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
MULTIPLE ZEROS OF POLYNOMIALS

by

CRAIG A. WOOD

Department of Mathematics
Oklahoma State University
Stillwater, Oklahoma

A National Aeronautics and Space Administration

Research Grant

Grant Number NASA NGR 37-002-084
Abstract: MULTIPLE ZEROS OF POLYNOMIALS

Grant: The National Aeronautics and Space Administration

Grant Number: NASA NGR 37-002-084

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

Various classical methods exist for extracting the zeros of a polynomial

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

where \( a_1 \neq 0 \) and \( a_1, a_2, \ldots, a_{N+1} \) are complex numbers, when \( N=1,2,3,4 \).

For polynomials of higher degree, iterative numerical methods must be used. In this material four iterative methods are presented for approximating the zeros of a polynomial using a digital computer. Newton's method and Muller's method are two well known iterative methods which are presented. They extract the zeros of a polynomial by generating a sequence of approximations converging to each zero. However, both of these methods are very unstable when used on a polynomial which has multiple zeros. That is, either they fail to converge to some or all of the zeros, or they converge to very bad approximations of the polynomial's zeros.

This material introduces two new methods, the 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 appendices 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

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. NEWTON'S METHOD</td>
<td>4</td>
</tr>
<tr>
<td>Derivation of the Algorithm</td>
<td>4</td>
</tr>
<tr>
<td>Convergence of Newton's Method</td>
<td>8</td>
</tr>
<tr>
<td>Procedure for Newton's Method</td>
<td>9</td>
</tr>
<tr>
<td>Geometrical Interpretation of Newton's Method</td>
<td>10</td>
</tr>
<tr>
<td>Determining Multiple Roots</td>
<td>10</td>
</tr>
<tr>
<td>III. MULLER'S METHOD</td>
<td>12</td>
</tr>
<tr>
<td>Derivation of the Algorithm</td>
<td>12</td>
</tr>
<tr>
<td>Procedure for Muller's Method</td>
<td>19</td>
</tr>
<tr>
<td>Geometrical Interpretation of Muller's Method</td>
<td>20</td>
</tr>
<tr>
<td>Determining Multiple Roots</td>
<td>20</td>
</tr>
<tr>
<td>IV. GREATEST COMMON DIVISOR METHOD</td>
<td>22</td>
</tr>
<tr>
<td>Derivation of the Algorithm</td>
<td>22</td>
</tr>
<tr>
<td>Determining Multiplicities</td>
<td>29</td>
</tr>
<tr>
<td>Procedure for the G.C.D. Method</td>
<td>29</td>
</tr>
<tr>
<td>V. REPEATED GREATEST COMMON DIVISOR.</td>
<td>30</td>
</tr>
<tr>
<td>Derivation of the Algorithm</td>
<td>30</td>
</tr>
<tr>
<td>Procedure for the Repeated G.C.D. Method</td>
<td>32</td>
</tr>
<tr>
<td>VI. CONCLUSION</td>
<td>34</td>
</tr>
<tr>
<td>Polynomials With all Distinct Roots</td>
<td>34</td>
</tr>
<tr>
<td>Polynomials With Multiple Roots</td>
<td>35</td>
</tr>
<tr>
<td>Time Comparisons</td>
<td>40</td>
</tr>
<tr>
<td>Sample Results</td>
<td>42</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>85</td>
</tr>
<tr>
<td>APPENDIX A - SPECIAL FEATURES OF NEWTON'S AND MULLER'S PROGRAMS</td>
<td>86</td>
</tr>
<tr>
<td>Generating Approximations</td>
<td>86</td>
</tr>
<tr>
<td>Altering Approximations</td>
<td>88</td>
</tr>
<tr>
<td>Searching the Complex Plane</td>
<td>89</td>
</tr>
</tbody>
</table>
## Chapter 1: Improving Zeros Found
- Solving Quadratic Polynomial .......................... 91
- Missing Roots ........................................ 91
- 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

- 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 ............ 214
  Use of the Program .................................. 214
  Input Data for G.C.D. - Muller's Method ............. 216
  Variables Used in G.C.D. - Muller's Method .......... 216
  Description of Program Output ......................... 216
  Informative and Error Messages ...................... 216
  Flow Charts ........................................... 222
  Program Listing ....................................... 234

APPENDIX G - REPEATED G.C.D. - NEWTON'S METHOD ...... 253
  Use of the Program .................................. 253
  Input Data for Repeated G.C.D. - Newton's Method ...... 255
  Variables Used in Repeated G.C.D. - Newton's Method .... 255
  Description of Program Output ......................... 255
  Informative and Error Messages ...................... 256
  Flow Charts ........................................... 260
  Program Listing ....................................... 270

APPENDIX H - REPEATED G.C.D. - MULLER'S METHOD ...... 286
  Use of the Program .................................. 286
  Input Data for Repeated G.C.D. - Muller's Method ...... 287
  Variables used in Repeated G.C.D. - Muller's Method .... 288
  Description of Program Output ......................... 288
  Informative and Error Messages ...................... 289
  Flow Charts ........................................... 290
  Program Listing ....................................... 301
CHAPTER I

INTRODUCTION

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

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

where \( a_1 \neq 0 \) and the coefficients \( a_1, a_2, \ldots, a_{N+1} \) are complex numbers.

Various classical methods calculate the exact roots of polynomials of degree 1, 2, 3, or 4. For polynomials of higher degree, no such methods exist. Thus, to solve for the zeros of such polynomials, numerical methods of iteration based on successive approximations must be employed. In the following material four such methods are given which are particularly suited for modern high speed computers.

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

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

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 re-checked 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

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

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

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 \( a \) from the \( n \)th approximation, \( x_n \), by

\[ x_{n+1} = x_n - \frac{P(x_n)}{P'(x_n)} \tag{2-2} \]

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

\[ x_0, x_1, x_2, \ldots, x_n, \ldots \tag{2-3} \]

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

In order to use equation (2-2), it is necessary to compute, for each \( x_n \), the value of the polynomial, \( P(x_n) \), and its derivative, \( P'(x_n) \). The division algorithm states that if \( P(X) \) and \( G(X) \) are polynomials, then there exists polynomials \( H(X) \) and \( K(X) \) such that

\[ P(X) = H(X) G(X) + K(X) \]

where \( K(X) = 0 \) or \( \deg. K(X) < \deg. G(X) \). From this expression of \( P(X) \), the following remainder theorem is obtained:
Theorem 2.1. If \( P(X) \) is a polynomial and \( c \) is a complex number, then the remainder obtained from dividing \( P(X) \) by \( (X - c) \) is \( P(c) \).

The proof of Theorem 2.1 is given in [3, P. 102]. Thus, \( P(X) \) can be written as \( P(X) = (X - c) \cdot H(X) + R \) 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 \( P(X) = (X - c) \cdot H(X) + R \) 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

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

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

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

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)

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

Thus

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

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 $\alpha_i$ is a zero of $P_{i-1}(X), P_{i-2}(X), \ldots, P_1(X), P(X)$.

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

2. Convergence of Newton's Method

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

**Theorem 2.4.** Let $P(X)$ be a polynomial and let the following conditions be satisfied on the closed interval $[a, b]$:
1. \( P(a) P(b) < 0 \)
2. \( P'(X) \neq 0, X \in [a, b] \).
3. \( P''(X) \) is either \( \geq 0 \) 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
\((i+1)\)th iteration of Newton's algorithm is proportional to the square
of the error obtained in the \( i \)th iteration. A proof of quadratic con-
vergence can be found in [L, 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
   \[
   X_{n+1} = X_n - \frac{P(X_n)}{P'(X_n)}.
   \]
2. Test for convergence; that is, test
   \[
   \left| \frac{P(X_n)}{P'(X_n)} \right| \left| X_{n+1} \right| < \varepsilon
   \]
   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

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

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

\[ P_1(X) = (X - α_1)^{e_1} (X - α_2)^{e_2} \cdots (X - α_j)^{e_j-1} \cdots (X - α_m)^{e_m}. \]

Evaluating P_1(X) at the zero, α_j, gives P_1(α_j) = 0 if e_j > 1. Thus, after a zero, α, of P(X) is determined by Newton's iterative process and the current polynomial is deflated giving P_1(X), then P_1(α) is evaluated. If P_1(α) ≤ ε for some small number ε, α is a root of P_1(X) and thus has multiplicity at least equal to two. P_1(X) is then deflated giving P_2(X). If P_2(α) ≤ ε, α 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 \( P_k(α) > ε \). α is then a zero of multiplicity k+1.

Figure 2.1. Geometrical Interpretation of Newton's Method
CHAPTER III

MULLER'S METHOD

1. Derivation of the Algorithm

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

Consider the polynomial

\[ P(X) = a_1 X^N + a_2 X^{N-1} + \ldots + a_N X + a_{N+1} \]  (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
\[ Q(x) = \frac{(X - X_{n-1})(X - X_{n-2})}{(X_n - X_{n-1})(X_n - X_{n-2})} P(X_n) \]

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

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

which can be rewritten as

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

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

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

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

In order to simplify this expression, introduce the quantities

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

Then

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

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

\[ - \frac{h(h + h_n + h_{n-1})}{h_n h_{n-1}} P(X_{n-1}) \]
\[
\begin{align*}
&= \frac{h(h + h_n)}{(h_n + h_{n-1})h_{n-1}} P(X_{n-2}) \\
&= \frac{h^2 + 2hh_n + hh_{n-1} + h_n^2 + h_nh_{n-1}}{h_n^2 + h_nh_{n-1}} P(X_n) \\
&- \frac{h^2 + hh_n + hh_{n-1}}{h} P(X_{n-1}) \\
&+ \frac{h^2 + hh_n}{h_nh_{n-1} + h_{n-1}} P(X_{n-2}).
\end{align*}
\]

Collecting terms containing like powers of \(h\) produces

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

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

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

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

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

\[
+ \left( \frac{(2h_n + h_{n-1}) P(X_n)}{h_n^2 + h_nh_{n-1}} - \frac{(h_n + h_{n-1}) P(X_{n-1})}{h_nh_{n-1}} + \frac{h_n P(X_{n-2})}{h_nh_{n-1} + h_{n-1}^2} \right) h
\]
Using the common denominator, \( h_n^2 h_{n-1} + h_n h_{n-1} \), and combining terms yields

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

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

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

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

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

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

\[
+ \left( \frac{h_n^2 h_{n-1} + h_n h_{n-1}}{h_n^3 + h_n^2} \right) P(X_n)
\]
\[
\begin{align*}
&\left(\frac{h_n}{h_{n-1}} + h_n^2 P(X_n)\right) + \\
&\left(\frac{h_n^3}{h_{n-1}} + h_n^2\right) P(X_n)
\end{align*}
\]

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

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

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

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

Now let

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

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

\[
C_n = (q_n + 1) P(X_n).
\]

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:

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

and the new approximation is found as follows:

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

Thus

\[
X_{n+1} = X_n + h_n \, q_{n+1}.
\]

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

\[
q_{n+1} = \frac{-B_n + \sqrt{B_n^2 - 4A_nC_n}}{2A_n} = \frac{B_n + \sqrt{B_n^2 - 4A_nC_n}}{2A_n} - \frac{B_n}{2A_n} \left( \frac{B_n}{B_n + \sqrt{B_n^2 - 4A_nC_n}} \right)
\]

\[
q_{n+1} = \frac{-2C_n}{B_n + \sqrt{B_n^2 - 4A_nC_n}}. \tag{3-3}
\]

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, \( \alpha \), of \( P(X) \) has been found, \( P(X) \) is deflated as described in Chapter II, §1, and the process repeated to extract a zero, \( \alpha_1 \), 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
taken 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
calculate the quantity \(|P(X_{n+1})|/|P(X_n)|\) whenever the value \(P(X_{n+1})\) is
calculated. If the former quantity exceeds ten (10), \(q_{n+1}\) is halved
and \(h_n, X_{n+1}, \) and \(P(X_{n+1})\) are recomputed accordingly.

2. Procedure for Muller's Method

The basic steps performed by Muller's method are listed sequen-
tially as follows, starting with initial approximations \(X_1, X_2, \) and \(X_3\).

1. Compute \(h_n, q_n, D_n, B_n, C_n, q_{n+1}\) as defined previously.
2. Compute the next approximation \(X_{n+1}\) by
   \[X_{n+1} = X_n + h_n q_{n+1}\]
3. Test for convergence; that is, test
   \[|X_{n+1} - X_n|/|X_{n+1}| < \varepsilon\]
   for some suitably small number \(\varepsilon\).
4. If the test fails, return to step 1 with the last
   three approximations \(X_{n+1}, X_n, X_{n-1}\).
5. If the test passes, do the following:
   a. Save $X_{n+1}$ as the desired approximation to a zero.
   b. Deflate the current polynomial using $X_{n+1}$.
   c. Replace the current polynomial by the deflated polynomial.
   d. Return to step 1 with a new set of initial approximations.

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

3. Geometrical Interpretation of Muller's Method

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

4. Determining Multiple Roots

For a discussion concerning multiple roots see Chapter II, §5.
Figure 3.1. Geometrical Interpretation of Muller's Method
CHAPTER IV

GREATEST COMMON DIVISOR METHOD

1. Derivation of the Algorithm

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

Consider the Nth degree polynomial

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

where \( a_1 \neq 0 \) and \( a_1, a_2, \ldots, a_{N+1} \) are complex numbers. If \( P(X) \) 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

\[ P'(X) = N a_1 X^{N-1} + (N-1) a_2 X^{N-2} + \ldots + a_N \]

which can also be expressed as
\[ P'(X) = a_1 (X - a_1)^{e_1-1} (X - a_2)^{e_2-1} \ldots (X - a_m)^{e_m-1} \sum_{i=1}^{m} \prod_{j=1, j \neq i}^{m} (X - a_j). \]

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

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

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., \( D(X) \).
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
\[ R_{i+1}(X) = R_{i+2}(X) S_{i+1}(X) + R_{i+2}(X) \]

where either \( R_{i+2}(X) = 0 \) or \( \text{deg. } R_{i+2}(X) < \text{deg. } R_{i+1}(X) \). Then we obtain a sequence \( R_0(X), R_1(X), \ldots, R_K(X), R_{K+1}(X) \) such that

\[ \text{deg. } R_1(X) < \text{deg. } R_{i-1}(X), \quad i = 1, 2, \ldots, K+1. \]

Since a polynomial cannot have degree less than zero, the above process, in a finite number of steps (at most \( N_1 \)), results in polynomials \( R_{K-1}(X), S_K(X) \) and \( R_K(X) \) with \( \text{deg. } R_K(X) < \text{deg. } R_{K-1}(X) \) 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

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

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

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

Then define \( T_1(X) \) by

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

\[
= \left[ r_{i-1,1} - r_{i,1} \left( r_{i-1,1} \big/ r_{i,1} \right) \right] X^{N_{i-1}}
\]

\[
+ \left[ r_{i-1,2} - r_{i,2} \left( r_{i-1,1} \big/ r_{i,1} \right) \right] X^{N_{i-1}-1}
\]

\[
+ \ldots
\]

\[
+ \left[ r_{i-1,N_i-1+1} - r_{i,N_i-1+1} \left( r_{i-1,1} \big/ r_{i,1} \right) \right]
\]

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

We consider three cases.

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

(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 \( \deg T_1(X) = M_1 > N_i \), then define \( U_2(X) \) by

\[
U_2(X) = \left( \frac{t_{1,1}}{r_{i,1}} \right)^{M_1-N_i} X^{M_1-N_i}
\]

where

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

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

\[
T_2(X) = \left[ t_{1,1} - \left( \frac{t_{1,1}}{r_{i,1}} \right) r_{i,1} \right] X^{M_1-1}
+ \left[ t_{1,2} - \left( \frac{t_{1,1}}{r_{i,1}} \right) r_{i,2} \right] X^{M_1-2}
+ \ldots
+ \left[ t_{1,M_1+1} - \left( \frac{t_{1,1}}{r_{i,1}} \right) r_{i,M_1+1} \right]
\]

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

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

(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 \( \deg T_2(X) = M_2 > N_i \), then define \( U_3(X) \) by

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

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

1. \( T_S(X) = 0 \) and \( R_i(X) \) is the g.c.d. of \( R_0(X) \) and \( R_1(X) \) or
2. \( T_S(X) \neq 0 \) but \( \deg T_S(X) < \deg R_i(X) \), in which case \( T_S(X) = R_{i+1}(X) \).

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

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

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

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

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

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

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

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

1. Given a polynomial, $P(X)$, in the form
   
   $$P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_N + a_{N+1}.$$ 

2. Calculate the derivative, $P'(X)$, of $P(X)$ in the form
   
   $$P'(X) = b_1X^{N-1} + b_2X^{N-2} + \ldots + b_N$$
   
   where $b_1 = Na_1$, 
   
   $$b_2 = (N-1)a_2, \ldots, b_N = a_N.$$ 

3. Find $D(X)$, the g.c.d. of $P(X)$ and $P'(X)$ using the algorithms developed above.

4. Calculate $H(X) = P(X)/D(X)$, the polynomial having only simple zeros.

5. Use some appropriate method to extract the zeros of $H(X)$.

6. Determine the multiplicity of each of the zeros obtained in step 5.
CHAPTER V

REPEATED GREATEST COMMON DIVISOR METHOD

1. Derivation of the Algorithm

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

Let

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

where \( a_1 \neq 0 \), each \( a_i \) is a complex number, and \( a_1, a_2, \ldots, a_m \) 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

\[
D_1(X) = (X - a_1)^{e_1-1} (X - a_2)^{e_2-1} \ldots (X - a_m)^{e_m-1}
\]

where we assume that if \( e_j = 1 \), then \( X - a_j \) 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

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

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

$$G_1(X) = \left[ \frac{P(X)}{D_1(X)} \right] / \left[ \frac{D_1(X)}{D_2(X)} \right] = \frac{P(X)D_2(X)}{[D_1(X)]^2}$$

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

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

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 \( i \geq 0 \), 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 of 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 \( \deg D_0(X) = M \), \( D_0(X) = d_1X^M + \ldots + d_{M+1} \),
and \( b_1 = Md_1, b_2 = (M-1)d_2, \ldots, b_M = d_M \).

4. Find \( D_1(X) \), the g.c.d. of \( D_0(X) \) and \( D'_0(X) \) using the algorithms developed in Chapter IV.

5. Similar to 3., calculate \( D'_1(X) \).

6. Find \( D_2(X) \), the g.c.d. of \( D_1(X) \) and \( D'_1(X) \) using the algorithms developed in Chapter IV.

7. Calculate \( G(X) = D_0(X) D_2(X) / [D_1(X)]^2 \).

8. Use some appropriate method to extract the zeros of \( G(X) \) and assign these zeros the correct multiplicity as zeros of \( P(X) \).

9. Set \( D_0(X) = D_1(X), D_0'(X) = D'_1(X) \), and \( D_1(X) = D_2(X) \). Then repeat 5.-8. above until all the zeros of \( P(X) \) are found.
CHAPTER VI

CONCLUSION

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

1. polynomials with all distinct zeros.

2. polynomials with multiple zeros.

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

1. Polynomials With all Distinct Zeros

First we consider the class of polynomials having distinct zeros. Newton's method is particularly suited for this class of polynomials. Its quadratic convergence is very fast which can save time and money to the user. The accuracy obtained is excellent as shown in Exhibit 6.1 which presents the zeros of a 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+1 (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+3i (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>
<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.
THE COEFFICIENTS OF \( p_i(x) \) ARE

\[
\begin{align*}
p_{11} &= 0.3000000000000000 \ D\ 00 + 0.0000000000000000 \ D\ 00 \ I \\
p_{21} &= -0.1700000000000000 \ D\ 00 + 0.0000000000000000 \ D\ 00 \ I \\
p_{31} &= 0.2100000000000000 \ D\ 00 - 0.6576000000000000 \ D\ 00 \ I \\
p_{41} &= 0.1745000000000000 \ D\ 00 + 0.2843600000000000 \ D\ 00 \ I \\
p_{51} &= -0.0394200000000000 \ D\ 00 + 0.3118000000000000 \ D\ 00 \ I \\
p_{61} &= 0.3276360000000000 \ D\ 00 - 0.3069000000000000 \ D\ 00 \ I \\
p_{71} &= 0.1329084000000000 \ D\ 00 + 0.4710520000000000 \ D\ 00 \ I \\
p_{81} &= -0.5611880000000000 \ D\ 00 + 0.1674270000000000 \ D\ 00 \ I \\
p_{91} &= 0.7226756000000000 \ D\ 00 - 0.1546260000000000 \ D\ 00 \ I \\
p_{101} &= 0.3270972000000000 \ D\ 00 - 0.2000000000000000 \ D\ 00 \ I \\
p_{111} &= -0.1241472000000000 \ D\ 00 - 0.4097180000000000 \ D\ 00 \ I \\
p_{121} &= 0.5490230000000000 \ D\ 00 - 0.2243480000000000 \ D\ 00 \ I \\
p_{131} &= 0.6468334000000000 \ D\ 00 + 0.1285440000000000 \ D\ 00 \ I \\
p_{141} &= 0.1467466000000000 \ D\ 00 + 0.2272000000000000 \ D\ 00 \ I \\
p_{151} &= 0.1077120000000000 \ D\ 00 - 0.5479600000000000 \ D\ 00 \ I \\
p_{161} &= 0.3459600000000000 \ D\ 00 - 0.1267200000000000 \ D\ 00 \ I \\
\end{align*}
\]

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0

MAXIMUM NUMBER OF ITERATIONS: 200

TEST FOR CONVERGENCE: 0.100-00

TEST FOR MULTIPLICITIES: 0.100-01

RADIUS TO START SEARCH: 0.000 00

RADIUS TO END SEARCH: 0.000 00

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

\[
\begin{align*}
\text{ROOT1} &= 0.3000000000000000 \ D\ 00 + 0.16095815165319-16 \ I \\
\text{ROOT2} &= 0.2300000000000000 \ D\ 00 + 0.0000000000000000 \ D\ 00 \ I \\
\text{ROOT3} &= 0.2950000000000000 \ D\ 00 - 0.1000000000000000 \ D\ 00 \ I \\
\text{ROOT4} &= 0.2950000000000000 \ D\ 00 - 0.1000000000000000 \ D\ 00 \ I \\
\text{ROOT5} &= 0.1000000000000000 \ D\ 00 - 0.53219557659946-16 \ I \\
\text{ROOT6} &= 0.1000000000000000 \ D\ 00 - 0.99999999999999 \ D\ 00 \ I \\
\text{ROOT7} &= 0.1000000000000000 \ D\ 00 - 0.99999999999999 \ D\ 00 \ I \\
\text{ROOT8} &= 0.2000000000000000 \ D\ 00 - 0.3000000000000000 \ D\ 00 \ I \\
\text{ROOT9} &= 0.1000000000000000 \ D\ 00 - 0.1000000000000000 \ D\ 00 \ I \\
\text{ROOT10} &= 0.2000000000000000 \ D\ 00 - 0.1000000000000000 \ D\ 00 \ I \\
\text{ROOT11} &= 0.2000000000000000 \ D\ 00 - 0.53219557659946-16 \ I \\
\text{ROOT12} &= 0.2000000000000000 \ D\ 00 - 0.53219557659946-16 \ I \\
\text{ROOT13} &= 0.53219557659946-16 \ I \\
\text{ROOT14} &= 0.53219557659946-16 \ I \\
\text{ROOT15} &= -0.3333333333333334 \ D\ 00 - 0.18503710770594-15 \ I \\
\end{align*}
\]

EXHIBIT 6.1.
AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF \( f(x) \) ARE

<table>
<thead>
<tr>
<th>ROOTS OF ( f(x) )</th>
<th>MULTIPlicITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 ) = 0.30000000000000000000</td>
<td>1</td>
<td>0.492942911564627900D 00 + 0.1294096520843081D 00</td>
</tr>
<tr>
<td>( c_2 ) = 0.70100000000000000000</td>
<td>1</td>
<td>0.70710677530434060D 00 + 0.70710677530434060D 00</td>
</tr>
<tr>
<td>( c_3 ) = 0.1126043043762753D 00</td>
<td>1</td>
<td>0.5174342551642240D 00 + 0.5174342551642240D 00</td>
</tr>
<tr>
<td>( c_4 ) = -0.10000000000000000000</td>
<td>1</td>
<td>0.3827375832074700D 00 + 0.3827375832074700D 00</td>
</tr>
<tr>
<td>( c_5 ) = 0.41999999999999999999</td>
<td>1</td>
<td>0.3340742133327901D 00 + 0.3340742133327901D 00</td>
</tr>
<tr>
<td>( c_6 ) = 0.20000000000000000000</td>
<td>1</td>
<td>0.2626466070759601D 00 + 0.2626466070759601D 00</td>
</tr>
<tr>
<td>( c_7 ) = -0.10000000000000000000</td>
<td>1</td>
<td>0.1051517646481460D 00 + 0.1051517646481460D 00</td>
</tr>
<tr>
<td>( c_8 ) = 0.40000000000000000000</td>
<td>1</td>
<td>0.3718767624893690D 00 + 0.3718767624893690D 00</td>
</tr>
<tr>
<td>( c_9 ) = 0.10000000000000000000</td>
<td>1</td>
<td>0.9048494013086560D 00 + 0.9048494013086560D 00</td>
</tr>
<tr>
<td>( c_{10} ) = -0.20000000000000000000</td>
<td>1</td>
<td>0.6131869319379140D 00 + 0.6131869319379140D 00</td>
</tr>
<tr>
<td>( c_{11} ) = 0.20000000000000000000</td>
<td>1</td>
<td>0.728776548089506D 00 + 0.728776548089506D 00</td>
</tr>
<tr>
<td>( c_{12} ) = -0.20000000000000000000</td>
<td>1</td>
<td>0.6131869319379140D 00 + 0.6131869319379140D 00</td>
</tr>
<tr>
<td>( c_{13} ) = 0.20000000000000000000</td>
<td>1</td>
<td>0.728776548089506D 00 + 0.728776548089506D 00</td>
</tr>
<tr>
<td>( c_{14} ) = 0.80000000000000000000</td>
<td>1</td>
<td>0.3718767624893690D 00 + 0.3718767624893690D 00</td>
</tr>
<tr>
<td>( c_{15} ) = -0.30000000000000000000</td>
<td>1</td>
<td>0.1294096520843081D 00 + 0.1294096520843081D 00</td>
</tr>
</tbody>
</table>

Exhibit 6.1, Roots Are: \(-1 - i, 1 + i, -2 - 3i, 2 - i, 3, 2, i, -i, -10/3, 3, -1, 1, 4 + 4i, -.2 + .2i, .2 - .2i.\)
MULLER'S METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
POLYNOMIAL NUMBER 7 OF DEGREE 15

THE COEFFICIENTS OF PI(X) ARE

| Pi | 1 | Pi | 2 | Pi | 3 | Pi | 4 | Pi | 5 | Pi | 6 | Pi | 7 | Pi | 8 | Pi | 9 | Pi | 10 | Pi | 11 |
|----|---|----|---|----|---|----|---|----|---|----|---|----|---|----|---|----|---|----|---|----|
| P1 | 1.5 | P2 | 0.174 | P3 | 0.251 | P4 | 0.174 | P5 | 0.759 | P6 | 0.874 | P7 | 0.132 | P8 | 0.761 | P9 | 0.727 | P10 | -0.124 | P11 | 0.540 |
| P2 | 0.707 | P3 | 0.707 | P4 | 0.707 | P5 | 0.707 | P6 | 0.707 | P7 | 0.707 | P8 | 0.707 | P9 | 0.707 | P10 | 0.707 | P11 | 0.707 | P12 | 0.707 |

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

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF PI(X)

| ROOT | 1 | ROOT | 2 | ROOT | 3 | ROOT | 4 | ROOT | 5 | ROOT | 6 | ROOT | 7 | ROOT | 8 | ROOT | 9 | ROOT | 10 | ROOT | 11 |
|------|---|------|---|------|---|------|---|------|---|------|---|------|---|------|---|------|---|------|---|------|
| RO1 | 0.300 | RO2 | 0.200 | RO3 | 0.949 | RO4 | 0.236 | RO5 | -0.100 | RO6 | -0.333 | RO7 | -0.100 | RO8 | -0.112 | RO9 | -0.100 | RO10 | -0.100 |
| RO11 | -0.100 | RO12 | -0.100 | RO13 | -0.100 | RO14 | -0.100 | RO15 | -0.100 |

MULTIPLICITIES

<table>
<thead>
<tr>
<th>MULT</th>
<th>1</th>
<th>MULT</th>
<th>1</th>
<th>MULT</th>
<th>1</th>
<th>MULT</th>
<th>1</th>
<th>MULT</th>
<th>1</th>
<th>MULT</th>
<th>1</th>
<th>MULT</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>1</td>
<td>M2</td>
<td>1</td>
<td>M3</td>
<td>1</td>
<td>M4</td>
<td>1</td>
<td>M5</td>
<td>1</td>
<td>M6</td>
<td>1</td>
<td>M7</td>
<td>1</td>
</tr>
</tbody>
</table>

INITIAL APPROXIMATIONS

| APPROX | 0.1294095294381290 | APPROX | 0.1290562680435400 | APPROX | 0.1294095294381290 |
|--------|----------------------|--------|----------------------|--------|
| PI(X)  | 0.3981987511785640  | PI(X)  | 0.3981987511785640  | PI(X)  | 0.3981987511785640 |

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>( \text{ROOT 1} )</td>
<td>-0.33763667932440 -16</td>
<td>0.0209579264431870 00 + 0.12949264431870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 2} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 3} )</td>
<td>-0.33763667932440 -16</td>
<td>0.0209579264431870 00 + 0.12949264431870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 4} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 5} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 6} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 7} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 8} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 9} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 10} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 11} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 12} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 13} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 14} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
<tr>
<td>( \text{ROOT 15} )</td>
<td>-0.3000000000000000 00</td>
<td>0.0341273052684870 00 + 0.7087068063184870 00 i</td>
</tr>
</tbody>
</table>

SOLVED BY DIRECT METHOD

Exhibit 6.2. Roots Are: -1 - i, 1 + i, -2 - 3i, 2 - i, 3, 2, i, -i, -10/3, .3, -1, 1, 4 + 4i, -.2 + .2i, .2 - .2i.
The coefficients of $P(x)$ are:

$$
egin{align*}
\text{P}(1) &= 0.1000000000000000 \times 10^{-1} + 0.0000000000000000 \times 10^{-1} \\
\text{P}(2) &= -0.7000000000000001 \times 10^{-1} + 0.1050000000000000 \times 10^{-1} \\
\text{P}(3) &= -0.2200000000000000 \times 10^{-2} + 0.5800000000000001 \times 10^{-2} \\
\text{P}(4) &= 0.1710000000000000 \times 10^{-3} + 0.1500000000000000 \times 10^{-1} \\
\text{P}(5) &= -0.7300000000000000 \times 10^{-2} + 0.2510000000000000 \times 10^{-2} \\
\text{P}(6) &= -0.2280000000000000 \times 10^{-3} + 0.1040000000000000 \times 10^{-1} \\
\text{P}(7) &= 0.7200000000000001 \times 10^{-2} + 0.1040000000000000 \times 10^{-1}
\end{align*}
$$

Number of initial approximations given: 0
Maximum number of iterations: 200
Test for convergence: $1 \times 10^{-9}$
Test for multiplicities: $1 \times 10^{-1}$
Radius to start search: 0.00
Radius to end search: 0.00

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>1</td>
<td>2</td>
<td>0.99999998831250190 00 + 0.200000052138240 01 i</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.199999873810254640 01 + 0.1905256231445640 01 i</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.490223197546240 00 + 0.5142344322938120 01 i</td>
</tr>
</tbody>
</table>

In the attempt to improve accuracy, Root 2 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>1</td>
<td>2</td>
<td>0.99999998831250190 00 + 0.200000052138240 01 i</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.199999873810254640 01 + 0.1905256231445640 01 i</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.490223197546240 00 + 0.5142344322938120 01 i</td>
</tr>
</tbody>
</table>

Exhibit 6.3. Roots Are: $2 \pm 2i$ (3), $1 \pm 2i$ (2), $-1 \pm 5i$
### Newtons Method to Find Zeros of Polynomials

**Polynomial Number 2 of Degree 15**

The coefficients of p(x) are

\[
\begin{align*}
p(1) &= 0.4800000000000000D02 + 0.0000000000000000D00 \quad \text{01} \\
p(2) &= -0.6531200000000000D03 - 0.3840000000000000D00 \quad \text{02} \\
p(3) &= -0.7305960000000000D02 + 0.2189460000000000D00 \quad \text{03} \\
p(4) &= -0.1355656600000000D04 + 0.6462815600000000D00 \quad \text{04} \\
p(5) &= -0.1793344000000000D03 + 0.1620528972000000D00 \quad \text{05} \\
p(6) &= -0.4478927000000000D01 \quad \text{06} \\
p(7) &= -0.1623949281300000D01 + 0.4036662732000000D00 \quad \text{07} \\
p(8) &= -0.1642742200560000D01 + 0.4137366230400000D00 \quad \text{08} \\
p(9) &= -0.1853899200000000D00 + 0.1827453020000000D00 \quad \text{09} \\
p(10) &= -0.1871255780010000D00 + 0.1928465440330000D00 \quad \text{10} \\
p(11) &= -0.1274997298590000D00 + 0.2171341227420000D00 \quad \text{11} \\
p(12) &= -0.2814692716800000D00 + 0.1928465440330000D00 \quad \text{12} \\
p(13) &= 0.1329434434800000D05 + 0.1038130226550000D00 \quad \text{13} \\
p(14) &= 0.3053900774700000D05 + 0.2998951413500000D00 \quad \text{14} \\
p(15) &= -0.1835899020000000D03 - 0.1827453020000000D00 \quad \text{15} \\
p(16) &= 0.2755620000000000D00 + 0.2755620000000000D00 \quad \text{16}
\end{align*}
\]

**Number of Initial Approximations Given.**

- **Maximum Number of Iterations.**
- **Test for convergence.**
- **Test for multiplicities.**
- **Radius to start search.**
- **Radius to end search.**

Test for convergence: 0.100-09
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

<table>
<thead>
<tr>
<th>ROOT OF P(X)</th>
<th>MULTIPlicITIES</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>2</td>
<td>0.4829069115656270D00 + 0.12994928431810D00 1</td>
</tr>
<tr>
<td>Root 2</td>
<td>2</td>
<td>0.7031067550442630D00 + 0.0703106754832910D00 1</td>
</tr>
<tr>
<td>Root 3</td>
<td>2</td>
<td>0.7031067550442630D00 + 0.0703106754832910D00 1</td>
</tr>
<tr>
<td>Root 4</td>
<td>1</td>
<td>0.12994928431810D00 1</td>
</tr>
<tr>
<td>Root 5</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 6</td>
<td>1</td>
<td>0.3307462843222300D00 1</td>
</tr>
<tr>
<td>Root 7</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 8</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 9</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 10</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 11</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 12</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 13</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 14</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 15</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
<tr>
<td>Root 16</td>
<td>1</td>
<td>0.2171341227420000D00 1</td>
</tr>
</tbody>
</table>

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

---

Exhibit 6.4.
**Exhibit 6.4. Roots Are:**  
-2.33, .003 (2), i (2),  
1.5i (2), -1.5i (2) 3i (3), -1-i (3)
NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS POLYNOMIAL NUMBER 3 OF DEGREE 8

THE COEFFICIENTS OF P(x) ARE

<table>
<thead>
<tr>
<th>Coefficient (x^2)</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(1)</td>
<td>0.10000000000000000000</td>
</tr>
<tr>
<td>P(2)</td>
<td>0.00000000000000000000</td>
</tr>
<tr>
<td>P(3)</td>
<td>0.15750000000000000000</td>
</tr>
<tr>
<td>P(4)</td>
<td>0.51750000000000000000</td>
</tr>
<tr>
<td>P(5)</td>
<td>0.15725000000000000000</td>
</tr>
<tr>
<td>P(6)</td>
<td>0.30750000000000000000</td>
</tr>
<tr>
<td>P(7)</td>
<td>0.49525000000000000000</td>
</tr>
<tr>
<td>P(8)</td>
<td>0.18100000000000000000</td>
</tr>
</tbody>
</table>

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.100-05
TEST FOR MULTIPlicITIES: 0.100-01
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

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

ROOTS OF P(x) | MULTIPlicITIES | INITIAL APPROXIMATION
-------------|----------------|------------------
ROOT(1)      | 2              | 0.48294927272727272727 + 0.20000000000000000000i |
ROOT(2)      | 1              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(3)      | 1              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(4)      | 1              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(5)      | 1              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(6)      | 1              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(7)      | 1              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(8)      | 1              | 0.19999999999999999999 + 0.20000000000000000000i |

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

ROOTS OF P(x) | MULTIPlicITIES | INITIAL APPROXIMATION
-------------|----------------|------------------
ROOT(1)      | 2              | 0.48294927272727272727 + 0.20000000000000000000i |
ROOT(2)      | 2              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(3)      | 2              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(4)      | 2              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(5)      | 2              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(6)      | 2              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(7)      | 2              | 0.19999999999999999999 + 0.20000000000000000000i |
ROOT(8)      | 2              | 0.19999999999999999999 + 0.20000000000000000000i |

Exhibit 6.5. Roots Are: 2+2i (3), 1+2i (2), -1+5i (3)
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 6 OF DEGREE 12

THE COEFFICIENTS OF PI(X) ARE

\[ P(0) = 0.1000000000000000D + 0.0000000000000000D \]
\[ P(1) = -0.1200000000000000D + 0.0000000000000000D \]
\[ P(2) = 0.0240000000000000D + 0.0000000000000000D \]
\[ P(3) = -0.2800000000000000D + 0.0000000000000000D \]
\[ P(4) = 0.7800000000000000D + 0.0000000000000000D \]
\[ P(5) = 0.7240000000000000D + 0.0000000000000000D \]
\[ P(6) = -0.1632000000000000D + 0.0000000000000000D \]
\[ P(7) = 0.2624000000000000D + 0.0000000000000000D \]
\[ P(8) = -0.3264000000000000D + 0.0000000000000000D \]
\[ P(9) = 0.3120000000000000D + 0.0000000000000000D \]
\[ P(10) = -0.2240000000000000D + 0.0000000000000000D \]
\[ P(11) = 0.1152000000000000D + 0.0000000000000000D \]
\[ P(12) = -0.3840000000000000D + 0.0000000000000000D \]

NUMBER OF INITIAL APPROXIMATIONS GIVEN, 0
MAXIMUM NUMBER OF ITERATIONS, 200
TEST FOR CONVERGENCE, 0.000-03
TEST FOR MULTIPLE ROOTS, 0.000-01
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

ROOT(1) = 0.9994079803369152D + 0.9951391489643941D
5 0.4829629115656279D + 0.1294095284438187D

ROOT(2) = 0.1020096885609720D + 0.5420646130387889D
1 0.7071067553046346D + 0.7071068070684595D

ROOT(3) = 0.1002721106681658D + 0.1023310661526634D
1 0.3882284792654056D + 0.1448888763117193D

ROOT(4) = 0.6189968959361400D + 0.7949702013736144D
1 -0.5176382551966724D + 0.1931851608368755D

ROOT(5) = 0.9851422047423414D + 0.1395597073813038D
1 -0.1767767455798852D + 0.1767767588885201D

ROOT(6) = 0.6515828215165421D + 0.1206987666454936D
1 -0.2897777583074990D + 0.7764567463987070D

ROOT(7) = 0.1399962790330915D + 0.4829629115656279D
1 0.4829629115656279D + 0.1294095284438187D

ROOT(8) = 0.1330620199428514D + 0.1206987666454936D
1 0.1206987666454936D + 0.7764567463987070D

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(7) = 0.1020096885609720D + 0.5420646130387889D
I DID NOT CONVERGE.

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(8) = 0.1330620199428514D + 0.1206987666454936D
I DID NOT CONVERGE.

