MULTIPLE ZEROS OF POLYNOMIALS

by

CRAIG A. WOOD

Department of Mathematics
Oklahoma State University
Stillwater, Oklahoma

A National Aeronautics and Space Administration
Research Grant
Grant Number NASA NCR 37-002-084

FINAL
MULTIPLE ZEROS OF POLYNOMIALS

by

CRAIG A. WOOD

Department of Mathematics
Oklahoma State University
Stillwater, Oklahoma

A National Aeronautics and Space Administration
Research Grant
Grant Number NASA 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 appendicies A through H.

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

<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>
<tr>
<td>Chapter</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Improving Zeros Found</td>
<td>91</td>
</tr>
<tr>
<td>Solving Quadratic Polynomial</td>
<td>91</td>
</tr>
<tr>
<td>Missing Roots</td>
<td>92</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>92</td>
</tr>
<tr>
<td><strong>APPENDIX B - NEWTON'S METHOD</strong></td>
<td>100</td>
</tr>
<tr>
<td>Use of the Program</td>
<td>100</td>
</tr>
<tr>
<td>Input Data for Newton's Method</td>
<td>102</td>
</tr>
<tr>
<td>Variables Used in Newton's Method</td>
<td>107</td>
</tr>
<tr>
<td>Description of Program Output</td>
<td>107</td>
</tr>
<tr>
<td>Informative and Error Message</td>
<td>109</td>
</tr>
<tr>
<td>Flow Charts</td>
<td>120</td>
</tr>
<tr>
<td>Program Listing</td>
<td>126</td>
</tr>
<tr>
<td><strong>APPENDIX C - MULLER'S METHOD</strong></td>
<td>135</td>
</tr>
<tr>
<td>Use of the Program</td>
<td>135</td>
</tr>
<tr>
<td>Input Data for Muller's Method</td>
<td>137</td>
</tr>
<tr>
<td>Variables Used in Muller's Method</td>
<td>137</td>
</tr>
<tr>
<td>Description of Program Output</td>
<td>138</td>
</tr>
<tr>
<td>Informative and Error Messages</td>
<td>139</td>
</tr>
<tr>
<td>Flow Charts</td>
<td>144</td>
</tr>
<tr>
<td>Program Listing</td>
<td>150</td>
</tr>
<tr>
<td><strong>APPENDIX D - SPECIAL FEATURES OF THE G.C.D. AND THE REPEATED G.C.D. PROGRAMS</strong></td>
<td>163</td>
</tr>
<tr>
<td>Generating Approximations</td>
<td>163</td>
</tr>
<tr>
<td>Altering Approximations</td>
<td>165</td>
</tr>
<tr>
<td>Searching the Complex Plane</td>
<td>166</td>
</tr>
<tr>
<td>Improving Zeros Found</td>
<td>168</td>
</tr>
<tr>
<td>Solving Quadratic Polynomial</td>
<td>168</td>
</tr>
<tr>
<td>Missing Roots</td>
<td>169</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>169</td>
</tr>
<tr>
<td><strong>APPENDIX E - G.C.D. - NEWTON'S METHOD</strong></td>
<td>171</td>
</tr>
<tr>
<td>Use of the Program</td>
<td>171</td>
</tr>
<tr>
<td>Input Data for G.C.D. - Newton's Method</td>
<td>173</td>
</tr>
<tr>
<td>Variables Used in G.C.D. - Newton's Method</td>
<td>178</td>
</tr>
<tr>
<td>Description of Program Output</td>
<td>188</td>
</tr>
<tr>
<td>Informative and Error Messages</td>
<td>189</td>
</tr>
<tr>
<td>Flow Charts</td>
<td>191</td>
</tr>
<tr>
<td>Program Listing</td>
<td>200</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>APPENDIX F - G.C.D. - MULLER'S METHOD</td>
<td>214</td>
</tr>
<tr>
<td>Use of the Program</td>
<td>214</td>
</tr>
<tr>
<td>Input Data for G.C.D. - Muller's Method</td>
<td>216</td>
</tr>
<tr>
<td>Variables Used in G.C.D. - Muller's Method</td>
<td>216</td>
</tr>
<tr>
<td>Description of Program Output</td>
<td>216</td>
</tr>
<tr>
<td>Informative and Error Messages</td>
<td>216</td>
</tr>
<tr>
<td>Flow Charts</td>
<td>222</td>
</tr>
<tr>
<td>Program Listing</td>
<td>234</td>
</tr>
<tr>
<td>APPENDIX G - REPEATED G.C.D. - NEWTON'S METHOD</td>
<td>253</td>
</tr>
<tr>
<td>Use of the Program</td>
<td>253</td>
</tr>
<tr>
<td>Input Data for Repeated G.C.D. - Newton's Method</td>
<td>255</td>
</tr>
<tr>
<td>Variables Used in Repeated G.C.D. - Newton's Method</td>
<td>255</td>
</tr>
<tr>
<td>Description of Program Output</td>
<td>255</td>
</tr>
<tr>
<td>Informative and Error Messages</td>
<td>256</td>
</tr>
<tr>
<td>Flow Charts</td>
<td>260</td>
</tr>
<tr>
<td>Program Listing</td>
<td>270</td>
</tr>
<tr>
<td>APPENDIX H - REPEATED G.C.D. - MULLER'S METHOD</td>
<td>286</td>
</tr>
<tr>
<td>Use of the Program</td>
<td>286</td>
</tr>
<tr>
<td>Input Data for Repeated G.C.D. - Muller's Method</td>
<td>287</td>
</tr>
<tr>
<td>Variables used in Repated G.C.D. - Muller's Method</td>
<td>288</td>
</tr>
<tr>
<td>Description of Program Output</td>
<td>288</td>
</tr>
<tr>
<td>Informative and Error Messages</td>
<td>289</td>
</tr>
<tr>
<td>Flow Charts</td>
<td>290</td>
</tr>
<tr>
<td>Program Listing</td>
<td>301</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

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

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

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 $X_1', X_2', \ldots, X_n'$ 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

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

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

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

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

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

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

followed by

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\[
b_1 = a_1
\]
\[
b_i = a_i + db_{i-1} \quad (i = 2, 3, \ldots, N+1).
\]

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

\[
c_1 = b_1
\]
\[
c_j = b_j + dc_{j-1} \quad (j = 2, 3, \ldots, N).
\]

Then $P(d) = b_{N+1}$ and $P'(d) = c_N$. The elements $b_1, b_2, \ldots, b_N$ are the coefficients of the polynomial $H(X)$ in Theorem 2.2 when $P(X)$ is divided by $(X - d)$. 
These formulas are derived in [3, PP. 106-107]. Thus with equation (2-2) and the iteration formulas of the previous theorem, Newton's method can now be applied to generate the sequence (2-3) which will converge to the root, \(a\), 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

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

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

After a zero, \(a\), of \(P(X)\) has been found, the term \((X - a)\) is synthetically divided out of \(P(X)\) by deflation using Theorem 2.3 obtaining
a polynomial, $P_1(X)$, of degree $N-1$. The root finding process is then repeated to extract a zero, $a_1$, of $P_1(X)$. $P(X)$ can be written as

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

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

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

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

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

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

respectively, are extracted. Each $\alpha_i$ ($i = 0, 1, 2, \ldots, N-1$) is a zero of $P(X)$ since each $\alpha_1$ is a zero of $P_{i-1}(X), P_{i-2}(X), \ldots, P_1(X), P(X)$.

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

2. Convergence of Newton's Method

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

**Theorem 2.4.** Let $P(X)$ be a polynomial and let the following conditions be satisfied on the closed interval $[a, b]$:
1. \( P(a) \cdot P(b) < 0 \)
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 [1, PP. 31-33].

3. Procedure for Newton's Method

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

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

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

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

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

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

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

4. Geometrical Interpretation of Newton’s Method

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

5. Determining Multiple Roots

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

$$P(X) = a_i (X - a_i)^{e_1} (X - a_2)^{e_2} \ldots (X - a_m)^{e_m}, \quad (m < N)$$

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

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

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

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

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

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

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

In order to simplify this expression, introduce the quantities

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

Then

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Now let

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

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

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

Then

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

and

\[
Q(X_n + qh_n) = \frac{A_n q^2 + B_n q + C_n}{q_n+1}.
\]
Solving the quadratic equation \( Q(X_n + qh_n) = 0 \) and denoting the result by \( q_{n+1} \) gives:

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

and the new approximation is found as follows:

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

Thus

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

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

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

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

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

(3-3)

The sign in the denominator should be chosen such that the magnitude of the denominator is largest, thus causing \(|q_{n+1}| \) to be smallest. This, in turn, will make \( X_{n+1} \) closest to \( X_n \).
Note that each iteration of this process requires three approximations, $X_{n-2}, X_{n-1}, X_n$, in order to compute $X_{n+1}$. Thus, when $X_{n+1}$ is found, $X_{n-1}, X_n, X_{n+1}$ are used to compute $X_{n+2}$; that is, the last three terms of the generated sequence are used to compute the next term.

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

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

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 obtained in the \((n+1)\)th step of the iterative process is proportional to the square of the error obtained in the nth iteration. However, no general proof of convergence has been obtained for Muller's method. It has produced convergence in the majority of the cases tested.

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

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

2. Procedure for Muller's Method

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

1. Compute \(h_n, q_n, D_n, B_n, C_n, q_{n+1}\) as defined previously.

2. Compute the next approximation \(X_{n+1}\) by

   \[X_{n+1} = X_n + h_n q_{n+1}^4\]

3. Test for convergence; that is, test

   \[\left|\frac{X_{n+1} - X_n}{X_{n+1}}\right| < \varepsilon\]

   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 $N^{th}$ 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
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

$$D(X) = (X - a_1)^{e_1} (X - a_2)^{e_2} \cdots (X - a_m)^{e_m}$$

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

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

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

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

Consider the polynomial $H(X)$ obtained by dividing $P(X)$ by its monic g.c.d., $D(X)$. 
\[ H(X) = P(X)/D(X) \]
\[ = \sum_{i=1}^{m} \frac{a_i}{(X - a_i)} \]
\[ = a_1 \prod_{i=1}^{m} (X - a_i). \]

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

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

\[ R_0(X) = R_1(X) S_1(X) + R_2(X) \]

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

\[ R_1(X) = S_2(X) R_2(X) + R_3(X) \]

where either \( R_3(X) = 0 \) or deg. \( R_3(X) < \) deg. \( R_2(X) \). Continuing in the above manner, suppose \( R_1(X) \) and \( R_{i+1}(X) \) have been found where deg. \( R_{i+1}(X) < \) deg. \( R_i(X) \). Then there exists polynomials \( R_{i+2}(X) \) and \( S_{i+1}(X) \) such that
\[ R_i(X) = R_{i+1}(X) S_{i+1}(X) + R_{i+2}(X) \]

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

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

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

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

where $N_i \leq N_i'$. Define $U_1(X)$ by

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

Then define $T_1(X)$ by

$$T_1(X) = R_{i-1}(X) - U_1(X) R_i(X)$$

$$= [r_{i-1,1} - r_{i,1} (r_{i-1,1}/r_{i,1})] X^{N_i-1}$$

$$+ [r_{i-1,2} - r_{i,2} (r_{i-1,1}/r_{i,1})] X^{N_i-1}$$

$$+ \ldots$$

$$+ [r_{i-1,N_i-1+1} - r_{i,N_i-1+1} (r_{i-1,1}/r_{i,1})]$$

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

We consider three cases.

(1) If $T_1(X) = 0$, then $R_i(X) = R_k(X)$; 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 \( \deg T_1(X) = M_1 \geq N_1 \), then define \( U_2(X) \) by

\[
U_2(X) = (t_{1,1}/r_{i,1})^{M_1-N_1} X^{N_i}
\]

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 \( T_2(X) = T_1(X) - U_2(X) R_1(X) \) which can be expressed by

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

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

\[
+ \ldots
\]

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

where \( r_{i,j} = 0 \) for \( j > N_1+i \). We again consider the following three cases.

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

(2) If \( T_2(X) \neq 0 \) and \( \deg T_2(X) < \deg R_1(X) \), then

\( R_{i+1}(X) = T_2(X) \).

(3) If \( T_2(X) \neq 0 \) and \( \deg T_2(X) = M_2 \geq N_1 \), then define \( U_3(X) \) by

\[
U_3(X) = (t_{2,1}/r_{i,1})^{M_2-N_1} X^{N_i}
\]
where

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

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

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

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

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

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

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

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

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

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

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

1. Given a polynomial, $P(X)$, in the form
   
   $$P(X) = a_1x^N + a_2x^{N-1} + \ldots + a_Nx + a_{N+1}.$$  

2. Calculate the derivative, $P'(X)$, of $P(X)$ in the form
   
   $$P'(X) = b_1x^{N-1} + b_2x^{N-2} + \ldots + b_N$$  
   
   where $b_1 = N a_1$,  
   
   $b_2 = (N-1)a_2, \ldots, b_N = a_N$.

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

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

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

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

REPEATED GREATEST COMMON DIVISOR METHOD

1. Derivation of the Algorithm

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

Let

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

where \( a_1 \neq 0 \), each \( a_i \) is a complex number, and \( \alpha_1, \alpha_2, \ldots, \alpha_m \) are the
distinct zeros of \( P(X) \) having multiplicity \( e_1, e_2, \ldots, e_m \), respectively.

If \( D_1(X) \) is the monic greatest common divisor of \( P(X) \) and \( P'(X) \), then
Theorem 4.1 shows that

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

where we assume that if \( e_j = 1 \), then \( X - \alpha_j \) does not appear in the
representation. Let $D_2(X)$ be the monic greatest common divisor of $D_1(X)$ and $D'_1(X)$. Then

$$D_2(X) = (X - a_1)\, e_1^{-2} \times (X - a_2)\, e_2^{-2} \times \cdots \times (X - a_m)\, e_m^{-2}$$

where we assume that if $e_j \leq 2$, then $X - a_j$ 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

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

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

$$G_i(X) = \frac{D_{i-1}(X)}{D_i(X)} / \frac{D_i(X)}{D_{i+1}(X)}$$

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

has simple zeros and they are just the zeros of $P(X)$ that have multiplicity $i$. Thus, we have proven the following theorem.
Theorem 5.1. Let \( P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1} \) where \( a_1 \neq 0 \) and \( a_1, a_2, \ldots, a_{N+1} \) are complex numbers. If \( D_0(X) = P(X) \) and if \( D_{i+1}(X) \) is the monic greatest common divisor of \( D_i(X) \) and \( D'_i(X) \) for \( i > 0 \), 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
\[
P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_NX + a_{N+1}.
\]

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

3. Calculate the derivative, \( D'_0(X) \), of \( D_0(X) \) in the form
\[
D'_0(X) = b_1X^{M-1} + b_2X^{M-2} + \ldots + b_M
\]
where \( \text{deg. } D_0(X) = M, \) \( D_0(X) = d_1X^M + \ldots + d_{M+1} \),
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+i (6)</td>
</tr>
<tr>
<td>1+2i (2)</td>
<td>.003 (2)</td>
<td>1+2i (2)</td>
<td>1-i (6)</td>
</tr>
<tr>
<td>-1+3i (1)</td>
<td>i (2)</td>
<td>-1.5i (2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.5i (2)</td>
<td>3i (3)</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 \( X_n \rightarrow c \), \( P(X_n) \rightarrow 0 \) and \( P'(X_n) \rightarrow 0 \) 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.
**Newton's Method to Find Zeros of Polynomials**

**The Coefficients of \( p(x) \) Are**

<table>
<thead>
<tr>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>( P_3 )</th>
<th>( P_4 )</th>
<th>( P_5 )</th>
<th>( P_6 )</th>
<th>( P_7 )</th>
<th>( P_8 )</th>
<th>( P_9 )</th>
<th>( P_{10} )</th>
<th>( P_{11} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
<td>(-0.20000000000000000)</td>
</tr>
</tbody>
</table>

**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:** 0.200 - 00

**Radius to End Search:** 0.200 - 00

**Before the Attempt to Improve Accuracy, the Zeros of \( p(x) \) Are**

| ROOT 1 | \( 0.30000000000000000 \) | \( 0.186098515370364818 \) | \( 1 \) |
| ROOT 2 | \( 0.40000000000000000 \) | \( 0.00000000000000000 \) | \( 0 \) |
| ROOT 3 | \( 0.50000000000000000 \) | \( -0.20000000000000000 \) | \( 0 \) |

**Exhibit 6.1.**
**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 the polynomial are:

<table>
<thead>
<tr>
<th>Coefficients</th>
<th>Multiplicities</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3000000000000000</td>
<td>1</td>
</tr>
<tr>
<td>-0.1790000000000000</td>
<td>1</td>
</tr>
<tr>
<td>0.3010000000000000</td>
<td>1</td>
</tr>
<tr>
<td>-0.1576000000000000</td>
<td>1</td>
</tr>
<tr>
<td>0.1745000000000000</td>
<td>1</td>
</tr>
<tr>
<td>-0.7542000000000000</td>
<td>1</td>
</tr>
<tr>
<td>0.8274900000000000</td>
<td>1</td>
</tr>
<tr>
<td>-0.3069000000000000</td>
<td>1</td>
</tr>
<tr>
<td>0.1329000000000000</td>
<td>1</td>
</tr>
<tr>
<td>-0.5411800000000000</td>
<td>1</td>
</tr>
<tr>
<td>0.7225700000000000</td>
<td>1</td>
</tr>
<tr>
<td>-0.3276990000000000</td>
<td>1</td>
</tr>
<tr>
<td>0.1241470000000000</td>
<td>1</td>
</tr>
<tr>
<td>-0.4097280000000000</td>
<td>1</td>
</tr>
<tr>
<td>0.0000000000000000</td>
<td>1</td>
</tr>
</tbody>
</table>

