EQ.1) GO TO 81
0084              RETURN
0085           90 WRITE(IO2,1070) NOPOLY
0086              RETURN
0087          100 IAPP=IAPP+1
0088              ITER=1
0089              NALTER=0
0090              GO TO 40
0091          120 NROOT=NROOT+1
0092              IROOT=NROOT
0093              MULT(NROOT)=1
0094              NOMULT=NOMULT+1
0095              UROOT(NROOT)=UX4
0096              VROOT(NROOT)=VX4
0097              URAPP(NROOT,1)=UAPP(IAPP,1)
0098              VRAPP(NROOT,1)=VAPP(IAPP,1)
0099              URAPP(NROOT,2)=UAPP(IAPP,2)
0100              VRAPP(NROOT,2)=VAPP(IAPP,2)
```

TABLE H.III (Continued)

```
0101              URAPP(NROOT,3)=UAPP(IAPP,3)
0102              VRAPP(NROOT,3)=VAPP(IAPP,3)
0103           125 IF(NOMULT.LT.NP) GO TO 130
0104              GO TO 80
0105           130 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0106              NWORK=NWORK-1
0107              KKK=NWORK+1
0108              DO 140 I=1,KKK
0109              UWORK(I)=UB(I)
0110           140 VWORK(I)=VB(I)
0111              CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0112              CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
0113              IF(CCC.LT.EPSM) GO TO 150
0114              IF(NWORK.GT.2) GO TO 75
0115              IROOT=NROOT
0116              KKK=NWORK+1
0117              DO 145 I=1,KKK
0118              UB(I)=UWORK(KKK+1-I)
0119           145 VB(I)=VWORK(KKK+1-I)
0120              CALL QUAD(NWORK,UB,VB,NROOT,UROOT,VROOT)
0121              GO TO 80
0122           150 MULT(NROOT)=MULT(NROOT)+1
0123              NOMULT=NOMULT+1
0124              GO TO 125
0125           110 UX1=UX2
0126              VX1=VX2
0127              UX2=UX3
0128              VX2=VX3
0129              UX3=UX4
0130              VX3=VX4
0131              UPX1=UPX2
0132              VPX1=VPX2
0133              UPX2=UPX3
0134              VPX2=VPX3
0135              UPX3=UPX4
0136              VPX3=VPX4
0137              ITER=ITER+1
0138              GO TO 50
0139          1090 FORMAT(///,1X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
                  1ZEROS WERE FOUND//)
0140          1080 FORMAT(///1X,13HROOTS OF G(X),83X,21HINITIAL APPROXIMATION//)
0141          1070 FORMAT(//,43H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,I2)
0142          1086 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,19X,23HSOLVED
                  1 BY DIRECT METHOD)
0143          1035 FORMAT(3X,A2,I2,4H) = ,D23.16,3H + ,D23.16,2H I)
0144          1085 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H I,18X,D23.16,3H
                  1 + ,D23.16,2H I)
0145          1000 FORMAT(3(I2,1X),9X,I3,8X,3(D6.0,1X),13X,2(D7.0,1X),I1)
0146              END
```

TABLE H.III (Continued)

```
0001              SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
                C ***************************************************************************
                C *                                                                         *
                C * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE    *
                C * DEGREE OF THE ORIGINAL POLYNOMIAL.                                      *
                C *                                                                         *
                C ***************************************************************************
0002              DOUBLE PRECISION APPR,APPI,XSTART,EPS1,EPS2,EPS3,BETA
0003              DOUBLE PRECISION EPSM
0004              DIMENSION APPR(25,3),APPI(25,3)
0005              COMMON EPSM,EPS1,EPS2,EPS3,IO2,MAX
0006              IF(XSTART.EQ.0.0) XSTART=0.5
0007              BETA=0.2617994
0008              DO 10 I=1,NAPP
0009              APPR(I,2)=XSTART*DCOS(BETA)
0010              APPI(I,2)=XSTART*DSIN(BETA)
0011              BETA=BETA+0.5235988
0012           10 XSTART=XSTART+0.5
0013              DO 20 I=1,NAPP
0014              APPR(I,1)=0.9*APPR(I,2)
0015              APPI(I,1)=0.9*APPI(I,2)
0016              APPR(I,3)=1.1*APPR(I,2)
0017           20 APPI(I,3)=1.1*APPI(I,2)
0018              RETURN
0019              END
```

TABLE H.III (Continued)

