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 NCR 37-002-084

FINAL
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 NCR 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 greatest 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 NCR 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  

I. INTRODUCTION. .................................... 1

II. NEWTON'S METHOD .................................. 4
   Derivation of the Algorithm. .................... 4
   Convergence of Newton's Method ................. 8
   Procedure for Newton's Method ................ 9
   Geometrical Interpretation of Newton's Method.. 10
   Determining Multiple Roots .................... 10

III. MULLER'S METHOD ................................ 12
   Derivation of the Algorithm. .................. 12
   Procedure for Muller's Method ................. 19
   Geometrical Interpretation of Muller's Method.. 20
   Determining Multiple Roots .................... 20

IV. GREATEST COMMON DIVISOR METHOD ............... 22
   Derivation of the Algorithm. .................. 22
   Determining Multiplicities .................... 29
   Procedure for the G.C.D. Method .............. 29

V. REPEATED GREATEST COMMON DIVISOR .............. 30
   Derivation of the Algorithm. .................. 30
   Procedure for the Repeated G.C.D. Method ...... 32

VI. CONCLUSION. ..................................... 34
   Polynomials With all Distinct Roots ........... 34
   Polynomials With Multiple Roots ............... 35
   Time Comparisons .................................. 40
   Sample Results .................................... 42

REFERENCES ......................................... 85

APPENDIX A - SPECIAL FEATURES OF NEWTON'S AND MULLER'S
PROGRAMS. .......................................... 86
   Generating Approximations ...................... 86
   Altering Approximations ....................... 88
   Searching the Complex Plane .................... 89
Chapter

Improving Zeros Found ........................................ 91
Solving Quadratic Polynomial ................................... 91
Missing Roots ....................................................... 92
Miscellaneous ..................................................... 92

APPENDIX B - NEWTON'S METHOD ...................................... 100

Use of the Program ................................................ 100
Input Data for Newton's Method .................................. 102
Variables Used in Newton's Method ............................... 107
Description of Program Output .................................. 107
Informative and Error Message .................................. 109
Flow Charts ......................................................... 120
Program Listing .................................................... 126

APPENDIX C - MULLER'S METHOD ..................................... 135

Use of the Program ................................................ 135
Input Data for Muller's Method .................................. 137
Variables Used in Muller's Method ............................... 137
Description of Program Output .................................. 138
Informative and Error Messages ................................. 139
Flow Charts ......................................................... 144
Program Listing .................................................... 150

APPENDIX D - SPECIAL FEATURES OF THE G.C.D. AND THE
REPEATED G.C.D. PROGRAMS ....................................... 163

Generating Approximations ..................................... 163
Altering Approximations ......................................... 165
Searching the Complex Plane .................................... 166
Improving Zeros Found ........................................... 168
Solving Quadratic Polynomial ................................... 168
Missing Roots ....................................................... 169
Miscellaneous ..................................................... 169

APPENDIX E - G.C.D. - NEWTON'S METHOD .......................... 171

Use of the Program ................................................ 171
Input Data for G.C.D. - Newton's Method ...................... 173
Variables Used in G.C.D. - Newton's Method .................. 178
Description of Program Output .................................. 188
Informative and Error Messages ................................. 189
Flow Charts ......................................................... 191
Program Listing .................................................... 200
## APPENDIX F - G.C.D. - MULLER'S METHOD

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of the Program</td>
<td>214</td>
</tr>
<tr>
<td>Input Data for G.C.D. - Muller's Method</td>
<td>216</td>
</tr>
<tr>
<td>Variables Used in G.C.D. - Muller's Method</td>
<td>216</td>
</tr>
<tr>
<td>Description of Program Output</td>
<td>216</td>
</tr>
<tr>
<td>Informative and Error Messages</td>
<td>216</td>
</tr>
<tr>
<td>Flow Charts</td>
<td>222</td>
</tr>
<tr>
<td>Program Listing</td>
<td>234</td>
</tr>
</tbody>
</table>

## APPENDIX G - REPEATED G.C.D. - NEWTON'S METHOD

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of the Program</td>
<td>253</td>
</tr>
<tr>
<td>Input Data for Repeated G.C.D. - Newton's Method</td>
<td>255</td>
</tr>
<tr>
<td>Variables Used in Repeated G.C.D. - Newton's Method</td>
<td>255</td>
</tr>
<tr>
<td>Description of Program Output</td>
<td>255</td>
</tr>
<tr>
<td>Informative and Error Messages</td>
<td>256</td>
</tr>
<tr>
<td>Flow Charts</td>
<td>260</td>
</tr>
<tr>
<td>Program Listing</td>
<td>270</td>
</tr>
</tbody>
</table>

## APPENDIX H - REPEATED G.C.D. - MULLER'S METHOD

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of the Program</td>
<td>286</td>
</tr>
<tr>
<td>Input Data for Repeated G.C.D. - Muller's Method</td>
<td>287</td>
</tr>
<tr>
<td>Variables used in Repated G.C.D. - Muller's Method</td>
<td>288</td>
</tr>
<tr>
<td>Description of Program Output</td>
<td>288</td>
</tr>
<tr>
<td>Informative and Error Messages</td>
<td>289</td>
</tr>
<tr>
<td>Flow Charts</td>
<td>290</td>
</tr>
<tr>
<td>Program Listing</td>
<td>301</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

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

\[
P(x) = a_1 x^N + a_2 x^{N-1} + \ldots + a_N x + a_{N+1}
\]

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

\[
x_{n+1} = x_n - \frac{P(x_n)}{P'(x_n)}.
\]

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

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

(2-1)

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, \( a \), of \( P(X) \). Using only the first two terms of the expansion, the expression

\[ P(X) \approx P(X_0) + P'(X_0)(X - X_0) \]

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

\[ 0 \approx P(X_0) + P'(X_0)(X - X_0) \]

results. Rearranging terms produces
\[0 = P(X_0) + P'(X_0) X - P'(X_0) X_0\]

followed by

\[P'(X_0) X_0 - P(X_0) = P'(X_0) X\]

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

\[X_0 - \frac{P(X_0)}{P'(X_0)} = X\]

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

\[X_{n+1} = X_n - \frac{P(X_n)}{P'(X_n)}, \quad (2-2)\]

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

\[X_0, X_1, X_2, \ldots, X_n, \ldots \quad (2-3)\]

of successive approximations of the root, \(\alpha\). 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 \(\text{deg. } K(X) < \text{deg. } G(X)\). 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

\[
\begin{align*}
b_1 &= a_1 \\
b_i &= a_i + db_{i-1} & (i = 2, 3, \ldots, N+1).
\end{align*}
\]

Define another sequence \( c_1, c_2, \ldots, c_N \) by

\[
\begin{align*}
c_1 &= b_1 \\
c_j &= b_j + dc_{j-1} & (j = 2, 3, \ldots, N).
\end{align*}
\]

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, \( a \), 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)

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

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

Thus

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

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

After a zero, \( a \), of \( P(X) \) has been found, the term \((X - a)\) 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, \( \alpha_1 \), 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, \( \alpha_1 \) 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 \( \alpha_i \) \( (i = 0,1,2,\ldots,N-1) \) is a zero of \( P(X) \) since each \( \alpha_1 \) 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 \( \geq 0 \) or \( \leq 0 \) 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,

\[
e_{i+1} = \frac{1}{2} P''(\eta_i) e_i^2
\]

where \( F(X_i) = X_i - P(X_i)/P'(X_i) \), \( \eta_i \) is between \( X_i \) and the zero, \( a \), and \( e_i \) is the error in \( X_i \). This means that the error obtained in the \((i+1)^{\text{th}}\) iteration of Newton's algorithm is proportional to the square of the error obtained in the \(i^{\text{th}}\) 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

\[
|P(X_n)/P'(X_n)|/|X_{n+1}| < \varepsilon
\]

for some \( \varepsilon \) 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. $X_1$ is an approximation to the zero, $a$. $P'(X_1)$ is the slope of the line tangent to $P(X)$ at $X_1$. $X_{i+1}$ 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

$$P(X) = a_1(X - a_1)^{e_1}(X - a_2)^{e_2}... (X - a_m)^{e_m}, \ (m \leq N)$$

where $a_1$ is a zero of $P(X)$ and $e_1$ is the multiplicity of $a_1$ ($i = 1, 2, \ldots, m$). Consider the root $a_j$. Dividing out the term
(X - α_j) by deflating P(X) gives P_1(X) of degree \(N-1\) which can be written as

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

Evaluating \( P_1(X) \) at the zero, \( \alpha_j \), gives \( P_1(\alpha_j) = 0 \) if \( e_j > 1 \). Thus, after a zero, \( \alpha \), 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 \( P_1(\alpha) \leq \epsilon \) for some small number \( \epsilon \), \( \alpha \) 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 \( P_2(\alpha) \leq \epsilon \), \( \alpha \) is of multiplicity at least three. This process is continued until a deflated polynomial \( P_k(X) \) is encountered such that either \( \text{deg. } P_k(X) = 0 \) or \( P_k(\alpha) > \epsilon \). \( \alpha \) is then a zero of multiplicity \( k+1 \).

![Figure 2.1. Geometrical Interpretation of Newton's Method](image)
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

\[ P(X) = a_1 X^N + a_2 X^{N-1} + \ldots + a_N X + a_{N+1} \]  

(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, \( \alpha \), of \( P(X) \), the problem is to determine \( X_{n+1} \) in such a way as to generate a sequence

\[ X_1, X_2, X_3, \ldots, X_n, X_{n+1}, \ldots \]  

(3-2)

of approximations converging to \( \alpha \). 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 Lagrangian interpolating polynomial, the representation of \( Q(X) \) is given by
\[
Q(X) = \frac{(X - X_{n-1})(X - X_n)(X - X_{n-1})}{(X_n - X_{n-1})(X_n - X_{n-2})} P(X_n)
\]

\[
\begin{align*}
&+ \frac{(X - X_n)(X - X_{n-2})}{(X_{n-1} - X_n)(X_{n-2} - X_{n-1})} P(X_{n-1}) \\
&+ \frac{(X - X_n)(X - X_{n-1})}{(X_{n-2} - X_n)(X_{n-2} - X_{n-1})} P(X_{n-2})
\end{align*}
\]

which can be rewritten as

\[
Q(X) = Q(X - X_n + X_n)
\]

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

\[
\begin{align*}
&- \frac{(X - X_n)(X - X_n + X_n - X_{n-1} + X_{n-1} - X_{n-2})}{(X_n - X_{n-1})(X_{n-1} - X_{n-2})} P(X_{n-1}) \\
&+ \frac{(X - X_n)(X - X_n + X_n - X_{n-1})}{(X_{n-1} + X_{n-1} - X_{n-2})(X_{n-1} - X_{n-2})} P(X_{n-2}).
\end{align*}
\]

In order to simplify this expression, introduce the quantities

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

Then

\[
Q(X) = Q(X_n + h)
\]

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

\[
- \frac{h(h + h_n + h_{n-1})}{h_n h_{n-1}} P(X_{n-1})
\]
Collecting terms containing like powers of $h$ produces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\[
\frac{\left( \frac{h_n^2}{h_{n-1}^2} + h_n \right) P(X_n) - h_n \left[ \frac{h_n}{h_{n-1}} \right]^2 \right] + P(X_{n-1}) + \frac{h_n^3}{h_{n-1}^2} - P(X_{n-2}) h
\]
Let \( q_n = \frac{h_n}{h_{n-1}} \) and \( q = \frac{h}{h_n} \). Then

\[
Q(X_n + h) = \left( \frac{p(X_n) q_n - p(X_{n-1}) (q_n^2 + q_n) + p(X_{n-2}) q_n^2}{q_n+1} \right) q^2
\]

\[
+ \left( \frac{(2q_n+1) p(X_n) - (q_n+1)^2 p(X_{n-1}) + q_n^2 p(X_{n-2})}{q_n+1} \right) q
\]

\[
+ \frac{(q_n+1) p(X_n)}{q_n+1}.
\]

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:

\[
q_{n+1} = \frac{-B_n + \sqrt{B_n^2 - 4A_n C_n}}{2A_n}
\]

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:

\[
q_{n+1} = \frac{-B_n + \sqrt{B_n^2 - 4A_n C_n}}{2A_n}, \quad \frac{B_n + \sqrt{B_n^2 - 4A_n C_n}}{2A_n}
\]

\[
= \frac{-B_n^2 + B_n^2 - 4A_n C_n}{2A_n (B_n + \sqrt{B_n^2 - 4A_n C_n})}
\]

\[
q_{n+1} = \frac{-2C_n}{B_n + \sqrt{B_n^2 - 4A_n C_n}}.
\]

(3-3)

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, \( a_c \), of \( P(X) \) has been found, \( P(X) \) is deflated as described in Chapter II, § 1, and the process repeated to extract a zero, \( a_1 \), 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 \(\left|\frac{P(X_{n+1})}{P(X_n)}\right|\) 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
   \[X_{n+1} = X_n + h_n q_{n+1}^t.\]
3. Test for convergence; that is, test \(\left|\frac{X_{n+1} - X_n}{X_{n+1}}\right| < \epsilon\)
   for some suitably small number \(\epsilon\).
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 3.0 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

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

where $e_i$ is the multiplicity of $\alpha_i$, $i = 1, 2, \ldots, m$. 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
\[ p'(x) = a_1(x - a_1)^{e_1-1} (x - a_2)^{e_2-1} \cdots (x - a_m)^{e_m-1} \sum_{i=1}^{m} e_i \prod_{j=1, j \neq i}^{m} (x - a_j). \]  

(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 \( a_1, a_2, \ldots, a_m \) of multiplicity \( e_1, e_2, \ldots, e_m \) respectively. Then the polynomial

\[ D(X) = (x - a_1)^{e_1-1} (x - a_2)^{e_2-1} \cdots (x - a_m)^{e_m-1} \]

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 \( X - a_i \) 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., \( D(X) \).
\[ H(X) = \frac{P(X)}{D(X)} \]

\[ = a_1 \prod_{i=1}^{m} (X - a_i) e_i \prod_{i=1}^{m} (X - a_i) e_i^{-1} \]

\[ = a_1 \prod_{i=1}^{m} (X - a_i). \]

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 \( N_1 < N_0 \). 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 \( R_1(X) \) 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 \( N_k \)), results in polynomials \( R_{K-1}(X), S_K(X) \) and \( R_K(X) \) with \( \deg. R_K(X) \leq \deg. R_{K-1}(X) \) 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 \( R_1(X) \), the \( L(X) \) 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

\[
U_1(X) = \left( \frac{r_{i-1,1}}{r_{i,1}} \right)^{N_{i-1}-N_i} X^{N_{i-1}-N_i}.
\]

Then define \( T_1(X) \) by

\[
T_1(X) = R_{i-1}(X) - U_1(X) R_i(X)
\]

\[
= [r_{i-1,1} - r_{i,1} \left( \frac{r_{i-1,1}}{r_{i,1}} \right)^{N_{i-1}-N_i} X^{N_{i-1}-N_i} + [r_{i-1,2} - r_{i,2} \left( \frac{r_{i-1,1}}{r_{i,1}} \right)^{N_{i-1}-N_i} X^{N_{i-1}-N_i} + \ldots + [r_{i-1,N_{i-1}+1} - r_{i,N_{i-1}+1} \left( \frac{r_{i-1,1}}{r_{i,1}} \right) X^{N_{i-1}}} \]
\]

where \( r_{i,j} = 0 \) for \( j > N_i + 1 \).

We consider three cases.

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

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 \( \deg. T_1(X) = M_1 \geq N_1 \), then define \( U_2(X) \) by

\[
U_2(X) = \left( t_{1,1} \right)^{M_1-N_1} X^{M_1-N_1}
\]

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

\[
T_2(X) = [t_{1,1} - (t_{1,1} / r_{1,1}) r_{1,1}] X^{M_1-1}
\]

\[
+ [t_{1,2} - (t_{1,1} / r_{1,1}) r_{1,2}] X^{M_1-2}
\]

\[
+ \ldots
\]

\[
+ [t_{1,M_1+1} - (t_{1,1} / r_{1,1}) r_{1,M_1+1}]
\]

where \( r_{1,j} = 0 \) for \( j > N_1+1 \). 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 \( \deg. T_2(X) = M_2 > N_1 \), then define \( U_3(X) \) by

\[
U_3(X) = \left( t_{2,1} / r_{1,1} \right)^{M_2-N_1} X^{M_2-N_1}
\]
where
\[ T_2(X) = t_{2,1} X^2 + t_{2,2} X^{2-1} + \ldots + t_{2,M_2} X^{M_2} + t_{2,M_2+1}. \]

Since deg. \( T_{i+1}(X) < \text{deg. } T_i(X) \), 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. \( T_S(X) < \text{deg. } R_i(X) \), 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 \( H(X) = \frac{P(X)}{D(X)} \) 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
   
   $$P(X) = a_1 x^N + a_2 x^{N-1} + \ldots + a_N x + a_{N+1}.$$ 

2. Calculate the derivative, $P'(X)$, of $P(X)$ in the form
   
   $$P'(X) = b_1 x^{N-1} + b_2 x^{N-2} + \ldots + b_N$$ 
   
   where $b_1 = N a_1$, $b_2 = (N-1) a_2$, $\ldots$, $b_N = a_N$.

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

\[
P(X) = a_1 X^N + a_2 X^{N-1} + \ldots + a_N X + a_{N+1} = a_1 (X - \alpha_1)^{e_1} (X - \alpha_2)^{e_2} \ldots (X - \alpha_m)^{e_m}
\]

where \( a_1 \neq 0 \), each \( \alpha_i \) 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 $P(X) = a_1 X^N + a_2 X^{N-1} + \ldots + a_N X + a_{N+1}$ 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 $i > 0$, then

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

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

$$P(X) = a_1 X^N + a_2 X^{N-1} + \ldots + a_N X + a_{N+1}.$$ 

2. Set $D_0(X) = P(X)$.

3. Calculate the derivative, $D'_0(X)$, of $D_0(X)$ in the form

$$D'_0(X) = b_1 X^{M-1} + b_2 X^{M-2} + \ldots + b_M.$$
where deg. \( D_0(X) = M, \ D_0'(X) = d_1^M + \ldots + d_{M+1}, \)
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 15th 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, \( H(X) = P(X)/\text{g.c.d. } P(X) = P(X) \); 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.

<table>
<thead>
<tr>
<th>Polynomial #1</th>
<th>Polynomial #2</th>
<th>Polynomial #3</th>
<th>Polynomial #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2+2i (3)</td>
<td>-2.33 (1)</td>
<td>2+2i (3)</td>
<td>1+i (6)</td>
</tr>
<tr>
<td>1+2i (2)</td>
<td>.003 (2)</td>
<td>1+2i (2)</td>
<td>1-i (6)</td>
</tr>
<tr>
<td>-1+.5i (1)</td>
<td>i (2)</td>
<td>-1+.5i (3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.5i (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.5i (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3i (3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1-i (3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

\[ x_{n+1} = x_n - \frac{P(x_n)}{P'(x_n)}. \]

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 6th degree equation with multiple roots is greater than the time taken by the same program to solve a 15th 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 repeatedly 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 6.1 gives a comparison of the execution times of the six methods for polynomials #1 - #4.

TABLE 6.1

<table>
<thead>
<tr>
<th>METHOD</th>
<th>EXECUTION TIME*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton</td>
<td>104.16 seconds</td>
</tr>
<tr>
<td>Muller</td>
<td>96.79 seconds</td>
</tr>
<tr>
<td>G.C.D. with Newton</td>
<td>7.51 seconds</td>
</tr>
<tr>
<td>G.C.D. with Muller</td>
<td>8.91 seconds</td>
</tr>
<tr>
<td>Repeated G.C.D. with Newton</td>
<td>7.71 seconds</td>
</tr>
<tr>
<td>Repeated G.C.D. with Muller</td>
<td>15.16 seconds</td>
</tr>
</tbody>
</table>

It is clear from Table 6.1 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.
4. Repeated G.C.D. with Muller.
5. Muller.
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 7 OF DEGREE 15

THE COEFFICIENTS OF \( P(x) \) ARE

\[
P(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0
\]

where \( a_7 = 0.20000000000000 \) and \( a_0 = 0.00000000000000 \). The other coefficients are:

\[
a_1 = 0.17900000000000, \quad a_2 = -0.30000000000000, \quad a_3 = -0.30000000000000, \quad a_4 = -0.17500000000000, \quad a_5 = 0.28436000000000, \quad a_6 = 0.31190000000000
\]

\[
P(x) = 0.28763000000000x^6 - 0.30000000000000x^5 + 0.30000000000000x^4 - 0.17900000000000x^3 + 0.28436000000000x^2 + 0.31190000000000x^1 - 0.30000000000000x^0
\]

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.10000
TEST FOR MULTIPLEITIES: 0.10000
RADIUS TO START SEARCH: 0.00000
RADIUS TO END SEARCH: 0.00000

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

<table>
<thead>
<tr>
<th>ROOTS OF ( P(x) )</th>
<th>MULTIPLEITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ROOT1} )</td>
<td>1</td>
<td>0.160935811516655310 \times 10^{-16}</td>
</tr>
<tr>
<td>( \text{ROOT2} )</td>
<td>1</td>
<td>0.239629151665627990 \times 10^{-8}</td>
</tr>
<tr>
<td>( \text{ROOT3} )</td>
<td>1</td>
<td>0.707106781180095950 \times 10^{-8}</td>
</tr>
<tr>
<td>( \text{ROOT4} )</td>
<td>1</td>
<td>0.517638255196672401 \times 10^{-15}</td>
</tr>
<tr>
<td>( \text{ROOT5} )</td>
<td>1</td>
<td>0.388908630007225808 \times 10^{-15}</td>
</tr>
<tr>
<td>( \text{ROOT6} )</td>
<td>1</td>
<td>0.388228427467390400 \times 10^{-15}</td>
</tr>
<tr>
<td>( \text{ROOT7} )</td>
<td>1</td>
<td>0.388228427467390400 \times 10^{-15}</td>
</tr>
<tr>
<td>( \text{ROOT8} )</td>
<td>1</td>
<td>0.1385507847222020 \times 10^{-12}</td>
</tr>
<tr>
<td>( \text{ROOT9} )</td>
<td>1</td>
<td>0.1385507847222020 \times 10^{-12}</td>
</tr>
<tr>
<td>( \text{ROOT10} )</td>
<td>1</td>
<td>0.1385507847222020 \times 10^{-12}</td>
</tr>
<tr>
<td>( \text{ROOT11} )</td>
<td>1</td>
<td>0.1385507847222020 \times 10^{-12}</td>
</tr>
<tr>
<td>( \text{ROOT12} )</td>
<td>1</td>
<td>0.1385507847222020 \times 10^{-12}</td>
</tr>
<tr>
<td>( \text{ROOT13} )</td>
<td>1</td>
<td>0.1385507847222020 \times 10^{-12}</td>
</tr>
<tr>
<td>( \text{ROOT14} )</td>
<td>1</td>
<td>0.1385507847222020 \times 10^{-12}</td>
</tr>
<tr>
<td>( \text{ROOT15} )</td>
<td>1</td>
<td>0.1385507847222020 \times 10^{-12}</td>
</tr>
</tbody>
</table>

Exhibit 6.1.
After the attempt to improve accuracy, the zeros of \( p(x) \) are

<table>
<thead>
<tr>
<th>Zeros (roots)</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.3333333333333340 01</td>
<td>0.3333333333333340 01</td>
<td>0.482942911545623790 00 + 0.126409528435843870 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.3333333333333340 01</td>
<td>0.482942911545623790 00 + 0.126409528435843870 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.3000000000000000 00</td>
<td>0.3968039820811357 00 + 0.126409528435843870 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
<tr>
<td>-0.2999999999999996 00</td>
<td>0.4000000000000000 00</td>
<td>0.4000000000000000 00 + 0.4000000000000000 00 1</td>
</tr>
</tbody>
</table>

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.
For finding the zeros of a polynomial, Muller's method is used. The coefficients of the polynomial are given.

The polynomial is of degree 15.

The initial radius is 0.300000000000000000.

The maximum number of initial approximations is given as 0.

The test for convergence is 100-09.

The test for multiplicities is 100-01.

The radius to start search is 0.000000000000000000.

The radius to end search is 0.000000000000000000.

Before attempting to improve accuracy, the roots of the polynomial and their multiplicities are shown.

Exhibit 6.2.
AFTER THE ATTEMPT TO IMPROVE ACCURACY

<table>
<thead>
<tr>
<th>ROOTS OF P(x)</th>
<th>MULTIPlicITIES</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT 11 = 0.30000000000000000000</td>
<td>0.337673962033440-16</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 21 = 0.20000000000000000000</td>
<td>0.337673962033440-16</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 31 = 0.49999999999999999999</td>
<td>0.337673962033440-16</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 41 = 0.1537567615825440+15</td>
<td>0.10000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 51 = -0.10000000000000000000</td>
<td>0.10000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 61 = -0.33333333333333333333</td>
<td>0.10000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 71 = -0.10000000000000000000</td>
<td>0.10000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 81 = 0.527039381772651-14</td>
<td>0.10000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 91 = 0.20000000000000000000</td>
<td>0.20000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 101 = 0.20000000000000000000</td>
<td>0.20000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 111 = 0.20000000000000000000</td>
<td>0.20000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 121 = 0.20000000000000000000</td>
<td>0.20000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 131 = 0.20000000000000000000</td>
<td>0.20000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 141 = 0.60000000000000000000</td>
<td>0.60000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>ROOT 151 = -0.20000000000000000000</td>
<td>0.20000000000000000000</td>
<td>0.00000000000000000000</td>
</tr>
</tbody>
</table>

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.
The coefficients of $p(x)$ are

$P_1 = 0.10000000000000000 + 0.00000000000000000 I$

$P_2 = -0.70000000000000000 - 0.10500000000000000 I$

$P_3 = -0.28000000000000000 + 0.58000000000000000 I$

$P_4 = 0.17100000000000000 + 0.15000000000000000 I$

$P_5 = -0.73000000000000000 - 0.25100000000000000 I$

$P_6 = -0.22800000000000000 + 0.10400000000000000 I$

$P_7 = 0.72000000000000000 + 0.10400000000000000 I$

The number of initial approximations given is 0.

Maximum number of iterations: 200

Test for convergence: 0.10-09

Test for multiplicities: 0.10-01

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 $p(x)$

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>2</td>
<td>0.99999998831250190 00 + 0.20000000000000000 I</td>
</tr>
<tr>
<td>R2</td>
<td>1</td>
<td>0.99999998831250190 00 + 1.00000000000000000 I</td>
</tr>
<tr>
<td>R3</td>
<td>1</td>
<td>-0.99999999999999999 00 + 0.00000000000000000 I</td>
</tr>
</tbody>
</table>

In the attempt to improve accuracy, root R2 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 $p(x)$

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>2</td>
<td>0.99999999999999999 00 + 0.00000000000000000 I</td>
</tr>
<tr>
<td>R2</td>
<td>1</td>
<td>1.00000000000000000 + 0.00000000000000000 I</td>
</tr>
<tr>
<td>R3</td>
<td>1</td>
<td>-1.00000000000000000 + 0.50000000000000000 I</td>
</tr>
</tbody>
</table>

Exhibit 6.3. Roots Are: 2+2i (3), 1+2i (2), -1+5i
Newton's Method to Find Zeros of Polynomials

Polynomial Number 2 of Degree 15

The coefficients of P(x) are

\[ P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{15}x^{15} \]

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_0</td>
<td>0.4800000000000000D02</td>
</tr>
<tr>
<td>a_1</td>
<td>-0.3840000000000000D03</td>
</tr>
<tr>
<td>a_2</td>
<td>0.2571200000000000D02</td>
</tr>
<tr>
<td>a_3</td>
<td>-0.2188460000000000D04</td>
</tr>
<tr>
<td>a_4</td>
<td>0.1356560000000000D04</td>
</tr>
<tr>
<td>a_5</td>
<td>-0.1402620000000000D05</td>
</tr>
<tr>
<td>a_6</td>
<td>0.1769870000000000D06</td>
</tr>
<tr>
<td>a_7</td>
<td>-0.4036640000000000D05</td>
</tr>
<tr>
<td>a_8</td>
<td>0.4137370000000000D06</td>
</tr>
<tr>
<td>a_9</td>
<td>-0.1871260000000000D06</td>
</tr>
<tr>
<td>a_10</td>
<td>0.2171340000000000D06</td>
</tr>
<tr>
<td>a_11</td>
<td>-0.2814690000000000D05</td>
</tr>
<tr>
<td>a_12</td>
<td>0.1329430000000000D05</td>
</tr>
<tr>
<td>a_13</td>
<td>-0.3053900000000000D05</td>
</tr>
<tr>
<td>a_14</td>
<td>0.2756620000000000D00</td>
</tr>
<tr>
<td>a_15</td>
<td>-0.1835890000000000D03</td>
</tr>
</tbody>
</table>

Number of initial approximations given: 0
Maximum number of iterations: 200
Test for convergence: 0.10D-05
Test for multiplicities: 0.100-01
Radius to start search: 0.000 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 = 0.3000000736398184D-02 + 0.2609394906552981D-08 0 0.1294652861381870 00 1
Root 2 = 0.8626623124099139D-04 + 0.1000080173722988D01 0 0.7071067530463460 00 1
Root 3 = 0.5768342464927569D-02 + 0.1508145174316570D01 1 0.3882284792654056D00 0 1
Root 4 = -0.5926634471058762D-32 + 0.1491722999922961D01 1 -0.5176382551966724D00 0 1
Root 5 = -0.1014904754401601D01 + 0.1028420931384370D01 1 -0.1767766788520150 00 1
Root 6 = -0.2333333617834863D01 + 0.2168626449049502D00 0 0.7071068070684595D00 0 1
Root 7 = -0.2333333617834863D01 + 0.2168626449049502D00 0 0.7071068070684595D00 0 1
Root 8 = -0.1014904754401601D01 + 0.1028420931384370D01 1 -0.1767766788520150 00 1
Root 9 = -0.2414652861381870 00 + 0.1294652861381870 00 1 -0.4629628831455027D00 0 1
Root 10 = 0.3000000736398184D-02 + 0.2609394906552981D-08 0 0.1294652861381870 00 1
Root 11 = 0.3000000736398184D-02 + 0.2609394906552981D-08 0 0.1294652861381870 00 1
Root 12 = 0.3000000736398184D-02 + 0.2609394906552981D-08 0 0.1294652861381870 00 1

After the attempt to improve accuracy, the zeros of P(x) are

Solved by Direct Method

Exhibit 6.4.
<table>
<thead>
<tr>
<th>ROOT OF P(X)</th>
<th>MULTIPlicITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(1) = 0.300000000000000000E+00</td>
<td>0</td>
<td>0.482964711564677000 + 0.129409529843816700</td>
</tr>
<tr>
<td>ROOT(2) = 0.646472510627489000E-02</td>
<td>0</td>
<td>0.707106781186547000 + 0.707106781186547000</td>
</tr>
<tr>
<td>ROOT(3) = 0.314159265358978000E-03</td>
<td>0</td>
<td>0.482962911565627900 + 0.129409528443818700</td>
</tr>
<tr>
<td>ROOT(4) = 0.764245775342978000E-03</td>
<td>0</td>
<td>0.149999101165093000 + 0.129409528443818700</td>
</tr>
<tr>
<td>ROOT(5) = 0.999999001101101000E-03</td>
<td>0</td>
<td>0.149999101165093000 + 0.129409528443818700</td>
</tr>
<tr>
<td>ROOT(6) = 0.261959910308546000E-04</td>
<td>0</td>
<td>0.999995174015841000 + 0.999995026367763000</td>
</tr>
<tr>
<td>ROOT(7) = 0.261959910308546000E-04</td>
<td>0</td>
<td>0.999995174015841000 + 0.999995026367763000</td>
</tr>
<tr>
<td>ROOT(8) = 0.261959910308546000E-04</td>
<td>0</td>
<td>0.999995174015841000 + 0.999995026367763000</td>
</tr>
<tr>
<td>ROOT(9) = 0.261959910308546000E-04</td>
<td>0</td>
<td>0.999995174015841000 + 0.999995026367763000</td>
</tr>
<tr>
<td>ROOT(10) = 0.261959910308546000E-04</td>
<td>0</td>
<td>0.999995174015841000 + 0.999995026367763000</td>
</tr>
<tr>
<td>ROOT(11) = 0.261959910308546000E-04</td>
<td>0</td>
<td>0.999995174015841000 + 0.999995026367763000</td>
</tr>
<tr>
<td>ROOT(12) = 0.261959910308546000E-04</td>
<td>0</td>
<td>0.999995174015841000 + 0.999995026367763000</td>
</tr>
</tbody>
</table>

Exhibit 6.4, Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (3), 3i (3), -1-i (3)
**NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS**

**POLYNOMIAL NUMBER 3 OF DEGREE 8**

The coefficients of $p(x)$ are:

<table>
<thead>
<tr>
<th>Degree</th>
<th>Coefficient</th>
<th>Degree</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$0.10000000000000000$</td>
<td>1</td>
<td>$-0.30000000000000000$</td>
</tr>
<tr>
<td>2</td>
<td>$-0.51750000000000000$</td>
<td>3</td>
<td>$-0.34750000000000000$</td>
</tr>
<tr>
<td>4</td>
<td>$0.19725000000000000$</td>
<td>5</td>
<td>$0.15725000000000000$</td>
</tr>
<tr>
<td>6</td>
<td>$-0.49525000000000000$</td>
<td>7</td>
<td>$-0.58575000000000000$</td>
</tr>
<tr>
<td>8</td>
<td>$0.18100000000000000$</td>
<td>9</td>
<td>$0.15800000000000000$</td>
</tr>
</tbody>
</table>

Number of Initial Approximations Given: 3

Maximum Number of Iterations: 200

Test for Convergence: 0.100-05

Test for Multiplicities: 0.10D-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:

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicity</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$0.99999962393979710$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$-0.99999627647975770$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$0.19999950599858870$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>$-0.10000053739815870$</td>
</tr>
</tbody>
</table>

After the attempt to improve accuracy, the zeros of $p(x)$ are:

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicity</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$0.99999962393979710$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$-0.99999627647975770$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$0.19999950599858870$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>$-0.10000053739815870$</td>
</tr>
</tbody>
</table>

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

\[ P(1) = 0.100000000000000000 01 + 0.000000000000000000 01 \]
\[ P(2) = -0.120000000000000000 02 + 0.000000000000000000 02 \]
\[ P(3) = 0.700000000000000000 03 + 0.000000000000000000 03 \]
\[ P(4) = -0.280000000000000000 04 + 0.000000000000000000 04 \]
\[ P(5) = 0.720000000000000000 05 + 0.000000000000000000 05 \]
\[ P(6) = -0.280000000000000000 06 + 0.000000000000000000 06 \]
\[ P(7) = 0.264000000000000000 07 + 0.000000000000000000 07 \]
\[ P(8) = -0.326400000000000000 08 + 0.000000000000000000 08 \]
\[ P(9) = 0.312000000000000000 09 + 0.000000000000000000 09 \]
\[ P(10) = -0.224000000000000000 10 + 0.000000000000000000 10 \]
\[ P(11) = 0.115200000000000000 11 + 0.000000000000000000 11 \]
\[ P(12) = -0.384000000000000000 12 + 0.000000000000000000 12 \]

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.100-03
TEST FOR MULTIPLEITIES: 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

ROOT(1) = 0.999407980336915200 00 + 0.9951391489643941D 00
ROOT(2) = 0.102009688560972001
ROOT(3) = 0.1002721106681658D 01 + 0.10233106615266340D 01
ROOT(4) = 0.61899689593614000
ROOT(5) = 0.985142204742341400
ROOT(6) = 0.6515828215165421D 00 + 0.1206987666454936D 01
ROOT(7) = 0.1393799984400508D 01
ROOT(8) = 0.1330620199428514D 01

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(1) = 0.102009688560972001 + 0.10233106615266340D 01 DID NOT CONVERGE.