**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**

Exhibit 6.2.
AFTER THE ATTEMPT TO IMPROVE ACCURACY

<table>
<thead>
<tr>
<th>ROOTS OF P(x)</th>
<th>MULTICIPICITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ROOT} (1) \approx 0.10000000000000000 )</td>
<td>1</td>
<td>0.12940924045100000</td>
</tr>
<tr>
<td>( \text{ROOT} (2) \approx 0.20000000000000000 )</td>
<td>1</td>
<td>0.13710837096539000</td>
</tr>
<tr>
<td>( \text{ROOT} (3) \approx 0.49999999999999999 )</td>
<td>1</td>
<td>0.38807965492200000</td>
</tr>
<tr>
<td>( \text{ROOT} (4) \approx 0.10000000000000000 )</td>
<td>1</td>
<td>0.41630790568200000</td>
</tr>
<tr>
<td>( \text{ROOT} (5) \approx 0.20000000000000000 )</td>
<td>1</td>
<td>0.51288634358200000</td>
</tr>
<tr>
<td>( \text{ROOT} (6) \approx 0.33333333333333333 )</td>
<td>1</td>
<td>0.62535910777800000</td>
</tr>
<tr>
<td>( \text{ROOT} (7) \approx 0.50000000000000000 )</td>
<td>1</td>
<td>0.75000000000000000</td>
</tr>
<tr>
<td>( \text{ROOT} (8) \approx 0.12500000000000000 )</td>
<td>1</td>
<td>0.87500000000000000</td>
</tr>
<tr>
<td>( \text{ROOT} (9) \approx 0.20000000000000000 )</td>
<td>1</td>
<td>0.93750000000000000</td>
</tr>
<tr>
<td>( \text{ROOT} (10) \approx 0.31250000000000000 )</td>
<td>1</td>
<td>1.06250000000000000</td>
</tr>
<tr>
<td>( \text{ROOT} (11) \approx 0.40000000000000000 )</td>
<td>1</td>
<td>1.18750000000000000</td>
</tr>
<tr>
<td>( \text{ROOT} (12) \approx 0.50000000000000000 )</td>
<td>1</td>
<td>1.31250000000000000</td>
</tr>
<tr>
<td>( \text{ROOT} (13) \approx 0.62500000000000000 )</td>
<td>1</td>
<td>1.43750000000000000</td>
</tr>
<tr>
<td>( \text{ROOT} (14) \approx 0.75000000000000000 )</td>
<td>1</td>
<td>1.56250000000000000</td>
</tr>
<tr>
<td>( \text{ROOT} (15) \approx 0.87500000000000000 )</td>
<td>1</td>
<td>1.68750000000000000</td>
</tr>
</tbody>
</table>

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

\[
P_{10} = 0.1000000000000000D + 0.0000000000000000D 01
\]

\[
P_{11} = -0.7000000000000001D + 0.1050000000000000D 02
\]

\[
P_{12} = -0.2000000000000000D + 0.5800000000000001D 02
\]

\[
P_{13} = 0.1710000000000000D + 0.1500000000000000D 01
\]

\[
P_{14} = -0.73000000000000000D + 0.2510000000000000D 03
\]

\[
P_{15} = 0.7200000000000001D + 0.1040000000000000D 03
\]

Number of initial approximations given: 0

Maximum number of iterations: 200

Test for convergence: 0.10D-09

Test for multiplicities: 0.10D-01

Radius to start search: 0.00D 00

Radius to end search: 0.000 00

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

Roots of \( P(x) \)

Multiplicities

Initial approximation

Root(1) = \( 0.9999998836129190D 09 \) + \( 0.2000000000000000D 28 \)

Root(2) = \( 0.1999992907503309D 01 \) + \( 0.1999959474001689D 01 \)

Root(3) = \( -0.9999999999998000 \) + \( 0.0000000000000000 \)

Solved by direct method

In the attempt to improve accuracy, Root(2) = \( 0.1999959474001689D 01 \) did not converge.

The present approximation after 200 iterations is printed below.

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

Roots of \( P(x) \)

Multiplicities

Initial approximation

Root(1) = \( 0.9999998836129190D 09 \) + \( 0.2000000000000000D 28 \)

Root(2) = \( 0.1999992907503309D 01 \) + \( 0.1999959474001689D 01 \)

Root(3) = \( -0.9999999999998000 \) + \( 0.0000000000000000 \)

Solved by direct method

Exhibit 6.3. Roots are: \( 2+2i \) (3), \( 1+2i \) (2), \( -1.5i \)
NEWTONS METHOD TO FIND ROOTS OF POLYNOMIALS

POLYNOMIAL NUMBER 2 OF DEGREE 15

THE COEFFICIENTS OF PI(x) ARE

\[ P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \]

\[ P(1) = a_0 + a_1 + a_2 + \cdots + a_n = 0 \]

\[ P(2) = 2^0 a_0 + 2^1 a_1 + 2^2 a_2 + \cdots + 2^n a_n = 0 \]

\[ P(3) = 3^0 a_0 + 3^1 a_1 + 3^2 a_2 + \cdots + 3^n a_n = 0 \]

\[ \vdots \]

\[ P(n) = n^0 a_0 + n^1 a_1 + n^2 a_2 + \cdots + n^n a_n = 0 \]

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

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

ROOTS OF PI(X) MULTIPlicITIES INITIAL APPROXIMATION

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicity</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2603946127400000</td>
<td>0</td>
<td>0.1294954243981700</td>
</tr>
<tr>
<td>0.1289436043080000</td>
<td>1</td>
<td>0.2394632158993100</td>
</tr>
<tr>
<td>0.2165249649480000</td>
<td>2</td>
<td>0.4294632158993100</td>
</tr>
<tr>
<td>0.1000080173622988</td>
<td>3</td>
<td>0.1000080173622988</td>
</tr>
<tr>
<td>0.4137366230400000</td>
<td>4</td>
<td>0.4137366230400000</td>
</tr>
<tr>
<td>0.7071067811866000</td>
<td>5</td>
<td>0.7071067811866000</td>
</tr>
<tr>
<td>0.2755620000000000</td>
<td>6</td>
<td>0.2755620000000000</td>
</tr>
<tr>
<td>0.1093899227670000</td>
<td>7</td>
<td>0.1093899227670000</td>
</tr>
<tr>
<td>0.4800000000000000</td>
<td>8</td>
<td>0.4800000000000000</td>
</tr>
<tr>
<td>0.9665441989282835</td>
<td>9</td>
<td>0.9665441989282835</td>
</tr>
<tr>
<td>0.2333333333333333</td>
<td>10</td>
<td>0.2333333333333333</td>
</tr>
<tr>
<td>0.1000000000000000</td>
<td>11</td>
<td>0.1000000000000000</td>
</tr>
<tr>
<td>0.2000000000000000</td>
<td>12</td>
<td>0.2000000000000000</td>
</tr>
</tbody>
</table>

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF PI(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)
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 3 OF DEGREE 8

THE COEFFICIENTS OF P(x) ARE

\[ p_0 = 0.10000000000000000 \]
\[ p_1 = -0.30000000000000000 \]
\[ p_2 = 0.20000000000000000 \]
\[ p_3 = 0.42000000000000000 \]
\[ p_4 = 0.30750000000000000 \]
\[ p_5 = -0.51750000000000000 \]
\[ p_6 = 0.49250000000000000 \]
\[ p_7 = -0.44200000000000000 \]
\[ p_8 = 0.18100000000000000 \]
\[ p_9 = 0.15800000000000000 \]

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.10-05
TEST FOR MULTIPlicITIES: 0.10D-01
RADIUS TO START SEARCH: 0.00000
RADIUS TO END SEARCH: 0.00000
BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(x) ARE

ROOT(1) = 0.9999929476004472D 00 + 0.1999998467904090D 01
ROOT(2) = -0.9999929476004472D 00 + 0.5027396405227564D 00
ROOT(3) = 0.9999929476004472D 00 - 0.1294095284438187D 00
ROOT(4) = 0.9999929476004472D 00 - 0.3882284792654056D 00
ROOT(5) = 0.9999929476004472D 00 + 0.1448888763117193D 00
ROOT(6) = -0.1015484227697537D 01 + 0.4945588881101592D 00

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

ROOT(1) = 0.9999929476004472D 00 + 0.1999998467904090D 01
ROOT(2) = -0.9999929476004472D 00 + 0.5027396405227564D 00
ROOT(3) = 0.9999929476004472D 00 - 0.1294095284438187D 00
ROOT(4) = 0.9999929476004472D 00 - 0.3882284792654056D 00
ROOT(5) = 0.9999929476004472D 00 + 0.1448888763117193D 00
ROOT(6) = -0.1015484227697537D 01 + 0.4945588881101592D 00

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

POLYNOMIAL NUMBER 4 OF DEGREE 12

THE COEFFICIENTS OF P(X) ARE

| P(1) | 0.1000000000000000 00 01 |
| P(2) | -0.1200000000000000 00 02 |
| P(3) | 0.0700000000000000 00 03 |
| P(4) | -0.2400000000000000 00 04 |
| P(5) | 0.7200000000000000 00 05 |
| P(6) | -0.2800000000000000 00 06 |
| P(7) | 0.2640000000000000 00 07 |
| P(8) | -0.3264000000000000 00 08 |
| P(9) | 0.3120000000000000 00 09 |
| P(10) | -0.2240000000000000 00 10 |
| P(11) | 0.1152000000000000 00 11 |
| P(12) | -0.3840000000000000 00 12 |
| P(13) | 0.6400000000000001 00 13 |

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

| ROOT(1) | 0.9994079803369152 00 + 0.9941591489643941 00 1 |
| ROOT(2) | 0.1020096885609720 01 + -0.5420646130387889 00 1 |
| ROOT(3) | 0.1002721106681658 01 + 0.1023310661526634 01 1 |
| ROOT(4) | 0.6189968959361400 00 + -0.7949702013736144 00 1 |
| ROOT(5) | 0.9851422047423414 00 + -0.1395675981313038 01 1 |
| ROOT(6) | 0.6515828215165421 00 + -0.1206987666454936 01 1 |
| ROOT(7) | 0.1393799984400508 01 + -0.8271924311176513 00 1 |
| ROOT(8) | 0.1330620199428514 01 + -0.1232093496230941 01 1 |

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(1) = 0.9994079803369152 00 1 DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(5) = 0.9851422047423414 00 1 DID NOT CONVERGE.
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

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

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

THE COEFFICIENTS OF $P(X)$ ARE

$P(1) = 0.1000000000000000 \times 10^1 + 0.0000000000000000 \times 10^0$

$P(2) = -0.7000000000000001 \times 10^1 + 0.1050000000000000 \times 10^0$

$P(3) = -0.2800000000000000 \times 10^2 + 0.5800000000000000 \times 10^1$

$P(4) = 0.1710000000000000 \times 10^2 - 0.3510000000000000 \times 10^1$

$P(5) = -0.2250000000000000 \times 10^2 + 0.4200000000000000 \times 10^1$

$P(6) = -0.7000000000000000 \times 10^2 - 0.2280000000000000 \times 10^1$

$P(7) = 0.7200000000000000 \times 10^2 - 0.1040000000000000 \times 10^1$

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0

MAXIMUM NUMBER OF ITERATIONS: 200

TEST FOR CONVERGENCE: 0.10D-09

TEST FOR MULTIPLICITIES: 0.10D-01

RADIUS TO START SEARCH: 0.00D 00

RADIUS TO END SEARCH: 0.000 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF $P(X)$

$P(1) = 0.1999995488874981 \times 10^1 + 0.2000017733363912 \times 10^0$

$P(2) = 0.1000000159779278 \times 10^1 + 0.1999999979600045 \times 10^0$

$P(3) = 0.9866348122507864 \times 10^0 + 0.2003806882325521 \times 10^0$

$P(4) = -0.1000002106173084 \times 10^1 - 0.4999768344218609 \times 10^0$

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF $P(X)$

$P(1) = 0.1999995488874981 \times 10^1 + 0.2000017733363912 \times 10^0$

$P(2) = 0.1000000159779278 \times 10^1 + 0.1999999979600045 \times 10^0$

$P(3) = 0.9866348122507864 \times 10^0 + 0.2003806882325521 \times 10^0$

$P(4) = -0.1000002106173084 \times 10^1 - 0.4999768344218609 \times 10^0$

Exhibit 6.7. Roots Are: 2+2i (3), 1+2i (2), -1+5i
THE COEFFICIENTS OF PI(X) ARE

<table>
<thead>
<tr>
<th>PI(1)</th>
<th>0.48000000000000000000 02</th>
<th>0.00000000000000000000 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI(2)</td>
<td>0.25542000000000000000 02</td>
<td>-0.39400000000000000000 03</td>
</tr>
<tr>
<td>PI(3)</td>
<td>-0.71255568000000000000 02</td>
<td>-0.21846960000000000000 04</td>
</tr>
<tr>
<td>PI(4)</td>
<td>-0.30556564000000000000 04</td>
<td>-0.96488415600000000000 04</td>
</tr>
<tr>
<td>PI(5)</td>
<td>-0.17333096000000000000 05</td>
<td>0.16026597200000000000 05</td>
</tr>
<tr>
<td>PI(6)</td>
<td>-0.64679897200000000000 01 01</td>
<td>-0.17638574640000000000 05</td>
</tr>
<tr>
<td>PI(7)</td>
<td>-0.10227965221300000000 06</td>
<td>-0.03064442320000000000 04</td>
</tr>
<tr>
<td>PI(8)</td>
<td>-0.16457420000000000000 06</td>
<td>0.01377663000000000000 05</td>
</tr>
<tr>
<td>PI(9)</td>
<td>-0.20344265888000000000 06</td>
<td>0.19380022750000000000 06</td>
</tr>
<tr>
<td>PI(10)</td>
<td>-0.18172597800000000000 06</td>
<td>0.19296034000000000000 06</td>
</tr>
<tr>
<td>PI(11)</td>
<td>-0.12714997208000000000 06</td>
<td>0.21713412740000000000 06</td>
</tr>
<tr>
<td>PI(12)</td>
<td>-0.29546971660000000000 05</td>
<td>0.19284987296000000000 06</td>
</tr>
<tr>
<td>PI(13)</td>
<td>0.13293454340000000000 05</td>
<td>0.10331322650000000000 06</td>
</tr>
<tr>
<td>PI(14)</td>
<td>0.30595107740000000000 05</td>
<td>0.29949914430000000000 05</td>
</tr>
<tr>
<td>PI(15)</td>
<td>0.61355949200000000000 03</td>
<td>0.18763216000000000000 03</td>
</tr>
<tr>
<td>PI(16)</td>
<td>0.27556200000000000000 00</td>
<td>0.27556200000000000000 00</td>
</tr>
</tbody>
</table>

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
M A X I M U M N U M B E R O F I T E R A T I O N S. 200
T E S T F O R C O N V E R G E N C E. 0.100-01
T E S T F O R M U L T I P L I C I T I E S. 0.100-01
R A D I U S T O S T A R T S E A R C H. 0.000 00
R A D I U S T O E N D S E A R C H. 0.000 00

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF PI(X) M U L T I P L I C I T I E S I N I T I A L A P P R O X I M A T I O N

| ROOT 1 | 0.3000000000562593720-02 02 | 0.8277931932153550-12 12 |
| ROOT 2 | 0.2804698716629-05 05 | -0.9499498729600000 01 |
| ROOT 3 | 0.6016878124161-01 01 | -0.29993646561890 01 |
| ROOT 4 | -0.18864593170430-04 04 | 0.49999165971620 00 |
| ROOT 5 | -0.23333333333333 00 00 | 0.23333333333333 00 00 |
| ROOT 6 | -0.00000000000000 00 00 | -0.28284628017860 00 |
| ROOT 7 | -0.10000000000000 01 01 | 0.10000000000000 01 01 |
| ROOT 8 | 0.14999999999999 01 01 | 0.29999999999999 01 01 |
| ROOT 9 | 0.99999999999999 01 01 | 0.99999999999999 01 01 |
| ROOT 10 | 0.3249409628400000 00 00 | 0.3249409628400000 00 00 |
| ROOT 11 | 0.11646848013699 01 01 | 0.11646848013699 01 01 |
| ROOT 12 | 0.29999999999999 01 01 | 0.29999999999999 01 01 |
| ROOT 13 | 0.3249409628400000 00 00 | 0.3249409628400000 00 00 |
| ROOT 14 | 0.11646848013699 01 01 | 0.11646848013699 01 01 |
| ROOT 15 | 0.29999999999999 01 01 | 0.29999999999999 01 01 |
| ROOT 16 | 0.11646848013699 01 01 | 0.11646848013699 01 01 |