IN THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF PI(X) ARE

Exhibit 6.6.
### Roots of Pxi

<table>
<thead>
<tr>
<th>ROOT</th>
<th>MULTICIPITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.999481137951166760</td>
<td>0.999404929244248187D 00</td>
</tr>
<tr>
<td>2</td>
<td>0.995253513151166760</td>
<td>0.99525655466464640D 00</td>
</tr>
<tr>
<td>3</td>
<td>0.999657676718664500</td>
<td>0.99965777655555550D 01</td>
</tr>
<tr>
<td>4</td>
<td>0.999657676718664500</td>
<td>0.99965777655555550D 01</td>
</tr>
<tr>
<td>5</td>
<td>0.999657676718664500</td>
<td>0.99965777655555550D 01</td>
</tr>
<tr>
<td>6</td>
<td>0.999657676718664500</td>
<td>0.99965777655555550D 01</td>
</tr>
<tr>
<td>7</td>
<td>0.999657676718664500</td>
<td>0.99965777655555550D 01</td>
</tr>
<tr>
<td>8</td>
<td>0.999657676718664500</td>
<td>0.99965777655555550D 01</td>
</tr>
</tbody>
</table>

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

\[ P_1 = 0.1000000000000000 \]
\[ P_2 = -0.7000000000000001 \]
\[ P_3 = -0.2850000000000000 \]
\[ P_4 = 0.1710000000000000 \]
\[ P_5 = -0.7500000000000000 \]
\[ P_6 = -0.2250000000000000 \]
\[ P_7 = 0.1200000000000000 \]

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

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)

<table>
<thead>
<tr>
<th>ROOT</th>
<th>MULTIPLEITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>3</td>
<td>0.482962911565627900D + 0.129409528443818700D</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>1</td>
<td>0.707106755046346000D + 0.707106755046346000D</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>1</td>
<td>SOLVED BY DIRECT METHOD</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>1</td>
<td>SOLVED BY DIRECT METHOD</td>
</tr>
</tbody>
</table>

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)

<table>
<thead>
<tr>
<th>ROOT</th>
<th>MULTIPLEITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>3</td>
<td>0.482962911565627900D + 0.129409528443818700D</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>1</td>
<td>0.707106755046346000D + 0.707106755046346000D</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>1</td>
<td>SOLVED BY DIRECT METHOD</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>1</td>
<td>SOLVED BY DIRECT METHOD</td>
</tr>
</tbody>
</table>

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>Degree</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.4800000000000000D02</td>
</tr>
<tr>
<td>1</td>
<td>0.2557120000000000D03</td>
</tr>
<tr>
<td>2</td>
<td>-0.3255569800000000D04</td>
</tr>
<tr>
<td>3</td>
<td>-0.3855656596000000D04</td>
</tr>
<tr>
<td>4</td>
<td>-0.4656790270000000D04</td>
</tr>
<tr>
<td>5</td>
<td>-0.1022294522130000D06</td>
</tr>
<tr>
<td>6</td>
<td>-0.2036625888420000D06</td>
</tr>
<tr>
<td>7</td>
<td>-0.1871255780010000D06</td>
</tr>
<tr>
<td>8</td>
<td>-0.1239434434800000D05</td>
</tr>
<tr>
<td>9</td>
<td>-0.3053900774700000D05</td>
</tr>
<tr>
<td>10</td>
<td>-0.1838999200000000D05</td>
</tr>
<tr>
<td>11</td>
<td>0.2755620000000000D00</td>
</tr>
<tr>
<td>12</td>
<td>0.2755620000000000D00</td>
</tr>
</tbody>
</table>

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0

MAXIMUM NUMBER OF ITERATIONS: 200

TEST FOR CONVERGENCE: 0.100-09

TEST FOR MULTIPlicITIES: 0.10D-01

RADIUS TO START SEARCH: 0.000 00

RADIUS TO END SEARCH: 0.00D 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF $P(X)$

<table>
<thead>
<tr>
<th>Root</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.30000000056259737020-02</td>
</tr>
<tr>
<td>1</td>
<td>0.8277039321155559D-12</td>
</tr>
<tr>
<td>2</td>
<td>0.7071067550488488D-04</td>
</tr>
<tr>
<td>3</td>
<td>0.7071067550488488D-04</td>
</tr>
<tr>
<td>4</td>
<td>0.9999999999999999D00</td>
</tr>
<tr>
<td>5</td>
<td>0.9999999999999999D00</td>
</tr>
<tr>
<td>6</td>
<td>0.9999999999999999D00</td>
</tr>
<tr>
<td>7</td>
<td>0.9999999999999999D00</td>
</tr>
<tr>
<td>8</td>
<td>0.9999999999999999D00</td>
</tr>
<tr>
<td>9</td>
<td>0.9999999999999999D00</td>
</tr>
<tr>
<td>10</td>
<td>0.9999999999999999D00</td>
</tr>
<tr>
<td>11</td>
<td>0.9999999999999999D00</td>
</tr>
<tr>
<td>12</td>
<td>0.9999999999999999D00</td>
</tr>
</tbody>
</table>

AFTER THE ATTEMPT TO IMPROVE ACCURACY

<table>
<thead>
<tr>
<th>Root</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.4800000000000000D02</td>
</tr>
<tr>
<td>1</td>
<td>0.2557120000000000D03</td>
</tr>
<tr>
<td>2</td>
<td>-0.3255569800000000D04</td>
</tr>
<tr>
<td>3</td>
<td>-0.3855656596000000D04</td>
</tr>
<tr>
<td>4</td>
<td>-0.4656790270000000D04</td>
</tr>
<tr>
<td>5</td>
<td>-0.1022294522130000D06</td>
</tr>
<tr>
<td>6</td>
<td>-0.2036625888420000D06</td>
</tr>
<tr>
<td>7</td>
<td>-0.1871255780010000D06</td>
</tr>
<tr>
<td>8</td>
<td>-0.1239434434800000D05</td>
</tr>
<tr>
<td>9</td>
<td>-0.3053900774700000D05</td>
</tr>
<tr>
<td>10</td>
<td>-0.1838999200000000D05</td>
</tr>
<tr>
<td>11</td>
<td>0.2755620000000000D00</td>
</tr>
<tr>
<td>12</td>
<td>0.2755620000000000D00</td>
</tr>
</tbody>
</table>

SOLVED BY DIRECT METHOD

Exhibit 6.8.
<table>
<thead>
<tr>
<th>ROOTS</th>
<th>MULTIPLECTIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>2 0.4820629115656270 00 + 0.1294092438438187 00 1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1 0.7071067811869566 00 + 0.9999991280047800 00 1</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1 0.3882286472650560 00 + 0.3889088292979509 00 1</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1 0.5176235506877200 00 + 0.1931851608368750 00 1</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1 0.1077676170007010 00 + 0.3889086630047250 00 1</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>1 0.3279611564291800 00 + 0.7763476639570700 00 1</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>1 0.9999998125742390 00 + 0.1931851608368750 00 1</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>1 0.2385422244363600 00 + 0.3889088292979509 00 1</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>1 0.9999998125742390 00 + 0.1931851608368750 00 1</td>
</tr>
</tbody>
</table>

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

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

\[
\begin{align*}
    p(1) &= 0.1000000000000006 \\ 
    p(2) &= -0.5175000000000010 \\ 
    p(3) &= 0.3075000000000003 \\ 
    p(4) &= -0.4952500000000001 \\ 
    p(5) &= -0.5857500000000001 \\ 
    p(6) &= -0.4952500000000001 \\ 
    p(7) &= -0.5857500000000001 \\ 
    p(8) &= 0.1810000000000000 \\ 
    p(9) &= 0.1580000000000000 \\
\end{align*}
\]

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: 0.000D00

Radius to End Search: 0.000D00

Before Attempt to Improve Accuracy

Roots of \( p(x) \)

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>3</td>
<td>0.4829629156562790</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>3</td>
<td>0.7071067539663460</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>3</td>
<td>0.5176285196872400</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>3</td>
<td>0.4952500000000001</td>
</tr>
<tr>
<td>( R_5 )</td>
<td>3</td>
<td>0.2000042175889590</td>
</tr>
<tr>
<td>( R_6 )</td>
<td>3</td>
<td>0.1000000023165255</td>
</tr>
<tr>
<td>( R_7 )</td>
<td>3</td>
<td>-0.1063528418749844</td>
</tr>
<tr>
<td>( R_8 )</td>
<td>3</td>
<td>-0.9722122518699260</td>
</tr>
<tr>
<td>( R_9 )</td>
<td>3</td>
<td>0.9999370860929820</td>
</tr>
</tbody>
</table>

After the Attempt to Improve Accuracy

Roots of \( p(x) \)

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>3</td>
<td>0.4829629156562790</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>3</td>
<td>0.7071067539663460</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>3</td>
<td>0.5176285196872400</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>3</td>
<td>0.4952500000000001</td>
</tr>
<tr>
<td>( R_5 )</td>
<td>3</td>
<td>0.2000042175889590</td>
</tr>
<tr>
<td>( R_6 )</td>
<td>3</td>
<td>0.1000000023165255</td>
</tr>
<tr>
<td>( R_7 )</td>
<td>3</td>
<td>-0.1063528418749844</td>
</tr>
<tr>
<td>( R_8 )</td>
<td>3</td>
<td>-0.9722122518699260</td>
</tr>
<tr>
<td>( R_9 )</td>
<td>3</td>
<td>0.9999370860929820</td>
</tr>
</tbody>
</table>

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

\[
\begin{align*}
p(1) & = 0.1000000000000000D+01 \\
p(2) & = -0.1200000000000000D+02 \\
p(3) & = 0.3200000000000000D+02 \\
p(4) & = -0.2600000000000000D+03 \\
p(5) & = 0.7800000000000000D+03 \\
p(6) & = -0.1620000000000000D+03 \\
p(7) & = 0.2624000000000000D+04 \\
p(8) & = -0.3264000000000000D+04 \\
p(9) & = 0.3120000000000000D+04 \\
p(10) & = -0.2240000000000000D+04 \\
p(11) & = 0.1152000000000000D+04 \\
p(12) & = -0.3840000000000000D+03 \\
p(13) & = 0.6400000000000001D+02
\end{align*}
\]

Exhibit 6.10.
Exhibit 6.10. Roots Are: $1+i$ (6), $1-i$ (6)
GREATEST COMMON DIVISOR METHOD USED WITH NEWTON'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 GCD: 0.100D-02
TEST FOR CONVERGENCE: 0.100D-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.100D-19
TEST FOR MULTIPlicITIES: 0.100D-01
RADIUS TO START SEARCH: 0.000D 00
RADIUS TO END SEARCH: 0.000D 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 + 0.1050000000000000D 02 I
P(5) = -0.2800000000000000D 02 + 0.5800000000000000D 03 I
P(4) = -0.1710000000000000D 03 + 0.1500000000000000D 03 I
P(3) = -0.7300000000000000D 03 + 0.2510000000000000D 03 I
P(2) = -0.2280000000000000D 03 + 0.1040000000000000D 03 I
P(1) = 0.7200000000000000D 02 + 0.1040000000000000D 03 I

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

THE DEGREE OF Q(X) IS 3 THE COEFFICIENTS ARE

Q(4) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
Q(3) = -0.2000000000000000D 01 + 0.4500000000000000D 01 I
Q(2) = -0.7000000000000000D 03 + 0.3500000000000000D 01 I
Q(1) = 0.9999999999997762D 00 + 0.6999999999981200D 01 I

ROOTS OF Q(X)

ROOT(1) = 0.9999999999997565D 00 + 0.1999999999995740D 01 I
ROOT(2) = 0.2000000000000529D 01 + 0.20000000000000529D 01 I
ROOT(3) = -0.9999999999993960D 00 + 0.5000000000000149D 00 I

INITIAL APPROXIMATION

0.4829429151565279D 00 + 0.1294095284438187D 00 I

RESULTS OF SUBROUTINE QUAD

ROOTS OF P(X)

ROOT(1) = 0.9999999999997565D 00 + 0.1999999999995740D 01 I
ROOT(2) = 0.2000000000000529D 01 + 0.20000000000000529D 01 I
ROOT(3) = -0.9999999999993960D 00 + 0.5000000000000149D 00 I

MULTIPlicITIES

2 3 1

INITIAL APPROXIMATION

0.4829429151565279D 00 + 0.1294095284438187D 00 I

RESULTS OF SUBROUTINE QUAD

Exhibit 6.11. Roots Are: 2+2i (3), 1+2i (2), -1+5i
GREATEST COMMON DIVISOR METHOD USED WITH NEWTON'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. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE quad. 0.100-19
TEST FOR MULTIPICITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

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

P10 ) = 0.4600000000000000 02 + 0.0000000000000000 D 00 00 1
P11 ) = 0.2591720000000000 03 - 0.3840000000000000 D 03 03 1
P12 ) = -0.1153956800000000 D 03 + 0.2189640000000000 D 04 1
P13 ) = -0.3855956960000000 D 04 - 0.8946319000000000 D 05 05 1
P14 ) = -0.1379366668000000 D 05 - 0.14305209726000000 D 06 06 1
P15 ) = -0.4467902000000000 D 06 - 0.1765957484000000 D 07 07 1
P16 ) = -0.10123955223100000 07 - 0.4030642320000000 D 08 08 1
P17 ) = -0.11442722224600000 08 + 0.4378623200000000 D 09 09 1
P18 ) = -0.20362258820000000 09 + 0.10938492726000000 06 06 1
P19 ) = -0.10123955223100000 09 + 0.1972964943000000 D 06 06 1
P20 ) = -0.12749720659000000 09 + 0.21713412274000000 D 06 06 1
P21 ) = -0.26143927160000000 09 - 0.1936497296000000 D 06 06 1
P22 ) = 0.13296343460000000 10 - 0.1593610225000000 D 06 06 1
P23 ) = 0.30590074102000000 10 - 0.2969969113000000 D 06 06 1
P24 ) = -0.18358999020000000 10 - 0.1827321460000000 D 06 06 1
P11 ) = 0.2795570000000000 D 00 + 0.2759620000000000 D 00 01

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

Q1 ) = 0.4600000000000000 02 + 0.0000000000000000 D 00 00 1
Q2 ) = 0.196659099999630 03 - 0.1442600000000000 D 03 03 1
Q3 ) = 0.26701999989682470 03 - 0.5257580000628390 D 04 04 1
Q4 ) = -0.327199999721904 04 - 0.914160000075920 D 05 05 1
Q5 ) = 0.232122494645530 05 + 0.1582252000003768 D 06 06 1
Q6 ) = -0.72020000005918400 06 - 0.1327497000996440 D 07 07 1
Q7 ) = -0.7557640000541930 06 - 0.5252004000333710 D 07 07 1
Q8 ) = 0.26260001335502310 01 + 0.2287499058028130 D 01 01

ROOTS OF Q(x)

ROOT 11 = 0.31000000306559970 -0.13705421768384330 -0.13705421768384330 D 00 00 00 00
ROOT 12 = 0.7071067530463590 D 02 + 0.7071067530463590 D 02 00 00
ROOT 13 = 0.1466335432590950 D 02 - 0.1466335432590950 D 02 00 00
ROOT 14 = 0.1466335432590950 D 02 - 0.1466335432590950 D 02 00 00
ROOT 15 = -0.118767219459390 -1.0506000000000607000 D 01 01

INITIAL APPROXIMATION

D 0.48296291156562790 00 + 0.12940652844381670 00 00 00
D 0.7071067530463590 D 02 + 0.7071067530463590 D 02 00 00
D 0.1466335432590950 D 02 - 0.1466335432590950 D 02 00 00
D 0.1466335432590950 D 02 - 0.1466335432590950 D 02 00 00
D 0.7071067530463590 D 02 + 0.7071067530463590 D 02 00 00
D 0.1466335432590950 D 02 - 0.1466335432590950 D 02 00 00

End Search.

POLYNOMIAL GREATEST

OF PtX1 IS

EXHIBIT 6.12
ROOT( 5) = -0.233333333340850 00 + 0.4291551250697099D-11 01 1
ROOT( 6) = 0.114045217715650-09 + 0.300000000059440 00 1
ROOT( 7) = -0.100000000322260 01 + 0.99999999857815 00 1

Exhibit 6.12. Roots Are: -2.33, .003 (2), i(2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)
GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 3

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.10D-02
TEST FOR CONVERGENCE: 0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.10D-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(0) = 0.10000000000000000 00 0.00000000000000000 00 I
P(1) = -0.51750000000000000 00 0.43000000000000000 00 I
P(2) = -0.51750000000000000 00 0.00000000000000000 00 I
P(3) = -0.51750000000000000 00 0.00000000000000000 00 I
P(4) = -0.51750000000000000 00 0.00000000000000000 00 I
P(5) = -0.51750000000000000 00 0.00000000000000000 00 I
P(6) = -0.51750000000000000 00 0.00000000000000000 00 I
P(7) = -0.51750000000000000 00 0.00000000000000000 00 I

THE DEGREE OF Q(X) IS 3 THE COEFFICIENTS ARE

Q(0) = 0.10000000000000000 00 0.00000000000000000 00 I
Q(1) = -0.70000000000000000 00 0.35000000000000000 00 I
Q(2) = -0.70000000000000000 00 0.35000000000000000 00 I
Q(3) = -0.70000000000000000 00 0.35000000000000000 00 I

ROOTS OF P(X)

ROOT(1) = 0.9999999999999983D 00 + 0.482942911566279D 00 I
ROOT(2) = 0.2000000000000273D 01 + 0.2999999999999985D 00 I
ROOT(3) = -0.9999999999999983D 00 + 0.482942911566279D 00 I

ROOTS OF Q(X)

ROOT(1) = 0.9999999999999983D 00 + 0.482942911566279D 00 I
ROOT(2) = 0.2000000000000273D 01 + 0.2999999999999985D 00 I
ROOT(3) = -0.9999999999999983D 00 + 0.482942911566279D 00 I

RESULTS OF SUBROUTINE QUAD

ROOTS: 2+2i (3), 1+2i (2), -1+.5i (3)

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
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 12 THE COEFFICIENTS ARE

\[
P(13) = 0.1000000000000000D 01 + 0.0000000000000000D 00 i
P(12) = -0.1200000000000000D 02 + 0.0000000000000000D 00 i
P(11) = 0.7200000000000000D 02 + 0.0000000000000000D 00 i
P(10) = -0.2800000000000000D 03 + 0.0000000000000000D 00 i
P(9) = 0.7800000000000000D 03 + 0.0000000000000000D 00 i
P(8) = -0.1637000000000000D 04 + 0.0000000000000000D 00 i
P(7) = 0.2864000000000000D 03 + 0.0000000000000000D 00 i
P(6) = -0.3244000000000000D 04 + 0.0000000000000000D 00 i
P(5) = 0.5212000000000000D 04 + 0.0000000000000000D 00 i
P(4) = -0.2242000000000000D 04 + 0.0000000000000000D 00 i
P(3) = 0.1152000000000000D 04 + 0.0000000000000000D 00 i
P(2) = -0.3360000000000000D 03 + 0.0000000000000000D 00 i
P(1) = 0.6400000000000000D 02 + 0.0000000000000000D 00 i
P(0) = 0.1000000000000000D 01 + 0.0000000000000000D 00 i

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

\[
Q(1) = 0.1000000000000000D 01 + 0.0000000000000000D 00 i
Q(0) = -0.2000000000000000D 01 + 0.0000000000000000D 00 i
Q(-1) = 0.1999999999999999D 01 + 0.0000000000000000D 00 i

ROOTS OF P(X) MULTIPLECTIES

\[
\text{ROOTI: 11} = 0.1000000000000000D 07 + 0.9999999999999740D 00 i \quad 6 \quad \text{RESULTS OF SUBROUTINE QUAD}
\text{ROOTI: 21} = 0.1000000000000000D 07 + -0.9999999999999740D 00 i \quad 6 \quad \text{RESULTS OF SUBROUTINE QUAD}
\]

Exhibit 6.14. Roots Are: 1+i (6), 1-i (6)
GREATEST COMMON DIVISION METHOD) USED WITH MULTIVERS 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: 0.10 - 02
TEST FOR CONVERGENCE: 0.100 - 09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.10 - 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 6, THE COEFFICIENTS ARE

P(7) = 0.1000000000000000D 01 + 0.0000000000000000D 00 01
P(6) = -0.7000000000000000D 01 + 0.1050000000000000D 02 02
P(5) = -0.2800000000000000D 02 + 0.5800000000000000D 03 03
P(4) = 0.1710000000000000D 03 + 0.1500000000000000D 04 04
P(3) = -0.7300000000000000D 03 + 0.2510000000000000D 04 04
P(2) = -0.2280000000000000D 03 + 0.1040000000000000D 04 04
P(1) = 0.7200000000000000D 02 + 0.1040000000000000D 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(4) = 0.1000000000000000D 01 + 0.0000000000000000D 00 01
Q(3) = -0.2000000000000000D 01 + 0.4500000000000000D 02 02
Q(2) = -0.7000000000000000D 01 + 0.3500000000000000D 02 02
Q(1) = 0.99999999997620 00 + 0.6999999998120 01 01

ROOTS OF P(X)
ROOT 1: 0.99999999999975650 00 + 0.199999999999740 01 01
ROOT 2: -0.200000000000070 01 + 0.2000000000000000D 02 02
ROOT 3: -0.9999999999998360 00 + 0.500000000000140 00 00

ROOTS OF Q(X)
ROOT 1: 0.99999999999997650 00 + 0.1999999999999740 01 01
ROOT 2: -0.2000000000000370 01 + 0.2000000000000000D 02 02
ROOT 3: -0.9999999999998560 00 + 0.500000000000140 00 00

MULTIPLEITIES
INITIAL APPROXIMATION
ROOT 1: 0.99999999999997650 00 + 0.1999999999999740 01 01
ROOT 2: -0.2000000000000370 01 + 0.2000000000000000D 02 02
ROOT 3: -0.9999999999998560 00 + 0.500000000000140 00 00

RESULTS OF SUBROUTINE QUAD
SOLVED BY DIRECT METHOD

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-19
TEST FOR MULTIPlicITIES. 0.10D-01
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

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

<table>
<thead>
<tr>
<th>P(16)</th>
<th>0.480000000000000000E+02</th>
<th>0.000000000000000000E+00</th>
<th>0.00E+00</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(15)</td>
<td>0.356771260000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(14)</td>
<td>-0.218946900000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(13)</td>
<td>-0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(12)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(11)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(10)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(9)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(8)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(7)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(6)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(5)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(4)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(3)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(2)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>P(1)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Q(8)</th>
<th>0.480000000000000000E+02</th>
<th>0.000000000000000000E+00</th>
<th>0.00E+00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q(7)</td>
<td>0.356771260000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>Q(6)</td>
<td>-0.218946900000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>Q(5)</td>
<td>-0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>Q(4)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>Q(3)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>Q(2)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>Q(1)</td>
<td>0.437959000000000000E+00</td>
<td>0.000000000000000000E+00</td>
<td>0.00E+00</td>
</tr>
</tbody>
</table>

ROOTS OF Q(X)

<table>
<thead>
<tr>
<th>ROOT(11)</th>
<th>0.300000000000000000E+01</th>
<th>0.137354317716148000E+00</th>
<th>0.124905248421870000E+00</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(12)</td>
<td>0.144463724917301000E+01</td>
<td>0.120000000000000000E+00</td>
<td>0.170126530763400000E+00</td>
</tr>
<tr>
<td>ROOT(13)</td>
<td>0.126210310000000000E+01</td>
<td>0.110999999999999900E+00</td>
<td>0.136842867269400000E+00</td>
</tr>
<tr>
<td>ROOT(14)</td>
<td>0.114045495976963200E+00</td>
<td>0.100000000000000000E+00</td>
<td>0.170582591966972000E+00</td>
</tr>
</tbody>
</table>

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

\[ P(9) = 0.1000000000000000D 01 + 0.0000000000000000 00 \]
\[ P(8) = -0.5000000000000000D 01 \]
\[ P(7) = -0.5175000000000000D 02 \]
\[ P(6) = 0.1572500000000000D 02 \]
\[ P(5) = 0.1075000000000000D 02 \]
\[ P(4) = -0.5957500000000000D 02 \]
\[ P(3) = 0.1810000000000000D 02 \]
\[ P(2) = 0.3075000000000000D 03 \]
\[ P(1) = -0.4952500000000000D 03 \]
\[ P(0) = 0.1572500000000000D 03 \]

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 MULTICILITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE DEGREE OF Q(X) IS 3 THE COEFFICIENTS ARE

\[ Q(4) = 0.1000000000000000D 01 + 0.0000000000000000 00 \]
\[ Q(3) = -0.2000000000000001D 01 \]
\[ Q(2) = -0.7000000000000000D 01 \]
\[ Q(1) = 0.9999999999999999D 00 \]
\[ Q(0) = 0.9999999999999999D 00 \]

ROOTS OF P(X)

\[ \text{ROOT}(1) = 0.1000000000000000D 01 + 0.0000000000000000 00 \]
\[ \text{ROOT}(2) = -0.4957500000000000D 02 \]
\[ \text{ROOT}(3) = 0.1572500000000000D 02 \]

INITIAL APPROXIMATION
0.4929291588562790 00 + 0.12940952844381870 00
SOLVED BY DIRECT METHOD

ROOTS OF Q(X)

\[ \text{ROOT}(1) = 0.1000000000000000D 01 + 0.0000000000000000 00 \]
\[ \text{ROOT}(2) = -0.2000000000000000D 01 \]
\[ \text{ROOT}(3) = 0.9999999999999999D 00 \]

INITIAL APPROXIMATION
0.4929291588562790 00 + 0.12940952844381870 00
SOLVED BY DIRECT METHOD

RESULTS OF SUBROUTINE QUAD

Exhibit 6.17. Roots Are: 2+2i (3), 1+2i (2), -1.5i (3)
GREATEST COMMON DIVISOR METHOD USED WITH MULLER'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. 0.100-19
TEST FOR MULTIPlicITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

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

P(13) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
P(12) = -0.1200000000000000D 02 + -0.0000000000000000D 00 I
P(11) = 0.7200000000000001D 02 + 0.0000000000000000D 00 I
P(10) = -0.2800000000000000D 03 + -0.0000000000000000D 00 I
P(9) = 0.7800000000000001D 02 + 0.0000000000000000D 00 I
P(8) = -0.16320000000000000 04 + 0.0000000000000000D 00 I
P(7) = 0.26240000000000000 04 + 0.0000000000000000D 00 I
P(6) = -0.32640000000000000 04 + 0.0000000000000000D 00 I
P(5) = 0.3120000000000000D 04 + 0.0000000000000000D 00 I
P(4) = -0.22400000000000000 04 + 0.0000000000000000D 00 I
P(3) = 0.11520000000000000 04 + 0.0000000000000000D 00 I
P(2) = -0.3840000000000000D 03 + 0.0000000000000000D 00 I
P(1) = 0.64000000000000010 02 + 0.0000000000000000D 00 I
P(0) = 0.0000000000000000D 00 I

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

Q(3) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
Q(2) = -0.2000000000000000D 02 + 0.0000000000000000D 00 I
Q(1) = 0.1999999999999830D 01 + 0.0000000000000000D 00 I
Q(0) = 0.0000000000000000D 00 I

ROOTS OF P(X)

ROOT 11 = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
ROOT 21 = 0.9999999999999970D 01 + 0.0000000000000000D 00 I

ROOT 11 = 1+i(6), ROOT 21 = 1-i(6)

EXHIBIT 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 Z0Z: 0.100-02
Test for zero in subroutine ZQZ: 0.300-09
Radius to start search: 0.000 00
Radius to end search: 0.000 00

The degree of P(x) is 6 and the coefficients are:

P(1) = -0.1000000000000000 01 + 0.0000000000000000 00 01
P(2) = -0.7000000000000000 01 + 0.1050000000000000 02 01
P(3) = -0.2890000000000000 02 + 0.5800000000000000 02 02
P(4) = 0.1710000000000000 02 + 0.1350000000000000 03 01
P(5) = -0.7390000000000000 02 + 0.2430000000000000 03 01
P(6) = -0.2280000000000000 03 + 0.1040000000000000 03 03
P(7) = 0.7260000000000000 03 + 0.1040000000000000 03 03

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

G(1) = 0.1000000000000000 01 + 0.0000000000000000 00 01
G(2) = 0.9999999999999999 01 + 0.6000000000000000 01 01

Roots of P(x)  Multiplicities  Initial Approximation

Root(1) = -0.999999999999973500 00 + 0.50000000000000175420 00 01 1

No initial approximations

The following polynomial, G(x), contains all the roots of P(x) which have multiplicity 2:

G(1) = 0.1000000000000000 01 + 0.0000000000000000 00 01
G(2) = -0.999999999999901790 00 + 0.199999999999984580 01 01

Exhibit 6.19.
<table>
<thead>
<tr>
<th>Roots of ( p(x) )</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1 ) = ( 0.99999999999991790 ) 0 ( +0.199999999968450 ) 0 ( 1 )</td>
<td>2</td>
<td>No Initial Approximations</td>
</tr>
</tbody>
</table>

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

\[
2 \quad 1 = 0.1000000000000000 \quad 0 \quad + \quad 0.0000000000000000 \quad 0 \quad 1 \\
1 \quad 1 = -0.1999999999999679 \quad 0 \quad + \quad -0.2000000000000151 \quad 0 \quad 1 
\]

<table>
<thead>
<tr>
<th>Roots of ( p(x) )</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 3 ) = ( 0.1999999999999679 ) 0 ( +0.2000000000000151 ) 0 ( 1 )</td>
<td>3</td>
<td>No Initial Approximations</td>
</tr>
</tbody>
</table>

Exhibit 6.19. Roots Are: \( 2+2i \) (3), \( 1+2i \) (2), \(-1.5i \)
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTON'S METHOD TO EXTRACT ROOTS AND MULTIPLEITIES OF POLYNOMIALS

POLYNOMIAL NUMBER 2

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.100-02
TEST FOR CONVERGENCE: 0.100-05
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(15) = 0.4800000000000000 02 + 0.0000000000000000 00 01
P(14) = 0.2557120000000000 03 + 0.1940000000000000 00 01
P(13) = 0.1855565696000000 00 + 0.6496851464000000 01 01
P(12) = 0.1753568000000000 00 + 0.1206239778000000 02 01
P(11) = 0.4967984270000000 00 + 0.1765875444000000 02 01
P(10) = 0.1022394922130000 02 + 0.6030444231000000 01 01
P(9) = -0.1642712200956000 00 + 0.4173662934000000 01 01
P(8) = -0.2033625998420000 00 + 0.1095894275600000 01 01
P(7) = -0.1872556380000000 01 + 0.1029864973000000 01 01
P(6) = -0.1274997918900000 00 + 0.2171312274200000 01 01
P(5) = -0.2836425771600000 00 + 0.1926497278000000 01 01
P(4) = 0.3053900774700000 00 + 0.2998989141300000 01 01
P(3) = -0.1835899020000000 00 + 0.1827621600000000 01 01
P(2) = 0.2755620000000000 00 + 0.2755620000000000 00 01
P(1) = 0.2755620000000000 00 + 0.2755620000000000 00 01

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

G(2) = 0.4800000000000000 00 + 0.0000000000000000 00 01
G(1) = 0.2557120000000000 00 + 0.1940000000000000 00 01

ROOTS OF P(X) MULTIPLEITIES INITIAL APPROXIMATION

ROOT 1 = -0.23333333185333970 01 + 0.172950705709930-07 01

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

\[
G(x) = 0.1000000000000000 00 + 0.0000000000000000 00 \ I
G(x) = -0.2999997065586363 00 - 0.4999999802627250 00 \ I
G(x) = 0.3260999317648500 00 + 0.3000000000000000 00 \ I
G(x) = -0.6501999904602760 00 - 0.2250000129063650 00 \ I
G(x) = 0.66439966532830 00 + 0.6746003982170120 00 \ I
\]

ROOTO OF\( G(x) \):

ROOT(0) = 0.299998232053216170 00 - 0.2056986264886490 00 \ I
ROOT(1) = 0.476758016450669510 00 + 0.1000000000000000 00 \ I
ROOT(2) = 0.2331716103693670 00 - 0.1499999723417250 00 \ I
ROOTO(3) = -0.3928234548750490 00 - 0.1499999791666400 00 \ I

INITIAL APPROXIMATION

\[
0.6829629115654570 00 + 0.1274093284381870 00 \ I
0.7071067079330446 00 + 0.7071067079330446 00 \ I
0.7646110847094056 00 + 0.7646110847094056 00 \ I
\]

RESULTS OF SUBROUTINE QNAD

RESULTS OF SUBROUTINE QNAD

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

\[
G(x) = 0.1000000000000000 00 + 0.0000000000000000 00 \ I
G(x) = -0.2999997065586363 00 - 0.4999999802627250 00 \ I
G(x) = 0.3260999317648500 00 + 0.3000000000000000 00 \ I
\]

ROOTO OF\( P(x) \):

ROOT(0) = 0.299998232053216170 00 - 0.2056986264886490 00 \ I
ROOT(1) = 0.476758016450669510 00 + 0.1000000000000000 00 \ I
ROOT(2) = 0.2331716103693670 00 - 0.1499999723417250 00 \ I
ROOTO(3) = -0.3928234548750490 00 - 0.1499999791666400 00 \ I

INITIAL APPROXIMATION

\[
0.6829629115654570 00 + 0.1274093284381870 00 \ I
0.7071067079330446 00 + 0.7071067079330446 00 \ I
0.7646110847094056 00 + 0.7646110847094056 00 \ I
\]

RESULTS OF SUBROUTINE QNAD

RESULTS OF SUBROUTINE QNAD

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 GCD: 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

$p(9) = 0.1000000000000000D 01 + 0.0000000000000000D 00 01$
p(8) = -0.5175000000000001D 01 + -0.1150000000000000D 02 02
$p(7) = -0.5175000000000001D 01 + -0.1150000000000000D 02 03
$p(6) = 0.30750000000000000 03 + 0.1446250000000000 03 03
$p(5) = -0.49525000000000000D 03 + -0.49487500000000000 03 03
$p(4) = -0.49525000000000000D 03 + -0.49487500000000000 03 03
$p(3) = -0.49525000000000000D 03 + -0.49487500000000000 03 03
$p(2) = 0.18100000000000000D 03 + 0.4420000000000001D 03 03
$p(1) = 0.15800000000000000D 03 + 0.6000000000000001D 01 03

NO ROOTS OF MULTIPlicity 1

THE FOLLOWING POLYNOMIAL, $g(X)$, CONTAINS ALL THE ROOTS OF $p(X)$ WHICH HAVE MULTIPlicity 2

$g(9) = 0.1000000000000000D 01 + 0.0000000000000000D 00 01$
g(8) = -0.99999999996468D 00 + -0.19999999996468D 01 01

ROOTS OF $p(X)$ MULTIPlicITIES INITIAL APPROXIMATION

ROOT(1) = 0.99999999996468D 00 + 0.19999999996468D 01 2 NO INITIAL APPROXIMATIONS

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

\[
G_1 = 0.1000000000000000D01 + 0.0000000000000000D00
\]
\[
G_2 = -0.1000000000002539D01 + -0.2500000000001629D01
\]
\[
G_3 = -0.3000000000005296D01 + -0.1000000000004210D01
\]

ROOTS OF \( P(x) \) | MULTIPLEITIES | INITIAL APPROXIMATIONS
--- | --- | ---
ROOT(1) = 0.2000000000002532D01 + 0.2000000000001653D01 | 3 | NO INITIAL APPROXIMATIONS
ROOT(2) = -0.9999999999999926D00 + 0.4999999999999759D00 | 3 | NO INITIAL APPROXIMATIONS

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


**Repeated Use of the Greatest Common Divisor and Newtons Method to Extract Roots and Multiplicities of Polynomials**

**Polynomial Number 1**

**Number of Initial Approximations Given:** 0

**Maximum Number of Iterations:** 200

**Test for Zero in Subroutine GCD:** 0.100-02

**Test for Convergence:** 0.120-09

**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 12**

The coefficients are:

- \( p_{13} = 0.1000000000000000 01 \times 0.0000000000000000 00 1 \)
- \( p_{12} = -0.1200000000000000 02 + -0.0000000000000000 00 1 \)
- \( p_{11} = 0.7200000000000000 02 + 0.0000000000000000 00 1 \)
- \( p_{10} = -0.3000000000000000 03 + -0.0000000000000000 00 1 \)
- \( p_{9} = 0.7800000000000000 03 + 0.0000000000000000 00 1 \)
- \( p_{8} = -0.0000000000000000 04 + 0.0000000000000000 00 1 \)
- \( p_{7} = 0.2400000000000000 04 + 0.0000000000000000 00 1 \)
- \( p_{6} = -0.3200000000000000 05 + -0.0000000000000000 00 1 \)
- \( p_{5} = -0.3100000000000000 05 + 0.0000000000000000 00 1 \)
- \( p_{4} = 0.3200000000000000 06 + 0.0000000000000000 00 1 \)
- \( p_{3} = 0.1100000000000000 06 + 0.0000000000000000 00 1 \)
- \( p_{2} = -0.0400000000000000 07 + -0.0000000000000000 00 1 \)
- \( p_{1} = 0.0400000000000000 07 + 0.0000000000000000 00 1 \)

------------------------------

**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:

$G(1) = 0.1000000000000000 01 + 0.0000000000000000 00$;
$G(2) = -0.2000000000000000 01 - 0.0000000000000000 00$;
$G(3) = -0.2000000000000000 01 - 0.0000000000000000 00$.

ROUTES OF PI(x) MULTICITIES INITIAL APPROXIMATION

ROOT: 1) $0.1000000000000000 01 + 0.9999999999999999 00$ 6 NO INITIAL APPROXIMATIONS
ROOT: 2) $0.1000000000000000 01 + 0.9999999999999999 00$ 6 NO INITIAL APPROXIMATIONS

Exhibit 6.22. Roots Are: $1+i (6), 1-i (6)$
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NULL POLYNOMIALS METHOD TO EXTRACT ROOTS AND MULTICLITIES OF POLYNOMIALS

POLYNOMIAL NUMBER 1

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.10E-09
TEST FOR CONVERGENCE. 0.10E-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.10E-10
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE DEGREE OF PI1 IS 6. THE COEFFICIENTS ARE

P1(1) = -0.1000000000000000 00 + 0.0000000000000000 00 I
P1(2) = -0.0000000000000000 00 + 0.1000000000000000 00 02 I
P1(3) = -0.2000000000000000 00 + 0.2000000000000000 00 02 I
P1(4) = 0.1700000000000000 00 + 0.1600000000000000 00 01 I
P1(5) = -0.2700000000000000 00 + 0.2500000000000000 00 03 I
P1(6) = 0.7200000000000000 00 + 0.1600000000000000 00 03 I

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

G(1) = 0.1000000000000000 00 + 0.0000000000000000 00 I
G(1) = 0.9999999999973500 00 + 0.5000000000000000 00 00 I

ROOTS OF G(1)

ROOT 1] = -0.9999999999973500 00 + 0.500000000000175410 00 : 0.4829629156562790 00 + 0.1294095284438167 00 00 00 00

ROOTS OF P(1)

MULTICLITIES

INITIAL APPROXIMATION

ROOT 1] = -0.9999999999973500 00 + 0.500000000000175410 00 : 1 0.4829629156562790 00 + 0.1294095284438167 00 00 00 00

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

$G(2) = 0.1000000000000000D01 + 0.0000000000000000D00 I$
$G(1) = -0.9999999999999999D01 + -0.1999999999999845D01 I$

ROOTS OF $G(x)$

$\text{ROOT}(1) = 0.9999999999999179D00 + 0.199999999999684501 I$  \hspace{2cm} \text{INITIAL APPROXIMATION}

$\text{ROOT}(1) = 0.9999999999999179D00 + 0.199999999999684501 I$  \hspace{2cm} \text{MULTIPLE} 2

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

$G(1) = -0.99999999999999670D00 + -0.20000000000001510D01 I$

ROOTS OF $P(x)$

$\text{ROOT}(3) = 0.19999999999999670D01 + 0.20000000000001510D01 I$  \hspace{2cm} \text{INITIAL APPROXIMATIONS}

Exhibit 6.23. Roots Are: 2+2i (3), 1+i (2), -1+5i
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEUMANN METHOD TO EXTRACT ROOTS AND MULTIPlicITIES OF POLYNOMIALS

POLYNOMIAL NUMBER 2

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GED: 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[x] = 0.4400000000000000 0 0.0000000000000000 00 1
P[x] = 0.2557123000000000 0 0.1840000000000000 00 1
P[x] = -0.7155546000000000 00 1
P[x] = -0.9335505690000000 00 1
P[x] = 0.1013388448000000 00 1
P[x] = 0.4946909270400000 00 1
P[x] = -0.1022992321000000 00 1
P[x] = 0.1023622500000000 00 1
P[x] = -0.2036625048200000 00 1
P[x] = -0.2009942720000000 00 1
P[x] = -0.1971295780300000 00 1
P[x] = -0.1274697089000000 00 1
P[x] = 0.2814692716000000 00 1
P[x] = 0.1339544349000000 00 1
P[x] = 0.3356040374000000 00 1
P[x] = 0.2089909130000000 00 1
P[x] = 0.1927462160000000 00 1
P[x] = 0.2755620000000000 00 1

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

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

G[x] = 0.4800000000000000 0 0.0000000000000000 00 1
G[x] = 0.1119999999999999 00 1

ROOTS OF G(X)

INITIAL APPROXIMATION

ROOTS OF G(X)

ROOTS OF P[X] MULTIPlicITIES

Exhibit 6.24.
THE FOLLOWING POLYNOMIAL, $G(x)$, CONTAINS ALL THE ROOTS OF $P(x)$ WHICH HAVE MULTIPLICITY 2

<table>
<thead>
<tr>
<th>$G(x)$</th>
<th>Roots of $G(x)$</th>
<th>Multiplicities</th>
</tr>
</thead>
<tbody>
<tr>
<td>G15</td>
<td>$0.1000000000000000$</td>
<td>0.20569820646868600</td>
</tr>
<tr>
<td>G14</td>
<td>$0.2999706554843626$</td>
<td>0.20569820646868600</td>
</tr>
<tr>
<td>G13</td>
<td>$0.2249993931798880$</td>
<td>0.20569820646868600</td>
</tr>
<tr>
<td>G12</td>
<td>$-0.6703509024202729$</td>
<td>0.20569820646868600</td>
</tr>
<tr>
<td>G11</td>
<td>$0.44387946868886800$</td>
<td>0.20569820646868600</td>
</tr>
</tbody>
</table>

INITIAL APPROXIMATION

<table>
<thead>
<tr>
<th>$G(x)$</th>
<th>Roots of $G(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>G15</td>
<td>$0.1000000000000000$</td>
</tr>
<tr>
<td>G14</td>
<td>$0.2999706554843626$</td>
</tr>
<tr>
<td>G13</td>
<td>$0.2249993931798880$</td>
</tr>
<tr>
<td>G12</td>
<td>$-0.6703509024202729$</td>
</tr>
<tr>
<td>G11</td>
<td>$0.44387946868886800$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$G(x)$</th>
<th>Roots of $G(x)$</th>
<th>Multiplicities</th>
</tr>
</thead>
<tbody>
<tr>
<td>G15</td>
<td>$0.1000000000000000$</td>
<td>0.20569820646868600</td>
</tr>
<tr>
<td>G14</td>
<td>$0.2999706554843626$</td>
<td>0.20569820646868600</td>
</tr>
<tr>
<td>G13</td>
<td>$0.2249993931798880$</td>
<td>0.20569820646868600</td>
</tr>
<tr>
<td>G12</td>
<td>$-0.6703509024202729$</td>
<td>0.20569820646868600</td>
</tr>
<tr>
<td>G11</td>
<td>$0.44387946868886800$</td>
<td>0.20569820646868600</td>
</tr>
</tbody>
</table>

INITIAL APPROXIMATION

<table>
<thead>
<tr>
<th>$G(x)$</th>
<th>Roots of $G(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>G15</td>
<td>$0.1000000000000000$</td>
</tr>
<tr>
<td>G14</td>
<td>$0.2999706554843626$</td>
</tr>
<tr>
<td>G13</td>
<td>$0.2249993931798880$</td>
</tr>
<tr>
<td>G12</td>
<td>$-0.6703509024202729$</td>
</tr>
<tr>
<td>G11</td>
<td>$0.44387946868886800$</td>
</tr>
</tbody>
</table>

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: 100
Test for Zero in Subroutine Gold: 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 pixi is 6. The coefficients are:

pi9 = 0.100 0000000000000000 01 * 0.000 0000000000000000 00 1
pi8 = -0.500 0000000000000000 01 * -0.114 0000000000000000 02 1
pi7 = -0.412 7900000000000000 03 * -0.532 0000000000000000 03 1
pi6 = 0.114 7900000000000000 03 * 0.144 6790000000000000 05 1
pi5 = 0.307 5600000000000000 05 * -0.214 7500000000000000 03 1
pi4 = -0.492 5000000000000000 03 * -0.493 8750000000000000 03 1
pi3 = -0.585 7500000000000000 03 * -0.424 7500000000000000 03 1
pi2 = 0.181 5000000000000000 03 * -0.392 0000000000000000 03 1
pi1 = 0.158 0000000000000000 03 * 0.660 0000000000000000 03 1

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

No roots of multiplicity 1

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

The following polynomial, giX1, contains all the roots of pixi which have multiplicity 2.

gi2 = 0.100 0000000000000000 01 * 0.000 0000000000000000 00 1
gi1 = -0.999 9999999999999999 99 00 + -0.199 9999999999999999 99 01

Roots of giX1

Roots (giX1) = 0.999 9999999999999999 98 00 + 0.199 9999999999999999 99 01

Initial Approximation

0.482 929 2115662790 00 + 0.129 42563204438170 00 1

Exhibit 6.25.
### Roots of $P(x)$, Multiplicities, Initial Approximation

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicity</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>$0.9999999999946498D00 + 0.19999999999964680D00$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>$0.4824629115656279D00 + 0.1294095284438187D00$</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>$0.4824629115656279D00 + 0.1294095284438187D00$</td>
</tr>
</tbody>
</table>

The following polynomial, $G(x)$, contains all the roots of $P(x)$ which have multiplicity 3.