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

Exhibit 6.6.
Exhibit 6.6. Roots Are: 1+i(6), 1-i(6)
MULLER'S METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 1 OF DEGREE 6

THE COEFFICIENTS OF $P(X)$ ARE

$P(1) = 0.1000000000000000D+01 + 0.0000000000000000D+00 I$
$P(2) = -0.7000000000000001D+01 + 0.1050000000000000D+00 I$
$P(3) = -0.2800000000000000D+02 + 0.5800000000000000D+00 I$
$P(4) = 0.1710000000000000D+02 + 0.1900000000000000D+00 I$
$P(5) = -0.7100000000000000D+02 + 0.2510000000000000D+00 I$
$P(6) = -0.2290000000000000D+03 + 0.1040000000000000D+00 I$
$P(7) = 0.1200000000000000D+03 + 0.1040000000000000D+00 I$

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 $P(X)$

ROOT(1) = 0.19999548354197870 01 + 0.2000017627898503D+00 3 0.482629156552790 00 + 0.12940952844381870 00 1
ROOT(2) = 0.10000001597792780 01 + 0.19999999796000450 01 1 0.7071066755304640 00 + 0.70710680706845950 00 1
ROOT(3) = -0.10000002166130900 01 + 0.482629156552790 00 1 SOLVED BY DIRECT METHOD

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF $P(X)$

ROOT(1) = 0.19999548354197870 01 + 0.2000017627898503D+00 3 0.482629156552790 00 + 0.12940952844381870 00 1
ROOT(2) = 0.10000001597792780 01 + 0.19999999796000450 01 1 0.7071066755304640 00 + 0.70710680706845950 00 1
ROOT(3) = -0.10000002166130900 01 + 0.482629156552790 00 1 SOLVED BY DIRECT METHOD

Exhibit 6.7. Roots Are: 2 + 2i (3), 1 + 2i (2), -1 + .5i
MULLER'S METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL

THE COEFFICIENTS OF P(X) ARE

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1274979749590000</td>
<td>0.1733306464000000</td>
<td>-0.1356429220000000</td>
<td>-0.2189649000000000</td>
<td>-0.3855746900000000</td>
<td>-0.1755175600000000</td>
<td>0.1022795221300000</td>
<td>0.10464244232200000</td>
<td>-0.20636628844200000</td>
<td>-0.18712957801000000</td>
<td>-0.12713412742000000</td>
<td>0.19284697729000000</td>
<td>0.1329454584000000</td>
<td>0.30515988740000000</td>
</tr>
</tbody>
</table>

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.100-09
TEST FOR MULTIPLEITIES: 0.100-01
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X) | MULTIPlicITIES | INITIAL APPROXIMATION
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(1)</td>
<td>0.3000000562553720-02</td>
<td>0.8277419327311153500</td>
</tr>
<tr>
<td>ROOT(2)</td>
<td>0.1664927461057680-04</td>
<td>0.100000193970490090 01</td>
</tr>
<tr>
<td>ROOT(3)</td>
<td>0.1160277316189690-05</td>
<td>0.14499947129639100 01</td>
</tr>
<tr>
<td>ROOT(4)</td>
<td>0.6016878122456100 01</td>
<td>0.29023656568000000 01</td>
</tr>
<tr>
<td>ROOT(5)</td>
<td>0.1666459399710450 01</td>
<td>0.99999993577996000 01</td>
</tr>
<tr>
<td>ROOT(6)</td>
<td>0.2333333333380140 02</td>
<td>0.3248405453532900 01</td>
</tr>
<tr>
<td>ROOT(7)</td>
<td>0.1000001477939140 01</td>
<td>0.482962883145302700 01</td>
</tr>
<tr>
<td>ROOT(8)</td>
<td>0.7071067753904900 01</td>
<td>0.217134127420000000</td>
</tr>
<tr>
<td>ROOT(9)</td>
<td>0.13294545840000000 01</td>
<td>0.104642442322000000</td>
</tr>
<tr>
<td>ROOT(10)</td>
<td>0.13294545840000000 01</td>
<td>0.104642442322000000</td>
</tr>
<tr>
<td>ROOT(11)</td>
<td>0.13294545840000000 01</td>
<td>0.104642442322000000</td>
</tr>
<tr>
<td>ROOT(12)</td>
<td>0.13294545840000000 01</td>
<td>0.104642442322000000</td>
</tr>
<tr>
<td>ROOT(13)</td>
<td>0.13294545840000000 01</td>
<td>0.104642442322000000</td>
</tr>
</tbody>
</table>

AFTER THE ATTEMPT TO IMPROVE ACCURACY

Exhibit 6.8.
Exhibit 6.8. Roots Are: -2.33, .003 (2), i(2), 1.5i (2), -1.5i (2) 3i (3), -1-i(3)
MULLER'S METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 3 OF DEGREE 8

THE COEFFICIENTS OF $p(x)$ ARE

$p_1 = 0.1000000000000000D \quad 01$
$p_2 = -0.5175000000000001 \quad 02$
$p_3 = 0.1572500000000000 \quad 03$
$p_4 = -0.9452500000000000 \quad 03$
$p_5 = -0.2827200000000000 \quad 03$
$p_6 = 0.1810000000000000 \quad 03$
$p_7 = -0.6800000000000001 \quad 01$
$p_8 = 0.3075000000000000 \quad 03$
$p_9 = -0.4952500000000000 \quad 03$
$p_{10} = -0.4948750000000000 \quad 03$
$p_{11} = -0.5857500000000001 \quad 03$
$p_{12} = 0.4247500000000001 \quad 03$

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.100D-09
TEST FOR MULTIPLICITIES: 0.100D-01
RADIUS TO START SEARCH: 0.000
RADIUS TO END SEARCH: 0.000

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF $p(x)$

MULTIPLICITIES

INITIAL APPROXIMATION

ROOT(1) = 0.2000042202873018 \quad 01
ROOT(2) = 0.1000563806909880 \quad 01
ROOT(3) = -0.1063528418749844 \quad 01
ROOT(4) = -0.9722128516869824 \quad 00
ROOT(5) = 0.9994370680506410 \quad 00
ROOT(6) = -0.9643868131418565 \quad 00

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF $p(x)$

MULTIPLICITIES

INITIAL APPROXIMATION

ROOT(1) = 0.2000042202873018 \quad 01
ROOT(2) = 0.1000563806909880 \quad 01
ROOT(3) = -6.1000006594122314 \quad 01
ROOT(4) = -0.1000000972571865 \quad 01
ROOT(5) = 0.9999999168409400 \quad 00
ROOT(6) = -0.9999925738129557 \quad 00

Exhibit 6.9. Roots Are: 2+2i (3), 1+2i (2), -1.5i (3)
MULLER'S METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 4 OF DEGREE 12

THE COEFFICIENTS OF P(X) ARE

\[
P(1) = 0.1000000000000000D01 + 0.0000000000000000D00 01
P(2) = -0.12000000000000000 02 + 0.0000000000000000D00 02
P(3) = 0.32000000000000000 03 - 0.0000000000000000D00 03
P(4) = 0.2000000000000000D03 + 0.0000000000000000D00 03
P(5) = 0.7800000000000000D04 + 0.0000000000000000D00 04
P(6) = -0.1420000000000000D05 + 0.0000000000000000D00 05
P(7) = 0.2620000000000000D06 + 0.0000000000000000D00 06
P(8) = -0.3120000000000000D07 + 0.0000000000000000D00 07
P(9) = 0.6400000000000001D02 + 0.0000000000000000D00 08
P(10) = -0.2800000000000000D03 + 0.0000000000000000D00 09
P(11) = -0.3264000000000000D04 + 0.0000000000000000D00 10
P(12) = 0.3120000000000000D04 + 0.0000000000000000D00 11
P(13) = 0.7800000000000000D04 + 0.0000000000000000D00 12

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTILITIES. 0.10D-01
RADIUS TO START SEARCH. 0.00D00
RADIUS TO END SEARCH. 0.00D00
BEFORE ATTEMPT TO IMPROVE ACCURACY

<table>
<thead>
<tr>
<th>ROOT(1)</th>
<th>MULTIPlicities</th>
<th>INITIAL APPROXIMAtION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1000446912996739D01</td>
<td>1</td>
<td>0.1002199177325175D01</td>
</tr>
<tr>
<td>0.1007069490520364D01</td>
<td>1</td>
<td>0.1007069490520364D01</td>
</tr>
<tr>
<td>0.7584518107423498D00</td>
<td>1</td>
<td>0.7584518107423498D00</td>
</tr>
<tr>
<td>0.8439013544164170D00</td>
<td>1</td>
<td>0.8439013544164170D00</td>
</tr>
<tr>
<td>0.1245479586151551D01</td>
<td>1</td>
<td>0.1245479586151551D01</td>
</tr>
<tr>
<td>0.1138034314281570D03</td>
<td>1</td>
<td>0.1138034314281570D03</td>
</tr>
</tbody>
</table>

AFTER THE ATTEMPT TO IMPROVE ACCURACY

<table>
<thead>
<tr>
<th>ROOT(1)</th>
<th>MULTIPlicities</th>
<th>INITIAL APPROXIMAtION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1000446912996739D01</td>
<td>1</td>
<td>0.1002199177325175D01</td>
</tr>
<tr>
<td>0.1007069490520364D01</td>
<td>1</td>
<td>0.1007069490520364D01</td>
</tr>
<tr>
<td>0.7584518107423498D00</td>
<td>1</td>
<td>0.7584518107423498D00</td>
</tr>
<tr>
<td>0.8439013544164170D00</td>
<td>1</td>
<td>0.8439013544164170D00</td>
</tr>
<tr>
<td>0.1245479586151551D01</td>
<td>1</td>
<td>0.1245479586151551D01</td>
</tr>
<tr>
<td>0.1138034314281570D03</td>
<td>1</td>
<td>0.1138034314281570D03</td>
</tr>
</tbody>
</table>

Exhibit 6.10.
Exhibit 6.10. Roots Are: 1+i (6), 1-i (6)
GREATER 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 MULTIPLOSITIES: 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) = -0.1000000000000000D 01 + 0.0000000000000000D 00 I
P(6) = -0.7000000000000000D 01 + 0.1050000000000000 I
P(5) = -0.2800000000000000D 02 + 0.5800000000000001 I
P(4) = 0.1710000000000000D 03 + 0.15000000000000000 I
P(3) = -0.7300000000000000D 02 + 0.25100000000000000 I
P(2) = -0.2280000000000000D 03 + 0.10400000000000000 I
P(1) = 0.7200000000000000D 02 + 0.10400000000000000 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) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
Q(3) = -0.2000000000000000D 01 + 0.4500000000000000 I
Q(2) = -0.7000000000000000D 01 + 0.3500000000000060 I
Q(1) = 0.9999999999999999D 00 + 0.6999999999999999D 01 I

ROOTS OF Q(X)

INITIAL APPROXIMATION

ROOT(1) = 0.9999999999999999D 00 + 0.1999999999999999D 01 I
ROOT(2) = 0.2000000000000000D 01 + 0.2000000000000000D 01 I
ROOT(3) = -0.9999999999999999D 00 + 0.5000000000000000D 01 I

RESULTS OF SUBROUTINE QUAD

ROOT(1) = 0.4829629315656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.2000000000000000D 01 + 0.2000000000000000D 01 I
ROOT(3) = -0.9999999999999999D 00 + 0.5000000000000000D 01 I

ROOTS OF P(X)

INITIAL APPROXIMATION

ROOT(1) = 0.9999999999999999D 00 + 0.1999999999999999D 01 I
ROOT(2) = 0.2000000000000000D 01 + 0.2000000000000000D 01 I
ROOT(3) = -0.9999999999999999D 00 + 0.5000000000000000D 01 I

RESULTS OF SUBROUTINE QUAD

ROOT(1) = 0.4829629315656279D 00 + 0.1294095284438187D 00 I
ROOT(2) = 0.2000000000000000D 01 + 0.2000000000000000D 01 I
ROOT(3) = -0.9999999999999999D 00 + 0.5000000000000000D 01 I

EXHIBIT 6.1: Roots Are: 2+2i (3), 1+2i (2), -1.5i
Greatest Common Divisor Method Used with Newton's Method to Find Zeros of Polynomials

<table>
<thead>
<tr>
<th>Polynomial Number 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GREATEST COMMON DIVISOR</strong></td>
</tr>
<tr>
<td><strong>METHOD USED WITH NEWTON'S METHOD</strong> TO FIND ZEROS OF POLYNOMIALS</td>
</tr>
<tr>
<td><strong>NUMBER OF INITIAL APPROXIMATIONS GIVEN.</strong></td>
</tr>
<tr>
<td><strong>NUMBER OF INITIAL APPROXIMATIONS GIVEN.</strong></td>
</tr>
<tr>
<td><strong>TEST FOR ZERO IN SUBROUTINE GCD.</strong></td>
</tr>
<tr>
<td><strong>TEST FOR CONVERGENCE.</strong></td>
</tr>
<tr>
<td><strong>TEST FOR ZERO IN SUBROUTINE QUAD.</strong></td>
</tr>
<tr>
<td><strong>TEST FOR MULTIPLECTICIES.</strong></td>
</tr>
<tr>
<td><strong>RADIUS TO START SEARCH.</strong></td>
</tr>
<tr>
<td><strong>RADIUS TO END SEARCH.</strong></td>
</tr>
<tr>
<td><strong>THE DEGREE OF P(1) IS</strong></td>
</tr>
<tr>
<td><strong>THE COEFFICIENTS ARE</strong></td>
</tr>
<tr>
<td>P(11) = 0.48000000000000000 D-02 + 0.00000000000000000 D+1 0 1</td>
</tr>
<tr>
<td>P(12) = 0.21797200000000000 D-01 + 0.21797200000000000 D-01 0 1</td>
</tr>
<tr>
<td>P(13) = -0.38979556900000000 D-01 + 0.21797200000000000 D-01 0 1</td>
</tr>
<tr>
<td>P(14) = 0.13206020000000000 D-01 + 0.21797200000000000 D-01 0 1</td>
</tr>
<tr>
<td>P(15) = 0.10000000000000000 D+00 + 0.21797200000000000 D-01 0 1</td>
</tr>
<tr>
<td>P(16) = 0.20000000000000000 D+00 + 0.21797200000000000 D-01 0 1</td>
</tr>
<tr>
<td><strong>THE DEGREE OF Q(X) IS</strong></td>
</tr>
<tr>
<td><strong>THE COEFFICIENTS ARE</strong></td>
</tr>
<tr>
<td>Q(1) = 0.48000000000000000 D-02 + 0.00000000000000000 D+1 0 1</td>
</tr>
<tr>
<td>Q(2) = 0.13206020000000000 D-01 + 0.21797200000000000 D-01 0 1</td>
</tr>
<tr>
<td>Q(3) = 0.10000000000000000 D+00 + 0.21797200000000000 D-01 0 1</td>
</tr>
<tr>
<td>Q(4) = 0.20000000000000000 D+00 + 0.21797200000000000 D-01 0 1</td>
</tr>
<tr>
<td><strong>ROOTS OF Q(X) ARE</strong></td>
</tr>
<tr>
<td>ROOT 1 = 0.48000000000000000 D+00 + 0.00000000000000000 D+1 0 1</td>
</tr>
<tr>
<td>ROOT 2 = 0.13206020000000000 D-01 + 0.21797200000000000 D-01 0 1</td>
</tr>
<tr>
<td><strong>ROOTS OF P(X) ARE</strong></td>
</tr>
<tr>
<td>ROOT 1 = 0.48000000000000000 D+00 + 0.00000000000000000 D+1 0 1</td>
</tr>
<tr>
<td>ROOT 2 = 0.13206020000000000 D-01 + 0.21797200000000000 D-01 0 1</td>
</tr>
</tbody>
</table>

**Exhibit 6.12.**
### Exhibit 6.12. Roots Are:

-2.33, 0.003 (2), i(2), 1.5i (2),
-1.5i (2) 3i (3), -1-i (3)
GREATEST COMMON DIVISOR METHOD USED WITH NEWTON'S 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 MULTIPLEITIES: 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

\[
P(x) = 0.1000000000000001 - 0.0000000000000000 00 01
\]
\[
P(x) = -0.3475000000000001 - 0.4247500000000000 00 00
\]
\[
P(x) = 0.3075000000000000 - 0.4948750000000000 03 00
\]
\[
P(x) = -0.1575000000000000 - 0.1584250000000000 03 01
\]
\[
P(x) = 0.1810000000000000 + 0.1446250000000000 03 03
\]
\[
P(x) = 0.3075000000000000 + 0.3475000000000000 03 03
\]
\[
P(x) = -0.4948750000000000 + 0.4247500000000000 03 03
\]
\[
P(x) = 0.1584250000000000 + 0.1575000000000000 03 03
\]
\[
P(x) = 0.4948750000000000 + 0.3475000000000000 03 03
\]

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

\[
Q(x) = 0.1000000000000000 - 0.0000000000000000 00 01
\]
\[
Q(x) = -0.3475000000000001 - 0.4247500000000000 00 00
\]
\[
Q(x) = 0.3075000000000000 - 0.4948750000000000 03 00
\]
\[
Q(x) = -0.1575000000000000 - 0.1584250000000000 03 01
\]
\[
Q(x) = 0.1810000000000000 + 0.1446250000000000 03 03
\]
\[
Q(x) = 0.3075000000000000 + 0.3475000000000000 03 03
\]
\[
Q(x) = -0.4948750000000000 + 0.4247500000000000 03 03
\]
\[
Q(x) = 0.1584250000000000 + 0.1575000000000000 03 03
\]
\[
Q(x) = 0.4948750000000000 + 0.3475000000000000 03 03
\]

ROOTS OF Q(X)

ROOT(1) = 0.9999999999995483 + 0.19999999999997550
ROOT(2) = 0.20000000000005390 + 0.20000000000005390
ROOT(3) = 0.9999999999998140 + 0.7000000000000470 1

ROOT(1) = 0.9999999999995483 - 0.19999999999997550
ROOT(2) = 0.20000000000005390 - 0.20000000000005390
ROOT(3) = 0.9999999999998140 - 0.7000000000000470

RESULTS OF SUBROUTINE QUAD

EXHIBIT 6.13. ROOTS ARE: 2+2i (3), 1+2i (2), -1.5i (3)
Greatest common divisor method used with Newton's 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: 0.100-02
Test for convergence: 0.100-09
Test for zero in subroutine QUAD: 0.100-12
Radius to start search: 0.000 00
Radius to end search: 0.000 00

The degree of \( p(x) \) is 12 the coefficients are

\[
\begin{align*}
p_{13} &= 0.1000000000000000D 01 + 0.0000000000000000D 00 I \\
p_{12} &= 0.1200000000000000D 02 + 0.0000000000000000D 00 I \\
p_{11} &= 0.7800000000000000D 02 + 0.0000000000000000D 00 I \\
p_{10} &= 0.1800000000000000D 02 + 0.0000000000000000D 00 I \\
p_9 &= 0.1630000000000000D 04 + 0.0000000000000000D 00 I \\
p_8 &= 0.2670000000000000D 04 + 0.0000000000000000D 00 I \\
p_7 &= 0.6340000000000000D 04 + 0.0000000000000000D 00 I \\
p_6 &= 0.3240000000000000D 04 + 0.0000000000000000D 00 I \\
p_5 &= 0.2320000000000000D 04 + 0.0000000000000000D 00 I \\
p_4 &= 0.1220000000000000D 04 + 0.0000000000000000D 00 I \\
p_3 &= 0.1120000000000000D 04 + 0.0000000000000000D 00 I \\
p_2 &= 0.3340000000000000D 03 + 0.0000000000000000D 00 I \\
p_1 &= 0.6400000000000000D 02 + 0.0000000000000000D 00 I \\
p_0 &= 0.6400000000000000D 02 + 0.0000000000000000D 00 0I
\end{align*}
\]

\( q(x) \) is the polynomial which has as its roots the distinct roots of \( p(x) \).

The degree of \( q(x) \) is 2 the coefficients are

\[
\begin{align*}
q_{12} &= 0.1000000000000000D 01 + 0.0000000000000000D 00 0I \\
q_{11} &= 0.09999999999999074D 00 I \\
q_{10} &= 0.1000000000000000D 01 + 0.0000000000000000D 00 0I \\
q_{9} &= 0.19999999999999074D 00 I \\
q_{8} &= 0.2000000000000000D 00 I \\
q_{7} &= 0.2000000000000000D 00 I \\
q_{6} &= 0.2000000000000000D 00 I \\
q_{5} &= 0.2000000000000000D 00 I \\
q_{4} &= 0.2000000000000000D 00 I \\
q_{3} &= 0.2000000000000000D 00 I \\
q_{2} &= 0.2000000000000000D 00 I \\
q_{1} &= 0.2000000000000000D 00 0I \\
q_{0} &= 0.2000000000000000D 00 0I
\end{align*}
\]

Roots of \( p(x) \)

<table>
<thead>
<tr>
<th>Index</th>
<th>Root (x)</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 10</td>
<td>0.1000000000000070D 01 + 0.9999999999999974D 00 I</td>
<td>6</td>
</tr>
<tr>
<td>Root 21</td>
<td>0.1000000000000070D 01 + 0.9999999999999974D 00 I</td>
<td>6</td>
</tr>
</tbody>
</table>

Exhibit 6.14. Roots Are: 1+i (6), 1-i (6)
GREATEST COMMON DIVISION METHOD USED WITH MULCER'S 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 GEO: 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) = 0.1000000000000000D + 0.00D 00
P(6) = -0.1000000000000000D + 0.00D 00
P(5) = -0.1000000000000000D + 0.00D 00
P(4) = -0.1000000000000000D + 0.00D 00
P(3) = -0.1000000000000000D + 0.00D 00
P(2) = -0.1000000000000000D + 0.00D 00
P(1) = 0.1000000000000000D + 0.00D 00

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) = 0.1000000000000000D + 0.00D 00
Q(3) = -0.1000000000000000D + 0.00D 00
Q(2) = -0.1000000000000000D + 0.00D 00
Q(1) = 0.1000000000000000D + 0.00D 00

ROOTS OF Q(X)

ROOT 1 = 0.1000000000000000D + 0.00D 00
ROOT 2 = -0.1000000000000000D + 0.00D 00
ROOT 3 = -0.1000000000000000D + 0.00D 00

INITIAL APPROXIMATION

ROOT 1 = 0.1000000000000000D + 0.00D 00
ROOT 2 = -0.1000000000000000D + 0.00D 00
ROOT 3 = -0.1000000000000000D + 0.00D 00

MULTIPlicITIES

ROOT 1 = 0.1000000000000000D + 0.00D 00
ROOT 2 = -0.1000000000000000D + 0.00D 00
ROOT 3 = -0.1000000000000000D + 0.00D 00

RESULTS OF SUBROUTINE QUAD

Exhibit 6.15. Roots Are: 2+2i (3), 1+2i (2), -1+5i
GREATEST COMMON DIVISOR METHOD USED WITH MILLER'S 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-10
TEST FOR MULTIPLE ZEROS: 0.10D-01
RADIUS TO START SEARCH: 0.10D 00
RADIUS TO END SEARCH: 0.10D 00

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

P(16) = 0.480000000000000000 00 + 0.000000000000000000 00 I
P(15) = 0.225571200000000000 00 + 0.000000000000000000 00 I
P(14) = -0.385568959600000000 00 + 0.218968514560000000 00 I
P(13) = 0.102324621300000000 00 + 0.603066423200000000 00 I
P(12) = 0.104774120360000000 00 + 0.437366231000000000 00 I
P(11) = -0.203625588420000000 00 + 0.109899276720000000 00 I
P(10) = -0.181256780100000000 00 + 0.193996544300000000 00 I
P(9) = -0.127499729859000000 00 + 0.217134122742000000 00 I
P(8) = 0.132943443480000000 00 + 0.103813022655000000 00 I
P(7) = -0.305390077470000000 00 + 0.227456700000000000 00 I
P(6) = 0.305990077470000000 00 + 0.128999914130000000 00 I
P(5) = 0.2301599996044553D 02 - 0.5275680000068238D 03 I
P(4) = 0.2873459999999999D 03 - 0.9144610000075942D 03 I
P(3) = 0.257578400000054654 03 - 0.275562000000000000 00 I
P(2) = 0.1389999999999999D 03 - 0.162732160000000000 00 I
P(1) = 0.1389999999999999D 03 - 0.162732160000000000 00 I

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

Q(8) = 0.480000000000000000 00 + 0.000000000000000000 00 I
Q(7) = 0.1598559999959083D 03 - 0.1440000000018463D 03 I
Q(6) = 0.267301999999862370 00 + 0.5275680000068238D 03 I
Q(5) = 0.3278999997819040 00 - 0.9144610000075942D 03 I
Q(4) = 0.2301599996044553D 02 - 0.162732160000000000 00 I
Q(3) = 0.127499729859000000 00 + 0.217134122742000000 00 I
Q(2) = 0.1389999999999999D 03 - 0.162732160000000000 00 I
Q(1) = 0.226400000000035520 01 - 0.275562000000000000 00 I

ROOTS OF Q(X)

ROOT(1) = 0.300000000000000000 00 + 0.1370541778161090-09 I
ROOT(2) = 0.1446537249177010-09 + 0.70710673963460 00 + 0.100000000000000000 00 I
ROOT(3) = 0.1626215316532160-09 + 0.140000000000000000 00 I
ROOT(4) = 0.11404559591875120-09 + 0.300000000000000000 00 I

Exhibit 6.16.
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 MULLER'S 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: 0.100-02
TEST FOR CONVERGENCE: 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.100-19
TEST FOR MULTIPLEITIES: 0.100-01
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

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

P(8) = 0.1000000000000000D 01 + 0.0000000000000000000 00 1
P(7) = 0.1725000000000000D 01 + 0.4462500000000000000 01 1
P(6) = 0.1075000000000000D 02 + 0.3475000000000000000 02 1
P(5) = 0.5457500000000000D 02 + 0.4427500000000000000 02 1
P(4) = 0.1812500000000000D 03 + 0.4200000000000000000 03 1
P(3) = 0.3075000000000000D 03 + 0.3475000000000000000 03 1
P(2) = 0.1810000000000000D 03 + 0.4420000000000000000 03 1
P(1) = 0.1580000000000000D 03 + 0.6000000000000000000 01

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(3) = 0.1000000000000000D 01 + 0.0000000000000000000 00 1
Q(2) = 0.1000000000000000D 01 + 0.4500000000000000000 00 1
Q(1) = 0.2000000000000000D 00 + 0.7000000000000000000 00 1
Q(0) = 0.9999999999999999D 00 + 0.7000000000000000000 00 1

ROOTS OF P(X)

ROOT(1) = 0.0000000000000000D 00 + 0.1999999999999999D 01 1
ROOT(2) = 0.2000000000000000D 00 + 0.2000000000000000D 01 1
ROOT(3) = 0.4000000000000000D 00 + 0.4999999999999999D 00 1

ROOTS OF Q(X)

ROOT(1) = 0.0000000000000000D 00 + 0.1999999999999999D 01 1
ROOT(2) = 0.2000000000000000D 00 + 0.2000000000000000D 01 1
ROOT(3) = 0.4000000000000000D 00 + 0.4999999999999999D 00 1

MULTIPLICITIES

INITIAL APPROXIMATION

ROOT(1) = 0.0000000000000000D 00 + 0.1999999999999999D 01 1
ROOT(2) = 0.2000000000000000D 00 + 0.2000000000000000D 01 1
ROOT(3) = 0.4000000000000000D 00 + 0.4999999999999999D 00 1

INITIAL APPROXIMATION

ROOT(1) = 0.0000000000000000D 00 + 0.1999999999999999D 01 1
ROOT(2) = 0.2000000000000000D 00 + 0.2000000000000000D 01 1
ROOT(3) = 0.4000000000000000D 00 + 0.4999999999999999D 00 1

RESULTS OF SUBROUTINE QUAD

Exhibit 6.17. Roots Are: 2+2i (3), 1+2i (2), -1.5i (3)
GREATEST COMMON DIVISOR METHOD USED WITH MUJERRS 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: 0.10E-02
TEST FOR CONVERGENCE: 0.10E-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.10E-19
TEST FOR MULTIPICITIES: 0.10E-01
RADIUS TO START SEARCH: 0.00E 00
RADIUS TO END SEARCH: 0.00E 00

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

\[
\begin{align*}
P(13) &= 0.1000000000000000E+00 + 0.0000000000000000E+00 \quad 01 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(12) &= -0.1200000000000000E+02 + -0.0000000000000000E+00 \quad 01 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(11) &= 0.7200000000000000E+02 + 0.0000000000000000E+00 \quad 04 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(10) &= -0.2800000000000000E+03 + -0.0000000000000000E+00 \quad 06 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(9) &= 0.7800000000000000E+03 + 0.0000000000000000E+00 \quad 08 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(8) &= -0.1632000000000000E+04 + 0.0000000000000000E+00 \quad 10 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(7) &= 0.2624000000000000E+04 + 0.0000000000000000E+00 \quad 12 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(6) &= 0.3944000000000000E+04 + 0.0000000000000000E+00 \quad 14 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(5) &= 0.3120000000000000E+04 + 0.0000000000000000E+00 \quad 16 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(4) &= -0.2240000000000000E+04 + -0.0000000000000000E+00 \quad 18 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(3) &= 0.1152000000000000E+04 + 0.0000000000000000E+00 \quad 20 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(2) &= -0.3640000000000000E+04 + -0.0000000000000000E+00 \quad 22 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
P(1) &= 0.6400000000000001E+02 + 0.0000000000000000E+00 \quad 24 + 0.0000000000000000E+00 \quad 00 \quad 1
\end{align*}
\]

\( Q(X) \) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF \( P(X) \).

THE DEGREE OF \( Q(X) \) IS 2. THE COEFFICIENTS ARE

\[
\begin{align*}
Q(3) &= 0.1000000000000000E+00 + 0.0000000000000000E+00 \quad 01 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
Q(2) &= -0.2000000000000000E+01 + -0.0000000000000000E+00 \quad 01 + 0.0000000000000000E+00 \quad 00 \quad 1 \\
Q(1) &= 0.1999999999999830E+01 + 0.0000000000000000E+00 \quad 01 + 0.0000000000000000E+00 \quad 00 \quad 1
\end{align*}
\]

ROOTS OF \( P(X) \)

\[
\begin{align*}
\text{ROOT 1) } &= 0.1000000000000000E+00 + 0.9999999999999740E+00 \quad 01 + 0.0000000000000000E+00 \quad 01 + 0.9999999999999740E+00 \quad 00 \quad 1 \quad 6 \quad \text{RESULTS OF SUBROUTINE QUAD} \\
\text{ROOT 2) } &= 0.1000000000000000E+00 + 0.9999999999999740E+00 \quad 01 + 0.0000000000000000E+00 \quad 01 + 0.9999999999999740E+00 \quad 00 \quad 1 \quad 6 \quad \text{RESULTS OF SUBROUTINE QUAD}
\end{align*}
\]


declareExhibit{6.18}{Roots Are: \( 1+i(6) \), \( 1-i(6) \) /}
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTON'S 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. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAZ. 0.100-39
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE DEGREE OF PI(X) IS 6 & THE COEFFICIENTS ARE

PI1 = -0.1000000000000000 01 + 0.0000000000000000 00 I
PI2 = -0.7000000000000000 01 + 0.1090000000000000 02 I
PI3 = -0.2890000000000000 02 + 0.5800000000000000 01 I
PI4 = 0.1710000000000000 03 + 0.1500000000000000 01 I
PI5 = -0.7300000000000000 02 + 0.2410000000000000 03 I
PI6 = -0.2280000000000000 03 + 0.1040000000000000 03 I
PI7 = 0.7260000000000000 02 + 0.1080000000000000 03 I

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

THE FOLLOWING POLYNOMIAL, GI(X), CONTAINS ALL THE ROOTS OF PI(X) WHICH HAVE MULTIPlicity 1

GI2 = 0.1000000000000000 01 + 0.0000000000000000 00 I
GI1 = 0.9999999999999999 00 + 0.6000000000000000 00 I

ROOTS OF PI(X) MULTIPlicITIES INITIAL APPROXIMATIONS

ROOT( 1) = -0.9999999999999999 00 + 0.5000000000000000 00 I 1 NO INITIAL APPROXIMATIONS

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

THE FOLLOWING POLYNOMIAL, GI(X), CONTAINS ALL THE ROOTS OF PI(X) WHICH HAVE MULTIPlicity 2

GI2 = 0.1000000000000000 01 + 0.0000000000000000 00 I
GI1 = -0.9999999999999999 00 + 0.1000000000000000 01 I

Exhibit 6.19.
### Roots of \( p(x) \)

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.13</td>
<td>0.99999999021790 01 0.199999999964820 01</td>
<td>2</td>
</tr>
</tbody>
</table>

The following polynomial, \( q(x) \), contains all the roots of \( p(x) \) which have multiplicity 3:

\[
2 \times 0.1060000000000000 01 + 0.6600000000000000 01 = -0.1999999999999967 01 + -0.200000000000015190 01 1
\]

### Roots of \( q(x) \)

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.11</td>
<td>0.199999999999670 01 0.2000000000001519 01</td>
<td>3</td>
</tr>
</tbody>
</table>

Exhibit 6.19. Roots Are: 2+2i (3), 1+2i (2), -1+5i
NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.100-02
TEST FOR CONVERGENCE: 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.100-19
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

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

P(16) = 0.4800000000000000 D 02 + 0.0000000000000000 D 00 0 1
P(15) = -0.2557120000000000 D 02 + 0.1960000000000000 D 00 0 3 1
P(14) = -0.1705670100000000 D 02 + -0.2198696000000000 D 04 0 4 1
P(13) = -0.11929384646000000 D 02 + -0.1206299778000000 D 06 0 5 1
P(12) = -0.07657392700000000 D 02 + -0.1765857644000000 D 08 0 6 1
P(11) = -0.04967942700000000 D 02 + -0.4137366230400000 D 10 0 7 1
P(10) = -0.016427122005600000 D 02 + 0.4137366230400000 D 12 0 8 1
P(9) = -0.0100366210884200000 D 02 + 0.1095849276760000 D 14 0 9 1
P(8) = -0.001215357001000000 D 02 + 0.1029836463300000 D 16 1 0 1
P(7) = -0.001215357001000000 D 02 + -0.1087114672400000 D 18 1 1 1
P(6) = -0.001215357001000000 D 02 + 0.2177151227400000 D 20 1 2 1
P(5) = -0.001215357001000000 D 02 + 0.1926497276000000 D 22 1 3 1
P(4) = -0.001215357001000000 D 02 + 0.1038130226550000 D 24 1 4 1
P(3) = -0.001215357001000000 D 02 + 0.2998989141300000 D 26 1 5 1
P(2) = -0.001215357001000000 D 02 + 0.1827632160000000 D 28 1 6 1
P(1) = -0.001215357001000000 D 02 + 0.2756420000000000 D 30 1 7 1

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

G(2) = 0.4800000000000000 D 02 + 0.0000000000000000 D 00 0 1
G(1) = -0.11929384646000000 D 02 + -0.8274988658760000 D 00 0 2

ROOTS OF P(X)  MULTIPlicITIES  INITIAL APPROXIMATION

ROOT(1) = -0.23333333185333970 0 1 + 0.1723959705709930-07 1 NO INITIAL APPROXIMATIONS

Exhibit 6.20.
THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPlicity 2

<table>
<thead>
<tr>
<th>G13</th>
<th>G14</th>
<th>G15</th>
<th>G16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10000000000000000000 01</td>
<td>0.00000000000000000000 01</td>
<td>-0.299997902655364360 02</td>
<td>-0.146699932200921417 02</td>
</tr>
<tr>
<td>-0.299997902655364360 02</td>
<td>0.220598820448846440 01</td>
<td>0.9999999999999999 00</td>
<td>0.220598820448846440 01</td>
</tr>
<tr>
<td>0.10000000000000000000 01</td>
<td>0.9999999999999999 00</td>
<td>0.220598820448846440 01</td>
<td>0.220598820448846440 01</td>
</tr>
<tr>
<td>0.146699932200921417 02</td>
<td>0.220598820448846440 01</td>
<td>0.220598820448846440 01</td>
<td>0.220598820448846440 01</td>
</tr>
</tbody>
</table>

ROOTS OF G(X)

<table>
<thead>
<tr>
<th>ROOT</th>
<th>MULTIPlicITIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

INITIAL APPROXIMATION

<table>
<thead>
<tr>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.462796291156562790 00</td>
</tr>
<tr>
<td>0.7071680768665950 00</td>
</tr>
<tr>
<td>0.7071680768665950 00</td>
</tr>
<tr>
<td>0.7071680768665950 00</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>G13</th>
<th>G14</th>
<th>G15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10000000000000000000 01</td>
<td>0.00000000000000000000 01</td>
<td>-0.299997902655364360 02</td>
</tr>
<tr>
<td>-0.299997902655364360 02</td>
<td>0.220598820448846440 01</td>
<td>0.9999999999999999 00</td>
</tr>
<tr>
<td>0.10000000000000000000 01</td>
<td>0.9999999999999999 00</td>
<td>0.220598820448846440 01</td>
</tr>
<tr>
<td>0.146699932200921417 02</td>
<td>0.220598820448846440 01</td>
<td>0.220598820448846440 01</td>
</tr>
</tbody>
</table>

ROOTS OF P(X)

<table>
<thead>
<tr>
<th>ROOT</th>
<th>MULTIPlicITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
</tbody>
</table>

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 NEWTON'S 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 GEO: 0.10D-02
TEST FOR CONVERGENCE: 0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.10D-19
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

THE DEGREE OF \( P(x) \) IS 8. THE COEFFICIENTS ARE

\[
P(9) = 0.1000000000000000D + 0.0000000000000000D 01 1
P(8) = -0.5179999999999999D 01 1
P(7) = -0.5179999999999999D 01 1
P(6) = -0.5179999999999999D 01 1
P(5) = -0.5179999999999999D 01 1
P(4) = -0.5179999999999999D 01 1
P(3) = -0.5179999999999999D 01 1
P(2) = -0.5179999999999999D 01 1
P(1) = -0.5179999999999999D 01 1

NO ROOTS OF MULTIPLICITY 1

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

\[
G(1) = 0.1000000000000000D + 0.0000000000000000D 01 1
G(0) = -0.1999999999999999D 01 1

ROOTS OF \( P(x) \\) MULTIPLEITIES INITIAL APPROXIMATION

\[
\text{ROOT(1)} = 0.9999999999999999D 01 1 2 \text{ NO INITIAL APPROXIMATIONS}

Exhibit 6.21.
THE FOLLOWING POLYNOMIAL, \( G(x) \), CONTAINS ALL THE ROOTS OF \( P(x) \) WHICH HAVE MULTIPLICITY 3

\[
\begin{align*}
G(3) &= 0.1000000000000000D\,01 + 0.0000000000000000D\,00 \\
G(2) &= -0.1000000000000025\,01 + -0.2500000000001629\,01 \\
G(1) &= -0.3000000000003296\,01 + -0.1000000000004210\,01
\end{align*}
\]

<table>
<thead>
<tr>
<th>ROOTS OF ( P(x) )</th>
<th>MULTIPLICITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(1) = 0.2000000000000032,01 + 0.2000000000001653,01</td>
<td>3</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
<tr>
<td>ROOT(2) = -0.9999999999999926,00 + 0.4999999999999759,00</td>
<td>3</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
</tbody>
</table>

Exhibit 6.21. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)
NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.100-02
TEST FOR CONVERGENCE: 0.120-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.160-19
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 03

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

\[ P(0) = 0.1000000000000000 D1 + 0.0000000000000000 D0 I \]
\[ P(1) = 0.1000000000000000 D2 + 0.0000000000000000 D1 I \]
\[ P(2) = 0.7200000000000000 D0 + 0.0000000000000000 D2 I \]
\[ P(3) = 0.3800000000000000 D2 + 0.0000000000000000 D3 I \]
\[ P(4) = 0.2800000000000000 D3 + 0.0000000000000000 D4 I \]
\[ P(5) = 0.7800000000000000 D4 + 0.0000000000000000 D5 I \]
\[ P(6) = -0.1600000000000000 D5 + 0.0000000000000000 D6 I \]
\[ P(7) = -0.7600000000000000 D6 + 0.0000000000000000 D7 I \]
\[ P(8) = -0.3200000000000000 D7 + 0.0000000000000000 D8 I \]
\[ P(9) = 0.3120000000000000 D8 + 0.0000000000000000 D9 I \]
\[ P(10) = -0.3240000000000000 D9 + 0.0000000000000000 D10 I \]
\[ P(11) = 0.1150000000000000 D10 + 0.0000000000000000 D11 I \]
\[ P(12) = -0.8600000000000000 D11 + 0.0000000000000000 D12 I \]
\[ P(13) = 0.5400000000000000 D12 + 0.0000000000000000 D13 I \]

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

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 PI(X) WHICH HAVE MULTIPlicity 6.

\[ G_1 = 0.1000000000000000 \times 10^{01} + 0.0000000000000000 \times 10^{00} \]
\[ G_2 = -0.2000000000000000 \times 10^{01} + 0.0000000000000000 \times 10^{00} \]
\[ G_3 = 0.2000000000000000 \times 10^{01} + 0.0000000000000000 \times 10^{00} \]

<table>
<thead>
<tr>
<th>ROOT(S) PI(X)</th>
<th>MULTIPlicITIES</th>
<th>INITIAL APPROXIMATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(1) = 0.1000000000000000 \times 10^{01} + 0.4999999999999999 \times 10^{00}</td>
<td>6</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
<tr>
<td>ROOT(2) = 0.1000000000000000 \times 10^{01} + 0.5000000000000000 \times 10^{00}</td>
<td>6</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
</tbody>
</table>

Exhibit 6.22. Roots Are: 1+i (6), 1-i (6)
Repeated use of the greatest common divisor and nullis 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 GEO: 0.100-09
Test for convergence: 0.100-09
Test for zero in subroutine QUAD: 0.100-19
Radius to start search: 0.000 00
Radius to end search: 0.000 00

The degree of p(x) is 6, the coefficients are:

P[1] = 0.100000000000000000000000 00 1
P[1] = -0.700000000000000000000000 00 1
P[1] = 0.280000000000000000000000 02 1
P[1] = 0.171000000000000000000000 01 1
P[1] = -0.210000000000000000000000 03 1
P[1] = 0.120000000000000000000000 03 1

The following polynomial, G(x), contains all the roots of p(x) which have multiplicity 1:

G[2] = 0.100000000000000000000000 00 1
G[1] = 0.9999999999973500 00 -0.5000000000000 00 1

Roots of G(x)

Root 1) = -0.9999999999973500 00 - 0.50000000000175410 00

Initial Approximation

0.4829629115656279D 00 + 0.129409528438167D 00 1

Roots of P(x)

Root 1) = -0.9999999999973500 00 - 0.50000000000175410 00

Multiplicities

Initial Approximation

0.4829629115656279D 00 + 0.129409528438167D 00 1

Exhibit 6.23.
THE FOLLOWING POLYNOMIAL, $G(x)$, CONTAINS ALL THE ROOTS OF $P(x)$ WHICH HAVE MULTIPLICITY 2

\[ G(3) = 0.1000000000000000D\quad 01 + 0.0000000000000000D\quad 02 i \]
\[ G(1) = -0.999999999999017900 + 0.1999999999998450D\quad 01 i \]

**ROOTS OF $G(x)$**

<table>
<thead>
<tr>
<th>ROOT(3)</th>
<th>MULTIPLEILITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.999999999999917900 + 0.19999999999968450D\quad 01 i</td>
<td>0.482942911565627900 + 0.129409528443818700 i</td>
<td></td>
</tr>
</tbody>
</table>

**ROOTS OF $P(x)$**

<table>
<thead>
<tr>
<th>ROOT(1)</th>
<th>MULTIPLEITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.999999999999917900 + 0.19999999999968450D\quad 01 i</td>
<td>0.482942911565627900 + 0.129409528443818700 i</td>
<td></td>
</tr>
</tbody>
</table>

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

\[ 3\quad 3 = 0.1000000000000000D\quad 01 + 0.0000000000000000D\quad 02 i \]
\[ 1\quad 3 = -0.199999999999996704 + 0.2000000000000190 D\quad 01 i \]

**ROOTS OF $P(x)$**

<table>
<thead>
<tr>
<th>ROOT(3)</th>
<th>MULTIPLEITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.199999999999996704 + 0.2000000000000190 D\quad 01 i</td>
<td>3 NO INITIAL APPROXIMATIONS</td>
<td></td>
</tr>
</tbody>
</table>

**Exhibit 6.23. Roots Are:**

2+2i (3), 1+2i (2), -1+5i
NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GED: 0.100-00
TEST FOR CONVERGENCE: 0.100-00
TEST FOR ZERO IN SUBROUTINE QEDU: 0.100-19
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

THE DEGREE OF P1(X) IS 15 THE COEFFICIENTS ARE

P1(0) = 0.4800000000000000 02 + 0.0000000000000000 00 1
P1(1) = 0.2457123000000000 03 + -0.3846010000000000 03 1
P1(2) = -0.7595556000000000 03 + -0.7169460000000000 04 1
P1(3) = -0.3556505690000000 04 + -0.6446851456000000 04 1
P1(4) = -0.1733386468000000 05 + -0.1420629973000000 06 1
P1(5) = -0.4927992704000000 05 + -0.1769847446000000 09 1
P1(6) = -0.1022594121300000 06 + -0.6030666232000000 04 1
P1(7) = -0.1037926205600000 06 + -0.4137607226520000 05 1
P1(8) = -0.2036624588426000 06 + -0.1059873927000000 06 1
P1(9) = -0.1812958630100000 06 + -0.1929853423000000 06 1
P1(10) = -0.1374099808908000 06 + -0.2171342174520000 06 1
P1(11) = -0.2814642716120000 05 + -0.1928489729640000 06 1
P1(12) = -0.1325953932860000 05 + -0.1038103226590000 06 1
P1(13) = -0.3055963734700000 05 + -0.2988097911300000 05 1
P1(14) = -0.1838599202000000 03 + -0.1827682160500000 03 1
P1(15) = 0.2755920000000000 00 + 0.2755920000000000 00 1

THE FOLLOWING POLYNOMIAL, GIX1, CONTAINS ALL THE ROOTS OF P1(X) WHICH HAVE MULTIPLICITY 1

GIX1 = 0.4800000000000000 02 + 0.0000000000000000 00 1
GIX1 = 0.1119999992094030 03 + -0.8274986587407600-06 01

ROOTS OF GIX1

ROOTS OF GIX1

MULTIPlicITIES

INITIAL APPROXIMATION


eXhibit 6.24.