AFTER ATTEMPT TO IMPROVE ACCURACY

Exhibit 6.8.
<table>
<thead>
<tr>
<th>ROOTS OF P(X)</th>
<th>MULTIPLICITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDOT(1) = -0.3000000056255190-02</td>
<td>+</td>
<td>0.3279370689255688-12</td>
</tr>
<tr>
<td>RDOT(2) = 0.9001086172562740-07</td>
<td>+</td>
<td>0.1000000043859274-01</td>
</tr>
<tr>
<td>RDOT(3) = 0.6189553790319120-07</td>
<td>+</td>
<td>0.1500000014679398-00</td>
</tr>
<tr>
<td>RDOT(4) = 0.2397527310798233-04</td>
<td>+</td>
<td>0.3000000873725504-03</td>
</tr>
<tr>
<td>RDOT(5) = 0.3797864564043244-07</td>
<td>+</td>
<td>0.1000000004245680-01</td>
</tr>
<tr>
<td>RDOT(6) = -0.2333333333333333-02</td>
<td>+</td>
<td>-0.3027956195429183-03</td>
</tr>
<tr>
<td>RDOT(7) = -0.1000000006600000-00</td>
<td>+</td>
<td>-0.9999999992572520-00</td>
</tr>
<tr>
<td>RDOT(8) = 0.9289422544363060-08</td>
<td>+</td>
<td>-0.1500000001312862-01</td>
</tr>
<tr>
<td>RDOT(9) = 0.1099000026510385-00</td>
<td>+</td>
<td>-0.1000000015749923-02</td>
</tr>
<tr>
<td>RDOT(10) = 0.1671714273636300-07</td>
<td>+</td>
<td>-0.1500000018522429-01</td>
</tr>
<tr>
<td>RDOT(11) = -0.2598765613057310-04</td>
<td>+</td>
<td>0.3000000216037187-05</td>
</tr>
<tr>
<td>RDOT(12) = 0.3110926108089250-04</td>
<td>+</td>
<td>0.3000001490736689-01</td>
</tr>
<tr>
<td>RDOT(13) = -0.5453799366602740-07</td>
<td>+</td>
<td>0.1499999637957010-01</td>
</tr>
</tbody>
</table>

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

THE COEFFICIENTS OF P(X) ARE

\[ P(1) = 0.1000000000000000D+01 + 0.0000000000000000D+00 \]
\[ P(2) = -0.5175000000000000D+02 + 0.0000000000000000D+02 \]
\[ P(3) = 0.3075000000000000D+03 + 0.3475000000000000D+03 \]
\[ P(4) = -0.4452500000000000D+03 + 0.0000000000000000D+03 \]
\[ P(5) = 0.2102500000000000D+03 + 0.4247500000000001D+03 \]
\[ P(6) = -0.3827500000000000D+03 + 0.4247500000000001D+03 \]
\[ P(7) = 0.1810000000000000D+03 + 0.4420000000000001D+03 \]
\[ P(8) = 0.1810000000000000D+03 + 0.6000000000000001D+01 \]

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.100D-09
TEST FOR MULTIPLEITS: 0.100-01
RADIUS TO START SEARCH: 0.000000
RADIUS TO END SEARCH: 0.000000
BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)

<table>
<thead>
<tr>
<th>ROOT</th>
<th>MULTIPLECTIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>0.48296291136656279D+00 + 0.1294095286438179D+00</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.7071067503346340D+01 + 0.7071067503346340D+01</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.3999999999999999D+01 + 0.4247500000000001D+03</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.7071067503346340D+01 + 0.7071067503346340D+01</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.48296291136656279D+00 + 0.1294095286438179D+00</td>
</tr>
</tbody>
</table>

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)

<table>
<thead>
<tr>
<th>ROOT</th>
<th>MULTIPLECTIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>0.48296291136656279D+00 + 0.1294095286438179D+00</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.7071067503346340D+01 + 0.7071067503346340D+01</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.3999999999999999D+01 + 0.4247500000000001D+03</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.7071067503346340D+01 + 0.7071067503346340D+01</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.48296291136656279D+00 + 0.1294095286438179D+00</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

\[ P(0) = 0.100000000000000000 01 \]
\[ P(1) = -0.120000000000000000 02 \]
\[ P(2) = 0.720000000000000000 02 \]
\[ P(3) = -0.560000000000000000 03 \]
\[ P(4) = 0.780000000000000000 04 \]
\[ P(5) = -0.160000000000000000 04 \]
\[ P(6) = 0.720000000000000000 04 \]
\[ P(7) = -0.280000000000000000 04 \]
\[ P(8) = 0.700000000000000000 04 \]
\[ P(9) = -0.224000000000000000 04 \]
\[ P(10) = 0.700000000000000000 04 \]
\[ P(11) = -0.163200000000000000 04 \]
\[ P(12) = 0.262400000000000000 04 \]
\[ P(13) = -0.326400000000000000 04 \]
\[ P(14) = 0.312000000000000000 04 \]
\[ P(15) = -0.224000000000000000 04 \]
\[ P(16) = 0.700000000000000000 04 \]

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. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00
BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)

MULTIPLEINITIAL

ROOT 1) = 0.10044681929487390 01
\[ 0.10021861175327510 01 \]
\[ 0.48296291156562790 00 \]
\[ 0.12940952844381870 00 \]

ROOT 2) = 0.98589784284950020 00
\[ 0.10068045189960200 01 \]
\[ 0.70710675530463460 00 \]
\[ 0.70710680706845950 00 \]

ROOT 3) = 0.99622622484739470 00
\[ 0.98443823578951680 00 \]
\[ 0.38822847926540560 00 \]
\[ 0.14488887631171930 01 \]

ROOT 4) = 0.10070694905203640 01
\[ -0.82784262477528560 02 \]
\[ -0.51763825519667240 00 \]
\[ 0.19318516083687550 01 \]

ROOT 5) = 0.75845181074234980 00
\[ -0.94103202050523650 01 \]
\[ -0.17647771470870100 01 \]
\[ 0.17647771470870100 01 \]

ROOT 6) = 0.84390135441641700 00
\[ -0.12151582772901180 01 \]
\[ -0.2897777583049900 01 \]
\[ 0.77645674639870700 00 \]

ROOT 7) = 0.12454795861515510 01
\[ -0.96097310967704480 00 \]
\[ 0.70710675530463460 00 \]
\[ 0.70710680706845950 00 \]

ROOT 8) = 0.11380314281571030 01
\[ -0.12271385678935710 01 \]
\[ 0.70710675530463460 00 \]
\[ 0.70710680706845950 00 \]

EXHIBIT 6.10.

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF P(X)

MULTIPLEINITIAL

ROOT 1) = 0.10044681929487390 01
\[ 0.10021861175327510 01 \]
\[ 0.48296291156562790 00 \]
\[ 0.12940952844381870 00 \]

ROOT 2) = 0.98589784284950020 00
\[ 0.10068045189960200 01 \]
\[ 0.70710675530463460 00 \]
\[ 0.70710680706845950 00 \]

ROOT 3) = 0.99622622484739470 00
\[ 0.98443823578951680 00 \]
\[ 0.38822847926540560 00 \]
\[ 0.14488887631171930 01 \]

ROOT 4) = 0.10070694905203640 01
\[ -0.82784262477528560 02 \]
\[ -0.51763825519667240 00 \]
\[ 0.19318516083687550 01 \]

ROOT 5) = 0.75845181074234980 00
\[ -0.94103202050523650 01 \]
\[ -0.17647771470870100 01 \]
\[ 0.17647771470870100 01 \]

ROOT 6) = 0.84390135441641700 00
\[ -0.12151582772901180 01 \]
\[ -0.2897777583049900 01 \]
\[ 0.77645674639870700 00 \]

ROOT 7) = 0.12454795861515510 01
\[ -0.96097310967704480 00 \]
\[ 0.70710675530463460 00 \]
\[ 0.70710680706845950 00 \]

EXHIBIT 6.10.
Exhibit 6.10. Roots Are: 1+i (6), 1-i (6)
GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 1

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.10D-02
TEST FOR CONVERGENCE. 0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.10D-19
TEST FOR MULTIPLEITIES. 0.10D-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 I
p(6) = -0.7000000000000001D 01 + 0.1050000000000000 02 I
p(5) = -0.28000000000000000 02 + 0.58000000000000010 02 I
p(4) = -0.1750000000000000D 03 + 0.15000000000000000 03 I
p(3) = -0.71300000000000000 03 + 0.2510000000000000 03 I
p(2) = -0.2280000000000000 03 + 0.10400000000000000 03 I
p(1) = 0.7200000000000000 02 + 0.10400000000000000 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 + 0.45000000000000000 01 I
q(2) = -0.70000000000000000 01 + 0.3500000000000000 01 I
q(1) = 0.9999999999999982D 00 + 0.6999999999998120 01 I

ROOTS OF Q(X)

ROOT( 1) = 0.9999999999999982D 00 + 0.6999999999998120 01 I
ROOT( 2) = 0.2000000000000000D 01 + 0.3500000000000000 01 I
ROOT( 3) = -0.2000000000000000D 01 + 0.45000000000000000 01 I

INITIAL APPROXIMATION

0.4829629115656279D 00 + 0.1294095284438187D 00 I

RESULTS OF SUBROUTINE QUAD

ROOTS OF P(X)

ROOT( 1) = 0.9999999999999982D 00 + 0.6999999999998120 01 I
ROOT( 2) = 0.2000000000000000D 01 + 0.3500000000000000 01 I
ROOT( 3) = -0.2000000000000000D 01 + 0.45000000000000000 01 I

INITIAL APPROXIMATION

0.4829629115656279D 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: 0.100-10
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 15
THE COEFFICIENTS ARE:

\[ P(16) = 0.48000000000000000D+02 \]
\[ P(15) = 0.21571200000000000D+03 \]
\[ P(14) = -0.73535568000000000D+03 \]
\[ P(13) = -0.38555656960000000D+04 \]
\[ P(12) = -0.17333864648000000D+05 \]
\[ P(11) = -0.49679892704000000D+05 \]
\[ P(10) = -0.10223945221300000D+06 \]
\[ P(9) = -0.16427422005600000D+06 \]
\[ P(8) = -0.20366258884200000D+06 \]
\[ P(7) = -0.18712557800100000D+06 \]
\[ P(6) = -0.12749972985900000D+06 \]
\[ P(5) = -0.28146927168000000D+05 \]
\[ P(4) = 0.13294344348000000D+05 \]
\[ P(3) = 0.30539007747000000D+05 \]
\[ P(2) = -0.18358990200000000D+03 \]
\[ P(1) = 0.27556200000000000D+00 \]

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

\[ Q(7) = 0.48000000000000000D+02 \]
\[ Q(6) = 0.15985599999590830D+03 \]
\[ Q(5) = 0.2675199999866287D+03 \]
\[ Q(4) = 0.32719599997B1904D+03 \]
\[ Q(3) = 0.2301599996044553D+02 \]
\[ Q(2) = -0.7207200004799688D+02 \]
\[ Q(1) = 0.2268000133550231D+01 \]

ROOTS OF O(X):

| ROOT 1 | 0.30000000000000000D-02 + 0.1370543176638413D-09 i |
| ROOT 2 | 0.1466635543509054D-09 + 0.10000000000018463D-03 i |
| ROOT 3 | 0.1626206809956170D-09 + 0.14400000000026803D-04 i |
| ROOT 4 | 0.11670221945839-10 + 0.15000000000067630D+01 i |

INITIAL APPROXIMATIONS:

| ROOT 1 | 0.482962911566279D+00 + 0.129409528438167D+00 i |
| ROOT 2 | 0.107106407066459D+00 + 0.707106407066459D+00 i |
| ROOT 3 | 0.38823470364586D+01 + 0.144888876171930D+01 i |
| ROOT 4 | -0.2170327190007240D+00 + 0.198189380386755D+01 i |

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

POLYNOMIAL NUMBER 3

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.10D-02
TEST FOR CONVERGENCE. 0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.10D-19
TEST FOR MULTIPLE VALUES. 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

\[
P(0) = 0.1000000000000000 + 0.0000000000000000 00 01
P(7) = -0.5179500000000000 + 0.8646250000000000 00 02
P(6) = 0.3105000000000000 + 0.3479500000000000 00 03
P(5) = -0.4925000000000000 + 0.4948750000000000 00 03
P(4) = 0.3937500000000000 + 0.2475000000000000 00 03
P(3) = -0.1012500000000000 + 0.4420000000000000 00 03
P(2) = 0.1250000000000000 + 0.1572500000000000 00 03
P(1) = 0.1580000000000000 + 0.4000000000000000 00 03
P(0) = 0.1000000000000000 + 0.1999999999999999 00 01
\]

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

\[
Q(0) = 0.1000000000000000 + 0.0000000000000000 00 01
Q(2) = -0.2000000000000000 + 0.4948750000000000 00 01
Q(1) = -0.7000000000000000 + 0.3500000000000000 00 01
Q(0) = 0.9999999999999999 + 0.7000000000000000 00 00
\]

ROOTS OF P(X):

\[
\begin{array}{l}
\text{ROOT(1)} = 0.999999999999954830 00 + 0.1999999999997559 00 01 1
\text{ROOT(2)} = 0.1000000000000000 + 0.2999999999999590 00 01 1
\text{ROOT(3)} = -0.99999999999998140 00 + 0.4499999999999690 00 01 1
\end{array}
\]

ROOTS OF Q(X):

\[
\begin{array}{l}
\text{ROOT(1)} = 0.999999999999954830 00 + 0.1999999999997559 00 01 1
\text{ROOT(2)} = 0.1000000000000000 + 0.2999999999999590 00 01 1
\text{ROOT(3)} = -0.99999999999998140 00 + 0.4499999999999690 00 01 1
\end{array}
\]

RESULTS OF SUBROUTINE QUAD
MULTIPLICITIES

\[
\begin{array}{l}
\text{MULTIPLICITIES: 2 3 3}
\end{array}
\]

INITIAL APPROXIMATION

\[
\begin{array}{l}
\text{ROOT(1)} = 0.999999999999954830 00 + 0.1999999999997559 00 01 1
\text{ROOT(2)} = 0.1000000000000000 + 0.2999999999999590 00 01 1
\text{ROOT(3)} = -0.99999999999998140 00 + 0.4499999999999690 00 01 1
\end{array}
\]

RESULTS OF SUBROUTINE QUAD

\[
\begin{array}{l}
\text{RESULTS OF SUBROUTINE QUAD: 2 3 3}
\end{array}
\]

Exhibit 6.13. Roots Are: 2+2i (3), 1+2i (2), -1+5i (3)
GREATEST COMMON DIVISOR METHOD USED WITH NEWTONS METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 4

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.100-02
TEST FOR CONVERGENCE: 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.100-12
RADIUS TO START SEARCH: 0.00D 00
RADIUS TO END SEARCH: 0.00D 00

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

P(13) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
P(12) = 0.1200000000000000D 01 + 0.0000000000000000D 00 I
P(11) = 0.7000000000000000D 02 + 0.0000000000000000D 00 I
P(10) = 0.2800000000000000D 03 + 0.0000000000000000D 00 I
P(9) = 0.7900000000000000D 03 + 0.0000000000000000D 00 I
P(8) = 0.1630000000000000D 04 + 0.0000000000000000D 00 I
P(7) = 0.2624000000000000D 04 + 0.0000000000000000D 00 I
P(6) = 0.3264000000000000D 04 + 0.0000000000000000D 00 I
P(5) = 0.3120000000000000D 04 + 0.0000000000000000D 00 I
P(4) = -0.2240000000000000D 04 + -0.0000000000000000D 00 I
P(3) = -0.1632000000000000D 04 + -0.0000000000000000D 00 I
P(2) = 0.2624000000000000D 04 + 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(3) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
Q(2) = 0.2000000000000000D 01 + 0.0000000000000000D 00 I
Q(1) = -0.1999999999999999D 00 + 0.0000000000000000D 00 I
Q(0) = 0.1999999999999999D 00 + 0.0000000000000000D 00 I

ROOTS OF P(X) MULTIPlicITIES

ROOT 11 = 0.1000000000000000D 01 + 0.9999999999999974D 00 I 6
ROOT 21 = 0.1000000000000000D 01 + -0.9999999999999974D 00 I 6

RESULTS OF SUBROUTINE QUAD

Exhibit 6.14. Roots Are: 1+i (6), 1-i (6)
Greatest Common Divisor Method Used with Muller'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: 0.100-02
Test for convergence: 0.100-09
Test for zero in subroutine QUAD: 0.10D-19
Test for multiplicities: 0.10D-01
Radius to start search: 0.000 00
Radius to end search: 0.000 00

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

\[
P(7) = 0.1000000000000000D+01 + 0.0000000000000000D+00
\]
\[
P(6) = -0.7000000000000000D+01 + -0.1050000000000000D+02
\]
\[
P(5) = -0.2800000000000000D+02 + 0.5800000000000001D+02
\]
\[
P(4) = 0.17100000000000000 03 + 0.1500000000000000D 01 1
\]
\[
P(3) = -0.7300000000000000D+02
\]
\[
P(2) = -0.2280000000000000D+03
\]
\[
P(1) = -0.2280000000000000D+03
\]

The degree of Q(x) is the polynomial which has as its roots the distinct roots of P(x). The coefficients are:

\[
Q(4) = 0.1000000000000000D+01 + 0.0000000000000000D+00
\]
\[
Q(3) = -0.2000000000000000D+01 + -0.45000000000000001D+01
\]
\[
Q(2) = -0.7300000000000000D+01 + 0.3500000000000000D+00
\]
\[
Q(1) = 0.9999999999977520 00 + 0.699999999998120 D 01
\]