<table>
<thead>
<tr>
<th>Root</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$0.1000000000000000D00 + 0.0999999999999930D00 + 0.4829629115656279D00 + 0.1294095284438187D00$</td>
</tr>
<tr>
<td>2</td>
<td>$-0.1000000000000000D00 + 0.2599999999999930D00 + 0.1294095284438187D00$</td>
</tr>
<tr>
<td>3</td>
<td>$-0.1000000000000000D00 + 0.1000000000000000D00 + 0.1294095284438187D00$</td>
</tr>
</tbody>
</table>

### Roots of $G(x)$

<table>
<thead>
<tr>
<th>Root</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$-0.99999999999280D00 + 0.49999999999759D00 + 0.4829629115656279D00 + 0.1294095284438187D00$</td>
</tr>
<tr>
<td>2</td>
<td>$0.2000000000000000D00 + 0.2000000000000000D00 + 0.1294095284438187D00$</td>
</tr>
</tbody>
</table>

In the attempt to improve accuracy, root 2 did not converge after 20 iterations.

The present approximation is $0.20019999742533260 + 0.20019999742533260$.

### Roots of $P(x)$

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicity</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>$0.9999999999946498D00 + 0.19999999999964680D00$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>$0.4824629115656279D00 + 0.1294095284438187D00$</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>$0.4824629115656279D00 + 0.1294095284438187D00$</td>
</tr>
</tbody>
</table>

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.  0.100-05
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) = 0.10000000000000000 01 + 0.00000000000000000 00 1
P(12) = 0.12000000000000000 01 + 0.00000000000000000 00 1
P(11) = 0.72000000000000000 01 + 0.00000000000000000 00 1
P(10) = 0.28000000000000000 01 + 0.00000000000000000 00 1
P(9) = 0.78000000000000000 01 + 0.00000000000000000 00 1
P(8) = 0.38000000000000000 01 + 0.00000000000000000 00 1
P(7) = 0.32000000000000000 01 + 0.00000000000000000 00 1
P(6) = 0.38000000000000000 01 + 0.00000000000000000 00 1
P(5) = 0.44000000000000000 01 + 0.00000000000000000 00 1
P(4) = 0.38000000000000000 01 + 0.00000000000000000 00 1
P(3) = 0.38000000000000000 01 + 0.00000000000000000 00 1
P(2) = 0.38000000000000000 01 + 0.00000000000000000 00 1
P(1) = 0.38000000000000000 01 + 0.00000000000000000 00 1

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

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

\[ G(3) = 0.1000000000000000001 + 0.000000000000000000D 001 \]

\[ G(1) = -0.2000000000000066D 01 + -0.000000000000000000000 \]

ROOTS OF G(X)

\[ \text{ROOT}(1) = 0.1000000000000033D 01 + 0.9999999999999707D 001 \text{ Solved by Direct Method} \]

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:

\[ X_k = (X_{\text{START}} + 0.5k) (\cos \beta + i \sin \beta) \]

where

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

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.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 4 5 7 8</td>
</tr>
<tr>
<td>N O P L Y</td>
<td>N N I A P</td>
</tr>
<tr>
<td></td>
<td>1 7</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: $(j = 1, 3)$

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

where

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

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

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

3. Searching the Complex Plane

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

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

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

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

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

Example A.4

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

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

5. Solving Quadratic Polynomial

After N-2 roots of an \(n\)th degree polynomial have been extracted, the remaining quadratic, \(aX^2 + bX + c\), is solved using the quadratic formula

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

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

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

7. Miscellaneous

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

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

<table>
<thead>
<tr>
<th>12</th>
<th>45</th>
<th>78</th>
<th>64</th>
<th>70</th>
<th>72</th>
<th>78</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>NIAP</td>
<td>XSTART</td>
<td>XEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POLY</td>
<td>17</td>
<td>3</td>
<td></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 7\textsuperscript{th} 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 4 0 2 8 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>N O P</td>
<td>N I A P</td>
<td>XSTART</td>
<td>XEND</td>
</tr>
<tr>
<td>1 7 2</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
**NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS**

**POLYNOMIAL NUMBER 2 OF DEGREE 3**

The coefficients of this polynomial are:

\[
\begin{align*}
M_1(1) &= 0.10000000000000000 \quad 00 \\
M_1(2) &= 0.20000000000000000 \quad 00 \\
M_1(3) &= 0.10000000000000000 \quad 00 \\
M_1(4) &= 0.20000000000000000 \quad 00 \\
\end{align*}
\]

Numbers of initial approximations given: 0

Maximum number of iterations: 3

Test for convergence: 0.10D-01

Test for multiplicities: 0.10D-01

Radius to start search: 0.000 00

Radius to end search: 0.00D 00

No convergence for the following approximations after 3 iterations.

<table>
<thead>
<tr>
<th>Initial Approximation</th>
<th>Altered Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4876291156562790D 00</td>
<td>0.1294095264438187D 00</td>
</tr>
<tr>
<td>-0.4876291156562790D 00</td>
<td>-0.1294095264438187D 00</td>
</tr>
<tr>
<td>0.353553249184020D 00</td>
<td>-0.353553249184020D 00</td>
</tr>
<tr>
<td>0.7071067553046346D 00</td>
<td>-0.7071067553046346D 00</td>
</tr>
</tbody>
</table>

**Exhibit A.i.**
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 1 OF DEGREE 7

THE COEFFICIENTS OF PI) ARE

\[ P_1 = 0.1000000000000000 \times 10^{8} \]
\[ P_2 = 0.1000000000000000 \times 10^{8} \]
\[ P_3 = 0.5000000000000000 \times 10^{8} \]
\[ P_4 = 0.1000000000000000 \times 10^{8} \]
\[ P_5 = 0.7000000000000000 \times 10^{8} \]
\[ P_6 = -0.1234567890123456789 \times 10^{-2} \]
\[ P_7 = 0.1234567890123456789 \times 10^{-2} \]
\[ P_8 = 0.1234567890123456789 \times 10^{-2} \]

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 3
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.100-00
TEST FOR MULTIPLEITIES: 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) ARE

ROOTS OF PI:

\[ \text{ROOT}_1 = -0.29999999999999 \times 10^{5} \]
\[ \text{ROOT}_2 = 0.29999999999999 \times 10^{5} \]
\[ \text{ROOT}_3 = -0.39999999999999 \times 10^{4} \]
\[ \text{ROOT}_4 = 0.39999999999999 \times 10^{4} \]

MULTICIPILITIES

\[ \text{INITIAL APPROXIMATION} \]

\[ \text{ROOT}_1 = -0.29999999999999 \times 10^{5} \]
\[ \text{ROOT}_2 = 0.29999999999999 \times 10^{5} \]
\[ \text{ROOT}_3 = -0.39999999999999 \times 10^{4} \]
\[ \text{ROOT}_4 = 0.39999999999999 \times 10^{4} \]

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF PI)

ROOTS OF PI:

\[ \text{ROOT}_1 = -0.29999999999999 \times 10^{5} \]
\[ \text{ROOT}_2 = 0.29999999999999 \times 10^{5} \]
\[ \text{ROOT}_3 = -0.39999999999999 \times 10^{4} \]
\[ \text{ROOT}_4 = 0.39999999999999 \times 10^{4} \]

MULTICIPILITIES

\[ \text{INITIAL APPROXIMATION} \]

\[ \text{ROOT}_1 = -0.29999999999999 \times 10^{5} \]
\[ \text{ROOT}_2 = 0.29999999999999 \times 10^{5} \]
\[ \text{ROOT}_3 = -0.39999999999999 \times 10^{4} \]
\[ \text{ROOT}_4 = 0.39999999999999 \times 10^{4} \]

COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

\[ d_1 = 0.1000000000000000 \times 10^{8} \]
\[ d_2 = 0.1000000000000000 \times 10^{8} \]
\[ d_3 = 0.6000000000000000 \times 10^{8} \]
\[ d_4 = 0.6000000000000000 \times 10^{8} \]

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

THE COEFFICIENTS OF P(x) ARE

P(1) = 0.1000000000000000D 01
P(2) = -0.1000000000000000D 02
P(3) = -0.5900000000000000D 01
P(4) = 0.1900000000000000D 01
P(5) = 0.7000000000000000D 02
P(6) = -0.1624000000000000D 03
P(7) = 0.1422000000000000D 04
P(8) = -0.1566000000000000D 04

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

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

ROOT(1) = -0.2999999999999997D 01
ROOT(2) = 0.2000000000000000D 01
ROOT(3) = 0.3000000000000001D 01
ROOT(4) = -0.4000000000003000D 01
ROOT(5) = 0.2000000000000000D 01
ROOT(6) = -0.9999999999999976D 00
ROOT(7) = -0.2000000000000000D 01

ROOT(1) = -0.2999999999999997D 01
ROOT(2) = 0.2000000000000000D 01
ROOT(3) = 0.3000000000000001D 01
ROOT(4) = -0.4000000000003000D 01
ROOT(5) = 0.2000000000000000D 01
ROOT(6) = -0.9999999999999976D 00
ROOT(7) = -0.2000000000000000D 01

ROOT(1) = -0.2999999999999997D 01
ROOT(2) = 0.2000000000000000D 01
ROOT(3) = 0.3000000000000001D 01
ROOT(4) = -0.4000000000003000D 01
ROOT(5) = 0.2000000000000000D 01
ROOT(6) = -0.9999999999999976D 00
ROOT(7) = -0.2000000000000000D 01

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

ROOT(1) = -0.2999999999999997D 01
ROOT(2) = 0.2000000000000000D 01
ROOT(3) = 0.3000000000000001D 01
ROOT(4) = -0.4000000000003000D 01
ROOT(5) = 0.2000000000000000D 01
ROOT(6) = -0.9999999999999976D 00
ROOT(7) = -0.2000000000000000D 01

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.1 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)
MULT(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**

<table>
<thead>
<tr>
<th>Double Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>DABS(d)</td>
</tr>
<tr>
<td>DCOS(d)</td>
</tr>
<tr>
<td>DSIN(d)</td>
</tr>
<tr>
<td>DATAN2(d₁,d₂)</td>
</tr>
<tr>
<td>DSQRT(d)</td>
</tr>
</tbody>
</table>

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>
</tbody>
</table>
### TABLE B.III (Continued)

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPSQ</td>
<td>c.c. 37-42</td>
<td>Tolerance check for zero (0) in subroutine QUAD. Double precision. Right justify. 1.D-20 is recommended.</td>
</tr>
<tr>
<td>EPSMUL</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 Nth degree polynomial, N+1 coefficients must be entered one per card. The coefficient of the highest degree term is entered first. For example, if the polynomial $x^5 + 3x^4 + 2x + 5$ were to be solved, the order in which the coefficients would be entered is: 1, 3, 0, 0, 2, 5. Each
coefficient is entered, one per card, as described in Table B.IV and illustrated in Figure B.3.

TABLE B.IV
COEFFICIENT DATA FOR NEWTON'S METHOD

<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</td>
<td>c.c. 1-30</td>
<td>Real part of complex number. Double precision.</td>
</tr>
<tr>
<td>precision)</td>
<td></td>
<td>Right justify. If none, leave blank or enter 0.0D00.</td>
</tr>
<tr>
<td>VXZERO (XZERO in single</td>
<td>c.c. 31-60</td>
<td>Imaginary part of complex number. Double precision.</td>
</tr>
<tr>
<td>precision)</td>
<td></td>
<td>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>
</table>
| KCHECK        | c.c. 80      | Must contain the number 1. Integer.
3. Variables Used in Newton's Method

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

- \( R \) - real variable
- \( I \) - integer variable
- \( C \) - complex variable
- \( D \) - double precision
- \( L \) - logical variable
- \( A \) - alphanumeric variable

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

- \( E \) - entered
- \( R \) - returned
- \( ECR \) - entered, changed, and returned

4. Description of Program Output

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

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

The coefficients are printed under the heading "THE COEFFICIENTS OF \( P(X) \) ARE." The coefficient of the highest degree term is listed first (Exhibit 6.1).
As an aid to ensure the control information is correct, the number of initial approximations given, maximum number of iterations, test for convergence, test for multiplicities, radius to start search, and radius to end search are printed as read from the control card (Exhibit 6.1).

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

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

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

The initial approximation producing convergence to a root is printed to the right of the corresponding root and headed by "INITIAL APPROXIMATION." The initial approximations may be those supplied by the user, or generated by the program, or a combination of both (Exhibit A.3). See Appendix A, § 1 and § 2 for discussion of approximations.

The message "SOLVED BY DIRECT METHOD" indicates that the corresponding root or roots was obtained by Subroutine QUAD. See Appendix A, § 5.

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

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

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

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

This message is printed as a result of the value of the derivative of the original polynomial at an approximation, XXX, being zero (0). It occurred in the attempt to improve the accuracy of a zero. The previous message is then printed.
Figure B.1. Sequence of Input Data for Newton's Method
Figure B.2. Control Card for Newton's Method
<table>
<thead>
<tr>
<th>A (RA)</th>
<th>A (VA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.621735E+01</td>
<td>-0.132714E-02</td>
</tr>
</tbody>
</table>

Figure B.3. Coefficient 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.4. Initial Approximation Card for Newton's Method

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>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOPOLY</td>
<td>I</td>
<td>NOPOLY</td>
<td>I</td>
<td></td>
<td>Number of the polynomial</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td></td>
<td>Degree of the polynomial</td>
</tr>
<tr>
<td>NIAP</td>
<td>I</td>
<td>NIAP</td>
<td>I</td>
<td></td>
<td>Number of initial approximations to be read</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td></td>
<td>Maximum number of iterations to be performed</td>
</tr>
<tr>
<td>EPSCNV</td>
<td>R</td>
<td>EPSCNV</td>
<td>D</td>
<td></td>
<td>Tolerance check for convergence</td>
</tr>
<tr>
<td>EPSMUL</td>
<td>R</td>
<td>EPSMUL</td>
<td>D</td>
<td></td>
<td>Tolerance check for multiplicities</td>
</tr>
<tr>
<td>EPSQ</td>
<td>R</td>
<td>EPSQ</td>
<td>D</td>
<td></td>
<td>Tolerance check for zero in subroutine QUAD</td>
</tr>
<tr>
<td>XSTART</td>
<td>R</td>
<td>XSTART</td>
<td>D</td>
<td></td>
<td>Magnitude from which to begin the search for zeros</td>
</tr>
<tr>
<td>XEND</td>
<td>R</td>
<td>XEND</td>
<td>D</td>
<td></td>
<td>Magnitude to end the search for zeros</td>
</tr>
<tr>
<td>KCHECK</td>
<td>I</td>
<td>KCHECK</td>
<td>I</td>
<td></td>
<td>Program Control. When KCHECK = 1, program will terminate execution.</td>
</tr>
<tr>
<td>NA</td>
<td>I</td>
<td>NA</td>
<td>I</td>
<td></td>
<td>Number of coefficients or original polynomial</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>RA,VA</td>
<td>D</td>
<td></td>
<td>Array containing the coefficients of original polynomial P(X)</td>
</tr>
<tr>
<td>NDEF</td>
<td>I</td>
<td>NDEF</td>
<td>I</td>
<td></td>
<td>Degree of current deflated polynomial</td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>L</td>
<td>I</td>
<td></td>
<td>Counter for number of initial approximations used</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td></td>
<td>Counter for number of iterations</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
<td>I</td>
<td></td>
<td>Counter for number of roots found (counting multiplicities)</td>
</tr>
<tr>
<td>IALTER</td>
<td>I</td>
<td>IALTER</td>
<td>I</td>
<td></td>
<td>Counter for number of alterations of each initial approximation</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
<td></td>
<td>Program control</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td></td>
<td>Counter for number of distinct roots found</td>
</tr>
<tr>
<td>ND</td>
<td>I</td>
<td>ND</td>
<td>I</td>
<td></td>
<td>Program control &amp; number of coefficient of deflated polynomial for which no zeros were found</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>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>RFX, VFX</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>D</td>
<td>C</td>
<td>RD,VD</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</td>
<td>Double Precision</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>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Variable</td>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>RA,VA</td>
<td>D</td>
<td>Array of coefficients of polynomial</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>RB,VB</td>
<td>D</td>
<td>Array of coefficients of deflated polynomial</td>
<td></td>
</tr>
<tr>
<td>NDEF</td>
<td>I</td>
<td>NDEF</td>
<td>I</td>
<td>Degree of polynomial</td>
<td></td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
<td>I</td>
<td>Number of coefficients of polynomial</td>
<td></td>
</tr>
<tr>
<td>X0</td>
<td>C</td>
<td>RX0,VX0</td>
<td>D</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>
<td></td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>R PX,VPX</td>
<td>D</td>
<td>Value of polynomial at (X_n)</td>
<td></td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX,VDPX</td>
<td>D</td>
<td>Value of the derivative of polynomial at (X_n)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>RC,VC</td>
<td>D</td>
<td>Array of containing sequence of values leading to the derivative</td>
<td></td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>R PX,VPX</td>
<td>D</td>
<td>Value of polynomial at (X_n)</td>
<td></td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX,VDPX</td>
<td>D</td>
<td>Derivative of polynomial at (X_n)</td>
<td></td>
</tr>
<tr>
<td>X0</td>
<td>C</td>
<td>RX0,VX0</td>
<td>D</td>
<td>Current approximation ((X_n)) to root</td>
<td></td>
</tr>
<tr>
<td>XNEW</td>
<td>C</td>
<td>RXNEW,VXNEW</td>
<td>D</td>
<td>New approximation ((X_{n+1})) to root</td>
<td></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>Tolerance for convergence or multiplicity check</th>
</tr>
</thead>
<tbody>
<tr>
<td>FX</td>
<td>C</td>
<td>RPX,VPX</td>
<td>D</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>Derivative of (P(X)) at (X_n)</td>
</tr>
<tr>
<td>XO</td>
<td>C</td>
<td>RX0,VX0</td>
<td>D</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>Unit number of output device</td>
</tr>
<tr>
<td>KANS</td>
<td>I</td>
<td>KANS</td>
<td>I</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 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>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>XZERO</td>
<td>C</td>
<td>RXZERO, VXZERO</td>
<td>D</td>
<td>E</td>
<td>Array of approximations</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>RX, VX</td>
<td>D</td>
<td>ECR</td>
<td>Array of roots</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>RA, VA</td>
<td>D</td>
<td>E</td>
<td>Coefficients of original (undefined) polynomial, P(X)</td>
</tr>
<tr>
<td>COEF</td>
<td>C</td>
<td>RCOEF, VCOEF</td>
<td>D</td>
<td>E</td>
<td>Working array for coefficients of polynomial</td>
</tr>
<tr>
<td>NA</td>
<td>I</td>
<td>NA</td>
<td>I</td>
<td>E</td>
<td>Number of coefficients of original polynomial</td>
</tr>
<tr>
<td>X0</td>
<td>C</td>
<td>RX0, VX0</td>
<td>D</td>
<td>E</td>
<td>Current approximation (Xn) to root</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 Xn</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>RPX, VPX</td>
<td>D</td>
<td>E</td>
<td>Value of P(X) at Xn</td>
</tr>
<tr>
<td>KANS</td>
<td>I</td>
<td>KANS</td>
<td>I</td>
<td>E</td>
<td>KANS = 1 implies convergence; KANS = 0 implies no convergence</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td></td>
<td>Counter for number of iterations</td>
</tr>
<tr>
<td>XNEW</td>
<td>C</td>
<td>RXNEW, VXNEW</td>
<td>D</td>
<td>E</td>
<td>New approximation (Xn+1) to root</td>
</tr>
<tr>
<td>NN</td>
<td>I</td>
<td>NN</td>
<td>I</td>
<td></td>
<td>Degree of polynomial</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>RC, VC</td>
<td>D</td>
<td>E</td>
<td>Array containing the sequence of values leading to the derivative</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td>E</td>
<td>Number of distinct roots of P(X) found</td>
</tr>
<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>B</td>
<td>C</td>
<td>RB, VB</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of deflated polynomial</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>EPSCHK</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
<td>E</td>
<td>Tolerance for checking convergence</td>
</tr>
</tbody>
</table>

Subroutine BETTER

Array containing initial approximations

Number of initial approximations to be generated
<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>IO2</td>
<td>I</td>
<td>IO2</td>
<td>I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
</tbody>
</table>

Subroutine ALTER

Subroutine QUAD

<p>| A                        | C    | UA, VA                    | D    | E                       | Coefficients of polynomial to be solved |
| NA                       | I    | NA                        | I    | E                       | Degree of polynomial |
| ROOT                     | C    | UROOT, VROOT              | D    | ECR                     | Array of roots of ( P(X) ) (original polynomial) |
| NROOT                    | I    | NROOT                     | I    | ECR                     | Number of distinct roots of ( P(X) ) (the original polynomial) |</p>
<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Single Precision Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Double Precision 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></td>
<td>Array containing multiplicities of each root</td>
<td></td>
</tr>
<tr>
<td>EPST</td>
<td>R</td>
<td>EPST</td>
<td>D</td>
<td>E</td>
<td></td>
<td>Tolerance check for the number zero</td>
<td></td>
</tr>
<tr>
<td>DISC</td>
<td>C</td>
<td>UDISC, VDISC</td>
<td>D</td>
<td></td>
<td></td>
<td>Value of the discriminate (b^2 - 4ac) of Quadratic</td>
<td></td>
</tr>
</tbody>
</table>

Subroutine COMSQT

- UX, VX: Complex number for which the square root is desired
- UY, VY: Square root of the complex number
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)
126

TABLE B. VIII
PROGRAM FOR NEWTON'S METHOD
C
C

*

C

* DOUBLE PRECISION PROGRAM FOR NEWTON'S METHOD

C

*
*

C
C
C
C
C

C
C
C

0001
0002
0003
0004
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025
0026
0027
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040

*
*
*

* NEWTONS METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
* POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX* IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION
* FORMULA
XIN+I) = XIN)-PIXIN)I/P'IXINI).
*
*
***************************************************************************

DOUBLE PRECISION RAtVA,RXZERO,VXZERODRBVB,RCOEFeVCOEFRX,VXtRXINI
ITVXINIT,RCvVC,RODVDRPXeVPXtRDPXXVDPX#RXNEWeVXNEW*RXO*VXO EPSCHK,
2EPSCNV,EPSQEPSMULtXSTARTXENOABPX,ABOPX
DIMENSION RA(26),VAI261,RBt26),VB(26)IRCI26),VC(26),RD(26),VDO26),
IRCOEF(26),VCOEF(26),MULTI251RXZERO(251,VXZEROI251,RX(25)1VX(25|1R
2XINIT(25 ,VXINIT(25)
COMMON EPSCHK,MAX,102
101=5
102=6
1 READ(101,10001 NOPOLYNNIAPMAX,EPSCNVEPQtSQEPSMULXSTARTXENDKC
IHECK
IF(KCHECK.EQ.1) STOP
NA=N+1
IRA(I),VAII),I=INA)
READIIOI,I010)
WRITE(1O2,10331 NOPOLYN
WRITE(IO2,1040) (IeRA(I),VA(I),=1,NAI
WRITE(I02,2060)
WRITE(1I2.2000) NIAP
WRITE(IO2,2010) MAX
WRITE(IO2,2020) EPSCNV
WRITE(102,2030) EPSMUL
WRITEII02,2040) XSTART
WRITE(IO2,2050) XEND
IF(NIAP.NE.OI GO TO 3
NIAP=N
CALL GENAPP(RXZEROVXZERO,NIAP.XSTARTI
GO TO 4
3 READ(ID1,1020) (RXZEROIII,VXZEROIII)I=INIAPI
4 NDEF=N
L=1
ITER=O
NROOT=O
=0
IROOT O
ITIME=O
ND=O
IALTER=O
K=O
RXO=RXZEROILI
VXO=VXZERO(L)
00 5 I=I1NA
RCOEF)II=RA(Il
5 VCOEF(I)=VA(I)
10 CALL HORNER(RCOEFVCOEF,RXOVXRONDEF,R8,VB.RCVCRPXVPXROPX,VOPX
13
ABPX=DSQRT(RPX*RPX+VPX*VPXI
ABDPX=DSQRTIRDPX*RDPX+VDPX*VDPXI

112
116

130
132

152
154
160

164
168
180
181
188
190
191


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(RXZERO(L), VXZERO(L), ITALTER, ITTIME)
0054 IF(ITALTER .GT. 5) GO TO 110
0055 RXO=RXZERO(L)
0056 VXO=VXZERO(L)
0057 ITER=0
0058 GO TO 10
0059 60 ND=NDEF+1
0060 DO 65 J=1, ND
0061 RD(J)=RCOEF(J)
0062 65 GO TO 140
0063 70 NRDEF=NRDEF+1
0064 K=K+1
0065 MULT(K)=1
0066 RX(K)=RXO
0067 VX(K)=VXO
0068 RXINIT(K)=RXZERO(L)
0069 VXINIT(K)=VXZERO(L)
0070 CALL HORNER(RCOEF, VCOEF, RXO, VXO, NDEF, RX, VX, RXZERO(L), VXZERO(L))
0071 GO TO 147
0072 IF(NRDEF .GE. N) GO TO 147
0073 NDEF=NDEF-1
0074 NUM=NUM-1
0075 DO 105 I=1, NUM
0076 RCOEF(I)=RB(I)
0077 105 CONTINUE
0078 CALL HORNER(RCOEF, VCOEF, RXO, VXO, NDEF, RX, VX, RXZERO(L), VXZERO(L))
0079 ABPX=DSQRT(RPX*RPX+VPX*VPX)
0080 ABDPX=DSQRT(RDPX*RDPX+VDPX*VDPX)
0081 IF(ABDPX .NE. 0.0) GO TO 107
0082 IF(ABPX .EQ. 0.0) GO TO 130
0083 GO TO 110
0084 107 CONTINUE
0085 EPSCHK=EPSCHV
0086 CALL CHECK(RPX, VPX, RDPX, VDPX, RXO, VXO, KANS)
0087 IF(KANS .EQ. 1) GO TO 130
0088 110 IF(NDEF .GT. 21) GO TO 113
0089 K=K+1
0090 CALL QUAD(RCOEF, VCOEF, NDEF, RX, VX, MULT, EPSQ)
0091 GO TO 150
0092 113 IF(IXADD .LT. NIAP) GO TO 115
0093 IF(IXEND .EQ. 0.0) GO TO 60
0094 IF(IXSTART .GT. IXEND) GO TO 60
0095 NIAP=N
0096 CALL GENAPP(RXZERO, VXZERO, NIAP, XSTART)
```
TABLE B. VIII (Continued)

```
128

0097       L=0
0098       L=L+1
0099       RX=RXZERO(L)
0100       VX=VXZERO(L)
0101       ITER=0
0102       IALTER=0
0103       GO TO 10
0104       130       MULT(K)=MULT(K)+1
0105       NROOT=NROOT+1
0106       GO TO 80
0107       140 IF(K.EQ.0) GO TO 160
0108       147       KROOT=K
0109       150 WRITE(I02,1025)
0110       WRITE(I02,1050)
0111       WRITE(I02,1060) I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,1,RO
0112       160 IF(K.EQ.0) GO TO 170
0113       WRITE(I02,1065)
0114       WRITE(I02,1070)
0115       WRITE(I02,1075) I,RX(I),VX(I),I=1,1
0116       170 IF(NROOT.EQ.0) GO TO 1
0117       WRITE(I02,1080)
0118       WRITE(I02,1085)
0119       WRITE(I02,1090) I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,1,RO
0120       180 WRITE(I02,1095)
0121       WRITE(I02,1096)
0122       WRITE(I02,1097)
0123       WRITE(I02,1098) I,RX(I),VX(I),MULT(I),RXINIT(I),VXINIT(I),I=1,1,RO
0124       190 WRITE(I02,1099)
0125       WRITE(I02,1100)
0126       WRITE(I02,1101)
0127       WRITE(I02,1102)
0128       200 FORMAT(1X,12,4HNEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS/9K,18
0129       201 FORMAT(1X,12,4HNUMBER OF DEGREE/12,12,4HCOEFFICIENT
0130       202 FORMAT(1X,12,4HOF POLYNOMIAL/12,12,4HOF ZEROS WERE FOUND/12,12,4H)
0131       203 FORMAT(1X,12,4HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS
0132       204 FORMAT(1X,12,4HOF PKX ARE/12,12,4H)
0133       205 FORMAT(1X,12,4HNOT OF PKX ARE/12,12,4H)
0134       206 FORMAT(1X,12,4HAPPROXIMATION/12,12,4H)
0135       207 FORMAT(1X,12,4HAPPROXIMATIONS/12,12,4H)
0136       208 FORMAT(1X,12,4HINITIAL/12,12,4H)
0137       209 FORMAT(1X,12,4HAPPROXIMATE, THE ZEROS
0138       210 FORMAT(1X,12,4HOF PKX ARE/12,12,4H)
0139       211 FORMAT(1X,12,4HFOR WHICH NO ZEROS WERE FOUND/12,12,4H)
0140       212 FORMAT(1X,12,4HOF PKX ARE/12,12,4H)
0141       213 FORMAT(1X,12,4HOF PKX ARE/12,12,4H)
0142       214 FORMAT(1X,12,4HOF PKX ARE/12,12,4H)
0143       215 FORMAT(1X,12,4HOF PKX ARE/12,12,4H)
0144       216 FORMAT(1X,12,4HOF PKX ARE/12,12,4H)
0145       217 FORMAT(1X,12,4HOF PKX ARE/12,12,4H)
```

TABLE B. VIII (Continued)

```c
0001  SUBROUTINE GENAPP(APPR,APPL,NAPP,XSTART)
        ************************************************************************
        * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE
        * DEGREE OF THE ORIGINAL POLYNOMIAL.
        ************************************************************************
        PROGRAM GENAPP
        COMMON DUMMY,MAX,102
        IF (XSTART.EQ.0.3) XSTART=0.5
        BETA=0.2617994
        DO 10 =1,NAPP
        APPRI=XSTART*DCOS(BETA)
        APPPI=XSTART*DSIN(BETA)
        10 XSTART=XSTART+0.5
        RETURN
        END

0001  SUBROUTINE ALTER(XOLDR,XOLDI,NALTER,TIME)
        ************************************************************************
        * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO
        * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
        ************************************************************************
        PROGRAM ALTER
        COMMON DUMMY,MAX,102
        IF(TIME.NE.0) GO TO 5
        WRITE(102,1010) MAX
        5 IF(NALTER.EQ.0) GO TO 10
        WRITE(102,1000) XOLDR,XOLDI
        GO TO 20
        10 XABOLD=DSORT(XOLDR*XOLDR*XOLDI) XOLDI
        BETA=DATAZ(XOLDI,XOLDI)
        11 WRITE(102,1020) XOLDR,XOLDI
        20 NALTER=NALTER+1
        IF(NALTER.GT.5) RETURN
        GO TO 30
        15 NALTER=NALTER+1
        20 NALTER=NALTER+1
        30 XOLDR=XOLDR
        31 XOLDI=XOLDI
        40 BETA=BETA+1.0471976
        41 XOLDR=ABXOLD*DCOS(BETA)
        42 XOLDI=ABXOLD*DSIN(BETA)
        50 RETURN
        1000 FORMAT(1X,023.16,3H + ,023.16,2H 1,10X,21MALERATED APPROXIMATION)
        1010 FORMAT(1X,5.6,HND CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
        ITER .1,12H ITERATIONS/)
        1020 FORMAT(1X,023.16,3H + ,023.16,2H 1,10X,21INITIAL APPROXIMATION)
        END
```
SUBROUTINE QUAD(UA, VA, UROOT, VROOT, MULTI, EPST)

DOUBLE PRECISION UA(261), VA(261), UROOT(251), VROOT(251), MULTI(251)

DIMENSION UA(261), VA(261), UROOT(251), VROOT(251), MULTI(251)

IF(INA.EQ.2) GO TO 7
IF(NA.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
7888 = UA(I) * UA(I) + VA(I) * VA(I)

UROOT(INROOT+1) = -I - U2 * U(A - VA) * VA(I) / 8B
VROOT(INROOT+1) = IVBB = DSQRT(UUDISC*UDISC+VDISC*VDISC)

IF(OBB.LT.EPST) GO TO 10
CALL COMSQT(UUDISC*UDISC, UDDUMMY, UDDUMMY)
UBBB = UA(I) + UDDUMMY
VBBB = VA(I) + VDDUMMY
RDUMMY = -UAI2 - UDDUMMY
SDUMMY = -VAI2 - VDDUMMY
UAAA = 2.0 * UA(I)
VAAA = 2.0 * VA(I)
BBB = UAAA * UAAA + VAAA * VAAA
UDISC = UA(I) * UA(I) - VA(I) * VA(I) - (4.0 * UA(I) * VA(I) - UA(I) * VA(I))

VBBB = UAAA * UAAA + VAAA * VAAA

UROOT(INROOT+1) = (-UA(I) * UAAA - VA(I) * VAAA) / BBB
VROOT(INROOT+1) = (SDUMMY * UAAA - RDUMMY * VAAA) / BBB
MULTI(INROOT+1) = 2
MULTI(INROOT+2) = 1
NROOT = NROOT + 2
GO TO 50
10 UAAA = 2.0 * UA(I)
VAAA = 2.0 * VA(I)
BBB = UA(I) * UA(I) + VA(I) * VA(I)

UROOT(INROOT+1) = (UUBB = UA(I) * UAAA + VAAA * VAAA) / BBB
VROOT(INROOT+1) = (UUBB = UA(I) * UAAA + VAAA * VAAA) / BBB
MULTI(INROOT+1) = 1
NROOT = NROOT + 1
GO TO 50

RETURN
END
### TABLE B. VIII (Continued)

```c
SUBROUTINE COMSQTIUXtVXiUYsVYI
C
*************************************************************************
C
* THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER. *
C
*************************************************************************
C
DOUBLE PRECISION UX, VX, VY, DUMMY, R, AAA, BBB