78
THE FOLLOWING POLYNOMIAL, G(x), CONTAINS ALL THE ROOTS OF P(x) WHICH HAVE MULTIPlicity 2

\[
\begin{align*}
G(1) &= a_1 x^3 + a_2 x^2 + a_3 x + a_4 \\
G(2) &= b_1 x^3 + b_2 x^2 + b_3 x + b_4 \\
G(3) &= c_1 x^3 + c_2 x^2 + c_3 x + c_4 \\
G(4) &= d_1 x^3 + d_2 x^2 + d_3 x + d_4 \\
G(5) &= e_1 x^3 + e_2 x^2 + e_3 x + e_4 \\
G(6) &= f_1 x^3 + f_2 x^2 + f_3 x + f_4 \\
isolution: \quad & \begin{pmatrix}
0.4879629115666793 \\
0.1294095284381870 \\
0.1294095284381870 \\
0.1294095284381870 \\
0.1294095284381870 \\
0.1294095284381870 \\
\end{pmatrix}
\end{align*}
\]

ROOTS OF P(x) AND MULTIPlicITIES

\[
\begin{align*}
\text{ROOTS OF P(x)} & \quad \text{MULTIPlicITIES} \\
\text{ROOTS OF G(x)} & \quad \text{INITIAL APPROXIMATION} \\
\text{ROOTS OF G(x)} & \quad \text{SOLVED BY DIRECT METHOD} \\
\text{ROOTS OF G(x)} & \quad \text{NO INITIAL APPROXIMATIONS}
\end{align*}
\]

THE FOLLOWING POLYNOMIAL, G(x), CONTAINS ALL THE ROOTS OF P(x) WHICH HAVE MULTIPlicity 3

\[
\begin{align*}
G(1) &= a_1 x^3 + a_2 x^2 + a_3 x + a_4 \\
G(2) &= b_1 x^3 + b_2 x^2 + b_3 x + b_4 \\
G(3) &= c_1 x^3 + c_2 x^2 + c_3 x + c_4 \\
G(4) &= d_1 x^3 + d_2 x^2 + d_3 x + d_4 \\
G(5) &= e_1 x^3 + e_2 x^2 + e_3 x + e_4 \\
G(6) &= f_1 x^3 + f_2 x^2 + f_3 x + f_4 \\
isolution: \quad & \begin{pmatrix}
0.4879629115666793 \\
0.1294095284381870 \\
0.1294095284381870 \\
0.1294095284381870 \\
0.1294095284381870 \\
0.1294095284381870 \\
\end{pmatrix}
\end{align*}
\]

ROOTS OF G(x)

\[
\begin{align*}
\text{ROOTS OF G(x)} & \quad \text{INITIAL APPROXIMATION} \\
\text{ROOTS OF G(x)} & \quad \text{SOLVED BY DIRECT METHOD} \\
\text{ROOTS OF G(x)} & \quad \text{NO INITIAL APPROXIMATIONS}
\end{align*}
\]

Exhibit 6.24.
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 MULLEIS 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. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE QRAO. 0.100-19
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE DEGREE OF PI3 IS 6 THE COEFFICIENTS ARE

P11 = 0.1000000000000000 01 + 0.0000000000000000 00 1
P12 = -0.1000000000000000 01 + 0.1100000000000000 02 1
P13 = 0.1000000000000000 01 + 0.1200000000000000 02 2
P14 = 0.1000000000000000 01 + 0.1300000000000000 03 0
P15 = 0.1000000000000000 01 + 0.1400000000000000 03 1
P16 = 0.1000000000000000 01 + 0.1500000000000000 03 2
P17 = 0.1000000000000000 01 + 0.1600000000000000 03 3
P18 = 0.1000000000000000 01 + 0.1700000000000000 03 4
P19 = 0.1000000000000000 01 + 0.1800000000000000 03 5

THE FOLLOWING POLYNOMIAL, GI11, CONTAINS ALL THE ROOTS OF PI3 WHICH HAVE MULTIPLICITY 2

GI11 = 0.1000000000000000 01 + 0.0000000000000000 00 1
GI11 = -0.1999999999999990 00 + 0.1999999999999990 01 1

ROOTS OF GI11

ROOT1 1 = 0.1999999999999990 00 + 0.1999999999999990 01 1
0.4829629135652790 00 + 0.1294952844381870 00 1

Exhibit 6.25.
ROOTS OF P(X)  

MULTIPlicITIES  

INITIAL APPROXIMATION

ROOT (1) = 0.9999999999944980 00 + 0.1999999999994000 01 1
2 0.4829629115662790 00 + 0.129409529044381870 00 1

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

G(3) = 0.1000000000000000 01 + 0.0000000000000000 00 1
G(2) = -0.1000000000000000 01 + -0.2500000000000000 01 1
G(1) = -0.10000000000003296 01 + -0.10000000000004210 01 1

ROOTS OF G(X)  

INITIAL APPROXIMATION

ROOT (2) = -0.99999999999999280 00 + 0.4999999999999759 00 1
ROOT (1) = 0.2000000000000000 01 + 0.200000000000016530 01 1
SOLVED BY DIRECT METHOD

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT (2) = 0.200000000000025320 01 + 0.20000000000000016530 01 1
DID NOT CONVERGE AFTER 200 ITERATIONS
THE PRESENT APPROXIMATION IS: 0.2001999974253260 01 + 0.20019999742526480 01 1

ROOTS OF P(X)  

MULTIPlicITIES  

INITIAL APPROXIMATION

ROOT (1) = -0.99999999999999300 00 + 0.4999999999999759 00 1
3 0.4829629115662790 00 + 0.129409529044381870 00 1

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 Muller's 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: 0.100-07
Test for convergence: 0.100-08
Test for zero in subroutine Quad: 0.100-19
Radius to start search: 0.000 00
Radius to end search: 0.000 00

The degree of P(x) is 12. The coefficients are

\[
P(13) = -0.1003000000000000D 01 + 0.0000000000000000D 00 01
P(1) = -0.1200000000000000D 02 + 0.0000000000000000D 00 01
P(11) = 0.7800000000000001D 02 + 0.0000000000000000D 00 01
P(10) = -0.2800000000000000D 03 + 0.0000000000000000D 00 01
P(9) = 0.7600000000000000D 03 + 0.0000000000000000D 00 01
P(8) = -0.1632000000000000D 04 + 0.0000000000000000D 00 01
P(7) = 0.2624000000000000D 04 + 0.0000000000000000D 00 01
P(6) = -0.3204000000000000D 04 + 0.0000000000000000D 00 01
P(5) = 0.3120000000000000D 04 + 0.0000000000000000D 00 01
P(4) = -0.2240000000000000D 04 + 0.0000000000000000D 00 01
P(3) = 0.1152000000000000D 04 + 0.0000000000000000D 00 01
P(2) = -0.3840000000000000D 03 + 0.0000000000000000D 00 01
P(1) = 0.6400000000000001D 02 + 0.0000000000000000D 00 01
\]

No roots of multiplicity 1

No roots of multiplicity 2

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

\[
G(3) = 0.10000000000000000 + 0.00000000000000000D \times 001
\]

\[
G(12) = 0.20000000000000066D \times 001 - 0.0000000000000000000D \times 001
\]

ROOTS OF G(X)

ROOT(1) = 0.1000000000000033D \times 001 + 0.9999999999999707D \times 001

\[
0.482967915656297000 + 0.1294095284438187D \times 001
\]

ROOT(2) = 0.1000000000000033D \times 001 - 0.9999999999999707D \times 001

\[
0.482967915656297000 - 0.1294095284438187D \times 001
\]

ROOTS OF P(X) MULTIPLICITIES INITIAL APPROXIMATION

ROOT(1) = 0.10000000000000000 + 0.9999999999999707D \times 001 6

\[
0.482967915656297000 + 0.1294095284438187D \times 001
\]

ROOT(2) = 0.10000000000000000 - 0.9999999999999707D \times 001 6

\[
0.482967915656297000 - 0.1294095284438187D \times 001
\]

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


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:

$$X_k = (\text{XSTART} + 0.5k) (\cos \beta + i \sin \beta)$$

where

$$\beta = \frac{\pi}{12} + \frac{\pi}{6} k, \quad k = 0,1,2,...$$

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.

```
Variable Name
Card Columns

<table>
<thead>
<tr>
<th>N</th>
<th>O</th>
<th>P</th>
<th>L</th>
<th>Y</th>
<th>6</th>
<th>7</th>
<th>7</th>
<th>7</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

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.
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: $(j = 1, 3)$

$$X_{j+1} = |X_0| \left( \cos \beta + i \sin \beta \right)$$

where

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

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.
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 \( XEND + 0.5N \).

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

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 \( n^{\text{th}} \) 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 NAP, 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).

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>O</td>
<td>P</td>
<td>L</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XSTART</td>
<td>XEND</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 7th degree polynomial. After these approximations are used the circle of radius 15 will be searched for the remaining roots.

<table>
<thead>
<tr>
<th>1 2</th>
<th>4 5</th>
<th>7 8</th>
<th>6 7</th>
<th>7 7</th>
<th>8 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOPOLY</td>
<td>N</td>
<td>N</td>
<td>XSTART</td>
<td>XEND</td>
<td></td>
</tr>
<tr>
<td>1 7 2</td>
<td></td>
<td></td>
<td>1.5D+01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 2 OF DEGREE 3

THE COEFFICIENTS OF FIRST ARE

P1 1) = 0.10000000000000000 01 + 0.00000000000000000 00 0
P1 2) = 0.20000000000000000 01 + 0.00000000000000000 00 0
P1 3) = -0.10000000000000000 01 + 0.00000000000000000 00 0
P1 4) = -0.20000000000000000 01 + 0.00000000000000000 00 0

NUMBERS OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 3
TEST FOR CONVERGENCE. 0.100-03
TEST FOR MULTIPLEITIES. 0.100-01
RADIUS TO START SEARCH. 0.00 00
RADIUS TO END SEARCH. 0.00 00

NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER 3 ITERATIONS.

D.48796291156562700 00 = 0.12940952644381870 00 1
D.12940952644381870 00 = 0.12940952644381870 00 1
D.0.12940952644381870 00 = 0.12940952644381870 00 1
D.0.353553249184020 00 = 0.353553249184020 00 1
D.0.353553249184020 00 = 0.353553249184020 00 1
D.0.70710675530463460 00 = 0.70710675530463460 00 1
D.0.70710675530463460 00 = 0.70710675530463460 00 1
D.0.2940949308846860 00 = 0.2940949308846860 00 1
D.0.48296291234546350 00 = 0.48296291234546350 00 1
D.0.48296291234546350 00 = 0.48296291234546350 00 1

COEFFICIENTS OF DEFATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

P1 1) = 0.10000000000000000 01 + 0.00000000000000000 00 0
P1 2) = 0.20000000000000000 01 + 0.00000000000000000 00 0
P1 3) = -0.10000000000000000 01 + 0.00000000000000000 00 0
P1 4) = -0.20000000000000000 01 + 0.00000000000000000 00 0

Exhibit A.1.
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 1 OF DEGREE 7

THE COEFFICIENTS OF PI(x) ARE

\[ p_1(x) = 0.10000000000000000 + 0.00000000000000000 \]
\[ p_2(x) = 0.11000000000000000 + 0.00000000000000000 \]
\[ p_3(x) = 0.59000000000000000 + 0.00000000000000000 \]
\[ p_4(x) = 0.19000000000000000 + 0.00000000000000000 \]
\[ p_5(x) = 0.70000000000000000 + 0.00000000000000000 \]
\[ p_6(x) = -0.16240000000000000 + 0.00000000000000000 \]
\[ p_7(x) = 0.19220000000000000 + 0.00000000000000000 \]
\[ p_8(x) = -0.159b00000000000000 + 0.00000000000000000 \]

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 3

MAXIMUM NUMBER OF ITERATIONS. 200

TEST FOR CONVERGENCE. 0.100-00

TEST FOR MULTIPlicITIES. 0.100-00

RADIUS TO START SEARCH. 0.000 00

RADIUS TO END SEARCH. 0.000 00

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF PI(x) ARE

ROOTS OF PI(x)  MULTIPLICITIES  INITIAL APPROXIMATION

\[ \text{ROOT } 1 = -0.29999999999997D \]  1  \[ -0.3500000000000000D \]
\[ \text{ROOT } 2 = -0.29999999999997D \]  1  \[ -0.3500000000000000D \]
\[ \text{ROOT } 3 = -0.39999999999999D \]  1  \[ -0.3500000000000000D \]

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF PI(x) ARE

ROOTS OF PI(x)  MULTIPLICITIES  INITIAL APPROXIMATION

\[ \text{ROOT } 1 = -0.29999999999997D \]  1  \[ -0.3500000000000000D \]
\[ \text{ROOT } 2 = -0.39999999999999D \]  1  \[ -0.3500000000000000D \]
\[ \text{ROOT } 3 = -0.45000000000001D \]  1  \[ -0.3500000000000000D \]

COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

\[ d_1(x) = 0.10000000000000000 + 0.00000000000000000 \]
\[ d_2(x) = 0.29999999999997D \]  1  \[ -0.4000000000000000D \]
\[ d_3(x) = -0.20000000000000000 + 0.00000000000000000 \]
\[ d_4(x) = -0.41000000000000000 + 0.00000000000000000 \]
\[ d_5(x) = -0.63000000000000000 + 0.00000000000000000 \]

Exhibit A.2. Roots Are: -1 - 4i, -2 - 3i, -3 - 3i, -1 - i, 2 + 2i, 4 - 1, 2 - i.
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 1 OF DEGREE 7

THE COEFFICIENTS OF P(x) ARE

\[ P(x) = -0.10000000000000000000 + 0.00000000000000000000 x + 0.00000000000000000000 x^2 + 0.00000000000000000000 x^3 + 0.00000000000000000000 x^4 + 0.00000000000000000000 x^5 + 0.00000000000000000000 x^6 + 0.00000000000000000000 x^7 + 0.00000000000000000000 x^8 \]

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 2

TEST FOR CONVERGENCE: 0.10000000000000000000

TEST FOR MULTIPLICITIES: 0.10000000000000000000

RADIUS TO START SEARCH: 0.70000000000000000000

RADIUS TO END SEARCH: 0.15000000000000000000

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

ROOT(1): -0.29999999999999960

ROOT(2): 0.20000000000000000

ROOT(3): 0.20000000000000000

ROOT(4): 0.25000000000000000

ROOT(5): 0.25000000000000000

ROOT(6): 0.25000000000000000

ROOT(7): 0.25000000000000000

HOME: Solved by direct method.

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

ROOT(1): -0.29999999999999960

ROOT(2): 0.20000000000000000

ROOT(3): 0.20000000000000000

ROOT(4): 0.25000000000000000

ROOT(5): 0.25000000000000000

ROOT(6): 0.25000000000000000

ROOT(7): 0.25000000000000000

HOME: 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(d1,d2) - arctangent of d1/d2
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

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOPOLY</td>
<td>c.c. 1-2</td>
<td>Number of the polynomial. Integer. Right justified.</td>
</tr>
<tr>
<td>N</td>
<td>c.c. 4-5</td>
<td>Degree of the polynomial. Integer. Right justified.</td>
</tr>
<tr>
<td>NIAP</td>
<td>c.c. 7-8</td>
<td>Number of initial approximations to be read. Integer. If no approximations are given, this should be left blank.</td>
</tr>
<tr>
<td>MAX</td>
<td>c.c. 19-21</td>
<td>Maximum number of iterations. Integer. Right justified. 200 is recommended.</td>
</tr>
<tr>
<td>EPSCNV</td>
<td>c.c. 30-35</td>
<td>Convergence requirement. Double precision. 1.D-10 is recommended.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Card Columns</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 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>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA (A in single precision)</td>
<td>c.c. 1-30</td>
<td>Real part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0D00.</td>
</tr>
<tr>
<td>VA (A in single precision)</td>
<td>c.c. 31-60</td>
<td>Imaginary part of complex coefficient. Double precision. Right justify. If none, leave blank or enter 0.0D00.</td>
</tr>
</tbody>
</table>

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**

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

**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**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCHECK</td>
<td>c.c. 80</td>
<td>Must contain the number 1. Integer.</td>
</tr>
</tbody>
</table>
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
### Variable Name

#### Card Columns

<table>
<thead>
<tr>
<th>Variable</th>
<th>N</th>
<th>I</th>
<th>A</th>
<th>MAX</th>
<th>EPSCNV</th>
<th>ESPQ</th>
<th>EPSMUL</th>
<th>XSTART</th>
<th>XEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>200</td>
<td>1.D-10</td>
<td>1.D-20</td>
<td>1.D-02</td>
<td>1.0D+01</td>
<td>5.0D+02</td>
</tr>
</tbody>
</table>

**Figure B.2.** Control Card for Newton's Method
### Figure B.3. Coefficient Card for Newton's Method

<table>
<thead>
<tr>
<th>A (RA)</th>
<th>A (VA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.621735D+01</td>
<td>-0.132714D-02</td>
</tr>
</tbody>
</table>
### Figure B.4. Initial Approximation Card for Newton's Method

<table>
<thead>
<tr>
<th>XZERO (RXZERO)</th>
<th>XZERO (VXZERO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.15D-01</td>
<td>-0.25D-00</td>
</tr>
</tbody>
</table>

### Figure B.5. End Card for Newton's Method
### TABLE B. VII

**VARIABLES USED IN NEWTON’S METHOD**

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

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XO</td>
<td>C</td>
<td>RXO, VXO</td>
<td>D</td>
<td></td>
<td>Current approximation ($X_n$) to root</td>
</tr>
<tr>
<td>COEF</td>
<td>C</td>
<td>RCOEF, VCOEF</td>
<td>D</td>
<td></td>
<td>Working array containing coefficients of current deflated polynomial</td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX, VDPX</td>
<td>D</td>
<td></td>
<td>Derivative of $P(X)$ at some value $X$</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>RPX, VPX</td>
<td>D</td>
<td></td>
<td>Value of $P(X)$ at some point $X$</td>
</tr>
<tr>
<td>XZERO</td>
<td>C</td>
<td>RXZERO, VXZERO</td>
<td>D</td>
<td></td>
<td>Array containing the initial approximations</td>
</tr>
<tr>
<td>XNEW</td>
<td>C</td>
<td>RXNEW, VXNEW</td>
<td>D</td>
<td></td>
<td>New approximation ($X_{n+1}$) obtained from old approximation ($X_n$) by Newton's Algorithm</td>
</tr>
<tr>
<td>KANS</td>
<td>I</td>
<td>KANS</td>
<td>I</td>
<td></td>
<td>$KANS = 1$ implies convergence, $KANS = 0$ implies no convergence</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td></td>
<td>Array containing the number of multiplicities of each root</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>RX, VX</td>
<td>D</td>
<td></td>
<td>Array containing the zeros of $P(X)$</td>
</tr>
<tr>
<td>XINIT</td>
<td>C</td>
<td>RXINIT, VXINIT</td>
<td>D</td>
<td></td>
<td>Array containing the initial or altered approximations which produced convergence to each root</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
<td>I</td>
<td></td>
<td>Number of coefficients of current deflated polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>RB, VB</td>
<td>D</td>
<td></td>
<td>Array containing the coefficients of newly deflated polynomial</td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</td>
<td>I</td>
<td></td>
<td>Number of distinct roots found by Newton's method, i.e. not solved for directly by subroutine QUAD</td>
</tr>
<tr>
<td>EPSCHK</td>
<td>R</td>
<td>EPSCHK</td>
<td>D</td>
<td></td>
<td>Array containing the coefficients of deflated polynomial for which no zeros were found</td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
<td></td>
<td>Unit number of input device</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td></td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>RC, VC</td>
<td>D</td>
<td></td>
<td>Array containing sequence of values leading to the derivative</td>
</tr>
<tr>
<td>EPSCHK</td>
<td>R</td>
<td>EPSCHK</td>
<td>D</td>
<td></td>
<td>Current tolerance for checking convergence or multiplicity</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Single Precision Type</td>
<td>Double Precision Variable</td>
<td>Double Precision Type</td>
<td>Disposition of Argument</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------</td>
<td>---------------------------</td>
<td>-----------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>RA, VA</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>RB, VB</td>
<td>D</td>
<td>R</td>
<td>Array of coefficients of deflated polynomial</td>
</tr>
<tr>
<td>NDEF</td>
<td>I</td>
<td>NDEF</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
<td>I</td>
<td></td>
<td>Number of coefficients of polynomial</td>
</tr>
<tr>
<td>X0</td>
<td>C</td>
<td>RXO, VXO</td>
<td>D</td>
<td>E</td>
<td>Point (X_n) at which to evaluate the polynomial and its derivative. Also current approximation (X_{n+1}) used to deflate the polynomial</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>RPX, VPX</td>
<td>D</td>
<td>R</td>
<td>Value of polynomial at (X_n)</td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX, VDPX</td>
<td>D</td>
<td>R</td>
<td>Value of the derivative of polynomial at (X_n)</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>RC, VC</td>
<td>D</td>
<td>R</td>
<td>Array of containing sequence of values leading to the derivative</td>
</tr>
</tbody>
</table>

**Subroutine HORNER**

**Subroutine NEWTON**

**Subroutine CHECK**

<table>
<thead>
<tr>
<th>EPSILON</th>
<th>R</th>
<th>EPS</th>
<th>D</th>
<th>C</th>
<th>Tolerance for convergence or multiplicity check</th>
</tr>
</thead>
<tbody>
<tr>
<td>PX</td>
<td>C</td>
<td>RPX, VPX</td>
<td>D</td>
<td>E</td>
<td>Value of (P(X)) at (X_n)</td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX, VDPX</td>
<td>D</td>
<td>E</td>
<td>Derivative of (P(X)) at (X_n)</td>
</tr>
<tr>
<td>X0</td>
<td>C</td>
<td>RXO, VXO</td>
<td>D</td>
<td>E</td>
<td>Current approximations ((X_{n+1})) to root</td>
</tr>
<tr>
<td>IO2</td>
<td>I</td>
<td>IO2</td>
<td>I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>KANS</td>
<td>I</td>
<td>KANS</td>
<td>I</td>
<td>R</td>
<td>(KANS = 1) implies convergence, (KANS = 0) implies no convergence</td>
</tr>
</tbody>
</table>
TABLE B. VII (Continued)

<table>
<thead>
<tr>
<th>Single Precision</th>
<th>Double Precision</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Variable</td>
<td>Type</td>
</tr>
<tr>
<td>IO2</td>
<td>I</td>
<td>IO2</td>
<td>I</td>
</tr>
<tr>
<td>XZERO</td>
<td>C</td>
<td>RXZERO,VXZERO</td>
<td>D</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>RX,VX</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>RA,VA</td>
<td>D</td>
</tr>
<tr>
<td>COEF</td>
<td>C</td>
<td>RCOEF,VCOEF</td>
<td>D</td>
</tr>
<tr>
<td>NA</td>
<td>I</td>
<td>NA</td>
<td>I</td>
</tr>
<tr>
<td>X0</td>
<td>C</td>
<td>RX0,VX0</td>
<td>D</td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX,VDPX</td>
<td>D</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>RPX,VPX</td>
<td>D</td>
</tr>
<tr>
<td>KANS</td>
<td>I</td>
<td>KANS</td>
<td>I</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
</tr>
<tr>
<td>XNEW</td>
<td>C</td>
<td>RXNEW,VXNEW</td>
<td>D</td>
</tr>
<tr>
<td>NN</td>
<td>I</td>
<td>NN</td>
<td>I</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>RC,VC</td>
<td>D</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>RB,VB</td>
<td>D</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
</tr>
<tr>
<td>EPSCHK</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
</tr>
<tr>
<td>Subroutine BETTER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>APPR,APPI</td>
<td>D</td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
</tr>
</tbody>
</table>
### TABLE B. VII (Continued)

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSTART</td>
<td>R</td>
<td>XSTART</td>
<td>D</td>
<td>ECR</td>
<td>Magnitude at which to begin generating approximations; also magnitude of the approximation being generated</td>
</tr>
<tr>
<td>BETA</td>
<td>R</td>
<td>BETA</td>
<td>D</td>
<td></td>
<td>Argument of the complex approximation being generated</td>
</tr>
<tr>
<td>U</td>
<td>R</td>
<td>APPR(I)</td>
<td>D</td>
<td></td>
<td>Real part of complex approximation</td>
</tr>
<tr>
<td>V</td>
<td>R</td>
<td>APPI(I)</td>
<td>D</td>
<td></td>
<td>Imaginary part. of complex approximation</td>
</tr>
<tr>
<td>XOLD</td>
<td>C</td>
<td>XOLDR,XOLDI</td>
<td>D</td>
<td>ECR</td>
<td>Old approximation to be altered to new approximation</td>
</tr>
<tr>
<td>NALTER</td>
<td>I</td>
<td>NALTER</td>
<td>I</td>
<td>ECR</td>
<td>Number of alterations performed on an initial approximation</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
<td>E</td>
<td>Program control</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td>C</td>
<td>Maximum number of iterations permitted</td>
</tr>
<tr>
<td>Y</td>
<td>R</td>
<td>XOLDI</td>
<td>D</td>
<td></td>
<td>Imaginary part of original initial approximation (unaltered)</td>
</tr>
<tr>
<td>X</td>
<td>R</td>
<td>XOLDR</td>
<td>D</td>
<td></td>
<td>Real part of original unaltered initial approximation</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>D</td>
<td></td>
<td>Magnitude of original unaltered initial approximation</td>
</tr>
<tr>
<td>BETA</td>
<td>R</td>
<td>BETA</td>
<td>D</td>
<td></td>
<td>Argument of new approximation</td>
</tr>
<tr>
<td>XOLDR</td>
<td>R</td>
<td>XOLDR</td>
<td>D</td>
<td></td>
<td>Real part of new approximation</td>
</tr>
<tr>
<td>XOLDI</td>
<td>R</td>
<td>XOLDI</td>
<td>D</td>
<td></td>
<td>Imaginary part of new approximation</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
</tbody>
</table>

**Subroutine ALTER**

- A: Coefficients of polynomial to be solved
- NA: Degree of polynomial
- ROOT: Array of roots of P(X) (original polynomial)
- NROOT: Number of distinct roots of P(X) (the original polynomial)
<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTI</td>
<td>I</td>
<td>MULTI</td>
<td>I</td>
<td>ECR</td>
<td>Array containing multiplicities of each root</td>
</tr>
<tr>
<td>EPST</td>
<td>R</td>
<td>EPST</td>
<td>D</td>
<td>E</td>
<td>Tolerance check for the number zero</td>
</tr>
<tr>
<td>DISC</td>
<td>C</td>
<td>UDISC, VDISC</td>
<td>D</td>
<td></td>
<td>Value of the discriminate ((b^2 - 4ac)) of Quadratic</td>
</tr>
<tr>
<td>UX, VX</td>
<td>D</td>
<td>E</td>
<td>Complex number for which the square root is desired</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UY, VY</td>
<td>D</td>
<td>R</td>
<td>Square root of the complex number</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subroutine COMSQT
Figure B.6. Flow Charts for Newton's Method
Figure B.6. (Continued)
Figure B.6. (Continued)
Figure 3.6. (Continued)
Figure 3.5. (Continued)
Figure B.6. (Continued)
**TABLE B. VIII**

**PROGRAM FOR NEWTON'S METHOD**

```fortran
C
C * DOUBLE PRECISION PROGRAM FOR NEWTON'S METHOD
C
C * NEWTON'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPlicILITIES 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(IN+1) = X(IN) - P(X(IN))/P'(X(IN)).
C
*****************************************************************************
0001 DOUBLE PRECISION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26),RD(26),VD(26),RXZERO(25),RX(25),RXZERO(25),RX(25),RXZERO(25),RX(25),RXZERO(25),RX(25)
0002 DIMENSION RA(26),VA(26),RB(26),VB(26),RC(26),VC(26),RD(26),VD(26),RXZERO(25),RX(25),RXZERO(25),RX(25),RXZERO(25),RX(25),RXZERO(25),RX(25)
0003 COMMON EPSCHK,MAX,102
0004 101=5
0005 102=6
0006 1 READ(101,1000) N,NAP,NMAX,EPSCNV,EPQS,EPSEMUL,XSTART,XEND,KECK
0007 IF(KECK.EQ.1) STOP
0008 NA=N+1
0009 CALL GENAPP(RXZERO,RXZERO,NAP,XSTART)
0010 NDEF=N
0011 L=1
0012 ITER=0
0013 NROOT=0
0014 ITIME=0
0015 NALT=0
0016 RXO=RXZERO(L)
0017 VXO=VXZERO(L)
0018 DO 10 I=1,NA
0019 RCOEF(I)=RA(I)
0020 VCOEF(I)=VA(I)
0021 CALL HORNER(RCOEF,VCOEF,RXO,VXO,NDEF,RO,VQ,RC,RPX,VXP,RPX,VPX,VOPX,NAP)
0022 ABPX=DSQRT(RPX*RPX+VPX*VPX)
0023 ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
```

This Fortran program implements Newton's Method for finding the zeros and their multiplicities of a polynomial of maximum degree 25. It reads the polynomial coefficients and other parameters from an external file and performs the calculations accordingly.
TABLE B. VIII (Continued)

0041 IF(ABOPX .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
0046 RXO=RXNEW
0047 VXO=VXNEW
0048 EPSCHK=EPSCHV
0049 CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
0050 IF(KANS.EQ.1) GO TO 70
0051 IF(ITER.GE.MAX) GO TO 40
0052 GO TO 10
0053 40 CALL ALTER(RXZEROIL),VXZEROIL),IALTER,ETIME)
0054 IF(ITALTER.GT.5) GO TO 110
0055 RXO=RXZEROIL)
0056 VXO=VXZEROIL)
0057 ITER=0
0058 GO TO 10
0059 60 NDEF=NDEF+1
0060 DO 65 J=1,N
0061 RD(J)=RCOF(J)
0062 65 VD(J)=VCOF(J)
0063 GO TO 140
0064 70 NROOT=NROOT+1
0065 K=K+1
0066 MULT(K)=1
0067 RX(K)=RXO
0068 VX(K)=VXO
0069 RXINIT(K)=RXZEROIL)
0070 VXINIT(K)=VXZEROIL)
0071 CALL HORNER(RCOEF,VCOF,RXO,VCXVDEFtRBV8,RCVCRPXVPXtROPXVDPX)
0072 IF(INROOT.GE.N) GO TO 147
0073 NDEF=NDEF-1
0074 NUM=NDEF-1
0075 DO 105 I=1,NUM
0076 RCOEF(I)=RB(I)
0077 105 VCOEF(I)=VB(I)
0078 CALL HORNER(RCOEF,VCOF,RXO,VCXVDEF,RXVXVINIT,EPSEQ)
0079 11 ABPK=DSQRT(RPX*RPX+VPX*VPX)
0080 ABDPK=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 EPSCHV=EPSMUL
0086 CALL CHECK(RPX,VPX,RDPX,VDPX,RXO,VXO,KANS)
0087 IF(KANS.EQ.1) GO TO 130
0088 110 IF(NDEF.GT.2) GO TO 113
0089 K=K
0090 CALL QUAD(RCOEF,VCOF,NDEF,RX,VK,MULT,EPSEQ)
0091 GO TO 150
0092 113 IF(INIT,L,NIAP) GO TO 115
0093 IF(KEND.EQ.0.0) GO TO 60
0094 IF(KSTART.EQ.0.) GO TO 60
0095 NIAP=N
0096 CALL GENAPP(RXZERO,VZERO,NIAP,XSTART)
TABLE B. VIII (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0097</td>
<td>$L = 0$</td>
</tr>
<tr>
<td>0098</td>
<td>$L = L + 1$</td>
</tr>
<tr>
<td>0099</td>
<td>$RX = RXZERO(L)$</td>
</tr>
<tr>
<td>0100</td>
<td>$VX = VXZERO(L)$</td>
</tr>
<tr>
<td>0101</td>
<td>$ITER = 0$</td>
</tr>
<tr>
<td>0102</td>
<td>$IALTER = 0$</td>
</tr>
<tr>
<td>0103</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0104</td>
<td>$MULT(K) = MULT(K) + 1$</td>
</tr>
<tr>
<td>0105</td>
<td>$NROOT = NROOT + 1$</td>
</tr>
<tr>
<td>0106</td>
<td>GO TO 80</td>
</tr>
<tr>
<td>0107</td>
<td>IF $(K.EQ.0)$ GO TO 170</td>
</tr>
<tr>
<td>0108</td>
<td>IF $(ROOT = 8)$</td>
</tr>
<tr>
<td>0109</td>
<td>WRITE(102,1025)</td>
</tr>
<tr>
<td>0110</td>
<td>WRITE(102,1050)</td>
</tr>
<tr>
<td>0111</td>
<td>WRITE(102,1060)</td>
</tr>
<tr>
<td>0112</td>
<td>$RX = RXZERO(L)$</td>
</tr>
<tr>
<td>0113</td>
<td>$VX = VXZERO(L)$</td>
</tr>
<tr>
<td>0114</td>
<td>$EPSCHK = EPSCHV$</td>
</tr>
<tr>
<td>0115</td>
<td>CALL BETTER($K, RXZERO, VXZERO, RXVX, NAtRA, VARCOEF, VCOEF, N, X, Y, RB, \ V8)$</td>
</tr>
<tr>
<td>0116</td>
<td>IF $(K.EQ.0)$ GO TO 170</td>
</tr>
<tr>
<td>0117</td>
<td>WRITE(102,1065)</td>
</tr>
<tr>
<td>0118</td>
<td>WRITE(102,1050)</td>
</tr>
<tr>
<td>0119</td>
<td>WRITE(102,1065)</td>
</tr>
<tr>
<td>0120</td>
<td>$K = K + 1$</td>
</tr>
<tr>
<td>0121</td>
<td>IF $(ROOT = 8)$</td>
</tr>
<tr>
<td>0122</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0123</td>
<td>WRITE(102,1075)</td>
</tr>
<tr>
<td>0124</td>
<td>WRITE(102,1080)</td>
</tr>
<tr>
<td>0125</td>
<td>IF $(IROOT .LT. K)$</td>
</tr>
<tr>
<td>0126</td>
<td>WRITE(102,1090)</td>
</tr>
<tr>
<td>0127</td>
<td>WRITE(102,1095)</td>
</tr>
<tr>
<td>0128</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0129</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0130</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0131</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0132</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0133</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0134</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0135</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0136</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0137</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0138</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0139</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0140</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0141</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0142</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0143</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0144</td>
<td>WRITE(102,1070)</td>
</tr>
<tr>
<td>0145</td>
<td>WRITE(102,1070)</td>
</tr>
</tbody>
</table>

128
TABLE B. VIII (Continued)

```plaintext
0001
SUBROUTINE GENAPP(APP1,APP2,NAPP,XSTART)

******************************************************************************
* SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
* DEGREE OF THE ORIGINAL POLYNOMIAL.                                        *
******************************************************************************

DOUBLE PRECISION APP1,APP2,XSTART,DUMMY,BETA
DIMENSION APP1(25),APP2(25)
COMMUN DUMMY,MAX,102
IF (XSTART.EQ.0.0) XSTART=0.5
BETA=0.2617994
DO 10 =1,NAPP
  APP1(I)=XSTART*DCOS(BETA)
  APP2(I)=XSTART*DSIN(BETA)
  BETA=BETA+0.5235988
10 XSTART=XSTART+0.5
RETURN
END

0001
SUBROUTINE ALTEN(XOLDI,KOLDI,NALTER,TIME)

******************************************************************************
* SUBROUTINE ALTEN ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO *
* CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
******************************************************************************