```
0001            SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)
              C   ************************************************************************
              C   *                                                                      *
              C   * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO   *
              C   * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.*
              C   *                                                                      *
              C   ************************************************************************
0002            DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA
0003            DOUBLE PRECISION EPSM
0004            COMMON EPSM,EPS1,EPS2,EPS3,IO2,MAX
0005            IF(ITIME.NE.0) GO TO 5
0006            ITIME=1
0007            WRITE(IO2,1010) MAX
0008          5 IF(NALTER.EQ.0) GO TO 10
0009            WRITE(IO2,1000) X1R,X1I,X2R,X2I,X3R,X3I
0010            GO TO 20
0011         10 R=DSQRT(X2R*X2R+X2I*X2I)
0012            BETA=DATAN2(X2I,X2R)
0013            WRITE(IO2,1020) X1R,X1I,X2R,X2I,X3R,X3I
0014         20 NALTER=NALTER+1
0015            IF(NALTER.GT.5) RETURN
0016            GO TO (30,40,30,40,30),NALTER
0017         30 X2R=-X2R
0018            X2I=-X2I
0019            GO TO 50
0020         40 BETA=BETA+1.0471976
0021            X2R=R*DCOS(BETA)
0022            X2I=R*DSIN(BETA)
0023         50 X1R=0.9*X2R
0024            X1I=0.9*X2I
0025            X3R=1.1*X2R
0026            X3I=1.1*X2I
0027            RETURN
0028       1000 FORMAT(1X,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HALTERED APPROXIM
                1ATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
                2,D23.16,2H I/)
0029       1020 FORMAT(1HO,5HX1 = ,D23.16,3H + ,D23.16,2H I,10X,22HINITIAL APPROXI
                1MATIONS/1X,5HX2 = ,D23.16,3H + ,D23.16,2H I/1X,5HX3 = ,D23.16,3H +
                2 ,D23.16,2H I/)
0030       1010 FORMAT(///1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
                1TER ,I3,12H ITERATIONS.//)
0031            END
```

TABLE H.III (Continued)

```
0001            SUBROUTINE BETTER(UA,VA,NP,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MUL
               1T)
          C    ********************************************************************************
          C    *                                                                              *
          C    * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND        *
          C    * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO      *
          C    * THE FULL, UNDEFLATED POLYNOMIAL.                                            *
          C    *                                                                              *
          C    ********************************************************************************
0002            DOUBLE PRECISION UROOT,VROOT,UA,VA,UBAPP,VBAPP,UX1,VX1,UX2,VX2,UX3
               1,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UB,VB,UROOTS,VROOTS,EPSRT,UX4,V
               2X4,URAPP,VRAPP,EPSQ,EPS,UQ4,VQ4,UH3,VH3
0003            DOUBLE PRECISION EPSM
0004            LOGICAL CONV
0005            DIMENSION UROOT(25),VROOT(25),UA(26),VA(26),UBAPP(25,3),VBAPP(25,3
               1),UB(26),VB(26),UROOTS(25),VROOTS(25),URAPP(25,3),VRAPP(25,3),MULT
               3(25)
0006            COMMON EPSM,EPS,EPSQ,EPSRT,IO2,MAX
0007            IF(NROOT.LE.1) RETURN
0008            L=0
0009            DO 10 I=1,NROOT
0010            UBAPP(I,1)=UROOT(I)*EPSRT
0011            VBAPP(I,1)=VROOT(I)*EPSRT
0012            UBAPP(I,2)=UROOT(I)
0013            VBAPP(I,2)=VROOT(I)
0014            UBAPP(I,3)=UROOT(I)*(2.0-EPSRT)
0015          10 VBAPP(I,3)=VROOT(I)*(2.0-EPSRT)
0016            DO 100 J=1,NROOT
0017            UX1=UBAPP(J,1)
0018            VX1=VBAPP(J,1)
0019            UX2=UBAPP(J,2)
0020            VX2=VBAPP(J,2)
0021            UX3=UBAPP(J,3)
0022            VX3=VBAPP(J,3)
0023            ITER=1
0024            CALL HORNER(NP,UA,VA,UX1,VX1,UB,VB,UPX1,VPX1)
0025            CALL HORNER(NP,UA,VA,UX2,VX2,UB,VB,UPX2,VPX2)
0026          20 CALL HORNER(NP,UA,VA,UX3,VX3,UB,VB,UPX3,VPX3)
0027            CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX
               14,VX4,UQ4,VQ4,UH3,VH3)
0028          30 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0029            IF(CONV) GO TO 50
0030            IF(ITER.LT.MAX) GO TO 40
0031            WRITE(IO2,1000) J,UROOT(J),VROOT(J),MAX
0032            WRITE(IO2,1010) UX4,VX4
0033            IF(J.LT.IROOT) GO TO 33
0034            IF(J.EQ.IROOT) GO TO 35
0035            GO TO 100
0036          33 KKK=IROOT-1
0037            DO 34 K=J,KKK
0038            URAPP(K,1)=URAPP(K+1,1)
0039            VRAPP(K,1)=VRAPP(K+1,1)
0040            URAPP(K,2)=URAPP(K+1,2)
0041            VRAPP(K,2)=VRAPP(K+1,2)
0042            URAPP(K,3)=URAPP(K+1,3)
0043          34 VRAPP(K,3)=VRAPP(K+1,3)
0044          35 IROOT=IROOT-1
0045            GO TO 100
```

TABLE H.III. (Continued)