Roots of P(x)

Root 1: 0.9999999999997537D+00 + 0.1999999999999474D+01
Root 2: 0.2000000000000000D+01 + 0.2000000000000000D+01
Root 3: -0.9999999999999534D+00 + 0.5000000000000000D+00

Exhibit 6.15. Roots Are: 2+2i (3), 1+2i (2), -1+5i
GREAT S COMMON DIVISOR METHOD USED WITH MULLER'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: 0.10E-02
TEST FOR CONVERGENCE: 0.10E-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.10E-19
TEST FOR MULTIPLEILITIES: 0.10E-01
RADIUS TO START SEARCH: 0.00D 00
RADIUS TO END SEARCH: 0.00D 00

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

| P(0) | = 0.4800000000000000D 02 + 0.0000000000000000D 00 1 |
| P(1) | = 0.1598559999959083D 03 + -0.1440000000018463D 03 1 |
| P(2) | = 0.2670199999962637D 03 + -0.2575600000068238D 03 1 |
| P(3) | = 0.3271999999978194D 03 + -0.9164600000075942D 03 1 |
| P(4) | = 0.2301999999946953D 02 + -0.1521350000037168D 02 1 |
| P(5) | = 0.2707600000047468D 02 + -0.152749999996559D 02 1 |
| P(6) | = 0.7257840000054156D 03 + -0.7529040000033821D 03 1 |
| P(7) | = 0.226800000003350231D 01 + 0.2267099999982630D 01 1 |

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

| Q(0) | = 0.4800000000000000D 02 + 0.0000000000000000D 00 1 |
| Q(1) | = 0.1598559999959083D 03 + -0.1440000000018463D 03 1 |
| Q(2) | = 0.2670199999962637D 03 + -0.2575600000068238D 03 1 |
| Q(3) | = 0.3271999999978194D 03 + -0.9164600000075942D 03 1 |
| Q(4) | = 0.2301999999946953D 02 + -0.1521350000037168D 02 1 |
| Q(5) | = 0.2707600000047468D 02 + -0.152749999996559D 02 1 |
| Q(6) | = 0.7257840000054156D 03 + -0.7529040000033821D 03 1 |
| Q(7) | = 0.226800000003350231D 01 + 0.2267099999982630D 01 1 |

ROOTS OF Q(X)

| ROOT | 11 | = 0.300000000538459970-CZ + -0.1370553778161480-09 |
| ROOT | 21 | = 0.4806292115565279D 00 + 0.1296095234308170D 00 1 |
| ROOT | 31 | = 0.707106753963460D 00 + 0.707106753963460D 00 1 |
| ROOT | 41 | = -0.05173825591606724D 00 + 0.493618683627550D 01 1 |

Exhibit 6.16.
Exhibit 6.16. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2) 3i (3), -1-i (3)
GREATEST COMMON DIVISOR METHOD USED WITH MILLER'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 | 0.150-19 |
| TEST FOR MULTIPLE ZEROS | 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) | = 0.100000000000000000000000 01 + 0.000000000000000000000000 01 |
| P(8) | = -0.50000000000000000000000 01 + 0.000000000000000000000000 01 |
| P(7) | = -0.317500000000000000000003 02 + 0.430000000000000000000003 02 |
| P(6) | = 0.172500000000000000000003 03 + 0.446250000000000000000003 03 |
| P(5) | = 0.100000000000000000000003 03 + 0.347500000000000000000003 03 |
| P(4) | = 0.487500000000000000000003 03 + 0.247500000000000000000003 03 |
| P(3) | = 0.181000000000000000000003 03 + 0.442000000000000000000003 03 |
| P(2) | = 0.150000000000000000000003 03 + 0.600000000000000000000003 03 |
| P(1) | = 0.1580000000000000000000000000 03 + 0.6000000000000000000000000 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) | = 0.1000000000000000000000000000 01 + 0.0000000000000000000000000000 01 |
| Q(3) | = -0.2000000000000000000000000000 01 + 0.4500000000000000000000000000 01 |
| Q(2) | = 0.7000000000000000000000000000 01 + 0.3500000000000000000000000000 01 |
| Q(1) | = 0.9999999999999999999999999999 01 + 0.7000000000000000000000000000 01 |

ROOTS OF Q(x) INITIAL APPROXIMATION

| ROOT(1) | = 0.0429729156562790 00 + 0.1294095234381870 00 |
| SOLVED BY DIRECT METHOD |
| ROOT(2) | = 0.2000000000000000000000000000 01 + 0.2000000000000000000000000000 01 |
| SOLVED BY DIRECT METHOD |
| ROOT(3) | = -0.9999999999999775 00 + 0.4999999999999998 00 |

ROOTS OF P(x) INITIAL APPROXIMATION

| ROOT(1) | = 0.0429729156562790 00 + 0.1294095234381870 00 |
| 2 |
| RESULTS OF SUBROUTINE GCD |
| ROOT(2) | = 0.2000000000000000000000000000 01 + 0.2000000000000000000000000000 01 |
| 3 |
| RESULTS OF SUBROUTINE GCD |
| ROOT(3) | = -0.9999999999999775 00 + 0.4999999999999998 00 |
| 3 |
| RESULTS OF SUBROUTINE GCD |

Exhibit 6.17. Roots Are: 2+2i (3), 1+2i (2), -1+5i (3)
GREATEST COMMON DIVISOR METHOD USED WITH MÜLLER'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 GCO: 0.10-02
- Test for convergence: 0.10-09
- Test for zero in subroutine quad: 0.10-19
- Test for multiplicities: 0.10-01
- Radius to start search: 0.000 00
- Radius to end search: 0.000 00

The degree of $p(x)$ is 12. The coefficients are:

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{13}$</td>
<td>0.100000000000000000001</td>
</tr>
<tr>
<td>$p_{12}$</td>
<td>-0.12000000000000000000</td>
</tr>
<tr>
<td>$p_{11}$</td>
<td>0.72000000000000000001</td>
</tr>
<tr>
<td>$p_{10}$</td>
<td>-0.28000000000000000000</td>
</tr>
<tr>
<td>$p_{9}$</td>
<td>0.78000000000000000000</td>
</tr>
<tr>
<td>$p_{8}$</td>
<td>-0.16320000000000000000</td>
</tr>
<tr>
<td>$p_{7}$</td>
<td>0.26240000000000000000</td>
</tr>
<tr>
<td>$p_{6}$</td>
<td>-0.32640000000000000000</td>
</tr>
<tr>
<td>$p_{5}$</td>
<td>0.31200000000000000000</td>
</tr>
<tr>
<td>$p_{4}$</td>
<td>-0.22400000000000000000</td>
</tr>
<tr>
<td>$p_{3}$</td>
<td>0.11520000000000000000</td>
</tr>
<tr>
<td>$p_{2}$</td>
<td>-0.38400000000000000000</td>
</tr>
<tr>
<td>$p_{1}$</td>
<td>0.64000000000000000000</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 2. The coefficients are:

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_{3}$</td>
<td>0.100000000000000000001</td>
</tr>
<tr>
<td>$Q_{2}$</td>
<td>-0.20000000000000000000</td>
</tr>
<tr>
<td>$Q_{1}$</td>
<td>0.19999999999998300000</td>
</tr>
</tbody>
</table>

Roots of $p(x)$:

1. $0.100000000000000000001$  
   2. $0.100000000000000000001$  

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 MULTIPLEITIES OF POLYNOMIALS

POLYNOMIAL NUMBER 1

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.100000
TEST FOR ZERO IN SUBROUTINE QADR: 0.100000
RADIUS TO START SEARCH: 0.0000 00
RADIUS TO END SEARCH: 0.0000 00

THE DEGREE OF PI(1) IS 6 & THE COEFFICIENTS ARE

P17(I) = 0.1000000000000000 01 + 0.0000000000000000 00 00
P18(I) = -0.7000000000000000 01 + 0.1000000000000000 02 02
P15(I) = -0.2890000000000000 02 + 0.5800000000000000 03 03
P14(I) = 0.1710000000000000 03 + 0.1500000000000000 04 04
P13(I) = -0.7390000000000000 02 + 0.7910000000000000 03 03
P12(I) = -0.2280000000000000 02 + 0.1040000000000000 03 03
P11(I) = 0.7200000000000000 02 + 0.1040000000000000 03 03

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

GI1(I) = 0.1000000000000000 01 + 0.1000000000000000 02 02
GI1(I) = 0.1000000000000000 01 + 0.1000000000000000 02 02

ROOTS OF PI(1) MULTIPLEITIES INITIAL APPROXIMATION

ROOT(1) = -0.9999999999973500 00 + 0.50000000000175420 00 00 1 NO INITIAL APPROXIMATIONS

THE FOLLOWING POLYNOMIAL, GI(1), CONTAINS ALL THE ROOTS OF PI(1) WHICH HAVE MULTIPLEITY 2

GI2(I) = 0.1000000000000000 01 + 0.1000000000000000 02 02
GI2(I) = -0.1999999999917590 00 + 0.1000000000000000 01 01

Exhibit 6.19.
ROOTS OF P(x)

MULTIPlicITIES

INITIAL APPROXIMATION

ROOT: 31 = 0.199999999796760 01 + 0.200000000015190 01 1 3  NO INITIAL APPROXIMATIONS

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

2 1 = 0.10000000000000 01 + 0.00000000000000 01 1
1 1 = -0.199999999999970 01 + -0.2000000000015190 01 1

ROOTS OF P(x)

MULTIPlicITIES

INITIAL APPROXIMATION

ROOT: 13 = 0.99999999991790 00 + 0.199999999958450 01 1 2  NO INITIAL APPROXIMATIONS

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 multiplicities 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:** 0.100-03
**Radius to Start Search:** 0.000 00
**Radius to End Search:** 0.000 00

**The Degree of \( p(x) \) is 15**
The coefficients are:

\[
P(15) = 0.2557120000000000 \times 10^2 + 0.0000000000000000 \times 10^0
\]
\[
P(14) = 0.7353568000000000 \times 10^2 - 0.2189696000000000 \times 10^4
\]
\[
P(13) = 0.4967982700000000 \times 10^3 - 0.1765875644000000 \times 10^5
\]
\[
P(12) = 0.1642712200000000 \times 10^5 + 0.4373662300000000 \times 10^6
\]
\[
P(11) = -0.2036625888420000 \times 10^6 + 0.1093899227670000 \times 10^6
\]
\[
P(10) = -0.4967982700000000 \times 10^5 - 0.1765875644000000 \times 10^5
\]
\[
P(9) = 0.1329434434800000 \times 10^5 + 0.1038130226550000 \times 10^6
\]
\[
P(8) = 0.2755620000000000 \times 10^0 + 0.2755620000000000 \times 10^0
\]
\[
P(7) = 0.3053900774700000 \times 10^0 + 0.2998989141300000 \times 10^0
\]
\[
P(6) = -0.1274997298590000 \times 10^6 + 0.2171341227420000 \times 10^6
\]
\[
P(5) = -0.2634692716800000 \times 10^6 + 0.1926497278600000 \times 10^6
\]
\[
P(4) = -0.1835899020000000 \times 10^3 + 0.1827632160000000 \times 10^3
\]
\[
P(3) = -0.1590769644000000 \times 10^0 + 0.1998991400000000 \times 10^0
\]
\[
P(2) = -0.1022394522130000 \times 10^0 + 0.6030444423000000 \times 10^0
\]
\[
P(1) = -0.4800000000000000 \times 10^2 + 0.0000000000000000 \times 10^0
\]

The following polynomial, \( g(x) \), contains all the roots of \( p(x) \) which have multiplicity 1.

\[
g(2) = 0.4800000000000000 \times 10^2 + 0.0000000000000000 \times 10^0
\]
\[
g(1) = 0.1119999992896031 \times 10^3 + 0.0000000000000000 \times 10^0
\]

**Roots of \( p(x) \)**

<table>
<thead>
<tr>
<th>Root</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>-0.233333333318333370</td>
</tr>
</tbody>
</table>

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

\[ G(x) = 0.100800000000000000000 01 + 0.00000000000000000000 00 \]
\[ G(x) = -0.2999970655634860 02 + 0.999999063627250 00 \]
\[ G(x) = 0.2129999931798450 01 + 0.3000000894178610 02 \]
\[ G(x) = -0.6750109992456270 02 - 0.2250001290676500 01 \]
\[ G(x) = 0.6834963563268300 00 + 0.674693921701720 02 \]

ROOTS OF G(x):

ROOT (1) = \(-0.2999983209321470 02 - 0.2056982846484699 00 \)
ROOT (2) = \(0.4765981655685800 01 + 0.1800000659501840 01 \)
ROOT (3) = \(-0.2331716169396970 07 - 0.1499999972431725 01 \)
ROOT (4) = \(-0.3526235710479 06 - 0.149999987916660 01 \)

INITIAL APPROXIMATION

\[ 0.4629629115656270 00 + 0.1294099284361870 00 \]
\[ 0.7071067393443460 00 + 0.7010640768654590 00 \]

RESULTS OF SUBROUTINE QUAAD

MULTIPLECTIES

2

RESULTS OF SUBROUTINE QUAAD

2

NO INITIAL APPROXIMATIONS

2

NO INITIAL APPROXIMATIONS

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

\[ G(x) = 0.100000000000000000000 01 + 0.00000000000000000000 00 \]
\[ G(x) = 2.9999998835795410 00 - 0.20000000971656320 01 \]
\[ G(x) = 0.30000000025339350 01 - 0.30000000017915140 00 \]

ROOTS OF P(x):

ROOT (1) = \(-0.1462936452349070 07 + 0.30000000171625330 01 \)
ROOT (2) = \(-0.9999999999999999 00 - 0.1000000000000000 02 \)

MULTIPLECTIES

3

INITIAL APPROXIMATIONS

Exhibit 6.20: Roots Are: -2.33, 0.003 (2), i (2), 1.51 (2), -1.51 (2), 3i (3), -1-i (3)
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS

POLYNOMIAL NUMBER 3

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

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

P0 = 0.1000000000000000D 01 + 0.0000000000000000D 00 01
P1 = -0.5175000000000000D 01 + -0.1150000000000000D 02 02
P2 = 0.3075000000000000D 03 + -0.3475000000000000D 03 03
P3 = -0.4952500000000000D 03 + -0.4948750000000000D 03 03
P4 = 0.1810000000000000D 03 + 0.4420000000000000D 03 03
P5 = 0.1580000000000000D 03 + 0.4420000000000000D 03 03

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

G0 = 0.10000000000000000D 01 + 0.0000000000000000D 00 01
G1 = -0.9999999999996498D 00 + -0.1999999999996498D 01 01

ROOTS OF P(X)  MULTIPlicITIES  INITIAL APPROXIMATION
0 = 0.9999999999996498D 00 + 0.1999999999996498D 01 2 NO INITIAL APPROXIMATIONS

Exhibit 6.21.
The following polynomial, \( G(x) \), contains all the roots of \( P(x) \) which have multiplicity 3.

\[
\begin{align*}
G(3) &= 0.1000000000000000D01 + 0.0000000000000000D00 \\
G(2) &= -0.1000000000002539D01 - 0.2500000000001629D01 \\
G(1) &= -0.3000000000002960D01 + 0.1000000000004210D01 \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>ROOTS OF ( P(x) )</th>
<th>MULTIPLEITIES</th>
<th>INITIAL APPROXIMATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(1) = 0.2000000000002332D01 + 0.700000000001653D01</td>
<td>3</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
<tr>
<td>ROOT(2) = -0.9999999999999750D00 + 0.4999999999999750D00</td>
<td>3</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
</tbody>
</table>

Exhibit 6.21. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)
NUMBER OF INITIAL APPROXIMATIONS GIVEN.  0
MAXIMUM NUMBER OF ITERATIONS.  200
TEST FOR ZER0 IN SUBROUTING GCD.  0.100-02
TEST FOR CONVERGENCE.  0.120-09
TEST FOR ZER0 IN SUBROUTING QUAD.  0.100-19
RADIUS TO START SEARCH.  0.000 00
RADIUS TO END SEARCH.  0.000 03

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

P(13) = 0.1000000000000000 01 + 0.0000000000000000 00 1
P(12) = 0.1200000000000000 02 + 0.0000000000000000 00 1
P(11) = 0.3200000000000000 02 + 0.0000000000000000 00 1
P(10) = -0.2800000000000000 03 + -0.0000000000000000 00 1
P(9) = 0.7800000000000000 03 + 0.0000000000000000 00 1
P(8) = -0.6400000000000000 04 + -0.0000000000000000 00 1
P(7) = 0.7620000000000000 04 + 0.0000000000000000 00 1
P(6) = -0.3260000000000000 06 + -0.0000000000000000 00 1
P(5) = 0.3120000000000000 06 + 0.0000000000000000 00 1
P(4) = -0.3260000000000000 06 + -0.0000000000000000 00 1
P(3) = 0.1150000000000000 06 + 0.0000000000000000 00 1
P(2) = -0.3840000000000000 06 + -0.0000000000000000 00 1
P(1) = 0.8400000000000000 10 02 + 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.

\[
G(x) = 0.1000000000000000 \ 01 + 0.0000000000000000 \ 00 \ ( \\
G(x) = -0.1000000000000000 \ 01 - 0.0000000000000000 \ 00 \ ( \\
G(x) = 0.0000000000000000 \ 01 + 0.0000000000000000 \ 00 \ )
\]