DOUBLE PRECISION XOLDI,KOLDI,DUMMY,ABXOLDI,BETA
COMMON DUMMY,MAX,102
IF(TIME.NE.0) GO TO 5
TIME =1
WRITE(I02,1010) MAX
5 IF(NALTER.EQ.0) GO TO 10
WRITE(I02,1000) XOLDI,KOLDI
GO TO 20
10 ABXOLDI=DSORT(XOLDI,XOLDI+XOLDI*XOLDI)
BETA=DATAZ2(XOLDI,1,KOLDI)
WRITE(I02,1020) XOLDI,1,KOLDI
20 NALTER=NALTER+1
IF(NALTER.GT.5) RETURN
GO TO 30
30 XOLDI=KOLDI
XOLDI=KOLDI
GO TO 50
40 BETA=BETA+1.0471976
40 BETA=BETA+KOLDI
ABXORD=ABXOLD+DCOS(BETA)
KOLDI=KOLDI+DSIN(BETA)
50 RETURN
1000 FORMAT(1X,0.23,15H + ,0.23,16,2H 1,x10.2B1MALTED APPROXIMATION)
1010 FORMAT(1X,5.4HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF *
ITER ,13,12H ITERATIONS//)
1020 FORMAT(1X,0.23,15H + ,0.23,16,2H 1,x10.2B1INITIAL APPROXIMATION)
END

```

TABLE B. VIII (Continued)

SUBROUTINE QUAD(UA, VA, UROOT, VROOT, MULTI, EPST)

* SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES
* OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE
* QUADRATIC IS DONE USING THE QUADRATIC FORMULA.

DOUBLE PRECISION UA(26), VA(26), UROOT(251), VROOT(251), MULTI(251)
DIMENSION U(26), VA(26), UROOT(251), VROOT(251), MULTI(251)

IF(INA.EQ.2) GO TO 7
IF(INA.EQ.1) GO TO 5

UROOT(INROOT+1) = 0.0
VROOT(INROOT+1) = 0.0
MULTI(INROOT+1) = 1
NROOT = NROOT + 1
GO TO 50

888 = UA(1) * UA(1) - VA(1) * VA(1)

UROOT(INROOT+1) = (-UA(1) * UA(1) - VA(2)) / (8 * 888)
VROOT(INROOT+1) = (-VA(1) * VA(1)) / (8 * 888)
MULTI(INROOT+1) = 1
MULTI(INROOT+2) = 1
NROOT = NROOT + 2
GO TO 50

UDISC = (UA(2) * UA(2) - VA(2) * VA(2)) - (4.0 * (UA(1) * UA(1) - VA(1) * VA(1)))

BBB = DSQRT(UDISC * UDISC + VDISC * VDISC)

IF(BBB.LT.EPST) GO TO 10

CALL CONSQT(UDISC, VDISC, UDISC, VDISC)

UROOT(INROOT+1) = (-UA(1) * UA(1) - VA(2)) / (8 * 888)
VROOT(INROOT+1) = (-VA(1) * VA(1)) / (8 * 888)
MULTI(INROOT+1) = 2
NROOT = NROOT + 1

50 RETURN
END
TABLE B. VIII (Continued)

0001 SUBROUTINE COMSQTIUXtVXiUYsVYI
0002 *****************************************************************
0003 * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER. *
0004 *****************************************************************
0005 DOUBLE PRECISION UX, VX, VX, VY, DUMMY, R, AAA, BBB
0006 R = DSQRTIUX*UX + VX*VX)
0007 AAA = DSQRT(DABS((IRUXI/2.01
0008 BBB = DSQRT(DABS((R- UX/2.0))
0009 IF(VX) 10, 20, 30
0010 10 UY = AAA
0011 20 IF(UX) 40, 50, 60
0012 30 UY = AAA
0013 40 DUMMY = DABS(UX)
0014 50 UY = 0.0
0015 60 UY = DSQRT(DUMMY)
0016 70 GO TO 100
0017 80 GO TO 100
0018 90 GO TO 100
0019 100 RETURN
0020 END
TABLE B. VIII (Continued)

SUBROUTINE HORNER(RA, VA, RXO, VXO, NDEF, RB, VB, RC, VC, RPX, VPX, RDPX, VDPX)

**HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL PI(X) AT A POINT D AND ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).**

DOUBLE PRECISION VDPX, RXO, VXO, RB, VB, RC, VC, RPX, VPX, RDPX, RA, VA

DIMENSION RA(26), VA(26), RB(26), VB(26), RC(26), VC(26)

II = RA(1)

VB(1) = VA(1)

NUM = NDEF + 1

DO 10 I = 2, NUM

RB(I) = RA(I) + (RB(I-1) * RXD - VB(I-1) * VXO)

VB(I) = VA(I) + (VB(I) * RXO + RB(I-1) * VXO)

RPX = RB(NUM)

VPX = VB(NUM)

RC = RB(1)

VC = VB()

IF (NDEF .LT. 2) GO TO 25

DO 20 J = 2, NDEF

RC(J) = RB(J) + (RC(J-1) * RXO - VC(J-1) * VXO)

VC(J) = VB(J) + (VC(J) * RXO + RC(J-1) * VXO)

RDPX = RC(NDEF)

VPX = VC(NDEF)

RETURN

END

SUBROUTINE NEWTON(RPX, VPX, RDPX, VDPX, RXO, VXO, RXNEW, VXNEW)

**THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROXIMATION BY USING THE ITERATION FORMULA**

\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \]

DOUBLE PRECISION RXO, VXO, RXNEW, VXNEW, ARG

DO 3 DO 2 DO 1

RETURN

END
TABLE B. VIII (Continued)

SUBROUTINE CHECK(RPX, VPX, RDPX, VDPX, RXO, VXO, KANS)

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

* THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
* MATIONS BY TESTING THE EXPRESSION
* ABSOLUTE VALUE OF (P(X(N))/P'(X(N)))/ABSOLUTE VALUE OF X(N+1).
* WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.

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

DOUBLE PRECISION RPX, VPX, RDPX, VDPX, RXO, VXO, ABSXO, ABSQUO, RDUMMY, YDU

DOUBLE PRECISION EPS

COMMON EPS, MAX, O02

ABSXO=DSQRT(RXO*RXO+VXO*VXO)

IF(ABSXO.EQ.0.) GO TO 25

ARG=RPX*RPX+VDPX*VDPX

DOD=DSQRT(ARG)

IF(DOD.EQ.0.01 GO TO 25

RDUMMY=(RPX*RDPX+VPX*VDPXI/ARG

VDUMMY=(VPX*RDPX-RPX*VOPX)/ARG

ABSQUO=DSQRT(RDUMMY*RDUMMY+VDUMMY*VDUMMY)

IF(ABSQUO/ABSXO.LT.EPS) GO TO 10

KANS=0

RETURN

10 KANS=1

RETURN

25 KANS=0

RETURN

END
TABLE B. VIII (Continued)

0001 SUBROUTINE BETTER(KtRXZERO, VXZERO, RX, VX, RA, VA, RCOEF, VCOEF, N, RC, RCV, RB, VB)

***************************************************************************
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.
***************************************************************************

0002 DOUBLE PRECISION RXZERO, VXZERO, RX, VX, RA, VA, RCOEF, VCOEF, RC, RCV, RB, VB

0003 DIMENSION RXZERO(25), VXZERO(25), RX(25), VX(25), RA(26), VA(26), RCOEF(126), VCOEF(26), RC(26), RCV(26), RB(26), VB(26)

0004 DOUBLE PRECISION A, B, D, P, X, RXNEW, VXNEW

0005 COMMON EPS, MAX, IO2

0006 DO 10 I=I, K

0007 RXZERO(I)=RX(I)

0008 VXZERO(I)=VX(I)

0009 DO 20 I=1, N

0010 RCOEF(I)=RA(I)

0011 VCOEF(I)=VA(I)

0012 DO 50 J=1, K

0013 RXO=RXZERO(J)

0014 VXO=VXZERO(J)

0015 NN=N

0016 ITER=0

0017 30 CALL HORNER(RCOEF, VCOEF, RX, VX, NN, RB, VB, RC, VX, VX, RX, VX, RX, VX)

0018 ABPX=DSQRT(RPX*RPX+VPX*VPX)

0019 ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)

0020 IF(ABOPX.NE.0.0) GO TO 33

0021 IF(ABDPX.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

0025 RXO=RXNEW

0026 VXO=VXNEW

0027 CALL CHECK(RPX, VPX, RDPX, VDPX, RXO, VXO, RXNEW, VXNEW)

0028 IF(KANS.EQ.1) GO TO 40

0029 IF(ITER.GE.MAX) GO TO 35

0030 GO TO 30

0031 34 WRITE(102, 1112) RXO, VXO

0032 35 WRITE(102, 100) J, RXZERO(J), VXZERO(J)

0033 WRITE(102, 200) RXO, VXO

0034 40 RXO=RXO

0035 VXO=VXO

0036 50 CONTINUE

0037 RETURN

0038 1112 FORMAT(1H13H THE VALUE OF THE DERIVATIVE AT XO = , D23, 14, 3H + , D23, 14, 3H + , D23

0039 100 FORMAT(42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(1, 12, 4H = , D23

0040 200 FORMAT(13H THE PRESENT APPROXIMATION AFTER , 13, 29H ITERRATIONS IS P

0041 END
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DABS(d)</td>
<td>obtain absolute value</td>
</tr>
<tr>
<td>DATAN2(d1,d2)</td>
<td>arctangent of d1/d2</td>
</tr>
<tr>
<td>DSQRT(d)</td>
<td>square root</td>
</tr>
<tr>
<td>DCOS(d)</td>
<td>cosine of angle</td>
</tr>
<tr>
<td>DSIN(d)</td>
<td>sine of angle</td>
</tr>
<tr>
<td>DSQRT(d)</td>
<td>square root</td>
</tr>
</tbody>
</table>

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, X0, is given for each root. The other two required by Muller's method are constructed within the program and are .9X0 and 1.1X0.

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

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

4. Description of Program Output

The output from Muller's method is the same as that for Newton's method as described in Appendix 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, \( \text{ROOT}(X) = YYY \) 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>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Single Precision</th>
<th>Type</th>
<th>Double Precision</th>
<th>Type</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>I</td>
<td>NP</td>
<td>I</td>
<td>NP</td>
<td>I</td>
<td>Main Program</td>
<td>Degree of polynomial ( P(X) )</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
<td>I</td>
<td></td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>NOMULT</td>
<td>I</td>
<td>NOMULT</td>
<td>I</td>
<td>NOMULT</td>
<td>I</td>
<td></td>
<td>Number of roots (counting multiplicities)</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT, VROOT</td>
<td>D</td>
<td>UROOT, VROOT</td>
<td>D</td>
<td></td>
<td>Array containing the roots</td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
<td></td>
<td>Number of initial approximations to be read in</td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>UAPP, VAPP</td>
<td>D</td>
<td>UAPP, VAPP</td>
<td>D</td>
<td></td>
<td>Array of initial approximations</td>
</tr>
<tr>
<td>WORK</td>
<td>C</td>
<td>UWORK, VWORK</td>
<td>D</td>
<td>UWORK, VWORK</td>
<td>D</td>
<td></td>
<td>Working array containing coefficients of current polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB, VB</td>
<td>D</td>
<td>UB, VB</td>
<td>D</td>
<td></td>
<td>Array containing coefficients of deflated polynomial</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA, VA</td>
<td>D</td>
<td>UA, VA</td>
<td>D</td>
<td></td>
<td>Array containing coefficients of original polynomial, ( P(X) )</td>
</tr>
<tr>
<td>RAPP</td>
<td>C</td>
<td>URAPP, VRAPP</td>
<td>D</td>
<td>URAPP, VRAPP</td>
<td>D</td>
<td></td>
<td>Array of initial or altered approximations for which convergence was obtained</td>
</tr>
<tr>
<td>X1</td>
<td>C</td>
<td>UX1, VX1</td>
<td>D</td>
<td>UX1, VX1</td>
<td>D</td>
<td></td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>UX2, VX2</td>
<td>D</td>
<td>UX2, VX2</td>
<td>D</td>
<td></td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3, VX3</td>
<td>D</td>
<td>UX3, VX3</td>
<td>D</td>
<td></td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>PX1</td>
<td>C</td>
<td>UPX1, VPX1</td>
<td>D</td>
<td>UPX1, VPX1</td>
<td>D</td>
<td></td>
<td>Value of polynomial ( P(X) ) at ( X_1 )</td>
</tr>
<tr>
<td>PX2</td>
<td>C</td>
<td>UPX2, VPX2</td>
<td>D</td>
<td>UPX2, VPX2</td>
<td>D</td>
<td></td>
<td>Value of polynomial ( P(X) ) at ( X_2 )</td>
</tr>
<tr>
<td>PX3</td>
<td>C</td>
<td>UPX3, VPX3</td>
<td>D</td>
<td>UPX3, VPX3</td>
<td>D</td>
<td></td>
<td>Value of polynomial ( P(X) ) at ( X_3 )</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4, VX4</td>
<td>D</td>
<td>UX4, VX4</td>
<td>D</td>
<td></td>
<td>Newest approximation ( (X_{n+1}) ) to root</td>
</tr>
<tr>
<td>PX4</td>
<td>C</td>
<td>UPX4, VPX4</td>
<td>D</td>
<td>UPX4, VPX4</td>
<td>D</td>
<td></td>
<td>Value of polynomial ( P(X) ) at ( X_4 )</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td></td>
<td>Array containing the multiplicities of each root found</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td></td>
<td>Counter for iterations</td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
<td></td>
<td>Unit number of input device</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td></td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Single Precision Type</td>
<td>Double Precision Variable</td>
<td>Double Precision Type</td>
<td>Disposition of Argument</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------</td>
<td>---------------------------</td>
<td>-----------------------</td>
<td>-------------------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPSRT</td>
<td>R</td>
<td>EPSRT</td>
<td>D</td>
<td></td>
<td>Number used in subroutine BETTER to generate two approximations from the one given</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOPOLY</td>
<td>I</td>
<td>NOPOLY</td>
<td>I</td>
<td></td>
<td>Number of the polynomial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td></td>
<td>Maximum number of iterations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
<td></td>
<td>Tolerance check for convergence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
<td>D</td>
<td></td>
<td>Tolerance check for zero (0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPSM</td>
<td>R</td>
<td>EPSM</td>
<td>D</td>
<td></td>
<td>Tolerance check for multiplicities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KCHECK</td>
<td>I</td>
<td>KCHECK</td>
<td>I</td>
<td></td>
<td>Program control, KCHECK = 1 stops execution of program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSTART</td>
<td>R</td>
<td>XSTART</td>
<td>D</td>
<td></td>
<td>Magnitude at which to start generating initial approximations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEND</td>
<td>R</td>
<td>XEND</td>
<td>D</td>
<td></td>
<td>Magnitude at which to end generating initial approximations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NWORK</td>
<td>I</td>
<td>NWORK</td>
<td>I</td>
<td></td>
<td>Degree of current deflated polynomial whose coefficients are in WORK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
<td></td>
<td>Program control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NALTER</td>
<td>I</td>
<td>NALTER</td>
<td>I</td>
<td></td>
<td>Number of alterations which have been performed on an initial approximation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IAPP</td>
<td>I</td>
<td>IAPP</td>
<td>I</td>
<td></td>
<td>Counter for number of initial approximations used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td></td>
<td>When CONV is true, convergence has been obtained</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</td>
<td>I</td>
<td></td>
<td>Number of distinct roots solved by Muller's method, i.e. not solved directly by subroutine QUAD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subroutine HORNER

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>UA,VA</th>
<th>D</th>
<th>E</th>
<th>Array of current polynomial coefficients (to be deflated or evaluated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>I</td>
<td>NA</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial to be deflated or evaluated</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>UX,VX</td>
<td>D</td>
<td>E</td>
<td>Approximation at which to evaluate or deflate the polynomial</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Double Precision Variable</td>
<td>Type</td>
<td>Disposition of Argument</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>---------------------------</td>
<td>------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB, VB</td>
<td>D</td>
<td>R</td>
<td>Array containing the coefficients of the deflated polynomial</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>UPX, VPX</td>
<td>D</td>
<td>R</td>
<td>Value of the polynomial at X</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
<td>I</td>
<td>R</td>
<td>Number of coefficients of polynomial to be deflated</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3, VX3</td>
<td>D</td>
<td>E</td>
<td>Approximation to Root (old) (Xₙ)</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4, VX4</td>
<td>D</td>
<td>E</td>
<td>New approximation to root (Xₙ₊₁)</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td>R</td>
<td>CONV = 'true implies convergence has been obtained</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
<td>C</td>
<td>Tolerance for convergence test</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
<td>D</td>
<td>C</td>
<td>Tolerance check for zero (0)</td>
</tr>
<tr>
<td>DENOM</td>
<td>R</td>
<td>DENOM</td>
<td>D</td>
<td></td>
<td>Magnitude of new approximation, (Xₙ₊₁)</td>
</tr>
</tbody>
</table>

Subroutine TEST

| 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)

<table>
<thead>
<tr>
<th>Single Precision</th>
<th>Variable</th>
<th>Type</th>
<th>Double Precision</th>
<th>Variable</th>
<th>Type</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td>C</td>
<td></td>
<td>UB, VB</td>
<td>D</td>
<td></td>
<td>Array containing coefficients of deflated polynomial</td>
</tr>
<tr>
<td></td>
<td>X1</td>
<td>C</td>
<td></td>
<td>UX1, VX1</td>
<td>D</td>
<td></td>
<td>One of three approximations to the root</td>
</tr>
<tr>
<td></td>
<td>X2</td>
<td>C</td>
<td></td>
<td>UX2, VX2</td>
<td>D</td>
<td></td>
<td>One of three approximations to the root</td>
</tr>
<tr>
<td></td>
<td>X3</td>
<td>C</td>
<td></td>
<td>UX3, VX3</td>
<td>D</td>
<td></td>
<td>One of three approximations to the root</td>
</tr>
<tr>
<td></td>
<td>PX1</td>
<td>C</td>
<td></td>
<td>UPX1, VPX1</td>
<td>D</td>
<td></td>
<td>Value of polynomial (P(X)) at X1</td>
</tr>
<tr>
<td></td>
<td>PX2</td>
<td>C</td>
<td></td>
<td>UPX2, VPX2</td>
<td>D</td>
<td></td>
<td>Value of polynomial (P(X)) at X2</td>
</tr>
<tr>
<td></td>
<td>PX3</td>
<td>C</td>
<td></td>
<td>UPX3, VPX3</td>
<td>D</td>
<td></td>
<td>Value of polynomial (P(X)) at X3</td>
</tr>
<tr>
<td></td>
<td>CONV</td>
<td>L</td>
<td></td>
<td>CONV</td>
<td>L</td>
<td></td>
<td>CONV = true implies convergence has been obtained</td>
</tr>
<tr>
<td></td>
<td>X4</td>
<td>C</td>
<td></td>
<td>UX4, VX4</td>
<td>D</td>
<td></td>
<td>Newest approximation to root</td>
</tr>
<tr>
<td></td>
<td>J</td>
<td>I</td>
<td></td>
<td>J</td>
<td>I</td>
<td></td>
<td>Program control – counts the number of roots used as initial approximations</td>
</tr>
<tr>
<td></td>
<td>MAX</td>
<td>I</td>
<td></td>
<td>MAX</td>
<td>I</td>
<td>C</td>
<td>Maximum number of iterations permitted</td>
</tr>
<tr>
<td></td>
<td>IO2</td>
<td>I</td>
<td></td>
<td>IO2</td>
<td>I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
</tbody>
</table>

Subroutine ALTER

|                      | X1       | C | XLR, XLI | 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 |     | Imaginary 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.
Figure C.1. Flow Charts for Muller's Method
Figure C.1. (Continued)
Figure C.17 (Continued)
Figure C.1. (Continued)
Figure C.1. (Continued)
Figure C.1: (Continued)
TABLE C.V.

PROGRAM FOR MULLER'S METHOD

C
C* DOUBLE PRECISION PROGRAM FOR MULLER'S METHOD
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 ZEROS OF THE QUADRATIC
C* CLOSEST TO THE OLD APPROXIMATIONS IS TAKEN AS THE NEW APPROXIMATION.
C* IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
C
C**************************************************************************

**D**OUBLE PRECISION UPX3,UPX3,UPX2,UPX2,VRDQ,VRDQ,UK1,UK1,UPAPP,UPAPP
1,UK2,UK2,UKR,UKR,UKX3,UKX3,UX3,UX3,UX3,UX3,UX3,UX3,UX3,UX3,UX3,UX3,UX3
2APP,UK4,UPX4,UPX4,UPX4,EPSEP,EPSEP,EPSEP,EPSEP,EPSEP,EPSEP,EPSEP,EPSEP
3,0,00,KSTART,XEND
**D**IMENSION VRDQ(251),VRDQ(251),MULT(251),UPAPP(25,3),UPAPP(25,3),UKR
1(261),UKR(261),UKR(261),UKR(261),UKR(261),UKR(261),UKR(261),UKR(261),UKR(261),UKR(261)
**D**ATA PNAME,DNAME,2P1,2P1,10/1/
**D**0005 COMMON EPSRT,EPSQ,EPS,102,MAX
**D**0006 101=5
**D**0007 102=6
**D**0008 EPSRT=0,999
**D**0009 10**N**GCD=0
**D**0100 URQD=0
**D**0110 [PATH]=1
**D**0120 NREQUEST=0
**D**0130 NALTER=0
**D**0140 NTIME=0
**D**0150 IAPP=1
**D**0160 ITER=1
**D**0170 READ(10),1000) NPOLY,NP,NAPP,MAX,EPS,EPS,EPS,XSTART,XEND,KCHECK
**D**0180 IF(KCHECK.EQ.11) STOP
**D**0190 KK=NPP=1
**D**0200 READ(10),1010) (UAI(1),VA(1),I=1,KKK)
**D**0210 WRITE(102,1020) NPoLY,NP
**D**0220 WRITE(102,1035) PNAME,1,UA(1),VA(1),I=1,1;KK)
**D**0230 WRITE(102,2060)
**D**0240 WRITE(102,2080) NAPP
**D**0250 WRITE(102,2010) MAX
**D**0260 WRITE(102,2020) EPS
**D**0270 WRITE(102,2030) EPSQ
**D**0280 WRITE(102,2040) XSTART
**D**0290 WRITE(102,2050) XEND
**D**0300 IF(NP,GT,2) GO TO 15
**D**0310 CALL QUAD(UA,YA,NP,VRDQ,VRDQ,NROOT,MULT,EPSQ)
**D**0320 WRITE(102,1037)
**D**0330 WRITE(102,1096) (1,VRDQ(I),VRDQ(I),MULT(I),I=1,NROOT)
**D**0340 GO TO 10
**D**0350 15 IF(NAPP,NE,0) GO TO 20
**D**0360 NAPP=NPP
**D**0370 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
**D**0380 GO TO 27
**D**0390 20 READ(101,1030) (UPAPP(I),2),VAPP(I),2,1=1,NAPP)
**D**0400 GO 25 I=1,NAPP
**D**0410 UAPP(I)=UAPP(I,2)
**D**0420 VAPP(I)=VAPP(I,2)
<table>
<thead>
<tr>
<th>Line No.</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0043</td>
<td>UAPP(I,3) = 1.1 * UAPP(I,2)</td>
</tr>
<tr>
<td>0044</td>
<td>VAPP(I,3) = 1.1 * VAPP(I,2)</td>
</tr>
<tr>
<td>0045</td>
<td>KKK = NP + 1</td>
</tr>
<tr>
<td>0046</td>
<td>DO 30 I = 1, KKK</td>
</tr>
<tr>
<td>0047</td>
<td>UWORK(I) = UX(I)</td>
</tr>
<tr>
<td>0048</td>
<td>VWORK(I) = VX(I)</td>
</tr>
<tr>
<td>0049</td>
<td>NWORK = NP</td>
</tr>
<tr>
<td>0050</td>
<td>40 UX = UAPP(IAPP, I)</td>
</tr>
<tr>
<td>0051</td>
<td>VX = VAPP(IAPP, I)</td>
</tr>
<tr>
<td>0052</td>
<td>UX2 = UAPP(IAPP, 2)</td>
</tr>
<tr>
<td>0053</td>
<td>VX2 = VAPP(IAPP, 2)</td>
</tr>
<tr>
<td>0054</td>
<td>UX3 = UAPP(IAPP, 3)</td>
</tr>
<tr>
<td>0055</td>
<td>VX3 = VAPP(IAPP, 3)</td>
</tr>
<tr>
<td>0056</td>
<td>CALL HORNER(NWORK, UWORK, VWORK, UX1, VX1, UB1, VB1, UPX1, VPX1)</td>
</tr>
<tr>
<td>0057</td>
<td>CALL HORNER(NWORK, UWORK, VWORK, UX2, VX2, UB2, VB2, UPX2, VPX2)</td>
</tr>
<tr>
<td>0058</td>
<td>CALL HORNER(NWORK, UWORK, VWORK, UX3, VX3, UB3, VB3, UPX3, VPX3)</td>
</tr>
<tr>
<td>0059</td>
<td>50 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UX1, VX1, UX2, VX2, UX3, VX3)</td>
</tr>
<tr>
<td>0060</td>
<td>60 CALL HORNER(NWORK, UWORK, VWORK, UX4, VX4, UB4, VB4, UPX4, VPX4)</td>
</tr>
<tr>
<td>0061</td>
<td>AXP4 = DSQRT(UPX4 * UPX4 + VPX4 * VPX4)</td>
</tr>
<tr>
<td>0062</td>
<td>AXP3 = DSQRT(UPX3 * UPX3 + VPX3 * VPX3)</td>
</tr>
<tr>
<td>0063</td>
<td>IF(ABPX3 .LE. 0.1) GO TO 70</td>
</tr>
<tr>
<td>0064</td>
<td>UB4 = 0.5 * UB4</td>
</tr>
<tr>
<td>0065</td>
<td>VX4 = VX3 + (VX3 * UB4 - UB3 * VX4)</td>
</tr>
<tr>
<td>0066</td>
<td>UX4 = UX3</td>
</tr>
<tr>
<td>0067</td>
<td>VX4 = VX3</td>
</tr>
<tr>
<td>0068</td>
<td>CALL TEST(UX3, VX3, UX4, VX4, CONV)</td>
</tr>
<tr>
<td>0069</td>
<td>IF(CONV) GO TO 120</td>
</tr>
<tr>
<td>0070</td>
<td>IF(ITER .LT. MAX) GO TO 110</td>
</tr>
<tr>
<td>0071</td>
<td>CALL ALTER(UAPP(IAPP, I), VAPP(IAPP, I), UAPP(IAPP, 2), VAPP(IAPP, 2), UAPP(IAPP, 3), VAPP(IAPP, 3), NALTER, NTIME)</td>
</tr>
<tr>
<td>0072</td>
<td>IF(NALTER .GT. 5) GO TO 75</td>
</tr>
<tr>
<td>0073</td>
<td>ITER = 1</td>
</tr>
<tr>
<td>0074</td>
<td>GO TO 40</td>
</tr>
<tr>
<td>0075</td>
<td>IF(IAPP .LT. NAPP) GO TO 100</td>
</tr>
<tr>
<td>0076</td>
<td>IF(IXEND .EQ. 0) GO TO 77</td>
</tr>
<tr>
<td>0077</td>
<td>IF(IDRIFT .EQ. 0) GO TO 77</td>
</tr>
<tr>
<td>0078</td>
<td>NAPP = NP</td>
</tr>
<tr>
<td>0079</td>
<td>CALL GENAPP(UAPP, VAPP, NAPP, XSTART)</td>
</tr>
<tr>
<td>0080</td>
<td>IF(IDRIFT .EQ. 0) GO TO 77</td>
</tr>
<tr>
<td>0081</td>
<td>IF(IXEND .EQ. 0) GO TO 77</td>
</tr>
<tr>
<td>0082</td>
<td>IF(IDRIFT .EQ. 0) GO TO 77</td>
</tr>
<tr>
<td>0083</td>
<td>IF(IDRIFT .EQ. 0) GO TO 77</td>
</tr>
<tr>
<td>0084</td>
<td>IF(IDRIFT .EQ. 0) GO TO 77</td>
</tr>
<tr>
<td>0085</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0086</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0087</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0088</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0089</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0090</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0091</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0092</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0093</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0094</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0095</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0096</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0097</td>
<td>WRITE(IO2, 1090)</td>
</tr>
<tr>
<td>0098</td>
<td>WRITE(IO2, 1090)</td>
</tr>
</tbody>
</table>

**TABLE C.V (Continued)**
TABLE C.V. (Continued)

0099 IF(IROOT.LT.NROOT) GO TO 85
0100 GO TO 87
0101 85 KKK=IROOT+1
0102 WRITE(102,1086)(I,URROOT(I),VRROOT(I),MULT(I),I=KKK,NROOT)
0103 87 IF(IPATH.EQ.1) GO TO 81
0104 GO TO 10
0105 90 WRITE(102,1070) 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 URROOT(IROOT)=UX4
0116 VRROOT(IROOT)=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(NMULT.LT.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=K
0128 GO TO 140
0129 UWORK(I)=UB(I)
0130 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.EPS) GO TO 150
0134 IF(NWORK.GT.2) GO TO 75
0135 IROOT=NROOT
0136 CALL QUAD(UWORK,VWORK,NWORK,UROOT,VROOT,EPS)
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(2030,0)
0156 1020 FORMAT(1X,52*MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL)
TABLE C.V (Continued)

<table>
<thead>
<tr>
<th>POLYNOMIAL NUMBER</th>
<th>DEGREE</th>
<th>THE COEFFICIENTS OF PI(X) ARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>12/11</td>
<td>12/11</td>
<td>12/11</td>
</tr>
<tr>
<td>11A/1H + 1X,18H</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

THE COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

<table>
<thead>
<tr>
<th>ROOTS OF P(X), MULTIPLICITIES, INITIAL APPROXIMATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER 12</td>
</tr>
<tr>
<td>1023.16, 1H + 023.16, 1H</td>
</tr>
</tbody>
</table>

BEFORE ATTEMPT TO IMPROVE ACCURACY

AFTER THE ATTEMPT TO IMPROVE ACCURACY

NUMBER OF INITIAL APPROXIMATIONS GIVEN

MAXIMUM NUMBER OF ITERATIONS

TEST FOR CONVERGENCE

RADIUS TO START SEARCH

RADIUS TO END SEARCH

END
TABLE C.V (Continued)

SUBROUTINE ALTER(X1R,X1I,X2R,X21,X3R,X31,NALTER,ITIME)

**************************************************************************
*
** SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO
** CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
******************************************************************************

DOUBLE PRECISION XIR,X1I,X2R,X21,X3R,X31,EPSI,EPS2,EPS3,R,BETA

COMMON EPSIEPS2*EPS3, MAX

IFIITIME.NE.0) GO TO 5

ITIME=1

WRITE(10,1100)

R=DSQRTIX2R*X2R+X2*X21

BETA=DATAN2(X2IX2RI

WRITE(1021,1020)

XIR,X1I,X2R,X21,X3R,X31

NALTER=NALTER+1

IFINALTER.GT.5) RETURN

X2R=-X2R

X21=-X21

GO TO 50

BETA=BETA+1.0471976

X2R=R*DCOS(BETA)

X21=R*DSIN(BETA)

X1R=0.9*X2R

X11=0.9*X21

X3R=1.1*X2R

X31=1.1*X21

RETURN

1000 FORMAT(1X,5HINITIAL APPROXIMATIONS/1X,5HALTERED APPROXIMATIONS/1X,5HITERATIONS/1X,5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER 13,12H ITERATIONS.//)

1010 FORMAT(/1X,5HHALTERED APPROXIMATIONS/1X,5HINITIAL APPROXIMATIONS/1X,2HHALTERED APPROXIMATIONS/1X,2HINITIAL APPROXIMATIONS/1X,5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER 13,12H ITERATIONS.//)

END
SUBROUTINE GENAPP(APPR,APP1,NAPP,XSTART)

***SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE***

DEGREE OF THE ORIGINAL POLYNOMIAL.

DOUBLE PRECISION APPR,APP1,XSTART,EPS1,EPS2,EPS3,BETA

DIMENSION APPR(25,3),APP1(25,3)

COMMON EPS1,EPS2,EPS3,IO2,MAX

IF(XSTART.EQ.0.0) XSTART=0.5

BETA=0.2617994

DO 10 I=1,NAPP

APPR(I,1)=XSTART*COS(BETA)

APP1(I,1)=XSTART*SIN(BETA)

BETA=BETA+0.5235988

10 XSTART=XSTART+0.5

DO 20 I=1,NAPP

APPR(I,1)=0.9*APPR(I,2)

APP1(I,1)=0.9*APP1(I,2)

20 APPR(I,3)=1.1*APPR(I,2)

APP1(I,3)=1.1*APP1(I,2)

RETURN

END
TABLE C.V (Continued)

0001 SUBROUTINE BETTER,VA,WA,UROOT,VROOT,NROOT,URAPP,VRAPP,IROOT,MUL

0002 ************************************************************************************
0003 SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND
0004 BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO
0005 THE FULL, UNDEFLATED POLYNOMIAL.

0006 ************************************************************************************

0007 DOUBLE PRECISION UROOT,VROOT,VA,VA,UBAPP,VBAPP,UX1,VX1,UX2,VX2,UX3
0008 VX3,UX1,VX1,UX2,VX2,UX3,VX3,UX3,UX3,UX3,VX3
0009 LOGICAL CONV

0010 DIMENSION UROOT(25),VROOT(25),VA(26),UA(26),UBAPP(25,3),VRAPP(25,3)
0011 UX4,VX4,UX4,VX4
0012 COMMON EPSRT,EPSOEPSI02,NMAX

0013 IF(NROOT.LE.1) RETURN

0014 L=0
0015 DO 10 I=1,NROOT

0016 UXI=UBAPP(I,1)
0017 VX1=VBAPP(I,1)
0018 UX2=UBAPP(I,2)
0019 VX2=VBAPP(I,2)
0020 UX3=UBAPP(I,3)
0021 VX3=VBAPP(I,3)
0022 ITER=I
0023 CALL HORNER(NPUAVAUX1,VX1IUBVB.UPX1VPX1I
0024 CALL HORIZ(NPUAVX1,VX2,UX2,VX2,UX2,VX2)
0025 CALL HORIZ(NPUAVX3,VX3,UX3,UBVBUPX3,VPX3)
0026 CALL CALCD(UX1,VX1,UX2,VX2,UX3,VX3,UX3,UX)
0027 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0028 IF(CONV) GO TO 50
0029 IF(ITER.LT.MAX) GO TO 40
0030 WRITE(10,2001) I,UROOT(I),VROOT(I)
0031 WRITE(10,2003) UX4,VX4
0032 IF(J.LT.IROOT) GO TO 33
0033 IF(J.EQ.IROOT) GO TO 35
0034 GO TO 100
0035 33 K=IROOT-1
0036 DO 3A K=1,K

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 VRAPP(K,3)=VRAPP(K+1,3)
0043 IROOT=IROOT-1
0044 GO TO 100
0045 3A UXI=UX2
TABLE C.V (Continued)

0044     VX1=VX2
0047     UX2=UX3
0049     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
0058     L=L+1
0059     UROOTS(L)=UX4
0060     VROOTS(L)=VX4
0060     MULT(L)=MULT(I)
0061
100 CONTINUE

0062     IF(L.EQ.0) GO TO 120
0063     DO 110 I=1,L
0064     UROOT(I)=UROOTS(I)
0065     VROOT(I)=VROOTS(I)
0066     MULT(I)=L
0067   110 RETURN
0068
120 NROOT=0
0069 RETURN
0070
1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(I,124H) = ,
1023.16,3H + ,023.16,2H 1/24H DID NOT CONVERGE AFTER ,13.11H ITERATIONS)
0071
1010 FORMAT(///3OH THE PRESENT APPROXIMATION IS ,D23.16,3H + ,D23.16,2H 1/
17)
0072 END
### TABLE C.V (Continued)

| 0001 | SUBROUTINE CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, V |

**------------------------------------------------------------------------**
* GVEN THREE APPROXIMATIONS XIN-2, XIN-1, AND XIN, SUBROUTINE CALC
* APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF
* THE QUADRATIC CLOSEST TO X(IN). THIS ZERO IS THE NEW APPROXIMATION
* XIN+1 TO THE ZERO OF THE POLYNOMIAL.

**------------------------------------------------------------------------**

| 0002 | DOUBLE PRECISION ARG1, ARG2 |

| 0003 | DOUBLE PRECISION UPX3, VPX3, UX3, VX3, UPX2, VPX2, UX2, VX2, UPX1, VPX1, |

| 0004 | UH3, VH3, VH2, VH1, UQ3, VQ3, UD0, VD0, UD00, VD00, UD000, VD000 |

| 0005 | COMMON EPSRT, EPS0, EPS, IQ2, 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 | UD0=1.0+UQ3 |

| 0014 | VD0=VQ3 |

| 0015 | UD=(UPX3-(UDDD*UPX2+VDDD*VPX2)+UQ3*VPX3)+UQ3*VPX3 |

| 0016 | VD=VPX3-(VDDD*UPX2+VDDD*VPX2)+VQ3*UPX3+VQ3*VPX3 |

| 0017 | UAAA=2.0*UQ3 |

| 0018 | VAAA=UAAA+1.0 |

| 0019 | UBBB=UDDD*UDDD*VDDD |

| 0020 | VBBB=VDDD*UDDD*VDDD |

| 0021 | UCCCC=UQ3*VQ3*VQ3 |

| 0022 | VCCCC=VQ3*VQ3*VQ3 |

| 0023 | U=(UAAA*UPX3-VAAA*VPX3)-(UBBB*UPX2-VBBB*VPX2)+(VCCC*UPX1-VCCC*V |

| 0024 | VPX1) |

| 0025 | VB=(VAAA*UPX3+UAAA*VPX3)-(VBBB*UPX2+VBBB*VPX2)+(VCCC*UPX1+VCCC*V |

| 0026 | VPX1) |

| 0027 | UC=UDDD*UPX3-VDDD*VPX3 |

| 0028 | VC=VDDD*UPX3-VDDD*VPX3 |

| 0029 | UDISC=UBB*UHH-VBB*VHH-(4.0*(UDUC-UC*VC)) |

| 0030 | VDISC=(2.0*(VBB*UHB-VBB*VHB)-(4.0*(VDUC+UD*VC)) |

| 0031 | AAA+OSRT(U DISC*UDISC+V DISC*VDISC) |

| 0032 | IF(AAAA.EQ.0.0) GO TO 5 |

| 0033 | GO TO 7 |

| 0034 | 5 THE=0.0 |

| 0035 | THE=DATA12*(V DISC*UDISC) |

| 0036 | 9 RAD=OSRT(AAAA) |

| 0037 | ANGLE+THE=2.0 |

| 0038 | UTES=RAD+DCOS(ANGLE) |

| 0039 | VTEST=RAD+DSIN(ANGLE) |

| 0040 | UD1N=UB+UTES |

| 0041 | VD1N=VB+VTEST |

| 0042 | UD2N=UB-VTEST |

| 0043 | VD2N=VB-VTEST |

| 0044 | ARG1=UD1N*UDEN1+YDEN1*YDEN1 |
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 V04=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
0054 GO TO 50
0055 10 IF(BBB.EQ.0.01 GO TO 60
0056 UAAA=-2.0*UC
0057 VAAA=-2.0*VC
0058 UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2
0059 V04=(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(X3,UX4,UX4,CONV)
C ****************************************************
C * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C * IMATIONS BY TESTING THE EXPRESSION
C * ABSOLUTE VALUE OF IX(XN+1)-X(N)/ABSOLUTE VALUE OF X(N+1).
C * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C *****************************************************
0002 DO DOUBLE PRECISION UX3,UX4,UX4,EPSP1,EPSP2,EPSP2,AAA,UDUMMY,VDUMMY,
C IDENOM
C LOGICAL CONV
C COMMON EPSRT,EPSP1,EPSP2,MAX
C UDUMMY=UX4-UX3
C VDUMMY=UX4-UX3
C AAA=DSORT(UDUMMY+UDUMMY+UDUMMY+UDUMMY)
C IDENOM=DSORT(UX4*UX4*UX4*UX4)
C IF(IDENOM.LT.EPSP1) GO TO 20
C IF(AAA.IDENOM.LT.EPSP2) GO TO 10
C 5 CONV=.FALSE.
C GO TO 100
C 10 CONV=.TRUE.
C GO TO 100
C 20 IF(AAA.LT.EPSP2) GO TO 10
C GO TO 5
C 100 RETURN
C END

0001 SUBROUTINE HORNER(NA,VA,UX,UX,UB,UB,UPX,VPX)
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 *****************************************************
0002 DO DOUBLE PRECISION UX,UX,UPX,VPX,UB,UB,VA,VA
C DIMENSION UX(26),VA(26),UB(26),VPX(26)
C 0004 UB(I)=VA(I)
C 0005 VB(I)=VA(I)
C 0006 NUM=NA+1
C GO TO 10 IF(1,NUM)
C 0008 VB(I)=VA(I)+UB(I-1)*UX+VB(I-1)*UX
C 10 VB(I)=VA(I)+VB(I-1)*UX+UB(I-1)*UX
C UPC=UB(NUM)
C VPX=VPX(NUM)
C RETURN
C END
TABLE C.V (Continued)

0001 SUBROUTINE QUAD(UA,VA,UROOT,VROOT,NROOT,MULTI,EPST)
0002 ***********************************************************************
0003 * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES *
0004 * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
0005 * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.                       *
0006 ***********************************************************************
0007 DOUBLE PRECISION UA,VA,UROOT,VROOT,6BBB,UAAA,VAAA,UDDISC,VDDISC,UDDUMM
0008 IY,VDUMMY,ROUMMY,SQUMMY,EPST,UBBB,VBBB
0009 DIMENSION UA(261),VA(261),URootT(251),VROOT(251),MULTI(251)
0000 IF (NA.EQ.2) GO TO 7
0001 IF (NA.EQ.1) GO TO 5
0002 UROOT(NROOT+1)=U.0
0003 VROOT(NROOT+1)=V.0
0004 MULTI(NROOT+1)=1
0005 NROOT=NROOT+1
0006 GO TO 50
0007 5 BBB=UA(1)*UA(1)+VA(1)*VA(1)
0008 UROOT(NROOT+1)=-(UA(1)*UA(1)-VA(1)*VA(1))/BBB
0009 VROOT(NROOT+1)=-(UA(1)*UA(1)+VA(1)*VA(1))/BBB
0010 MULTI(0ROOT+1)=1
0011 NROOT=NROOT+1
0012 GO TO 50
0013 7 UDDISC=UA(1)*UA(1)-VA(1)*VA(1)-(4.0*(UA(1)*UA(1)-VA(1)*VA(1)))
0014 VDDISC=VA(1)*VA(1)-(4.0*(UA(1)*UA(1)-VA(1)*VA(1)))
0015 BBBB=DSORT(UDDISC*UDDISC+VDDISC*VDDISC)
0016 IF (BBBB.LT.EPST) GO TO 10
0017 CALL COMSRT(UDDISC,VDISCS,UDUMM,VDUMM)
0018 UBBB=-UA(1)+UDUMM
0019 VBBB=-VA(1)+VDUMM
0020 ROUMMY=-UA(1)-UDUMM
0021 SDUMMY=-VA(1)-VDUMM
0022 UAAA=2.0*UA(1)
0023 VAAA=2.0*VA(1)
0024 BBB=UAAA*UAAA+VAAA*VAAA
0025 UROOT(NROOT+1)=-(UAAA+BBB*VAAA)/(BBB
0026 VROOT(NROOT+1)=-(VAAA+BBB*UAAA)/(BBB
0027 MULTI(NROOT+1)=2
0028 NROOT=NROOT+1
0029 GO TO 50
0030 10 UAAA=2.0*UA(1)
0031 VAAA=2.0*VA(1)
0032 BBB=UAAA*UAAA+VAAA*VAAA
0033 UROOT(NROOT+1)=-(UA(1)*UA(1)-VA(1)*VA(1))/BBB
0034 VROOT(NROOT+1)=-(UA(1)*UA(1)+VA(1)*VA(1))/BBB
0035 MULTI(NROOT+1)=2
0036 NROOT=NROOT+1
0037 GO TO 50
0038 50 RETURN
0039 END
TABLE C.V (Continued)

SUBROUTINE COMSQRT(UX,UY,VX,VY)
C**************************************************************************
C*
C* THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
C*
C**************************************************************************

DOUBLE PRECISION UX,UX+VX+VY,DUMMY,AAA,BBB
R=DSQRT(UX*UX+VX*VX)
AAA=DSQRT(DABS(R+UX/2.0))
BBB=DSQRT(DABS(R-UX/2.0))

IFVX) 10,20,30
10 UY=AAA
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

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

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.

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.
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: ($j = 1, 3$)

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

$$\beta = \tan^{-1} \left( \frac{\text{Im} X_0}{\text{Re} X_0} + \frac{K}{3} \right) ; K = 1 \text{ if } j = 1$$

$$K = 2 \text{ if } j = 3.$$

If the number of the alteration is even: ($j = 0, 2, 4$)
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.
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 \( XEND + 0.5N \).

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

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>8</th>
<th>6</th>
<th>7</th>
<th>7</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XSTART</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XEND</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
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 7th degree polynomial. After these approximations are used the circle of radius 15 will be searched for the remaining roots.

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 Table E.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. - 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)
UAPP(N), VAPP(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.1 (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**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOPOLY</td>
<td>c.c. 1-2</td>
<td>Number of the polynomial. Integer. Right justified.</td>
</tr>
<tr>
<td>NP</td>
<td>c.c. 4-5</td>
<td>Degree of the polynomial. Integer. Right justified.</td>
</tr>
<tr>
<td>NAPP</td>
<td>c.c. 7-8</td>
<td>Number of initial approximations to be read. Integer. Right justified. If no initial approximations are given, leave blank.</td>
</tr>
<tr>
<td>MAX</td>
<td>c.c. 19-21</td>
<td>Maximum number of iterations. Integer. Right justified. 200 is recommended.</td>
</tr>
<tr>
<td>EPS1</td>
<td>c.c. 23-28</td>
<td>Test for zero in subroutine GCD. Double precision. Right justify. 1.D-03 is recommended.</td>
</tr>
</tbody>
</table>
### TABLE E.II (Continued)

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

**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

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

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

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

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

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCHECK</td>
<td>c.c. 80</td>
<td>Must contain the number 1. Integer.</td>
</tr>
</tbody>
</table>

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
### Example

#### Figure E.2. Control Card for G.C.D. - Newton's Method

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00000000011111111122222222233333333344444455555555566666666777777778</td>
</tr>
<tr>
<td></td>
<td>1234567890123456789012345678901234567890123456789012345678901234567890</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

#### Example

+0.125768D+01

#### Figure E.3. Coefficient Card for G.C.D. - Newton's Method

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00000000011111111122222222233333333344444455555555566666666777777778</td>
</tr>
<tr>
<td></td>
<td>1234567890123456789012345678901234567890123456789012345678901234567890</td>
</tr>
<tr>
<td></td>
<td>UP</td>
</tr>
<tr>
<td></td>
<td>+0.125768D+01</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>I</td>
<td>J</td>
<td>I</td>
<td></td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
<td></td>
<td>Program control</td>
</tr>
<tr>
<td>NOPOLY</td>
<td>I</td>
<td>NOPOLY</td>
<td>I</td>
<td></td>
<td>Number of the polynomial</td>
</tr>
<tr>
<td>NF</td>
<td>I</td>
<td>NF</td>
<td>I</td>
<td></td>
<td>Degree of the original polynomial</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>UP,VP</td>
<td>D</td>
<td></td>
<td>Array of coefficients of original polynomial, P(X)</td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
<td></td>
<td>Number of initial approximation to be read</td>
</tr>
<tr>
<td>EPS1</td>
<td>R</td>
<td>EPS1</td>
<td>D</td>
<td></td>
<td>Tolerance check for zero (0) in Subroutine GCD</td>
</tr>
<tr>
<td>EPS2</td>
<td>R</td>
<td>EPS2</td>
<td>D</td>
<td></td>
<td>Tolerance check for convergence</td>
</tr>
<tr>
<td>EPS3</td>
<td>R</td>
<td>EPS3</td>
<td>D</td>
<td></td>
<td>Tolerance check for zero (0) in Subroutine QUAD</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td></td>
<td>Maximum number of iterations permitted</td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
<td></td>
<td>Unit number of input device</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td></td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>KCHECK</td>
<td>I</td>
<td>KCHECK</td>
<td>I</td>
<td></td>
<td>Program control, KCHECK = 1 implies stop execution</td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>UAPP,VAPP</td>
<td>D</td>
<td></td>
<td>Array of initial approximations</td>
</tr>
<tr>
<td>XSTART</td>
<td>R</td>
<td>XSTART</td>
<td>D</td>
<td></td>
<td>Magnitude at which to start search for roots</td>
</tr>
<tr>
<td>XEND</td>
<td>R</td>
<td>XEND</td>
<td>D</td>
<td></td>
<td>Magnitude at which to end search for roots</td>
</tr>
<tr>
<td>ANAME</td>
<td>A</td>
<td>ANAME</td>
<td>A</td>
<td></td>
<td>Contains name of method used &quot;NEWTONS&quot;</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT,VROOT</td>
<td>D</td>
<td></td>
<td>Array of roots found</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td></td>
<td>Array of multiplicities</td>
</tr>
<tr>
<td>DP</td>
<td>C</td>
<td>UDP,VDP</td>
<td>D</td>
<td></td>
<td>Array containing coefficients of the derivative, (P'(X)), of P(X)</td>
</tr>
<tr>
<td>NDP</td>
<td>I</td>
<td>NDP</td>
<td>I</td>
<td></td>
<td>Degree of the derivative of original polynomial</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>UD,VD</td>
<td>D</td>
<td></td>
<td>Array of coefficients of the greatest common divisor of P(X) and P'(X)</td>
</tr>
<tr>
<td>ND</td>
<td>I</td>
<td>ND</td>
<td>I</td>
<td></td>
<td>Degree of g.c.d. of P(X) and P'(X)</td>
</tr>
<tr>
<td>Q</td>
<td>C</td>
<td>UQ,VQ</td>
<td>D</td>
<td></td>
<td>Array of coefficients of quotient polynomial P(X)/g.c.d.</td>
</tr>
</tbody>
</table>
### TABLE E.VI (Continued)

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NQ</td>
<td>I</td>
<td>NQ</td>
<td>I</td>
<td>Degree of quotient polynomial $Q(X)$</td>
<td></td>
</tr>
<tr>
<td>ZRO</td>
<td>C</td>
<td>UZRO, VZRO</td>
<td>D</td>
<td>Value at which to evaluate or deflate polynomial</td>
<td></td>
</tr>
<tr>
<td>DUMMY</td>
<td>C</td>
<td>UDUMMY, VDUMMY</td>
<td>D</td>
<td>Dummy variable</td>
<td></td>
</tr>
<tr>
<td>QQ</td>
<td>C</td>
<td>UQQ, VQQ</td>
<td>D</td>
<td>Working array of coefficients of current polynomial</td>
<td></td>
</tr>
<tr>
<td>NQQ</td>
<td>I</td>
<td>NQQ</td>
<td>I</td>
<td>Degree of current polynomial, $QQ(X)$</td>
<td></td>
</tr>
<tr>
<td>IALTER</td>
<td>I</td>
<td>IALTER</td>
<td>I</td>
<td>Number of alterations of an initial approximation</td>
<td></td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td>CONV = TRUE implies convergence to a root</td>
<td></td>
</tr>
<tr>
<td>EPS4</td>
<td>R</td>
<td>EPS4</td>
<td>D</td>
<td>Tolerance for checking multiplicities</td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td>C</td>
<td>UAP, VAP</td>
<td>D</td>
<td>Array of approximations (initial or altered) producing convergence</td>
<td></td>
</tr>
<tr>
<td>QD</td>
<td>C</td>
<td>UQD, VQD</td>
<td>D</td>
<td>Array of coefficients of newly deflated polynomial</td>
<td></td>
</tr>
<tr>
<td>JAP</td>
<td>I</td>
<td>JAP</td>
<td>I</td>
<td>Number of distinct roots found by iterative process</td>
<td></td>
</tr>
<tr>
<td>J1</td>
<td>I</td>
<td>J1</td>
<td>I</td>
<td>i.e. not as a result of Subroutine QUAD</td>
<td></td>
</tr>
<tr>
<td>ROOTS</td>
<td>C</td>
<td>UROOTS, VROOTS</td>
<td>D</td>
<td>Number of distinct roots found in the attempt to improve roots</td>
<td></td>
</tr>
<tr>
<td>NEWT</td>
<td>L</td>
<td>NEWT</td>
<td>L</td>
<td>Array of improved roots</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Program control. NEWT = TRUE implies that Newton's method was used instead of Subroutine QUAD</td>
<td></td>
</tr>
</tbody>
</table>

**Subroutine NEWTON**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>C</td>
<td>Starting approximation (initial or altered)</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>Degree of current polynomial</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>Array of coefficients of current polynomial</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>Maximum number of iterations</td>
</tr>
<tr>
<td>EPSILON</td>
<td>R</td>
<td>Tolerance for checking convergence</td>
</tr>
<tr>
<td>XO</td>
<td>C</td>
<td>Current approximation to root</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>Array of coefficients of newly deflated polynomial</td>
</tr>
<tr>
<td>DPXO</td>
<td>C</td>
<td>Derivative of the polynomial at XO</td>
</tr>
</tbody>
</table>
### TABLE E.VI (Continued)

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIFF</td>
<td>C</td>
<td>UDIFF, VDIFF</td>
<td>D</td>
<td></td>
<td>PXO/DFXO</td>
</tr>
<tr>
<td>PXO</td>
<td>C</td>
<td>UPXO, VPXO</td>
<td>D</td>
<td></td>
<td>Value of polynomial at X0</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td>R</td>
<td>CONV = TRUE implies convergence to root</td>
</tr>
</tbody>
</table>

**Subroutine HORNER**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Variable</th>
<th>Type</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>C</td>
<td>UX, VX</td>
<td>D</td>
<td>E</td>
<td>Value at which to evaluate or deflate polynomial</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>U, VP</td>
<td>D</td>
<td></td>
<td>Array of coefficients of polynomial</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>UC, VC</td>
<td>D</td>
<td>R</td>
<td>Updated at each iteration to yield derivative of polynomial at X</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB, VB</td>
<td>D</td>
<td></td>
<td>Array of coefficients of newly deflated polynomial</td>
</tr>
</tbody>
</table>

**Subroutine QUAD**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Variable</th>
<th>Type</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial to be solved</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA, VA</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of polynomial to be solved</td>
</tr>
<tr>
<td>J</td>
<td>I</td>
<td>J</td>
<td>I</td>
<td>ECR</td>
<td>Number of distinct roots found of original polynomial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(J = -1 implies original polynomial is of degree 2 or 1)</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT, VROOT</td>
<td>D</td>
<td>ECR</td>
<td>Array of roots found</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td>ECR</td>
<td>Array of multiplicities</td>
</tr>
<tr>
<td>DISC</td>
<td>C</td>
<td>UDISC, VDISC</td>
<td>D</td>
<td>ECR</td>
<td>Discriminate of quadratic</td>
</tr>
<tr>
<td>TEMP</td>
<td>C</td>
<td>UTEMP, VTEMP</td>
<td>D</td>
<td>ECR</td>
<td></td>
</tr>
<tr>
<td>EPSLON</td>
<td>R</td>
<td>EPSLON</td>
<td>D</td>
<td>C</td>
<td>Tolerance for zero (0)</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>UD, VD</td>
<td>D</td>
<td></td>
<td>Twice leading coefficient of quadratic</td>
</tr>
</tbody>
</table>
TABLE E.VI (Continued)

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Single Precision Type</th>
<th>Double Precision Variable</th>
<th>Double Precision Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>C</td>
<td>UR,VR</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of original polynomial, P(X)</td>
</tr>
<tr>
<td>S</td>
<td>C</td>
<td>US,VS</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of derivative polynomial, P'(X)</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
<td>Degree of original polynomial, P(X)</td>
</tr>
<tr>
<td>M</td>
<td>I</td>
<td>M</td>
<td>I</td>
<td>E</td>
<td>Degree of derivative polynomial, P'(X)</td>
</tr>
<tr>
<td>RR</td>
<td>C</td>
<td>URR,VR</td>
<td>D</td>
<td>R</td>
<td>Array of coefficients of dividend polynomial</td>
</tr>
<tr>
<td>SS</td>
<td>C</td>
<td>USS,VSS</td>
<td>D</td>
<td>R</td>
<td>Array of coefficients of divisor polynomial also array of coefficients of g.c.d. of P(X) and P'(X) when returned</td>
</tr>
<tr>
<td>N1</td>
<td>I</td>
<td>N1</td>
<td>I</td>
<td>R</td>
<td>Degree of dividend polynomial, RR(X)</td>
</tr>
<tr>
<td>M1</td>
<td>I</td>
<td>M1</td>
<td>I</td>
<td>R</td>
<td>Degree of divisor polynomial, SS(X), also degree of g.c.d. of P(X) and P'(X) when returned</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>UD,VD</td>
<td>D</td>
<td>R</td>
<td>Quotient ( \frac{RR_{N1+1}}{SS_{M1+1}} )</td>
</tr>
<tr>
<td>T</td>
<td>C</td>
<td>UT,VT</td>
<td>D</td>
<td>R</td>
<td>Array of coefficients of difference polynomial(RR - D(SS))</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td>R</td>
<td>Degree of difference polynomial T(X)</td>
</tr>
<tr>
<td>EPSLON</td>
<td>R</td>
<td>EPSLON</td>
<td>D</td>
<td>R</td>
<td>Tolerance check for zero (0)</td>
</tr>
</tbody>
</table>

Subroutine GCD

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Single Precision Type</th>
<th>Double Precision Variable</th>
<th>Double Precision Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
<td>Degree of original polynomial, P(X)</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>UP,VP</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of original polynomial, P(X)</td>
</tr>
<tr>
<td>J</td>
<td>I</td>
<td>J</td>
<td>I</td>
<td>E</td>
<td>Number of distinct roots of P(X)</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT,VROOT</td>
<td>D</td>
<td>E</td>
<td>Array of distinct roots of P(X)</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA,VA</td>
<td>D</td>
<td>E</td>
<td>Working array of coefficients of current polynomial</td>
</tr>
<tr>
<td>M</td>
<td>I</td>
<td>M</td>
<td>I</td>
<td>E</td>
<td>Degree of current polynomial, A(X)</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td>R</td>
<td>Array of multiplicities of the roots</td>
</tr>
<tr>
<td>IO2</td>
<td>I</td>
<td>IO2</td>
<td>I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB,VB</td>
<td>D</td>
<td>C</td>
<td>Array of coefficients of newly deflated polynomial</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>UC,VC</td>
<td>D</td>
<td>C</td>
<td>Derivative of polynomial at ( \text{ROOT}_i )</td>
</tr>
<tr>
<td>EPSLON</td>
<td>R</td>
<td>EPSLON</td>
<td>D</td>
<td>C</td>
<td>Tolerance for checking multiplicities</td>
</tr>
</tbody>
</table>
TABLE E.VI (Continued)

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Single Precision Type</th>
<th>Double Precision Variable</th>
<th>Double Precision Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial, ( P(X) )</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>UP, VP</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of polynomial, ( P(X) )</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA, VA</td>
<td>D</td>
<td>R</td>
<td>Array of coefficients of derivative, ( P'(X) )</td>
</tr>
<tr>
<td>M</td>
<td>I</td>
<td>M</td>
<td>I</td>
<td>R</td>
<td>Degree of derivative polynomial, ( P'(X) )</td>
</tr>
</tbody>
</table>

Subroutine DERIV

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)

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOLD</td>
<td>C</td>
<td>XOLDR,XOLDI</td>
<td>D</td>
<td>ECR</td>
<td></td>
<td>Old approximation to be altered to new approximation</td>
<td></td>
</tr>
<tr>
<td>NALTER</td>
<td>I</td>
<td>NALTER</td>
<td>I</td>
<td>ECR</td>
<td></td>
<td>Number of alterations performed on an initial approximation</td>
<td></td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
<td>E</td>
<td></td>
<td>Program control</td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td>C</td>
<td></td>
<td>Maximum number of iterations permitted</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>R</td>
<td>XOLDI</td>
<td>D</td>
<td></td>
<td></td>
<td>Imaginary part of original initial approximation (unaltered)</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>R</td>
<td>XOLDR</td>
<td>D</td>
<td></td>
<td></td>
<td>Real part of original, unaltered initial approximation</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>ABXOLD</td>
<td>D</td>
<td></td>
<td></td>
<td>Magnitude of original unaltered initial approximation</td>
<td></td>
</tr>
<tr>
<td>BETA</td>
<td>R</td>
<td>BETA</td>
<td>D</td>
<td></td>
<td></td>
<td>Argument of new approximation</td>
<td></td>
</tr>
<tr>
<td>XOLDR</td>
<td>R</td>
<td>XOLDR</td>
<td>D</td>
<td></td>
<td></td>
<td>Real part of new approximation</td>
<td></td>
</tr>
<tr>
<td>XOLDI</td>
<td>R</td>
<td>XOLDI</td>
<td>D</td>
<td></td>
<td></td>
<td>Imaginary part of new approximation</td>
<td></td>
</tr>
<tr>
<td>IO2</td>
<td>I</td>
<td>IO2</td>
<td>I</td>
<td>C</td>
<td></td>
<td>Unit number of output device</td>
<td></td>
</tr>
</tbody>
</table>

**Subroutine ALTER**

**Subroutine COMSQRT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UX,VX</td>
<td>D</td>
<td>E</td>
<td>Complex number for which the square root is desired</td>
</tr>
<tr>
<td>UY,VY</td>
<td>D</td>
<td>R</td>
<td>Square root of the complex number</td>
</tr>
</tbody>
</table>
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.
Figure E.6. Flow Charts for C.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 * DOUBLE PRECISION PROGRAM FOR G.C.D.-NEWTON'S METHOD
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 *
0001 DOUBLE PRECISION UP,VP,UAPP,VAPP,VRoot,VRoot1,UDP,VDP,UD0,VD0,UX0,V2
0002 1R0,UX,VO,UDummy,V Dummy,UD0,VD0,UAPP,VAPP,URoot1,VRoot,VRoot1,Max
0003 EPS1, EPS2, EPS3, EPS4,1D2, MAX
0004 COMMON EPS1, EPS2, EPS3, EPS4,1D2, MAX
0005 LOGICAL NEWT, CONV
0007 DATA PHASE,OJNAME,OJNAME,2PHI,2HI,3H001/
0008 DATA NENTH(1),H1(1),H1(1),H1(1),H5(1),H5(1),H5(1),H8(1),H10(1),H11(1),H12(1),H13
0009 1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
0010 DATA ANAME1(1), ANAME2(1), NENTH,4HNEWT,4HCONS
0011 101=0
0012 10 J=0
0013 11 TIME=0
0014 READ(101,1000) NOPOLYP, NAPP, MAX, EPS1, EPS2, EPS3, EPS4, XSTART, XEND.
IMCHECK
0015 IF IMCHECK.EQ.11 STOP
0016 WRITE(102,2000) NAPP
0017 WRITE(102,2000) NAPP
0018 WRITE(102,2000) MAX
0019 WRITE(102,2000) EPS1
0020 WRITE(102,2000) EPS2
0021 WRITE(102,2000) EPS3
0022 WRITE(102,2000) EPS4
0023 WRITE(102,2000) XSTART
0024 WRITE(102,2000) XEND
0025 WRITE(102,2000) EPS
0026 KK=NP+1
0027 NNN=KK+1
0028 DO 20 I=1,KK
0029 JJ=NNN+1
0030 20 READ(101,1010) UP(I), DJI, VP(I)
0031 IF INAPP .NE. 01 GO TO 22
0032 NAPP=NP
0033 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0034 GO TO 23
0035 22 READ(101,1015) (UAPP(I),VAPP(I),I=1,NAPP)
0036 23 WRITE(102,1030) NP
0037 KK=NP+1
0038 NNN=KK+1
0039 DO 25 I=1,KK
0040
TABLE E.VII (Continued)

0040  JJJ=NNN-1
0041  WRITE(IO2,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
0042  IF(NP.GE.3) GO TO 30
0043  J=1
0044  CALL QUAD(INP,UP,VP,J,URoot,VRoot,MULT)
0045  WRITE(IO2,1070)
0046  WRITE(IO2,1165) (1,URoot(1),VRoot(1),MULT(1),J=1,J)
0047  GO TO 10
0048  30 CALL DERIV(NP,UP,VP,NDP,UPD,VPD)
0049  CALL GCD(NP,UP,VP,NDP,UPD,VPD,ND,UD,VD)
0050  IF(ND.GE.3) GO TO 70
0051  IF(ND.EQ.0) GO TO 65
0052  UDUMM=UD(1)+UD(2)+VD(1)+VD(2)
0053  UZRO=-(UD(1)+UD(2))-(VD(1)+VD(2))/UDUM
0054  VZRO=-(UD(1)+UD(2))-(VD(1)+VD(2))/UDUM
0055  CALL HOMER(UZRO,VDRO,VP,UP,UD,VD,UDUM,VDUM)
0056  NQ=NP-1
0057  DO 60 I=1,NP
0058  UQ(I)=UQ(I+1)
0059  VQ(I)=VQ(I+1)
0060  GO TO 80
0061  60 KKK=NP+1
0062  DO 66 I=1,KKK
0063  UQ(I)=UQ(I)
0064  VQ(I)=VQ(I)
0065  NQ=NP
0066  GO TO 80
0067  70 CALL DIVIDE(NP,UP,VP,ND,UD,VD,UDQ,VDQ)
0068  80 WRITE(IO2,1120) NQ
0069  KKK=NP-1
0070  NNN=KKK+1
0071  DO 83 I=1,NK
0072  JJJ=NNN-I
0073  WRITE(IO2,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(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  UQ(I)=UQ(I)
0079  VQ(I)=VQ(I)
0080  NQ=NQ
0081  GO TO 120
0082  110 CALL QUAD(INP,UP,VP,J,URoot,VRoot,MULT)
0083  NEWT=.FALSE.
0084  GO TO 310
0085  120 DO 200 I=1,NAPP
0086  130 CALL NEWTON(UAPP(I),VAPP(I),NQQ,QQQ,QQD,QQD,CONV)
0087  IF(CONV) GO TO 160
0088  CALL ALTER(UAPP(I),VAPP(I),IALTER,ITIME)
0089  IF(IALTER.GT.5) GO TO 200
0090  GO TO 130
0091  160 J=J+1
0092  URoot(J)=UZRO
0093  VRoot(J)=VZRO
0094  UAPP(J)=UAPP(I)
0095  VAPP(J)=VAPP(I)
0096  CALL HOMER(UZRO,VDRO,NQQ,QQQ,QQD,QQD,UDUM,VDUM)
TABLE E.VII (Continued)