```
0046           40 UX1=UX2
0047              VX1=VX2
0048              UX2=UX3
0049              VX2=VX3
0050              UX3=UX4
0051              VX3=VX4
0052              UPX1=UPX2
0053              VPX1=VPX2
0054              UPX2=UPX3
0055              VPX2=VPX3
0056              ITER=ITER+1
0057              GO TO 20
0058           50 L=L+1
0059              UROOTS(L)=UX4
0060              VROOTS(L)=VX4
0061          100 CONTINUE
0062              IF(L.EQ.0) GO TO 120
0063              DO 110 I=1,L
0064              UROOT(I)=UROOTS(I)
0065          110 VROOT(I)=VROOTS(I)
0066              NROOT=L
0067              RETURN
0068          120 NROOT=0
0069              RETURN
0070         1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(,I2,4H) = ,
                 1D23.16,3H + ,D23.16,2H I/24H DID NOT CONVERGE AFTER ,I3,11H ITERAT
                 2IONS)
0071         1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,D23.16,3H + ,D23.16,2H I/
                 1/)
0072              END
```

TABLE H.III (Continued)

```
0001            SUBROUTINE TEST(UX3,VX3,UX4,VX4,CONV)
              C ********************************************************************************
              C *                                                                              *
              C * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-            *
              C * IMATIONS BY TESTING THE EXPRESSION                                           *
              C * ABSOLUTE VALUE OF (X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).                   *
              C * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.                    *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UX3,VX3,UX4,VX4,EPSRT,EPSO,EPS,AAA,UDUMMY,VDUMMY,
               1DENOM
0003            DOUBLE PRECISION EPSM
0004            LOGICAL CONV
0005            COMMON EPSM,EPS,EPSO,EPSRT,IO2,MAX
0006            UDUMMY=UX4-UX3
0007            VDUMMY=VX4-VX3
0008            AAA=DSQRT(UDUMMY*UDUMMY+VDUMMY*VDUMMY)
0009            DENOM=DSQRT(UX4*UX4+VX4*VX4)
0010            IF(DENOM.LT.EPSO) GO TO 20
0011            IF(AAA/DENOM.LT.EPS) GO TO 10
0012          5 CONV=.FALSE.
0013            GO TO 100
0014         10 CONV=.TRUE.
0015            GO TO 100
0016         20 IF(AAA.LT.EPSO) GO TO 10
0017            GO TO 5
0018        100 RETURN
0019            END
```

```
0001            SUBROUTINE HORNER(NA,UA,VA,UX,VX,UB,VB,UPX,VPX)
              C ********************************************************************************
              C *                                                                              *
              C * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D.      *
              C * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE     *
              C * FACTOR (X-D).                                                               *
              C *                                                                              *
              C ********************************************************************************
0002            DOUBLE PRECISION UX,VX,UPX,VPX,UB,VB,UA,VA
0003            DIMENSION UA(26),VA(26),UB(26),VB(26)
0004            UB(1)=UA(1)
0005            VB(1)=VA(1)
0006            NUM=NA+1
0007            DO 10 I=2,NUM
0008            UB(I)=UA(I)+(UB(I-1)*UX-VB(I-1)*VX)
0009         10 VB(I)=VA(I)+(VB(I-1)*UX+UB(I-1)*VX)
0010            UPX=UB(NUM)
0011            VPX=VB(NUM)
0012            RETURN
0013            END
```