R = DSQRTIUX*UX + VX*VX))
AAA = DSQRT(DABS((IRUXI/2.0))
BBB = DSQRT(DABS((R - UXI/2.0))
IF(VXI 10, 20, 30
10 UY=AAA
20 UY=-1.0*BBB
0009 GO TO 100
0010 IF(VXI 40, 50, 60
0011 30 UY=AAA
0012 VY=BBB
0013 GO TO 100
0014 40 DUMMY=DABS(UX)
0015 UY=0.0
0016 VY=DSQRT(DUMMY)
0017 GO TO 100
0018 50 UY=0.0
0019 VY=0.0
0020 GO TO 100
0021 60 DUMMY=DABS(UX)
0022 UY=DSQRT(DUMMY)
0023 VY=0.0
0024 100 RETURN
0025 END
```
TABLE B. VIII (Continued)

0001 SUBROUTINE HORNER(RA, VA, RX0, VX0, NDEF, RB, VB, RC, VX, RPX, VPX, RDPX, VDPX)
C *************************************************************************************************
C * HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL PIXI AT A POINT D AND *
C * ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO DEFLATE THE *
C * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D). *
C *************************************************************************************************
C
0002 DOUBLE PRECISION VDPX, RXO, VX0, RB, VB, RC, VX, RPX, RDPX, RA, VA
0003 DIMENSION RA(26), VA(26), RB(26), VB(26), RC(26), VC(26)
0004 NUM=NDEF+1
0005 DO 10 I=2, NUM
0006 RB(I)=RA(I)+(RB(I-1)*RXD-VB(I-1)*VXO)
0007 VB(I)=VA(I)+4VB(I)*RXO+RB(I-1)*VXO)
0008 RPX=RB(NUM)
0009 VPX=VB(NUM)
0010 RC(I)=RB(I) 540
0011 VC(I)=VB(I)
0012 IF(NDEF.LT.2) GO TO 25
0013 DO 20 J=2, NDEF
0014 RC(J)=RB(J)+(RC(J-1)*RXO-VC(J-1)*VXO)
0015 VC(J)=VBIJ)+IVC(J-1)*RXO+RC(J-1)*VXO)
0016 25 RDPX=RC(NDEF)
0017 20 VPX=VC(NDEF)
0018 RETURN
0019 END
0020

0001 SUBROUTINE NEWTON(RPX, VPX, RDPX, VDPX, RX0, VX0, RXNEW, VXNEW)
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)-PI(X(N))/P'(X(N)). *
C *************************************************************************************************
C
0002 DOUBLE PRECISION RPX, VPX, RDPX, VDPX, RX0, VX0, RXNEW, VXNEW, ARG
0003 DOUBLE PRECISION DDD
0004 ARG=RPX*RDPX+VPX*VDPX
0005 DDD=DSSORT(ARG)
0006 IF(DDD.EQ.0.0) RETURN
0007 RXNEW=RX0-((RPX*RDPX+VPX*VDPX)/ARG)
0008 VXNEW=VX0-((VPX*RDPX+RPX*VDPX)/ARG)
0009 RETURN
0010 END
0011
TABLE E. VIII (Continued)

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

* THIS SUBROUTINE CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROXIMATIONS 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, VOU

IMM, EPS

COMMON EPS, MAX, ID2

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.) GO TO 25

RDUMMY=(RPX*RDPX+VPX*VDPX)/ARG

VDUMMY=(VPX*RDPX-RPX*VDPX)/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)

SUBROUTINE BETTER(K, RXZERO, VZERO, RX, VX, RA, RN, VCOEF, NC,
               VC, RB, VB)
***************************************************************************
C * SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS
C * FOUND BY USING THEM AS INITIAL APPROXIMATIONS WITH NEWTON'S METHOD
C * APPLIED TO THE FULL, UNDEFLATED POLYNOMIAL.
C ****************************************************************************
DOUBLE PRECISION RXZERO, VZERO, RX, VX, RA, RN, VCOEF, NC, VC, RB, VB
IMPLEMENT
DIMENSION RXZERO(25), VZERO(25), RX(25), VX(25), RA(26), RN(26), VCOEF(26), NC(26), VC(26), RB(26), VB(26)
DOUBLE PRECISION RXNEW, VXNEW
COMMON EPSMAX, O02
DO 10 I = 1, K
RXZERO(I) = RX(I)
10 VXZERO(I) = VX(I)
DO 20 I = 1, NA
RCOEF(I) = RA(I)
20 VCOEF(I) = VA(I)
DO 50 J = 1, K
RXO = RXZERO(J)
VXO = VXZERO(J)
NN = N
ITER = 0
30 CALL HORNER(RCOEF, VCOEF, RXO, RXO, NN, RB, RB, VC, VXO, RX, VX, VX, RXNEW, VB, VB)
ABPX = DSQRT(RPX*RPX + VDPX*VDPX)
ABDPX = DSQRT(RDPX*RDPX + VDPX*VDPX)
IF(ABPX .NE. 0.0) GO TO 33
IF(ABDPX .EQ. 0.0) GO TO 40
GO TO 34
33 CALL NEWTON(RPX, VPX, RDPX, VDPX, RXO, VXO, RXNEW, VXNEW)
ITER = ITER + 1
RXO = RXNEW
VB = VBNEW
VZERO = RN
CALL CHECK(RPX, VPX, RDPX, VDPX, RXO, VXO, KANS)
IF(KANS .LE. 0) GO TO 40
IF(ITER .GE. MAX) GO TO 35
GO TO 30
34 WRITE(102, 1112) RXO, VXO
35 WRITE(102, 100) J, RXZERO(J), VXZERO(J)
36 WRITE(102, 200) MAX
37 RXJ = RXO
38 VX(J) = VXO
39 CONTINUE
40 RETURN
1112 FORMAT(11H, 15H, THE VALUE OF THE DERIVATIVE AT X0 = , D23, 16.3H + , D10
13.16.2H 1.10H IS ZERO.)
100 FORMAT(12H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(162, 4H1 = , D23
1.16.3H + , D023, 16.2H 1.10H DID NOT CONVERGE.)
200 FORMAT(13H THE PRESENT APPROXIMATION AFTER 13, 29H ITERATIONS IS PRINTED BELOW.)
END

880
2. Use of the Program

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

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

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

Double Precision

Main Program

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

Subroutine BETTER

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

Subroutine GENAPP

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

Subroutine HORNER

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

Subroutine QUAD

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

Table C.II lists the system functions used in the program of Muller's method. In the table "d" denotes a double precision variable name.
TABLE C.II
SYSTEM FUNCTIONS USED IN MULLER'S METHOD

Double Precision

DABS(d) - obtain absolute value
DATAN2(d1, d2) - arctangent of d1/d2
DSQRT(d) - square root
DCOS(d) - cosine of angle
DSIN(d) - sine of angle
DSQRT(d) - square root

2. Input Data for Muller's Method

The input data for Muller's method is identical to the input data for Newton's method as described in Appendix B, § 2 except for the variable names. The correspondence of input variable names is given in Table C.III. Only one (not three) initial approximation, 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><strong>Control Card</strong></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><strong>Coefficient Card</strong></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><strong>Initial Approximation Card</strong></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><strong>End Card</strong></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, 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>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>Degree of polynomial P(X)</td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>NRROOT</td>
<td>I</td>
<td>NRROOT</td>
<td>I</td>
<td>Number of distinct roots (counting multiplicities)</td>
<td></td>
</tr>
<tr>
<td>NOMULT</td>
<td>I</td>
<td>NOMULT</td>
<td>I</td>
<td>Array containing the roots</td>
<td></td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT, VROOT</td>
<td>D</td>
<td>Number of initial approximations to be read in</td>
<td></td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
<td>Array of initial approximations</td>
<td></td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>UAPP, VAPP</td>
<td>D</td>
<td>Working array containing coefficients of current polynomial</td>
<td></td>
</tr>
<tr>
<td>WORK</td>
<td>C</td>
<td>UWORK, VWORK</td>
<td>D</td>
<td>Array containing coefficients of deflated polynomial</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB, VB</td>
<td>D</td>
<td>Array containing coefficients of original polynomial, P(X)</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA, VA</td>
<td>D</td>
<td>Array of initial or altered approximations for which convergence was obtained</td>
<td></td>
</tr>
<tr>
<td>RAPP</td>
<td>C</td>
<td>URAPP, VRAAPP</td>
<td>D</td>
<td>One of three current approximations to a root</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>C</td>
<td>UX1, VX1</td>
<td>D</td>
<td>One of three current approximations to a root</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>UX2, VX2</td>
<td>D</td>
<td>One of three current approximations to a root</td>
<td></td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3, VX3</td>
<td>D</td>
<td>Value of polynomial P(X) at X1</td>
<td></td>
</tr>
<tr>
<td>PX1</td>
<td>C</td>
<td>UFX1, VFX1</td>
<td>D</td>
<td>Value of polynomial P(X) at X2</td>
<td></td>
</tr>
<tr>
<td>PX2</td>
<td>C</td>
<td>UFX2, VFX2</td>
<td>D</td>
<td>Value of polynomial P(X) at X3</td>
<td></td>
</tr>
<tr>
<td>PX3</td>
<td>C</td>
<td>UFX3, VFX3</td>
<td>D</td>
<td>Newest approximation (Xn+1) to root</td>
<td></td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4, VX4</td>
<td>D</td>
<td>Value of polynomial P(X) at X4</td>
<td></td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td>Array containing the multiplicities of each root found</td>
<td></td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td>Counter for iterations</td>
<td></td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
<td>Unit number of input device</td>
<td></td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td>Unit number of output device</td>
<td></td>
</tr>
</tbody>
</table>
TABLE C.IV. (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>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>
</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>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td></td>
<td>Maximum number of iterations</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
<td></td>
<td>Tolerance check for convergence</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
<td>D</td>
<td></td>
<td>Tolerance check for zero (0)</td>
</tr>
<tr>
<td>EPSM</td>
<td>R</td>
<td>EPSM</td>
<td>D</td>
<td></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></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></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></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>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>Array of current polynomial coefficients (to be deflated or evaluated)</td>
</tr>
<tr>
<td>NA</td>
<td>I</td>
<td>Degree of polynomial to be deflated or evaluated</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>Approximation at which to evaluate or deflate the polynomial</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Type</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB, VB</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>UPX, VPX</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3, VX3</td>
</tr>
<tr>
<td>X₄</td>
<td>C</td>
<td>UX₄, VX₄</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
</tr>
<tr>
<td>DENOM</td>
<td>R</td>
<td>DENOM</td>
</tr>
</tbody>
</table>

Subroutine TEST

| MULT                     | I    | I    | ECR | Array of multiplicities of each root |
| A                        | C    | UA, VA | D   | Array of coefficients of original undeflated polynomial |
| NP                       | I    | I    | ECR | Degree of original polynomial |
| ROOT                     | C    | UROOT, VROOT | D | Array of roots |
| NROOT                    | I    | NROOT | I   | Number of roots stored in root |
| BAPP                     | C    | UBAPP, VBAPP | D | Array of initial approximations (old roots) |
| IROOT                    | I    | IROOT | I   | Number of roots solved by the iterative process (Not QUAD) |
| ROOTS                    | C    | UROOTS, VROOTS | D | Temporary storage for new (better) roots |
| L                        | I    | L    | ECR | Number of roots found by BETTER |
| EPSRT                    | R    | EPSRT | D   | A small number used to generate two of the three approximations when given one |
| ITER                     | I    | I    | C   | Counter for number of iterations |
TABLE C.IV (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>B</td>
<td>C</td>
<td>UB, VB</td>
<td>D</td>
<td></td>
<td>Array containing coefficients of deflated polynomial</td>
</tr>
<tr>
<td>X1</td>
<td>C</td>
<td>UX1, VX1</td>
<td>D</td>
<td></td>
<td>One of three approximations to the root</td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>UX2, VX2</td>
<td>D</td>
<td></td>
<td>One of three approximations to the root</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3, VX3</td>
<td>D</td>
<td></td>
<td>One of three approximations to the root</td>
</tr>
<tr>
<td>PX1</td>
<td>C</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></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></td>
<td>Value of polynomial (P(X)) at (X_3)</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td></td>
<td>CONV = true implies convergence has been obtained</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4, VX4</td>
<td>D</td>
<td></td>
<td>Newest approximation to root</td>
</tr>
<tr>
<td>J</td>
<td>I</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>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td>C</td>
<td>Maximum number of iterations permitted</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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>X2R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>X2I</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>

Subroutine QUAD

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPST</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>

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.1. (Continued)
Figure C.1. (Continued)
Figure C.1. (Continued)
TABLE C.V.

PROGRAM FOR MULLER'S METHOD

C

C ************************************************************
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 ZERO OF THE QUADRATIC
C * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
C * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
C
C ************************************************************
C
C 0001 DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,URD0T,VRD0T,U1,VK1,UAPP,VAPP
C 0025 1,UX2,UX2,UXWORK,VWORK,VWORK,UX3,UX3,UXV3,UXV3,UXE,UXE,VAY,UPX1,UPX1,UAPP,VAPP
C 0255 URAPP,UPX4,VPX4,EPSTEPS,EPSON,EPSON,CCCV,EPSON,URH3,URH3,URH4,URH4,URV4,URV4,URB4,URB4,URP4,URP4
C 0300 \quad \text{DIMENSION URD0T(25)}, VRD0T(25), MULT(25), UAPP(25), VRAPP(25, 3), UWORK
C 0301 \quad \text{1(25)}, VWORK(26), URB(26), URB(26), VA(26), VA(26), VRAPP(25, 3), VRAPP(25, 3)
C 003 DATA PNAME, DNAME, 2HP, 1MDOT/
C 0004 LOGICAL CONV
C 0005 COMMON EPSRT, EPSD, EPS, IO2, MAX
C 0006 101 = 5
C 0007 102 = 6
C 0008 EPSRT = 0.999
C 0009 10 NACOT = 0
C 0010 1 RGOOT = 0
C 0011 1 PATH = 1
C 0012 1 NOSGT = 0
C 0013 1 NALT = 0
C 0014 1 ITIME = 0
C 0015 1 IAPP = 1
C 0016 1 ITER = 1
C 0017 1 READ(0), 1000) NOPLSY, NPAPP, MAXEPS, EPS, EPSD, EPSX, XSTART, XEND, XCHECK
C 0018 1 IF(IXCH, 1, 11) STOP
C 0019 1 XXK = NP = 1
C 0020 READ(0100) (XAI(1), VA(1), I, 1, KKK)
C 0021 WRITE(102, 1020) NOPLSY, NP
C 0022 WRITE(102, 1035) PHASE, XAI(1), VA(1), I, 1, KKK
C 0023 WRITE(102, 2060)
C 0024 WRITE(102, 2000) NAPP
C 0025 WRITE(102, 2010) MAX
C 0026 WRITE(102, 2001) EPS
C 0027 WRITE(102, 2030) EPSM
C 0028 WRITE(102, 2040) XSTART
C 0029 WRITE(102, 2050) XEND
C 0030 IF(IXPG, 0.2) GO TO 15
C 0031 CALL QUADD(UX1, VX1, VROOT, VRD0T, NAROOT, MULT, EPSM)
C 0032 WRITE(102, 1037)
C 0033 WRITE(102, 1096) (1, UROOTT, VRD0T, MULT, I, 1, NAROOT)
C 0034 GO TO 10
C 0035 15 IF(INAPP, 0, 1) GO TO 20
C 0036 IF(NAPP = NP)
C 0037 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
C 0038 GO TO 27
C 0039 20 READ(01030) (UAPP(1, Z), VAPP(1, Z, 1, 1, NAPP)
C 0040 GO 25 = 1, NAPP
C 0041 UAPP(1, 1) = 0.4*VAPP(1, Z)
C 0042 VAPP(1, 1) = 0.4*VAPP(1, Z)
TABLE C.V (Continued)

```plaintext
0043  UAPP(I,3)=1.0*UAPP(I,2)
0044  VAPP(I,3)=1.0*VAPP(I,2)
0045  KKK=NP+1
0046  DO 30 I=1,KKK
0047  UWORK(I)=UAPP(I,1)
0048  VWORK(I)=VAPP(I,1)
0049  30 CONTINUE
0050  UAPP(I1),VAPP(I1)=0.0
0051  VX1=VAPP(I1,1)
0052  UX3=UAPP(IAPP,3)
0053  CALL HORNER(UWORK,VWORK,UX1,VX1)
0054  CALL HORNER(UWORK,VWORK,UX2,VX2)
0055  CALL HORNER(UWORK,VWORK,UX3,VX3)
0056  CALL HORTER(UWORK,VWORK,UX4,VX4)
0057  ABPX4=DSQRT(UX4**2+VX4**2)
0058  ABPX3=DSQRT(UX3**2+VX3**2)
0059  IF(ABPX3.EQ.0.0) GO TO 70
0060  QQQ=ABPX4/ABPX3
0061  IF(QQQ.LE.1.0) GO TO 70
0062  UQ4=0.5*UQ4
0063  VQ4=0.5*VQ4
0064  UX4=UX3+(UH3*UQ4-VH3*VQ4)
0065  VX4=VX3+(VH3*UQ4-UH3*VQ4)
0066  GO TO 60
0067  70 CALL TEST(UX3,VX3,UX4,VX4,CONV1)
0068  IF(CONV1) GO TO 120
0069  IF(ITER.LE.MAX) GO TO 110
0070  CALL ALTER(UAPP,VAPP,NAPP,XSTART,IAPP)
0071  IF(NALTER.GT.51) GO TO 75
0072  ITER=1
0073  GO TO 40
0074  75 IF(IAPP.LT.NAPP) GO TO 100
0075  IF(XEND.EQ.0.0) GO TO 77
0076  IF(XSTART.EQ.0.0) GO TO 77
0077  NAPP=NP
0078  CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0079  IAPP=0
0080  GO TO 100
0081  77 WRITE(IO2,1090)
0082  WRITE(IO2,10351) (ONAME,J,UWORK(J),J=1,KKK)
0083  80 IF(NROOT.EQ.0) GO TO 90
0084  IF(IROOT.EQ.0) GO TO 90
0085  WRITE(IO2,120)
0086  WRITE(IO2,1200)
0087  WRITE(IO2,1201)
0088  WRITE(IO2,1202)
0089  WRITE(IO2,1203)
0090  WRITE(IO2,1204)
0091  WRITE(IO2,1205)
0092  WRITE(IO2,1206)
0093  WRITE(IO2,1207)
0094  WRITE(IO2,1208)
0095  WRITE(IO2,1209)
0096  WRITE(IO2,1210)
0097  WRITE(IO2,1211)
0098  WRITE(IO2,1212)
```

TABLE C.V. (Continued)

0099 IF(IROOT.LT.NROOT) GO TO 85
0100 GO TO 87
0101 85 KKK=IROOT+1
0102 WRITE(102,1036) (I,UROOT(I),VROOT(I),MULT(I),KKK,NROOT)
0103 87 IF(IPATH.LE.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 UROOT(NROOT)=UX4
0116 VROOT(NROOT)=VX4
0117 URAPP(NROOT,1)=UAPP(IAPP,1)
0118 VRAPP(NROOT,1)=VAPP(IAPP,1)
0119 URAPP(NROOT,2)=UAPP(IAPP,2)
0120 VRAPP(NROOT,2)=VAPP(IAPP,2)
0121 URAPP(NROOT,3)=UAPP(IAPP,3)
0122 VRAPP(NROOT,3)=VAPP(IAPP,3)
0123 125 IF(NOMULT.LT.NP) GO TO 130
0124 GO TO 80
0125 130 CALL HORNRE(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0126 NWORK=NWORK-1
0127 KKK=NWORK+1
0128 GO TO 140 I=1,KKK
0129 UWORK(I)=UB(I)
0130 VWORK(I)=VB(I)
0131 CALL HORNRE(UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)
0132 CALL QUAD(UWORK,VWORK,NWORK,UROOT,VROOT,NROOT,MULT,EPSEM)
0133 GO TO 150
0134 IF(NWORK.GT.2) GO TO 75
0135 IROOT=NROOT
0136 CALL QUAD(UWORK,VWORK,NWORK,UROOT,VROOT,NROOT,MULT,EPSEM)
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(1H14,1X,52H MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOM...
TABLE C.V (Continued)

<table>
<thead>
<tr>
<th>Polynomial Number</th>
<th>Coefficients of Deflated Polynomial for Which No Zeros Were Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td>(0.23 \times 10^{-3} + 0.23 \times 10^{-2} + 0.23 \times 10^{-1} + 0.23 )</td>
</tr>
</tbody>
</table>

The coefficients of \(P(x)\) are:

<table>
<thead>
<tr>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.23 \times 10^{-3} + 0.23 \times 10^{-2} + 0.23 \times 10^{-1} + 0.23 )</td>
</tr>
</tbody>
</table>

Before attempts to improve accuracy:

<table>
<thead>
<tr>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.23 \times 10^{-3} + 0.23 \times 10^{-2} + 0.23 \times 10^{-1} + 0.23</td>
</tr>
</tbody>
</table>

After the attempts to improve accuracy:

<table>
<thead>
<tr>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.23 \times 10^{-3} + 0.23 \times 10^{-2} + 0.23 \times 10^{-1} + 0.23 )</td>
</tr>
</tbody>
</table>

Number of initial approximations given:

<table>
<thead>
<tr>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.23 \times 10^{-3} + 0.23 \times 10^{-2} + 0.23 \times 10^{-1} + 0.23</td>
</tr>
</tbody>
</table>

Maximum number of iterations:

<table>
<thead>
<tr>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.23 \times 10^{-3} + 0.23 \times 10^{-2} + 0.23 \times 10^{-1} + 0.23 )</td>
</tr>
</tbody>
</table>

Test for convergence:

<table>
<thead>
<tr>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.23 \times 10^{-3} + 0.23 \times 10^{-2} + 0.23 \times 10^{-1} + 0.23 )</td>
</tr>
</tbody>
</table>

Radius to start search:

<table>
<thead>
<tr>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.23 \times 10^{-3} + 0.23 \times 10^{-2} + 0.23 \times 10^{-1} + 0.23 )</td>
</tr>
</tbody>
</table>

Radius to end search:

<table>
<thead>
<tr>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.23 \times 10^{-3} + 0.23 \times 10^{-2} + 0.23 \times 10^{-1} + 0.23 )</td>
</tr>
</tbody>
</table>
**TABLE C.V (Continued)**

0001 SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,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 **************************************************************

0002 DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA

0003 COMMON EPS1, EPS2, EPS3, IO2, MAX

0004 IF (ITIME .NE. 0) GO TO 5

0005 ITIME = 1

0006 WRITE(IO2,1100) MAX

0007 IF (NALTER .EQ. 0) GO TO 10

0008 WRITE(IO2,1000) X1R, X1I, X2R, X2I, X3R, X3I

0009 GO TO 20

0010 NALTER = NALTER + 1

0011 IF (NALTER .GT. 5) RETURN

0012 GO TO (30, 40, 30, 40, 30, 40)

0013 X2R = -X2R

0014 X2I = -X2I

0015 GO TO 50

0016 X1R = 0.9*X2R

0017 X1I = 0.9*X2I

0018 RETURN

0019 1000 FORMAT(1X,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF

0020 ITER , 13,12H ITERATIONS.//)

0021 1020 FORMAT(///IX,54HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF

0022 ITER , 13,12H ITERATIONS.//)

0023 END
**TABLE C.V (Continued)**

```
0001 SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)
  **********************************************************************
  C  *
  C * SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE *
  C * DEGREE OF THE ORIGINAL POLYNOMIAL.
  C  *
                                         **********************************************************************
0002 DOUBLE PRECISION APPR,APPI,XSTART,EPS1,EPS2,EPS3,BETA
0003 DIMENSION APPR(25,3),APPI(25,3)
0004 COMMON EPS1,EPS2,EPS3,IX2,MAX
0005 IF (XSTART.EQ.0.0) XSTART=0.5
0006 BETA=0.2817994
0007 DO 10 I=1,NAPP
0008 APPR(1,2)=XSTART*COS(BETA)
0009 APPI(1,2)=XSTART*SIN(BETA)
0010 BETA=BETA+.5235988
0011 10 XSTART=XSTART+.5
0012 DO 20 I=1,NAPP
0013 APPR(1,1)=0.9*APPR(1,2)
0014 APPI(1,1)=0.9*APPI(1,2)
0015 APPR(1,3)=1.1*APPR(1,2)
0016 20 APPI(1,3)=1.1*APPI(1,2)
0017 RETURN
0018 END
```
TABLE C.V (Continued)

0001 SUBROUTINE BETTER, U, NP, UROOT, VR0OT, NROOT, URAPP, VRAPP, IROOT, MUL

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

0002 DOUBLE PRECISION UROOT, VR0OT, U, VA, URAPP, VRAPP, UX1, UX2, UX3,


0003 LOGICAL CONV

0004 DIMENSION UROOT(25), VR0OT(25), U, VA(26), URAPP(25, 3), VRAPP(25, 3)

0005 COMMON EPSRT, EPSOEPS, IOZ, MAX

0006 IF (IROOT.LE.1) RETURN

0007 L=0

0008 DO 10 I=1, NROOT

0009 UBAPP(I, 1)=UROOT(I)*EPSRT

0010 VBAPP(I, 1)=VR0OT(I)*EPSRT

0011 UBAPP(I, 2)=UROOT(I)

0012 VBAPP(I, 2)=VR0OT(I)

0013 UBAPP(I, 3)=UROOT(I)*(2.0-EPSRT)

0014 VBAPP(I, 3)=VR0OT(I)*(2.0-EPSRT)

0015 DO 100 J=1, NROOT

0016 UX1=UBAPP(J, 1)

0017 VX1=VBAPP(J, 1)

0018 UX2=UBAPP(J, 2)

0019 VX2=VBAPP(J, 2)

0020 UX3=UBAPP(J, 3)

0021 VX3=VBAPP(J, 3)

0022 ITER=1

0023 CALL HORNER(NP, U, VA, UX1, UX2, UX3, VB, UX1, VX1, VX2, VX3)

0024 CALL HORNER(NP, U, VA, UX1,UX2, UX3, VB, UX1, VX1, VX2, VX3)

0025 CALL HORNER(NP, U, VA, UX1,UX2, UX3, VB, UX1, VX1, VX2, VX3)

0026 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UX1, VPX1, VPX2, VPX3, UX3)

0027 CALL TEST(UX3, VX3, UX4, VX4, CONV)

0028 IF (CONV) GO TO 50

0029 IF (ITER.LT.MAX) GO TO 40

0030 WRITE(IO2, 100) J, UROOT(J), VR0OT(J)

0031 WRITE(IO2, 1010) UX4, VX4

0032 IF (J.EQ.1) IROOT=J

0033 IF (J.EQ.IROOT) GO TO 35

0034 GO TO 10

0035 KK=IROOT-1

0036 DO 34 K=J, KK

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 UX1=UX2
TABLE C.V (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0044</td>
<td>( \text{UX}1 = \text{UX}2 )</td>
</tr>
<tr>
<td>0047</td>
<td>( \text{UX}2 = \text{UX}3 )</td>
</tr>
<tr>
<td>0049</td>
<td>( \text{UX}3 = \text{UX}4 )</td>
</tr>
<tr>
<td>0050</td>
<td>( \text{UX}4 = \text{UX}2 )</td>
</tr>
<tr>
<td>0052</td>
<td>( \text{VX}1 = \text{VX}2 )</td>
</tr>
<tr>
<td>0053</td>
<td>( \text{VX}2 = \text{VX}3 )</td>
</tr>
<tr>
<td>0054</td>
<td>( \text{VX}3 = \text{VX}4 )</td>
</tr>
<tr>
<td>0055</td>
<td>( \text{UX}4 = \text{UX}5 )</td>
</tr>
<tr>
<td>0056</td>
<td>( \text{VX}4 = \text{VX}5 )</td>
</tr>
<tr>
<td>0057</td>
<td>( \text{UX}5 = \text{UX}1 )</td>
</tr>
<tr>
<td>0058</td>
<td>( \text{VX}5 = \text{VX}1 )</td>
</tr>
<tr>
<td>0059</td>
<td>( \text{UX}1 = \text{UX}2 )</td>
</tr>
<tr>
<td>0060</td>
<td>( \text{VX}1 = \text{VX}2 )</td>
</tr>
<tr>
<td>0061</td>
<td>( \text{UX}2 = \text{UX}3 )</td>
</tr>
<tr>
<td>0062</td>
<td>( \text{VX}2 = \text{VX}3 )</td>
</tr>
<tr>
<td>0063</td>
<td>( \text{UX}3 = \text{UX}4 )</td>
</tr>
<tr>
<td>0064</td>
<td>( \text{VX}3 = \text{VX}4 )</td>
</tr>
<tr>
<td>0065</td>
<td>( \text{UX}4 = \text{UX}1 )</td>
</tr>
<tr>
<td>0066</td>
<td>( \text{VX}4 = \text{VX}1 )</td>
</tr>
<tr>
<td>0067</td>
<td>( \text{UX}1 = \text{UX}2 )</td>
</tr>
<tr>
<td>0068</td>
<td>( \text{VX}1 = \text{VX}2 )</td>
</tr>
<tr>
<td>0069</td>
<td>( \text{UX}2 = \text{UX}3 )</td>
</tr>
<tr>
<td>0070</td>
<td>( \text{VX}2 = \text{VX}3 )</td>
</tr>
<tr>
<td>0071</td>
<td>( \text{UX}3 = \text{UX}4 )</td>
</tr>
<tr>
<td>0072</td>
<td>( \text{VX}3 = \text{VX}4 )</td>
</tr>
<tr>
<td>0073</td>
<td>( \text{UX}4 = \text{UX}1 )</td>
</tr>
<tr>
<td>0074</td>
<td>( \text{VX}4 = \text{VX}1 )</td>
</tr>
<tr>
<td>0075</td>
<td>( \text{UX}1 = \text{UX}2 )</td>
</tr>
<tr>
<td>0076</td>
<td>( \text{VX}1 = \text{VX}2 )</td>
</tr>
<tr>
<td>0077</td>
<td>( \text{UX}2 = \text{UX}3 )</td>
</tr>
<tr>
<td>0078</td>
<td>( \text{VX}2 = \text{VX}3 )</td>
</tr>
<tr>
<td>0079</td>
<td>( \text{UX}3 = \text{UX}4 )</td>
</tr>
<tr>
<td>0080</td>
<td>( \text{VX}3 = \text{VX}4 )</td>
</tr>
<tr>
<td>0081</td>
<td>( \text{UX}4 = \text{UX}1 )</td>
</tr>
<tr>
<td>0082</td>
<td>( \text{VX}4 = \text{VX}1 )</td>
</tr>
</tbody>
</table>

**Table Notes:**

- In the attempt to improve accuracy, root 1.3244 did not converge after 13 iterations.
- The present approximation is 1.323163 + 1023.1624/24.
TABLE C.V (Continued)

<table>
<thead>
<tr>
<th>0001</th>
<th>SUBROUTINE CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, UPX2, UPX3, V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>************************************************************************</td>
</tr>
</tbody>
</table>
|      | * GIVEN THREE APPROXIMATIONS XN-21, XN-11, AND XN1, SUBROUTINE CALC *
|      | * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF *
|      | * THE QUADRATIC CLOSEST TO XN1. THIS ZERO IS THE NEW APPROXIMATION *
|      | * XN+1 TO THE ZERO OF THE POLYNOMIAL. *
|      | ************************************************************************|
| 0002 | DOUBLE PRECISION ARG1, ARG2 |
| 0003 | DOUBLE PRECISION UPX3, VPX3, UX3, VX3, UPX1, UPX2, VPX2, UX2, VX2, UX1, |
|      | VPX1, UX3, VX3, UX2, VX2, VX1, UPX1, |
| 0004 | COMMON EPSRT, EPSO, EPS, IQ2, MAX |
| 0005 | UH3=UX3-UX2 |
| 0006 | VM3=VX3-VX2 |
| 0007 | UM2=UX2-UX1 |
| 0008 | VH2=VX2-VX1 |
| 0009 | QB=UX2*UH2+VH2*VH2 |
| 0010 | VG3=(UH3*UH3+VH3*VH3)/688 |
| 0011 | VQ3=(UH3*UH3+VH3*VH3)/688 |
| 0012 | VD=VPX3-(UDDO*UPX2+UDDO*VXX2+VQ3*UPX3+VQ3*VXX2) |
| 0013 | V0A=2.0*UQ3 |
| 0014 | VAAA=UAAA+1.0 |
| 0015 | UBB=VDDD-UDDD*VDDD |
| 0016 | VBB=VDDD-UDDD*VDDD |
| 0017 | UCCC=UQ3*VQ3*VDDD |
| 0018 | VCCC=VQ3*VQ3*VDDD |
| 0019 | US=(UAAA*UPX3+UAAA*VPX3)-(UAAA*VPX3-UAAA*UPX3)+(UCCC*UPX3-UCCC*}| |
| 0020 | VPX3) |
| 0021 | VDDD=UDDD-UDDD*VDDD |
| 0022 | VDDD=UDDD-UDDD*VDDD |
| 0023 | VDDD=UDDD-UDDD*VDDD |
| 0024 | VDDD=UDDD-UDDD*VDDD |
| 0025 | VDDD=UDDD-UDDD*VDDD |
| 0026 | UDDD=UDDD-UDDD*VDDD |
| 0027 | VDDD=UDDD-UDDD*VDDD |
| 0028 | VDDD=UDDD-UDDD*VDDD |
| 0029 | VDDD=UDDD-UDDD*VDDD |
| 0030 | AAA=AAA+VDDD |
| 0031 | IF(AAA.EQ.0.0) GO TO 5 |
| 0032 | GO TO 7 |
| 0033 | 5 THE1=0.0 |
| 0034 | GO TO 9 |
| 0035 | 9 THE1=THE1+2.0 |
| 0036 | THE1=THE1+2.0 |
| 0037 | RAD=DOT(RAD) |
| 0038 | UTET=UTET+DOT(UDDO) |
| 0039 | THE1=THE1+2.0 |
| 0040 | VL1=VL1+VDDD |
| 0041 | VDD2=VDDD |
| 0042 | VDD2=VDDD |
| 0043 | VDD2=VDDD |
| 0044 | VDD2=VDDD |
TABLE C.V (Continued)

0045  \texttt{\textasciitilde ARG} = \texttt{UOEN2*UDEN2+VDEN2*VDEN2}
0046  \texttt{AAA} = \texttt{DSQRT2}(	exttt{ARG1})
0047  \texttt{BBB} = \texttt{DSQRT2}(	exttt{ARG2})
0048  \texttt{IF} \texttt{(AAA.LT.BBB)} \texttt{GO TO} 10
0049  \texttt{IF} \texttt{(AAA.EQ.0.0)} \texttt{GO TO} 60
0050  \texttt{UAAA} = -2.0*\texttt{UC}
0051  \texttt{VAAA} = -2.0*\texttt{VC}
0052  \texttt{UQ4} = \texttt{(UAAA*UDEN1+VAAA*VDEN1)/ARG1}
0053  \texttt{VQ4} = \texttt{(VAAA*UDEN1-UAAA*VDEN1)/ARG1}
0054  \texttt{GO TO} 50
0055  \texttt{10 IF} \texttt{(BBB.EQ.0.01} \texttt{GO TO} 60
0056  \texttt{UAAA} = -2.0*\texttt{UC}
0057  \texttt{VAAA} = -2.0*\texttt{VC}
0058  \texttt{UQ4} = \texttt{(UAAA*UDEN2+VAAA*VDEN2)/ARG2}
0059  \texttt{VQ4} = \texttt{(VAAA*UDEN2-UAAA*VDEN2)/ARG2}
0060  \texttt{GO TO} 50
0061  \texttt{50 UX4} = \texttt{UX3+(UH3*UQ4-VH3*VQ4)}
0062  \texttt{VX4} = \texttt{VX3+(VH3*UQ4+UH3*VQ4)}
0063  \texttt{RETURN}
0064  \texttt{60 UQ4 = 1.0}
0065  \texttt{VQ4} = 0.0
0066  \texttt{GO TO} 50
0067  \texttt{END}
TABLE C.V (Continued)

0001 SUBROUTINE TEST(U3, UX, V4, VX, VX, VN, VN, VN)
  C *****************************************************************
  C C SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
  C C IMATIONS BY TESTING THE EXPRESSION
  C C ABSOLUTE VALUE OF X(N+1)-X(N))/ABSOLUTE VALUE OF X(N+1).
  C C WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
  C C *****************************************************************
  DOUBLE PRECISION UX, UX, UX, VX, VX, VX, VX, VX, VX, VX
  IDENOM LOGICAL CONV
  COMMON EPSRT, EPSG, EPS, 102, MAX
  UDDUMMY=UX-UX
  VDDUMMY=VX-UX3
  AAA=DSORT(UDDUMMY+VDDUMMY+VDDUMMY+VDDUMMY)
  DENOM=DSORT(UX4+UX+UX+UX4)
  IF(DENOM.LT.EPS0) GO TO 20
  IF(AAA/GDENOM,L,T,EPS) GO TO 10
  5 CONV=.FALSE.
  GO TO 100
  10 CONV=.TRUE.
  GO TO 100
  20 IF(AAA,L,T,EPS0) GO TO 10
  GO TO 5
  100 RETURN
  END

0001 SUBROUTINE HORNERTHMA, VA, UX, UY, UY, UB, UPX, VPX)
  C *****************************************************************
  C C HORNERT'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL PI AT A POINT D.
  C C SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE
  C C FACTOR (X-D).
  C C *****************************************************************
  DOUBLE PRECISION UX, VX, UPX, VPX, UY, UY, UA, VA
  DIMENSION UX(265), VX(265), UB(265), VB(265)
  UY(1)=UA(1)
  VB(1)=VA(1)
  NUM=UA+1
  GO TO 12-2.NUM
  UB(1)=UA(1)*(UB(1-1)+UX-VB(1-1)*VX)
  GO TO 12-2.NUM
  UPX=UB(NUM)
  VPX=VP(NUM)
  RETURN
  END
**TABLE C.V (Continued)**

```fortran
0001 SUBROUTINE QUAD(UA, VA, UAROOT, VRDOT, NRDOT, MULTI, EPST)

***********************************************************************
0002 DOUBLE PRECISION UA, VA, UROOT, VROOT, BBB, UAAA, VAAA, UDISC, VDISC, UDDUMY

0003 DIMENSION UA(251), VA(251), UROOT(251), VROOT(251), MULTI(251)

0004 IF(INA.EQ.21) GO TO 7
0005 IF(INA.EQ.11) GO TO 5
0006 UROOT(NROOT+1) = 0.0
0007 VROOT(NROOT+1) = 0.0
0008 MULTI(NROOT+1) = 1
0009 NROOT = NROOT + 1
0010 GO TO 50

0011 BBB = UA(1)*UA(1) + VA(1)*VA(1)
0012 UROOT(NROOT+1) = -(UA(1)*UA(1) - VA(1)*VA(1))/BBBB
0013 VROOT(NROOT+1) = -(VA(1)*UA(1) + UA(1)*VA(1))/BBBB
0014 MULTI(NROOT+1) = 1
0015 NROOT = NROOT + 1
0016 GO TO 50

0017 UDISC = UA(2)*UA(1) - VA(2)*VA(1)
0018 VDISC = VA(2)*UA(1) - VA(1)*UA(2)
0019 BBB = DSQRD(UDISC*UDISC + VDISC*VDISC)
0020 IF(BBB.LT.EPST) GO TO 10
0021 CALL COMSOT(UDISC, VDISC, UDDUMY, VDDUMY)
0022 UBBB = -(UA(1)*UA(1) + VAAA*VAAA)
0023 VBBB = -(VA(1)*VA(1) + UAAA*UAAA)
0024 UDDD = -(UA(1)*UA(1) + VAAA*VAAA)
0025 VDDD = -(VA(1)*VA(1) + UAAA*UAAA)
0026 UAAA = 2.0*UA(1)
0027 VAAA = 2.0*VA(1)
0028 BBB = UAAA*UAAA + VAAA*VAAA
0029 UROOT(NROOT+1) = -(UAAA + BBB/UBBB*VAAA)
0030 VROOT(NROOT+1) = -(VAAA + BBB/UAAA*VBBB)
0031 UROOT(NROOT+2) = -(UDDD + BBB/UBBB*VDDD)
0032 VROOT(NROOT+2) = -(VDDD + BBB/VAAA*VDDD)
0033 MULTI(NROOT+1) = 1
0034 MULTI(NROOT+2) = 1
0035 NROOT = NROOT + 2
0036 GO TO 50

0037 10 UAAA = 2.0*UA(1)
0038 VAAA = 2.0*VA(1)
0039 BBB = UAAA*UAAA + VAAA*VAAA
0040 UROOT(NROOT+1) = -(UA(1)*UA(1) + VAAA*VAAA)/BBBB
0041 VROOT(NROOT+1) = -(VA(1)*VA(1) + UAAA*UAAA)/BBBB
0042 MULTI(NROOT+1) = 2
0043 NROOT = NROOT + 1
0044 50 RETURN

END
```

---

**TABLE C.V (Continued)**

0001 SUBROUTINE QUAD(UA, VA, UAROOT, VRDOT, NRDOT, MULTI, EPST)

***********************************************************************
0002 DOUBLE PRECISION UA, VA, UROOT, VROOT, BBB, UAAA, VAAA, UDISC, VDISC, UDDUMY

0003 DIMENSION UA(251), VA(251), UROOT(251), VROOT(251), MULTI(251)

0004 IF(INA.EQ.21) GO TO 7
0005 IF(INA.EQ.11) GO TO 5
0006 UROOT(NROOT+1) = 0.0
0007 VROOT(NROOT+1) = 0.0
0008 MULTI(NROOT+1) = 1
0009 NROOT = NROOT + 1
0010 GO TO 50

0011 BBB = UA(1)*UA(1) + VA(1)*VA(1)
0012 UROOT(NROOT+1) = -(UA(1)*UA(1) - VA(1)*VA(1))/BBBB
0013 VROOT(NROOT+1) = -(VA(1)*UA(1) + UA(1)*VA(1))/BBBB
0014 MULTI(NROOT+1) = 1
0015 NROOT = NROOT + 1
0016 GO TO 50

0017 UDISC = UA(2)*UA(1) - VA(2)*VA(1)
0018 VDISC = VA(2)*UA(1) - VA(1)*UA(2)
0019 BBB = DSQRD(UDISC*UDISC + VDISC*VDISC)
0020 IF(BBB.LT.EPST) GO TO 10
0021 CALL COMSOT(UDISC, VDISC, UDDUMY, VDDUMY)
0022 UBBB = -(UA(1)*UA(1) + VAAA*VAAA)
0023 VBBB = -(VA(1)*VA(1) + UAAA*UAAA)
0024 UDDD = -(UA(1)*UA(1) + VAAA*VAAA)
0025 VDDD = -(VA(1)*VA(1) + UAAA*UAAA)
0026 UAAA = 2.0*UA(1)
0027 VAAA = 2.0*VA(1)
0028 BBB = UAAA*UAAA + VAAA*VAAA
0029 UROOT(NROOT+1) = -(UAAA + BBB/UBBB*VAAA)
0030 VROOT(NROOT+1) = -(VAAA + BBB/UAAA*VBBB)
0031 UROOT(NROOT+2) = -(UDDD + BBB/UBBB*VDDD)
0032 VROOT(NROOT+2) = -(VDDD + BBB/VAAA*VDDD)
0033 MULTI(NROOT+1) = 1
0034 MULTI(NROOT+2) = 1
0035 NROOT = NROOT + 2
0036 GO TO 50

0037 10 UAAA = 2.0*UA(1)
0038 VAAA = 2.0*VA(1)
0039 BBB = UAAA*UAAA + VAAA*VAAA
0040 UROOT(NROOT+1) = -(UA(1)*UA(1) + VAAA*VAAA)/BBBB
0041 VROOT(NROOT+1) = -(VA(1)*VA(1) + UAAA*UAAA)/BBBB
0042 MULTI(NROOT+1) = 2
0043 NROOT = NROOT + 1
0044 50 RETURN

END
### TABLE C.V (Continued)

```plaintext
0001 SUBROUTINE COMSQT(UK,VX,UY,VY) 
C ************************************************************************** 
C 0002 DOUBLE PRECISION UXVX,UYVVYeDUMMY,R,AAA,BBB 
C 0003 R=DSQRT(UK/U+VX*VX) 
C 0004 AAA=DSQRT(DABS((R-U+VX)/2.0)) 
C 0005 BBB=DSQRT(DABS((R-U-X)/2.0)) 
C 0006 IF(UX) 10,20,30 
C 0007 10 UY=AAA 
C 0008 20 IF(UX) 40,50,60 
C 0010 30 UY=AAA 
C 0011 40 DUMMY=DABS(UX) 
C 0012 50 UY=0.0 
C 0013 60 DUMMY=DABS(UX) 
C 0014 70 UY=0.0 
C 0015 80 GO TO 100 
C 0016 90 GO TO 100 
C 0017 100 RETURN 
C 0018 END 
```

---

**TABLE C.V (Continued)**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE COMSQT(UK,VX,UY,VY)</td>
</tr>
<tr>
<td>0002</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0003</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0004</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0005</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0006</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0007</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0008</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0009</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0010</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0011</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0012</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0013</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0014</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0015</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0016</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0017</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0018</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0019</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0020</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0021</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0022</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0023</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0024</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0025</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0026</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0027</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0028</td>
<td>C**************************************************************************</td>
</tr>
<tr>
<td>0029</td>
<td>C**************************************************************************</td>
</tr>
</tbody>
</table>
APPENDIX D

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

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

1. Generating Approximations

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

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

where

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

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

*These illustrations are representative of G.C.D.-Newton's method in double precision. Control cards for other methods should be prepared accordingly.
(7-8) may be left blank. If XSTART is left blank, it is interpreted as 0.

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