```
0098  DO 100  I=1,NQQ
0099  UQQ(II)=UQQ(II+1)
0100  180  VQQ(I)=VQQ(I+1)
0101  NQQ=NQQ-1
0102  IF(NQQ.LT.1) GO TO 220
0103  200  CONTINUE
0104  IF(JJ.GE.EQ.0) GO TO 205
0105  IF(XEND.EQ.0.02) GO TO 205
0106  IF(XSTART.GE.XEND) GO TO 205
0107  NAPP=NQ
0108  CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0109  GO TO 120
0110  205  IF(NQQ.LE.21) GO TO 210
0111  WRITE(102,1200)
0112  120  KKK=NQQ+1
0113  NNN=KKK+1
0114  DO 157  L=1,KKK
0115  157  JJJ=NNN-L
0116  WRITE(102,1100) QNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0117  210  IF(JJ.EQ.0) GO TO 10
0118  JAP=J
0119  GO TO 230
0120  220  JAP=J
0121  CALL QUADUQQ(UQQ,VQQ,J,UROOT,VRVD,MULTI)
0122  230  WRITE(102,1133) I,UROOT(I),VRVD(I),UAP(I),VAP(I),I=1,JAP
0123  235  KKK=JAP+1
0124  WRITE(102,1134) I,UROOT(I),VRVD(I),I=KKK,J
0125  240  J=J+1
0126  DO 300  I=1,J
0127  300  CALL NEWTON(UROOT(V),VRVD(I),UQ,VZRO,CONV)
0128  305  IF(CONV) GO TO 280
0129  WRITE(102,1140) I,UROOT(I),VRVD(I)
0130  IF(I.EQ.JAP) GO TO 241
0131  IF(I.EQ.JAP) GO TO 250
0132  303  KKK=JAP+1
0133  310  DO 306  I=1,J
0134  306  UROOT(I)=UZRO
0135  307  VROOT(I)=VZRO
0136  315  CONTINUE
0137  320  KKK=JAP+1
0138  325  DO 345  I=1,KKK
0139  330  UAPP(I)=UAP(I)
0140  345  VAPP(I)=VAP(I)
0141  350  JAP=JAP-1
0142  360  GO TO 300
0143  370  IF(JJ.EQ.0) GO TO 305
0144  380  J=J+1
0145  390  DO 303  I=1,J
0146  303  UROOT(I)=URVOTS(I)
0147  308  VROOT(I)=VRVOTS(I)
0148  313  GO TO 307
0149  315  WRITE(102,1150) NQ
0150  320  KK=NQQ+1
0151  325  NEW=KK+1
0152  330  DO 360  L=1,KKK
```
TABLE E.VII (Continued)

```
156 WRITE(102,1040) QNAME,ENTRY,NNN,L
158 GO TO 10
159 307 NEWT=.TRUE.
160 CALL MULTI(NP,UP,VP,J,ROOT,VROOT,MULTI)
161 IF(NEWT) GO TO 330
162 WRITE(102,1070)
163 WRITE(102,1165) (L,ROOT(L),VROOT(L),MULTI(L),L=1,J)
164 GO TO 10
165 330 WRITE(102,1101)
166 WRITE(102,1190) (L,ROOT(L),VROOT(L),MULTI(L),L=1,JAP)
167 KKK=JAP+1
168 IF(JAP.LT.J) WRITE(102,165) (L,ROOT(L),VROOT(L),MULTI(L),L=1,J)
169 GO TO 10
170 1000 FORMAT(3/(I2,1X,9X,13X,4(0D6.0,1X,13X,2(D7.0,1X,11))
171 1010 FORMAT(2D30.0)
172 1020 FORMAT(42030.0)
173 1030 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
174 1040 FORMAT(2X,42A2,4H) = .D23.16,3H + .D23.16,2H)
175 1050 FORMAT(///,73HTHE DEGREE OF Q(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
176 1060 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
177 1070 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
178 1080 FORMAT(2X,42A2,4H) = .D23.16,3H + .D23.16,2H)
179 1090 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
180 1100 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
181 1110 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
182 1120 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
183 1130 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
184 1140 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
185 1150 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
186 1160 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
187 1170 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
188 1180 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
189 1190 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
190 1200 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
191 1210 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
192 1220 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
193 1230 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
194 1240 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
195 1250 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
196 1260 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
197 1270 FORMAT(///,73HTHE DEGREE OF P(X) IS ,I2,///,22HTHE COEFFICIENTS ARE//)
198 END
```
SUBROUTINE GENAPP(APPR,APP1,NAPP,XSTART)

C******************************************************************************
C* SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
C* DEGREE OF THE ORIGINAL POLYNOMIAL.                                      *
C******************************************************************************
*C******************************************************************************

DOUBLE PRECISION APPR,APP1,XSTART,BETA, EPS1,EPS2,EPS3,EPS4
COMMON EPS1,EPS2,EPS3,EPS4,IO2,MAX
IF(XSTART.EQ.0.0) XSTART=0.5
BETA=0.2617994
JO 10 J=1,NAPP
APPR(J)=XSTART*DCOS(BETA)
APPI(J)=XSTART*DSIN(BETA)
BETA=BETA+0.5235988
10 XSTART=XSTART+0.5
RETURN
END
TABLE E.VII (Continued)

0001       SUBROUTINE ALTER(KOLD1,KOLD2,NALTER,ITIME)
0002       COMMON EPS1, EPS2, EPS3, EPS4, IO2, MAX
0003       IF(ITIME.NE.0) GO TO 5
0004       ITIME = 1
0005       WRITE(102,1010) MAX
0006       IF(NALTER.EQ.0) GO TO 10
0007       WRITE(102,1000) KOLD1,KOLD2
0008       GO TO 20
0009       ABXOLD=DSQRT(KOLD1*KOLD2+(KOLD1*KOLD2))
0010       BETA=DATAN2(KOLD2,KOLD1)
0011       WRITE(102,1020) KOLD1,KOLD2
0012       IF(NALTER.GT.5) RETURN
0013       GO TO (30,40,50,60,70,NALTER)
0014       30 KOLD1=KOLD1
0015       40 KOLD2=KOLD2
0016       50 RETURN
0017       100 FORMAT(1X,'D23.16, 3H+ .D23.16,2H 1,10X,2IHALTED APPROXIMATION')
0018       1010 FORMAT(1X,'H9, CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS IF
0019       1020 FORMAT(1X,'D23.16, 3H+ .D23.16,2H 1,10X,2IINITIAL APPROXIMATION')
0020       END
TABLE E.VII (Continued)

SUBROUTINE GCD(IN, V, UR, VR, US, VS, M, N, USS, VSS)

C ******************************************************************
C * GIVEN POLYNOMIALS P(X) AND D(P(X)) WHERE DEG. D(P(X)) IS LESS THAN DEG.
C * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND
C * D(P(X)).
C ******************************************************************

DOUBLE PRECISION US, VS
DOUBLE PRECISION UR, VR, US, VS, USS, VSS, URR, VRR, U, V, UT, VT, EPSLON, EPS, EPS2, EPS3, EPS4
DIMENSION UR(26), VR(126), US(26), VS(26), USS(26), VSS(26), URR(26), VRR(126), UT(26), VT(26)
COMMON EPSLONE(102), EPS2(102), EPS3(102), EPS4(102), MAX

NI = N
MK = M
KKK = N - M

DO I = 1, KKK
UR(II) = U(i)
VR(II) = V(i)

DO I = KKK + 1, N - M
USS(II) = U(i)
VSS(II) = V(i)

BBB = USS(II) * USS(II) + VSS(II) * VSS(II)
UD = U(i) * USS(II) + V(i) * VSS(II)
VD = V(i) * USS(II) + U(i) * VSS(II)
BBB = SQRT(UD * UD + VD * VD)
USS = (UD * V(i) + VD * U(i)) / BBB
VSS = (U(i) * UD - V(i) * VD) / BBB

IF(MI.EQ.M) GO TO 70

K = N - M

DO 10 I = 1, KKK
UT(II) = U(i)
VT(II) = V(i)

10 CONTINUE

IF(MI.EQ.M) GO TO 70

100 CONTINUE

IF(KK.EQ.1) GO TO 170

IF(KK.EQ.M) GO TO 140

GO TO 200
TABLE E.VII (Continued)

```
0050  140  KKK=K+1
0051  DO  150  J=1,KKK
0052    URR(J)=USSI(J)
0053    VRR(J)=VSSI(J)
0054  150  USSI(J)=UT(J)
0055  150  VSSI(J)=VT(J)
0056    KKK=K+2
0057    NNN=NN+1
0058    DO  160  J=KKK,NNN
0059    URR(J)=USS(J)
0060  160  VRR(J)=VSS(J)
0061    M1=M1
0062    M1=K
0063    GO TO 30
0064  170  USSI(J)=1.0
0065  170  VSSI(J)=0.0
0066    M1=0
0067  200  RETURN
0068  END
```
TABLE E.VII (Continued)

SUBROUTINE QUAD(UA,VA,J,UROOT,VROOT,MULT)

* SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLICITIES
* OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE
* QUADRATIC IS DONE USING THE QUADRATIC FORMULA.

DOUBLE PRECISION UA,VA,UROOT,VROOT,UDISC,VDISC,UTEMP,VTEMP,UD,VD,E

DIMENSION UA(26),VA(26),UROOT(25),VROOT(25),MULT(25)

COMMON EPS1,EPS2,EPS4,EPSLON,MAX

IF(J.LT.0) GO TO 40

J=J+1

GO TO 50

MULT(1)=I

GO TO 50

MULT(I)=2

J=1

GO TO 80

MULT(I)=1

J=1

GO TO 130

J=J+2

GO TO 200

100 IF(J.LT.0) GO TO 110

J=J+1

GO TO 130

110 MULT(I)=2

J=1

GO TO 130

200 RETURN

END
TABLE E.VII (Continued)

0001 SUBROUTINE NEWTON(UX, VX, UP, VP, UXO, VXO, CONV)

0002 DOUBLE PRECISION UX, VX, UP, VP, UXO, VXO, U, V
0003 DOUBLE PRECISION DDD, BBB

0004 DIMENSION UP(26), VP(26), UB, VB

0005 COMMON EPSLON, EPS3, EPS4, IO2, MAX

0006 LOGICAL CONV

0007 UXO=UX
0008 VXO=VX
0010 DO 10 I=1, MAX
0011 CALL HORNER(UXO, VXO, N, UP, VP, UB, VB)

0012 UDIFF=U*U+V*V
0013 VDIFF=V*V

0014 DDD=DSQRT(U*U+V*V)

0015 IF(DDD.EQ.0.0) GO TO 5

0016 IF(DDD.EQ.0.0) GO TO 20

0017 5 BBB=U*V
0018 UDIFF=U*U+V*V
0019 VDIFF=V*V

0020 IF(DDD.EQ.0.0) GO TO 10

0021 IF(DDD.EQ.0.0) GO TO 20

0022 10 CONTINUE

0023 15 CONV=.FALSE.

0024 20 CONV=.TRUE.

0025 RETURN

0026 END
**TABLE E.VII (Continued)**

```
SUBROUTINE DIVIDE(N,UP,VP,M,UD,VD,K,UQ,VQ)

* GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE *
* QUOTIENT POLYNOMIAL H(X) = F(X)/G(X).*

DOUBLE PRECISION UP,VP,UD,VD,UQ,VQ,UTERN,VTERM,UDUMMY

DIMENSION UP(261),VP(261),UD(261),VD(261),UQ(261),VQ(261)

K=N-M

DO 1 K=1,N

J=J+1

UTERN=UP(J-N)

VTERM=VP(J-N)

IF(J .LE. 1) GO TO 10

DO 4 K=1,N

IF(K .LE. 1) GO TO 10

IF(J .GE. 1) GO TO 20

UTERN=UTERN-UTERN*VTERM*UD(1)!

VTERM=VTERM*VTERM*UD(1)

40 K=K+1

UTERN=UTERN*UD(1)

VTERM=VTERM*VD(1)