ROOTS OF PI(x)          MULTIPLICITIES          INITIAL APPROXIMATIONS

<table>
<thead>
<tr>
<th>ROOT</th>
<th>0.1000000000000000 01</th>
<th>0.9999999999999999 00</th>
<th>0</th>
<th>NO INITIAL APPROXIMATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT 2</td>
<td>0.1000000000000000 01</td>
<td>-0.9999999999999999 00</td>
<td>0</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
</tbody>
</table>

Exhibit 6.22. Roots Are: 1+i (6), 1-i (6)
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NULLS Method TO EXTRACT ROOTS AND MULTICLICITIES OF POLYNOMIALS
POLYNOMIAL NUMBER 1

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.100-09
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE DEGREE OF PIX1 IS 6 THE COEFFICIENTS ARE

P1 [1] = 0.1000000000000000 01 + 0.0000000000000000 02 00
P1 [2] = -0.7000000000000000 01 + 0.1000000000000000 02 02
P1 [3] = -0.2800000000000000 02 + 0.3000000000000000 02 02
P1 [4] = 0.1700000000000000 01 + 0.1600000000000000 02 01
P1 [5] = -0.2100000000000000 02 + -0.2100000000000000 02 01
P1 [6] = -0.2200000000000000 03 + 0.1000000000000000 03 03
P1 [7] = 0.1200000000000000 03 + 0.1000000000000000 03 03

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

THE FOLLOWING POLYNOMIAL, GIX1, CONTAINS ALL THE ROOTS OF PIX1 WHICH HAVE MULTICLICITY 1

G 01 = 0.1000000000000000 01 + 0.0000000000000000 02 00
G 02 = 0.9999999999997850 00 + -0.5000000000017540 00 01

ROOTS OF GIX1

ROOT 1) = -0.9999999999997850 00 + 0.5000000000017540 00 01

ROOT 2) = -0.9999999999997850 00 + 0.5000000000017540 00 01

MULTICLICITIES

0.4829629115656279D 00 + 0.1294095284438187D 00 1

INITIAL APPROXIMATION

0.4829629115656279D 00 + 0.1294095284438187D 00 1

INITIAL APPROXIMATION

0.4829629115656279D 00 + 0.1294095284438187D 00 1

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

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

\[
G(2^i) = 0.1000000000000000 D 01 + 0.0000000000000000 D 00 I \\
G(1^j) = -0.9999999999999999 D 01 + 0.1999999999999998 D 01 I
\]

ROOTS OF \( G(x) \)

ROOT(1) = 0.9999999999999999 D 00 + 0.1999999999999998 D 01 I

ROOTS OF \( P(x) \) MULTIPLEITIES

ROOT(1) = 0.9999999999999999 D 00 + 0.1999999999999998 D 01 I 2

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

\[
G(2^i) = 0.1000000000000000 D 01 + 0.0000000000000000 D 00 I \\
G(1^j) = -0.9999999999999999 D 01 + 0.2000000000000000 D 01 I
\]

ROOTS OF \( P(x) \) MULTIPLEITIES

ROOT(3) = 0.1999999999999999 D 01 + 0.2000000000000000 D 01 I 3

 Exhibit 6.23. Roots Are: 2+2i (3), 1+2i (2), -1+5i
NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GED. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE QUNAD. 0.100-19
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE DEGREE OF P[I] IS 15 THE COEFFICIENTS ARE

P[0] = 0.4800000000000000 02 = 0.0000000000000000 00 1
P[1] = 0.2457123000000000 03 = 0.1366010000000000 03 1
P[2] = 0.1756556800000000 02 = 0.7185460000000000 04 1
P[3] = 0.1322584668000000 05 = 0.1240269970000000 05 1
P[4] = 0.0867902700000010 05 = 0.1765867460000000 09 1
P[5] = 0.1022394223160000 06 = 0.3053662390000000 04 1
P[6] = 0.2936624884260000 06 = 0.1089942270000000 06 1
P[7] = 0.1871239572000000 06 = 0.1926439433000000 06 1
P[8] = 0.1274093208500000 06 = 0.2174334274000000 06 1
P[9] = 0.2844293271600000 05 = 0.1924492276000000 06 1
P[10] = 0.1325465343960000 05 = 0.1038150265000000 06 1
P[11] = 0.3059403737400000 05 = 0.2958591411000000 05 1
P[12] = 0.1875892805000000 03 = 0.1827982160000000 03 1
P[13] = 0.2759200000000000 00 = 0.2755620000000000 00 1

THE FOLLOWING POLYNOMIAL, G[I], CONTAINS ALL THE ROOTS OF P[I] WHICH HAVE MULTIPLICITY 1:

G[0] = 0.4800000000000000 02 = 0.0000000000000000 00 1
G[1] = 0.1119999999296030 03 = 0.8274908658740760-06 1

ROOTS OF G[I] INITIAL APPROXIMATION

ROOT (1) = -0.2333333319333970 01 = 0.1723959570799939-07 1

ROOTS OF G[I] MULTIPlicITIES INITIAL APPROXIMATION

Exhibit 6.24.
The following polynomial, \( g(x) \), contains all the roots of \( p(x) \) which have multiplicity 2.

<table>
<thead>
<tr>
<th>Roots of ( g(x) )</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ROOT(1)} = 0.2309629115662790 )</td>
<td>2</td>
<td>0.4829629115662790</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Roots of ( g(x) )</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ROOT(1)} = 0.329409528431870 )</td>
<td>3</td>
</tr>
</tbody>
</table>

Exhibit 6.24.
Exhibit 6.24. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2), i (2), 1.5i (2), -1 -i (3), 3i (3), -1-i (3)
REPEATED USE OF THE GREATEST COMMON DIVISOR AND MULLER'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: 0.100-02
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 PI(x) IS 6. THE COEFFICIENTS ARE

\[
P(i) = \begin{array}{c}
0.10000000000000000000000000000000 01 + 0.00000000000000000000000000000000 00 1 \\
-0.50000000000000000000000000000000 01 + -0.11580000000000000000000000000000 02 1 \\
-0.17290000000000000000000000000000 01 + -0.93000000000000000000000000000000 02 1 \\
0.15790000000000000000000000000000 03 + 0.14487900000000000000000000000000 03 1 \\
0.30760000000000000000000000000000 03 + -0.31470000000000000000000000000000 03 1 \\
-0.49220000000000000000000000000000 03 + -0.49487500000000000000000000000000 03 1 \\
-0.58570000000000000000000000000000 03 + -0.42745000000000000000000000000000 03 1 \\
0.18180000000000000000000000000000 03 + 0.44210000000000000000000000000000 03 1 \\
0.15800000000000000000000000000000 03 + 0.60000000000000000000000000000000 03 1
\end{array}
\]

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

NO ROOTS OF MULTIPlicity 1

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

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

GI(x) = 0.10000000000000000000000000000000 01 + 0.00000000000000000000000000000000 00 1
GI(x) = -0.999999999999946980 00 + -0.199999999999946980 01 1

ROOTS OF GI(x)

\[
\begin{array}{c}
0.999999999999946980 00 + 0.199999999999946980 01 1 \\
0.4829620156562790 00 + 0.12940952844381870 00 1
\end{array}
\]

Exhibit 6.25.
ROOTS OF $P(x)$  \hspace{1cm} MULTIPlicITIES  \hspace{1cm} INITIAL APPROXIMATION

**ROOT(1)** = 0.99999999999946980 00 + 0.19999999999964680 01
**ROOT(2)** = 0.48296291156542790 00 + 0.48296291156542790 00

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

**ROOT(1)** = 0.10000000000000000 01 + 0.00000000000000000 00
**ROOT(2)** = -0.10000000000000000 01 - 0.25000000000000000 01
**ROOT(3) = -0.10000000000000000 01 + 0.10000000000000000 01

The roots of $G(x)$ are:

**ROOT(1)** = -0.99999999999999280 00 + 0.49999999999997590 00
**ROOT(2)** = 0.20000000000025320 01 + 0.20000000000016530 01

Roots of $G(x)$, solved by direct method:

**ROOT(1)** = 0.20000000000025320 01 + 0.20000000000016530 01

Not all roots of the above polynomial, $G$, were found.

Exhibit 6.25. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)
Number of initial approximations given: 0
Maximum number of iterations: 200
Test for zero in subroutine GCD: 0.100-07
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 12, the coefficients are:

\[
P(0) = 0.10000000000000000
P(1) = -0.12000000000000000
P(2) = 0.72000000000000000
P(3) = -0.10000000000000000
P(4) = -0.11520000000000000
P(5) = 0.31200000000000000
P(6) = -0.32400000000000000
P(7) = 0.31240000000000000
P(8) = -0.38400000000000000
P(9) = 0.64000000000001000
P(10) = 0.38400000000000000
P(11) = -0.16320000000000000
P(12) = 0.26240000000000000
P(13) = -0.10000000000000000
P(14) = 0.00000000000000000
P(15) = 0.00000000000000000
P(16) = 0.00000000000000000
P(17) = 0.00000000000000000
P(18) = 0.00000000000000000
\]

No roots of multiplicity 1.

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

\[
G(X) = 0.1000000000000000D01 + 0.0000000000000000D01 + 0.9999999999999707D00 \text{ I} + 0.4829629115656279D00 + 0.1294095284438187D00 \text{ I} + 0.1294095284438187D00
\]

ROOTS OF \( G(X) \)

\[ \text{ROOT}(1) = 0.1000000000000033D01 + 0.9999999999999707D00 \text{ I} + 0.4829629115656279D00 + 0.1294095284438187D00 \text{ I} \]

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:

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

where

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

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.

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.
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

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

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

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

3. Searching the Complex Plane

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

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

By generating initial approximations internally, the program can search an annulus centered at the origin of inner radius \(X_{\text{START}}\) and outer radius \(X_{\text{END}}\). Values for \(X_{\text{START}}\) and \(X_{\text{END}}\) 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 \( X\text{END} \) but not greater than \( X\text{END} + .5N \).

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 Nth 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>1 2 4 5 7 8</th>
<th>6 4 7 7 7 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>N P O L Y</td>
<td>XSTART</td>
</tr>
<tr>
<td>N I A P</td>
<td>XEND</td>
</tr>
<tr>
<td>1 7 3</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 $7^{\text{th}}$ degree polynomial. After these approximations are used the circle of radius 15 will be searched for the remaining roots.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>8</th>
<th>6</th>
<th>7</th>
<th>7</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>O</td>
<td>P</td>
<td>L</td>
<td>Y</td>
<td>N</td>
<td>I</td>
<td>A</td>
<td>P</td>
<td>XSTART</td>
<td>XEND</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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.2. Altering Approximations
Figure A.3. Distribution of Approximations
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 2 OF DEGREE 3

THE COEFFICIENTS OF FIRST ARE

\[ \begin{align*}
P(1) &= 0.10000000000000000 \text{ D1} + 0.00000000000000000 \text{ D0} \\
P(2) &= 0.20000000000000000 \text{ D1} + 0.00000000000000000 \text{ D0} \\
P(3) &= -0.10000000000000000 \text{ D1} - 0.00000000000000000 \text{ D0} \\
P(4) &= -0.20000000000000000 \text{ D1} - 0.00000000000000000 \text{ D0}
\end{align*} \]

NUMBERS OF INITIAL APPROXIMATIONS GIVEN: 0

MAXIMUM NUMBER OF ITERATIONS: 3

TEST FOR CONVERGENCE: 0.10D-D1

TEST FOR MULTIPLICITIES: 0.10D-D1

RADIUS TO START SEARCH: 0.00D 00

RADIUS TO END SEARCH: 0.00D 00

NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER 3 ITERATIONS.

\[ \begin{align*}
&\text{INITIAL APPROXIMATION} \\
&\text{ALtered APPROXIMATION}
\end{align*} \]

COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

\[ \begin{align*}
P(1) &= 0.10000000000000000 \text{ D1} + 0.00000000000000000 \text{ D0} \\
P(2) &= 0.20000000000000000 \text{ D1} + 0.00000000000000000 \text{ D0} \\
P(3) &= -0.10000000000000000 \text{ D1} - 0.00000000000000000 \text{ D0} \\
P(4) &= -0.20000000000000000 \text{ D1} - 0.00000000000000000 \text{ D0}
\end{align*} \]

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

THE COEFFICIENTS OF P1(x) ARE

\[ P_1(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \]

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
<th>Coefficient</th>
<th>Value</th>
<th>Coefficient</th>
<th>Value</th>
<th>Coefficient</th>
<th>Value</th>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_7</td>
<td>0.10000000000000000</td>
<td>a_6</td>
<td>0.00000000000000000</td>
<td>a_5</td>
<td>0.10000000000000000</td>
<td>a_4</td>
<td>0.00000000000000000</td>
<td>a_3</td>
<td>0.00000000000000000</td>
</tr>
<tr>
<td>a_2</td>
<td>0.50000000000000000</td>
<td>a_1</td>
<td>0.10000000000000000</td>
<td>a_0</td>
<td>0.00000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

BEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P1(x) ARE

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(x) )</td>
<td>D1</td>
<td>(-0.35000000000000000 )</td>
</tr>
<tr>
<td>( P(x) )</td>
<td>D2</td>
<td>(-0.35000000000000000 )</td>
</tr>
<tr>
<td>( P(x) )</td>
<td>D3</td>
<td>(-0.35000000000000000 )</td>
</tr>
</tbody>
</table>

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P1(x) ARE

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(x) )</td>
<td>D1</td>
<td>(-0.35000000000000000 )</td>
</tr>
<tr>
<td>( P(x) )</td>
<td>D2</td>
<td>(-0.35000000000000000 )</td>
</tr>
<tr>
<td>( P(x) )</td>
<td>D3</td>
<td>(-0.35000000000000000 )</td>
</tr>
</tbody>
</table>

COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
<th>Coefficient</th>
<th>Value</th>
<th>Coefficient</th>
<th>Value</th>
<th>Coefficient</th>
<th>Value</th>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>0.10000000000000000</td>
<td>D2</td>
<td>0.00000000000000000</td>
<td>D3</td>
<td>0.00000000000000000</td>
<td>D4</td>
<td>0.00000000000000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>0.10000000000000000</td>
<td>D6</td>
<td>0.00000000000000000</td>
<td>D7</td>
<td>0.00000000000000000</td>
<td>D8</td>
<td>0.00000000000000000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Exhibit A.2. Roots Are: \(-1 - 4i, -2 - 3i, -3 - 3i, -1 - 1, 2 + 2i, 4 - 1, 2 - 1\).
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 1 OF DEGREE 7

THE COEFFICIENTS OF P(x) ARE

P: ... = -0.1000000000000000D 01
P: ... = -0.1000000000000000D 01
P: ... = -0.5900000000000000D 02
P: ... = 0.1400000000000000D 03
P: ... = 0.7000000000000000D 02
P: ... = -0.1000000000000000D 01
P: ... = 0.1420000000000000D 04
P: ... = -0.1820000000000000D 04
P: ... = 0.1594000000000000D 04

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 2
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.10D 09
TEST FOR MULTIPlicITIES: 0.10D 01
RADIUS TO START SEARCH: 0.700 01
RADIUS TO END SEARCH: 0.15D 02

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

ROOTS OF P(x)
MULTIPLICITIES
INITIAL APPROXIMATION
ROOT(1) = -0.2999999999999997D 01 + 0.3500000000000000D 01
ROOT(2) = 0.2000000000000000D 01 + 0.2860000000000000D 01
ROOT(3) = 0.3000000000000000D 01 + 0.1830000000000000D 01
ROOT(4) = 0.1999999999999997D 01 + 0.5303301053034420D 01
ROOT(5) = 0.4000000000000000D 01 + 0.2070551889415497D 01
ROOT(6) = 0.1999999999999997D 01 + 0.7727406736625032D 01
ROOT(7) = 0.2000000000000000D 01 + 0.1594000000000000D 04

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

ROOTS OF P(x)
MULTIPLICITIES
INITIAL APPROXIMATION
ROOT(1) = -0.2999999999999997D 01 + 0.3500000000000000D 01
ROOT(2) = 0.2000000000000000D 01 + 0.2860000000000000D 01
ROOT(3) = 0.3000000000000000D 01 + 0.1830000000000000D 01
ROOT(4) = 0.1999999999999997D 01 + 0.5303301053034420D 01
ROOT(5) = 0.4000000000000000D 01 + 0.2070551889415497D 01
ROOT(6) = 0.1999999999999997D 01 + 0.7727406736625032D 01
ROOT(7) = 0.2000000000000000D 01 + 0.1594000000000000D 04

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

NEWTON'S METHOD

1. Use of the Program

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

The program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree $N$, where $N > 25$, certain array dimensions must be changed. These are listed in Table B.I for the main program and subprograms in double precision.
TABLE B.1

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)
RXINT(N), VXINT(N)

Subroutine HORNER

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

Subroutine BETTER

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

Subroutine GENAPP

APPR(N), APPI(N)

Subroutine QUAD

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

Table B.11 lists the system functions used in the program of Newton's method. In the table "d" denotes a double precision variable name.
TABLE B.II

SYSTEM FUNCTIONS USED IN NEWTON'S METHOD

**Double Precision**

- **DABS(d)** - obtain absolute value
- **DCOS(d)** - obtain cosine of angle
- **DSIN(d)** - obtain sine of angle
- **DATAN2(d_1, d_2)** - arctangent of \( \frac{d_1}{d_2} \)
- **DSQRT(d)** - square root