```
Variable Name   Card Columns
1 2 4 5 7 8 6 7 7 7 8
N O P N N A P P
P L Y
1 7
```

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: ($j = 1, 3$)

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

where

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

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

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

where

$$\beta = \tan^{-1} \left( \frac{\text{Im} X_0}{\text{Re} X_0} + K \frac{\pi}{3} \right) ; K = 1 \text{ if } j = 1 \quad K = 2 \text{ if } j = 3.$$
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 + .5N.

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

Example D.4
Figure A.3 shows the distribution of initial and altered approximations for an annulus of width 2 and inner radius a.

4. Improving Zeros Found

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

5. Solving Quadratic Polynomial

After N-2 roots of an N\textsuperscript{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>N</th>
<th>O</th>
<th>P</th>
<th>N</th>
<th>A</th>
<th>XSTART</th>
<th>XEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example D.5
Note that if several roots are known to the user, they may be "divided out" of the original polynomial by using this procedure.

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

<table>
<thead>
<tr>
<th>N</th>
<th>O</th>
<th>P</th>
<th>L</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>2</td>
<td>XSTART</td>
<td>XEND</td>
</tr>
</tbody>
</table>

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 E.I

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

Main Program

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

Subroutine GENAPP

APPR(N), APPI(N)

Subroutine GCD

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

Subroutine QUAD

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

Subroutine NEWTON

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

Subroutine DIVIDE

UP(N+1), VP(N+1)
UD(N+1), VD(N+1)
UQ(N+1), VQ(N+1)
TABLE E.I (Continued)

Subroutine HORNER

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

Subroutine DERIV

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

Subroutine MULTI

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

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

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

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

An end card follows the entire collection of data sets. It indicates that there is no more data to follow and terminates execution of the
program. This information is displayed in Figure E.1 and described below. All data should be right justified and the D-type specification should be used. The recommendations given in Table E.II are those found to give best results on the IBM 360/50 computer which has a 32 bit word.

**Control Information**

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

**TABLE E.II**

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

<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>EPSL</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>
<tr>
<td>Variable Name</td>
<td>Card Columns</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<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
### 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></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N</th>
<th>O</th>
<th>F</th>
<th>R</th>
<th>N</th>
<th>A</th>
<th>P</th>
<th>EPS1</th>
<th>EPS2</th>
<th>EPS3</th>
<th>EPS4</th>
<th>XSTART</th>
<th>XEND</th>
<th>KCHECK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>7</td>
<td></td>
<td>200</td>
<td>1.0D-03</td>
<td>1.0D-10</td>
<td>1.0D-20</td>
<td>1.0D-02</td>
<td>1.0D+01</td>
<td>5.0D+02</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

### 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></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UP</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0.125768D+01</td>
<td>-0.37225D+02</td>
</tr>
</tbody>
</table>

Example
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</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>J</td>
<td>I</td>
<td>J</td>
<td>I</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
</tr>
<tr>
<td>NOPOLY</td>
<td>I</td>
<td>NOPOLY</td>
<td>I</td>
</tr>
<tr>
<td>NF</td>
<td>I</td>
<td>NF</td>
<td>I</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>U,P,VP</td>
<td>D</td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
</tr>
<tr>
<td>EPS1</td>
<td>R</td>
<td>EPS1</td>
<td>D</td>
</tr>
<tr>
<td>EPS2</td>
<td>R</td>
<td>EPS2</td>
<td>D</td>
</tr>
<tr>
<td>EPS3</td>
<td>R</td>
<td>EPS3</td>
<td>D</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
</tr>
<tr>
<td>KCHECK</td>
<td>I</td>
<td>KCHECK</td>
<td>I</td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>UAPP,VAAPP</td>
<td>D</td>
</tr>
<tr>
<td>XSTART</td>
<td>R</td>
<td>XSTART</td>
<td>D</td>
</tr>
<tr>
<td>XEND</td>
<td>R</td>
<td>XEND</td>
<td>D</td>
</tr>
<tr>
<td>ANAME</td>
<td>A</td>
<td>ANAME</td>
<td>A</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT,VRROOT</td>
<td>D</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
</tr>
<tr>
<td>DP</td>
<td>C</td>
<td>UDP,VDP</td>
<td>D</td>
</tr>
<tr>
<td>NDP</td>
<td>I</td>
<td>NDP</td>
<td>I</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>UD,VD</td>
<td>D</td>
</tr>
<tr>
<td>ND</td>
<td>I</td>
<td>ND</td>
<td>I</td>
</tr>
<tr>
<td>Q</td>
<td>C</td>
<td>UQ,VQ</td>
<td>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></td>
<td>Degree of quotient polynomial Q(X)</td>
</tr>
<tr>
<td>ZRO</td>
<td>C</td>
<td>UZRO, VZRO</td>
<td>D</td>
<td></td>
<td>Value at which to evaluate or deflate polynomial</td>
</tr>
<tr>
<td>DUMMY</td>
<td>C</td>
<td>UDUMMY, VDUMMY</td>
<td>D</td>
<td></td>
<td>Dummy variable</td>
</tr>
<tr>
<td>QQ</td>
<td>C</td>
<td>UQQ, VQQ</td>
<td>D</td>
<td></td>
<td>Working array of coefficients of current polynomial</td>
</tr>
<tr>
<td>NQQ</td>
<td>I</td>
<td>NQQ</td>
<td>I</td>
<td></td>
<td>Degree of current polynomial, QQ(X)</td>
</tr>
<tr>
<td>IALTER</td>
<td>I</td>
<td>IALTER</td>
<td>I</td>
<td></td>
<td>Number of alterations of an initial approximation</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td></td>
<td>CONV = TRUE implies convergence to a root</td>
</tr>
<tr>
<td>EPS4</td>
<td>R</td>
<td>EPS4</td>
<td>D</td>
<td></td>
<td>Tolerance for checking multiplicities</td>
</tr>
<tr>
<td>AP</td>
<td>C</td>
<td>UAP, VAP</td>
<td>D</td>
<td></td>
<td>Array of approximations (initial or altered) producing convergence</td>
</tr>
<tr>
<td>QD</td>
<td>C</td>
<td>UQD, VQD</td>
<td>D</td>
<td></td>
<td>Array of coefficients of newly deflated polynomial</td>
</tr>
<tr>
<td>JAP</td>
<td>I</td>
<td>JAP</td>
<td>I</td>
<td></td>
<td>Number of distinct roots found by iterative process</td>
</tr>
<tr>
<td>J1</td>
<td>I</td>
<td>J1</td>
<td>I</td>
<td></td>
<td>i.e. not as a result of Subroutine QUAD</td>
</tr>
<tr>
<td>ROOTS</td>
<td>C</td>
<td>UROOTS, VROOTS</td>
<td>D</td>
<td></td>
<td>Number of distinct roots found in the attempt to improve roots</td>
</tr>
<tr>
<td>NEWT</td>
<td>L</td>
<td>NEWT</td>
<td>L</td>
<td></td>
<td>Array of improved roots</td>
</tr>
<tr>
<td></td>
<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>
</tr>
</tbody>
</table>

Subroutine NEWTON

| X                          | C    | UX, VX                     | D    | E                      | Starting approximation (initial or altered) |
| N                          | I    | N                          | I    | E                      | Degree of current polynomial |
| P                          | C    | UP, VP                     | D    | E                      | Array of coefficients of current polynomial |
| MAX                        | I    | MAX                        | I    | C                      | Maximum number of iterations |
| EPSILON                    | R    | EPSILON                    | D    | C                      | Tolerance for checking convergence |
| XO                         | C    | UXO, VXO                   | D    | R                      | Current approximation to root |
| B                          | C    | UB, VB                     | D    |                         | Array of coefficients of newly deflated polynomial |
| DPXO                       | C    | UDFXO, VDFXO               | D    |                         | Derivative of the polynomial at XO |
### 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/DPXO</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

| X             | C    | UX, VX                     | D    | E                       | Value at which to evaluate or deflate polynomial |
| N             | I    | N                          | I    |                         | Degree of polynomial |
| P             | C    | UP, VP                     | D    |                         | Array of coefficients of polynomial |
| C             | C    | UC, VC                     | D    | R                       | Updated at each iteration to yield derivative of polynomial at X |
| B             | C    | UB, VB                     | D    |                         | Array of coefficients of newly deflated polynomial |

Subroutine QUAD

<p>| N             | I    | N                          | I    | E                       | Degree of polynomial to be solved |
| A             | C    | UA, VA                     | D    | E                       | Array of coefficients of polynomial to be solved |
| J             | I    | J                          | I    | ECR                     | Number of distinct roots found of original polynomial |
| ROOT          | C    | UROOT, VROOT               | D    | ECR                     | Array of roots found |
| MULT          | I    | MULT                       | I    | ECR                     | Array of multiplicities |
| DISC          | C    | UDISC, VDISC               | D    | ECR                     | Discriminate of quadratic |
| TEMP          | C    | UTEMP, VTEMP               | D    |                         | √DISC |
| EPSLON        | R    | EPSLON                     | D    | C                       | Tolerance for zero (0) |
| D             | C    | UD, VD                     | D    |                         | Twice leading coefficient of quadratic |</p>
<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>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, VRR</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 RR&lt;sub&gt;N1+1&lt;/sub&gt;/SS&lt;sub&gt;M1+1&lt;/sub&gt;</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>C</td>
<td>Tolerance check for zero (0)</td>
</tr>
</tbody>
</table>

Subroutine GCD

Subroutine MULTI

<table>
<thead>
<tr>
<th>N</th>
<th>I</th>
<th>N, VP</th>
<th>I</th>
<th>E</th>
<th>Degree of original polynomial, P(X)</th>
</tr>
</thead>
<tbody>
<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>E</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>E</td>
<td>Derivative of polynomial at ROOT&lt;sub&gt;i&lt;/sub&gt;</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>
<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>Subroutine DERIV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>
<tr>
<td>Subroutine DIVIDE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>UP,VP</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of dividend polynomial</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
<td>Degree of dividend polynomial</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>UD,VD</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of divisor polynomial</td>
</tr>
<tr>
<td>M</td>
<td>I</td>
<td>M</td>
<td>I</td>
<td>E</td>
<td>Degree of divisor polynomial</td>
</tr>
<tr>
<td>Q</td>
<td>C</td>
<td>UQ,VQ</td>
<td>D</td>
<td>R</td>
<td>Array of coefficients of quotient polynomial P(X)/D(X)</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td>R</td>
<td>Degree of quotient polynomial, Q(X)</td>
</tr>
<tr>
<td>J</td>
<td>I</td>
<td>J</td>
<td>I</td>
<td>Counter</td>
<td></td>
</tr>
<tr>
<td>TERM</td>
<td>C</td>
<td>UTERM, VTERM</td>
<td>D</td>
<td>Dummy variable used for temporary storage of products</td>
<td></td>
</tr>
<tr>
<td>KK</td>
<td>I</td>
<td>KK</td>
<td>I</td>
<td>Number of coefficients of quotient polynomial, Q(X)</td>
<td></td>
</tr>
<tr>
<td>Subroutine GENAPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>APPR,APPPI</td>
<td>D</td>
<td>R</td>
<td>Array containing initial approximations</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 generated</td>
</tr>
<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>Argument of complex approximation being generated</td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>R</td>
<td>APPR(I)</td>
<td>D</td>
<td>Real part of complex approximation</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>R</td>
<td>APPI(I)</td>
<td>D</td>
<td>Imaginary part of complex approximation</td>
<td></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>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>ABXOLD</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>IO2</td>
<td>I</td>
<td>IO2</td>
<td>I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
</tbody>
</table>

Subroutine COMSQT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UX,VX</td>
<td>D</td>
<td>Complex number for which the square root is desired</td>
</tr>
<tr>
<td>UY,VY</td>
<td>D</td>
<td>Square root of the complex number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></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 G.C.D.-Newton's Method
Figure E.6. (Continued)
Figure E.6. (Continued)
Figure E.6. (Continued)
Figure E.6. (Continued)
Figure E.6. (Continued)
Figure E.6. (Continued)
Figure E.6. (Continued)
Figure E.6. (Continued)
TABLE E.VII

PROGRAM FOR G.C.D. - NEWTON'S METHOD

******************************************************************************
C  * 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  *******************************************************************************
C
0001  DOUBLE PRECISION UP, VP, UAPP, VAPP, UROOT, VROOT, UDUP, VP, UD, VD, UDD, VDD, Z
0002  DOUBLE PRECISION XSTART
0003  DOUBLE PRECISION XEND
0004  DIMENSION UP(261), VP(261), UAPP(261), VAPP(261), UROOT(261), VROOT(261), MULT
0005  COMMON EPS1, EPS2, EPS3, EPS4, ID2, MAX
0006  LOGICAL NEWT, CONV
0007  DATA PHAK, GNAM, DGNAM, 2HPI, 2H01, 3H001/
0008  DATA ENTRY(1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 2H10, 2H11, 2H12, 2H13
0009  DATA (1H14, 2H15, 2H16, 2H17, 2H18, 2H19, 2H20, 2H21, 2H22, 2H23, 2H24, 2H25, 2H26/
0010  DATA ANAME(11), ANAME(21)/4HNEWT, 4H0NS /
0011  10  I=6
0012  10  J=6
0013  10  I=6
0014  READ(101, 1000) NPDOLY, NP, NAPP, MAX, EPS1, EPS2, EPS3, EPS4, XSTART, XEND,
0015  IF(1) CHECK. E1.1  STOP
0016  WRITE(102, 1020) ANAME(1), ANAME(21), NPDOLY
0017  WRITE(102, 2000) NAPP
0018  WRITE(102, 2010) MAX
0019  WRITE(102, 20701) EPS1
0020  WRITE(102, 2070) EPS2
0021  WRITE(102, 2040) EPS3
0022  WRITE(102, 2030) EPS4
0023  WRITE(102, 2040) XSTART
0024  WRITE(102, 2050) XEND
0025  WRITE(102, 2060)
0026  KKK=NP+1
0027  NNN=KKK+1
0028  DO 20  I=1, KKK
0029  JJJ=NNN-I
0030  20 READ(101, 1010) UP(IJJJ), VP(IJJJ)
0031  IF(NAPP.NE.0) GO TO 22
0032  NAPP=NP
0033  CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
0034  GO TO 23
0035  22 READ(101, 1015) (UAPP(I), VAPP(I), I=1, NAPP)
0036  23 WRITE(102, 1030) NP
0037  KKK=NP+1
0038  NNN=KKK+1
0039  DO 29  I=1, KKK
0040  29
TABLE E.VII (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0040</td>
<td>( JJJ=NNN-1 )</td>
</tr>
<tr>
<td>0041</td>
<td>WRITE(IO2,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)</td>
</tr>
<tr>
<td>0042</td>
<td>IF(NP.GE.3) GO TO 30</td>
</tr>
<tr>
<td>0043</td>
<td>J=-1</td>
</tr>
<tr>
<td>0044</td>
<td>CALL QUADNP,UP,VP,J,URoot,VRoot,MULT</td>
</tr>
<tr>
<td>0045</td>
<td>WRITE(IO2,1070)</td>
</tr>
<tr>
<td>0046</td>
<td>WRITE(IO2,1165) ([1,URoot(1),VRoot(1),MULT(1),J=1,J] )</td>
</tr>
<tr>
<td>0047</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0048</td>
<td>30 CALL DERIV(NP,UP,VP,NDP,UDP,VD)</td>
</tr>
<tr>
<td>0049</td>
<td>CALL GCD(NP,UP,VP,NDP,UDP,VD)</td>
</tr>
<tr>
<td>0050</td>
<td>IF(ND.GE.3) GO TO 70</td>
</tr>
<tr>
<td>0051</td>
<td>UDUMMY=UD(I2)*UD(I2)+VD(I2)*VD(I2)</td>
</tr>
<tr>
<td>0052</td>
<td>UZRO=(UD(I2)*UD(I2)-(VD(I2)*VD(I2)))/UDUMMY</td>
</tr>
<tr>
<td>0053</td>
<td>VZRO=(UD(I2)*VD(I2)-(UD(I2)*VD(I2)))/UDUMMY</td>
</tr>
<tr>
<td>0054</td>
<td>CALL HORNER(UZRO,VZRO,NP,UP,VP,UDUMMY,VDUMMY)</td>
</tr>
<tr>
<td>0055</td>
<td>NQ=NP-1</td>
</tr>
<tr>
<td>0056</td>
<td>DO 60 I=1,NP</td>
</tr>
<tr>
<td>0057</td>
<td>UQ(I)=U(I+1)</td>
</tr>
<tr>
<td>0058</td>
<td>VQ(I)=V(I+1)</td>
</tr>
<tr>
<td>0059</td>
<td>GO TO 80</td>
</tr>
<tr>
<td>0060</td>
<td>65 KKK=NP+1</td>
</tr>
<tr>
<td>0061</td>
<td>DO 66 I=1,KKK</td>
</tr>
<tr>
<td>0062</td>
<td>UQ(I)=U(I)</td>
</tr>
<tr>
<td>0063</td>
<td>VQ(I)=V(I)</td>
</tr>
<tr>
<td>0064</td>
<td>GO TO 85</td>
</tr>
<tr>
<td>0065</td>
<td>60 UQ(I)=UQ(I+1)</td>
</tr>
<tr>
<td>0066</td>
<td>GO TO 80</td>
</tr>
<tr>
<td>0067</td>
<td>70 CALL DIVIDE(NP,UP,VP,ND,UD,VD)</td>
</tr>
<tr>
<td>0068</td>
<td>WRITE(IO2,1120) NQ</td>
</tr>
<tr>
<td>0069</td>
<td>KKK=NQ+1</td>
</tr>
<tr>
<td>0070</td>
<td>NNN=KKK+1</td>
</tr>
<tr>
<td>0071</td>
<td>DO 83 I=1,KKK</td>
</tr>
<tr>
<td>0072</td>
<td>JJJ=NNN-1</td>
</tr>
<tr>
<td>0073</td>
<td>WRITE(IO2,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)</td>
</tr>
<tr>
<td>0074</td>
<td>IF(NQ.GE.3) GO TO 85</td>
</tr>
<tr>
<td>0075</td>
<td>GO TO 110</td>
</tr>
<tr>
<td>0076</td>
<td>85 KKK=NOQ+1</td>
</tr>
<tr>
<td>0077</td>
<td>DO 90 I=1,KKK</td>
</tr>
<tr>
<td>0078</td>
<td>UQQ(I)=UQ(I)</td>
</tr>
<tr>
<td>0079</td>
<td>VQQ(I)=VQ(I)</td>
</tr>
<tr>
<td>0080</td>
<td>GO TO 120</td>
</tr>
<tr>
<td>0081</td>
<td>90 UQQ(I)=UQQ(I+1)</td>
</tr>
<tr>
<td>0082</td>
<td>GO TO 110</td>
</tr>
<tr>
<td>0083</td>
<td>CALL QUAD(UQQ,VP,Q,J,URoot,VRoot,MULT)</td>
</tr>
<tr>
<td>0084</td>
<td>NEWT=.FALSE.</td>
</tr>
<tr>
<td>0085</td>
<td>GO TO 310</td>
</tr>
<tr>
<td>0086</td>
<td>120 DO 200 I=1,NAPP</td>
</tr>
<tr>
<td>0087</td>
<td>CALL NEWTON(UAPP(I),VP(I),Q(I),NNQ,NNQ,NNQ,NNQ,URoot,VRoot,CONV)</td>
</tr>
<tr>
<td>0088</td>
<td>IF(CONV) GO TO 160</td>
</tr>
<tr>
<td>0089</td>
<td>CALL ALTER(UAPP(I),VP(I),IALTER,ITIME)</td>
</tr>
<tr>
<td>0090</td>
<td>IF(IALTER.GT.5) GO TO 200</td>
</tr>
<tr>
<td>0091</td>
<td>GO TO 130</td>
</tr>
<tr>
<td>0092</td>
<td>130 J=J+1</td>
</tr>
<tr>
<td>0093</td>
<td>URoot(J)=UZRO</td>
</tr>
<tr>
<td>0094</td>
<td>VRoot(J)=VRZRO</td>
</tr>
<tr>
<td>0095</td>
<td>UAPP(J)=UAPP(I)</td>
</tr>
<tr>
<td>0096</td>
<td>VAPP(J)=VAPP(I)</td>
</tr>
<tr>
<td>0097</td>
<td>CALL HORNER(UZRO,VRZRO,NNQ,NNQ,NNQ,NNQ,NNQ,NNQ,NNQ,NNQ,NNQ,NNQ)</td>
</tr>
</tbody>
</table>
TABLE E.VII (Continued)

```plaintext
1098       DD I=1,NQ
1099       UQO(I) = UQO(I+1)
1100       DO I=1,NQ
1101       NQ = NQ - 1
1102       IF(NQ < 1) GO TO 220
1103       CONTINUE
1104       IF(I.EQ.1) GO TO 205
1105       IF(XEQO.EQ.0.0) GO TO 205
1106       IF(XSTART.GT.XEND) GO TO 205
1107       NAP = NQ
1108       CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
1109       GO TO 120
1110       205 IF(NQ.LT.2) GO TO 210
1111       WRITE(I02,1200)
1112       NXX = NQ + 1
1113       CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
1114       IF(N22.OEQ.O) GO TO 205
1115       IF(N22.GT.NQ) GO TO 210
1116       WRITE(I02,116) QNAME, ENTRY(JJJ), UQO(JJJ), VQO(JJJ)
1117       210 IF(J.EQ.0) GO TO 10
1118       JAP = J
1119       GO TO 230
1120       220 JAP = J
1121       CALL QUADQQ(UQQ, VQQ, JROOT, VROOT, MULT)
1122       230 WRITE(I02,1133) I, JROOT(I), VROOT(I), UAP(I), VAP(I), I=1, JAP
1123       240 IF(J.EQ.0) GO TO 10
1124       241 KKK = JAP + 1
1125       235 WRITE(I02,1134) I, JROOT(I), VROOT(I), I=KKK, J
1126       240 IF(I.GT.JAP) GO TO 241
1127       241 JAP = J
1128       GO TO 230
1129       245 KK = JAP + 1
1130       IF(IGEEQ) GO TO 280
1131       WRITE(I02,1104) I, UROOT(I), VROOT(I)
1132       241 IF(N1.EQ.0) GO TO 241
1133       245 CALL NEWTON(UROOT(1), VROOT(1), CONV)
1134       240 IF(N1.EQ.0) GO TO 241
1135       241 JAP = JAP - 1
1136       245 GO TO 230
1137       250 JAP = JAP - 1
1138       250 GO TO 300
1139       250 GO TO 300
1140       250 GO TO 300
1141       250 GO TO 300
1142       280 J1 = J1 + 1
1143       URODST(J1) = UZRD
1144       VRODST(J1) = VZRD
1145       300 CONTINUE
1146       241 IF(J1.EQ.0) GO TO 305
1147       305 WRITE(I02,1135) NQ
1148       305 WRITE(I02,1135) NQ
1149       305 WRITE(I02,1135) NQ
1150       305 WRITE(I02,1135) NQ
1151       305 WRITE(I02,1135) NQ
1152       305 WRITE(I02,1135) NQ
1153       305 WRITE(I02,1135) NQ
1154       305 WRITE(I02,1135) NQ
1155       305 WRITE(I02,1135) NQ
1156       305 WRITE(I02,1135) NQ
1157       305 WRITE(I02,1135) NQ
1158       305 WRITE(I02,1135) NQ
1159       305 WRITE(I02,1135) NQ
```

```
TABLE E.VII (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0156</td>
<td>JJJ=NNN-L</td>
</tr>
<tr>
<td>0157</td>
<td>WRITE(IO2,1040) QNAME,ENTRY,1J, venta, 1Q(1JJJ), VQ(1JJJ)</td>
</tr>
<tr>
<td>0158</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0159</td>
<td>NNEW=TRUE</td>
</tr>
<tr>
<td>0160</td>
<td>CALL MULTI(NPUPeVPeJUROOTVROOTMULTI)</td>
</tr>
<tr>
<td>0161</td>
<td>IF(NEW) GO TO 330</td>
</tr>
<tr>
<td>0162</td>
<td>WRITE(IO2,10701)</td>
</tr>
<tr>
<td>0163</td>
<td>WRITE(IO2,1165) (L,ROOT(L),VROOT(L),MULTI(L),L=1,J)</td>
</tr>
<tr>
<td>0164</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0165</td>
<td>330 WRITE(IO2,11601)</td>
</tr>
<tr>
<td>0166</td>
<td>WRITE(IO2,11901) (L,ROOT(L),VROOT(L),MULTI(L),UAPI(L),VAPI(L),L=1,JAP)</td>
</tr>
<tr>
<td>0167</td>
<td>K++=JAP+1</td>
</tr>
<tr>
<td>0168</td>
<td>IF(JAP.LT.J) WRITE(102,165) (L,ROOT(L),VROOT(L),MULTI(L),L=KKK,J)</td>
</tr>
<tr>
<td>0169</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0170</td>
<td>1000 FORMAT(3X,12,1X),X,3X,13X,4(D9.0,1X),13X,2(D7.0,1X),11)</td>
</tr>
<tr>
<td>0171</td>
<td>1010 FORMAT(2030.0)</td>
</tr>
<tr>
<td>0172</td>
<td>1015 FORMAT(2030.0)</td>
</tr>
<tr>
<td>0173</td>
<td>1020 FORMAT(///IX113HROOTS OF P(X),52X,14HMULTIPLICITIES///)</td>
</tr>
<tr>
<td>0174</td>
<td>1030 FORMAT(///IX,22HTHE DEGREE OF P(X) IS ,12,22H THE COEFFICIENTS ARE///)</td>
</tr>
<tr>
<td>0175</td>
<td>1040 FORMAT(2X,A2,4H) = D23.16,3H + D23.16,2H</td>
</tr>
<tr>
<td>0176</td>
<td>1070 FORMAT(///IX,13HROOTS OF P(X),15X,14HMULTIPLICITIES///)</td>
</tr>
<tr>
<td>0177</td>
<td>1080 FORMAT(2X,5,ROOT(12,4H) = D23.16,3H + D23.16,2H</td>
</tr>
<tr>
<td>0178</td>
<td>1100 FORMAT(2X,A3,2,4H) = D23.16,3H + D23.16,2H</td>
</tr>
<tr>
<td>0179</td>
<td>1120 FORMAT(///IX,13HROOT(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE</td>
</tr>
<tr>
<td>0180</td>
<td>1130 FORMAT(///IX,7OHCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH</td>
</tr>
<tr>
<td>0181</td>
<td>1132 FORMAT(///IX,13HROOTS OF Q(X),15X,14HMULTIPLICITIES///)</td>
</tr>
<tr>
<td>0182</td>
<td>1133 FORMAT(2X,5,ROOT(12,4H) + ,D23.16,3H + D23.16,2H</td>
</tr>
<tr>
<td>0183</td>
<td>1134 FORMAT(2X,5,ROOT(12,4H) + ,D23.16,3H + D23.16,2H</td>
</tr>
<tr>
<td>0184</td>
<td>1140 FORMAT(///IX,40HROOTS FOR INITIAL APPROXIMATION ROOT,11,24H</td>
</tr>
<tr>
<td>0185</td>
<td>1150 FORMAT(///IX,45HROOTS OF Q(X),15X,14HMULTIPLICITIES///)</td>
</tr>
<tr>
<td>0186</td>
<td>1160 FORMAT(2X,5,ROOT(12,4H) + ,D23.16,3H + D23.16,2H</td>
</tr>
<tr>
<td>0187</td>
<td>1180 FORMAT(///IX,13HROOTS OF Q(X),15X,14HMULTIPLICITIES///)</td>
</tr>
<tr>
<td>0188</td>
<td>1190 FORMAT(2X,5,ROOT(12,4H) + ,D23.16,3H + D23.16,2H</td>
</tr>
<tr>
<td>0189</td>
<td>2000 FORMAT(///IX,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN, ,12)</td>
</tr>
<tr>
<td>0190</td>
<td>2010 FORMAT(///IX,29HMNUMBER OF ITERATIONS,,11X,13)</td>
</tr>
<tr>
<td>0191</td>
<td>2020 FORMAT(///IX,21HTEST FOR CONVERGENCE,,13X,09,2)</td>
</tr>
<tr>
<td>0192</td>
<td>2030 FORMAT(///IX,24HTEST FOR MULTIPLICITIES,,10X,09,2)</td>
</tr>
<tr>
<td>0193</td>
<td>2040 FORMAT(///IX,23HRADIUS TO START SEARCH,,11X,09,2)</td>
</tr>
<tr>
<td>0194</td>
<td>2050 FORMAT(///IX,21HRADIUS TO END SEARCH,,13X,09,2)</td>
</tr>
<tr>
<td>0195</td>
<td>2060 FORMAT(///IX)</td>
</tr>
<tr>
<td>0196</td>
<td>2070 FORMAT(///IX,34HTEST FOR ZERO IN SUBROUTINE GCD,,09,2)</td>
</tr>
<tr>
<td>0197</td>
<td>2080 FORMAT(///IX,34HTEST FOR ZERO IN SUBROUTINE QUAD,,09,2)</td>
</tr>
<tr>
<td>0198</td>
<td>END</td>
</tr>
</tbody>
</table>
TABLE E.VII (Continued)

0001 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 DOUBLES PRECISION APPR,APP1,XSTART,BETA, EPS1,EPS2,EPS3,EPS4

0002 COMMON EPS1,EPS2,EPS3,EPS4

0003 IFIXSTART.EQ.0.0 XSTART=0.5

0004 BETA=0.2617994

0005 I=1,NAPP

0006 APPR(I)=XSTART*DCOS(BETA)

0007 APPR(I)=XSTART*DSIN(BETA)

0008 BETA=BETA+0.5235988

0009 RETURN

0010 END
### TABLE E.VII (Continued)

```

```

```
TABLE E.VII (Continued)

SUBROUTINE GCDIN, UR, VR, MS, VS, MUS, VSS

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

DOUBLE PRECISION USSSSS, VS

DOUBLE PRECISION URR, UR, VR, MSR, VS, MSS, URR, UR, VR, MS, MSS, URR, UR, VR, MS, MSS

DIMENSION UR(261), VR(126), US(261), VS(261), USS(261), VSS(261), URR(261), VRR(126), UT(261), VT(126)

COMMON EPSLON, EPS2, EPS3, EPS4, 102, MAX

0006 N=I

0007 M=I

0008 K=I

DO 20 I=1, K

0010 URR(I)=UR(I)

0011 VRR(I)=VR(I)

K=I+1

DO 25 I=1, K

25 URR(I)=UR(I)-((URR(I-1)*USS(M-1)+VR(I-1)*VSS(M-1))/BBB)*VRR(I-1)*VSS(M-1)+VR(I-1)*VSS(M-1))

20 CONTINUE

DO 30 I=1, N

30 BBB=DSQRT(UU(I)+UU(I-1)+UU(I-1)+UU(I-1))/BBB

30 CONTINUE

DO 40 I=1, M

40 VV(I)=(USS(I)+VSS(I)-USS(M)+VSS(M))/BBB

40 CONTINUE

IF(M.EQ.N) GO TO 70

K=I+1

DO 25 I=1, K

25 URR(I)=UR(I)

25 VRR(I)=VR(I)

70 CONTINUE

DO 90 I=1, N

90 BBB=DSQRT(UU(I)+UU(I-1)+UU(I-1)+UU(I-1))/BBB

90 CONTINUE

DO 100 I=1, M

100 K=I+1

IF(K.EQ.N) GO TO 170

100 CONTINUE

170 GO TO 30
TABLE E.VII (Continued)

0050 140 KKK+K+1
0051  DO 150 J=1.KKK
0052    URR(J)=USS(J)
0053    VRR(J)=VSS(J)
0054    USS(J)=UT(J)
0055    150 VSS(J)=VT(J)
0056    KKK+K+2
0057    NNN,M+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 USS(J)=1.0
0065    VSS(J)=0.0
0066    M1=0
0067    200 RETURN
0068    END
SUBROUTINE QUAD(UAVAJUROOT,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 UAVAJUROOT,VROOT,UDISC,VDISC,UTEMP,VDISC,UDISC,VO DISC  
DIMENSION UAV(26),VA(26),UROOT(25),VROOT(25),MULT(25)  
COMMON EPS1,EPS2,EPSLON,EPS5,IO2,MAX  

IF (N,N.EQ.1) GO TO 60  
IF (J,J.LT.0) GO TO 40  
J=J+1  
GO TO 50  
40  
MULT=I  
J=1  
50  
BBB=UAV(2)*UAV(2)*VA(2)*VA(2)  
UROOT(J)=-(UAV(1)*UAV(2)*VA(2)*VA(2))/BBB  
VROOT(J)=-(VA(1)*UAV(2)-UAV(1)*VA(2))/BBBB  
GO TO 200  
60  
UDISC=(UAV(2)*UAV(2)*VA(2)*VA(2))-4.0*(UAV(3)*UAV(1)-VA(3)*VA(1))  
VDISC=UAV(2)*UAV(2)*VA(2)*VA(2)-4.0*(UAV(3)*VA(1)+VA(3)*UAV(1))  
BBB=DSQRT(UOISC*UDISC+VDISC*VOISC)  
IF (BBB.LE.EPSLON) GO TO 100  
IF (J.GE.0) GO TO 80  
MULT=1  
J=0  
80 CALL CONSQUO(UDISC,VDISC,UTEMP,VDISC)  
UD=2.0*UAV(3)  
VD=2.0*VA(3)  
BBB=UD*UD+VD*VD  
UROOT(J+1)=((-UAV(2)+UTEMP)*UD+(-VA(2)+UTEMP)*VD)/BBB  
VROOT(J+1)=((-VA(2)-VTEMP)*UD-(-UAV(2)+VTEMP)*VD)/BBB  
UROOT(J+2)=((-UAV(2)+UTEMP)*UD+(-VA(2)+VTEMP)*VD)/BBB  
VROOT(J+2)=((-VA(2)+VTEMP)*UD-(-UAV(2)+UTEMP)*VD)/BBB  
J=J+1  
GO TO 200  
100  
IF (J,J.LT.0) GO TO 110  
J=J+1  
110  
MULT=2  
J=1  
130  
UD=2.0*UAV(3)  
VD=2.0*VA(3)  
BBB=UD*UD+VD*VD  
UROOT(J+1)=((-UAV(2)+UTEMP)*UD+(-VA(2)+UTEMP)*VD)/BBB  
VROOT(J+1)=((-VA(2)-VTEMP)*UD-(-UAV(2)+VTEMP)*VD)/BBB  
200 RETURN  
END
SUBROUTINE NEWTON(UX, VX, N, UP, VP, UXO, VXO, CONV)

* THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-
  * INATION BY USING THE ITERATION FORMULA
  * \( X(n+1) = X(n) - \frac{P(X(n))}{P'(X(n))} \).

DOUBLE PRECISION UX, VX, UP, VP, UXO, VXO, UB, VB, UDPO, VDPQ, UDPX, VDPQ, U
DIFF, VDIFF, EPS1, EPS2, EPS3, EPS4, A, B, BBB

DOUBLE PRECISION UDD

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

COMMON EPS1, EPS2, EPS3, EPS4, IO2, MAX

LOGICAL CONV

UXO = UX
VXO = VX

DO 10 I = 1, MAX

CALL HORNER(UXO, VXO, N, UP, VP, UB, VB, UDPO, VDPQ)

UPXO = UB(I)

VPXO = VB(I)

DDD = DSQRT(UDPXO*UDPXO + VDPQ*VDPQ)

IF(DDD .NE. 0.0) GO TO 5

ABPXO = DSQRT(UDPXO*UDPXO + VDPQ*VDPQ)

IF(ABPXO .EQ. 0.0) GO TO 20

GO TO 15

5 BBB = UDPXO*UDPXO + VDPQ*VDPQ

UDIFF = (UDPXO*UDPXO + VDPQ*VDPQ) / BBB

VDIFF = (VDPQ*VDPQ - UDPXO*UDPXO) / BBB

UXO = UXO - UDIFF

VXO = VXO - VDIFF

AAA = DSQRT(UDIFF*UDIFF + VDIFF*VDIFF)

BBB = DSQRT(UOXUOXO + VXO*VXO)

IF(AAA .EQ. 0.0) GO TO 10

IF(AAA .LT. EPSLON) GO TO 20

CONTINUE

15 CONV = .FALSE.

RETURN

20 CONV = .TRUE.

RETURN

END
**SUBROUTINE DIVIDE(N,UP,VP,M,UQ,VD,K,UQ,VQ)**

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

**DOUBLE PRECISION UP,VP,UD,UDM,UM,UDM,1**

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

**DOUBLE PRECISION UOUMM4Y.UDIM.13*UDIM1I)*VOIM.11*VDIM*#11**

**DOUBLE PRECISION UQIK,I.IVPIN,11*UD(M1I$-UPIN*II*VDIM*,1)I/UDUMMY**

**DOUBLE PRECISION UPIN,1I*U0IN*1I*VPIt4~1I*VDtM*1I3IUOUMMY**

**DOUBLE PRECISION IPEKaEQ.0I GO TO 100**

**J=-1**

**DO 50 I=1,K**

**UTERM=UP(N-J)**

**VTERM=VP(N-J)**

**KK=K+1**

**NN=N-J**

**M=NN+I=NN+K**

**IF(KK.GT.11) GO TO 10**

**GO TO 50**

**10 IF(M.LGE.1) GO TO 20**

**GO TO 40**

**UTERN=UTERN-(UQ(KK)*UD(M1I)-VQ(KK)*VD(M1I))**

**VTERM=VTERM-(UQ(KK)*VD(M1I)+VQ(KK)*UD(M1I))**

**40 KK=KK-1**

**45 UDUMMY=UD(M1I-11+VD(M1I)*VQ(KK+1))**

**50 VQ(KK-1)=UTERM/UD(M1I)+VTERM/VD(M1I)/UDUMMY**

**100 RETURN**

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

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

* HORNER'S METHOD Computes the value of the polynomial P(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 UX, VX, UP, VP, UB, VB, UC, VC
DIMENSION UP(26), VP(26), UB(26), VB(26)

UB(N+1)=UP(N+1)

VB(N+1)=UX*UB(N+1)+VX*VB(N+1) + UP(N)

VB(N)=UX*VB(N+1)-VX*UB(N+1) + VP(N)

UC=UB(N+1)

VC=VB(N+1)

KKK=N-1

DO 10 I=1, KKK

AAA=I-1

UA(I-1)=AAA*UP(I)

VA(I-1)=AAA*VP(I)

10 UC=UC+UB(KKK+2-I)

VC=VC+VC(KKK+2-I)

RETURN
END

SUBROUTINE DERIV(In, Up, VP, UA, VA)

* GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV Computes the coefficients of its derivative P'(X). *

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

AAA=N+1

AAA=AAA+UP(I)

VA(I-1)=AAA+VP(I)

AAA=N-1

RETURN
END
```
TABLE E.VII (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE MULTI(N,UP,VP,J,URD0T,VRDOT,MULT)</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION UP,VP,URDOT,VRDOT,UA,VA,UB,VB,UC,VC,EPS1,EPS2,EPS</td>
</tr>
<tr>
<td>0003</td>
<td>DIMENSION UP(26),VP(26),URDOT(25),VRDOT(25),UA(26),VA(26),UB(26),VC</td>
</tr>
<tr>
<td>0004</td>
<td>COMMON EPS1,EPS2,EPS3,EPSLON,1O2,MAX</td>
</tr>
<tr>
<td>0005</td>
<td>DO 100 I=1,J</td>
</tr>
<tr>
<td>0006</td>
<td>KKK=KKK+1</td>
</tr>
<tr>
<td>0007</td>
<td>DO 10 K=1,KKK</td>
</tr>
<tr>
<td>0008</td>
<td>UA(K)=UP(K)</td>
</tr>
<tr>
<td>0009</td>
<td>10 VA(K)=VP(K)</td>
</tr>
<tr>
<td>0010</td>
<td>M=N</td>
</tr>
<tr>
<td>0011</td>
<td>MULT(1)=0</td>
</tr>
<tr>
<td>0012</td>
<td>20 CALL HORN(J,URDOT(1),VRDOT(1),M,UA,VA,UB,VC)</td>
</tr>
<tr>
<td>0013</td>
<td>BBB=SQRT(U(1)+UB(1)+VB(1)+VH(1))</td>
</tr>
<tr>
<td>0014</td>
<td>IF(BBB.LT.EPSLON) GO TO 50</td>
</tr>
<tr>
<td>0015</td>
<td>IF(MULT(1).EQ.0) GO TO 40</td>
</tr>
<tr>
<td>0016</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0017</td>
<td>40 WRITE(102,1001) EPSLON,1,URDOT(1),VRDOT(1)</td>
</tr>
<tr>
<td>0018</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0019</td>
<td>50 MULT(1)=MULT(1)+1</td>
</tr>
<tr>
<td>0020</td>
<td>IF(M.GT.1) GO TO 60</td>
</tr>
<tr>
<td>0021</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0022</td>
<td>60 DO 70 K=1,M</td>
</tr>
<tr>
<td>0023</td>
<td>UA(K)=UB(K+1)</td>
</tr>
<tr>
<td>0024</td>
<td>70 VA(K)=VB(K+1)</td>
</tr>
<tr>
<td>0025</td>
<td>M=M-1</td>
</tr>
<tr>
<td>0026</td>
<td>GO TO 20</td>
</tr>
<tr>
<td>0027</td>
<td>100 CONTINUE</td>
</tr>
<tr>
<td>0028</td>
<td>RETURN</td>
</tr>
<tr>
<td>0029</td>
<td>1000 FORMAT('///15H THE EPSILON 1,010.3.48H CHECK IN SUBROUTINE MULTI</td>
</tr>
<tr>
<td>0030</td>
<td>INDICATES THAT ROOT(1,12,4H) = D23.16,3H + D23.10+2H 1,48H IS NO</td>
</tr>
<tr>
<td>0031</td>
<td>C CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP</td>
</tr>
<tr>
<td>0032</td>
<td>LICITY 0/1)</td>
</tr>
<tr>
<td>0033</td>
<td>END</td>
</tr>
</tbody>
</table>
TABLE E.VII (Continued)

0001 SUBROUTINE CONSRT(UX, VX, UY, VY)
0002                                                                                       
0003 C   ************************************************************************************
0004 C   * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.                     
0005 C   *
0006 C   ************************************************************************************
0007                                                                                       
0008 DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, AAA, BBB
0009                                                                                       
0010 R = DSQRT(UX*UX + VX*VX)                                                              
0011 AAA = DSQRT(DABS((R+UX)/2.0))                                                          
0012 BBB = DSQRT(DABS((R-UX)/2.0))                                                          
0013 IF (UX) 10, 20, 30                                                                  
0014 10 UY = AAA                                                                          
0015 20 VY = -1.0*BBB                                                                     
0016 30 GO TO 100                                                                          
0017 40 IF (VX) 50, 60                                                                    
0018 50 UY = 0.0                                                                          
0019 60 VY = DSQRT(DUMMY)                                                                  
0020 70 GO TO 100                                                                          
0021 80 VY = 0.0                                                                          
0022 90 GO TO 100                                                                          
0023 100 RETURN                                                                          
0024                                                                                       
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 \( 0.9X_0 \) and \( 1.1X_0 \).