100 RETURN

END
```
TABLE E.VII (Continued)

0001 SUBROUTINE HORNER(UX, VX, N, UP, VP, UB, VB, UC, VC)

C *
C ** HOMER'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 *
C EXECUTIVE STATEMENTS
C 0002 DOUBLE PRECISION UX, VX, UP, VP, UB, VB, UC, VC
C 0003 DOUBLE PRECISION UDUMMY, VDUMMY
C 0004 DIMENSION UP(26), VP(26), UB(26), VB(26)
C 0005 UB(N+1)=UP(N+1)
C 0006 VB(N+1)=VP(N+1)
C 0007 UB(N)=UX*UB(N+1)-VX*VB(N+1)+UP(N)
C 0008 VB(N)=UX*VB(N+1)+VX*UB(N+1)+VP(N)
C 0009 UC=UB(N+1)
C 0010 VC=VB(N+1)
C 0011 KKK=N-1
C 0012 DO 10 I=1, KKK
C 0013 UB(KKK+I-1)=(UX*UB(KKK+2-I)-VX*VB(KKK+2-I)+UP(KKK+I-1)
C 0014 VB(KKK+I-1)=(UX*VB(KKK+2-I)+VX*UB(KKK+2-I)+VP(KKK+I-1)
C 0015 UDUMMY=UX*UC-VX*VC
C 0016 VDUMMY=UX*VC-VX*UC
C 0017 UC=UDUMMY*UB(KKK+2-I)
C 0018 10 VC=VDUMMY*VB(KKK+2-I)
C 0019 RETURN
C 0020 END

0001 SUBROUTINE DERIV(N, UP, VP, UA, VA)

C *
C ** GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF
C ** ITS DERIVATIVE P'(X).
C *
C EXECUTIVE STATEMENTS
C 0002 DOUBLE PRECISION UP, VP, UA, VA, AAA
C 0003 DIMENSION UP(26), VP(26), UA(26), VA(26)
C 0004 AAA=1
C 0005 DO 10 I=2, KKK
C 0006 AAA=AAA*UP(I)
C 0007 UA(I-1)=AAA*UP(I)
C 0008 VA(I-1)=AAA*VP(I)
C 0009 M=N-1
C 0010 RETURN
C 0011 END
TABLE E.VII (Continued)

0001  SUBROUTINE MULTI(N,UP,VP,J,URD0T,VD0T,MULT)
0002  ***********************************************************************
0003  * GIVEN N ZEROS OF A POLYNOMIAL, SUBROUTINE MULTI COMPUTES THEIR *
0004  * MULTICLITIES. *
0005  **********************************************************************
0006  DOUBLE PRECISION UP,VP,URD0T,VD0T,UA,VA,UB,VB,UC,VC,EPSL,EPS2,EPS
0007  DIMENSION UP(26),VP(26),URD0T(25),VD0T(25),UA(26),VA(26),UB(26),VC
0008  COMMON EPSL,EPS2,EPS3,EPSLON,102,MAX
0009  DO 100 I=1,J
0010  M=N
0011  MULT(I)=0
0012  CALL HORNEM(URD0T(I),VD0T(I),M,UA,VA,UB,VC,VC)
0013  BBBD=DSQRT(UB(I)**2+VA(I)**2+VC(I)**2)
0014  IF(EBBBD.EQ.0) GO TO 50
0015  IF(MULT(I).EQ.0) GO TO 40
0016  GO TO 100
0017  40 WRITE(102,10001) EPSLON,1,URD0T(I),VD0T(I)
0018  GO TO 100
0019  50 MULT(I)=MULT(I)+1
0020  IF(N.GT.1) GO TO 60
0021  GO TO 100
0022  60 DO 20 K=1,M
0023  UA(K)=UB(K)+1
0024  VA(K)=VB(K)+1
0025  M=M-1
0026  GO TO 20
0027  100 CONTINUE
0028  RETURN
0029  1000 FORMAT(/'15H THE EPSILON 1.010.3,48H CHECK IN SUBROUTINE MULTI
0030  *INDICATES THAT ROOT(1.0214H = +0.23,16,3H + 0.23,10+2H 1/80H IS NO
0031  *CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP
0032  *PLICITY 0/1
0033  END
TABLE E.VII (Continued)

0001 SUBROUTINE CONSQTI(UX, VX, UY, VY)
0002 C*************************************************************************
0003 C THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
0004 C*************************************************************************
0005 DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, AAA, BBB
0006 R = DSQRT((UX*UX + VX*VX))
0007 AAA = DSQRT(DABS((R + UX)/2.0))
0008 BBB = DSQRT(DABS((R - UX)/2.0))
0009 IF(VX) 10, 20, 30
0010 10 UY = AAA
0011 20 IF(UX) 40, 50, 60
0012 30 UY = AAA
0013 40 DUMMY = DABS(UX)
0014 50 UY = 0.0
0015 60 VY = DSQRT(DUMMY)
0016 70 GO TO 100
0017 80 GO TO 100
0018 90 UY = 0.0
0019 100 VY = 0.0
0020 GO TO 100
0021 110 DUMMY = DABS(UX)
0022 120 UY = DSQRT(DUMMY)
0023 130 VY = 0.0
0024 140 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.1

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 Appendix 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 0.9\( X_0 \) and 1.1\( X_0 \).

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 ENOUGH 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 Appendix 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

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>I</td>
<td>NP</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial (P(X))</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
<td>I</td>
<td>R</td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>NOMULT</td>
<td>I</td>
<td>NOMULT</td>
<td>I</td>
<td>D</td>
<td>Number of roots (counting multiplicities)</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT,VROOT</td>
<td>D</td>
<td>R</td>
<td>Array containing the roots</td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
<td>E</td>
<td>Number of initial approximations to be read in</td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>UAPP,VAPP</td>
<td>D</td>
<td>E</td>
<td>Array of initial approximations</td>
</tr>
<tr>
<td>WORK</td>
<td>C</td>
<td>UWORK,VWORK</td>
<td>D</td>
<td>E</td>
<td>Working array containing coefficients of current polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UE,VB</td>
<td>D</td>
<td>E</td>
<td>Array containing coefficients of deflated polynomial</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA,VA</td>
<td>D</td>
<td>E</td>
<td>Array containing coefficients of original polynomial, (P(X))</td>
</tr>
<tr>
<td>RAPP</td>
<td>C</td>
<td>URAPP,VRAPP</td>
<td>D</td>
<td>E</td>
<td>Array of initial or altered approximation for which convergence was obtained</td>
</tr>
<tr>
<td>X1</td>
<td>C</td>
<td>UX1,VX1</td>
<td>D</td>
<td>E</td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>UX2,VX2</td>
<td>D</td>
<td>E</td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3,VX3</td>
<td>D</td>
<td>E</td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>PX1</td>
<td>C</td>
<td>UPX1,VPX1</td>
<td>D</td>
<td>E</td>
<td>Value of polynomial (P(X)) at (X_1)</td>
</tr>
<tr>
<td>PX2</td>
<td>C</td>
<td>UPX2,VPX2</td>
<td>D</td>
<td>E</td>
<td>Value of polynomial (P(X)) at (X_2)</td>
</tr>
<tr>
<td>PX3</td>
<td>C</td>
<td>UPX3,VPX3</td>
<td>D</td>
<td>E</td>
<td>Value of polynomial (P(X)) at (X_3)</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4,VX4</td>
<td>D</td>
<td>E</td>
<td>Newest approximation ((X_{n+1})) to root</td>
</tr>
<tr>
<td>PX4</td>
<td>C</td>
<td>UPX4,VPX4</td>
<td>D</td>
<td>E</td>
<td>Value of polynomial (P(X)) at (X_4)</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td>E</td>
<td>Array containing the multiplicities of each root found</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td>E</td>
<td>Counter for iterations</td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
<td>E</td>
<td>Unit number of input device</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>EPSRT</td>
<td>R</td>
<td>EPSRT</td>
<td>D</td>
<td>C</td>
<td>Number used in subroutine BETTER to generate two approximations from the one given</td>
</tr>
<tr>
<td>NOPOLY</td>
<td>I</td>
<td>NOPOLY</td>
<td>I</td>
<td>E</td>
<td>Number of the polynomial</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Double Precision Variable</td>
<td>Type</td>
<td>Disposition of Argument</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>---------------------------</td>
<td>------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td>C</td>
<td>Maximum number of iterations</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
<td>C</td>
<td>Tolerance check for convergence</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
<td>D</td>
<td>C</td>
<td>Tolerance check for zero (0)</td>
</tr>
<tr>
<td>EPSM</td>
<td>R</td>
<td>EPSM</td>
<td>D</td>
<td>C</td>
<td>Tolerance check for multiplicities</td>
</tr>
<tr>
<td>KCHECK</td>
<td>I</td>
<td>KCHECK</td>
<td>I</td>
<td></td>
<td>Program control, KCHECK = 1 stops execution of program</td>
</tr>
<tr>
<td>XSTART</td>
<td>R</td>
<td>XSTART</td>
<td>D</td>
<td>E</td>
<td>Magnitude at which to start generating initial approximations</td>
</tr>
<tr>
<td>XEND</td>
<td>R</td>
<td>XEND</td>
<td>D</td>
<td>E</td>
<td>Magnitude at which to end generating initial approximations</td>
</tr>
<tr>
<td>NWORK</td>
<td>I</td>
<td>NWORK</td>
<td>I</td>
<td></td>
<td>Degree of current deflated polynomial whose coefficients are in WORK</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
<td></td>
<td>Program control</td>
</tr>
<tr>
<td>NALTER</td>
<td>I</td>
<td>NALTER</td>
<td>I</td>
<td></td>
<td>Number of alterations which have been performed on an initial approximation</td>
</tr>
<tr>
<td>IAPP</td>
<td>I</td>
<td>IAPP</td>
<td>I</td>
<td></td>
<td>Counter for number of initial approximations used</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td></td>
<td>When CONV is true, convergence has been obtained</td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</td>
<td>I</td>
<td>R</td>
<td>Number of distinct roots solved by Muller's method, i.e. not solved directly by subroutine QUAD</td>
</tr>
</tbody>
</table>

Subroutine HORNER

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>UA,VA</th>
<th>D</th>
<th>E</th>
<th>Array of current polynomial coefficients (to be deflated or evaluated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>I</td>
<td>NA</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial to be deflated or evaluated</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>UX,VX</td>
<td>D</td>
<td>E</td>
<td>Approximation at which to evaluate or deflate the polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB,VB</td>
<td>D</td>
<td>R</td>
<td>Array containing the coefficients of the deflated polynomial</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>UPX,VPX</td>
<td>D</td>
<td>R</td>
<td>Value of the polynomial at X</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
<td>I</td>
<td></td>
<td>Number of coefficients of polynomial to be deflated</td>
</tr>
</tbody>
</table>
### TABLE F.II (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Single Precision</th>
<th>Disposition</th>
<th>Double Precision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3,VX3</td>
<td>D</td>
<td>E</td>
<td>Approximation to root (old) (X_n)</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4,VX4</td>
<td>D</td>
<td>E</td>
<td>New approximation to root (X_{n+1})</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td>R</td>
<td>CONV = True implies convergence has been obtained</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
<td>C</td>
<td>Tolerance for convergence test</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
<td>D</td>
<td>C</td>
<td>Tolerance check for zero (0)</td>
</tr>
<tr>
<td>DENOM</td>
<td>R</td>
<td>DENOM</td>
<td>D</td>
<td></td>
<td>Magnitude of new approximation, (X_{n+1})</td>
</tr>
</tbody>
</table>

Subroutine BETTER

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Single Precision</th>
<th>Disposition</th>
<th>Double Precision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td>ECR</td>
<td>Array of multiplicities of each root</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA,VA</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of original undeflated polynomial</td>
</tr>
<tr>
<td>NP</td>
<td>I</td>
<td>NP</td>
<td>I</td>
<td></td>
<td>Degree of original polynomial</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT,VROOT</td>
<td>D</td>
<td>ECR</td>
<td>Array of ROOTS</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
<td>I</td>
<td>ECR</td>
<td>Number of roots stored in ROOT</td>
</tr>
<tr>
<td>BAPP</td>
<td>C</td>
<td>UBAPP,VBAPP</td>
<td>D</td>
<td>E</td>
<td>Array of initial approximations (old roots)</td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</td>
<td>I</td>
<td>ECR</td>
<td>Number of roots solved by the iterative process (Not QUAD)</td>
</tr>
<tr>
<td>ROOTS</td>
<td>C</td>
<td>UROOTS,VROOTS</td>
<td>D</td>
<td>ECR</td>
<td>Temporary storage for new (better) roots</td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>L</td>
<td>I</td>
<td></td>
<td>Number of roots found by BETTER</td>
</tr>
<tr>
<td>EPSRT</td>
<td>R</td>
<td>EPSRT</td>
<td>D</td>
<td>C</td>
<td>A small number used to generate two of the three approximations when given one</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td>C</td>
<td>Counter for number of iterations</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB,VB</td>
<td>D</td>
<td>Array containing coefficients of deflated polynomial</td>
<td></td>
</tr>
<tr>
<td>XL</td>
<td>C</td>
<td>UX1,VX1</td>
<td>D</td>
<td>One of three approximations to the root</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>UX2,VX2</td>
<td>D</td>
<td>One of three approximations to the root</td>
<td></td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3,VX3</td>
<td>D</td>
<td>One of three approximations to the root</td>
<td></td>
</tr>
<tr>
<td>PX1</td>
<td>C</td>
<td>UPX1,VPX1</td>
<td>D</td>
<td>Value of polynomial (P(X)) at (X_1)</td>
<td></td>
</tr>
<tr>
<td>PX2</td>
<td>C</td>
<td>UPX2,VPX2</td>
<td>D</td>
<td>Value of polynomial (P(X)) at (X_2)</td>
<td></td>
</tr>
<tr>
<td>PX3</td>
<td>C</td>
<td>UPX3,VPX3</td>
<td>D</td>
<td>Value of polynomial (P(X)) at (X_3)</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE F.II (Continued)

<table>
<thead>
<tr>
<th>Single Precision</th>
<th>Double Precision</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
<td>Variable Type</td>
<td>of Argument</td>
<td></td>
</tr>
<tr>
<td>CONV L</td>
<td>CONV L</td>
<td></td>
<td>CONV = true implies convergence has been obtained</td>
</tr>
<tr>
<td>X4 C</td>
<td>UX4,VX4 D</td>
<td></td>
<td>Newest approximation to root</td>
</tr>
<tr>
<td>J I</td>
<td>J I</td>
<td></td>
<td>Program control — counts the number of roots used as initial approximations</td>
</tr>
<tr>
<td>MAX I</td>
<td>MAX I</td>
<td>C</td>
<td>Maximum number of iterations permitted</td>
</tr>
<tr>
<td>I02 I</td>
<td>I02 I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
</tbody>
</table>

**Subroutine ALTER**

- XI 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  ECR  Real part of complex approximation
- X2I R  X2I D  ECR  Imaginary part of complex approximation

**Subroutine CALC**

These variables are dummy variables used for temporary storage and thus, are not defined.
Figure F.1. Flow Charts for G.C.D.-Muller's Method
Figure F.1. (Continued)
Figure F.1. (Continued)
Figure F.1. (Continued)
Figure F.1. (Continued)
Figure F.1. (Continued)
Figure F.1. (Continued)
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

DOUBLE PRECISION PROGRAM FOR G.C.D.-MULLER'S METHOD

THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY
DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL
AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED
AND THEIR MULTIPLICITIES DETERMINED.

DOUBLE PRECISION URAPP,VRAPP

DIMENSION URAPP(25,3),VRAPP(25,3),UAPP(25,31),VAPP(25,31)
DIMENSION UP(361,VP(26),UROOT(25),VROOTI25),MULT(25),UDPI26IVDP(26)
DIMENSION UD(26),V22D()(26),UQ26),VQ26),UQQ126).VQQI26).UBI26),VB(26),ANAME
DIMENSION XSTART DOUBLE PRECISION EPSRT
DIMENSION XEND

COMMON EPSRTEPSI,EPS2,EPS3,EPS402i2MAX

DATA PNAME,QNAME,QQNAME/2HP(,2HQ(,3HQQI/

DATA ENTRY/IHIH2.H3,1H4,IHSelH6,1H7tIHB,H9,2H10,2HI1,2HI2.2H13
1.2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/

DATA ANAME(1),ANAME(2)4HMULL,4HERS

LOGICAL NEWT

10 J=0

10 J=0

10 J=0

10 J=0

10 J=0

READ(101,1000) NOPOLY,NP,NAPP.MAX,1EPS1,1EPS2,1EPS3,1EPS4,XSTART,XEND,

READ(101,1000) NOPOLY,NP,NAPP.MAX,1EPS1,1EPS2,1EPS3,1EPS4,XSTART,XEND,

READ(101,1000) NOPOLY,NP,NAPP.MAX,1EPS1,1EPS2,1EPS3,1EPS4,XSTART,XEND,

READ(101,1000) NOPOLY,NP,NAPP.MAX,1EPS1,1EPS2,1EPS3,1EPS4,XSTART,XEND,
TABLE F.III (Continued)

0042  DO 25 I=1,KKK
0043    JJJ=NNN-1
0044  25  WRITE(102,1040) 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(102,1070)  
0049  CALL WRITE(102,1165) (I,UROOT(I),VROOT(I),MULT(I),I=1,J)
0050  GO TO 10
0051  30  CALL DERIV(U,V,NP,UP,VP,NDP,UDP,VDP)
0052  CALL GCDINP,UP,VP,NDP,UDP,VDP,NO,UD,VD)
0053 IF(NO.EQ.0)  GO TO 70
0054  UDDUMY=UDDU1+VD(2)*VD(2)
0055  UDRD=UDU1+VD(1)*VD(2)*UDDUMY
0056  UDRD=UDU1+VD(1)*VD(2)*UDDUMY
0057  KKK=NP+1
0058  DO 55 I=1,KKK
0059  UQQ(I)=UP(I)
0060  VQQ(I)=VP(I)
0061  NQQ=NP
0062  GO TO 120
0063  CALL HORNERN(NQQ,UQQ,VQQ,UIRD,VIRD,UD,VD,UDUM,VDUM)
0064  Q=NP
0065  DO 60 I=1,NP
0066  U(I)=UP(I)
0067  V(I)=VP(I)
0068  GO TO 80
0069  KKK=NP+1
0070  DO 65 I=1,KKK
0071  U(I)=UP(I)
0072  V(I)=VP(I)
0073  NQ=NP
0074  GO TO 80
0075  70  CALL DIVIDE(NP,UP,VP,ND,UD,VD,Q,UQ,VQ)
0076  WRITE(102,120)  
0077  KKK=Q
0078  NNN=KKK+1
0079  DO 83 I=1,NNN
0080  JJJ=NNN-1
0081  83  WRITE(102,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
0082 IF(NQ.GE.3)  GO TO 85
0083  GO TO 110
0084  85  KKK=NP
0085  DO 90 I=1,KKK
0086  UQQ(I)=UP(I)
0087  VQQ(I)=VP(I)
0088  NQ=NP
0089  GO TO 120
0090  110  CALL QUAD(NQ,UQ,VQ,J,UROOT,VROOT,MULT)
0091 NEWT=.FALSE.
0092  GO TO 310
0093  CALL MULLER(UQ,VQ,NQ,UAPP,VAPP,NAP,NAPP,START,XEND,UROOT,VROOT,J,J
0094  1AP,URAPP,VRAPP,NOPOLY)
0095  NEWT=.TRUE.
0096  310  CALL MULLER(NP,UP,VP,J,UROOT,VROOT,MULT)
0097 IF(NEWT)  GO TO 330
0098  WRITE(102,1070)  
0099
TABLE F.III (Continued)

0090  GO TO 10
0100  330 WRITE(102,1180)
0101  DO 350 L=1,JAP
0102  350 WRITE(102,1190) L,UROOT(L),VRROOT(L),MULTIL,L,URAPP(L),VRAPP(L)
0103  KKK=JAP+1
0104  INT(JAP,L,J) WRITE(102,1165) L,UROOT(L),VRROOT(L),MULTIL,L=KKK,J
0105  GO TO 10
0106  1000 FORMAT(12,1X,9X,13,1X,4(D6.0,1X),13X,2(D7.0,1X),11)
0107  1010 FORMAT(2030.0
0108  1015 FORMAT(2D30.0)
0109  1020 FORMAT(1H,1OX,4L30.0 METHOD TO FIND ZEROS OF POLYNOMIALS
0110  1030 FORMAT(1X,22HTHE DEGREE OF P(X) IS 12,22H THE COEFFICIENTS ARE//
0111  1040 FORMAT(2X,A2,A2,4H) = .023,16,3H + .023,16,6H II
0112  1070 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES//)
0113  1080 FORMAT(2X,5HROOT1,12,4H) = .023,16,3H + .023,16,2H 1,10X,12
0114  1100 FORMAT(2X,4H,A2,A2,4H) = .023,16,3H + .023,16,2H II
0115  1120 FORMAT(///1X,73H(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE
0116  1165 FORMAT(2X,5HROOT1,12,4H) = .023,16,3H + .023,16,2H II 1,7X,12,10X,26H
0117  1180 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21INITIAL
0118  1190 FORMAT(2X,5HROOT1,12,4H) = .023,16,3H + .023,16,2H 1,7X,12,9X,023,
0119  2000 FORMAT(1X,4HNUMBER OF INITIAL APPROXIMATIONS GIVEN.//
0120  2010 FORMAT(1X,24HMULTIPLICITIES,11X,13)
0121  2020 FORMAT(1X,21HTEST FOR CONVERGENCE,13X,09.2)
0122  2030 FORMAT(1X,24HMULTIPLICITIES,10X,09.2)
0123  2040 FORMAT(1X,23HRADIUS TO START SEARCH,11X,09.2)
0124  2050 FORMAT(1X,21HRADIUS TO END SEARCH,13X,09.2)
0125  2060 FORMAT(1X)
0126  2070 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE GCD.//09.2)
0127  2080 FORMAT(1X,34HTEST FOR ZERO IN SUBROUTINE QUAD.//09.2)
0128  END
TABLE F.III (Continued)

0001 SUBROUTINE MULTI(UP,VP,J,URoot,VRoot,Mult)
0002 C **********************************************************************
0003 C * GIVEN N ZEROS OF A POLYNOMIAL, SUBROUTINE MULTI COMPUTES THEIR *
0004 C * MULTIPLEITIES.                                                      *
0005 C **********************************************************************
0006 DOUBLE PRECISION UP,VP,URoot,VRoot,UA,VA,UB,VB,UC,VC,EPS1,EPS2,EPS
0007 ILON,EPS3,EPSL
0008 DIMENSION UP(26),VP(26),URoot(251),VRoot(251),UA(261),VA(261),UB(261),VC(261)
0009 COMMON EPSRT,EPSL,EPS2,EPS3,EPSL0,MAX
0010 DO 10 I=1,J
0011 KKK=N+I
0012 DO 10 K=1,KKK
0013 UA(K)=UP(KKK+1-K)
0014 VA(K)=VP(KKK+1-K)
0015 M=N
0016 MULT(I)=0
0017 20 CALL HORNER(UP,VA,URoot,VRoot,U,V,EPSL0)
0018 BRB=DABS(U*UC+VC*VCI)
0019 IF(MULT(I).EQ.0) GO TO 50
0020 IF(BRB.EQ.0.0) GO TO 40
0021 DO 30 K=1,M
0022 UA(K)=UB(K)
0023 VA(K)=V8(K)
0024 M=M-1
0025 GO TO 20
0026 50 MULT(I)=MULT(I)+1
0027 IF(M.GT.11) GO TO 60
0028 40 WRITE(102,1000) EPSL0,URoot(I),VRoot(I)
0029 GO TO 100
0030 60 GO TO 100
0031 100 CONTINUE
0032 RETURN
0033 1000 FORMAT(/15H THE EPSLON (,D010.3,48HI CHECK IN SUBROUTINE MULTI
0034 INDICATES THAT ROOT(I,12,4H) = ,D23.16,3H + ,D23.16,3H + ,D23.16,3H + ,D23.16,2H I,120H IS NO
0035 T CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP
0036 LICITY 0//)
0037 END
TABLE F.III (Continued)

0001  SUBROUTINE DIVIDE(N,UP,VP,M,UD,VD,K,UG,VQ)
                C  ******************************************************************************************
                C  * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE           *
                C  * QUOTIENT POLYNOMIAL Q(X) = F(X)/G(X).                                         *
                C  ******************************************************************************************
0002  DOUBLE PRECISION UP,VP,UD,VD,UG,VQ,UTERM,UDUMMY
0003  DIMENSION UP(126),VP(126),UD(26),VD(26),UG(26),VQ(26)
0004  K=N-M
0005  UDUMMY=UD(N+1)*UD(M+1)+VD(N+1)
0006  UG(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(10-J)
0013  VTERM=VP(10-J)
0014  KK=K+1
0015  KNN=M-J
0016  DO 40 M=NNN,M
0017  IF(KK.GT.1) GO TO 10
0018  GO TO 45
0019  10 IF(FLG.LE.1) GO TO 20
0020  GO TO 40
0021  20 UTERM=UTERM+UQ(KK)*UD(M+1)-VQ(KK)*VD(M+1)
0022  VTERM=VTERM-UQ(KK)*VD(M+1)+UQ(KK)*UD(M+1)
0023  40 KK=KK-1
0024  IF(KK.LT.1) GO TO 40
0025  UQ(K+1)=UQ(K)+UTERM*UD(M+1)+VTERM*VD(M+1)/UDUMMY
0026  VQ(K+1)=VQ(K)-UTERM*VD(M+1)-UQ(K)*UD(M+1)/UDUMMY
0027  50 RETURN
0028  100 END

0001  SUBROUTINE DERIV(N,UP,VP,M,VA,VA)
                C  ******************************************************************************************
                C  * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF        *
                C  * ITS DERIVATIVE P'(X).                                                          *
                C  ******************************************************************************************
0002  DOUBLE PRECISION UP,VP,VA,AAA
0003  DIMENSION UP(126),VP(126),VA(126),AAA(26)
0004  KKK=N+1
0005  DO 10 I=2,KKK
0006  AAA(I-1)=AAA*UP(I)
0007  UAI(I)=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(IN,UR,VR,M,US,VS,M1,US$1,VS$1)

0002 C

0003 C******************************************************************************

0004 C

0005 C * GIVEN POLYNOMIALS P(X) AND Q(X) WHERE DEG. P(X) IS LESS THAN DEG. Q(X)

0006 C * P(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND

0007 C * Q(X).

0008 C

0009 C******************************************************************************

0010 C

0011 C

0012 C************************************************************************************

0013 C

0014 DO 25 M=1,KKK

0015 C

0016 DO 25 US$1=US$1

0017 C

0018 DO 25 VSS$1=VSS$1

0019 C

0020 DO 25 J=1,KKK

0021 C

0022 DO 25 UT=UT

0023 C

0024 IF(M.EQ.0) GO TO 70

0025 C

0026 DO 60 I=1,3

0027 C

0028 DO 60 UT(1)=UT(1)

0029 C

0030 DO 60 UT(1)=UT(1)+1

0031 C

0032 DO 60 UT(1)=UT(1)-1

0033 C

0034 DO 60 UT(I)=UT(I)

0035 C

0036 DO 60 UT(I)=UT(I)-1

0037 C

0038 DO 60 UT(I)=UT(I)+1

0039 C

0040 DO 60 UT(I)=UT(I)

0041 C

0042 IF(M.EQ.1) GO TO 70

0043 C

0044 IF(M.EQ.0) GO TO 70

0045 C

0046 IF(M.EQ.1) GO TO 70

0047 C

0048 IF(M.EQ.1) GO TO 70

0049 C

0050 C

0051 C

0052 C

0053 C

0054 C

0055 C

0056 C

0057 C

0058 C

0059 C

0060 C
TABLE F.III (Continued)

0050  GO TO 30
0051  KKK=K+1
0052  DO 150 J=1,KKK
0053  URR(J)=USSI(J)
0054  VRR(J)=VSS(J)
0055  USSI(J)=UTI(J)
0056  VSS(J)=VT(J)
0057  KKK=K+2
0058  NNN=M1+1
0059  DO 160 J=KKK,NNN
0060  URR(J)=USSI(J)
0061  VRR(J)=VSS(J)
0062  M1=M1
0063  M1=K
0064  GO TO 30
0065  USSI(J)=1.0
0066  VSS(J)=0.0
0067  M1=0
0068  200 RETURN
0069  END
TABLE F.III (Continued)

0001  SUBROUTINE QUAD(Y,UA,VJ,URoot,VRoot,MULT)
0003                                                                                               *
0004                                                                                               *
0006                                                                                               *
0007                                                                                               *
0008                                                                                               *
0010                                                                                               *
0012                                                                                               *
0014                                                                                               *
0016                                                                                               *
0017                                                                                               *
0018                                                                                               *
0020                                                                                               *
0021                                                                                               *
0022                                                                                               *
0023                                                                                               *
0025                                                                                               *
0026                                                                                               *
0027                                                                                               *
0028                                                                                               *
0029                                                                                               *
0030                                                                                               *
0031                                                                                               *
0032                                                                                               *
0033                                                                                               *
0034                                                                                               *
0035                                                                                               *
0036                                                                                               *
0037                                                                                               *
0038                                                                                               *
0039                                                                                               *
0040                                                                                               *
0041                                                                                               *
0042                                                                                               *
0043                                                                                               *
0044                                                                                               *
0045                                                                                               *

0001  SUBROUTINE QUAD(Y,UA,VJ,URoot,VRoot,MULT)
0003                                                                                               *
0004                                                                                               *
0006                                                                                               *
0007                                                                                               *
0008                                                                                               *
0010                                                                                               *
0012                                                                                               *
0014                                                                                               *
0016                                                                                               *
0017                                                                                               *
0018                                                                                               *
0020                                                                                               *
0021                                                                                               *
0022                                                                                               *
0023                                                                                               *
0025                                                                                               *
0026                                                                                               *
0027                                                                                               *
0028                                                                                               *
0029                                                                                               *
0030                                                                                               *
0031                                                                                               *
0032                                                                                               *
0033                                                                                               *
0034                                                                                               *
0035                                                                                               *
0036                                                                                               *
0037                                                                                               *
0038                                                                                               *
0039                                                                                               *
0040                                                                                               *
0041                                                                                               *
0042                                                                                               *
0043                                                                                               *
0044                                                                                               *
0045                                                                                               *
MULLER’S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE QUADRATIC CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION. IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.

DOUBLE PRECISION UAPP, VPAPP, UAPPX, VPAPPX, UAPPX2, VPAPPX2, UAPPX3, VPAPPX3
DOUBLE PRECISION EPSR, EPSI
LOGICAL CONV
COMMON EPSR, EPSI, EPSM, Z02, AX
DATA PNAME, DNAME / 'HP', 'HD'/
EPSRT = 0.999
TABLE F.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0044</td>
<td>IF(ABPX3.EQ.0.0) GO TO 70</td>
</tr>
<tr>
<td>0045</td>
<td>QQ=ABPX4/ABPX3</td>
</tr>
<tr>
<td>0046</td>
<td>IF(QQ.LE.10.) GO TO 70</td>
</tr>
<tr>
<td>0047</td>
<td>UQ4=0.5*UQ4</td>
</tr>
<tr>
<td>0048</td>
<td>VX4=VX3+(VH3<em>UQ4+VH3</em>VQ4)</td>
</tr>
<tr>
<td>0051</td>
<td>GO TO 60</td>
</tr>
<tr>
<td>0052</td>
<td>CALL TEST(UX3,VX3,UX4,VX4,CONVI)</td>
</tr>
<tr>
<td>0053</td>
<td>IF(CONVI GO TO 120</td>
</tr>
<tr>
<td>0054</td>
<td>IF(ITER.LT.MAX) GO TO 110</td>
</tr>
<tr>
<td>0055</td>
<td>CALL ALTER(UAPP1),VAPP1(UAPP1),VAPP2(UAPP2),UAP</td>
</tr>
<tr>
<td>0057</td>
<td>IF(NALTER.GT.5) GO TO 75</td>
</tr>
<tr>
<td>0058</td>
<td>ITER=1</td>
</tr>
<tr>
<td>0059</td>
<td>GO TO 40</td>
</tr>
<tr>
<td>0060</td>
<td>IF(IAPP.LT.NAPP) GO TO 100</td>
</tr>
<tr>
<td>0061</td>
<td>IF(XSTART.GT.XEND) GO TO 77</td>
</tr>
<tr>
<td>0062</td>
<td>NAPP=NP</td>
</tr>
<tr>
<td>0063</td>
<td>CALL GENAPP(UAPP1,VAPP1,NAPP,XSTART)</td>
</tr>
<tr>
<td>0064</td>
<td>IAPP=0</td>
</tr>
<tr>
<td>0065</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0066</td>
<td>77 WRITE(102,1090)</td>
</tr>
<tr>
<td>0067</td>
<td>KKK=NWORK+1</td>
</tr>
<tr>
<td>0068</td>
<td>WRITE(102,1035) (DONAME,J,UWORK(J),VWORK(J),J=1,KKK)</td>
</tr>
<tr>
<td>0069</td>
<td>80 IF(IAPP.LT.MAX) GO TO 90</td>
</tr>
<tr>
<td>0070</td>
<td>IF(IPATH.EQ.1) GO TO 82</td>
</tr>
<tr>
<td>0071</td>
<td>81 IPATH=2</td>
</tr>
<tr>
<td>0072</td>
<td>CALL BETTER(UA,VA,VA,URoot,VRoot,NRoot,VRAPP,VRAPP,VRAPP,Mult)</td>
</tr>
<tr>
<td>0073</td>
<td>RETURN</td>
</tr>
<tr>
<td>0074</td>
<td>82 IF(NROOT.EQ.0) GO TO 90</td>
</tr>
<tr>
<td>0075</td>
<td>83 IF(IROOT.EQ.1) GO TO 85</td>
</tr>
<tr>
<td>0076</td>
<td>WRITE(102,1080)</td>
</tr>
<tr>
<td>0077</td>
<td>DO 85 I=1,IROOT</td>
</tr>
<tr>
<td>0078</td>
<td>55 WRITE(102,1085) I,URoot(I),VRoot(I),URAPP(I,1),VRAPP(I,1),VRAPP(I,2)</td>
</tr>
<tr>
<td>0079</td>
<td>IF(IROOT.LT.NRoot) GO TO 85</td>
</tr>
<tr>
<td>0080</td>
<td>GO TO 87</td>
</tr>
<tr>
<td>0081</td>
<td>85 KK=IROOT=1</td>
</tr>
<tr>
<td>0082</td>
<td>WRITE(102,1086) I,URoot(I),VRoot(I),I=KK,NRoot</td>
</tr>
<tr>
<td>0083</td>
<td>87 IF(IPATH.EQ.1) GO TO 81</td>
</tr>
<tr>
<td>0084</td>
<td>RETURN</td>
</tr>
<tr>
<td>0085</td>
<td>90 WRITE(102,1070) NRoot</td>
</tr>
<tr>
<td>0086</td>
<td>RETURN</td>
</tr>
<tr>
<td>0087</td>
<td>100 IAPP=IAPP+1</td>
</tr>
<tr>
<td>0088</td>
<td>ITER=1</td>
</tr>
<tr>
<td>0089</td>
<td>NALTER=0</td>
</tr>
<tr>
<td>0090</td>
<td>GO TO 40</td>
</tr>
<tr>
<td>0091</td>
<td>120 NRoot=NRoot+1</td>
</tr>
<tr>
<td>0092</td>
<td>IRoot=IRoot+1</td>
</tr>
<tr>
<td>0093</td>
<td>MULT=MULT+1</td>
</tr>
<tr>
<td>0094</td>
<td>NMultiple=NMultiple+1</td>
</tr>
<tr>
<td>0095</td>
<td>URoot(NRoot)=UX4</td>
</tr>
<tr>
<td>0096</td>
<td>VRoot(NRoot)=VX4</td>
</tr>
<tr>
<td>0097</td>
<td>URAPP(NRoot,1)=URAPP(IAPP,1)</td>
</tr>
<tr>
<td>0098</td>
<td>VRAPP(NRoot,1)=VRAPP(IAPP,1)</td>
</tr>
<tr>
<td>0099</td>
<td>URAPP(NRoot,2)=URAPP(IAPP,2)</td>
</tr>
<tr>
<td>1000</td>
<td>VRAPP(NRoot,2)=VRAPP(IAPP,2)</td>
</tr>
</tbody>
</table>
TABLE F.III (Continued)

```plaintext
244
URAPP(NROOT,3)=UAPP(IAPP,3)
VRAPP(NROOT,3)=VAPP(IAPP,3)
125 IF(NMULT.LT.NPI) GO TO 130
DO TO 80
130 CALL HORNER(NWORK, UWORK, VX4, UB, VB, UPX4, VPX4)
   NWORK=NWORK-1
   KKK=NWORK+1
   DO 140 I=1, KKK
   UWORK(I)=UB(I)
   VWORK(I)=VB(I)
   CALL HORNER(NWORK, UWORK, VX4, UB, VB, UPX4, VPX4)
   CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
   IF(CCC.LT.EPSM) GO TO 150
   IF(INWORK.GT.2) GO TO 75
   IROOT=NROOT
   KKK=NWORK+1
   DO 145 I=1, KKK
   UB(I)=UWORK(KKK+1-I)
   VB(I)=VWORK(KKK+1-I)
   CALL QUAD(NWORK, UB, VB, NROOT, UROOT, VROOT, MULT)
   GO TO 80
   150 MULT(NROOT)=MULT(NROOT)+1
   121 GO TO 125
   110 UX1=UX2
   VX1=VX2
   UX2=UX3
   VX2=VX3
   UX3=UX4
   VX3=VX4
   UPX1=UPX2
   VPX1=VPX2
   UPX2=UPX3
   VPX2=VPX3
   UPX3=UPX4
   VPX3=VPX4
   ITER=ITER+1
   GO TO 50
   1090 FORMAT(///,1X,65COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
   ZEROS WERE FOUND///)
   1091 FORMAT(///,1X,13ROOTS OF Q(X)=2.0 INITIAL APPROXIMATION///)
   1070 FORMAT(///,4H NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER 121)
   1086 FORMAT(///,12,4H = .023.16,3H + .023.16,2H 1,19X,2HSOLVED
   1 BY DIRECT METHOD))
   1035 FORMAT(///,12,4H = .023.16,3H + .023.16,2H 1)
   1050 FORMAT(///,12,4H = .023.16,3H + .023.16,2H 1/82X,023.16,3H + .023.16,2H 1)
   1045 FORMAT(///,12,4H = .023.16,3H + .023.16,2H 1,18X,023.16,3H
   1 + .023.16,2H 1)
   END
```
TABLE F.III (Continued)

0001 SUBROUTINE BETTER(UA,VA,NP,UROOT,VROOT,URAPPI,VRAPP,ROOT,MUL)
0002 ***************************************************************************
0003 SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND 
0004 BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO 
0005 THE FULL, UNDEFLATED POLYNOMIAL.
0006 ***************************************************************************
0007 DOUBLE PRECISION UROOT,VROOT,UA,VA,UBAPP,VBAPP,UX1,UX2,UX3,UX4
0008 UX3,UPX1,UPX2,VPX2,VPX3,VPX4,UB,VB,UROOTS,ROOTS,EPSRT,EPS,EPSM
0009 2.0,EPSRT
0010 LOGICAL CONV
0011 DIMENSION UROOT(25),VROOT(25),UARRAY(25,3),VRAPP(25,3)
0012 MULT,MAX
0013 DOUBLE PRECISION EPS1,EPSM
0014 COMMON EPSRT,EPS1,EPS,EPSM,IO2,MAX
0015 IF(NROOT.LE.11) RETURN
0016 L=0
0017 DO 10 I=1,NROOT
0018 UBAPP(I,1)=UROOT(I)*EPSRT
0019 VBAPP(I,1)=VROOT(I)*EPSRT
0020 UBAPP(I,2)=UROOT(I)
0021 VBAPP(I,2)=VROOT(I)
0022 UBAPP(I,3)=UROOT(I)*2.0-EPSRT)
0023 10 VBAPP(I,3)=VROOT(I)*2.0-EPSRT)
0024 DO 100 J=1,NROOT
0025 UX1=UBAPP(J,1)
0026 UX2=UBAPP(J,2)
0027 UX3=UBAPP(J,3)
0028 UX4=UX1*UX2*UX3
0029 I=1
0030 IF(ITER.LT.MAX) GO TO 40
0031 WRITE(IO2,1000) UROOT(J),VROOT(J)
0032 WRITE(IO2,100) UX4
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 34 K=KKK
0038 UBAPP(K,1)=URAPP(K-1,1)
0039 VRAPP(K,1)=URAPP(K-1,1)
0040 UBAPP(K,2)=URAPP(K-1,2)
0041 VRAPP(K,2)=URAPP(K-1,2)
0042 UBAPP(K,3)=URAPP(K-1,3)
0043 VRAPP(K,3)=URAPP(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     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(I,12,4H I = ,
0071     1023.16,3H + \(0.023.16,2H I/24H DID NOT CONVERGE AFTER ,13,11H ITERAT
0072     ZIONS))
0073   1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,023.16,3H + \(0.023.16,2H 1/
0074     1/1)
0075   END
TABLE F.III (Continued)

SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)

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

* SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO
* CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.

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

DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA
COMMON EPS1,EPS2,EPS3,EPS4,EPS5,IO2,MAX

IF(ITIME.NE.0) GO TO 5
ITIME=1
WRITE(IO2,1010) MAX
IF(INALTER.EQ.0) GO TO 10
WRITE(IO2,1000) X1R,X1I,X2R,X2I,X3R,X3I
GO TO 20

10 R=DSQRT(X2R*X2R+X2I*X2I)
BETA=DATAN2(X2I,X2R)
WRITE(IO2,1020) X1R,X1I*X2R*X3I*X3I
NALTER=NALTER+1
IF(INALTER.GT.5) RETURN
GO TO 1300

30 X2R=-X2R
X2I=-X2I
GO TO 50

40 BETA=BETA+1.047196
X2R=R*DCOS(BETA)
X2I=R*DSIN(BETA)
GO TO 50

50 X1R=0.9*X2R
X1I=0.9*X2I
X3R=1.1*X2R
X3I=1.1*X2I
RETURN

1000 FORMAT(1X,5H1X=S16.3H + S16.2H I,10X,22HALTERED APPROXIM
ATIONS/X,5H1X=S16.3H + S16.2H I/1X,S16.3H + Z+S16.2H I/1
1020 FORMAT(1H1X=S16.3H + S16.2H I/1X,S16.3H + Z+S16.2H I/1
1010 FORMAT(1X,5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
ITER I=12 I=12H ITERATIONS///)
END
**TABLE F.III (Continued)**

```fortran
SUBROUTINE GENAPP(APPR, APP1, NAPP, XSTART)

C ***********************************************************************
C * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
C * DEGREE OF THE ORIGINAL POLYNOMIAL.                                  *
C ***********************************************************************

DOUBLE PRECISION APPR, APP1, EPS1, EPS2, EPS3, EPS4, EPS5, EPS6
DIMENSION APPR(25,3), APP1(25,3)
COMMON EPSRTEPS1EIEPS2, EPS3, EPS4, EPS5, EPS6

IF (XSTART.EQ.0.0) XSTART=0.5

DO 10 I=1, NAPP
   APPRI(I,2)=XSTART*DCOS(BETA)
   APPPI(I,2)=XSTART*DSIN(BETA)
   BETA=BETA+0.5235988
   XSTART=XSTART+0.5
   DO 20 I=1, NAPP
      APPRI(I,3)=1.1*APPRI(I,2)
      APPPI(I,3)=1.1*APPPI(I,2)

RETURN
END
```
TABLE F.III (Continued)

0001 SUBROUTINE TEST(UX3,VX3,UX4,VX4,CONV)

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))
C WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C **********************************************************************

0002 DOUBLE PRECISION UX3,VX3,UX4,VX4,EPSRT,EPSO,EPS,EPSM,AAA,UDUMMY,VDUMMY

0003 LOGICAL CONV

0004 COMMON EPSRT,EPS,EPSM

0005 EPSRT=EPS*EPS

0006 EPSM=EPSM*EPSM

0007 UDUMMY=UX4-UX3

0008 VDUMMY=VX4-VX3

0009 AAA=DSQRT(UDUMMY*UDUMMY+VDUMMY*VDUMMY)

0010 DENOM=DSQRT(UX4*UX4+VX4*VX4)

0011 IF(AAA/DENOM.LT.EPSO) GO TO 20

0012 CONV=.FALSE.

0013 GO TO 100

0014 IF(AAA.DT.EPSO) GO TO 5

0015 100 RETURN

END

0011 SUBROUTINE HORNER(NA,UA,VA,UX,VX,UB,VB,UPX,VPX)

C **********************************************************************
C HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT 0.
C SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE
C FACTOR (X-D).
C **********************************************************************

0002 DOUBLE PRECISION UX,VX,UPX,VPX,UB,VB,UA,VA

0003 DIMENSION UAI(26),VAI(26),UBI(26),VBI(26)

0004 UBI=UA(1)

0005 VBI=VA(1)

0006 NUM=NA+1

0007 DO 10 I=2,NUM

0008 UBI=UAI(I)+UBI(I)*UX-VBI(I-1)*VX

0009 VBI=VAI(I)+VBI(I)*UX-UAI(I-1)*VX

0010 UPX=UAI(NUM)

0011 VPX=VBI(NUM)

0012 RETURN

END
TABLE F.11 (Continued)

0001  SUBROUTINE CALC(UX1, UX2, VX1, VX2, VX3, VX4, UPP1, UPP2, UPP3, UPP4, VX3, V)

0002  DOUBLE PRECISION ARG1, ARG2

0003  DOUBLE PRECISION UPX3, UPX4, UPX5, UX1, UX2, UX3, VX1, VX2, VX3, VX4, VX5, VQ3, UO, VO, U, UB, UC, UD, UDIAG, UDSC, VDISC, UCC, VCC, UDEN, UDEN1, UDEN2, VDEN, VDEN1, U, VD, VQ, VX, VW, EPRT, EPS, EPS2, IO2, MAX

0004  DOUBLE PRECISION EPSRT, EPS1, EPS2, EPS3, EPSM, EPSL, EPSM, IO2, MAX

0005  COMMON EPSRT, EPS1, EPS2, EPS3, EPSL, EPSM, IO2, MAX

0006  COMMON EPSRT, EPS1, EPS2, EPS3, EPSL, EPSM, IO2, MAX

0007  UH3 = UX3 - UX2

0008  UX3 = VX3 - UX2

0009  UX2 = UX2 - UX1

0010  VX2 = VX2 - VX1

0011  UB = UX2 + UX2 + UX2 + VX2 + VX2

0012  UO3 = UX3 + UX3 + UX3 + VX3 + VX3 + VX3

0013  UO3 = UX3 + UX3 + UX3 + VX3 + VX3 + VX3

0014  UO3 = UX3 + UX3 + UX3 + VX3 + VX3 + VX3

0015  UD = UPX3 + UPX2 + UPX1 + VQ3

0016  UD = UPX3 + UPX2 + UPX1 + VQ3

0017  UD = UPX3 + UPX2 + UPX1 + VQ3

0018  UA = 2.0 * UO3

0019  UAAA = 2.0 * UO3

0020  UAAA = 2.0 * UO3

0021  UAAA = 2.0 * UO3

0022  UAAA = 2.0 * UO3

0023  VCC = UCC + UO3 - UO3 + UO3

0024  VCC = UCC + UO3 - UO3 + UO3

0025  VBD = (VAAA + UPX3 + VX4) - (UBBB + UPX2 + UBBB + VX2) + (UCCC + UPX1 + UCCC + V)

0026  VBD = (VAAA + UPX3 + VX4) - (UBBB + UPX2 + UBBB + VX2) + (UCCC + UPX1 + UCCC + V)

0027  UCC = UD + UX3 - UPPP + UPX3

0028  UCC = UD + UX3 - UPPP + UPX3

0029  UCC = UD + UX3 - UPPP + UPX3

0030  UDSC = (UPPP + UPPP + UPPP + VV) - (4.0 * (UD * UC - UD * UC))

0031  VDISC = (VPPP + VPPP + VPPP + VPPP) - (4.0 * (VPPP + VPPP + VPPP + VPPP))

0032  AAA = UPPP + VPPP + VPPP + VPPP

0033  IF (AAA.EQ.0.0) GO TO 5

0034  GO TO 7

0035  IF (AAA.EQ.0.0) GO TO 5

0036  5 THETA = 0.0

0037  GO TO 9

0038  7 THETA = ATAN2 (VDISC, UDSC)

0039  9 RAD = DSRQ (AAA)

0040  0.0

0041  0.0

0042  0.0

0043  0.0

0044  0.0
TABLE F.III (Continued)

0045  \( \text{ARG1} = \text{UDEN1} \times \text{UDEN1} + \text{VDEN1} \times \text{VDEN1} \)
0046  \( \text{ARG2} = \text{UDEN2} \times \text{UDEN2} + \text{VDEN2} \times \text{VDEN2} \)
0047  \( \text{AAA} = \text{DSQRT1(ARGS1)} \)
0048  \( \text{BBB} = \text{DSQRT1(ARGS2)} \)
0049  IF(\( \text{AAA} \lt \text{BBB} \)) GO TO 10
0050  IF(\( \text{AAA} = \text{EQ} 0.0 \)) GO TO 60
0051  \( \text{UAAA} = -2.0 \times \text{UC} \)
0052  \( \text{VAAA} = -2.0 \times \text{VC} \)
0053  \( \text{UQ4} = \left\{ \text{UAAA} \times \text{UDEN1} + \text{VAAA} \times \text{VDEN1} \right\} / \text{ARG1} \)
0054  \( \text{VQ4} = \left\{ \text{VAAA} \times \text{UDEN1} - \text{UAAA} \times \text{VDEN1} \right\} / \text{ARG1} \)
0055  GO TO 50
0056  10 IF(\( \text{BBB} \lt \text{EQ} 0.0 \)) GO TO 60
0057  \( \text{UAAA} = -2.0 \times \text{UC} \)
0058  \( \text{VAAA} = -2.0 \times \text{VC} \)
0059  \( \text{UQ4} = \left\{ \text{UAAA} \times \text{UDEN2} + \text{VAAA} \times \text{VDEN2} \right\} / \text{ARG2} \)
0060  \( \text{VQ4} = \left\{ \text{VAAA} \times \text{UDEN2} - \text{UAAA} \times \text{VDEN2} \right\} / \text{ARG2} \)
0061  GO TO 50
0062  50 \( \text{UX4} = \text{UX3} + (\text{VH3} \times \text{UQ4} - \text{UH3} \times \text{VQ4}) \)
0063  \( \text{VX4} = \text{VX3} + (\text{VH3} \times \text{UQ4} + \text{UH3} \times \text{VQ4}) \)
0064  RETURN
0065  60 \( \text{UQ4} = 1.0 \)
0066  \( \text{VQ4} = 0.0 \)
0067  GO TO 50
0068  END
TABLE F. III (Continued)

```plaintext
C "SUBROUTINE CONSQT(UX, VX, UY, VY)

C ************************************************************************************
C *
C * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
C *
C ************************************************************************************

DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, AAA, BBB

R = DSQRT(UX*UX + VX*VX)
AAA = DSQRT(DABS((R+UX)/2.0))
BBB = DSQRT(DABS((R-UX)/2.0))

IF (UX) 10, 20, 30
10  UY = AAA
20  VY = -1.0*BBB
30  GO TO 100
40  DUMMY = DABS(UX)
50  UY = 0.0
60  VY = DSQRT(DUMMY)
70  GO TO 100
80  UY = 0.0
90  VY = 0.0
100 RETURN
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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAPP(N)</td>
<td>VAPP(N)</td>
</tr>
<tr>
<td>UDDO(N+1)</td>
<td>VDDO(N+1)</td>
</tr>
<tr>
<td>UDL(N+1)</td>
<td>VDL(N+1)</td>
</tr>
<tr>
<td>UD2(N+1)</td>
<td>VD2(N+1)</td>
</tr>
<tr>
<td>UDD1(N+1)</td>
<td>VDD1(N+1)</td>
</tr>
<tr>
<td>UG(N+1)</td>
<td>VG(N+1)</td>
</tr>
<tr>
<td>UD3(2N+1)</td>
<td>VD3(2N+1)</td>
</tr>
<tr>
<td>UD4(2N+1)</td>
<td>VD4(2N+1)</td>
</tr>
<tr>
<td>UZROS(N)</td>
<td>VZROS(N)</td>
</tr>
<tr>
<td>UAP(N)</td>
<td>VAP(N)</td>
</tr>
<tr>
<td>UROOT(N)</td>
<td>VROOT(N)</td>
</tr>
<tr>
<td>NULT(N)</td>
<td>ENTRY(N+1)</td>
</tr>
</tbody>
</table>

Subroutine PROD

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UH(2N+1)</td>
<td>VH(2N+1)</td>
</tr>
<tr>
<td>UF(N+1)</td>
<td>VF(N+1)</td>
</tr>
<tr>
<td>UG(N+1)</td>
<td>VG(N+1)</td>
</tr>
</tbody>
</table>

Subroutine ZROS

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAPP(N)</td>
<td>VAPP(N)</td>
</tr>
<tr>
<td>UROOT(N)</td>
<td>VROOT(N)</td>
</tr>
<tr>
<td>UQ(N+1)</td>
<td>VQ(N+1)</td>
</tr>
<tr>
<td>UQQ(N+1)</td>
<td>VQQ(N+1)</td>
</tr>
<tr>
<td>UAP(N)</td>
<td>VAP(N)</td>
</tr>
<tr>
<td>UQD(N+1)</td>
<td>VQD(N+1)</td>
</tr>
<tr>
<td>ENTRY(N+1)</td>
<td>UROOTS(N), VROOTS(N)</td>
</tr>
</tbody>
</table>

Subroutines GENAPP, GCD, NEWTON, DIVIDE, HORNER, and DERIV

See corresponding subroutine in Table E.I.