2. **Input Data for Newton's Method**

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

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

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

Control Information

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

TABLE B.III
CONTROL DATA FOR NEWTON'S METHOD

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

**Initial Approximations**

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

**INITIAL APPROXIMATION DATA FOR NEWTON'S METHOD**

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

**End Card**

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

### TABLE B.VI

**DATA TO END EXECUTION OF NEWTON'S METHOD**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCHECK</td>
<td>c.c. 80</td>
<td>Must contain the number 1. Integer.</td>
</tr>
</tbody>
</table>
3. Variables Used in Newton's Method

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

\[ \begin{align*}
    R & \quad \text{real variable} \\
    I & \quad \text{integer variable} \\
    C & \quad \text{complex variable} \\
    D & \quad \text{double precision} \\
    L & \quad \text{logical variable} \\
    A & \quad \text{alphanumeric variable}
\end{align*} \]

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:

\[ \begin{align*}
    E & \quad \text{entered} \\
    R & \quad \text{returned} \\
    ECR & \quad \text{entered, changed, and returned} \\
    C & \quad \text{variable in common}
\end{align*} \]

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 "MULTIPLCITIES" (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
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
0000000001111111122222222333333333344444444455555555566666666666777777778
1234567890123456789012345678901234567890123456789012345678901234567890
<table>
<thead>
<tr>
<th>N</th>
<th>N</th>
<th>I</th>
<th>A</th>
<th>P</th>
<th>MAX</th>
<th>EPSCNV</th>
<th>ESPQ</th>
<th>EPSMUL</th>
<th>XSTART</th>
<th>XEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>7</td>
<td>200</td>
<td>1.D-10</td>
<td>1.D-20</td>
<td>1.D-02</td>
<td></td>
<td>1.OD+01</td>
<td>5.OD+02</td>
<td></td>
</tr>
</tbody>
</table>
```

**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>0.621735D+01</td>
<td>-0.132714D-02</td>
</tr>
</tbody>
</table>

**Figure B.3.** Coefficient Card for Newton's Method
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>Single Precision Type</th>
<th>Double Precision Variable</th>
<th>Double Precision Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>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>EPSCV</td>
<td>R</td>
<td>EPSCV</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>END</td>
<td>R</td>
<td>END</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>RPX, VPX</td>
<td>D</td>
<td></td>
<td>Value of (P(X)) at some point (X)</td>
</tr>
<tr>
<td>XZERO</td>
<td>C</td>
<td>RXZERO, VXZERO</td>
<td>D</td>
<td></td>
<td>Array containing the initial approximations</td>
</tr>
<tr>
<td>XNEW</td>
<td>C</td>
<td>RXNEW, VXNEW</td>
<td>D</td>
<td></td>
<td>New approximation (X_{n+1}) obtained from old approximation (X_n) by Newton's Algorithm</td>
</tr>
<tr>
<td>KANS</td>
<td>I</td>
<td>KANS</td>
<td>I</td>
<td></td>
<td>(KANS = 1) implies convergence, (KANS = 0) implies no convergence</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td></td>
<td>Array containing the number of multiplicities of each root</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>RX, VX</td>
<td>D</td>
<td></td>
<td>Array containing the zeros of (P(X))</td>
</tr>
<tr>
<td>XINIT</td>
<td>C</td>
<td>RXINIT, VXINIT</td>
<td>D</td>
<td></td>
<td>Array containing the initial or altered approximations which produced convergence to each root</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
<td>I</td>
<td></td>
<td>Number of coefficients of current deflated polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>R B, 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>
</tbody>
</table>
TABLE B. VII  (Continued)

<table>
<thead>
<tr>
<th>Single Precision</th>
<th>Type</th>
<th>Variable</th>
<th>Type</th>
<th>Variable</th>
<th>Type</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>A</strong></td>
<td><strong>C</strong></td>
<td><strong>RA,VA</strong></td>
<td><strong>D</strong></td>
<td><strong>E</strong></td>
<td>Array of coefficients of polynomial</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>B</strong></td>
<td><strong>C</strong></td>
<td><strong>RB,VB</strong></td>
<td><strong>D</strong></td>
<td><strong>R</strong></td>
<td>Array of coefficients of deflated polynomial</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NDEF</strong></td>
<td><strong>I</strong></td>
<td><strong>NDEF</strong></td>
<td><strong>I</strong></td>
<td><strong>E</strong></td>
<td>Degree of polynomial</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NUM</strong></td>
<td><strong>I</strong></td>
<td><strong>NUM</strong></td>
<td><strong>I</strong></td>
<td></td>
<td>Number of coefficients of polynomial</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>X0</strong></td>
<td><strong>C</strong></td>
<td><strong>RXO, VXO</strong></td>
<td><strong>D</strong></td>
<td><strong>E</strong></td>
<td>Point ($X_n$) at which to evaluate the polynomial and its derivative. Also current approximation ($X_{n+1}$) used to deflate the polynomial</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PX</strong></td>
<td><strong>C</strong></td>
<td><strong>RPX, VPX</strong></td>
<td><strong>D</strong></td>
<td><strong>R</strong></td>
<td>Value of polynomial at $X_n$</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>DPX</strong></td>
<td><strong>C</strong></td>
<td><strong>RDPX, VDPX</strong></td>
<td><strong>D</strong></td>
<td><strong>R</strong></td>
<td>Value of the derivative of polynomial at $X_n$</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>C</strong></td>
<td><strong>C</strong></td>
<td><strong>RC, VC</strong></td>
<td><strong>D</strong></td>
<td></td>
<td>Array of containing sequence of values leading to the derivative</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PX</strong></td>
<td><strong>C</strong></td>
<td><strong>RPX, VPX</strong></td>
<td><strong>D</strong></td>
<td><strong>E</strong></td>
<td>Value of polynomial at $X_n$</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>DPX</strong></td>
<td><strong>C</strong></td>
<td><strong>RDPX, VDPX</strong></td>
<td><strong>D</strong></td>
<td><strong>E</strong></td>
<td>Derivative of polynomial at $X_n$</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>X0</strong></td>
<td><strong>C</strong></td>
<td><strong>RXO, VXO</strong></td>
<td><strong>D</strong></td>
<td><strong>E</strong></td>
<td>Current approximation ($X_n$) to root</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>XNEW</strong></td>
<td><strong>C</strong></td>
<td><strong>RXNEW, VXNEW</strong></td>
<td><strong>D</strong></td>
<td><strong>R</strong></td>
<td>New approximation ($X_{n+1}$) to root</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>EPSLON</strong></td>
<td><strong>R</strong></td>
<td><strong>EPS</strong></td>
<td><strong>D</strong></td>
<td><strong>C</strong></td>
<td>Tolerance for convergence or multiplicity check</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>FX</strong></td>
<td><strong>C</strong></td>
<td><strong>RPX, VPX</strong></td>
<td><strong>D</strong></td>
<td><strong>E</strong></td>
<td>Value of $P(X)$ at $X_n$</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>DPX</strong></td>
<td><strong>C</strong></td>
<td><strong>RDPX, VDPX</strong></td>
<td><strong>D</strong></td>
<td><strong>E</strong></td>
<td>Derivative of $P(X)$ at $X_n$</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>X0</strong></td>
<td><strong>C</strong></td>
<td><strong>RXO, VX0</strong></td>
<td><strong>D</strong></td>
<td><strong>E</strong></td>
<td>Current approximations ($X_{n+1}$) to root</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>IO2</strong></td>
<td><strong>I</strong></td>
<td><strong>IO2</strong></td>
<td><strong>I</strong></td>
<td><strong>C</strong></td>
<td>Unit number of output device</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>KANS</strong></td>
<td><strong>I</strong></td>
<td><strong>KANS</strong></td>
<td><strong>I</strong></td>
<td><strong>R</strong></td>
<td>KANS = 1 implies convergence, KANS = 0 implies no convergence</td>
</tr>
<tr>
<td>Single Precision</td>
<td>Double Precision</td>
<td>Disposition of Argument</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>-------------------------</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Variable</td>
<td>Type</td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>XZERO</td>
<td>C</td>
<td>RXZERO, VXZERO</td>
<td>D</td>
<td>E</td>
<td>Array of approximations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>RX, VX</td>
<td>D</td>
<td>ECR</td>
<td>Array of roots</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>RA, VA</td>
<td>D</td>
<td>E</td>
<td>Coefficients of original (undeflated) polynomial, P(X)</td>
<td></td>
<td></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>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>X0</td>
<td>C</td>
<td>RX0, VX0</td>
<td>D</td>
<td>E</td>
<td>Current approximation (X_n) to root</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX, VDPX</td>
<td>D</td>
<td>E</td>
<td>Derivative of P(X) at X_n</td>
<td></td>
<td></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 X_n</td>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td></td>
<td>Counter for number of iterations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XNEW</td>
<td>C</td>
<td>RXNEW, VXNEW</td>
<td>D</td>
<td></td>
<td>New approximation (X_{n+1}) to root</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>I</td>
<td>NN</td>
<td>I</td>
<td></td>
<td>Degree of polynomial</td>
<td></td>
<td></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>
<td></td>
<td></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>
<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>
<td></td>
<td></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>
<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 permitted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPSCHK</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
<td>C</td>
<td>Tolerance for checking convergence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>APPR, APPI</td>
<td>D</td>
<td>R</td>
<td>Array containing initial approximations</td>
<td></td>
<td></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>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>---------------------------</td>
<td>------</td>
<td>------------------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>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>
<td></td>
<td></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>
<td></td>
<td></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>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subroutine ALTER

<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>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 QUAD

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

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

PROGRAM FOR NEWTON'S METHOD

***************************************************************************
DOUBLE PRECISION RA, VA, RXZERO, VXZERO, RB, VB, RCOEF, VCOEF, RX, VX, RXINI
IT, VXINIT, RC, VC, R0, VR, RPK, VPX, RDPX, VDPX, RANK, VXNEW, RX0, VXG, EPSCHK,  
2EPSCNV, EPSQ, EPSMUL, XSTART, XEND, ABPX, ABDPX
DIMENSION RA(26), VA(26), RB(26), VB(26), RC(26), VC(26), R0(26), VR(26),  
2RCOEF(26), VCOEF(26), RANK(25), RXZERO(25), VXZERO(25), RX(25), VX(25), RX  
2INIT(25), VXINIT(25)
COMMON EPSCHK, MAX, IO2
101=5 112
102=6 116
READ(101, 1000) NOPOLYNNIAPMAX, EPSCNV, EPSMUL, XSTART, XEND, KCHECK
IF(KCHECK.EQ.1) STOP
NA=NA+1
READ(101, 1000) (RA(I), VA(I), I=NA, NA)
WRITE(102, 1033) (RA(I), VA(I), I=1, NA)
WRITE(IO2, 1040)
WRITE(102, 1000)
WRITE(102, 1050)
WRITE(IO2, 1050) XEND
IF(NIAP.NE.0) GO TO 3
NIAP=NA
CALL GENAPP(RXZERO, VXZERO, NIAP, XSTART)
GO TO 4
3 READ(IO1, 1020) RXZERO(I), VXZERO(I), I=1, NIAP
4 NIAP=NIAP-1
5 CALL HOMERK(RCOEF, VCOEF, RX, VX, RXZERO, VXZERO, R0, R, RXNEW, RXG, EPSCHK,  
2EPSCNV, EPSQ, EPSMUL, XSTART, XEND, ABPX, ABDPX)
6 RXO=RXZERO(I)
7 VXO=VXZERO(I)
8 I=0
9 WHILE I.EQ.0
10 I=I+1
11 DO 5 I=1, NA
12 RX0=RXZERO(I)
13 VX0=VXZERO(I)
14 RCOEF(I)=RA(I)
15 VCOEF(I)=VA(I)
16 CALL HOMERK(RCOEF, VCOEF, RX, VX, RXNEW, RXG, EPSCHK, EPSCNV, EPSQ, EPSMUL,  
2XSTART, XEND, ABPX, ABDPX)
17 RXO=RXNEW
18 VXO=VXNEW
19 RXO=RXNEW
20 VXO=VXNEW
21 R0=R0+R0
22 R=R0
23 DO 5 I=1, NA
24 RXO=RXZERO(I)
25 VXO=VXZERO(I)
26 RCOEF(I)=RA(I)
27 VCOEF(I)=VA(I)
28 CALL HOMERK(RCOEF, VCOEF, RX, VX, RXNEW, RXG, EPSCHK, EPSCNV, EPSQ, EPSMUL,  
2XSTART, XEND, ABPX, ABDPX)
29 RXO=RXNEW
30 VXO=VXNEW
31 I=I+1
32 K=K+1
33 RXO=RXZERO(I)
34 VXO=VXZERO(I)
35 DO 5 I=1, NA
36 RCOEF(I)=RA(I)
37 VCOEF(I)=VA(I)
38 CALL HOMERK(RCOEF, VCOEF, RX, VX, RXNEW, RXG, EPSCHK, EPSCNV, EPSQ, EPSMUL,  
2XSTART, XEND, ABPX, ABDPX)
39 RXO=RXNEW
40 VXO=VXNEW
CALL GENAPP(RXZERO, VXZERO, NIAP, XSTART)
GO TO 3
41 DO 5 I=1, NA
5 END
TABLE B. VIII (Continued)

0041 IF(APOX NE.0.0) GO TO 20
0042 IF(APOX 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(RZEROIL),VZEROIL),IALTER,ITIME)
0054 IF(IALTER GT 53) GO TO 110
0055 RXO=RZEROIL)
0056 VXO=VZEROIL)
0057 ITER=0
0058 GO TO 10
0059 60 NDEF=NDEF+1
0060 DO 65 J=1,N
0061 RDLJ=RCOEF(J)
0062 65 VDI(J)=VCOEF(J)
0063 GO TO 140
0064 70 NROOT=NROOT+1
0065 K=K+1
0066 MULT(K)=1
0067 RX(K)=RXO
0068 VX(K)=VXO
0069 RXINIT(K)=RZEROIL)
0070 VXINIT(K)=VZEROIL)
0071 CALL HORNERRCOEF,VCOEF,RXO,VXO,NDEF,R8,V8,RC,VC,RPX,VPX,RDPX,VDPX)
0072 GO IF(NROOT GE N) GO TO 147
0073 NDEF=NDEF-1
0074 NUM=NUM+1
0075 DO 105 I=1,NUM
0076 RCOEF(I) R8(I)
0077 105 VCOEF(I) V8(I)
0078 CALL HORNERRCOEF,VCOEF,RXO,VXO,NDEF,R8,V8,RC,VC,RPX,VPX,RDPX,VDPX)
0079 ABI=DSQRT(RPX*RPX+VPX*VPX)
0080 ABDP=DSQRT(RDPX*RDPX+VDPX*VDPX)
0081 IF(ABI NE 0.0) GO TO 207
0082 IF(APOX EQ.0.0) GO TO 130
0083 GO TO 110
0084 107 CONTINUE
0085 EPSCHV=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 IKOT=1
0090 IROOT=0
0091 CALL QUADR(COEF,VCOEF,NDEF,RX,VX,K,MULT,EPSQ)
0092 GO TO 150
0093 113 IF(ULTH,NIAP) GO TO 115
0094 IF(XEND EQ.0.0) GO TO 60
0095 IF(XSTART EQ.0.0) GO TO 60
0096 NIAP=N
0097 CALL GENAPP(RZERORXO,VZERO,NIAP,XSTART)
TABLE B. VIII (Continued)

0097  
0098  
0099  
0100  
0101  
0102  
0103  
0104  
0105  
0106  
0107  
0108  
0109  
0110  
0111  
0112  
0113  
0114  
0115  
0116  
0117  
0118  
0119  
0120  
0121  
0122  
0123  
0124  
0125  
0126  
0127  
0128  
0129  
0130  
0131  
0132  
0133  
0134  
0135  
0136  
0137  
0138  
0139  
0140  
0141  
0142  
0143  
0144  
0145  

L=0
L=L+1 
RX=RXZERO(L) 
VX=VXZERO(L) 
ITER=0 
IALTER=0 
GO TO 10 
MULT(K)=MULT(K)+1 
NROOT=NROOT+1  
GO TO 80 
140 IF(K.EQ.0) GO TO 170 
147 IROOT=K 
150 WRITE(102,1025) 
151 FORMAT(3(12,IX),9X,13HBEFORE THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P(X) ARE) 
152 FORMAT(///IX,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND/) 
153 FORMAT(3X,2D16.4H) = .D23.16,3H + .D23.16,2H I) 
154 FORMAT(///IX,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN) 
155 FORMAT(3X,2HMAX(NUMBER OF ITERATIONS)+11X,13) 
156 FORMAT(3X,2HTEST FOR CONVERGENCE) 
157 FORMAT(3X,2HTEST FOR MULTIPLICITIES) 
158 FORMAT(///IX,2HNUMBER OF INITIAL APPROXIMATIONS GIVEN) 
159 FORMAT(3X,2HMAX(NUMBER OF ITERATIONS)+11X,13) 
160 FORMAT(3X,2HTEST FOR CONVERGENCE) 
161 FORMAT(3X,2HTEST FOR MULTIPLICITIES)
TABLE B. VIII (Continued)

0001  SUBROUTINE GENAPP(APPRI,APPRI,NAPP,XSTART)

0002  END

0003  COMMON DUMMY,MAX,102

0004  IF (XSTART.EQ.0.5) XSTART=0.9

0005  BETA=0.261794

0006  DO 10 =1,NAPP

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

0008  APPR(I)=XSTART*DSIN(BETA)

0009  10 XSTART=XSTART*0.5

0010  RETURN

0011  END

0012  RETURN

0013  END

0014  RETURN

0015  END

0016  RETURN

0017  RETURN

0018  RETURN

0019  RETURN

0020  RETURN

0021  RETURN

0022  RETURN

0023  RETURN

0024  RETURN

0025  RETURN

0026  RETURN
TABLE B. VIII (Continued)