5. Informative Messages and Error Messages

The informative messages and error messages in this program are described as follows. For other messages not listed here, see Appendix E, § 5.

"THE EPSILON (XXX) CHECK IN SUBROUTINE MULTI INDICATES THAT ROOT YY = ZZZ IS NOT CLOSE 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>
<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>R</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>UB, 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>R</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>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>Array of current polynomial coefficients (to be deflated or evaluated)</td>
</tr>
<tr>
<td>NA</td>
<td>I</td>
<td>Degree of polynomial to be deflated or evaluated</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>Approximation at which to evaluate or deflate the polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>Array containing the coefficients of the deflated polynomial</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>Value of the polynomial at X</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>Number of coefficients of polynomial to be deflated</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Single Precision Variables</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
</tr>
<tr>
<td>DENOM</td>
<td>R</td>
<td>DENOM</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA, VA</td>
</tr>
<tr>
<td>NP</td>
<td>I</td>
<td>NP</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT, VROOT</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
</tr>
<tr>
<td>BAPP</td>
<td>C</td>
<td>UBAPP, VBAPP</td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</td>
</tr>
<tr>
<td>ROOTS</td>
<td>C</td>
<td>UROOTS, VROOTS</td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>L</td>
</tr>
<tr>
<td>EPSRT</td>
<td>R</td>
<td>EPSRT</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB, VB</td>
</tr>
<tr>
<td>XL</td>
<td>C</td>
<td>UX₁, VX₁</td>
</tr>
<tr>
<td>X₂</td>
<td>C</td>
<td>UX₂, VX₂</td>
</tr>
<tr>
<td>X₃</td>
<td>C</td>
<td>UX₃, VX₃</td>
</tr>
<tr>
<td>PX₁</td>
<td>C</td>
<td>UPX₁, VPX₁</td>
</tr>
<tr>
<td>PX₂</td>
<td>C</td>
<td>UPX₂, VPX₂</td>
</tr>
<tr>
<td>PX₃</td>
<td>C</td>
<td>UPX₃, VPX₃</td>
</tr>
</tbody>
</table>
TABLE F.II (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>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td></td>
<td>CONV = true implies convergence has been obtained</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4, VX4</td>
<td>D</td>
<td></td>
<td>Newest approximation to root</td>
</tr>
<tr>
<td>J</td>
<td>I</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>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td>C</td>
<td>Maximum number of iterations permitted</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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>C</td>
<td>One of the three approximations to be altered</td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>One of the three approximations to be altered</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>One of the three approximations to be altered</td>
</tr>
<tr>
<td>X2R</td>
<td>R</td>
<td>Real part of complex approximation</td>
</tr>
<tr>
<td>X2I</td>
<td>R</td>
<td>Imaginary part of complex approximation</td>
</tr>
</tbody>
</table>

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)
DIVIDE

START

K = N - M
Q = q_m / q_n

N = 0

J = -1

K = K + 1

J = J + 1

TERM = P

RETURN

Q = TERM / Q

DERIV

START

N = N - 1

RETURN

M = P

Figure F.1. (Continued)
Figure F.1. (Continued)
Figure F.1. (Continued)
Figure F.1. (Continued)
Figure F.1. (Continued)
COMSQT

Figure F.1. (Continued)
TABLE F.III

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

C
* DOUBLE PRECISION PROGRAM FOR G.C.D.-MULLER'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 URAPP,VRAPP
0003 DIMENSION URAPP(25,3),VRAPP(25,3),UAPPI25.31,VAPP(25,31
0004 OIMENSION UP(Z61,VP(26),UROOT(25),VROOTI25),MULT(25),UDPI26IVDP(2
212),ENTRY(26)
0005 DOUBLE PRECISION XSTART
0006 DOUBLE PRECISION XEND
0007 DOUBLE PRECISION EPSRT
0008 COMMON EPSRTEPSI,EPS2,EPS3,EPS402i2MAX
0009 DATA PNAME,QNAME,QQNAME/2HP(,2HQ(,3HQQI/
0010 DATA ENTRY/IHIH2.H3,1H4,IHSelH6,1H7tIHB,H9,2H10,2HI1,2HI2.2HI4,2H15,2H16,2HI7,2H18,2119,
322H02H21,2H22,2H232H24,2H25,2H26/
0011 DATA ANAME(1),ANAME(2)/4HMULL,4HERS
0012 LOGICAL NEWT
0013 10 J=0
0014 10 J=0
0015 JJ=J+1
KCHECK
0018 IF(KCHECK.LT.0) STOP
0019 WRITE(I02,10201) ANAME(I),ANAME(I2),NOPOLY
0020 WRITE(I02,10201) NAPP
0021 WRITE(I02,10201) MAX
0022 WRITE(I02,10201) EPS1
0023 WRITE(I02,10201) EPS2
0024 WRITE(I02,10201) EPS3
0025 WRITE(I02,10201) EPS4
0026 WRITE(I02,10201) XSTART
0027 WRITE(I02,10201) XEND
0028 WRITE(I02,10201)
0029 KKK=NAP+1
0030 NNN=KKK+1
0031 DO 20 J=1,KKK
0032 JJ=J-1
0033 READ(I01,10001) UPIJJJ,VPJJJ
0034 IF(NAPP.NE.0) GO TO 22
0035 NAPP=NPP
0036 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0037 GO TO 23
0038 22 READ(I01,1015) UAPP(I21),VAPP(I21),I1=NAPP
0039 23 WRITE(I02,1030) I1,NP
0040 KKK=NAP+1
0041 NNN=KKK+1
TABLE F.1II (Continued)

0042      DD 25 I=1,KKK
0043      JJJ=NNN-1
0044      CALL QUAD(NP,UP,VP,J,UROOT,VROOT,MULT)
0045      WRITE((02,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ))
0046      IF(NP.GE.3) GO TO 30
0047      J=1
0048      CALL QUAD(NP,UP,VP,J,UROOT,VROOT,MULT)
0049      WRITE((02,1070) (I,UROOT(I),VROOT(I),MULT(I),I=1,1)
0050      GO TO 10
0051      30  CALL DERIV(NP,UP,VP,UD,VD)
0052      CALL GCD(NP,UP,VP,UD,VD,ND,UD,VD)
0053      IF(ND.EQ.0) GO TO 70
0054      UDUMMY=UO(I)+UO(1)+VD(I)+VD(1)/UDUM
0055      VJRDN=V(I-1)+V(I-2)+V(I+2)/UDUM
0056      VJRDN=V(I)/UDUM
0057      KKK=NP+1
0058      DO 65 I=1,KKK
0059      UQQ(I)=UP(I),kkk
0060      VQQ(I)=VP(I)
0061      NQQ=NP
0062      CALL DIVIDE(NP,UP,VP,ND,UD,VD,NQ,UDUM)
0063      GO TO 80
0064      65  KKK=NP+1
0065      DO 66 I=1,KKK
0066      UQQ(I)=UP(I)
0067      VQQ(I)=VP(I)
0068      GO TO 80
0069      66  KKK=NP+1
0070      DO 70 I=1,KKK
0071      UQQ(I)=UP(I)
0072      VQQ(I)=VP(I)
0073      NQ=NP
0074      GO TO 80
0075      70  CALL DIVIDE(NP,UP,VP,ND,UD,VD,NQ,UDUM)
0076      WRITE((02,120) NQ)
0077      KKK=NP+1
0078      NWW=KKK+1
0079      DO 83 I=1,KKK
0080      JJJ=NNN-1
0081      93  WRITE((02,1040) QNAME,ENTRY(JJJ),UQ(JJJ),VP(JJJ))
0082      IF(NQ.GE.3) GO TO 65
0083      GO TO 110
0084      85  KKK=NP+1
0085      DO 90 I=1,KKK
0086      UQQ(I)=UQ(I)
0087      90  VQQ(I)=VQQ(I)
0088      NQQ=NP
0089      GO TO 120
0090      CALL QUAD(NQ,UQ,VP,J,UROOT,VROOT,MULT)
0091      NEW=.FALSE.
0092      GO TO 310
0093      CALL MULLER(UQ,VP,NQ,UAPP,VAPP,XSTART,XEND,UROOT,VROOT,J)
0094      NEW=.TRUE.
0095      110 CALL MULLER(NP,UP,VP,J,UROOT,VROOT,MULT)
0096      IF(NEW) GO TO 330
0097      WRITE((02,1070) I,UROOT(I),VROOT(I),MULT(I),I=1,1)
TABLE F.III (Continued)

```
0090  GO TO 10
0100  330 WRITE(102,1180)
0101    DD 350 L=1,JAP
0102    350 WRITE(102,1190) L,UROOT(L),VRROOT(L),MULTI(L),VRAPP(L),21,VRAPP(L),21
0103    KKK=JAP+1
0104    TF(JAP,L,J) WRITE(102,1165) L,UROOT(L),VRROOT(L),MULTI(L),L=KKK,J
0105    GO TO 10
0106    1000 FORMAT(3(12,1X),9X,13,1X,41D6.0,IXI,13K,2(DT7.0IXIl1)
0107    1010 FORMAT(2030.0
0108    1015 FORMAT(2D30.0)
0109    1020 FORMAT(1H,1OX,41HGREATEST COMMON DIVISOR METHOD USED WITH .2(A4),
     135H METHOD TO FIND ZEROS OF POLYNOMIALS/.1X,10H POLYNOMIAL NUMBER .1
     22//1
0110    1030 FORMAT(1X,22HTHE DEGREE OF P(X) IS ,12,22H THE COEFFICIENTS ARE,,
     11)
0111    1040 FORMAT(2X,.A2,.A2,4H) = .023,16,3H + .023,16,2H 11
0112    1070 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES//)
0113    1080 FORMAT(2X,.5HROOT(,12,4H)
     ,D23.16,3H +
     D23.16,2H 11
0114    1100 FORMAT(2X,.3D23.16,4H) = .023,16,3H + .023,16,2H 11
0115    1120 FORMAT(///1X,7H(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE
     10DISTINCT ROOTS OF P(X)./1X,22HTHE DEGREE OF Q(X) IS ,12,22H THE C
     20EFFICIENTS ARE,,
0116    1165 FORMAT(2X,.5HROOT(1,12,4H) = .023,16,3H + .023,16,2H 11
     7X,12,10X,26H
     1RESULTS OF SUBROUTINE QUAD)
0117    1180 FORMAT(///1X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITIAL
     1 APPROXIMATION/1
0118    1190 FORMAT(2X,.5HROOT(1,12,4H) = .023,16,3H + .023,16,2H 11
     7X,12,9X,23,
     116,3H + .023,16,2H 11
0119    2000 FORMAT(///1X,4HNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,12)
0120    2010 FORMAT(1X,24HMAXIMUM NUMBER OF ITERATIONS. ,11X,13)
0121    2020 FORMAT(1X,21HTEST FOR CONVERGENCE. ,13X,09,2)
0122    2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES. ,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,3HTEST FOR ZERO IN SUBROUTINE GCD. ,09,2)
0127    2080 FORMAT(1X,4HTEST FOR ZERO IN SUBROUTINE QUAD. ,09,2)
0128    END
```
TABLE F.III (Continued)

SUBROUTINE MULTI(N,UP,VP,J,URoot,VRoot,MULT)
C
*************************************************************************
C
* GIVEN N ZEROS OF A POLYNOMIAL, SUBROUTINE MULTI COMPUTES THEIR
C * MULTICLILITIES.
C *
*************************************************************************
C
DOUBLE PRECISION UP,VP,URoot,VRoot,UA,VA,UB,VB,VC,EPS1,EPS2,EPS3,EPS
ILOH,EPSLON
DIMENSION UP(26),VP(26),URoot(251),VRoot(251),UA(261),VA(261),UB(261),VC
EPSLON,IO2,MULT(25)
DOUBLE PRECISION EPSRT
COMMON EPSRTEPSLEPS2,EPS39EPSLON,IO2,MAX

DO 100 I=1,J

DO 70 K=1,KKK

UA(K)=UP(KKK+1-K)

VA(K)=VP(KKK+1-K)

M=N
MULT(II)=O

CALL HORNEM-UA,VA,URoot,II,VRoot,II,UB,VB,VC

B7B=DSQRT(UC*UC+VC*VCI

IF(BBB.LT.EPSLON) GO TO 50

IF(MULT(II.EQ.O0) GO TO 40

GO TO 100

40 WRITE(II02,1000) EPSLON,I,URoot,II,VRoot,II

GO TO 100

GO TO 20

50 MULT(II)=MULT(II)+1

IF(MULT(II).EQ.0) GO TO 40

GO TO 100

70 UA(K)=UB(K)

VA(K)=VB(K)

M=M-1

GO TO 20

100 CONTINUE

RETURN

100 FORMAT(///15H THE EPSILON (,D10.3,48H) CHECK IN SUBROUTINE MULTI
* INDICATES THAT ROOT(II,12.4H) = .023.16.34 + .023.16.2H 1,90H IS NO
* 24 CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP
* LICLITY 0790)

1000 END
TABLE F.III (Continued)

0001 SUBROUTINE DIVIDE(N,UP,VP,M,UD,VD,K,UQ,VQ)
C **************************************************************************
C * GIVN TWO POLYNOMIALS FIXI AND GIXJ, SUBROUTINE DIVIDE COMPUTES THE *
C * QUOTIENT POLYNOMIAL (MIX = FIXI/GIXJ).
C **************************************************************************

0002 DOUBLE PRECISION UP,VP,UD,VD,UQ,VQ,UTERM,UTERM,UDDummy
0003 DIMENSION UP(26),VP(26),UD(26),VD(26),UQ(26),VQ(26)
0004 K=N-M
0005 UDDummy=UD(N+1)*UD(M+1)+VD(N+1)
0006 UQ(K+1)=UP(N+1)+UD(N+1)+VD(N+1)/UDDummy
0007 VQ(K+1)=(VP(N+1)+UD(N+1)+VD(N+1))/UDDummy
0008 IF(K.EQ.0) GO TO 100
0009 J=1
0010 DO 50 I=1,K
0011 C UTERM=UP(N-J)
0012 VTERM=VP(N-J)
0013 KK=K+1
0014 RNN=M-J
0015 DO 40 M=NNN,N
0016 C IF(KK.GT.I1) GO TO 10
0017 GO TO 45
0018 10 IF(KM.EQ.1) GO TO 20
0019 GO TO 40
0020 20 UTERM=TERM-(UQ(KK)*UD(M+1)-VQ(KK)*VD(M+1))
0021 VTERM=TERM-(UQ(KK)*UD(M+1)+VQ(KK)*VD(M+1))
0022 40 K=KK-1
0023 45 UDDummy=UD(M+1)*UD(M+1)+VD(M+1)
0024 UQ(KK+1)=UQ(KK+1)+VTERM*VQ(KK+1)/UDDummy
0025 VQ(KK+1)=VQ(KK+1)+UTERM*UQ(KK+1)/UDDummy
0026 50 K=KK-1
0027 100 RETURN
END

0001 SUBROUTINE DERIV(N,UP,VP,M,UA,VA)
C **************************************************************************
C * GIVN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF *
C * ITS DERIVATIVE P'(X).
C **************************************************************************

0002 DOUBLE PRECISION UP,VP,UA,VA,AAA
0003 DIMENSION UP(26),VP(26),UA(26),VA(26)
0004 KKK=N+1
0005 DO 10 I=2,KKK
0006 AAA=I-1
0007 UA(I-1)=AAA*UP(I)
0008 VA(I-1)=AAA*VP(I)
0009 M=1
010 RETURN
END
### TABLE F. III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>C</td>
<td>SUBROUTINE GCDIN, UR, VR, M, US, VS, M1, USS, VSS</td>
</tr>
<tr>
<td>0002</td>
<td>C</td>
<td>DOUBLE PRECISION EPSRT</td>
</tr>
<tr>
<td>0003</td>
<td>C</td>
<td>DOUBLE PRECISION US5555, V5555</td>
</tr>
<tr>
<td>0004</td>
<td>C</td>
<td>DOUBLE PRECISION UT, VN, UM, VM, US, USS, VSS, UR, VR, UD, VD, UT, VT, EPSLON, EP</td>
</tr>
<tr>
<td>0005</td>
<td>C</td>
<td>L57, EPS3, EPS4, BBB</td>
</tr>
<tr>
<td>0006</td>
<td></td>
<td>DIMENSION UR(26), VR(26), M(26), US(26), VSS(26), USS(26), VSS(26), VR(26), VRT(26)</td>
</tr>
<tr>
<td>0007</td>
<td></td>
<td>COMMON EPSRT, EPSLON, EPS3, EPS4, IO2, MAX</td>
</tr>
<tr>
<td>0008</td>
<td>M</td>
<td>N1=N</td>
</tr>
<tr>
<td>0009</td>
<td></td>
<td>KKK=N+1</td>
</tr>
<tr>
<td>0100</td>
<td></td>
<td>DO 20 =1,KKK</td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td>URR111=UR(1)</td>
</tr>
<tr>
<td>0121</td>
<td></td>
<td>20 VR111=VR(1)</td>
</tr>
<tr>
<td>0131</td>
<td></td>
<td>KKK=M+1</td>
</tr>
<tr>
<td>0141</td>
<td></td>
<td>DO 25 =1,KKK</td>
</tr>
<tr>
<td>0151</td>
<td></td>
<td>US111=US(1)</td>
</tr>
<tr>
<td>0161</td>
<td></td>
<td>25 V5551=V555(1)</td>
</tr>
<tr>
<td>0171</td>
<td></td>
<td>30 BBB=US551+M1<em>US(V5551+M1)+V5551</em>V555(M1+1)</td>
</tr>
<tr>
<td>0181</td>
<td></td>
<td>UD=UR551+M1<em>US5551+V5551</em>V555(M1+1)+/BBB</td>
</tr>
<tr>
<td>0191</td>
<td></td>
<td>VD=US5551+M1*VAR(N1+1)-UAR(N1+1)+/BBB</td>
</tr>
<tr>
<td>0201</td>
<td></td>
<td>KKK=M1-1</td>
</tr>
<tr>
<td>0211</td>
<td></td>
<td>DO 40 =1,KKK</td>
</tr>
<tr>
<td>0221</td>
<td></td>
<td>UTR111=UR(1)</td>
</tr>
<tr>
<td>0231</td>
<td></td>
<td>40 VR111=VR(1)-U0<em>US5551-1+V0</em>V5551-1+M1)</td>
</tr>
<tr>
<td>0241</td>
<td></td>
<td>IFM1.LEQ0, N1 GO TO 70</td>
</tr>
<tr>
<td>0251</td>
<td></td>
<td>KKK=N1-N1</td>
</tr>
<tr>
<td>0261</td>
<td></td>
<td>DO 60 =1, KKK</td>
</tr>
<tr>
<td>0271</td>
<td></td>
<td>UTR111=UR(1)</td>
</tr>
<tr>
<td>0281</td>
<td></td>
<td>60 VR111=VR(1)</td>
</tr>
<tr>
<td>0291</td>
<td></td>
<td>70 VI40 =1, N1</td>
</tr>
<tr>
<td>0301</td>
<td></td>
<td>BBB=US551(U0(N1+1)-1)+UT(N1+1-1)+VT(N1+1-1)+VT(N1+1-1)</td>
</tr>
<tr>
<td>0311</td>
<td></td>
<td>IFB100, GT.EPSLON1 GO TO 100</td>
</tr>
<tr>
<td>0321</td>
<td></td>
<td>90 CONTINUE</td>
</tr>
<tr>
<td>0331</td>
<td></td>
<td>DO 95 =1, M1</td>
</tr>
<tr>
<td>0341</td>
<td></td>
<td>BBB=US551+M1<em>US(V5551+M1)+V5551</em>V555(M1+1)</td>
</tr>
<tr>
<td>0351</td>
<td></td>
<td>US5555=US5551<em>US555(M1+1)+V5551</em>V555(M1+1)+/BBB</td>
</tr>
<tr>
<td>0361</td>
<td></td>
<td>V55555=V5551<em>V555(M1+1)-US5551</em>V555(M1+1)+/BBB</td>
</tr>
<tr>
<td>0371</td>
<td></td>
<td>95 V5551=V55555</td>
</tr>
<tr>
<td>0381</td>
<td></td>
<td>95 US551+M1=1</td>
</tr>
<tr>
<td>0391</td>
<td></td>
<td>V5551+M1=0</td>
</tr>
<tr>
<td>0401</td>
<td></td>
<td>GO TO 200</td>
</tr>
<tr>
<td>0421</td>
<td></td>
<td>100 K=N1-1</td>
</tr>
<tr>
<td>0431</td>
<td></td>
<td>IFK .EQ. 0 GO TO 170</td>
</tr>
<tr>
<td>0441</td>
<td></td>
<td>IFK .LT. M1 GO TO 140</td>
</tr>
<tr>
<td>0451</td>
<td></td>
<td>KKK=K+1</td>
</tr>
<tr>
<td>0461</td>
<td></td>
<td>DO 130 =1, KKK</td>
</tr>
<tr>
<td>0471</td>
<td></td>
<td>URR(J)+UT(J)</td>
</tr>
<tr>
<td>0481</td>
<td></td>
<td>130 VR1(J)=VT1(J)</td>
</tr>
<tr>
<td>0491</td>
<td></td>
<td>N1=K</td>
</tr>
</tbody>
</table>
TABLE F.III (Continued)

0050        GO TO 30
0051        140  KKK=K+1
0052  DO 150  J=1,KKK
0053        URRIJ=USSIJ
0054        VRRI(J)=VSSI(J)
0055        USSI(J)=URRIJ
0056        150  VSSI(J)=VT(IJ)
0057        KKK=K+2
0058        NNN=M1+1
0059  DO 160  J=KKK,NNN
0060        URRIJ=USSIJ
0061  160  VRRI(J)=VSSI(J)
0062        M1=M1
0063        M1=K
0064        GO TO 30
0065        170  USSI(J)=1.0
0066        VSSI(J)=0.0
0067        M1=0
0068  200 RETURN
0069        END
TABLE F.III (Continued)

0001  SUBROUTINE QUAD4,UA,VA,J,UREDIT,DUTMP,MULT
C
C ***********************************************
C *
C * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPlicITIES *
C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
C *
C *************************************************
C
0002  DOUBLe PRECISION EPS1, Eps2, EPS3, EPS4, EPSLON,BBB
0003  DIMENSION UAI(261), VA(261), UREDIT(251, VDUTMP(251), MULT(25)
0004  COMMON EPS1,EPS2,EPS3,EPS4,EPSLON, EPS3,1D2,MAX
0006  IF(NC<1) GO TO 60
0007  IF(J,JL) GO TO 40
0008  J=J+1
0009  GO TO 50
0010  40 MULT(11)=1
0011  J=1
0012  50 BBB=UA(2)*UA(2)*VA(2)*VA(2)
0013  VREDIT(J)-=-UA(I)*UA(2)+VA(1)*VA(2) /BBB
0014  GO TO 200
0015  60 UDSC=(UA(2)*UA(2)*VA(2)*VA(2)-4*(UA(1)*UA(1)-VA(2)*VA(1)))
0016  VREDIT(J)=-VA(2)-UA(1)*UA(2) /BBB
0017  GO TO 100
0018  70 IF(J,JL,0) GO TO 60
0019  MULT(11)=1
0020  MULT(22)=1
0021  IF(J,JL,0) GO TO 100
0022  80 CALL CMOSOT(UDSC,VAI,TEMPT, VTENP)
0023  BBD=UD*UD+VD
0024  UD=2,0*UA(1)
0025  UD=2,0*UA(1)
0026  UD=2,0*UA(1)
0027  IF(J,JL,0) GO TO 110
0028  IF(J,JL,0) GO TO 130
0029  IF(J,JL,0) GO TO 130
0030  IF(J,JL,0) GO TO 130
0031  IF(J,JL,0) GO TO 130
0032  100 IF(J,JL,0) GO TO 110
0033  110 MULT(11)=2
0034  J=J+1
0035  GO TO 130
0036  130 UD=2,0*UA(1)
0037  UD=2,0*UA(1)
0038  UD=2,0*UA(1)
0039  UD=2,0*UA(1)
0040  IF(J,JL,0) GO TO 110
0041  IF(J,JL,0) GO TO 110
0042  IF(J,JL,0) GO TO 110
0043  IF(J,JL,0) GO TO 110
0044  200 RETURN
0045  END
TABLE F.111 (Continued)

0001 SUBROUTINE MULLERIUAVA, IN, RROOT, VROOT, NP, UAPP, VAPP, NAPP, XSTART, XEND, UROOT, VROOT, INROOT, IROOT, UAPP, VAPP, NAPP, N0POLY)
C******************************************************************************
C MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLEITIES 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.
C******************************************************************************

0002 DOUBLE PRECISION UPX3, VPX3, UPX2, VPX2, UROOT, VROOT, UX1, VX1, UAPP, VAPP
1,UX2, VX2, UWORK, VWORK, UX3, VX3, UB, VB, UX4, VX4, UAPP, VAPP, UAPP, VAPP, XSTART, XEND
2,APP4, VPX4, EPSRT, EPSO, EPSD, EPSM, E02, MAX
3, UAPP, VAPP, XSTART, XEND
0003 DIMENSION UROOT(25), VROOT(25), MULTIZSUAPP(25, 3) VAPP(25, 3)
0004 LOGICAL CONV
0005 DOUBLE PRECISION EPS
0006 COMMON EPSRT, EPS1, EPSO, EPSM, E02, MAX
0007 DATA PNAME, DNAME/2HP(, 2HD(1/1
0008 EPSRT = 0.999
0009 NROOT = 0
0010 IROOT = 0
0011 IPATH = 1
0012 NWORK = 0
0013 NALTER = 0
0014 ITIME = 0
0015 ITIME = 0
0016 ITER = 1
0017 IF(INAPP .NE. 0) GO TO 18
0018 NAPP = NP
0019 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
0020 GO TO 27
0021 18 DO 25 I = 1, NAPP
0022 UAPP(I, 1) = 0.9 * UAPP(I, 2)
0023 VAPP(I, 2) = 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 = NP + 1
0027 DO 30 I = 1, KKK
0028 UX1 = UAPP(I, 1)
0029 VX1 = VAPP(I, 1)
0030 UX2 = UAPP(I, 2)
0031 VX2 = VAPP(I, 2)
0032 UX3 = UAPP(I, 3)
0033 VX3 = VAPP(I, 3)
0034 CALL HORNER(UWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, UX4, VX4, VPX3, UX4)
0035 CALL HORNER(UWORK, VWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, VPX3, VPX4)
0036 CALL HORNER(UWORK, VWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, VPX3, VPX4)
0037 CALL HORNER(UWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, VPX3, VPX4)
0038 CALL HORNER(UWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, VPX3, VPX4)
0039 CALL HORNER(UWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, VPX3, VPX4)
0040 CALL HORNER(UWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, VPX3, VPX4)
0041 CALL HORNER(UWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, VPX3, VPX4)
0042 CALL HORNER(UWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, VPX3, VPX4)
0043 CALL HORNER(UWORK, UX1, VX1, UX2, VX2, UX3, VX3, VB, VPX3, VPX4)
TABLE F.III (Continued)

0044 IF(IPX3.EQ.0.0) GO TO 70
0045 QQ=ABPX6/ABPX3
0046 IF(QQ.LE.10.) GO TO 70
0047 UQ=0.5*UQ4
0048 VQ=0.5*VQ4
0049 UX=UX3+(UX3*UQ4-VQ4)
0050 VX=VX3+(UX3*UQ4+VQ4)
0051 GO TO 60
0052 CALL TEST(UX3,VX3,UX4,VX4,CONVI)
0053 IF(CONVI GO TO 120
0054 IF(ITER.LT.MAX) GO TO 110
0055 CALL ALTER(UAPP(IAPP,1),VAPP(IAPP,1),UAPP(IAPP,2),VAPP(IAPP,2),UAP
0057 IF(NALTER.GT.5) GO TO 75
0058 ITER=1
0059 GO TO 40
0060 IF(IAPP.LT.NAPP) GO TO 100
0061 IF(XEND.EQ.0.0) GO TO 77
0062 NAPP=NP
0063 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0064 IAPP=0
0065 GO TO 100
0066 77 WRITE(102,1090)
0067 KKK=WORK
0068 WRITE(102,1035) (NAME(J),UWORK(J),VWORK(J),J=1,KKK)
0069 IF(NROOT.EQ.0.GO TO 90
0070 IF(IPATH.EQ.1) GO TO 82
0071 IF(NPATH.EQ.0.GO TO 82
0072 CALL BETTER(UA,VA,NP,UROOT,VROOT,1,URAPP,VRAPP,KROOT,MULT
0073 RETURN
0074 IF(NROOT.EQ.0.GO TO 90
0075 IF(IROOT.EQ.1) GO TO 95
0076 WRITE(102,1080)
0077 DO 55 I=1,IROOT
0078 55 WRITE(102,1085) (URROOT(I),VRROOT(I),I=1,NROOT)
0079 IF(IROOT.LT.NROOT) GO TO 85
0080 GO TO 97
0081 85 KKK=IROOT
0082 WRITE(102,1086) (URROOT(I),VRROOT(I),I=1,NROOT)
0083 87 IF(IPATH.EQ.0) GO TO 81
0084 RETURN
0085 90 WRITE(102,1070) NOPOLY
0086 RETURN
0087 100 IAPP=IAPP+1
0088 ITER=1
0089 NALTER=0
0090 GO TO 40
0091 120 NROOT=NROOT+1
0092 IROOT=NROOT
0093 MULT=NROOT
0094 NDUMULT=NDUMULT+1
0095 URROOT(NROOT)=UX4
0096 VRROOT(NROOT)=VX4
0097 URAPP(NROOT,1)=UPAPP(IAPP,1)
0098 VRAPP(NROOT,1)=VAPP(IAPP,1)
0099 URAPP(NROOT,2)=UPAPP(IAPP,2)
0100 VRAPP(NROOT,2)=VAPP(IAPP,2)
244

TABLE F.III (Continued)

0101  URAPP(NROOT,3)=UAPP(1APP,3)
0102  VRAPP(NROOT,3)=VAPP(1APP,3)
0103  IF(NOMULT.LT.NPI) GO TO 130
0104  DO TO 80
0105  CALL HORNERN(WORK,WORK,UX,UX,UX,UB,UB,UPX,VPX)
0106  NWORK=NWORK+1
0107  DO 140 I=1,KKK
0108  WORK(I)=UB(I)
0109  WORK(I)=VB(I)
0110  CALL HORNER(WORK,WORK,UX,UX,UX,UB,UB,UPX,VPX)
0111  CCC=SQRT(UPX*UPX+VPX*VPX)
0112  IF(CCC.LT.EPSM) GO TO 150
0113  IF(INWORK.GT.2) GO TO 75
0114  IROOT=NROOT
0115  KKK=NWORK-1
0116  DO 145 I=1,KKK
0117  UB(I)=UWORK(KKK+1-I)
0118  VB(I)=VWORK(KKK+1-I)
0119  CALL QUAD(INWORK,UB,VB,NROOT,UROOT,VROOT,MULT)
0120  GO TO 80
0121  MULT(NROOT)=MULT(NROOT)+1
0122  NDMULT=NDMULT+1
0123  GO TO 125
0124  UX1=UX2
0125  VX1=VX2
0126  UX2=UX3
0127  VX2=VX3
0128  UX2=UX3
0129  VX3=VX4
0130  UX3=UX4
0131  UPX1=UPX2
0132  VPX1=VPX2
0133  UPX2=UPX3
0134  VPX2=VPX3
0135  UPX3=UPX4
0136  VPX3=VPX4
0137  ITER=ITER+1
0138  GO TO 50
0139  1090 FORMAT(1X,65HCOEFFICIENTS OF DEFATED POLYNOMIAL FOR WHICH NO ZEROES WERE FOUND/)
0140  1080 FORMAT(1X,43HINITIAL APPROXIMATION/)
0141  1070 FORMAT(1X,43HNO ZEROES WERE FOUND FOR POLYNOMIAL NUMBER ,12)
0142  1085 FORMAT(2X,SHROOT(1H9I2*4H10.5D23.16,3H# 023.16,2H I,98X#023.16,2H I)
0143  1035 FORMAT(3X,023.16,2H I,98X#023.16,2H I)
0144  1050 FORMAT(1X,023.16,2H I,98X#023.16,2H I)
0145  1089 FORMAT(2X,5HROOT(1H9I2*4H10.5D23.16,3H# 023.16,2H I,98X#023.16,2H I)
0146  END
TABLE F.III (Continued)

0001  SUBROUTINE BETTER(UA,VA,NP,URoot,VRoot,NRoot,UAPP,VRAPP,1ROOT,MUL)

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

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

0002  DOUBLE PRECISION UROOT,VROOT,UA,VA,UAPP,VRAPP,UX1,UX2,UX3

0003  LOGICAL CONV

0004  DIMENSION UROOT(25),VROOT(25),UA(25),VA(25),UAPP(25,3),VRAPP(25,3)

0005  DOUBLE PRECISION EPS1,EPSH

0006  COMMON EPSRT,EPSL,EPS,POLY,IO2,MAX

0007  IF(NROOT.LE.11 RETURN

0008  L=0

0009  DO 10 I=1,NROOT

0010  UAPP(I,1)=UROOT(I)*EPSRT

0011  VRAPP(I,1)=VROOT(I)*EPSRT

0012  UAPP(I,2)=UROOT(I)

0013  VRAPP(I,2)=VROOT(I)

0014  UAPP(I,3)=UROOT(I)*(2.0-EPSRT)

0015  10  VRAPP(I,3)=VROOT(I)*(2.0-EPSRT)

0016  DO 100 J=1,NROOT

0017  UXI=UAPP(I,J)

0018  VXI=VRAPP(J,1)

0019  UX2=UAPP(I,J)

0020  VX2=VRAPP(J,2)

0021  UX3=UAPP(I,J)

0022  VX3=VRAPP(J,3)

0023  ITER=1

0024  CALL HORNER(NP,UA,VA,UX1,UX2,UX3,UB,VB,UPX1,UPX2,VPX1)

0025  CALL HORNER(NP,UA,VA,UX1,UX2,UX3,UB,VB,UPX2,VPX2)

0026  20  CALL HORNER(NP,UA,VA,UX1,UX2,UX3,UB,VB,UPX3,VPX3)

0027  CALL CALC(UA,VA,UX1,UX2,UX3,UX4,UPX1,UPX2,VPX1,VPX2,VPX3)

0028  30  CALL TEST(UX3,VX3,UX4,VPX4,CONV)

0029  IF(CONV) GO TO 50

0030  IF(ITER.LT.MAX) GO TO 40

0031  WRITE(102,1000) J,UROOT(I),VRoot(J),MAX

0032  WRITE(102,1010) UX4,VPX4

0033  IF(J.LE.IROOT) GO TO 33

0034  IF(J.EQ.IROOT) GO TO 35

0035  GO TO 100

0036  33  KKK=1

0037  DO 34 K=1,3

0038  UAPP(I,K)=UAPP(I,K+1)

0039  VRAPP(I,K)=VRAPP(I,K+1)

0040  34  VRAPP(I,K)=VRAPP(I,K+1)

0041  35  IROOT=IROOT-1

0042  GO TO 100
TABLE F.111 (Continued)

0046 40 UX1=UX2
0047 VX1=VX2
0048 UX2=UX3
0049 VX2=VX3
0050 UX3=UX4
0051 VX3=VX4
0052 UX4=UX1
0053 VX4=VX1
0054 UPX1=UPX2
0055 VPX1=VPX2
0056 UX2=UX3
0057 VX2=VX3
0058 UPX2=UPX3
0059 VPX2=VPX3
0060 ITER=ITER+1
0061 GO TO 20
0062 50 L=L+1
0063 UROOTS(L)=UX4
0064 VROOTS(L)=VX4
0065 100 CONTINUE
0066 IF(L.EQ.0) GO TO 120
0067 DO 110 I=1,L
0068 UROOT(I)=UROOTS(I)
0069 VROOT(I)=VROOTS(I)
0070 NROOT=L
0071 RETURN
0072 120 NROOT=0
0073 RETURN
0074 1000 FORMAT(//42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT1,124H = , L023.16,3H + ,023.16,2H 1/24H DID NOT CONVERGE AFTER ,13,11H ITERAT 20NS)
0075 1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,023.16,3H + ,023.16,2H 1/ 1/)
0076 END
TABLE F.11 (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, 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 NALTER = NALTER + 1
IF (INALTER .GT. 5) RETURN
GO TO 130

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

20 BETA = BETA + 1.047196
X2R = R * DCSIN(BETA)
X2I = R * DCOS(BETA)

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

100 FORMAT (1H0, 5H X1R, X1I, X2R, X2I, X3R, X3I, NALTER, ITIME)
1010 FORMAT (/// 1H0, 5H NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
ITER, 13, 12H ITERATIONS ///)
1020 FORMAT (1H0, 5H HALTED APPRXIM
ATIONS/1X, 13.12H ITERATIONS/1X, 13.12H IT,)
TABLE F.III (Continued)

 SUBROUTINE GENAPP(APPR, APP!, NAPP, XSTART)

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

SUBROUTINE GENAPP generates N initial approximations, where N is the **
degree of the original polynomial.

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

DOUBLE PRECISION APPR, APP!, XSTART, EPS1, EPS2, EPS3, BETA

COMMON EPSRTEPS1IEPS2, EPS3, EPS4, 102, MAX

IF (XSTART.EQ.0.0) XSTART=0.5

BETA=0.2617994

DO 10 I=1,NAPP

APPRI!2)=XSTART*DCOS(BETA)

APPRI!3)=XSTART*DSIN(BETA)

BETA=BETA+0.5235988

DO 20 I=1,NAPP

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

APPRI!,2)=0.9*APPRI!,1)

APPRI!,3)=1.1*APPR(I,2)

APPRI!,3)=1.1*APPRI!,2)

RETURN

END
TABLE F.III (Continued)

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

C SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C IMATIONS BY TESTING THE EXPRESSION
C ABSOLUTE VALUE OF \( |(XIN+1)-XIN|/\|XIN+1\| \). WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.

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

C IDENOM
C LOGICAL CONV
C DOUBLE PRECISION EPSR, EPS1, EPSM
C COMMON EPSRT, EPS1, EPS, EPSM, IO2, MAX
C UDUMMY = UX4 - UX3
C VDUMMY = VX4 - VX3
C AAA = DSQRT(UDUMMY*UDUMMY + VDUMMY*VDUMMY)
C DENOM = DSQRT(UX4*UX4 + VX4*VX4)
C IF (DENOM < EPSO) GO TO 20
C IF (AAA / DENOM < EPS) GO TO 10
C CONV = .FALSE.
C GO TO 100
C CONV = .TRUE.
C GO TO 100
C GO TO 10
C IF (AAA < EPSO) GO TO 10
C GO TO 5
C 100 RETURN
C END

0003 SUBROUTINE HORNER(NA, UA, VA, UX, VX, UB, VB, UX, VX)

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 \( (X-D) \).