Subroutine QUAD

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UROOT(N)</td>
<td>VROOT(N)</td>
</tr>
<tr>
<td>UA(N+1)</td>
<td>VA(N+1)</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>N</th>
<th>O</th>
<th>P</th>
<th>MAX</th>
<th>EPS1</th>
<th>EPS2</th>
<th>EPS3</th>
<th>XSTART</th>
<th>XEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>7</td>
<td>200</td>
<td>1.D-03</td>
<td>1.D-10</td>
<td>1.D-20</td>
<td>1.0D+01</td>
<td>2.0D+01</td>
</tr>
</tbody>
</table>
### TABLE G.II
**REPEATED GCD - NEWTON'S METHOD**

<table>
<thead>
<tr>
<th>Single Precision</th>
<th>Double Precision</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
<td>Variable Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KD I</td>
<td>KD I</td>
<td></td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>K I</td>
<td>K I</td>
<td></td>
<td>Number of roots found</td>
</tr>
<tr>
<td>J1 I</td>
<td>J1 I</td>
<td></td>
<td>Multiplicity of given root</td>
</tr>
<tr>
<td>D0 C</td>
<td>UD0,VD0 D</td>
<td></td>
<td>Array of coefficients of original polynomial</td>
</tr>
<tr>
<td>ND0 I</td>
<td>ND0 I</td>
<td></td>
<td>Degree of original polynomial</td>
</tr>
<tr>
<td>DDD0 C</td>
<td>UDDD0,VDD0 D</td>
<td></td>
<td>Array of coefficients of derivative of $D_0(X)$ i.e. $D_0'(X)$</td>
</tr>
<tr>
<td>NDDD0 I</td>
<td>NDDD0 I</td>
<td></td>
<td>Degree of $D_{DD0}(X)$</td>
</tr>
<tr>
<td>D1 C</td>
<td>UD1,VD1 D</td>
<td></td>
<td>Array of coefficients of g.c.d. of $D_0(X)$ and $D_{DD0}(X)$</td>
</tr>
<tr>
<td>ND1 I</td>
<td>ND1 I</td>
<td></td>
<td>Degree of $D_1(X)$</td>
</tr>
<tr>
<td>DDD1 C</td>
<td>UDDD1,VDD1 D</td>
<td></td>
<td>Array of coefficients of derivative of $D_1(X)$ i.e. $D_1'(X)$</td>
</tr>
<tr>
<td>NDDD1 I</td>
<td>NDDD1 I</td>
<td></td>
<td>Degree of $D_{DD1}(X)$</td>
</tr>
<tr>
<td>D2 C</td>
<td>UD2,VD2 D</td>
<td></td>
<td>Array of coefficients of g.c.d. of $D_1(X)$ and $D_{DDD1}(X)$</td>
</tr>
<tr>
<td>ND2 I</td>
<td>ND2 I</td>
<td></td>
<td>Degree of $D_2(X)$</td>
</tr>
<tr>
<td>D3 C</td>
<td>UD3,VD3 D</td>
<td></td>
<td>Array of coefficients of the product of $D_0(X)$ and $D_2(X)$</td>
</tr>
<tr>
<td>ND3 I</td>
<td>ND3 I</td>
<td></td>
<td>Degree of $D_3(X)$</td>
</tr>
<tr>
<td>D4 C</td>
<td>UD4,VD4 D</td>
<td></td>
<td>Array of coefficients of the square of $D_1(X)$</td>
</tr>
<tr>
<td>ND4 I</td>
<td>ND4 I</td>
<td></td>
<td>Degree of $D_4(X)$</td>
</tr>
<tr>
<td>G C</td>
<td>UG, VG D</td>
<td></td>
<td>Array of coefficients of the quotient $D_3(X)/D_4(X)$</td>
</tr>
<tr>
<td>NG I</td>
<td>NG I</td>
<td></td>
<td>Degree of $G(X)$</td>
</tr>
<tr>
<td>ZROS C</td>
<td>UZROS, VZROS D</td>
<td></td>
<td>Array of roots of $G(X)$</td>
</tr>
</tbody>
</table>

#### Main Program

#### Subroutine ZROS

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Variable Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APROX C</td>
<td>UAPROX, VAPROX D</td>
<td>R</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Double Precision Variable</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>M</td>
<td>I</td>
<td>M</td>
</tr>
<tr>
<td>F</td>
<td>C</td>
<td>UF, VF</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td>UC, VG</td>
</tr>
<tr>
<td>MN</td>
<td>I</td>
<td>MN</td>
</tr>
<tr>
<td>H</td>
<td>C</td>
<td>UH, VH</td>
</tr>
<tr>
<td>LIMIT</td>
<td>I</td>
<td>LIMIT</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
</tr>
</tbody>
</table>
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)
Figure G.2. (Continued)
Figure G.2. (Continued)
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

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

DO DOUBLE PRECISION EPSI,EPS2,EPS3,UP,VP,UAPP,VAPP,UD0,VDO,UD00,VDO0,
1UD1,V01,VD2,UD01,VOD1,UG,VG,UD3,V03,UD04,VDO4,VZROS,VZROS,UAP,VA
2P,UROOT,UROOT,DEND

DO DOUBLE PRECISION XSTART

DIMENSION ANAME(2),UP(26),VP(26),UAPP(251),VAPP(251),UD0(26),VDO(26)
1UD00(26),VDOO(26),UD01(26),VDO1(26),UD02(26),VDO2(26),UD03(26),VDO3(26)
26),UG(26),VG(26),UD3(51),V03(51),UD04(51),V04(51),UZROS(251),VZROS(251)
351),UAPP(251),VAPP(251),UROOT(251),VROOT(251),MULT(251),ENTRY(26)

COMMON EPSI,EPS2,EPS3,I02,I03

DATA ASTER/4H****/

DATA PNAME,GNAME/2HP(,2HG(/,

DATA ENTR Y/IHI,IH2,LH3.1H4,IH5,IH6,1H71,H8,IH9,2H102H112H12,2HI3
1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/

DATA ANAME(1),ANAME(2)/4HNEWT,4HONS /

101=5

102=6

READ(IO1,10001) NOPOLYNP,NAPP,MAXEPSI,E2,EPS3,XSTART,XEND,KCHEC

IF(KCHECK.EQ.1)

STOP

WRITE(IO2,10201) ANAME(l),ANAME21,NOPOLYN

WRITE(IO2,20101) NAPP

WRITE(IO2,20701) EPSI

WRITE(IO2,20801) EPS2

WRITE(IO2,20401) EPS3

WRITE(IO2,20501) XSTART

WRITE(IO2,20601) XEND

WRITE(IO2,10301) NP

KKK=NP+1

NNN=KKK+1

DO 8 I=1,KKK

JJJ=NNN-I

WRITE(IO2,10401) PNAMEENTRYIJJJ),UP(JJJ),VP(JJJ)

K=O

KD=O

5 READ(IO1,10101) UP(JJJ),VP(JJJ)

IF(NAPP.NE.01 GO TO 22

4 NAPP=NP

CALL GENAPP(UAPP,VAPP,NAPP,XSTART)

GO TO 23

22 READ(IO1,10151) (UAPP(I),VAPP(I)),I=1,NAPP

23 WRITE(IO2,10301) NP

KKK=NP+1

NAPP=NP

DO 8 I=1,KKK

7 JJJ=NNN-I

8 WRITE(IO2,10401) PNAME,ENTRYIJJJJ),U(JJJ),VP(JJJ)

K=O

KD=O
TABLE G.III (Continued)

JL=1
KKK=NP+1
DO 10 I=1,KKK
   UDO(I)=UP(I)
   VDO(I)=VP(I)
10
ND0=NP
CALL DERIV(NDO,UDO,VD0,NDO0,UDO0,VD00)
CALL GCD(NDO,UDO,VD0,NDDO,UDDDO,VDDDO)
WRITE(102,3000) J1
IF(NO1.LE.1) GO TO 30
GO TO 40
VDO(I)=0.0
ND2=1
CALL DERIV(NDO1,UDO1,VD01,NDDO1,UDDDO1)
CALL GCD(NDO1,UDO1,VD01,NDDO1,UDDDO1)
WRITE(102,10251 J1
IF(NO1.EQ.0) GO TO 80
GO TO 90
80
K KK=ND0+1
DO 85 I=1,KKK
   UG(I)=UDO(I)
   VG(I)=VDO(I)
85
ND=G=ND0
GO TO 110
90
IF(NO2.EQ.0) GO TO 115
CALL PROD(NDO,UDO,VD0,NDO2,UDDO2,VD2,NDO3,UDDO3,VD3)
CALL PROD(NDO1,UDO1,VD01,NDDO1,UDDDO1,VDDDO1)
CALL divide(NDO2,VD02,UDDO2,VD2,ND2,UDDO2,VD2)
WRITE(102,10351 J1
K=NG+1
N=KK+KKK+1
DO 112 I=1,KKK
   UO3(I)=UO0(I)
   VO3(I)=VOO(I)
   NO3=ND0
112
GO TO 120
115
IF(NAP.EQ.0) GO TO 120
WRITE(102,1085) J1
IJ=JAP+1
WRITE(102,1095) (J1,UZROS(I),VZROS(I),J1,I=KKK)
GO TO 140
WRITE(102,1090) (J1,UZROS(I),VZROS(I),J1,UAP(I),VAP(I),I=1,JAP)
GO TO 120
WRITE(102,1180) I1,UZROS(I),VZROS(I),UAP(I),VAP(I),I=IJAP
IF(JAP.LT.JI) GO TO 140
WRITE(102,1190) (I1,UZROS(I),VZROS(I),J1,I=1,JAP)
GO TO 120
GO TO 100
120
K=K+JAP+1
WRITE(102,1085) (I1,UZROS(I),VZROS(I),J1,I=KKK)
GO TO 140
WRITE(102,1090) (I1,UZROS(I),VZROS(I),J1,UAP(I),VAP(I),I=1,JAP)
GO TO 120
WRITE(102,1180) I1,UZROS(I),VZROS(I),UAP(I),VAP(I),I=IJAP
IF(JAP.LT.JI) GO TO 140
WRITE(102,1190) (I1,UZROS(I),VZROS(I),J1,I=1,JAP)
GO TO 120
GO TO 100
GO TO 100

JXX=JAP+1
WRITE(102,1085) (I1,UZROS(I),VZROS(I),J1,I=KKK)
GO TO 140
WRITE(102,1090) (I1,UZROS(I),VZROS(I),J1,UAP(I),VAP(I),I=1,JAP)
GO TO 120
WRITE(102,1180) I1,UZROS(I),VZROS(I),UAP(I),VAP(I),I=IJAP
IF(JAP.LT.JI) GO TO 140
WRITE(102,1190) (I1,UZROS(I),VZROS(I),J1,I=1,JAP)
GO TO 120
GO TO 100

K=K+JAP+1
WRITE(102,1085) (I1,UZROS(I),VZROS(I),J1,I=KKK)
GO TO 140
WRITE(102,1090) (I1,UZROS(I),VZROS(I),J1,UAP(I),VAP(I),I=1,JAP)
GO TO 120
WRITE(102,1180) I1,UZROS(I),VZROS(I),UAP(I),VAP(I),I=IJAP
IF(JAP.LT.JI) GO TO 140
WRITE(102,1190) (I1,UZROS(I),VZROS(I),J1,I=1,JAP)
GO TO 120
GO TO 100

TABLE G.III (Continued)

0094
0095 IF(J.EQ.0 GO TO 170
0096 170 J=J+1
0097 IF(ND1.EQ.1) GO TO 200
0098 DO 180 I=1,ND1
0099 UD(I)=UDI(I)
0100 VD(I)=VDI(I)
0101 180 DO 190 I=1,KKK
0102 JJJ=NNN-I
0103 WRITE(102,1100)
0104 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1
0105 GO TO 1
0106 1020 FORMAT(IHI,10X.48HREPEATED
0107 USE
0108 OF THE
0109 GREATEST
0110 COMMON
0111 DIVISOR
0112 AND
0113 METHOD
0114 TO EXTRACT
0115 ROOTS
0116 AND
0117 MULTIPLICITIES
0118 OF POLYNOMIAL
0119 AND
0120
0121 1025 FORMAT///IX,25HNO
0122 ROOTS OF MULTIPLICITY
0123 ,12///
0124 1030 FORMAT(///1X,87HTHE
0125 FOLLOWING
0126 POLYNOMIAL,
0127 GIXI,
0128 CONTAINS ALL THE
0129 ROOTS
0130 OF PXI, WHICH
0131 HAVE
0132 MULTIPLICITY
0133 I
0134 NOT
0135 ALL
0136 ROOTS
0137 OF THE ABOVE
0138 POLYNOMIAL,
0139 WERE
0140 FOUND
0141 1040 FORMAT(2X,A2,A2,4H)
0142 =
0143 ,D23.16,3H
0144 +
0145 ,023.16,2H
0146 I
0147 1050 FORMAT(2X,A3,A2,4H)
0148 =
0149 0D23.16,3H
0150 +
0151 ,023.16,2H
0152 I
0153 000 FORMAT(312,1X),9X13,IX,3(D6.O,IX)t20X,207.,OIX,II)
0154 010 FORMAT(2030.0I
0155 020 FORMAT(IX,22HTHE DEGREE OF
0156 P(X)
0157 IS
0158 22H
0159 THE COEFFICIENTS
0160 ARE/
0161 030 FORMAT(2X,22HTHE DEGREE OF P(X) IS ,12,22H THE COEFFICIENTS ARE//
0162 040 FORMAT(2X,22HTHE DEGREE OF P(X) IS ,12,22H THE COEFFICIENTS ARE//
0163 050 FORMAT(2X,22HTHE DEGREE OF P(X) IS ,12,22H THE COEFFICIENTS ARE//
TABLE G.III (Continued)

0150  1100 FORMAT(///1X,13HROOTS OF PI,2X,14HMULTIPLICITIES,17X,21HINITIAL
\   APPROXIMATION/1)
0151  1190 FORMAT(///1X,12,4H) = ,023.16,3H + ,023.16,2H 1,7X,12,7X,023.
\   116,3H + ,023,16,2H 1)
0152  2000 FORMAT(///1X,4HNUMBER OF INITIAL APPROXIMATIONS GIVEN.,12)
0153  2010 FORMAT(///1X,29HMAXIMUM NUMBER OF ITERATIONS.,11X,13)
0154  2020 FORMAT(///1X,21HTEST FOR CONVERGENCE,13X,09.2)
0155  2040 FORMAT(///1X,23HTEST FOR ZERO IN SUBROUTINE GCQ.
0156  2050 FORMAT(///1X,21HTEST FOR ZERO IN SUBROUTINE QUAD.,09.2)
0157  2060 FORMAT(///1X,
0158  2070 FORMAT(///1X,34HTEST FOR ZERO IN SUBROUTINE GCD.,09.2)
0159  2080 FORMAT(///1X,34HTEST FOR ZERO IN SUBROUTINE QUAD.,09.2)
0160  3000 FORMAT(///1X,4X,32A4)
0161  ENO
TABLE G.III (Continued)

SUBROUTINE PROOCM, UF, VF, VG, MN, UH, VH

C
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 ****************************************************************************

DOUBLE PRECISION UH, VH, UF, VF, VG
DIMENSION UH(MN+1), VH(KK+1), UF(Z6), VF(5), VG(Z6)

MN=M+N
KK=M+1

DO 10 I=1, KK
   VH(I) = 0.0
   UH(I) = 0.0
   IF (I.LE.M) GO TO 10
   LIMIT = M+1
   GO TO 20
   10 LIMIT = I
   20 J = I, LIMIT
   IF (J.GT.N+I) GO TO 50
   IF (J+K.EQ.I+1) GO TO 40
   GO TO 50
   40 UH(I) = UH(I) + (VF(J)*UG(K) - VF(K)*UG(J))*UG(I)
   VH(I) = VH(I) + (VF(J)*UG(K) + UF(K)*VG(J))*UG(I)
   K = I - 1
   50 CONTINUE
   RETURN
END

SUBROUTINE GENAPPIAPPR, APPR, APPI, NAPP, XSTART

C
C ****************************
C
C SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE
C DEGREE OF THE ORIGINAL POLYNOMIAL.
C
C ***************************************************************************

DOUBLE PRECISION APPR, APPI, EPS1, EPS2, EPS3
DIMENSION APPR(NAPP), APPI(NAPP)
COMMON EPS1, EPS2, EPS3, 102, AX
IF (XSTART.EQ.0.0) XSTART = 0.5
8ETA = 0.2617994

DO 10 I = 1, NAPP
   APPRI = XSTART*COS(BETA)
   APPPI = XSTART*SIN(BETA)
   BETA = BETA*0.5235988
   XSTART = XSTART + 0.5
10 RETURN
END

SUBROUTINE GENAPP(A, APP, APPI, NAPP, XSTART)

C
C **********************
C
C SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE
C DEGREE OF THE ORIGINAL POLYNOMIAL.
C
C **************************************************************************

DOUBLE PRECISION A(NAPP), APP, APPI(NAPP)
COMMON EPS1, EPS2, EPS3, 102, MAX
IF (XSTART.EQ.0.0) XSTART = 0.5
BETAA = 0.2617994
DO 10 I = 1, NAPP
   APP = XSTART*COS(BETA)
   APPI = XSTART*SIN(BETA)
   BETA = BETA + 0.5235988
10 RETURN
END
SUBROUTINE ALTER(XOLDX, XOLDI, NALTER, ITIME)
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 ************************************************************
DOUBLE PRECISION XOLDX, XOLDI, ABXOLO, BETA, EPS1, EPS2, EPS3
COMMON EPS1, EPS2, EPS3, IO2, MAX
IF(ITIME.NE.0) GO TO 5
ITIME = 1
WRITE(IO2,1010) MAX
IFINALTER.EQ.0 GO TO 10
WRITE(IO2,10001) XOLDX, XOLDI
GO TO 20
10 ABXOLO = DSQRT((XOLDX*XOLDRI*IXOLOI*XOLDII))
BETA = ATAN2(XOLDI, XOLDX)
WRITE(IO2,10201) XOLDX, XOLDI
NALTER=NALTER+1
IFINALTER.GT.5 RETURN
GO TO (30, 40, 30, 40, 30), NALTER
30 XOLDX = -XOLDX
XOLDI = -XOLDI
GO TO 50
40 BETA = BETA + 1.0471976
XOLDX = ABXOLO*DCOS(BETA)
XOLDI = ABXOLO*DSIN(BETA)
GO TO 50
50 RETURN
1000 FORMAT(1X,02.16,3H + ,023.16,2H I,10X,21HALTERED APPROXIMATION)
1010 FORMAT(1X,02.16,3H + ,023.16,2H I,10X,13H NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
1110 ITER + ,13,12H ITERATIONS,/)
TABLE G.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE ZEROS(INQ, UQ, VQ, NAPP, VAPP, J, UROOT, VROOT, JAP, UAP, VAP, ENTRY, XSTART, XEND)</td>
</tr>
<tr>
<td>C</td>
<td>*************************************************************</td>
</tr>
<tr>
<td>C</td>
<td>* NEWTON'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A</td>
</tr>
<tr>
<td>C</td>
<td>* POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-</td>
</tr>
<tr>
<td>C</td>
<td>* IMATING CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION</td>
</tr>
<tr>
<td>C</td>
<td>* FORMULA</td>
</tr>
<tr>
<td>C</td>
<td>* X(N+1) = X(N) - P(X(N))/P'(X(N)).</td>
</tr>
<tr>
<td>C</td>
<td>*************************************************************</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION UAPP = VAPP = UROOT = VROOT = UZRO = VZRO = UQQ = VQQ = UDUMMY = VDUM</td>
</tr>
<tr>
<td>0003</td>
<td>JAY, UQQ, VQQ, UAP, VAP, UQD, VQD, UROOTS, VROOTS, EPS1, EPS2, EPS3, UAPROX, VAPROX</td>
</tr>
<tr>
<td>0004</td>
<td>DOUBLE PRECISION XSTART</td>
</tr>
<tr>
<td>0005</td>
<td>DIMENSION UAPP(25), VAPP(25), UROOT(25), VROOT(25), UQ1(25), VQ1(25), UQQ(25), VQQ(25), UQ2(25), VQ2(25), UQQ2(25), VQQ2(25), XAPPROX, VAPPROX</td>
</tr>
<tr>
<td>0006</td>
<td>COMMON EPS1, EPS2, EPS3, IO2, MAX</td>
</tr>
<tr>
<td>0007</td>
<td>DATA QNAME, QNAME, HIQQI, 2HIQI</td>
</tr>
<tr>
<td>0008</td>
<td>LOGICAL CONV</td>
</tr>
<tr>
<td>0009</td>
<td>J = 0</td>
</tr>
<tr>
<td>0010</td>
<td>ITIME = 0</td>
</tr>
<tr>
<td>0011</td>
<td>IF(NQ .GE. 3) GO TO 85</td>
</tr>
<tr>
<td>0012</td>
<td>GO TO 110</td>
</tr>
<tr>
<td>0013</td>
<td>85 KKK = NQ + 1</td>
</tr>
<tr>
<td>0014</td>
<td>GO 90 1 = 1, KKK</td>
</tr>
<tr>
<td>0015</td>
<td>UQQ(1) = UQQ(1)</td>
</tr>
<tr>
<td>0016</td>
<td>90 VQQ(1) = VQQ(1)</td>
</tr>
<tr>
<td>0017</td>
<td>NQQ = NQ</td>
</tr>
<tr>
<td>0018</td>
<td>GO TO 120</td>
</tr>
<tr>
<td>0019</td>
<td>110 CALL QUAD(NQ, UQ, J, UROOT, VROOT)</td>
</tr>
<tr>
<td>0020</td>
<td>JAP = 0</td>
</tr>
<tr>
<td>0021</td>
<td>GO TO 310</td>
</tr>
<tr>
<td>0022</td>
<td>120 DO 200 I = 1, NAPP</td>
</tr>
<tr>
<td>0023</td>
<td>IF(ALT = 0)</td>
</tr>
<tr>
<td>0024</td>
<td>VAPROX = UAPP(1)</td>
</tr>
<tr>
<td>0025</td>
<td>VAPROX = VAPP(1)</td>
</tr>
<tr>
<td>0026</td>
<td>CALL NEWTON(UAPROX, VAPROX, NQQ, UQQ, UQQ, UZRO, VZRO, CONV)</td>
</tr>
<tr>
<td>0027</td>
<td>IF(CONV) GO TO 160</td>
</tr>
<tr>
<td>0028</td>
<td>CALL ALTRE(UAPPROX, VAPPROX, IALT, ITIME)</td>
</tr>
<tr>
<td>0029</td>
<td>IF(IALT .GT. 5) GO TO 200</td>
</tr>
<tr>
<td>0030</td>
<td>UAPPROX = UAPP(1)</td>
</tr>
<tr>
<td>0031</td>
<td>VAPPROX = VAPP(1)</td>
</tr>
<tr>
<td>0032</td>
<td>GO TO 130</td>
</tr>
<tr>
<td>0033</td>
<td>130 UQQ(I) = UQQ(I)</td>
</tr>
<tr>
<td>0034</td>
<td>VQQ(I) = VQQ(I)</td>
</tr>
<tr>
<td>0035</td>
<td>UAP(1) = UAPPROX</td>
</tr>
<tr>
<td>0036</td>
<td>VAP(1) = VAPPROX</td>
</tr>
<tr>
<td>0037</td>
<td>CALL HORNER(UZRO, VZRO, NQQ, UQQ, UQQ, UQQ, UDUMMY, VDUM)</td>
</tr>
<tr>
<td>0038</td>
<td>HD 180 I = 1, NQQ</td>
</tr>
<tr>
<td>0039</td>
<td>UQQ(I) = UQQ(I)</td>
</tr>
<tr>
<td>0040</td>
<td>180 UQQ(I) = UQQ(I)</td>
</tr>
<tr>
<td>0041</td>
<td>NQQ = NQQ - 1</td>
</tr>
<tr>
<td>0042</td>
<td>IF(NQQ .GE. 3) GO TO 200</td>
</tr>
<tr>
<td>0043</td>
<td>JAP = 0</td>
</tr>
<tr>
<td>0044</td>
<td>GO TO 220</td>
</tr>
</tbody>
</table>
TABLE G.III (Continued)

200 CONTINUE
0045 IF(J.GE.NQ) GO TO 205
0046 IF(J.EQ.NQ+1) GO TO 300
0047 IF(J.EQ.NQ) GO TO 205
0048 IF(J.EQ.NQ+1) GO TO 205
0049 IF(J.EQ.NQ+2) GO TO 205
0050 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
0051 GO TO 120
0052 IF(J.EQ.NQ+2) GO TO 210
0053 WRITE(I02,1200) KKK, NQQ
0054 WRITE(I02,1200) 0.0
0055 WRITE(I02,1200) NNN
0056 WRITE(I02,1200) 1.0
0057 WRITE(I02,1200) JJJ, NNN
0058 157 WRITE(I02,1100) QNAME, ENTRY(JJJ), UQQ(JJJ), VQQ(JJJ)
0059 210 IF(J.EQ.0) GO TO 310
0060 JAP = 1
0061 GO TO 230
0062 CALL QUAD(UQQ, VQQ, J, UROOT, VROOT)
0063 WRITE(I02,1132) UROOT(1), VROOT(1), UAPI(1), VAPI(1)
0064 IF(JAP.LT.J) GO TO 235
0065 GO TO 240
0066 220 WRITE(I02,1134) UROOT(1), VROOT(1), KKK, J
0067 230 WRITE(I02,1134) UROOT(1), VROOT(1), KKK, J
0068 240 WRITE(I02,1134) UROOT(1), VROOT(1), KKK, J
0069 250 WRITE(I02,1134) UROOT(1), VROOT(1), KKK, J
0070 DO 300 I = 1, J
0071 CALL NEWTON(UROOT(1), VROOT(1), NQ, UQQ, VQQ, U2RQQ, V2RQQ, CONV)
0072 IF(J.AND.0.0) GO TO 280
0073 WRITE(I02,1140) UROOT(1), VROOT(1), MAX, NQ
0074 KKK = NQQ + 1
0075 NNN = KKK + 1
0076 GO TO 242
0077 JJJ = NNN
0078 WRITE(I02,1140) QNAME, ENTRY(JJJ), UQQ(JJJ), VQQ(JJJ)
0079 240 IF(I.EQ.0) GO TO 241
0080 IF(I.EQ.0) GO TO 240
0081 GO TO 300
0082 241 KKK = JAP + 1
0083 DO 245 I = 1, KKK
0084 UAPI(I) = UAPI(I-1)
0085 VAPI(I) = VAPI(I-1)
0086 245 JAP = I + 1
0087 GO TO 240
0088 280 JJJ = JAP + 1
0089 WRITE(I02,1150) UROOT(I), VROOT(I)
0090 WRITE(I02,1150) UROOT(I), VROOT(I)
0091 300 CONTINUE
0092 JJJ = J
0093 IF(J.EQ.0) GO TO 305
0094 DO 305 I = 1, J
0095 WRITE(I02,1150) UROOT(I), VROOT(I)
0096 WRITE(I02,1150) UROOT(I), VROOT(I)
0097 WRITE(I02,1150) UROOT(I), VROOT(I)
0098 305 WRITE(I02,1150) NQ
0099 300 NNN = KKK + 1
0100 KKK = NQQ + 1
0101 DO 306 L = 1, KKK
0102 JJJ = NNN

TABLE G.III (Continued)

306 WRITE(102,1040) QNAME,ENTRY(IJJJ),UQ(IJJJ),VQ(IJJJ)
110 RETURN
1200 FORMAT(/X,13HCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH
1 NO ZEROS WERE FOUND./)
1132 FORMAT(/X,13HROOTS OF G(X),84X,21HINITIAL APPROXIMATION/)  
1133 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H 1.17X,D23.16,3H
1 * ,D23.16,2H 1)  
1134 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H 1.22X,26HRESULT
15 OF SUBROUTINE Quad)  
1140 FORMAT(/X,1X,40HNO ROOTS FOR INITIAL APPROXIMATION ROOT(12,4H) =
1 1,D23.16,3H + ,D23.16,2H 1.6H AND 13,4GH ITERATIONS ON THE POLYN
2OMIAL OF DEGREE 12,18H WITH COEFFICIENTS/)  
1150 FORMAT(/X,1X,45HNO ROOTS FOR THE POLYNOMIAL Q(X) OF DEGREE = 12,
13H WITH GENERATED INITIAL APPROXIMATIONS/)  
1040 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H 1)  
110 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H 1)  
END
TABLE G.111 (Continued)

SUBROUTINE GCD(NUR,VRM,USVS,MI,USSVSS)

C * *****
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C *
C 
TABLE G.III (Continued)

0050 140 KKK=KKK+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=KKK+2
0057 NNN=NNN+1
0058 DO 160 J=KKK, NNN
0059 URR(J)=USS(J)
0060 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.11 (Continued)**

```fortran
0001 SUBROUTINE NEWTONIUXVX,N,UPsVP,UDVXO,CONV)  
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                                                                                     
******************************************************************************
0002 DOUBLE PRECISION UX,VX,UP,VP,UXO,VDXO,UDXO,VPXO,UDPO,VPDO,UPDO,VPDO,U          
0003 DOUBLE PRECISION DDIFF,VDIFF,EPSI,EPSON,EPSP,AAA,BBB                           
0004 DOUBLE PRECISION UDD,ABPXO                                                     
0005 DIMENSION UP(26),VP(26),UB(26),VB(26)                                         
0006 COMMON EPSI,EPSON,EPSP,IOZ,MAX                                                 
0007 LOGICAL CONV                                                                    
0008 UXO=UX                                                                        
0009 VXO=VX                                                                        
0100 DO 10 I=1,MAX                                                                
0101 CALL HORNEX(UXO,VPXO,UP,VP,UB,VP,UB,UDPDXO,VPDVO)                            
0102 UPXO=UB(1)                                                                    
0103 VPXO=VB(1)                                                                    
0104 DDD=DSQRT(UPXO*UPXO+VPXO*VPXO)                                                
0105 IF(DDD.EQ.0.0) GO TO 20                                                        
0106 AAA=DSQRT(UPXO*UPXO+VPXO*VPXO)                                                
0107 BBB=DSQRT(UOIFF*UDIFF+VDIFF*VDIFF)                                            
0108 IF(AAA/BBB.EQ.0.01 GO TO 20                                                    
0109 5 BBB=UDPDXO*UPDXO+VPDXO*VPDXO                                               
0110 UDIFF=UPDXO*UPDXO+VPDXO+VPDXO/BBB                                              
0111 VDIFF=VPDXO*VPDXO+VPDXO+VPDXO/BBB                                              
0112 UDD=DDER(TUPDXO*VPDXO+VPDXO*VPDXO)                                            
0113 IF(UDD.EQ.0.01 GO TO 10                                                        
0114 IF(AAA/BBB.EQ.0.01 GO TO 20                                                    
0115 10 CONTINUE                                                                     
0116 15 CONV=.FALSE.                                                                
0117 RETURN                                                                        
0118 20 CONV=.TRUE.                                                                 
0119 RETURN                                                                        
0120 END                                                                            
```
TABLE G.III (Continued)

0001 SUBROUTINE DIVIDE(UP,VP,UD,VD,UQ,VQ)
0002 C *******************************************************
0003 C *
0004 C * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE *
0005 C * QUOTIENT POLYNOMIAL H(X) = F(X)/G(X). *
0006 C *
0007 *******************************************************
0008 DOUBLE PRECISION UP,VP,UD,VD,UQ,VQ
0009 DIMENSION UP(261),VP(261),UD(261),VD(261),UQ(261),VQ(261)
0010 K=N-M
0011 UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0012 UQ(K+1)=(UP(N+I)*UD(M+1)+VP(N+I)*VD(M+1))/UDUMMY
0013 VQ(K+1)=(VP(N+M+1)*UD(M+1)-UP(N+I)*VD(M+1))/UDUMMY
0014 IF(K.EQ.0) GO TO 100
0015 J=1
0016 DO 50 I=1,K
0017 UTERM=UP(N-J)
0018 VTERM=VP(N-J)
0019 KK=K+1
0020 NNN=M-J
0021 DO 40 M=NNN,M
0022 IF(KK.GE.1) GO TO 10
0023 GO TO 45
0024 10 UTERM=UTERM-(UQ(KK)*UD(M)+VQ(KK)*VD(M))/UDUMMY
0025 45 UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)
0026 40 KK=KK-1
0027 50 UTERM=UTERM*(UQ(KK)+VQ(KK))*UD(M)+VTERM*VD(M)
0028 20 UTERM=UTERM-(UQ(KK)*UD(M)+VQ(KK)*VD(M))/UDUMMY
0029 10 IF(KK.GE.1) GO TO 20
0030 GO TO 40
0031 20 UTERM=UTERM-(UQ(KK)*UD(M)+VQ(KK)*VD(M))/UDUMMY
0032 GO TO 40
0033 100 RETURN
0034 END
### TABLE G.III (Continued)

```
0001 SUBROUTINE HORNER(UX, VX, N, UP, VP, UB, VB, UC, VC)
C
C * HOMER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A
C POINT 0 AND ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO
C DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X - D).
C
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) = VP(N+1)
0006 VB(N+1) = UB(N+1) - UX*UB(N+1) - VX*VB(N+1) + UP(N+1)
0007 VB(N+1) = UX*VB(N+1) + VX*UB(N+1) + VP(N+1)
0008 UC = VB(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 = UC - VX*VC
0016 VB = UDUMMY + VB(KKK+2-I)
0017 UC = UDUMMY + VB(KKK+2-I)
0018 VC = UDUMMY + VB(KKK+2-I)
0019 RETURN
0020 END
```
**TABLE G.III (Continued)**

```fortran
SUBROUTINE QUAD(NUA,VA, J, UROOT, VROOT)
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**************************************************************************
CDOUBLE PRECISION EPSI,EPS2,EPSLON,UROOT,VROOT,UA,VA,UDISC,VDISC,UD
1,VD,UDD,UTEMP,VTEMP,BBB
DIMENSION UROOT(25),VROOT(25),UA(26),VA(26)
COMMON EPSI,EPS2,EPSLON,IO2,MAX
IF(IN.GT.1) GO TO 10
J=J+1
BBB=UA(2)*UA(2)+VA(2)*VA(2)
UROOT(J)=-UA(2)+VA(2)/BBBB
VROOT(J)=(VAT2-UA(2)+VA(2))/BBBB
GO TO 100
10 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(3)*UA(1)-VA(3)*VA(1)))
VDISC=(UA(2)*UA(2)+VA(2)*VA(2))-(4.0*(UA(3)*VA(1)+VA(3)*UA(1)))
UD=2.0*UA(3)
VD=2.0*VA(3)
DDD=SQRT(UDISC*UDISC+VDISC*VDISC)
IF(DDD.LT.EPSLON) GO TO 20
CALL COMSQT(UDISC,VDISC,UTEMP,VTEMP)
BBBB=UD*UD+VD*VD
UROOT(J+1)=((-UA(2)+UTEMP)*UD+(-VA(2)+VTEMP)*VD)/BBBB
VROOT(J+1)=(VAT2*(-VA(2)+VTEMP)*UD-(-UA(2)+UTEMP)*VD)/BBBB
UROOT(J+2)=(VAT2-VA(2)*UD+UA(3)*VD)/BBBB
VROOT(J+2)=(UA(2)*VD-(-UA(2)+UTEMP)*VTEMP)/BBBB
J=J+1
GO TO 100
20 J=J+1
BBB=UD*UD+VD*VD
UROOT(J+1)=-UA(2)+VAT2*VTEMP*VD+UA(2)*VTEMP*VD/BBBB
VROOT(J+1)=(-UA(2)+VAT2*VTEMP*VD-(-UA(2)+VTEMP*VD)/BBBB
WRITE(IO2,1000) UROOT(J),VROOT(J)
1000 FORMAT(/A19,13H QUAD FOUND ,1F3.16,2H ,1F3.16,2H TO BE A MULTIPLE ROOT)!
RETURN
END
```
TABLE G.III (Continued)

0001 SUBROUTINE DERIV(N,UP,VP,U,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,U,UA,AAA
0003 DIMENSION UP(26),VP(26),U(26),VA(26)
0004 KK=N+1
0005 DO 10 I=2,KKK
0006 AAA=I-I
0007 UA(I-1)=AAA*UP(I)
0008 VA(I-1)=AAA*VP(I)
0009 M=N-1
0010 RETURN
0011 END

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+UXI/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(UXI 40,50,60
0011 30 UY=AAA
0012 VY=BBB
0013 GO TO 100
0014 40 DUMMY=DABS(UX)
0015 50 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 70 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/LH1,LH2,...,LH9,2H10,2H11,...,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)
UDDO(N+1), VDDO(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)
UDO(N+1), VDO(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 COMSQRT

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 $0.9X_0$ 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)
Figure H.1. (Continued)
Figure II.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)
TABLE H.111

PROGRAM FOR REPEATED G.C.D. - MULLER'S METHOD

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

* DOUBLE PRECISION PROGRAM FOR THE REPEATED G.C.D. - MULLER'S METHOD *
* THIS METHOD REPEATEDLY FINDS THE GREATEST COMMON DIVISOR OF TWO          *
* POLYNOMIALS IN ORDER TO EXTRACT THE ZEROS IN GROUPS ACCORDING TO          *
* MULTIPLICITY USING NEWTON'S METHOD. ALL ZEROS OF MULTIPLICITY 1          *
* ARE EXTRACTED FOLLOWED BY THOSE OF MULTIPLICITY 2, ETC.                  *
******************************************************************************

0001       DOUBLE PRECISION EPS1,EPS2,EPS3,UP,VP,UAPP,VAPP,UDDO,VDDO,  
           UDD3,UDOO,VDDU,VDDO,VOO,VDOD,  
           UD01,VD01,VD02,UD01,VDD01,UG,UG03,VO3,UD4,V04,UDKUS,VK05,UP,VA  
           2P,UK001,UK002,UK003,UK004  
0002       DOUBLE PRECISION XSTART  
0003       DOUBLE PRECISION XEND  
0004       DOUBLE PRECISION URAPP,VAPP  
0005       DOUBLE PRECISION EPS4  
0006       DIMENSION VAPP(25,3),UAPP(25,3),URAPP(25,3),VRAPP(25,3)  
0007       DIMENSION UP(26),VP(26),MULTI25,UDDDO(26),VDDD0(26),UD01(26),  
           VDD01(26),VOO1(25)  
0008       COMMON EPS1,EPS2,EPS3,EPS4,IO2,MAX  
0009       DATA PNAME,GNAME,HP2,2HGI,DLNAME,SHDL,TI  
0010       DATA ASTER/*****/  
0011       DATA ENTRY/HLI1,HL2,H23,HL4,H45,HL5,H56,HL6,H67,H78,H89,H90,H101,  
           H112,H123,H134,H145,H156,H167,  
           H213,H234,H245,H256,H267,  
           1,2H4,2H5,5H6,2H7,2H8,2H9,2H10,2H11,2H12,2H13,  
           1,2H14,2H15,5H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/  
0012       DATA ANAME(2),ANAME(2),ANAME(2),ANAME(2),ANAME(2)/  
0013       IO1=5  
0014       T1=6  
0015       1 READ(101,1000) NOPOLY,NP,NAPP,MAX,EPS1,EPS2,EPS3,XSTART,XEND,  
           KCMEC  
0016       IF(KCHECK.EQ.1) STOP  
0017       WRITE(102,1020) ANAME(1),ANAME(2),NPOLY  
0018       WRITE(102,2000) NAPP  
0019       WRITE(102,2010) MAX  
0020       WRITE(102,2070) EPS1  
0021       WRITE(102,2070) EPS2  
0022       WRITE(102,2080) EPS3  
0023       WRITE(102,2040) XSTART  
0024       WRITE(102,2050) XEND  
0025       WRITE(102,2060) EPS4  
0026       KKK=NP+1  
0027       NNN=KKK+1  
0028       GO 5 1=1,KKK  
0029       JJJ=NNN+1  
0030       5 READ(101,1010) UP(JJJ),VP(JJJ)  
0031       IF(NAPP.NE.0) GO TO 22  
0032       NAPP=NP  
0033       CALL GENUP(UP,VAPP,NAPP,XSTART)  
0034       GO TO 23  
0035       22 READ(101,1015) (UAPP(I,2),VAPP(I,2),I=1,NAPP)  
0036       23 WRITE(102,1030) NP  
0037       NNN=KKK+1  
0038       DD 8 1=1,KKK  
0039       JJJ=NNN+1  
0040       STOP