```fortran
SUBROUTINE QUAD(UA,VA,UROOT,VRoot,NROOT,MULTI,EPST)

DOUBLE PRECISION UA,VA,UROOT,VRoot,BBB,AAA,VAAA,UDISC,VDISC,UDUMMY

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

0001 IF(INA.EQ.2) GO TO 7

0002 IF(NA.EQ.1) GO TO 5

0003 NROOT=NROOT+1

0004 UROOT(NROOT)=UA(1)+VA(1)

0005 VROOT(NROOT)=UA(2)+VA(2)

0006 MULTI(NROOT)=1

0007 NROOT=NROOT+1

0008 GO TO 50

0009 NROOT=NROOT+1

0010 GO TO 50

0011 BBB=UA(1)*UA(1)+VA(1)*VA(1)

0012 UROOT(NROOT)=-(UA(1)-VA(1)+VA(1))/BBB

0013 VROOT(NROOT)=-(UA(1)-VA(1)+VA(1))/BBB

0014 MULTI(NROOT)=1

0015 NROOT=NROOT+1

0016 IF(UDISC.LT.EPST) GO TO 10

0017 CALL COMSQT(UDISC,UDUMMY,VDISC,UDUMMY)

0018 BBB=-UA(1)+UDUMMY

0019 VBBB=-VA(1)+VDUMMY

0020 RDummy=-UA(1)-UDummy

0021 SDummy=-VA(1)-VDummy

0022 BBB=2.0*UA(1)

0023 VAAA=2.0*VA(1)

0024 BBB=UA(1)+VA(1)

0025 VAAA=UA(1)+VA(1)

0026 NROOT=NROOT+1

0027 UROOT(NROOT)=(-UA(1)+VAAA+BBB)/BBB

0028 VROOT(NROOT)=(-VA(1)+VAAA+BBB)/BBB

0029 MULTI(NROOT)=2

0030 NROOT=NROOT+2

0031 GO TO 50

0032 UROOT(NROOT)=UA(1)+VA(1)

0033 VROOT(NROOT)=UA(1)+VA(1)

0034 MULTI(NROOT)=1

0035 NROOT=NROOT+1

0036 GO TO 50

0037 UROOT(NROOT)=UA(1)+VA(1)

0038 VROOT(NROOT)=UA(1)+VA(1)

0039 NROOT=NROOT+1

0040 GO TO 50

0041 UROOT(NROOT)=UA(1)+VA(1)

0042 VROOT(NROOT)=UA(1)+VA(1)

0043 NROOT=NROOT+1

0044 RETURN

END
```
TABLE B. VIII (Continued)

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

0001 SUBROUTINE HORNER(RA, VA, RXO, VXO, NDEF, RB, VB, RC, VC, RPX, VPX, RDPX, VDPX)
C
C * HORNER'S METHOD COMPUTES THE VALUE OF A POLYNOMIAL P(X) AT A POINT D AND
C * ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO DEFLATE THE
C * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).
C *
C*****************************************************************************

0002 DOUBLE PRECISION VDPX, RXO, VXO, RB, VB, RC, VC, RPX, VPX, RDPX, RA, VA
0003 DIMENSION RA(26), VA(26), RB(26), VB(26), RC(26), VC(26)
0004 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)+(VB(I-1)*RXO+RB(I-1)*VXO)
0008 RPX=RB(NUM)
0009 VPX=VB(NUM)
0010 RC(I)=RB(I)+VC(I-1)*RXO
0011 VC(I)=VB(I)+RC(I-1)*RXO
0012 IF(NDEF.LT.2) GO TO 25
0013 DO 20 J=2, NDEF
0014 RC(J)=RB(J)+RC(J-1)*RXO
0015 VC(J)=VB(J)+VC(J-1)*RXO
0016 20 CONTINUE
0017 RPX=RC(NDEF)
0018 VPX=VC(NDEF)
0019 RETURN
0020 END

0021 SUBROUTINE NEWTON(RPX, VPX, RDPX, VDPX, RXO, VXO, 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) - P(X(N))/P'(X(N)).
C *
C*****************************************************************************

0022 DOUBLE PRECISION RPX, VPX, RDPX, VDPX, RXO, VXO, RXNEW, VXNEW, ARG
0023 DOUBLE PRECISION DDD
0024 ARG=RPX*RDPX+VPX*VDPX
0025 DDD=DSQRT(ARG)
0026 IF(DDD.EQ.0.0) RETURN
0027 RXNEW=RXO-(RPX*RDPX+VPX*VDPX)/ARG
0028 VXNEW=VXO-(VPX*RDPX-RPX*VDPX)/ARG
0029 RETURN
0030 END

0031
### TABLE E. VIII (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE CHECK(RPX, VPX, RDPX, VDPX, RXO, VKO, KANS)</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION RPX, VPX, RDPX, VDPX, RXO, VKO, ABSXO, ABSQUO, RDUMMY, YO</td>
</tr>
<tr>
<td>0003</td>
<td>DOUBLE PRECISION ARG</td>
</tr>
<tr>
<td>0004</td>
<td>COMMON EPS, MAX, ID2</td>
</tr>
<tr>
<td>0006</td>
<td>ABSXO=DSQRT(RXO<em>RXO+VKO</em>VKO)</td>
</tr>
<tr>
<td>0007</td>
<td>IF(ABSXO.EQ.0.) GO TO 25</td>
</tr>
<tr>
<td>0008</td>
<td>ARG=RPX<em>RPX+VDPX</em>VDPX</td>
</tr>
<tr>
<td>0009</td>
<td>ODD=DSQRT(ARG)</td>
</tr>
<tr>
<td>0010</td>
<td>IF(ODD.EQ.0.) GO TO 25</td>
</tr>
<tr>
<td>0011</td>
<td>RDUMMY=(RPX<em>RDPX+VPX</em>VDPX)/ARG</td>
</tr>
<tr>
<td>0012</td>
<td>VDUMMY=(VPX<em>RDPX-RPX</em>VDPX)/ARG</td>
</tr>
<tr>
<td>0013</td>
<td>ABSQUO=DSQRT(RDUMMY<em>RDUMMY+VDUMMY</em>VDUMMY)</td>
</tr>
<tr>
<td>0014</td>
<td>IF(ABSQUO/ABSXO.LT.EPS) GO TO 10</td>
</tr>
<tr>
<td>0015</td>
<td>KANS=0</td>
</tr>
<tr>
<td>0017</td>
<td>10 KANS=1</td>
</tr>
<tr>
<td>0018</td>
<td>RETURN</td>
</tr>
<tr>
<td>0019</td>
<td>25 KANS=0</td>
</tr>
<tr>
<td>0020</td>
<td>RETURN</td>
</tr>
<tr>
<td>0021</td>
<td>END</td>
</tr>
</tbody>
</table>

---

**Explanation:**

- **SUBROUTINE CHECK(RPX, VPX, RDPX, VDPX, RXO, VKO, KANS):** This subroutine checks for convergence of the sequence of approximations by testing the expression $\frac{|P(x_{n+1})|}{|P'(x_{n+1})|}$, where $P(x_{n+1})$ is a function and $P'(x_{n+1})$ is its derivative. When it is as small as desired, convergence is obtained.

- **Variables:**
  - **RPX, VPX, RDPX, VDPX, RXO, VKO:** Double precision real variables.
  - **ABSXO, ABSQUO:** Absolute values.
  - **RDUMMY, VDUMMY:** Intermediate variables.
  - **EPS, MAX, ID2:** Common variables.

---

**Notes:**

- Line 0010: The subroutine checks if the argument is zero before proceeding.
- Line 0013: The absolute value of the dummy variables is calculated using the $\sqrt{\text{expression}}$ function.
- Line 0017: If the condition is met, KANS is set to 1, indicating convergence; otherwise, it remains 0.

---

**Purpose:**

The subroutine facilitates the determination of convergence in iterative methods, ensuring the accuracy of the approximation process.
TABLE B. VIII (Continued)

SUBROUTINE BETTER(K, RXZERO, VXZERO, RX, VX, NA, RA, VA, RCOEF, VCOEF, N, RC, 
1VC, RB, VB)

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

DOUBLE PRECISION RXZERO, VXZERO, RX, VX, NA, RA, VA, RCOEF, VCOEF, RC, 
VC, RB, VB

DIMENSION RXZERO(25), VXZERO(25), RX(25), RA(26), VA(26), RCOEF(25), 
VCOEF(26), RC(26), VC(26), RB(26), VB(26)

COMMON EPSMAX, EPS, MAX, IO2

DO 10 I = 1, K

RXZERO(I) = RX(I)

VXZERO(I) = VX(I)

DO 20 I = 1, NA

RCOEFF(I) = RA(I)

DO 50 J = 1, K

RXO = RXZERO(I) + RX(O)

VXO = VXZERO(I) + VX(O)

NN = N

ITER = 0

30 CALL HORNER(RCOEF, VCOEF, RXO, VXO, NN, RB, VB, RC, VC, RPX, VPX, RDPX, VDPX)

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

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

IF(ABPX .NE. 0.0) GO TO 33

IF(ITER .GE. MAXI) GO TO 35

GO TO 30

33 CALL NEWTON(RPX, VPX, RPX, VPX, RXO, VXO, RXNEW, VXNEW)

IF(ITER .GE. MAXI) GO TO 35

GO TO 30

34 WRITE(IO2, 1112) RXO, VXO

35 WRITE(IO2, 1001) J, RXZERO(J), VXZERO(J)

36 WRITE(IO2, 200) MAX

40 RX(J) = RXO

VX(J) = VXO

50 CONTINUE

1112 FORMAT(13H THE VALUE OF THE DERIVATIVE AT X0 = ,D23.16, 3H + O2 
13.16, 3H IS ZERO.)

100 FORMAT(13H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(12.4H1 = ,D23 
1.16, 3H + O2.16, 2H 1.10H DID NOT CONVERGE.)

200 FORMAT(33H THE PRESENT APPROXIMATION AFTER 13.29H ITERATIONS IS P 
PRINTED BELOW.)

END
APPENDIX C

MULLER'S METHOD

2. Use of the Program

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

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

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

Double Precision

Main Program

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

Subroutine BETTER

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

Subroutine GENAPP

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

Subroutine HORNER

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

Subroutine QUAD

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

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

SYSTEM FUNCTIONS USED IN MULLER'S METHOD

Double Precision

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

2. Input Data for Muller's Method

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

3. Variables Used in Muller's Method

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

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

<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>
</tbody>
</table>

| **Coefficient Card** |                   |
| A (RA)              | A (UA)           |
| A (VA)              | A (VA)           |

| **Initial Approximation Card** |                   |
| XZERO (RXZERO)          | APP (UAPP)       |
| XZERO (VXZERO)          | APP (VAAPP)      |

| **End Card** |                   |
| KCHECK       | KCHECK            |

### 4. Description of Program Output

The output from Muller's method is the same as that for Newton's method as described in 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 C. IV

**VARIABLES USED IN MULLER'S METHOD**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Single Precision</th>
<th>Variable</th>
<th>Type</th>
<th>Double Precision</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>I</td>
<td>NP</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td>Degree of polynomial $P(X)$</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>NOMULT</td>
<td>I</td>
<td>NOMULT</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td>Number of roots (counting multiplicities)</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT,VROOT</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Array containing the roots</td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td>Number of initial approximations to be read in</td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>UAPP,VAPP</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Array of initial approximations</td>
</tr>
<tr>
<td>WORK</td>
<td>C</td>
<td>UWORK,VWORK</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Working array containing coefficients of current polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB,VB</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Array containing coefficients of deflated polynomial</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA,VA</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Array containing coefficients of original polynomial, $P(X)$</td>
</tr>
<tr>
<td>RAPP</td>
<td>C</td>
<td>URAPP,VRAPPM</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Array of initial or altered approximations for which convergence was obtained</td>
</tr>
<tr>
<td>X1</td>
<td>C</td>
<td>UX1,VX1</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>UX2,VX2</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3,VX3</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>PX1</td>
<td>C</td>
<td>UPX1,VXP1</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Value of polynomial $P(X)$ at $X_1$</td>
</tr>
<tr>
<td>PX2</td>
<td>C</td>
<td>UPX2,VXP2</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Value of polynomial $P(X)$ at $X_2$</td>
</tr>
<tr>
<td>PX3</td>
<td>C</td>
<td>UPX3,VXP3</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Value of polynomial $P(X)$ at $X_3$</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4,VX4</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Newest approximation $(X_{n+1})$ to root</td>
</tr>
<tr>
<td>PX4</td>
<td>C</td>
<td>UXP4,VXP4</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>Value of polynomial $P(X)$ at $X_4$</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td>Array containing the multiplicities of each root found</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td>Counter for iterations</td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
<td></td>
<td></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></td>
<td></td>
<td>Unit number of output device</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

| A                          | C    | UA,VA                      | D    | E                        | Array of current polynomial coefficients (to be deflated or evaluated) |
| NA                         | I    | NA                         | I    | E                        | Degree of polynomial to be deflated or evaluated |
| X                          | C    | UX,VX                      | D    | E!                       | Approximation at which to evaluate or deflate the polynomial |
### TABLE C. IV (Continued)

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

| X1                          | C    | X1R, X1I                  | D    | ECR                     | One of the three approximations to be altered |
| X2                          | C    | X2R, X2I                  | D    | ECR                     | One of the three approximations to be altered |
| X3                          | C    | X3R, X3I                  | D    | ECR                     | One of the three approximations to be altered |
| X2R                         | R    | X2R                       | D    |                         | Imaginary part of complex approximation |
| X2I                         | R    | X2I                       | D    |                         | Imaginary part of complex approximation |

**Subroutine QUAD**

| EPST                        | R    | E                         | D    |                         | Tolerance check for zero (0) |

**Subroutine CALC**

These variables are dummy variables used for temporary storage and thus, are not defined.
Figure C.1. Flow Charts for Muller's Method
Figure C.1. (Continued)
Figure C.17 (Continued)
Figure C.1. (Continued)
Figure C.1. (Continued)
Figure C.1: (Continued)
### TABLE C.V.

**PROGRAM FOR MULLER'S METHOD**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td><code>DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,URDO,T,VRDO,T,UX1,UX1,UAPP,YAPP</code></td>
</tr>
<tr>
<td>0002</td>
<td><code>DIMENSION URODT(251),VRDO,T(251),MULT(251),UAPP(25,3),VAPP(25,3),UWlk</code></td>
</tr>
<tr>
<td>0003</td>
<td><code>I(251),VUWlk(256),U2(256),U2(256),UA(261),VA(261),UAPP25,3),VAPP25,3)</code></td>
</tr>
<tr>
<td>0006</td>
<td><code>eps=0.999</code></td>
</tr>
<tr>
<td>0009</td>
<td><code>dim=0</code></td>
</tr>
<tr>
<td>0010</td>
<td><code>VRODT=0</code></td>
</tr>
<tr>
<td>0011</td>
<td><code>I PATH=1</code></td>
</tr>
<tr>
<td>0012</td>