0004 DOUBLE PRECISION UX, VA, UX, VX, UB, VB, UB, VA

C DIMENSION UA(26), VA(26), UB(126), VB(126)
C NUM = UA + 1
C GO 10 (=2, NUM)
C UB(I) = UA(I - 1) + UB(I - 1) * UX - VB(I - 1) * VX
C GO 10
C VPX = VB(NUM)
C RETURN
C END
TABLE F.III (Continued)

0001  SUBROUTINE CALC(UX1, UX2, VX2, UX3, VX3, UPX1, UPX2, UPX3, V
               ************************************************************
               * GIVE THREE APPROXIMATIONS XN-2), XIN-1, AND XN1, SUBROUTINE CALC.
               * APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF
               * THE QUADRATICCLOSEST TO XN1. THIS ZERO IS THE NEW APPROXIMATION
               * XN1 TO THE ZERO OF THE POLYNOMIAL.
               ************************************************************
               DOUBLE PRECISION ARG1, ARG2
               DOUBLE PRECISION UPX3, VX3, UPX2, VX2, UPX1, VX1, UX2, UX3, VX3, UPX1, UX1
               V03, U03, V03, U03, VO3, VO3, UO3, VO3, UO3, VO3, UO3, VO3, UO3, VO3, UO3
               3D, AAA, BB, RAD, UAAA, VAAA, UBBB, VBBB
               DOUBLE PRECISION EPSRT, EPS1, EPS2, EPSM, 102, MAX
               COMMON EPSRT, EPS1, EPS2, EPSM, IO2, MAX
               UX3=UX3-UX2
               VX3=VX3-VX2
               VH2=UX2-UX1
               U00D=1.0-U03
               V00D=V03
               U0=UPX3-(U00D*UPX2*V00D*UPX2*V00D)-V03*UX1+(U03*U03-U03*U03*V03)
               V0=UPX3-(V00D*UPX2*V00D*UPX2*V00D)-V03*UPX1+(V03*V03*V03)
               VAAA=2.0*U03
               UAAA=UAAA*UAAA
               UBBB=U00D+U00D+U00D+U00D
               VBBB=V00D*V00D*V00D*V00D
               UCC=U03*V03*V03*V03
               VCC=V03*V03*V03*V03
               UB=(VAAA*UPX3-VAAA*UPX3)-(UBBB*UPX2-VBBB*UPX2)+(VCCC*UPX1*VCCC*V
               UPX1)
               VD=(VAAA*UPX3+VAAA*UPX3)-(UBBB*UPX2*VBBB*UPX2)+VCCC*UPX1*UCCC*V
               UPX1)
               UC=U00D*UPX3-V00D*UPX3
               VC=V00D*UPX2-V00D*UPX2
               UDISC=UBBB+UBBB+VBBB-14.0*(U00D*V00D)-V03*UX1-(U03*U03*V03)
               VDISC=12.0*(V03*V03)-14.0*(V03*V03)-U03*V03
               AAA=DSRT(UDISC=UDISC*VDISC*VDISC)
               IF (AAA.EQ.0.0) GO TO 5
               GO TO 7
               5 THTA=0.0
               GO TO 9
               7 THTA=DATAN2(VDISC, UDISC)
               9 RAD=DSRT(AAA)
               ANGLE=THETA/2.0
               UTES=RAD=0.0
               VTEST=RAD=DSIN(ANGLE)
               ANGLE=THETA/2.0
               UTES=RAD=0.0
               VTEST=RAD=DSIN(ANGLE)
               UDE1=UB*UTES
               UDE2=UB*UTEST
               UDE3=UB*UTEST
               VDE1=UB*UTEST
               VDE2=UB*UTEST
               VDE3=UB*UTEST
               0044
TABLE F.III (Continued)

ARG1 = UDEN1*UDEN1 + VDEN1*VDEN1
ARG2 = UDEN2*UDEN2 + VDEN2*VDEN2
AAA = SQRT(ARG1)
BBB = SQRT(ARG2)
IF (AAA .LT. BBB) GO TO 10
IF (AAA .EQ. 0.0) GO TO 60
UAAA = -2.0*UC
VAAA = -2.0*VC
UQ4 = (UAAA*UDEN1 + VAAA*VDEN1)/ARG1
VQ4 = (VAAA*UDEN1 - UAAA*VDEN1)/ARG1
GO TO 50
10 IF (BBB .EQ. 0.0) GO TO 60
UAAA = -2.0*UC
VAAA = -2.0*VC
UQ4 = (UAAA*UDEN2 + VAAA*VDEN2)/ARG2
VQ4 = (VAAA*UDEN2 - UAAA*VDEN2)/ARG2
GO TO 50
50 UX4 = UX3 + (UH3*UQ4 - VH3*VQ4)
VX4 = VX3 + (VH3*UQ4 + UH3*VQ4)
RETURN
60 UQ4 = 1.0
VQ4 = 0.0
GO TO 50
END
TABLE F.III (Continued)

```
C SUBROUTINE CONSQT(UX, VX, UY, VY)
C
C *********** ************************************************************
C
C ** THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER. **
C
C *********** ************************************************************
0001 C DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, AAA, BBB
0002 0003 R = DSQRT(UX + VX + VX + VX)
0004 AAA = DSQRT(DABS(R + UX1/2.01))
0005 BBB = DSQRT(DABS(R - UX1/2.01))
0006 IF (UX) 10, 20, 30
0007 10 UY = AAA
0008 0009 VY = 1.0*BBB
0010 GO TO 100
0011 20 IF (UX) 40, 50, 60
0012 30 UY = AAA
0013 0014 VY = BBB
0015 GO TO 100
0016 40 DUMMY = DABS(UX)
0017 0018 UY = 0.0
0019 VY = DSQRT(DUMMY)
0020 GO TO 100
0021 50 UY = 0.0
0022 0023 VY = 0.0
0024 60 DUMMY = DABS(UX)
0025 0026 100 RETURN
0027 END
```
APPENDIX G

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

1. Use of the Program

A double precision FORTRAN IV program using the repeated G.C.D. method with Newton's method as a supporting method is presented here. Flow charts for this program are given in Figure G.2 while Table G.III gives a FORTRAN IV listing of this program. Single precision variables are listed in Table G.II. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from Table G.II.

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

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

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

Main Program

Data Entry/1H1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1

    UP(N+1), VP(N+1)
    UAPP(N), VAPP(N)
    UDO(N+1), VDO(N+1)
    UDDO(N+1), VDDO(N+1)
    UDL(N+1), VDL(N+1)
    UD2(N+1), VD2(N+1)
    UDDL(N+1), VDDL(N+1)
    UG(N+1), VG(N+1)
    UD3(2N+1), VD3(2N+1)
    UD4(2N+1), VD4(2N+1)
    UZROS(N), VZROS(N)
    UAP(N), VAP(N)
    UROOT(N), VROOT(N)
    NULT(N)
    ENTRY(N+1)

Subroutine PROD

    UH(2N+1), VH(2N+1)
    UF(N+1), VF(N+1)
    UG(N+1), VG(N+1)

Subroutine ZROS

    UAPP(N), VAPP(N)
    UROOT(N), VROOT(N)
    UQ(N+1), VQ(N+1)
    UQQ(N+1), VQQ(N+1)
    UAP(N), VAP(N)
    UQD(N+1), VQD(N+1)
    ENTRY(N+1)
    UROOTS(N), VROOTS(N)

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

See corresponding subroutine in Table E.I.

Subroutine QUAD

    UROOT(N), VROOT(N)
    UA(N+1), VA(N+1)
2. Input Data for Repeated G.C.D. - Newton's Method

The input data for repeated G.C.D. - Newton's method is prepared as described for G.C.D. - Newton's method in Appendix E, § 2 except that the item EPS4 on the control card (Figure E.2) is omitted. An example control card for the repeated G.C.D. - Newton's method is given in Figure G.1.

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

The definitions of variables used in repeated G.C.D. - Newton's method are given in Table G.II. For definitions of variables not listed in this table, see the main program or corresponding subprogram of Table E.VI. The notation and symbols used are defined in Appendix E, § 3.

4. Description of Program Output

The number of the polynomial, control data, degree and coefficients of the polynomial are printed as described in Appendix E, § 4.

All roots of multiplicity one are extracted first. Following the first row of asterixes, the message "THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPLICITY 1." This is followed by the coefficients of G(X) with the leading coefficient listed first. If there are no roots of multiplicity one, then the message "NO ROOTS OF MULTIPLICITY ONE" is printed.

The roots of G(X) are printed under the heading "ROOTS OF G(X)." These are the roots obtained before the attempt to improve accuracy. The initial approximations producing convergence to the corresponding root are printed under the heading "INITIAL APPROXIMATION." The
message "RESULTS OF SUBROUTINE QUAD" means that the corresponding root was obtained from subroutine QUAD.

The roots found as a result of attempting to improve accuracy are printed under the heading "ROOTS OF P(X)." Their multiplicity is given under the heading "MULTIPLICITIES." The initial approximation is printed above where "NO INITIAL APPROXIMATION" means the same as "RESULTS OF SUBROUTINE QUAD."

A line of asterixes is then printed. This procedure is then repeated for the roots of multiplicity 2, 3, 4, etc. until all roots have been found.

5. Informative Messages and Error Messages

The informative messages and error messages for repeated G.C.D. - Newton's method are given below. For those not listed, see Appendix E, § 5.

"NOT ALL ROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUND." This message indicates that some of the roots of the polynomial G(X) were not extracted.

"QUAD FOUND XXX TO BE A MULTIPLE ROOT." XXX represents the value of the root found as a multiple root by Subroutine QUAD.
<table>
<thead>
<tr>
<th>N</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>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>

Figure G.1 Control Card for Repeated G.C.D. - Newton's Method
### TABLE G.II

**REPEATED GCD - NEWTON'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>KD</td>
<td>I</td>
<td>KD</td>
<td>I</td>
<td>Main Program</td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td>Main Program</td>
<td>Number of roots found</td>
</tr>
<tr>
<td>J1</td>
<td>I</td>
<td>J1</td>
<td>I</td>
<td>Main Program</td>
<td>Multiplicity of given root</td>
</tr>
<tr>
<td>DO</td>
<td>C</td>
<td>UDO, VDO</td>
<td>D</td>
<td>Main Program</td>
<td>Array of coefficients of original polynomial</td>
</tr>
<tr>
<td>ND0</td>
<td>I</td>
<td>ND0</td>
<td>I</td>
<td>Main Program</td>
<td>Degree of original polynomial</td>
</tr>
<tr>
<td>DDO</td>
<td>C</td>
<td>UDDO, VDDO</td>
<td>D</td>
<td>Main Program</td>
<td>Array of coefficients of derivative of DO(X) i.e. DO'(X)</td>
</tr>
<tr>
<td>NDDO</td>
<td>I</td>
<td>NDDO</td>
<td>I</td>
<td>Main Program</td>
<td>Degree of DDO(X)</td>
</tr>
<tr>
<td>D1</td>
<td>C</td>
<td>UD1, VD1</td>
<td>D</td>
<td>Main Program</td>
<td>Array of coefficients of g.c.d. of DO(X) and DDO(X)</td>
</tr>
<tr>
<td>ND1</td>
<td>I</td>
<td>ND1</td>
<td>I</td>
<td>Main Program</td>
<td>Degree of D1(X)</td>
</tr>
<tr>
<td>DDI</td>
<td>C</td>
<td>UDD1, VDD1</td>
<td>D</td>
<td>Main Program</td>
<td>Array of coefficients of derivative of D1(X) i.e. D1'(X)</td>
</tr>
<tr>
<td>NDD1</td>
<td>I</td>
<td>NDD1</td>
<td>I</td>
<td>Main Program</td>
<td>Degree of DDD1(X)</td>
</tr>
<tr>
<td>D2</td>
<td>C</td>
<td>UD2, VD2</td>
<td>D</td>
<td>Main Program</td>
<td>Array of coefficients of g.c.d. of D1(X) and DDD1(X)</td>
</tr>
<tr>
<td>ND2</td>
<td>I</td>
<td>ND2</td>
<td>I</td>
<td>Main Program</td>
<td>Degree of D2(X)</td>
</tr>
<tr>
<td>D3</td>
<td>C</td>
<td>UD3, VD3</td>
<td>D</td>
<td>Main Program</td>
<td>Array of coefficients of the product of DO(X) and D2(X)</td>
</tr>
<tr>
<td>ND3</td>
<td>I</td>
<td>ND3</td>
<td>I</td>
<td>Main Program</td>
<td>Degree of D3(X)</td>
</tr>
<tr>
<td>D4</td>
<td>C</td>
<td>UD4, VD4</td>
<td>D</td>
<td>Main Program</td>
<td>Array of coefficients of the square of D1(X)</td>
</tr>
<tr>
<td>ND4</td>
<td>I</td>
<td>ND4</td>
<td>I</td>
<td>Main Program</td>
<td>Degree of D4(X)</td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td>UG, VG</td>
<td>D</td>
<td>Subroutine ZROS</td>
<td>Array of coefficients of the quotient D3(X)/D4(X)</td>
</tr>
<tr>
<td>NG</td>
<td>I</td>
<td>NG</td>
<td>I</td>
<td>Subroutine ZROS</td>
<td>Degree of G(X)</td>
</tr>
<tr>
<td>ZROS</td>
<td>C</td>
<td>UZROS, VZROS</td>
<td>D</td>
<td>Subroutine ZROS</td>
<td>Array of roots of G(X)</td>
</tr>
</tbody>
</table>

**Subroutine ZROS**

<table>
<thead>
<tr>
<th>APROX</th>
<th>C</th>
<th>UAPROX, VAPROX</th>
<th>D</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Starting approximation (initial or altered)</td>
</tr>
<tr>
<td>Single Precision</td>
<td>Double Precision</td>
<td>Disposition of Argument</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>-------------------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Variable Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>I</td>
<td>M</td>
<td>I</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>C</td>
<td>UF,VF</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td>UC, VG</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>MN</td>
<td>I</td>
<td>MN</td>
<td>I</td>
<td>R</td>
</tr>
<tr>
<td>H</td>
<td>C</td>
<td>UH, VH</td>
<td>D</td>
<td>R</td>
</tr>
<tr>
<td>LIMIT</td>
<td>I</td>
<td>LIMIT</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>

Subroutine PROD
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

**************************************************************************
0001 COMMON EPS1, EPS2, EPS3, UP, VP, UAPP, VAPP, UDD0, VDD0, 1
UDD1, UDD2, VDD1, UDD3, UDD4, UDD5, VDD5, UDD6, VDD6, 2
UDD7, VDD7, UDD8, VDD8, UDD9, VDD9, UDD10, VDD10, 3
UDD11, VDD11, UDD12, VDD12, UDD13, VDD13, UDD14, VDD14, 4
UDD15, VDD15, UDD16, VDD16, UDD17, VDD17, UDD18, VDD18, 5
UDD19, VDD19, UDD20, VDD20, UDD21, VDD21, UDD22, VDD22, 6
UDD23, VDD23, UDD24, VDD24, UDD25, VDD25, UDD26, VDD26, 7
UDD27, VDD27, UDD28, VDD28, UDD29, VDD29, UDD30, VDD30, 8
UDD31, VDD31

0002 COMMON XSTART, XEND, MULT, ENTRY, UP(26), VP(26)

0003 COMMON ANAME(2), ANAME1(251), ANAME2(251), ANAME3(251)

0004 COMMON EPS1, EPS2, EPS3, UAPP(26), VAPP(26)

0005 COMMON UP(26), VP(26)

0006 DATA PNAMEGNAME/2HP(,2HG(/,

0007 DATA ENTRY/IHI,IH2,LH3,1H4,IH5,IH6,1H7,1H8,1H9,1HO,1HP,1HQ,1HR,1HT,1HU,1HV,1HW,1HX,1HY,1HZ,

0008 2HZ1,2HZ2,2HZ3,2HZ4,2HZ5,2HZ6,2HZ7,2HZ8,2HZ9,2HZ10,2HZ11,2HZ12,2HZ13

0009 DATA ANAME1(2), ANAME2(251), ANAME3(251)

0010 IF(KCHECK.EQ.1)

0011 STOP

0012 WRITE(IO2,10201 ANAME(1), ANAME2(1), NAPP

0013 WRITE(IO2,10301 NAPP

0014 WRITE(IO2,10401 MAX

0015 WRITE(IO2,10501 EPS1

0016 WRITE(IO2,10601 EPS2

0017 WRITE(IO2,10701 EPS3

0018 WRITE(IO2,10801 XSTART

0019 WRITE(IO2,10901 XEND

0020 READ(IO1,10001) NOPOLY, NP, NAPP, MAX, EPS1, EPS2, EPS3, XSTART, XEND, KCHECK

0021 IF(KCHECK.EQ.1) STOP

0022 WRITE(IO2,10201 ANAME1(1), ANAME1(2), NOPOLY

0023 WRITE(IO2,10301 NAPP

0024 WRITE(IO2,10401 MAX

0025 WRITE(IO2,10501 EPS1

0026 WRITE(IO2,10601 EPS2

0027 WRITE(IO2,10701 EPS3

0028 WRITE(IO2,10801 XSTART

0029 WRITE(IO2,10901 XEND

0030 IF(NAPP.NE.0) GO TO 22

0031 READ(IO1,10101) UP(IJJJ), VP(IJJJ)

0032 IF(NAPP.NE.01) GO TO 22

0033 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)

0034 GO TO 23

0035 READ(IO1,10151) UAPP(I), VAPP(I), I=1,NAPP

0036 WRITE(IO2,10201) NP

0037 WRITE(IO2,10301) NP

0038 WRITE(IO2,10401) NAME, ENTRY(IJJJ), UP(IJJJ), VP(IJJJ)

0039 K = 0

0040 KD = 0
TABLE G.11 (Continued)

0041  J1=1
0042  KKK=NP+1
0043  DO 10 1=1,KKK
0044  UDO(1)=UP(1)
0045  10 VDO(1)=VP(1)
0046  NDO=NP
0047  CALL DERIV(NDO,UDO,VDO,NDDO,UDDO,VD0,VDDO,NDO1,UD01,VD01)
0048  20 WRITE(IO2,3000) (STER(1)=1,33)
0049  IF(ND1.LE.1) GO TO 30
0050  GO TO 40
0051  30 UO2(1)=1.0
0052  VD2(1)=0.0
0053  NDO=0
0054  GO TO 50
0055  40 CALL DERIV(ND1,UD1,VD1,NDD1,UDD1,VD1,NDD2,UD2,VD2)
0056  50 IF(ND0+ND2.LE.2*ND1) GO TO 60
0057  CALL DERIV(ND1,UD1,VD1,NDD1,UDD1,VD1,NDD2,UD2,VD2)
0058  60 WRITE(IO2,10251)
0059  IF(NO1.LE.0.0) GO TO 80
0060  70 IF(NO1.EQ.0.0) GO TO 90
0061  GO TO 90
0062  80 KKK=ND0+1
0063  DO 85 I=1,KKK
0064  UG(I)=UGO(I)
0065  VG(I)=VGO(I)
0066  NG=NO0
0067  GO TO 110
0068  90 IF(NDDO.EQ.0.0) GO TO 115
0069  CALL PROD(NDDO,UGO,VG0,NDDO1,UGO1,VG01)
0070  100 CALL PROD(NDD1,UD1,VD1,NDD11,UD1,VD11)
0071  110 CALL PROD(NDD2,UD2,VD2,NDD21,UD2,VD21)
0072  WRITE(IO2,10351)
0073  IF(J.LT.0.0) GO TO 120
0074  115 WRITE(IO2,10351)
0075  120 IF(NJAP.EQ.0.0) GO TO 150
0076  CALL ZEROING,UG,VG,UAPP,UAPP1,UGVAP,VGJAP,VAP,ENTER
0077  130 WRITE(IO2,1180)
0078  IF(JAP.LT.0.0) GO TO 140
0079  140 WRITE(IO2,1190)
0080  IF(JAP.EQ.0.0) GO TO 150
0081  WRITE(IO2,1190)
0082  150 WRITE(IO2,1095)
TABLE G. III (Continued)

```
0098  IF (J.EQ.0) GO TO 170
0099  155 00 160  I = 1, J
0100  UROOT(KO+II=UZROSIII
0101  VROOTIKD+II=VZROS(II
0102  160  MULTIKD+II=J1
0103  K=(J*J1)+K
0104  KD=KD+J
0105  IF (K.GE.NP) GO TO 1
0106  170  J=J+1
0107  IF (NDI.LE.1) GO TO 200
0108  DO 180 I = 1, NDI
0109  UDO(I)=UDII(I)
0110  VDO(I)=VDII(I)
0111  UDDOII(I)=UDDOII(I)
0112  VDDOII(I)=VDDOII(I)
0113  UD0I(I)=UDII(I)
0114  VDOI(I)=VDII(I)
0115  NDI=NDI+1
0116  KKK=ND2+1
0117  DO 180 I = 1, KKK
0118  UDI(I)=UDII(I)
0119  VDI(I)=VDII(I)
0120  190  VOLI(I)=VOII(I)
0121  KD=KD+1
0122  DENOM=UDI(2)*UDO(2)*VDI(2)*VD0(2)
0123  UROOT(KD)=I-UD0(1)*UD0(2)-VOI(1)*VD0(2)/DENOM
0124  VROOTIKDI=I-VOI(1)*UD0(1)+UD0(2)*VD0(2)/DENOM
0125  MULT(KD)=J1
0126  WRITE(002,30001 (ASTERI=11331
0127  WRITE(IO2,10351 JI
0128  KKK=KKK+1
0129  NNN=KKK+1
0130  DO 210 I = 1, KKK
0131  JJJ=NNN-I
0132  WRITE(IO2,1100) OINAMEENTRY(JJJ),UD0IJJJtVD0IJJJ
0133  WRITE(IO2,1180) 0134  WRITE(IO2,1085) KD,UROOTIKDI),VROOTIKDI,J1
0135  GO TO 1
0136  210  WRITE(IO2,1100) OINAMEENTRY(JJJ),UD0IJJJtVD0IJJJ
0137  WRITE(IO2,1180) 0138  WRITE(IO2,1085) KD,UROOTIKDI),VROOTIKDI,J1
0139  GO TO 1
0140  1020 FORMAT(1H1,10X,48HREPEATED USE OF THE GREATEST COMMON DIVISOR AND
0141  1,A4,A4,5H METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIAL
0142  125/1X,1HMUPOLYNOMIAL NUMBER ,12//1
0143  1025 FORMAT(//1X,25HROOTS OF MULTIPlicity ,12//1
0144  1035 FORMAT(//1X,25HTHE FOLLOWING POLYNOMIAL, GIX, CONTAINS ALL THE R
0145  1042  1085 FORMAT(2X,5HROOTS OF P(x) WHICH HAVE MULTIPLICITY ,12//1
0146  1095 FORMAT(1H1,10X,5H NOT ALL ROOTS OF THE ABOVE POLYNOMIAL, G6, WERE FOUN
0147  1000 FORMAT(3I12,1X),9X,1X,3(06.0,1X),20X,2(07.0,1X)
0148  1010 FORMAT(2D30.0)
0149  1015 FORMAT(2D30.0)
0150  1030 FORMAT(1X,22H THE DEGREE OF P(x) IS ,12,22H THE COEFFICIENTS ARE//
0151  1040 FORMAT(2X,42,A2,A2,4H1 =,D23.16,3H +,D23.16,2H 1
0152  1100 FORMAT(2X,42,A3,A2,4H1 =,D23.16,3H +,D23.16,2H 1

```
### TABLE G.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0150</td>
<td>Format for roots of $\pi_1$ and multiplicities of initial approximation</td>
</tr>
<tr>
<td>1180</td>
<td>Format for roots of $\pi_1$ and multiplicities of initial approximation</td>
</tr>
<tr>
<td>1190</td>
<td>Format for roots of $\pi_1$ and multiplicities of initial approximation</td>
</tr>
<tr>
<td>0190</td>
<td>Format for roots of $\pi_1$ and multiplicities of initial approximation</td>
</tr>
<tr>
<td>2000</td>
<td>Format for number of initial approximations given</td>
</tr>
<tr>
<td>0153</td>
<td>Format for maximum number of iterations</td>
</tr>
<tr>
<td>0154</td>
<td>Format for test for convergence</td>
</tr>
<tr>
<td>0155</td>
<td>Format for radius to start search</td>
</tr>
<tr>
<td>0156</td>
<td>Format for radius to end search</td>
</tr>
<tr>
<td>0157</td>
<td>Format for test for zero in subroutine GCQ</td>
</tr>
<tr>
<td>0158</td>
<td>Format for test for zero in subroutine QUAD</td>
</tr>
<tr>
<td>0160</td>
<td>Format for end of subroutine</td>
</tr>
<tr>
<td>0161</td>
<td>End</td>
</tr>
</tbody>
</table>

The table continues with various formats for printing and testing values, indicating how the program handles roots, approximations, and convergence criteria.
TABLE G.III (Continued)

0001 SUBROUTINE PROOCM,UF,VF,UG,VG,MN,UH,VH
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 *******************************************************
0002 DOUBLE PRECISION UH,VH,UF,VF,UG,VG
0003 DIMENSION UH(51),VH(51),UF(26),VF(26),UG(26),VG(26)
0004 MN=M+N
0005 KKK=MN+1
0006 DO 100 I=1,KKK
0007 VHIII=0.0
0009 VHII=0.0
0010 IF(I.LE.M+1) GO TO 10
0011 LIMIT=M+1
0012 GO TO 10
0013 LIMIT=I
0014 20 J=I,LIMIT
0015 IF(J+I.GT.N+I) GO TO 40
0016 IF(J+K.EQ.I+1) GO TO 50
0017 GO TO 50
0018 40 UH(I)=UH(I)+(UF(JI)*UG(KI)-VF(JI)*VG(KI))
0019 VH(I)=VH(I)+(VF(JI)*UG(KI)+UF(JI)*VG(KI))
0020 50 K=K-1
0021 100 CONTINUE
0022 RETURN
0023 END

0001 SUBROUTINE GENAPPIAPP,APPI,NAPP,XSTART
C ************************************************************************
C • SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE
C • DEGREE OF THE ORIGINAL POLYNOMIAL.
C ************************************************************************
0002 DOUBLE PRECISION APPR,APPI,XSTART,BETA, EPSI,EPS2,EPS3
0003 DIMENSION APPR(25),APPI(25)
0004 COMMON EPSI,EPS2,EPS3,102,AX,IXSTART,EQU
0005 IFIXSTART.EQ.O) XSTART=0.5
0006 BETA=0.2617994
0007 DO 10 I=1,NAPP
0008 APPRI=XSTART*OCOS(BETA)
0009 APPI(I)=XSTART*DSIN(BETA)
0010 BETA=BETA+0.2617994
0011 10 XSTART=XSTART+0.5
0012 RETURN
0013 END
TABLE G.III (Continued)

SUBROUTINE ALTER(XOLDR, XOLDI, NALTER, ITIME)
C
C * ***************************************** *****************************************
C * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO *
C * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
C *
C ******************************************************* ***********************
C
DOUBLE PRECISION XOLDR, XOLDI, ABXOLDO, BETA, EPS1, EPS2, EPS3
COMMON EPS1, EPS2, EPS3, IO2, MAX

IF (ITIME .NE. 0) GO TO 5
ITIME = 1
WRITE(IO2, 1010) MAX
5 IFINALTER .EQ. 0 GO TO 10
WRITE(IO2, 1000) XOLDR, XOLDI
GO TO 20
10 ABXOLDO = DSQRT((XOLDR*XOLDRI*IXOLOI*XOLDII)
BETA = OATAN2(XOLDI, XOLDR)
WRITE(IO2, 1020) XOLDR, XOLDI
NALTER = NALTER + 1
IFINALTER .GT. 5 RETURN
GO TO (30, 40, 30, 40, 30), NALTER
30 XOLDR = -XOLDR
XOLDI = -XOLDI
GO TO 50
40 BETA = BETA * 1.0471976
XOLDR = ABXOLDO*DCOS(BETA)
XOLDI = ABXOLDO*DSIN(BETA)
50 RETURN
1000 FORMAT(1X, 02, 16, 3H + .023, 16, 2H I, 10X, 21HALTED APPROXIMATION)
1010 FORMAT(/1X, 54H NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
ITER, 13, 12H ITERATIONS, //)
1020 FORMAT(1X, 02, 16, 3H + .023, 16, 2H I, 10X, 21INITIAL APPROXIMATION)
END
TABLE G. III (Continued)

SUBROUTINE ZEROS(UQ,VQ,UG,UGP,J,VROOT,JAP,UP,VP,VPX,VPXH,VPXH1,VPXH2,VPXH3,
ENTRY,XSTART,XEND)

C ***********************************************************************
C * NEWTON'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPlicITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-
C * IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION
C * FORMULA
C *
C * X(N+1) = X(N) - P(X(N))/P'(X(N)).
C ***********************************************************************

0002 DOUBBLE PRECISION UAPP,VAPP,J,VROOT,UZRO,VPX,UP,VP,VPX,VPXH,VPXH1
0003 DOUBLE PRECISION XAPP,XSTART
0004 DIMENSION UAPP(25),VAPP(25),JAPP(25),UP(25),VPX(25),VPXH(25),VPXH1(25),VPXH2(25)
0005 COMMON EPS1,EPS2,EPS3,IO2,MAX
0006 DATA QNAME,INVNAME,0NAME,2NAME,2NAME/
0007 LOGICAL CONV
0008 J=0
0009 ITIME=0
0010 IF(NQ.GT.3) GO TO 85
0011 GO TO 110
0012 85 KKK=NO+1
0013 GO 90 1=1,KKK
0014 UQ(11)=UQ(1)
0015 90 VQ(11)=VQ(1)
0016 NQQ=NO
0017 GO TO 120
0018 120 CALL QUAD(UQ,VQ,J,VROOT)
0019 JAP=0
0020 GO TO 310
0021 130 CALL NEWTON(UAPRX,VAPRX,NQ,UQ,VQ,UZRO,VPX,CONV)
0022 IF(CONV) GO TO 160
0023 CALL ALTER(UAPP1,J,VAPP1,JALTER,ITIME)
0024 IF(JALTER.GT.5) GO TO 200
0025 UAPRX=UAPP1
0026 VAPRX=VAPP1
0027 GO TO 130
0028 160 J=J+1
0029 UQ0(TJ)=UZRO
0030 VQ0(TJ)=VZRO
0031 AIP(J)=AIPX
0032 VPX(J)=VPXH
0033 CALL HOMER(UZRO,VZRO,QQQ,VQQ,VQQ,VQQ,VQQ,QDUMMY,QDUMMY)
0034 GO 180 11=1,NQQ
0035 UQ011=UQ011+1
0036 180 QQQ(QQQ)=UQQ(1)
0037 QQQ+NO=1
0038 IF(NQ.GT.3) GO TO 200
0039 JAP=J
0040 GO TO 220

276
TABLE G.III (Continued)

0045 200 CONTINUE
0046 IF(J.GE.NQ) GO TO 205
0047 IF(XEND.EQ.0.0) GO TO 205
0048 IF(XSTART.GT.XEND) GO TO 205
0049 NAPP=NO
0050 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0051 GO TO 120
0052 205 IF(NQQ.LE.2) GO TO 210
0053 WRITE(102,1200)
0054 KKK=NQQ+1
0055 NNN=KKK+1
0056 DO 157 L=1,KKK
0057 JJJ=NNN-L
0058 157 WRITE( 102,1100) QNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0059 210 IF(J.J.EQ.0) GO TO 310
0060 JAP=J
0061 GO TO 230
0062 220 CALL QUADINQQ,UQQ,VQQ,J,UROOT,VROOT)
0063 230 WRITE(102,1133) I,UROOT(I),VROOT(I),UAPI(I),VAP(I),I=1,JAP
0064 IF(JAP.I.EQ.1) GO TO 235
0065 235 KKK=JAP+1
0066 WRITE(102,1134) I,UROOT(I),VROOT(I),I=KKK,J
0067 240 J=0
0068 DO 300 I=1,J
0069 CALL NEWTON(UROOT(I),VROOT(I),UQQ,I,UQQ,I,UQQ,I)
0070 IF(CONV) GO TO 280
0071 WRITE(102,1140) I,UROOT(I),VROOT(I),MAX,NQ
0072 245 VAP(I+1)=VAP(I)+1
0073 250 JAP=JAP-1
0074 280 J=J+1
0075 WRITE(102,1150) NQ
0076 KKK=NO+1
0077 NNN=KKK+1
0078 DO 306 L=1,KKK
0079 JJJ=NNN-L
0080 242 WRITE(102,1040) QNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0081 241 KKK=JAP+1
0082 DO 245 I=1,KKK
0083 UAP(I)=UAP(I-1)
0084 245 VAP(I)=VAP(I-1)
0085 250 JAP=JAP-1
0086 GO TO 300
0087 280 IF(J.LT.JAP) GO TO 241
0088 290 UKROOTS(JJ)=UROOT(JJ)
0089 VKROOTS(JJ)=VROOT(JJ)
0090 300 CONTINUE
0091 300 J=J+1
0092 301 IF(J.EQ.0) GO TO 305
0093 DO 303 I=1,J
0094 303 UKROOTS(I)=UKROOTS(I)
0095 305 VKROOTS(I)=VKROOTS(I)
0096 305 GO TO 310
0097 305 WRITE(102,1150) NQ
0099 KKK=NO+1
0100 NNN=KKK+1
0101 306 L=1,KKK
0102 JJJ=NNN-L
TABLE G.III (Continued)

0103    306 WRITE(102,1040) QNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0104    310 RETURN
0105    1200 FORMAT///I1X,T0HCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH
0106    1132 FORMAT///I1X,13HROOTS OF G(X),04X,21HINITIAL APPROXIMATION//)
0107    1133 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,2H I,17XD23.16,3H
0108    1134 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H I,22X#26HRESULT
0109    1140 FORMAT(//I1X,4OHNO ROOTS FOR INITIAL APPROXIMATION ROOT(,I2,4H) =
0110    1150 FORMAT///I1X,45H1ST Iteration ON THE POLYN
0111    1140 FORMAT(2X,A3,A2,4H) = ,D23.16,2H I,138H WITH GENERATED INITIAL APPROXIMATIONS//)
0112    1150 FORMAT(2X,A3,A2,4H) = ,D23.16,3H I,138H WITH GENERATED INITIAL APPROXIMATIONS//)
0113     END

C ***********
C * GIVEN POLYNOMIALS P(X) AND Q(X) WHERE DEG. P(X) IS LESS THAN DEG. Q(X)
C * P(X), SUBROUTINE GCD Computes the GREATEST COMMON DIVISOR OF P(X) AND
C * Q(X).
C
C ****************************************************************************

DOUBLE PRECISION US, V, US, V
DOUBLE PRECISION U, V, US, V
DIMENSION U(26), V(26), US(26), US(26), US(26), US(26)
COMMON EPSLON, EPS2, EPS3, MAX

N1 = N
N2 = M
K = N1 - M
DO 25 I = 1, K

20 VRI = VR(I)
KVR = K
KVR = KVR + 1
DO 25 VRI = VR(I)

25 DO 10 VRI = VR(I)
10 VRI = VR(I)

DO 20 VRI = VR(I)

DO 30 I = 1, N1

30 BBB = U(11) + V(11) + U(11) + V(11)

DO 30 I = 1, N1

40 VRI = VRI - BBB

DO 30 I = 1, M1

50 BBB = U(11) + V(11) + U(11) + V(11)

DO 30 I = 1, M1

60 VRI = VRI

DO 70 I = 1, N1

70 VRI = VRI

DO 80 I = 1, N1

80 VRI = VRI

DO 90 I = 1, N1

90 CONTINUE

DO 100 I = 1, M1

100 CONTINUE

DO 120 I = 1, M1

120 CONTINUE

DO 130 I = 1, M1

130 CONTINUE

DO 140 I = 1, M1

140 CONTINUE

DO 150 I = 1, M1

150 CONTINUE

DO 160 I = 1, M1

160 CONTINUE

DO 170 I = 1, M1

170 CONTINUE

DO 180 I = 1, M1

180 CONTINUE

DO 190 I = 1, M1

190 CONTINUE

DO 200 I = 1, M1

200 CONTINUE

DO 210 I = 1, M1

210 CONTINUE

DO 220 I = 1, M1

220 CONTINUE

DO 230 I = 1, M1

230 CONTINUE

DO 240 I = 1, M1

240 CONTINUE

DO 250 I = 1, M1

250 CONTINUE

DO 260 I = 1, M1

260 CONTINUE

DO 270 I = 1, M1

270 CONTINUE

DO 280 I = 1, M1

280 CONTINUE

DO 290 I = 1, M1

290 CONTINUE

DO 300 I = 1, M1

300 CONTINUE

DO 310 I = 1, M1

310 CONTINUE

DO 320 I = 1, M1

320 CONTINUE

DO 330 I = 1, M1

330 CONTINUE

DO 340 I = 1, M1

340 CONTINUE

DO 350 I = 1, M1

350 CONTINUE

DO 360 I = 1, M1

360 CONTINUE

DO 370 I = 1, M1

370 CONTINUE

DO 380 I = 1, M1

380 CONTINUE

DO 390 I = 1, M1

390 CONTINUE

DO 400 I = 1, M1

400 CONTINUE

DO 410 I = 1, M1

410 CONTINUE

DO 420 I = 1, M1

420 CONTINUE

DO 430 I = 1, M1

430 CONTINUE

DO 440 I = 1, M1

440 CONTINUE

DO 450 I = 1, M1

450 CONTINUE

DO 460 I = 1, M1

460 CONTINUE

DO 470 I = 1, M1

470 CONTINUE

DO 480 I = 1, M1

480 CONTINUE

DO 490 I = 1, M1

490 CONTINUE
TABLE G.III (Continued)

140 KKK=K+1
0051 DO 150 J=1,KKK
0052 URR(J)=USS(J)
0053 VRR(J)=VSS(J)
0054 USS(J)=UT(J)
0055 150 VSS(J)=VT(J)
0056 KKK=K+2
0057 NNN=Ml+1
0058 DO 160 J=KKK,NNN
0059 URR(J)=USS(J)
0060 VRR(J)=VSS(J)
0061 Ml=M
0062 Ml=K
0063 GO TO 30
0064 170 USS(1)=1.0
0065 VSS(1)=0.0
0066 Ml=0
0067 200 RETURN
0068 END
Unable to load image.
TABLE G. III (Continued)

0001 SUBROUTINE DIVIDEIN,UP,VP,M,UD,VD,K,UQ,VQ

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

0002 DOUBLE PRECISION UP,VP,UD,VD,UQ,VQ,UTERM,VTERM,UDUMMY

0003 DIMENSION UP(261),VP(261),UD(261),VD(261),UQ(261),VQ(261)

0004 K=N-M

0005 UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)

0006 UQ(K+1)=(UP(N+)*UD(M+1)+VP(N+)*VD(M+1))/UDUMMY

0007 VQ(K+1)=(VP(N+)*UD(M+1)-UP(N+)*VD(M+1))/UDUMMY

0008 IF(K.EQ.0) GO TO 100

0009 J=0

0010 DO 30 I=1,K

0011 J=J+1

0012 UTERM=UP(N-J)

0013 VTERM=VP(N-J)

0014 KK=K+1

0015 NNN=M-J

0016 DO 45 MI=NNN,M

0017 IF(KK.GT.1) GO TO 10

0018 GO TO 45

0019 10 IF(KK.GT.1) GO TO 20

0020 GO TO 45

0021 UTERM=UTERM-(VQ(KK)*UD(MI)-UQ(KK)*VD(MI))/UDUMMY

0022 VTERM=VTERM-(UQ(KK)*VD(MI)+VQ(KK)*UD(MI))/UDUMMY

0023 KK=KK-1

0024 45 UDUMMY=UD(MI)*UD(MI)+VD(MI)*VD(MI)

0025 UQ(KK+1)=UTERM*UD(MI)-VTERM*VD(MI)/UDUMMY

0026 VQ(KK+1)=VTERM*UD(MI)+UTERM*VD(MI)/UDUMMY

0027 100 RETURN

0028 END
TABLE G.III (Continued)

0001 SUBROUTINE HORNERTUX, V, N, UP, VP, UB, VB, UC, VC)
C
C *********************************************************
C
C * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A
C * POINT D AND ITS DERIVATIVE AT D. SYTHETIC DIVISION IS USED TO
C * DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X - D).
C
C *********************************************************
C
0002 DOUBLE PRECISION UX, V, U, V, UP, VP, UB, VB, UC, VC
0003 DOUBLE PRECISION UDIMM, VDIMM
0004 DIMENSION UP(26), VP(26), UB(26), VB(26)
0005
0006 UB(N+1)=UP(N+1)
0007 VP(N+1)=UP(N+1)
0008 UB(N+1)=UX*UB(N+1)-VX*VB(N+1)+UP(N+1)
0009 UC=UB(N+1)
0010 VC=VB(N+1)
0011 KKK=N-1
0012 DO 10 I=1, KKK
0013 UB(KKK+1-I)=UX*UB(KKK+2-I)-VX*VB(KKK+2-I)+UP(KKK+1-I)
0014 VB(KKK+1-I)=UX*VB(KKK+2-I)+VX*UB(KKK+2-I)+VP(KKK+1-I)
0015 UDIMM=UX-U*VC
0016 VDIMM=UX-V*UC
0017 UC=UDIMM+UB(KKK+2-I)
0018 VC=VDIMM+VB(KKK+2-I)
0019 10 RETURN
0020 END
SUBROUTINE QUAD(NUA, VA, UROOT, VROOT)

C**************************************************************************
C
C
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.