301
TABLE H.111 (Continued)

0041 \( \text{WRITE}(102,1040) \) PNAME, ENTRY(JJJ), UP(JJJ), VP(JJJ)
0042 \( K=0 \)
0043 \( K=0 \)
0044 \( J=1 \)
0045 \( \text{DO } 10 \ I=1, KKK \)
0046 \( \text{UD}(1)=UP(I) \)
0047 \( 10 \ \text{VDO} = \text{VP}(I) \)
0048 \( \text{NDO} = \text{NP} \)
0049 \( \text{CALL DERIV}(\text{NDO}, \text{UDO}, \text{VDO}, \text{NDOO}, \text{UDOO}, \text{VDOO}) \)
0050 \( \text{CALL GCOD}(\text{NDO}, \text{UDO}, \text{VDO}, \text{NDOO}, \text{UDOO}, \text{NDO1}, \text{UD1}, \text{V01}) \)
0051 \( \text{IF}(\text{NDO}+\text{ND}, \text{LE}, 2 \times \text{ND}) \text{ GO TO } 60 \)
0052 \( \text{GO TO } 40 \)
0053 \( 30 \ \text{UD}(1)=0.0 \)
0054 \( \text{V0}(1)=0.0 \)
0055 \( \text{NDO}=0 \)
0056 \( \text{GO TO } 50 \)
0057 \( \text{GO TO } 60 \)
0058 \( \text{CALL PROD}(\text{ND}, \text{UDI}, \text{V01}, \text{NDI0}, \text{UD01}) \)
0059 \( \text{CALL GCOD}(\text{NDI}, \text{UDI}, \text{V01}, \text{NDI0}, \text{UD01}, \text{ND2}, \text{UD2}, \text{V02}) \)
0060 \( \text{GO TO } 70 \)
0061 \( \text{IF}(\text{NO1}+\text{NO2}, \text{LE}, 2 \times \text{ND1}) \text{ GO TO } 60 \)
0062 \( \text{GO TO } 80 \)
0063 \( \text{IF}(\text{NDL}, \text{LE}, 1) \text{ GO TO } 30 \)
0064 \( \text{GO TO } 90 \)
0065 \( 80 \ \text{KKE}=\text{NDO}+1 \)
0066 \( \text{DO } 85 \ I=1, KKK \)
0067 \( \text{UG}(I)=\text{UD}(I) \)
0068 \( \text{V0}(I)=\text{V0}(I) \)
0069 \( \text{NG} = \text{NDO} \)
0070 \( \text{GO TO } 110 \)
0071 \( \text{GO TO } 120 \)
0072 \( 90 \ \text{IF}(\text{ND1}, \text{EQ}, 0) \text{ GO TO } 115 \)
0073 \( \text{CALL PROD}(\text{ND}, \text{UD0}, \text{V01}, \text{NO1}, \text{UD01}, \text{NO2}, \text{UD2}, \text{V02}) \)
0074 \( \text{CALL PROD}(\text{ND1}, \text{UD1}, \text{V01}, \text{NO1}, \text{UD1}, \text{NO4}, \text{UD4}, \text{V04}) \)
0075 \( \text{CALL DIVER}(\text{ND}, \text{UD}, \text{V0}), \text{ND}, \text{UD}, \text{V0}) \)
0076 \( \text{CALL PROD}(\text{ND}, \text{UD}, \text{V0}) \)
0077 \( \text{CALL MULLER}(\text{NG}, \text{UG}, \text{V0}, \text{NG}, \text{UG}, \text{V0}) \)
0078 \( \text{GO TO } 110 \)
0079 \( \text{GO TO } 120 \)
0080 \( 100 \ \text{IF}(\text{J} . \text{EQ}, 0) \text{ GO TO } 150 \)
0081 \( \text{WRITE}(102,1040) \) J
0082 \( \text{KKE}=\text{NG}+1 \)
0083 \( \text{NN}=\text{KKK}+1 \)
0084 \( \text{DO } 112 \ I=1, KKK \)
0085 \( \text{UD}(1)=\text{UD}(I) \)
0086 \( \text{V0}(1)=\text{V0}(I) \)
0087 \( \text{GO TO } 115 \)
0088 \( 112 \ \text{WRITE}(102,1040) \) GNAME, ENTRY(JJJ), UGI(JJJ), VGI(JJJ)
0089 \( \text{KKE}=\text{NG}+1 \)
0090 \( \text{N0}=\text{KKK}+1 \)
0091 \( \text{DO } 118 \ I=1, KKK \)
0092 \( \text{UD}(1)=\text{UD}(I) \)
0093 \( \text{GO TO } 120 \)
0094 \( 118 \ \text{WRITE}(102,1040) \) J
0095 \( \text{KKE}=\text{NDO}+1 \)
0096 \( \text{DO } 116 \ I=1, KKK \)
0097 \( \text{UD}(1)=\text{UD}(I) \)
0098 \( \text{GO TO } 100 \)
TABLE H.III (Continued)

0098 120  KKK=JAP+1
0099  WRITE(102,1085) (I,JUZROS(I),VZROS(I),J1,J=KKK,J)
0100  GO TO 140
0101 130 DO 135 I=1,JAP
0102 135 WRITE(102,1105) (I,JUZROS(I),VZROS(I),J1,JURAPPI(I),VRAPP(I,J)
0103  IF(JAP.LT.J) GO TO 120
0104 140 IF(J.EQ.NG) GO TO 155
0105 150 WRITE(102,1095)
0106 151 IF(J.EQ.0) GO TO 170
0107 155 DO 160 I=1,J
0108 160 UROOT(KOI+)=UZROS(I)
0109  VROOT(KOI+)=VZROS(I)
0110  MULT(KD+I)=J
0111  K=J+1
0112  K=K+J
0113  IF(K.GE.NP) GO TO 1
0114  170 J=J+1
0115  IF(ND.IE.1) GO TO 200
0116  UD(I)=UD1(I)
0117  VD(I)=VD1(I)
0118  IF(J.GE.1) GO TO 110
0119  UD(I)=UD2(I)
0120  VD(I)=VD2(I)
0121  UDG(I)=UDG(I)
0122  VDG(I)=VDG(I)
0123  VD(I)=VD1(I)
0124  ND=ND1
0125  KKK=ND1+1
0126  DO 190 I=1,ND
0127  UDI(I)=UDI(I)
0128  VDI(I)=VDI(I)
0129  ND1=ND1
0130  GO TO 20
0131  200 IF(ND.IEQ.0) GO TO 1
0132  KD=KD+1
0133  DENOM=UD1(I)*UD1(I)+VD1(I)*VD1(I)
0134  UROOT(KD)=UD1(I)*UD1(I)+VD1(I)*VD1(I)*DENOM
0135  VROOT(KD)=VD1(I)*VD1(I)+UD1(I)*UD1(I)*DENOM
0136  MULT(KD+1)=J
0137  WRITE(102,3005) (ASTER.I=1,33)
0138  WRITE(102,1055) J
0139  KD=KD+1
0140  ND=ND+1
0141  140 KD=KD+1
0142  ND=ND+1
0143  210 WRITE(102,1103) OLNAME,ENTRY(I,J),UD1(I),VD1(I)
0144  WRITE(102,1104) KD,UROOT(KD),VRAPP(K),J
0145  GO TO 1
0146  1020 FORMAT(13H,10X,4HREPEATED USE OF THE GREATEST COMMON
0147  14X,K,4H,5H METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIA
0148  15X,K,4H,5H NUMBER ,12//)
0149  1025 FORMAT(13X,25HNO ROOTS OF MULTIPlicity ,12//)
0150  1035 FORMAT(13X,8HTHE FOLLOWING POLYNOMIAL, GEX, CONTAINS ALL THE R
0151  15X,K,4H WHICH HAVE MULTIPLICITY ,12//)
0152  1065 FORMAT(13X,5HROOTS(12.4H)= ,D23.16,4H+ ,D23.16,4H I,8X,12,9K,2SHM
0153  10INITIAL APPROXIMATIONS)
0154  1095 FORMAT(13X,5HNOT ALL ROOTS OF THE ABOVE POLYNOMIAL6, WERE FOUN
TABLE H.III (Continued)

10//

0152 1000 FORMAT(3(12,1X1,9X,13,1X,3(06.0,1X1,20X,2(07.0,1X),11)
0153 1010 FORMAT(2D30.0)
0154 1015 FORMAT(2D30.0)
0155 1030 FORMAT(1X,22H THE DEGREE OF P(X) IS ,12,22H THE COEFFICIENTS ARE//
11)
0156 1040 FORMAT(2X,9A2,A2,4H)
0157 1100 FORMAT(2X,43,A2,4H)
0158 1150 FORMAT(1X,13H ROOTS OF P(X),52X,14MULTIPLEITIES,17X,2INITIAL
1 APPROXIMATION//)
0159 1190 FORMAT(2X,5H ROOT,12,4H)
0160 2000 FORMAT(1X,41H NUMBER OF INITIAL APPROXIMATIONS GIVEN . ,12)
0161 2010 FORMAT(1X,29H MAXIMUM NUMBER OF ITERATIONS . ,11X,13)
0162 2020 FORMAT(1X,21H TEST FOR CONVERGENCE . ,11X,09.2)
0163 2040 FORMAT(1X,23H RADIUS TO START SEARCH . ,11X,09.2)
0164 2050 FORMAT(1X,21H RADIUS TO END SEARCH . ,13X,09.2)
0165 2060 FORMAT(1X,34H TEST FOR ZERO IN SUBROUTINE GCD . ,09.2)
0166 2070 FORMAT(1X,34H TEST FOR ZERO IN SUBROUTINE QUAD . ,09.2)
0167 3000 FORMAT(1X,32X)
0168 END
TABLE H.III (Continued)

SUBROUTINE PROD(M,UF,VF,N,VG,NN,UH,VI)

*----------------------------------------------------------------------------------------*
* GIVEN POLYNOMIALS R(XI) AND S(XI), THIS SUBROUTINE COMPUTES THE                        *
* COEFFICIENTS OF THE PRODUCT POLYNOMIAL T(XI) = R(XI) S(XI).                           *
*----------------------------------------------------------------------------------------*

DOUBLE PRECISION UH,VM,UF,VF,UG,VG

DIMENSION UH(51),VM(51),UF(26),VF(26),UG(26),VG(26)

MN=M+N
KKK=MN*

DO 100 I=1,KKK
  K=I
  UH(I)=O.O
  VM(I)=O.O
  IF(I.LE.M+1) GO TO 10
  LIMIT=M+1
  GO TO 20
10  LIMIT=I
20  DO 50 J=L,LIMIT
    IF(K.GT.N+L) GO TO 50
    IF(J.K.GT.N+L) GO TO 40
    GO TO 50
40  UH(I)=UH(I)+UF(J)*UG(K)-VF(J)*VG(K)
    VM(I)=VM(I)+VF(J)*UG(K)+UF(J)*VG(K)
50  K=K-1
100 CONTINUE
RETURN
END
SUBROUTINE QUAD(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************************************************************************

DOUBLE PRECISION EPS, EPS2, EPSLN, UROOT, VROOT, UA, VA, UDISC, VDISC, U0

COMMON EPS, EPS2, EPSLN, EPS4, IO2, MAX

IF (N.GT.1) GO TO 10

J = J + 1

BBB = UAI2*UA(2) + VA12*VA(2)

UROOT(J) = - (UA12*UA(2) + VA12*VA(2)) / BBB

VROOT(J) = -(VA12*UA(2) + UA12*VA(2)) / BBB

GO TO 100

UDISC = (UA12*UA(2) - VA12*VA(2) - (4.0*(UA3*UA12 - VA3*VA12))

VDISC = (2.0*UA2*UA(2) - (4.0*(UA12*UA12 - VA12*VA12))

UD = 2.0*UA(3)

VD = 2.0*VA(3)

DDD = DSQRTIUDISC*UDISC + VDISC*VDISC)

IF (DDD.EQ.0.0) GO TO 20

CALL COMSQT(UDISC, VDISC, UTEMP, VTEMP)

BBB = U0*U0 + V0*V0

UROOT(J+I) = ((-UA12*UA(2) + UTEMP) + U0*U0 + V0*V0) / BBB

VROOT(J+I) = ((-VA12*UA(2) + UTEMP) + U0*U0 + V0*V0) / BBB

GO TO 100

20 J = J + 1

BBB = U0*U0 + V0*V0

UROOT(J+I) = ((-UA12*UA(2) + U0*VA12) / BBB

VROOT(J+I) = ((-VA12*UA(2) + U0*UA12) / BBB

WRITE(02, 1000) UROOT(J), VROOT(J)

100 FORMAT(///LX, 11HQUAD FOUND, 023.16H + , 023.16H TO BE A M
MULTIPLE ROOT/)

100 RETURN

END
SUBROUTINE DERIVUP,VP,M,UA,VA

C

C

GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF
its DERIVATIVE P'(X).

C

C

DOUBLE PRECISION UP,VP,UA,VA
DIMENSION UP(26),VP(26),UA(26),VA(26)

DO 10 I=2,MMM
AAA=I-1
UA(I-1)=AAA*UP(I)
10 VA(I-1)=AAA*VP(I)

RETURN
END
SUBROUTINE GCODNUR, VR, MUS, VS, MI, US, VS
C
C  *****
C  *******************************************************************
C  *
C  GIVEN POLYNOMIALS PI(X) AND DPI(X) WHERE DEG. DPI(X) IS LESS THAN DEG.
C  *
C  PI(X), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF PI(X)
C  *
C  *****
C  ********** ******** ** **********************************************
DOUBLE PRECISION USSSSS, VSSSSS
DOUBLE PRECISION URVRUSVSUSSVSSURRtRRRUDVDUTVTEPSLONEP
DIMENSION UR126, VR126, US(26), VS(26), USS(26), VSS(26), URR(26), VRR(26), UT(26), VT(26)
COMMON EPSLONEPS2, EPS3, EPS4, #102, MAX
N1 = N
MI = M
KKK = N + 1
DO 20 I = 1, KKK
URR(I) = UR(I)
DO 20 VRR(I) = VR(I)
KKK = M + 1
DO 25 I = 1, KKK
USSII) = US(I)
DO 25 VSS(I) = VS(I)
BBB = USS(M + 1) * USS(MI + 1) + VSS(MI + 1) * VSS(MI)
UD = (URR(NI + 1) * US(MI + 1) + VRR(NI + 1) * VSS(MI + 1)) / BBB
VD = (USS(MI + 1) * VRR(NI + 1) - URR(NI + 1) * VSS(MI + 1)) / BBB
KKK = N1 - 1
DO 40 I = KKK, N1
UTI(I) = URR(I) - (UD * USSII) + VDD * VSSII)
VT(I) = VRR(I) - (UDD * VSSII) + UDD * USSII)
IF(MI.EQ.N1) GO TO 70
KKK = N1 - M
DO 45 I = 1, KKK
UTI(I) = URR(I)
DO 45 VTI(I) = VRR(I)
IF(MI.EQ.N1) GO TO 70
CONTINUE
DO 95 I = 1, M
BBA = USS(MI + 1) * USS(MI + 1) + VSS(MI + 1) * VSS(MI + 1)
USSS55 = (USS[I] * USSM[I] + VSS[I] * VSS[I]) / BBB
VSSS55 = (VSS[I] * VSSM[I] - USS[I] * USSM[I]) / BBB
USS(MI) = USSS55
VSS(MI) = VSSS55
USS(MI + 1) = 0.0
VSS(MI + 1) = 0.0
GO TO 200
K = NI - 1
IF(K.EQ.0) GO TO 170
IF(K.LT.M1) GO TO 140
KKK = K + 1
DO 130 J = K, KKK
URR(J) = UT(J)
DO 130 VRR(J) = VT(J)
GO TO 30
TABLE H.111 (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0050</td>
<td>140 KKK=K+1</td>
</tr>
<tr>
<td>0051</td>
<td>DO 150 J=1,KKK</td>
</tr>
<tr>
<td>0052</td>
<td>URR(J)=US(J)</td>
</tr>
<tr>
<td>0053</td>
<td>VRR(J)=VS(J)</td>
</tr>
<tr>
<td>0054</td>
<td>US(J)=UT(J)</td>
</tr>
<tr>
<td>0055</td>
<td>150 VS(J)=VT(J)</td>
</tr>
<tr>
<td>0056</td>
<td>KKK=K+2</td>
</tr>
<tr>
<td>0057</td>
<td>NNN=M+1</td>
</tr>
<tr>
<td>0058</td>
<td>DO 160 J=KKK,NNN</td>
</tr>
<tr>
<td>0059</td>
<td>URR(J)=US(J)</td>
</tr>
<tr>
<td>0060</td>
<td>160 VRR(J)=VS(J)</td>
</tr>
<tr>
<td>0061</td>
<td>MI=M</td>
</tr>
<tr>
<td>0062</td>
<td>MI=K</td>
</tr>
<tr>
<td>0063</td>
<td>GO TO 30</td>
</tr>
<tr>
<td>0064</td>
<td>170 US(J)=1.0</td>
</tr>
<tr>
<td>0065</td>
<td>VS(J)=0.0</td>
</tr>
<tr>
<td>0066</td>
<td>MI=0</td>
</tr>
<tr>
<td>0067</td>
<td>200 RETURN</td>
</tr>
<tr>
<td>0068</td>
<td>END</td>
</tr>
</tbody>
</table>
### TABLE H.III (Continued)

```fortran
0001 SUBROUTINE DIVIDE(N, UP, VP, UD, VO, UQ, VQ)
C *******************************************************************************
C  * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE  *
C  * QUOTIENT POLYNOMIAL H(X) = F(X)/G(X).                                     *
C *******************************************************************************
0002 DOUBLE PRECISION UP, VP, UD, VO, UQ, VQ, UTERM, VTERM, UDUMMY
0003 DIMENSION UP(26), VP(26), UD(26), VO(26), UQ(26), VQ(26)
0004 K = N - M
0005 UDUMMY = UD(M+1) * UD(M+1) * VO(M+1) * VO(M+1)
0006 UQ(K+1) = UP(N+1) * UD(M+1) * VP(N+1) * VO(M+1) / UDUMMY
0007 VQ(K+1) = VP(N+1) * UD(M+1) - UP(N+1) * VO(M+1) / UDUMMY
0008 IF(K.EQ.0) GO TO 100
0009 J = 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 = N - J
0016 DO 40 ML = 1, NNN
0017 IF(KK.GT.1) GO TO 10
0018 GO TO 45
0019 10 IF(NL.GE.1) GO TO 20
0020 UTERM = UTERM - UQ(KK) * UD(M) - VQ(KK) * VO(M)
0021 VTERM = VTERM - UQ(KK) * VO(M) - VQ(KK) * UD(M)
0022 20 DO 40 ML = 1, NNN
0023 40 UDUMMY = UD(M) * UD(M) * VO(M) * VO(M)
0024 UQ(K+1) = UTERM * UD(M) * VTERM * VO(M) / UDUMMY
0025 VQ(K+1) = VTERM * UD(M) * UTERM * VO(M) / UDUMMY
0026 50 GO TO 100
0027 RETURN
0028 END
```
SUBROUTINE CONSQTIUXVXUY,VY)
C ***********************************************************************
C *
C * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
C *
C ***********************************************************************
DOUBLE PRECISION UX,UX,UY,VY,DUMMY,AAA,BBB
R=DSQRT(UX*UX+VX*VX)
AAA=DSQRT(DABS((R+UX)/2.0))
BBB=DSQRT(DABS((R-UX)/2.0))
IF(VX) 10,20,30
10 UY=AAA
VY=-1.0*BBB
GO TO 100
20 IF(UX) 40,50,60
30 UY=AAA
VY=BBB
GO TO 100
40 DUMMY=DABS(UX)
UY=0.0
VY=DSQRT(DUMMY)
GO TO 100
50 UY=0.0
VY=0.0
GO TO 100
60 DUMMY=DABS(UX)
UY=DSQRT(DUMMY)
VY=0.0
100 RETURN
END
TABLE II.3 (Continued)

0001 SUBROUTINE CALCUX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3

C晅EXISTENCE OF AN ELEMENT OF A POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF

C晅 THE QUADRATIC CLOSEST TO XIN). THIS ZERO IS THE NEW APPROXIMATION.

C晅 X(N+1) TO THE ZERO OF THE POLYNOMIAL.

C晅

0002 DOUBLE PRECISION ARG1, ARG2
0003 DOUBLE PRECISION UPX3, VPX3, UPX2, VPX2, UPX1, VPX1, UH3, VH3, UH2, VH2, UQ3, VQ3, UD, UB, VB, UC, UDISC, VDISC, UCCC, VC
0004 COMMON EPS1, EPS2, EPSRT, 102, MAX
0005 UH3=UX3-UX2
0006 VH3=VX3-VX2
0007 UH2=UX2-UX1
0008 VH2=VX2-VX1
0009 BBB=UH2*UH2+VH2*VH2
0010 UQ3=(UH3*UH2-UH3*VH2)/BBB
0011 VQ3=(VH3*UH2-UH3*VH2)/BBB
0012 UDDD=(UH3*UH2-UH3*VH2)/BBB
0013 VDDD=(VH3*UH2-UH3*VH2)/BBB
0014 UAAA=2.0*UQ3
0015 VAAA=2.0*VQ3
0016 UAAA=UAAA+1.0
0017 VB=VAAA+UAAA
0018 VC=VAAA-UAAA
0019 UCCC=UQ3*UQ3-VQ3*VQ3
0020 VCCC=VQ3*UQ3+UQ3*VQ3
0021 UH=((UAAA*UPX3-VAAA*VPX3)-(UBBB*UPX2-VBBB*VPX2)+I.UCCC*UPX1-UCCC*VPC1)
0022 VB=VAAA+UAAA
0023 VC=VAAA-UAAA
0024 UDISC=(UB*UB-VB*VB)+I(4.0*(UD*UC-VD*VC))
0025 VDISC=(2.0*(VB*UB))-I(4.0*(VD*UC+UD*VC))
0026 AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
0027 THETA=0.0
0028 GO TO 9
0029 IF (AAA.EQ.0.0) GO TO 5
0030 IF (AAA.EQ.0) GO TO 5
0031 GO TO 7
0032 GO TO 9
0033 UAAA=UAAA+1.0
0034 VAAA=VAAA+1.0
0035 GO TO 7
0036 IF (THETA.EQ.0.0) GO TO 9
0037 UAAA=UAAA+1.0
0038 VAAA=VAAA+1.0
0039 VTEST=RAD*UCOS(ANGLE)
0040 VTEST=RAD*SIGN(ANGLE)
0041 UDDD=UB*VTEST
0042 VDDD=VB*VTEST
0043 UDDD=UB*VTEST
0044 VDDD=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(HP, UA, VA, NAPP, UAPP, VAPP, NROOT, UROOT, VROOT, IROOT,
ISTART, XEND, NPOLY, URAPP, VRAPP)

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

0002 DOUBLE PRECISION UPX3, UPX2, UPX1, VPX3, VPX2, VPX1, UROOT, VROOT, UX1, VX1, UAPP, VAPP
C 1, UX2, VX2, UAPP, VAPP, UWORK, VWORK, UX3, UX4, UX5, UX6, VA, UA, VPX1, VPX2, URAPP, VRAPP
C ZAPP, UPX4, VPX4, EPSRT, EPS0, EPS5, CCP, EPSM, UH3, VX3, UQ4, VX4, ABPX4, ABPX3
C 3, 000, XSTART, XEND

0003 DIMENSION UROOT(25), VROOT(25), MULTI(25), UAPP(25, 3), VAPP(25, 3), UWORK
C 1(26), VWORK(26), UX1(26), UX2(26), UX3(26), UX4(26), UX5(26), VRAPP(25, 3), VRAPP(25, 3)

0004 LOGICAL CONV

0005 COMMON EPSM, EPS, EPS0, EPSRT, IO2, MAX

0006 DATA PNAME, ONAME / 2HP1, 2HD1 /

0007 EPSM=0.0000

0008 EPSRT=0.0009

0009 NROOT=0

0010 INROOT=0

0011 IPATH=1

0012 NOMUX=0

0013 NALTER=0

0014 ITIME=0

0015 [APP]=1

0016 ITER=1

0017 IF(NAPP.EQ.0) GO TO 10

0018 NAPP=HP

0019 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)

0020 GO TO 27

0021 10 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 VAPP(I, 3)=1.1*VAPP(I, 2)

0026 25 KKK=NP1

0027 GO 30 I=1, KKK

0028 UWORK(I)=UA(I)

0029 30 VWORK(I)=VA(I)

0030 NWORK=NP

0031 60 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(UWORK, VWORK, UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, VPX2, VPX3)

0038 CALL HORNER(UWORK, VWORK, UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, VPX2, VPX3)

0039 CALL HORNER(UWORK, VWORK, UX3, VX3, UPX3, VPX3)

0040 50 CALL CALL(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, VPX2, VPX3, UX4, VX4, UX5, VX5, UX6, VX6)

0041 60 CALL HORNER(UWORK, VWORK, UX4, VX4, UX5, VX5, UX6, VX6)

0042 ABPX4=DSORT(UPX4, UPX4+VPX4, VPX4+VPX4)

0043 ABPX3=DSORT(UPX3+VPX3, VPX3+VPX3)
TABLE H.III (Continued)

0044  IF(ABPX3.EQ.0.0) GO TO 70
0045  QQQA=ABPX4/ABPX3
0046  IF(QQQ.LE.1.0) GO TO 70
0047  UV4=0.5*QQQ
0048  UV4=0.5*UV4
0049  UX4=UX3*(UH3*QQQ-1.0-UH3*UV4)
0050  VX4=VX3*(UH3*QQQ+1.0-UH3*UV4)
0051  GO TO 60
0052  CALL TEST(UX3,VX3,UX4,VX4,CONV)
0053  IF(QUESTION) GO TO 120
0054  IF(ITER.LT.MAX) GO TO 110
0055  CALL ALTER(UAPP(IAPP),VAPP(IAPP),UX3,VX3,ITER,TIME)
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(102,1040)
0067  KKK=NWORK+1
0068  WRITE(102,1035) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0069  IF(IPATH.EQ.1) GO TO 82
0070  WRITE(102,1041) GO TO 82
0071  IF(NROOT.EQ.0) GO TO 90
0072  IF(IROOT.EQ.0) GO TO 81
0073  IF(NROOT.EQ.0) GO TO 90
0074  IF(XEND.EQ.0.0) GO TO 90
0075  IF(XSTART.GT.XEND) GO TO 90
0076  WRITE(102,1040) GO TO 100
0077  RETURN
0078  WRITE(I02,1045) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0079  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0080  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0081  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0082  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0083  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0084  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0085  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0086  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0087  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0088  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0089  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0090  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0091  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0092  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0093  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0094  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0095  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0096  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0097  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0098  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0099  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
0100  55 WRITE(102,1040) (DNAME,J,UWORK(J),VWORK(J),J=1,KKK)
TABLE H.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101</td>
<td>URAPP(NROOT, I) = UAPP(I, APP, I)</td>
</tr>
<tr>
<td>0102</td>
<td>VRAPP(NROOT, I) = VAPP(I, APP, I)</td>
</tr>
<tr>
<td>0103</td>
<td>125 IF (NOMULT .LT. NP) GO TO 130</td>
</tr>
<tr>
<td>0104</td>
<td>GO TO 60</td>
</tr>
<tr>
<td>0105</td>
<td>130 CALL HORNERN(WORK, UX4, VX4, UB, VP, VPX4)</td>
</tr>
<tr>
<td>0106</td>
<td>NWORK = NWORK - 1</td>
</tr>
<tr>
<td>0107</td>
<td>KKK = NWORK + 1</td>
</tr>
<tr>
<td>0108</td>
<td>DO 140 I = 1, KKK</td>
</tr>
<tr>
<td>0109</td>
<td>UWORK(I) = UB(I)</td>
</tr>
<tr>
<td>0110</td>
<td>VWORK(I) = VB(I)</td>
</tr>
<tr>
<td>0111</td>
<td>CALL HORNERN(WORK, UX4, VX4, UB, VP, VPX4)</td>
</tr>
<tr>
<td>0112</td>
<td>CCC = DSORT(UPX4<em>UPX4 + VPX4</em>VPX4)</td>
</tr>
<tr>
<td>0113</td>
<td>IF (CCC .LT. EPSM) GO TO 150</td>
</tr>
<tr>
<td>0114</td>
<td>IF (NWORK .GT. 2) GO TO 75</td>
</tr>
<tr>
<td>0115</td>
<td>IROOT = NROOT</td>
</tr>
<tr>
<td>0116</td>
<td>KKK = NWORK + 1</td>
</tr>
<tr>
<td>0117</td>
<td>DO 145 I = 1, KKK</td>
</tr>
<tr>
<td>0118</td>
<td>UB(I) = UWORK(KKK + 1 - I)</td>
</tr>
<tr>
<td>0119</td>
<td>VB(I) = VWORK(KKK + 1 - I)</td>
</tr>
<tr>
<td>0120</td>
<td>CALL QUADN(WORK, UB, VB, NROOT, UROOT, VROOT)</td>
</tr>
<tr>
<td>0121</td>
<td>GO TO 80</td>
</tr>
<tr>
<td>0122</td>
<td>MULT(NROOT) = MULT(NROOT) + 1</td>
</tr>
<tr>
<td>0123</td>
<td>NOMULT = NOMULT + 1</td>
</tr>
<tr>
<td>0124</td>
<td>GO TO 125</td>
</tr>
<tr>
<td>0125</td>
<td>110 UX1 = UX2</td>
</tr>
<tr>
<td>0126</td>
<td>VX1 = VX2</td>
</tr>
<tr>
<td>0127</td>
<td>UX2 = UX3</td>
</tr>
<tr>
<td>0128</td>
<td>VX2 = VX3</td>
</tr>
<tr>
<td>0129</td>
<td>UX3 = UX4</td>
</tr>
<tr>
<td>0130</td>
<td>VX3 = VX4</td>
</tr>
<tr>
<td>0131</td>
<td>UPX1 = UPX2</td>
</tr>
<tr>
<td>0132</td>
<td>VPX1 = VPX2</td>
</tr>
<tr>
<td>0133</td>
<td>UPX2 = UPX3</td>
</tr>
<tr>
<td>0134</td>
<td>VPX2 = VPX3</td>
</tr>
<tr>
<td>0135</td>
<td>UPX3 = UPX4</td>
</tr>
<tr>
<td>0136</td>
<td>VPX3 = VPX4</td>
</tr>
<tr>
<td>0137</td>
<td>IITER = IITER + 1</td>
</tr>
<tr>
<td>0138</td>
<td>GO TO 50</td>
</tr>
<tr>
<td>0139</td>
<td>1090 FORMAT(1X, 1X, 13H COEFFICIENTS OF DEFLECTED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND)</td>
</tr>
<tr>
<td>0140</td>
<td>1091 FORMAT(/1X, 13H ROOTS OF G(X), 21H INITIAL APPROXIMATION//)</td>
</tr>
<tr>
<td>0142</td>
<td>1086 FORMAT(1X, 9H SMROOT(I), 12, 4H) =, .023, 16, 3H + .023, 16, 2H</td>
</tr>
<tr>
<td>0143</td>
<td>1087 FORMAT(1X, 9H SMROOT(I), 12, 4H) =, .023, 16, 3H + .023, 16, 2H</td>
</tr>
<tr>
<td>0144</td>
<td>1085 FORMAT(1X, 9H SMROOT(I), 12, 4H) =, .023, 16, 3H + .023, 16, 2H</td>
</tr>
<tr>
<td>0145</td>
<td>1060 FORMAT(1X, I2, IX), 19, I3, IX) =, .023, 16, 3H</td>
</tr>
<tr>
<td>0146</td>
<td>END</td>
</tr>
</tbody>
</table>
SUBROUTINE GENAPP(APPR,APP1,NAPP,XSTART)

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

* SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE
* DEGREE OF THE ORIGINAL POLYNOMIAL.
******************************************************************************

DOUBLE PRECISION APPR,APP1,XSTART,EPSP1,EPSP2,EPSP3,BETA

DIMENSION APPR(25,3),APP1(25,3)

COMMON EPSM,EPSP1,EPSP2,EPSP3,IO2,MAX

IF XSTART.EQ.0.0) XSTART*0.5

BETA=0.2617994

DO 10 I=1,NAPP

APP1(I,2)=XSTART*DCOS(BETA)

APP1(I,2)=XSTART*DSIN(BETA)

BETA=BETA+0.5235988

10 XSTART=XSTART+0.5

DO 20 I=1,NAPP

APP1(I,1)=0.9*APP1(I,2)

APP1(I,1)=0.9*APP1(I,2)

APP1(I,3)=1.1*APP1(I,2)

APP1(I,3)=1.1*APP1(I,2)

RETURN

END
TABLE H.11 (Continued)

0001 C SUBROUTINE ALTER(X1R,X11,X2R,X21,X3R,X31,NALTER,TIME)  
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,X11,X2R,X21,X3R,X31  
0003 DOUBLE PRECISION EPS1, EPS2, EPS3, R, BETA  
0004 COMMON EPS1, EPS2, EPS3, 102, MAX  
0005 IF TIME.NE.0  
0006 TIME=1  
0007 WRITE(102,1010) MAX  
0008 5 IFINALTER.EQ.0  
0009 GO TO 10  
0010 TO 20  
0011 10 R=DSUM1(X2R*X2R+X21*X21)  
0012 BETA=DATAN2(X2R,X2R)  
0013 WRITE(102,10201) X1R,X11,X2R,X21,X3R,X31  
0014 20 NALTER=NALTER+1  
0015 IFINALTER.GT.5 RETURN  
0016 GO TO (30,40,50,60,70) NALTER  
0017 30 X2R=-X2R  
0018 X21=-X21  
0019 GO TO 50  
0020 40 BETA=BETA+1.0471976  
0021 X2R=R*DCOS(BETA)  
0022 X21=R*DSSIN(BETA)  
0023 50 X1R=0.9*X2R  
0024 X11=0.9*X21  
0025 X3R=1.1*X2R  
0026 X31=1.1*X21  
0027 RETURN  
0028 1000 FORMAT(1X,5HMX1 = ,D23.16,H3 + ,D23.16,2H I,10X,22HALTERED APPROXIM  
4ATIONS/X1,5HX2 = ,D23.16,3H + ,D23.16,2H I/I1X,5HX3 = ,D23.16,3H +  
4 Z - D23.16,2H )/  
0029 1020 FORMAT(1H0,5HMX1 = ,D23.16,3H + ,D23.16,2H I,10X,22INITIAL APPROXI  
4ATIONS/X1,5HX2 = ,D23.16,3H + ,D23.16,2H I/I1X,5HX3 = ,D23.16,3H +  
4 Z -D23.16,2H )/  
0030 1010 FORMAT(///1X,5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF  
4TER 15,12H ITERATIONS//)  
0031 END
TABLE H.111 (Continued)

```fortran
001 SUBROUTINE BETTER(UA, VA, NP, UROOT, VROOT, NROOT, UAPP, VRAPP, IROOT, MUL
002 ************************************************************************************
003 * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND *
004 * BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO *
005 * THE FULL, UNDEFLATED POLYNOMIAL. *
006 ************************************************************************************
007 DOUBLE PRECISION UROOT, VROOT, UA, VA, UAPP, VRAPP, UX, VX, U, V, UX, VX, U2, V2, U3, V3
008 COMMON EPSM, EPS, EPSD, EPSRT, IO2, MAX
009 IF(NROOT.LE.1) RETURN
010 L=0
011 DO 10 I=1,NROOT
012 UAPP(I,1)=UROOT(I)*EPSRT
013 UAPP(I,2)=UROOT(I)
014 UAPP(I,3)=UROOT(I)*12.0*EPSRT
015 10 U=UAPP(I,3)
016 DO 100 J=1,NROOT
017 UX=UAPP(J,1)
018 VX=VAPP(J,1)
019 UX2=UAPP(J,2)
020 VX2=VAPP(J,2)
021 UX3=UAPP(J,3)
022 VX3=VAPP(J,3)
023 ITER=1
024 CALL HORNER(NP, UA, VX, UX, UX, VX, U, V, UX, VX)
025 CALL HORNER(NP, VA, VX, UX, UX, VX, U, V, UX, VX)
026 CALL HORNER(NP, UA, UX, VX, UX, VX, U, V, UX, VX)
027 CALL CALC(UX, VX, UX2, UX3, VX3, UX, VX, UX3, VX2, VX3, VX3, UX)
028 DO 20 I=1,14, VX4, VX4, VX4, VX4, VX4, VX4
029 CALL TEST(UX3, VX3, VX4, VX4, VX4, CONV)
030 IF(CONV) GO TO 50
031 IF(ITER.LE.MAX) GO TO 40
032 WRITE(IO2, 1000) J, UROOT(IJ), VROOT(IJ), MAX
033 IF(J.LT.IROOT) GO TO 33
034 IF(J.GT.IROOT) GO TO 35
035 GO TO 100
036 IF(KK+KROOT+1) GO TO 33
037 IF(KK+KROOT+1) GO TO 33
038 UAPP(K,1)=UAPP(K+1,1)
039 VRAPP(K,1)=VRAPP(K+1,1)
040 UAPP(K,2)=UAPP(K+1,2)
041 VRAPP(K,2)=VRAPP(K+1,2)
042 UAPP(K,3)=UAPP(K+1,3)
043 VRAPP(K,3)=VRAPP(K+1,3)
044 IF(KK+KROOT+1) GO TO 33
045 GO TO 100
```
TABLE H.III. (Continued)

0046 40 UX1=UX2
0047 UX1=UX2
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(I)=UX4
0060 VROOTS(I)=VX4
0061 100 CONTINUE
0062 IF(L.EQ.0) GO TO 120
0063 DO 110 I=1,L
0064 UROOT(I)=UROOTS(I)
0065 VROOT(I)=VROOTS(I)
0066 NROOT=L
0067 RETURN
0068 120 NROOT=0
0069 RETURN
0070 1000 FORMAT(30H THE PRESENT APPROXIMATION IS ,D23.16,2H I/1/)
0071 1010 FORMAT(30H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(I),12,4H) * ,
0072 1023.16,3H + ,023.16,2H I/24H DID NOT CONVERGE AFTER ,13.11H ITERAT
0073 2IONS)
0074 END
TABLE H. III (Continued)

0001
SUBROUTINE TEST(U3, V3, U4, V4, VX, VX)  

****************************************************
 * SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
 * IMATIONS BY TESTING THE EXPRESSION
 * ABSOLUTE VALUE OF [X(N)(1)-X(N)/ABSOLUTE VALUE OF X(N)(1)].
 * WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
****************************************************

0002
DOUBLE PRECISION U3, V3, U4, V4, VX, VX

0003
DOUBLE PRECISION EPSM

0004
LOGICAL CONV

0005
COMMON EPSM, EPS, EPSO, EPSRT, IO2, MAX

0006
UDUMMY = U3 - U4

0007
VDUMMY = VX - U3

0008
AAA = DSQRT(UDUMMY*UDUMMY + VDUMMY*VDUMMY)

0009
DENOM = DSQRT(U3*U3 + V4*V4 + VX)

0100
IF(DENOM, LT, EPS) GO TO 10

0101
IF(AAA/DENOM, LT, EPS) GO TO 10

0102
5 CONV. = FALSE.

0103
10 GO TO 100

0104
10 CONV. = TRUE.

0105
15 GO TO 100

0106
20 IF(AAA, LT, EPSO) GO TO 10

0107
10 GO TO 5

0118
100 RETURN

0119
END

0001
SUBROUTINE HORNER(NA, UA, VA, UX, VX, U8, VB, UPX, VPX)

****************************************************
 * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D.
 * SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE
 * FACTOR (X-D).
****************************************************

0002
DOUBLE PRECISION U3, V3, UPX, VPX, UB, VB, UA, VA

0003
DIMENSION UA[24], VA[24], UB[24], VB[24]

0004
UB(1) = U1(1)

0006
VB(1) = VA(1)

0008
NUM = N+1

0100
DO 10 I = 2, NUM

0108
UB(1) = UB(1) + UB(I-1)*UX - VB(I-1)*VX

0110
UPX = UB(NUM)

0111
VPX = VB(NUM)

0112
RETURN

0113
END