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

DOUBLE PRECISION EPS1, EPS2, EPSLON, UROOT, VROOT, VA, UD, UD, UD
DIMENSION UROOT(25), VROOT(25), UAI(26), VAI(26)
COMMON EPS1, EPS2, EPSLON, IO2, MAX

IF(IN, GT, 1) GO TO 10

J = J + 1

BBD = UAI(2) * UAI(2) + VA(2) * VA(2)

UROOT(J) = -(UAI(2) * VA(2)) / BBB

VROOT(J) = -(VA(2) * UAI(2)) / BBB

GO TO 100

UDISC = (UA(2) * UA(2) - VA(2) * VA(2)) - (4.0 * (UA(3) * UA(1) - VA(3) * VA(1)))

VDISC = (UA(2) * VA(2)) - (4.0 * (UA(3) * VA(1) + VA(3) * UA(1))

U = 2.0 * UAI(2)

V = 2.0 * VA(2)

DDD = SQRT(UDISC * UDISC + VDISC * VDISC)

IFI(DDD .LT. EPSLON) GO TO 20

CALL COMSQT(UDISC, VDISC, UTEMP, VTEMP)

BBB = UD * UD + VD * VD

UROOT(J) = ((-UAI(2) * UTEMP) + (-VA(2) * VTEMP)) / VD

VROOT(J) = ((-VA(2) * UTEMP) + (-UA(2) * VTEMP)) / UD

GO TO 100

J = J + 1

BBD = UD * UD + VD * VD

UROOT(J) = (UA(2) * UD - VA(2) * VD) / BBB

VROOT(J) = (VA(2) * UD + UA(2) * VD) / BBB

WRITE(IO2, 1000) UROOT(J), VROOT(J)

1000 FORMAT(1X, 3H QUAD FOUND, 23.16, +, 23.16, 2H 1, 2H TO BE A N
MULTIPLE ROOT/)

100 RETURN

END
TABLE G.III (Continued)

0001 SUBROUTINE DERIV(N,UP,VP,U,V,A)
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,V,A
0003 KKK=N+1
0004 DO 10 I=2,KKK
0005 AAA=I-I
0006 U(AA)=AAA*UP(I)
0007 V(AA)=AAA*VP(I)
0008 10 RETURN
0010 END

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

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

1. Use of the Program

A double precision FORTRAN IV program using the repeated G.C.D. method with Muller's method as a supporting method is presented here. Flow charts for this program are given in Figure H.1 while Table H.III gives a FORTRAN IV listing of this program.

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

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

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

Main Program

Data Entry/1H1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1
UAPP(N,3), VAPP(N,3)
URAPP(N,3), URAPP(N,3)
UP(N+1), VP(N+1)
MULT(N)
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)
UDEL(N+1), VDEL(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 $.9X_0$ 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 H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
TABLE H.11

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

C
C
C  ********************************************************************
C
C  DOUBLE PRECISION PROGRAM FOR THE REPEATED G.C.D.-MULLER'S METHOD
C
C  THIS METHOD REPEATEDLY FINDS THE GREATEST COMMON DIVISOR OF TWO
C  POLYNOMIALS IN ORDER TO EXTRACT THE ZEROS IN GROUPS ACCORDING TO
C  MULTIPLICITY USING NEWTON'S METHOD. ALL ZEROS OF MULTIPLICITY 1
C  ARE EXTRACTED FOLLOWED BY THOSE OF MULTIPLICITY 2, ETC.
C
C  ********************************************************************
C
0001  DOUBLE PRECISION EPS1, EPS2, EPS3, UP, VP, UAPP, VAPP, UDD0, UDD0, UDD0,
       1, UDD1, UDD1, UDD0, UDD0, UDD0, UDD1, UDD1, UDD1, UDD1, UDD1,
0006  1, UDD1, UDD1, UDD1, UDD1, UDD1, UDD1, UDD1, UDD1, UDD1, UDD1,
0007  DIMENSION UAPP(25,3), VAPP(25,3), UDD0(25,3), VAPP(25,3), UDD0(25,3)
0008  1, UDD1(25,3), UDD1(25,3), UDD1(25,3), UDD1(25,3), UDD1(25,3)
0009  COMMON EPS1, EPS2, EPS3, EPS4, EPS5, EPS6, EPS7, EPS8, EPS9
0010  DATA PNAME,GNAME,2/HPL,2/HPL, BLNAME/3/HPL/
0011  DATA ASTER/4/HPL/
0012  DATA ENTRY/11H,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,2H12,2H13
0013  1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26
0014  DATA ANAME(11), ANAME(2), ANAME(2), ANAME(2), ANAME(2), ANAME(2), ANAME(2)
0015  IF(ISI=5) 10=5
0016  IF(ISI=5) 10=5
0017  WRITE(10,2,2000) NAPP
0018  WRITE(10,2,2000) NAPP
0019  WRITE(10,2,2000) MAX
0020  WRITE(10,2,2000) EPS1
0021  WRITE(10,2,2000) EPS2
0022  WRITE(10,2,2000) EPS3
0023  WRITE(10,2,2000) XSTART
0024  WRITE(10,2,2000) XEND
0025  WRITE(10,2,2000) XSTART
0026  KKK=NP+1
0027  NNN=KKK+1
0028  GO TO 10,1=1, KKK
0029  JJJ=NNN-1
0030  READ(10,1,1010) UP, VP, JJJ
0031  IF(NAPP.NE.0) GO TO 22
0032  NAPP=NP
0033  CALL GENAPP(UAPP, VAPP, NAPP, JJJ)
0034  GO TO 23
0035  READ(10,1,1015) UP, VP, JJJ
0036  WRITE(10,2,1030) NAPP
0037  KKK=NP+1
0038  NNN=KKK+1
0039  GO TO 10,1=1, KKK
0040  JJJ=NNN-1
TABLE H.III (Continued)

      WRITE(IO2,1040) PNAME,ENTRY(JJJJ),UP(JJJJ),VP(JJJJ)
0041   K=0
0042   KO=0
0043   J=1
0044   KKK=NP+1
0045   DO 10 I=1,KKK
0046   UD(I)=UP(I)
0047   10 VDO(I)=VP(I)
0048   ND0=NP
0049   CALL DERIV(ND0,UD0,VD0,ND00,UD00,VD00)
0050   CALL GCOD(ND0,UD0,VD0,ND00,UD00,VD00,ND1,UD1,VD1)
0051   WRITE(IO2,3000) (ASTER,=1,33)
0052   IF(NDI.LE.1) GO TO 30
0053   ND0=NP
0054   GO TO 40
0055   30 UD2(I)=1.0
0056   VD2(I)=0.0
0057   ND2=0
0058   GO TO 50
0059   40 CALL DERIV(NDI,UDI,VDI,ND01,UD01,VD01)
0060   CALL GCOD(NDI,UDI,VDI,ND01,UD01,VD01,ND2,UD2,VD2)
0061   50 IF(ND0+NDI.LE.2*NDI) GO TO 60
0062   GO TO 70
0063   60 WRITE(IO2,1025) JI
0064   GO TO 170
0065   70 IF(NDI.EQ.01) GO TO 80
0066   GO TO 90
0067   80 KKK=ND0+1
0068   DO 85 I=1,KKK
0069   UG(I)=UD(I)
0070   VG(I)=VD(I)
0071   NG=ND0
0072   GO TO 110
0073   90 IF(ND0.EQ.01) GO TO 115
0074   CALL PROD(ND0,UD0,VD0,ND2,UD2,VD2,ND3,UD3,VD3)
0075   CALL PROD(NDI,UDI,VDI,ND01,UD01,VD01,ND4,UD4,VD4)
0076   CALL DIVIDE(ND3,UD3,VD3,ND4,UD4,VD4,NG,UG,VG)
0077   WRITE(IO2,1035) JI
0078   KKK=NG+1
0079   NKN=KKK+1
0080   DO 112 I=1,KKK
0081   JJJ=NNN-I
0082   112 WRITE(IO2,1040) GNAME,ENTRY(JJJJ),UG(JJJJ),VG(JJJJ)
0083   KKK=NG+1
0084   ND 113 I=1,KKK
0085   UP(I)=UG(KKK+I-1)
0086   VAP(I)=VG(KKK+I-1)
0087   CALL NULLER(NG,UG,VP,HAP,HAPP,VAPP,VAPP,J,URGOS,VIROS,JAP,XSTART,XENID,NOPOLY,URAPP,VRAPP)
0088   IF(J.LE.0) GO TO 150
0089   WRITE(IO2,1180)
0090   IF(JAP.EQ.0) GO TO 120
0091   GO TO 130
0092   115 KKK=ND0+1
0093   DO 116 I=1,KKK
0094   UD3(I)=UD0(I)
0095   116 VD3(I)=VD0(I)
0096   ND3=ND0
0097   GO TO 100
TABLE H.III (Continued)

0098 120 KKK=JAP+1
0099 WRITE(102,1085) (I,UZROS(I),VZROS(I),J1,J=KKK,J)
0100 GO TO 140
0101 130 DO 135 J=1,JAP
0102 135 WRITE(102,1190) I,UZROS(I),VZROS(I),J1,URAPP(I,2),VRAPP(I,2)
0103 IF(JAP.LT.J) GO TO 120
0104 140 IF(J.EQ.NG) GO TO 155
0105 WRITE(102,1095)
0106 IF(J.EQ.0) GO TO 170
0107 155 DO 160 I=1,J
0108 UROOT(KO+I)=UZROS(I)
0109 VROOT(KO+I)=VZROS(I)
0110 160 KD=KD+1
0111 IF(K.DEQ.NP) GO TO 1
0112 ND1=ND1+1
0113 IF(K.GE.NP) GO TO 1
0114 170 IF(J.JE.1) GO TO 200
0115 ND1=ND1+1
0116 DO 190 I=1,ND1
0117 UD(I)=UD(I)
0118 VD(I)=VD(I)
0119 190 KD=KD+1
0120 IF(J.JE.NN) GO TO 20
0121 WRITE(102,3000) (ASTER9I=1,331
0122 WRITE(012,1035)
0123 KKK=ND2+1
0124 200 IF(NODEQ.0) GO TO 1
0125 KD=KD+1
0126 DENOM=UD(2)*VD(2)*VD(2)*VD(2)
0127 210 WRITE(102,1001)
0128 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0129 ND1=ND2
0130 GO TO 20
0131 220 IF(Nentineq.0) GO TO 1
0132 KD=KD+1
0133 230 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0134 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0135 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0136 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0137 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0138 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0139 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0140 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0141 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0142 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0143 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0144 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0145 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J1.
0146 GO TO 1
0147 GO TO 1
0148 1025 FORMAT(1X,25HNO ROOTS OF MULTIPLICITY ,12//)
0149 1035 FORMAT(1X,25HTHE FOLLOWING POLYNOMIAL, GEX), CONTAINS ALL THE R
0150 1050 FORMAT(2X,5HROOT(I,12,4H)= ,D23.16,3H I=023.16,2H I,18X,12,9X,25HN
0151 1095 FORMAT(1X,5HNOT ALL ROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUN
TABLE H.III (Continued)

10//1

0152 1000 FORMAT(3(I2,1X,4X,1X,3(06.0,1X),20X,2(I7.0,1X)),11)
0153 1010 FORMAT(2(I30.0))
0154 1015 FORMAT(2(I30.0))
0155 1030 FORMAT(1X,22X) THE DEGREE OF P(X) IS ,12,22X THE COEFFICIENTS ARE//
0156 1040 FORMAT(2X,9A2,A2,4H)
0157 1100 FORMAT(2X,43,A2,4H) = \,D23.16,3H + ,D23.16,2H I)
0158 1180 FORMAT(///IX,22HTHE DEGREE OF P(X) IS
0159 1190 FORMAT(2X,5HROOTI,12,4H) = ,D23.16,3H + ,D23.16,2H II
0160 2000 FORMAT(1X,41XNUMBER OF INITIAL APPROXIMATIONS GIVEN. ,12)
0161 2010 FORMAT(1X,29XMAXIMUM NUMBER OF ITERATIONS. ,11X,13)
0162 2020 FORMAT(1X,21XTEST FOR CONVERGENCE.,11X,09.2)
0163 2040 FORMAT(1X,23XRADIUS TO START SEARCH.,11X,09.2)
0164 2050 FORMAT(1X,21XRADIUS TO END SEARCH.,11X,09.2)
0165 2060 FORMAT(1X,34XTEST FOR ZERO IN SUBROUTINE GCD.,09.2)
0166 2070 FORMAT(1X,34XTEST FOR ZERO IN SUBROUTINE QUAD.,09.2)
0167 2080 FORMAT(1X,32X)
0168 3000 FORMAT(/////1X,A3,32X)
0169 END
SUBROUTINE PROD(M,UF,VF,N,UG,VG,NN,UM,VH)

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

DOUBLE PRECISION UM,VM,UF,VF,UG,UG

DIMENSION UM(N+1),VM(N+1),UF(26),VF(26),UG(26),VG(26)

MN=M+N
KK=MN

DO 100 I=1,KK

K=I

UH(I)=O.O

VH(I)=O.O

IF(I.LE.M+1) GO TO 10

LIMIT=M+1

GO TO 20

10 LIMIT=I

GO TO 50

20 DO 50 J=L,LIMIT

IF(K.GT.N+L) GO TO 50

IF(K.JE.N+1) GO TO 40

GO TO 50

40 UH(I)=UH(I)+(UF(I)*UG(K)-VF(I)*VG(K))

VH(I)=VH(I)+(VF(I)*UG(K)+UF(I)*VG(K))

K=K-1

100 CONTINUE

RETURN

END
**TABLE H.III. (Continued)**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE QUAD(NUAVA,J,UROOT,VROOT)</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION EPS1, EPS2, EPSLON, UROOT, VROOT, UA, VA, UDISC, VDISC, UD</td>
</tr>
<tr>
<td>0003</td>
<td>DOUBLE PRECISION EPS4</td>
</tr>
<tr>
<td>0004</td>
<td>DIMENSION UROOT(251), VROOT(251), UA(261), VA(261)</td>
</tr>
<tr>
<td>0005</td>
<td>COMMON EPS1, EPS2, EPSLON, EPS4, 102, MAX</td>
</tr>
<tr>
<td>0006</td>
<td>IF (N.GT.1) GO TO 10</td>
</tr>
<tr>
<td>0007</td>
<td>J=J+1</td>
</tr>
<tr>
<td>0008</td>
<td>BBB=UA(2)*UA(2)+VA(2)*VA(2)</td>
</tr>
<tr>
<td>0009</td>
<td>UROOT(J)=-(UA(2)*UA(2)+VA(2)*VA(2))/BBB</td>
</tr>
<tr>
<td>0010</td>
<td>VROOT(J)=-((VA(2)*VA(2)+UA(2)*UA(2))/BBB</td>
</tr>
<tr>
<td>0011</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0012</td>
<td>UDISC=(UA(2)*UA(2)-VA(2)<em>VA(2))-4</em>(UA(3)*UA(1)-VA(3)*VA(1))</td>
</tr>
<tr>
<td>0013</td>
<td>VDISC=(2*UA(2)<em>VA(2))-4</em>(UA(3)*VA(1)+VA(3)*UA(1))</td>
</tr>
<tr>
<td>0014</td>
<td>UD=2*UA(3)</td>
</tr>
<tr>
<td>0015</td>
<td>VD=2*VA(3)</td>
</tr>
<tr>
<td>0016</td>
<td>DDD=SQRT(UDISC<em>UDISC+VDISC</em>VDISC)</td>
</tr>
<tr>
<td>0017</td>
<td>IF (DDD.LT.EPSLON) GO TO 20</td>
</tr>
<tr>
<td>0018</td>
<td>CALL COMSQRT(UDISC, VDISC, UTEMP, VTEMP)</td>
</tr>
<tr>
<td>0019</td>
<td>BBB+UD<em>UD+VD</em>VD</td>
</tr>
<tr>
<td>0020</td>
<td>UROOT(J)=(-UA(2)+UTEMP)*UD-(-VA(2)+VTEMP)*VD/BBB</td>
</tr>
<tr>
<td>0021</td>
<td>VROOT(J)=(-VA(2)+VTEMP)*UD-(-UA(2)+UTEMP)*VD/BBB</td>
</tr>
<tr>
<td>0022</td>
<td>UROOT(J+1)=(-UA(2)-UTEMP)*UD-(-VA(2)+VTEMP)*VD/BBB</td>
</tr>
<tr>
<td>0023</td>
<td>VROOT(J+1)=(-VA(2)-VTEMP)*UD-(-UA(2)-UTEMP)*VD/BBB</td>
</tr>
<tr>
<td>0024</td>
<td>J=J+2</td>
</tr>
<tr>
<td>0025</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0026</td>
<td>10 J=J+1</td>
</tr>
<tr>
<td>0027</td>
<td>BBB+UD<em>UD+VD</em>VD</td>
</tr>
<tr>
<td>0028</td>
<td>UROOT(J)=(-UA(2)*UD-VA(2)*VD/BBB</td>
</tr>
<tr>
<td>0029</td>
<td>VROOT(J)=(-VA(2)+UD+UA(2)*VD/BBB</td>
</tr>
<tr>
<td>0030</td>
<td>WRITE(102,1000) UROOT(J), VROOT(J)</td>
</tr>
<tr>
<td>0031</td>
<td>1000 FORMAT(/1X,11X=QUAD FOUND, ,D23.16,3H + ,D23.16,2H 1,22H TO BE A M</td>
</tr>
<tr>
<td>0032</td>
<td>100 RETURN</td>
</tr>
<tr>
<td>0033</td>
<td>END</td>
</tr>
</tbody>
</table>
#### TABLE H.III (Continued)

```plaintext
SUBROUTINE DERIVIN, UP, VP, M, UA, VA

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

DOUBLE PRECISION UP, VP, UA, VA

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

KKK = N + 1

DO 10 I = 2, KKK
   AAA = I - 1
   UA(AAA - 1) = AAA * UP((I)

10    VA(AAA - 1) = AAA * VP((I)

M = N - 1
RETURN
END
```
**TABLE H.III (Continued)**

```fortran
0001 SUBROUTINE GCD(UR,VR,MI,US,VS,MI,USS,VS)

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

0002 DOUBLE PRECISION USSS,SUSSS

0003 DOUBLE PRECISION UR,VR,US,VS,URR,VRR,DD,DD,VDD,EPS1,EPS2,EPS3,EPS4

0004 DIMENSION UR(26),VR(26),US(26),VS(26),URR(26),VRR(26)

0005 COMMON EPS1,EPS2,EPS3,EPS4,R,MAX

0006 MI=NN

0007 MI=M

0008 KKK=N+1

0009 DO 20 I=1,KKK

0010 URR(I)=UR(I)

0011 20 VRR(I)=VR(I)

0012 KKK=M+1

0013 DO 25 I=1,KKK

0014 USSII)=US(I)

0015 25 VSS(I)=VS(I)

0016 BBB=US(N+1)*US(N+1)+VS(N+1)*VS(N+1)

0017 DD=(URR(N+1)*VR(N+1)+VRR(N+1)*UR(N+1))/BBB

0018 VDD=(US(N+1)*VR(N+1)-VRR(N+1)*US(N+1))/BBB

0019 KKK=N+1-MI

0020 DO 40 I=KKK,N+1

0021 UT(I)=URR(I)-DD*US(I-N+1)-VDD*VS(I-N+1)

0022 40 VT(I)=VR(I)-DD*US(I-N+1)+VDD*VS(I-N+1)

0023 IF(MI.EQ.N+1) GO TO 70

0024 KKK=N+1-MI

0025 DO 20 I=1,KKK

0026 UT(I)=URR(I)

0027 60 VT(I)=VR(I)

0028 70 DO 90 I=N+1

0029 BBB=DSORT(UT(N+1-I)*UT(N+1-I)+VT(N+1-I)*VT(N+1-I))

0030 IF(BBB.GT.EPS1) GO TO 100

0031 CONTINUE

0032 DO 95 I=1,M

0033 BBB=US(M+1)*US(M+1)+VS(M+1)*VS(M+1)

0034 USSSS=(US(M+1)*US(M+1)+VS(M+1)*VS(M+1))/BBB

0035 VSSSS=(VS(M+1)*US(M+1)+VS(M+1)*VS(M+1))/BBB

0036 US(M)+USSSS

0037 VS(M)+VSSSS

0038 US(M+1)+0.0

0039 GO TO 200

0040 100 K=MI

0041 IF(K.EQ.0) GO TO 170

0042 IF(K.LT.MI) GO TO 140

0043 KKK=K+1

0044 DD=130 J=1,KKK

0045 URR(J)=UT(J)

0046 130 VRR(J)=VT(J)

0047 NI=K

0048 GO TO 30
```
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0050</td>
<td></td>
<td><strong>TABLE H-III</strong> (Continued)</td>
</tr>
<tr>
<td>140</td>
<td>KKK+1</td>
<td><strong>GO</strong> 150 J = 1, KKK</td>
</tr>
<tr>
<td>0051</td>
<td></td>
<td><strong>URR(J) = USS(J)</strong></td>
</tr>
<tr>
<td>0052</td>
<td></td>
<td><strong>VRR(J) = VSS(J)</strong></td>
</tr>
<tr>
<td>0053</td>
<td></td>
<td><strong>USS(J) = UT(J)</strong></td>
</tr>
<tr>
<td>150</td>
<td></td>
<td><strong>VSS(J) = VT(J)</strong></td>
</tr>
<tr>
<td>0056</td>
<td></td>
<td>KKK = K + 2</td>
</tr>
<tr>
<td>0057</td>
<td></td>
<td><strong>DO</strong> 160 J = KKK, NNN</td>
</tr>
<tr>
<td>0059</td>
<td></td>
<td><strong>URR(J) = USS(J)</strong></td>
</tr>
<tr>
<td>0060</td>
<td></td>
<td><strong>VRR(J) = VSS(J)</strong></td>
</tr>
<tr>
<td>0061</td>
<td></td>
<td><strong>NI = ML</strong></td>
</tr>
<tr>
<td>0062</td>
<td></td>
<td><strong>ML = K</strong></td>
</tr>
<tr>
<td>0063</td>
<td></td>
<td><strong>GO</strong> TO 30</td>
</tr>
<tr>
<td>170</td>
<td></td>
<td><strong>USS(J) = 1.0</strong></td>
</tr>
<tr>
<td>0065</td>
<td></td>
<td><strong>VSS(J) = 0.0</strong></td>
</tr>
<tr>
<td>0066</td>
<td></td>
<td><strong>ML = 0</strong></td>
</tr>
<tr>
<td>200</td>
<td></td>
<td><strong>RETURN</strong></td>
</tr>
<tr>
<td>0068</td>
<td></td>
<td><strong>END</strong></td>
</tr>
</tbody>
</table>
SUBROUTINE DIVIDE(N,UP,VP,UD,VO,K,UQ,VQ)

C ******************************************************
C *
C * GIVEN TWO POLYNOMIALS P(X) AND Q(X), SUBROUTINE DIVIDE COMPUTES THE *
C * QUOTIENT POLYNOMIAL H(X) = P(X)/Q(X).
C *
C ******************************************************

DOUBLE PRECISION UP,VP,UD,VO,UQ,VQ,UTERM,VTERM,UDUMMY
DIMENSION UP(26),VP(26),UD(26),VO(26),UQ(26)

K=N-M
UDUMMY=UD(M+1)*UD(M+2)+VD(M+1)
UQ(K+1)=(UP(N+1)*UD(M+2)+VP(N+1)*VQ(M+1))/UDUMMY
VQ(K+1)=(VP(N+1)*UD(M+2)+UP(N+1)*VD(M+1))/UDUMMY
IF(K.EQ.0) GO TO 100
J=1
DO 50 I=1,K
J=J+1
UTERM=UP(N-J)
VTERM=VP(N-J)
KK=K+1
NNN=M-J
DO 40 MI=NNN,M
IF(KK.GT.1) GO TO 10
GO TO 45
20 UTERM=UTERM-4(UQ(KK)*UD(M+1)-VQ(KK)*VD(M+1))
VTERM=VTERM-(UQ(KK)*VD(M+1)+VQ(KK)*UD(M+1))
40 KK=KK-1
45 UDUMMY=UD(M+1)*UD(M+2)+UD(M+1)*VD(M+1)
UQ(K+1)=UTERM*UD(M+1)-VTERM*VD(M+1))/UDUMMY
VQ(K+1)=VTERM*UD(M+1)-UTERM*VD(M+1))/UDUMMY
100 RETURN
END
SUBROUTINE CONSQT(UX, VX, UY, VY)
C
***************************************************************************
C
** THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER. **
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(VX) 10, 20, 30
10 UY = AAA
VY = -1.0*BBB
GO TO 100
20 IF(UY) 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 H.III (Continued)

0001 SUBROUTINE CALCIUX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VX3
0002                                                                                         
0003                                                                                         
0004                                                                                         
0005                                                                                         
0006                                                                                         
0007                                                                                         
0008                                                                                         
0009                                                                                         
0010                                                                                         
0011                                                                                         
0012                                                                                         
0013                                                                                         
0014                                                                                         
0015                                                                                         
0016                                                                                         
0017                                                                                         
0018                                                                                         
0019                                                                                         
0020                                                                                         
0021                                                                                         
0022                                                                                         
0023                                                                                         
0024                                                                                         
0025                                                                                         
0026                                                                                         
0027                                                                                         
0028                                                                                         
0029                                                                                         
0030                                                                                         
0031                                                                                         
0032                                                                                         
0033                                                                                         
0034                                                                                         
0035                                                                                         
0036                                                                                         
0037                                                                                         
0038                                                                                         
0039                                                                                         
0040                                                                                         
0041                                                                                         
0042                                                                                         
0043                                                                                         
0044                                                                                         

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

0002 DOUBLE PRECISION ARG1,ARG2
0003 DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UX1,VX1,UX2,VX2,UX3,VX3,UPX1,
0004 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0005 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0006 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0007 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0008 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0009 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0010 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0011 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0012 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0013 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0014 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0015 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0016 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0017 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0018 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0019 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0020 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0021 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0022 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0023 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0024 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0025 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0026 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0027 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0028 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0029 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0030 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0031 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0032 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0033 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0034 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0035 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0036 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0037 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0038 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0039 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0040 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0041 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0042 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0043 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,
0044 VX1,UX3,UX2,UX1,UX2,UX3,UX3,UPX1,

C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
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>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0045</td>
<td>ARG1 = UDEN1<em>UDEN1+VDEN1</em>VDEN1</td>
</tr>
<tr>
<td>0046</td>
<td>ARG2 = UDEN2<em>UDEN2+VDEN2</em>VDEN2</td>
</tr>
<tr>
<td>0047</td>
<td>AAA = DSQRT(ARG1)</td>
</tr>
<tr>
<td>0048</td>
<td>BBB = DSQRT(ARG2)</td>
</tr>
<tr>
<td>0049</td>
<td>IF AAA.LT.0 GO TO 10</td>
</tr>
<tr>
<td>0050</td>
<td>IF AAA.EQ.0.0 GO TO 60</td>
</tr>
<tr>
<td>0051</td>
<td>UAAA = -2.0*UC</td>
</tr>
<tr>
<td>0052</td>
<td>VAAA = 2.0*VC</td>
</tr>
<tr>
<td>0053</td>
<td>UQ4 = (UAAA<em>UDEN1+VAAA</em>VDEN1)/ARG1</td>
</tr>
<tr>
<td>0054</td>
<td>VQ4 = (VAAA<em>UDEN1-UAAA</em>VDEN1)/ARG1</td>
</tr>
<tr>
<td>0055</td>
<td>GO TO 50</td>
</tr>
<tr>
<td>0056</td>
<td>10 IF BBB.EQ.0.0 GO TO 60</td>
</tr>
<tr>
<td>0057</td>
<td>UAAA = -2.0*UC</td>
</tr>
<tr>
<td>0058</td>
<td>VAAA = 2.0*VC</td>
</tr>
<tr>
<td>0059</td>
<td>UQ4 = (UAAA<em>UDEN2+VAAA</em>VDEN2)/ARG2</td>
</tr>
<tr>
<td>0060</td>
<td>VQ4 = (VAAA<em>UDEN2-UAAA</em>VDEN2)/ARG2</td>
</tr>
<tr>
<td>0061</td>
<td>GO TO 50</td>
</tr>
<tr>
<td>0062</td>
<td>50 UX4 = UX3+(UH3<em>UQ4-VH3</em>VQ4)</td>
</tr>
<tr>
<td>0063</td>
<td>VX4 = VX3+(VH3<em>UQ4+UH3</em>VQ4)</td>
</tr>
<tr>
<td>0064</td>
<td>RETURN</td>
</tr>
<tr>
<td>0065</td>
<td>60 UQ4 = 1.0</td>
</tr>
<tr>
<td>0066</td>
<td>VQ4 = 0.0</td>
</tr>
<tr>
<td>0067</td>
<td>GO TO 50</td>
</tr>
<tr>
<td>0068</td>
<td>END</td>
</tr>
</tbody>
</table>
TABLE H.11 (Continued)

SUBROUTINE MULLER(HP, U, VA, NAPP, UAPP, VAPP, VRAPPS, XSTART, XEND, NORD, VRAPPS, VRAPPS)

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

* 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 ZEROS 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 UPX, VPX3, UPX2, UROOT, XROOT, U1, VX, UAPP, VAPP
1, UX2, VX2, UVORK, VXORK, UX3, VX3, UB, UX4, VX4, U, VA, UPX1, VPX1, UAPP, V

ZAPP, UPX4, VPX4, EPSRT, EPS0, EPS, XCM, EPSH, VB3, VX3, UB4, VX4, ABPR3

3, UB4, XSTART, XEND

DIMENSION UROOT(25), VRAPPS(25), Multiply(25), VAPP(25, 1), VRAPPS(25, 2), VWORK

LOGICAL CNN

COMMON EPSM, EPS, EPS0, EPSRT, 101, MAX

DATA PNAME, PNAME/2HP1, 12HD1/

EPSM=0.0000

EPSRT=0.099

NROOT=0

UROOT=0

IPATH=1

NOMUL=0

ALTER=0

ITIME=0

IF (UAPP .LT. 0.0) GO TO 10

NAPP=NP

CALL GENAPP(UAPP, VAPP, NAPP, XSTART)

GO TO 27

DO 25 1=1, NAPP

UAPP(1, 1)=0.9*UAPP(1, 2)

VAPP(1, 1)=0.9*VAPP(1, 2)

UAPP(1, 2)=1.1*UAPP(1, 2)

VAPP(1, 2)=1.1*VAPP(1, 2)

25 VAPP(1, 2)=1.1*VAPP(1, 2)

27 KK=NPP1

GO 30 T=1, KK

UWORK(1)=U(1)

VWORK(1)=V(1)

NWORK=NP

UX1=UAPP(IAPP, 1)

UX2=UAPP(IAPP, 2)

UX3=UAPP(IAPP, 3)

VX1=VAPP(IAPP, 1)

VX2=VAPP(IAPP, 2)

VX3=VAPP(IAPP, 3)

CALL HORNEM(WORK, UWORK, VXORK, UX1, VX1, UX, VA, UPX1, VPX1)

CALL HORNEM(WORK, UWORK, VXORK, UX2, VX2, UB, VX2, VPX2)

CALL HORNEM(WORK, UWORK, VXORK, UX3, VX3, UB, VX3, VPX3)

50 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, VPX2, VPX2, UPX3, VPX3, UX

14, VX4, VX4, VX4, UB, VX4, VPX4)

CALL HORNEM(WORK, UWORK, VXORK, UX4, VX4, UB, VX4, VPX4)

ABPX3=39D0, (UPX4, VPX4, VPX4, VPX4, VPX4, VPX4, VPX4, VPX4, VPX4, VPX4, VPX4)}
TABLE H. II (Continued)

0044 IF(ABPX3.EQ.0.0) GO TO 70
0045 QQQ=ABPX4/ABPX3
0046 IF(QQQ.LE.10.) GO TO 70
0047 UQ4=0.5*UQ4
0048 VX4=VX3+(UH3*UQ4+VH3*VQ4)
0049 VX=VX3+(UH3*UQ4+VH3*VQ4)
0050 GO TO 80
0051 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0052 IF(ITER.LT.MAX) GO TO 110
0053 CALL ALTER(UAPP(IAPP),VAPP(IAPP,1),VAPPIAPP,2,URAPP,VRAPP,NALTER,ITIME)
0054 IF(NALTER.GT.5) GO TO 75
0055 ITER=1
0056 GO TO 40
0057 70 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0058 IF(XEND.EQ.0.0) GO TO 77
0059 IF(XSTART.GT.XEND) GO TO 77
0060 NAPP=NP
0061 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0062 NROOT=NP
0063 GO TO 100
0064 77 WRITE(102,1090)
0065 KKK=NWORK+1
0066 WRITE(102,1035)
0067 WRITE(102,1035)
0068 WRITE(102,1035)
0069 WRITE(102,1035)
0070 WRITE(102,1035)
0071 IF(IPATH.EQ.1) GO TO 82
0072 CALL BETTER(UA,VA,NP,UROOTVROOTNROOT,URAPPVRAPP,NROOT,MULT)
0073 RETURN
0074 IF(NROOT.EQ.0) GO TO 90
0075 IF(IROOT.EQ.0) GO TO 85
0076 WRITE(102,1040)
0077 DD 55 1,1,IROOT
0078 WRITE(102,1085) 1,UROOTII),VROOT(I),URAPP(I),VRAPP(I)
0079 IF(IROOT.LT.NROOT) GO TO 85
0080 WRITE(102,1040)
0081 IF(IROOT.EQ.0) GO TO 90
0082 WRITE(102,1085) 1,UROOTII),VROOT(I),I=KKK,NROOT
0083 87 IF(IPATH.EQ.1) GO TO 81
0084 RETURN
0085 WRITE(102,1070) NROOTY
0086 RETURN
0087 100 IAPP=IAPP+L
0088 RETURN
0089 NALTER=0
0090 GO TO 40
0091 120 NROOT=NROOT+1
0092 IF(NROOT.EQ.0) RETURN
0093 NROOT=NROOT+1
0094 NROOT=NROOT+1
0095 NROOT=NROOT+1
0096 NROOT=NROOT+1
0097 URAPP(NROOT,1)=UAPP(IAPP,1)
0098 VRAPP(NROOT,1)=VAPP(IAPP,1)
0099 VRAPP(NROOT,2)=APP(IAPP,2)
0100 VRAPP(NROOT,2)=APP(IAPP,2)
TABLE H.III (Continued)

0101 URAPPINROOT,3) = URAPP(IAPP,3)
0102 VRAPP(NROOT,3) = VAPP(IAPP,3)
0103 IF (NOMULT.LT.NP) GO TO 130
0104 NWORK = NWORK - 1
0105 KKK = NWORK + 1
0106 DO 140 (I=1.KKK)
0107 UWORK(I) = UB(I)
0108 VWORK(I) = VB(I)
0109 CALL HORNER(NWORK,UX4,VX4,UB,VPX4)
0110 CCC = DSORT(UPX4*UPX4 + VPX4*VPX4)
0111 IF (CCC.LT.EPSM) GO TO 150
0112 IF (NWORK.GT.2) GO TO 75
0113 IROOT = NROOT
0114 KKK = NWORK + 1
0115 DO 145 (I=1.KKK)
0116 UB(I) = UWORK(KKK+1-I)
0117 VB(I) = VWORK(KKK+1-I)
0118 CALL QUAD(NWORK,UB,VB,NROOT,UBOR,VBOR)
0119 GO TO 80
0120 MULT(NROOT) = MULT(NROOT+1)
0121 NOMULT = NOMULT + 1
0122 GO TO 125
0123 UX1 = UX2
0124 VX1 = VX2
0125 UX2 = UX3
0126 VX2 = VX3
0127 UX3 = UX4
0128 VX3 = VX4
0129 UPX1 = UPX2
0130 VPX1 = VPX2
0131 UPX2 = UPX3
0132 VPX2 = VPX3
0133 UPX3 = UPX4
0134 VPX3 = VPX4
0135 ITER = ITER + 1
0136 GO TO 50
0137 FORMAT(///,LX,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROSES WERE FOUND///)
0138 FORMAT(///IX,13HROOTS OF G(X),83X,21HINITIAL APPROXIMATION///)
0139 FORMAT(//,43HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER,12)
0140 FORMAT(2X,5HROOT,,I2,4H) = ,023.16,3H + ,023.16,2H
0141 FORMAT(3K,A2,12,4H) = ,023.16,3H + ,023.16,2H
0142 FORMAT(1D7.0,1) + ,023.16,2H
0143 FORMAT(I3,I2,13,8X,3I8.0,13X,13X,2I07.0,1X,11)
0144 FORMAT(I3,I2,13,8X,3I8.0,13X,13X,2I07.0,1X,11)
0145 FORMAT(I3,I2,13,8X,3I8.0,13X,13X,2I07.0,1X,11)
0146 END
SUBROUTINE GENAPP(APPR,APPI,NAPP,XSTART)

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

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

DIMENSION APPR(25,3), APPI(25,3)

COMMON EPSM, EPS1, EPS2, EPS3, IO2, MAX

IF(XSTART.EQ.0.0) XSTART=0.5

BETA=0.2617994

DO 10 I=1,NAPP
  APPR(I,2)=XSTART*DCOS(BETA)
  APPI(I,2)=XSTART*DSIN(BETA)
  BETA=BETA+0.5235988
  XSTART=XSTART+0.5
10    CONTINUE

DO 20 I=1,NAPP
  APPR(I,1)=0.9*APPR(I,2)
  APPR(I,2)=0.9*APPI(I,2)
  APPR(I,3)=1.1*APPR(I,2)
  APPI(I,3)=1.1*APPI(I,2)
20    CONTINUE

RETURN

END
TABLE H. III (Continued)

```
0001 SUBROUTINE ALTER(X1R, X11, X2R, X21, X3R, X31, NALTER, TIME)
0002 C ***********************************************************************
0003 C * SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO *
0004 C * CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
0005 C ***********************************************************************
0006 0007 DOUBLE PRECISION X1R, X11, X2R, X21, X3R, X31, EPS1, EPS2, EPS3, R, BETA
0008 COMMON EPS1, EPS2, EPS3, IO2, MAX
0009 IF( TIME .NE. 0.1 ) GO TO 5
0010 ITEM = 1
0011 WRITE( IO2, 1010 ) MAX
0012 5 IF( NALTER .EQ. 0 ) GO TO 10
0013 WRITE( IO2, 1000 ) X1R, X11, X2R, X21, X3R, X31
0014 GO TO 20
0015 NALTER = NALTER + 1
0016 RETURN
0017 20 X2R = X2R - R
0018 X21 = X21 - R
0019 GO TO 50
0020 50 R = DSQRT( X1R * X2R * X21 * X21 )
0021 X2R = X2R - DSQRT( BETA )
0022 X21 = X21 - DSQRT( BETA )
0023 X1R = 0.99 * X1R
0024 X11 = 0.99 * X11
0025 X3R = 1.1 * X3R
0026 X31 = 1.1 * X31
0027 RETURN
0028 1000 FORMAT( 1X, 5H X1 = , 0D23.16, 3H + , 0D23.16, 2H I, 10X, 2H ALTERED APPROXIM
0029 ATIONS/IX, 5H X2 = , 0D23.16, 3H + , 0D23.16, 2H I/I, 5H X3 = , 0D23.16, 3H +
0030 Z , 0D23.16, 2H I/I )
0031 1020 FORMAT( 1H0, 5H X1 = , 0D23.16, 3H + , 0D23.16, 2H I, 10X, 2H INITIAL APPROX
0032 IMATIONS/IX, 5H X2 = , 0D23.16, 3H + , 0D23.16, 2H I/I, 5H X3 = , 0D23.16, 3H +
0033 Z , 0D23.16, 2H I/I )
0034 1010 FORMAT( 1X, 5H NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
0035 TER/15, 12H ITERATIONS/ )
0036 END
```
TABLE H. III. (Continued)

0046 40 UX1=UX2
0047 VX1=VX2
0048 UX2=UX3
0049 VX2=VX3
0050 UX3=UX4
0051 VX3=VX4
0052 UX1=UX2
0053 VX1=VX2
0054 UX2=UX3
0055 VX2=VX3
0056 ITER=ITER+1
0057 GO TO 20
0058 50 L=L+1
0059 UROOT(L)=UX4
0060 VROOT(L)=VX4
0061 CONTINUE
0062 IF(L.LE.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(9H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(I,12,4H) = ,
0071 1023.16,3H + ,023.16,2H I/120H DID NOT CONVERGE AFTER ,13,11H ITERATIONS)
0072 1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,023.16,3H + ,023.16,2H I/120H)
0073 END
TABLE H.3III (Continued)

0001

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

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

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

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

0002

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

DENOM

0003

DOUBLE PRECISION EPSM

0004

LOGICAL CONV

0005

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

0006

UDUMMY-UX4-UX3

0007

VDUMMY=VX4-VX3

0008

AAA=DSPRT(UDUMMY*UDUMMY*VJMMY*VJMMY)

0009

DENOM=DSQRT(UX4+UX3+VX4+VX3)

0100

IF(DENOM.LT.EPSO) GO TO 20

0110

IF(AAA/DENOM.LT.EPS) GO TO 10

0120

5 CONV=.FALSE.

0130

GO TO 100

0140

10 CONV=.TRUE.

0150

GO TO 100

0160

20 IF(AAA.LT.EPSO) GO TO 10

0170

GO TO 5

0180

100 RETURN

0190

END

0001

SUBROUTINE HORNER(NA, UA, VA, UX, VX, UB, VB, UFX, VFX)

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

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

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

0002

DOUBLE PRECISION UX, VX, UFX, VFX, UB, VB, UA, VA

0003

DIMENSION UA(241), VA(241), UB(126), VB(126)

0004

UB(1) = UA(1)

0005

VB(1) = VA(1)

0006

NUM=NA+1

0007

gO 10 I=2,NUM

0008

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

0009

10 VB(I) = VB(I)+VFX(I)*UX-UB(I)*VX

0100

UFX=VFX(NUM)

0110

RETURN

0113

END