<td><code>NOMULT=0</code></td>
</tr>
<tr>
<td>0013</td>
<td><code>NALT=0</code></td>
</tr>
<tr>
<td>0014</td>
<td><code>ITIME=0</code></td>
</tr>
<tr>
<td>0015</td>
<td><code>IAPP=1</code></td>
</tr>
<tr>
<td>0016</td>
<td><code>ITER=1</code></td>
</tr>
<tr>
<td>0017</td>
<td><code>READ(1000) NODL,NP,NAPP,MAX,EPS,EPSM,XSTART,XEND,KCHECK</code></td>
</tr>
<tr>
<td>0018</td>
<td><code>IF(KCHECK.EQ.11) STOP</code></td>
</tr>
<tr>
<td>0019</td>
<td><code>KK=NP+1</code></td>
</tr>
<tr>
<td>0020</td>
<td><code>READ(1010) (UA(I),VA(I),I=1,KK)</code></td>
</tr>
<tr>
<td>0021</td>
<td><code>WRITE(102,1020) NODL,NP</code></td>
</tr>
<tr>
<td>0022</td>
<td><code>WRITE(102,1035) (NAME,I,UA(I),VA(I),I=1,KK)</code></td>
</tr>
<tr>
<td>0023</td>
<td><code>WRITE(102,2060)</code></td>
</tr>
<tr>
<td>0024</td>
<td><code>WRITE(102,2002) NAPP</code></td>
</tr>
<tr>
<td>0025</td>
<td><code>WRITE(102,2010) MAX</code></td>
</tr>
<tr>
<td>0026</td>
<td><code>WRITE(102,2020) EPS</code></td>
</tr>
<tr>
<td>0027</td>
<td><code>WRITE(102,2030) EPSM</code></td>
</tr>
<tr>
<td>0028</td>
<td><code>WRITE(102,2040) XSTART</code></td>
</tr>
<tr>
<td>0029</td>
<td><code>WRITE(102,2050) XEND</code></td>
</tr>
<tr>
<td>0030</td>
<td><code>IF(NP.NE.2) GO TO 15</code></td>
</tr>
<tr>
<td>0031</td>
<td><code>CALL QUAD(UA,VA,NP,URDO,T,VRDO,T,MULT,EPSM)</code></td>
</tr>
<tr>
<td>0032</td>
<td><code>WRITE(102,1037)</code></td>
</tr>
<tr>
<td>0033</td>
<td><code>WRITE(102,1096) (1,URDO(T),VRDO(T),MULT(I),I=1,NRODT)</code></td>
</tr>
<tr>
<td>0034</td>
<td><code>GO TO 10</code></td>
</tr>
<tr>
<td>0035</td>
<td><code>IF(NAPP.NE.0) GO TO 20</code></td>
</tr>
<tr>
<td>0036</td>
<td><code>NAPP=NP</code></td>
</tr>
<tr>
<td>0037</td>
<td><code>CALL GENAPP(UAPP,YAPP,NAPP,XSTART)</code></td>
</tr>
<tr>
<td>0038</td>
<td><code>GO TO 27</code></td>
</tr>
<tr>
<td>0039</td>
<td><code>READ(1030) (UAAPP,Z),VAPP(21,1),NAPP)</code></td>
</tr>
<tr>
<td>0040</td>
<td><code>DO 25 I=1,NAPP</code></td>
</tr>
<tr>
<td>0041</td>
<td><code>UAPP(1,1)=0.9*VAPP(1,2)</code></td>
</tr>
<tr>
<td>0042</td>
<td><code>VAPP(1,1)=0.9*VAPP(1,2)</code></td>
</tr>
</tbody>
</table>
TABLE C.V (Continued)

0043 UAPP(I,3) = 1.1*UAPP(I,2)
0044 VAPP(I,3) = 1.1*VAPP(I,2)
0045 KKK = NP + 1
0046 DO 30 I = 1, KKK
0047 UMOR(II) = UX(TI)
0048 VWORK(J) = VA(J)
0049 NWORK = NP
0050 U1 = UAPP(IAPP,1)
0051 UX1 = VAPP(IAPP,1)
0052 UX2 = UAPP(IAPP,2)
0053 VX2 = VAPP(IAPP,2)
0054 UX3 = UAPP(IAPP,3)
0055 VX3 = VAPP(IAPP,3)
0056 CALL HORNER(NWORK, UWORK, VWORK, UX1, VX1, UX2, V/X2, UX3, VX3)
0057 CALL HORNER(NWORK, UWORK, VWORK, UX1, VX1, UX2, V/X2, UX3, VX3)
0058 CALL HORNER(NWORK, UWORK, VWORK, UX1, VX1, UX2, V/X2, UX3, VX3)
0059 CALL HORNER(NWORK, UWORK, VWORK, UX1, VX1, UX2, V/X2, UX3, VX3)
0060 CALL HORNER(NWORK, UWORK, VWORK, UX1, VX1, UX2, V/X2, UX3, VX3)
0061 APRX4 = DSQRT(UPX4*UPX4 + VPX4*VPX4)
0062 APRX3 = DSQRT(UPX3*UPX3 + VPX3*VPX3)
0063 IF(IQEQ.EQ.0) GO TO 70
0064 UQ4 = 0.5*UQ4
0065 VQ4 = 0.5*VQ4
0066 UX4 = UX3 + (UH3*UQ4 - VH3*VQ4)
0067 VX4 = VX3 + (VH3*UQ4 - UH3*VQ4)
0068 GO TO 60
0069 IF(IAPP.EQ.0) GO TO 70
0070 IF(ITER .LT. MAX) GO TO 110
0071 IF(NALTER .GT. 5) GO TO 75
0072 IF(IAPP .EQ. IAPP) GO TO 100
0073 IF(XEND .EQ. 0) GO TO 77
0074 IF(XSTART .GT. XEND) GO TO 77
0075 CALL GENAPP(UAPP, VAPP, NAPP, XSTART, IAPP)
0076 CALL BETTER(UA, VA, NP, UROOT, VROOT, NROOT, UAPP, VAPP)
0077 WRITE(IO2, 1090)
0078 DO 55 I = 1, NAPP
0079 WRITE(IO2, 1080) 1, I, UROOT(I), VROOT(I), NROOT, UAPP(I), VAPP(I)
0080 55 WRITE(IO2, 1085) 1, UROOT(I), VROOT(I), NROOT, UAPP(I), VAPP(I)
TABLE C.V. (Continued)

0099 IF (IROOT.LT.NROOT) GO TO 85
0100 GO TO 87
0101 85 KKK = IROOT + 1
0102 WRITE (102,106) (1, UROOT(I), VROOT(I), MULT(I), I = KKK, NROOT)
0103 87 IF (IPATH .EQ. 1) GO TO 81
0104 GO TO 10
0105 90 WRITE (102,107) 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 NOMUX = NOMUX + 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 (NOMUX.LT.LP) GO TO 130
0124 GO TO 80
0125 130 CALL HORNTER(NWORK, UWORK, VWORK, UX4, VX4, UB, VB, UPX4, VPX4)
0126 NWORK = NWORK - 1
0127 KK = -NWORK + 1
0128 GO TO 140, 1, KKK
0129 UWORK(I) = UB(I)
0130 VWORK(I) = VB(I)
0131 CALL HORNTER(NWORK, UWORK, VWORK, UX4, VX4, UB, VB, UPX4, VPX4)
0132 CCC = DSQRT(UPX4*UPX4 + VPX4*VPX4)
0133 IF (CCC .LT. EPSI) GO TO 150
0134 IF (NWORK .GT. 2) GO TO 75
0135 IROOT = NROOT
0136 CALL QUADUWORK(NWORK, UWORK, VROOT, UROOT, VROOT, MULT, EPSI)
0137 GO TO 80
0138 150 MULT(NROOT) = MULT(NROOT) + 1
0139 NOMUX = NOMUX + 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 (12030.0)
0156 1020 FORMAT (14, 1x, 52) MULLER'S METHOD FOR FINDING THE ZEROS OF A POLYNOMIAL
### TABLE C.V (Continued)

<table>
<thead>
<tr>
<th>Polynomial Number</th>
<th>Degree</th>
<th>Coefficients of Deflated Polynomial</th>
<th>No. of Zeros Found</th>
<th>Initial Approximation</th>
<th>After the Attempt to Improve Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 11A/1H \times 1X/1H \times 18H )</td>
<td>12/12H</td>
<td>( 023.16 \times 3H + 023.16 \times 2H \times 023.16 \times 3H )</td>
<td>Did not apply</td>
<td>023.16, 023.16, 023.16, 023.16</td>
<td>023.16, 023.16, 023.16, 023.16</td>
</tr>
</tbody>
</table>

**Notes:**
- The coefficients of the deflated polynomial are given in the format \( \times \) unless otherwise noted.
- The initial and final approximations are listed for each polynomial.
- The attempt to improve accuracy was not applicable in some cases.

**Format:**
- Initial Approximation: Format is \( (1x) \times (1x) \times (1x) \times (1x) \) for the polynomial number.
- After the Attempt: Format is \( (1x) \times (1x) \times (1x) \times (1x) \) for the polynomial number.

**Additional Information:**
- The table continues with additional details on the polynomials and their approximations.
Subroutine ALTER alters the initial approximations which produce no convergence to a zero. This is done a maximum of 5 times for each root.

Common EPS1, EPS2, EPS3, IO2, MAX

If (ITIME .NE. 0) go to 5

Write io2, 1100

If (FINALTER .EQ. 0) go to 10

Write io2, 1000

Go to 20

R = DSQRT(X2R * X2R + X2 * X21)

BETA = DATAN2(X2I, X2R)

Write io2, 1020

XIR, XIIX2R, X21, X3R, X31

NALTER = NALTER + 1

If (NALTER .GT. 5) return

X2R = -X2R

X21 = -X21

Go to 50

BETA = BETA + 1.0471976

X2R = R * COS(BETA)

X21 = R * SIN(BETA)

X1R = 0.9 * X2R

X11 = 0.9 * X21

X3R = 1.1 * X2R

X3I = 1.1 * X21

Return


1010 FORMAT(IK, SHK = NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF ITER < 13, 12M IERATIONS.//)

End
SUBROUTINE GENAPP(APPR, APPI, NAPP, XSTART)

* SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE
* DEGREE OF THE ORIGINAL POLYNOMIAL.
*
DOUBLE PRECISION APPR, APP1, XSTART, EPS1, EPS2, EPS3, BETA
DIMENSION APPR(25, 3), APP1(25, 3)
COMMON EPS1, EPS2, EPS3, IO02, MAX

IF (XSTART.EQ.0.0) XSTART=0.5
BETA=0.2817994
DO 10 I=1, NAPP
APPR(1, 2)=XSTART*DOS(BETA)
APPR(1, 3)=XSTART*DOSIN(BETA)
BETA=BETA+0.5235988
10 XSTART=XSTART+0.5
DO 20 I=1, NAPP
APPR(1, 1)=0.9*APPR(1, 2)
APPR(1, 1)=0.9*APPR(1, 2)
20 APPI(1, 3)=1.1*APPI(1, 2)
RETURN
END
**TABLE C.V (Continued)**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td><strong>SUBROUTINE BETTER</strong></td>
<td>VA, NP, UROOT, VROOT, NROOT, URAPP, VRAPP, IROOT, MUL</td>
</tr>
<tr>
<td>0002</td>
<td><strong>DOUBLE PRECISION</strong></td>
<td>UROOT, VA, VBAPP, UX1, UX2, VX1, VX2, UX3, VX3</td>
</tr>
<tr>
<td>0003</td>
<td><strong>LOGICAL CONV</strong></td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td><strong>COMMON EPSRT, EPSO, EPSI02, NMAX</strong></td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td><strong>DO 10 I=1,NROOT</strong></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td><strong>UX1=UBAPP(J,1)</strong></td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td><strong>UX2=UBAPP(J,2)</strong></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td><strong>UX3=UBAPP(J,3)</strong></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td><strong>VX1=VBAPP(J,1)</strong></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td><strong>VX2=VBAPP(J,2)</strong></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td><strong>VX3=VBAPP(J,3)</strong></td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td><strong>CALL HORNER(NP, VA, UX1, VX1, UB, VB, UPX1, VPX1)</strong></td>
<td></td>
</tr>
<tr>
<td>0013</td>
<td><strong>CALL HORNER(NP, VA, UX2, VX2, UB, VB, UPX2, VPX2)</strong></td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td><strong>CALL HORNER(NP, VA, UX3, VX3, UB, VB, UPX3, VPX3)</strong></td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td><strong>CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UX4, VX4, UB, VB, UPX4, VPX4)</strong></td>
<td></td>
</tr>
<tr>
<td>0016</td>
<td><strong>CALL TEST(UX3, VX3, UX4, VX4, CONV)</strong></td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td><strong>IF (COND) GO TO 50</strong></td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td><strong>IF (ITER.LT.MAX) GO TO 40</strong></td>
<td></td>
</tr>
<tr>
<td>0019</td>
<td><strong>WRITE(100) J, UROOT(J), VRROOT(J), MAX</strong></td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td><strong>WRITE(100) UX4, VX4</strong></td>
<td></td>
</tr>
<tr>
<td>0021</td>
<td><strong>WRITE(100) UX3, VX3</strong></td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td><strong>IF (COND) GO TO 33</strong></td>
<td></td>
</tr>
<tr>
<td>0023</td>
<td><strong>IF (COND) GO TO 35</strong></td>
<td></td>
</tr>
<tr>
<td>0024</td>
<td><strong>IF (COND) GO TO 40</strong></td>
<td></td>
</tr>
<tr>
<td>0025</td>
<td><strong>IF (COND) GO TO 50</strong></td>
<td></td>
</tr>
<tr>
<td>0026</td>
<td><strong>IF (COND) GO TO 60</strong></td>
<td></td>
</tr>
<tr>
<td>0027</td>
<td><strong>IF (COND) GO TO 70</strong></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td><strong>IF (COND) GO TO 80</strong></td>
<td></td>
</tr>
<tr>
<td>0029</td>
<td><strong>IF (COND) GO TO 90</strong></td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td><strong>IF (COND) GO TO 100</strong></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td><strong>IF (COND) GO TO 110</strong></td>
<td></td>
</tr>
<tr>
<td>0032</td>
<td><strong>IF (COND) GO TO 120</strong></td>
<td></td>
</tr>
<tr>
<td>0033</td>
<td><strong>IF (COND) GO TO 130</strong></td>
<td></td>
</tr>
<tr>
<td>0034</td>
<td><strong>IF (COND) GO TO 140</strong></td>
<td></td>
</tr>
<tr>
<td>0035</td>
<td><strong>IF (COND) GO TO 150</strong></td>
<td></td>
</tr>
<tr>
<td>0036</td>
<td><strong>IF (COND) GO TO 160</strong></td>
<td></td>
</tr>
<tr>
<td>0037</td>
<td><strong>IF (COND) GO TO 170</strong></td>
<td></td>
</tr>
<tr>
<td>0038</td>
<td><strong>IF (COND) GO TO 180</strong></td>
<td></td>
</tr>
<tr>
<td>0039</td>
<td><strong>IF (COND) GO TO 190</strong></td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td><strong>IF (COND) GO TO 200</strong></td>
<td></td>
</tr>
<tr>
<td>0041</td>
<td><strong>IF (COND) GO TO 210</strong></td>
<td></td>
</tr>
<tr>
<td>0042</td>
<td><strong>IF (COND) GO TO 220</strong></td>
<td></td>
</tr>
<tr>
<td>0043</td>
<td><strong>IF (COND) GO TO 230</strong></td>
<td></td>
</tr>
<tr>
<td>0044</td>
<td><strong>IF (COND) GO TO 240</strong></td>
<td></td>
</tr>
<tr>
<td>0045</td>
<td><strong>IF (COND) GO TO 250</strong></td>
<td></td>
</tr>
</tbody>
</table>
TABLE C.V (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0046</td>
<td>VX1=UX2</td>
<td></td>
</tr>
<tr>
<td>0047</td>
<td>UX2=UX3</td>
<td></td>
</tr>
<tr>
<td>0048</td>
<td>VX2=UX3</td>
<td></td>
</tr>
<tr>
<td>0049</td>
<td>UX3=UX4</td>
<td></td>
</tr>
<tr>
<td>0050</td>
<td>VX3=UX4</td>
<td></td>
</tr>
<tr>
<td>0051</td>
<td>UPX1=UPX2</td>
<td></td>
</tr>
<tr>
<td>0052</td>
<td>VPX1=VPX2</td>
<td></td>
</tr>
<tr>
<td>0053</td>
<td>UPX2=UPX3</td>
<td></td>
</tr>
<tr>
<td>0054</td>
<td>VPX2=VPX3</td>
<td></td>
</tr>
<tr>
<td>0055</td>
<td>ITER=ITER+1</td>
<td></td>
</tr>
<tr>
<td>0056</td>
<td>GO TO 20</td>
<td></td>
</tr>
<tr>
<td>0057</td>
<td>UPX1=UPX2</td>
<td></td>
</tr>
<tr>
<td>0058</td>
<td>VPX1=VPX2</td>
<td></td>
</tr>
<tr>
<td>0059</td>
<td>UPX2=UPX3</td>
<td></td>
</tr>
<tr>
<td>0060</td>
<td>VPX2=VPX3</td>
<td></td>
</tr>
<tr>
<td>0061</td>
<td>ITER=ITER+1</td>
<td></td>
</tr>
<tr>
<td>0062</td>
<td>GO TO 20</td>
<td></td>
</tr>
<tr>
<td>0063</td>
<td>50 L=L+1</td>
<td></td>
</tr>
<tr>
<td>0064</td>
<td>UROOTS(L)=UX4</td>
<td></td>
</tr>
<tr>
<td>0065</td>
<td>VROOTS(L)=VX4</td>
<td></td>
</tr>
<tr>
<td>0066</td>
<td>MULT(L)=MULT(L)</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>CONTINUE</td>
<td></td>
</tr>
<tr>
<td>0067</td>
<td>IF(L.LE.0) GO TO 120</td>
<td></td>
</tr>
<tr>
<td>0068</td>
<td>DO 110 I=L+1</td>
<td></td>
</tr>
<tr>
<td>0069</td>
<td>UROOT(I)=UROOTS(I)</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>VROOT(I)=VROOTS(I)</td>
<td></td>
</tr>
<tr>
<td>0070</td>
<td>NROOT=NROOT+1</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>NROOT=0</td>
<td></td>
</tr>
<tr>
<td>0071</td>
<td>RETURN</td>
<td></td>
</tr>
</tbody>
</table>

In the attempt to improve accuracy, Root(12,4) = 1023.16,3 * 10-23.16,2 Did not converge after 13.11 iterations.

The present approximation is 1023.16,3 * 10-23.16,2.
### TABLE C.V (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE CALCI(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,UPX2,UPX3,Y</td>
<td>* GIVEN THREE APPROXIMATIONS XIN-3), XIN-2), AND XIN-1, SUBROUTINE CALC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZEROS OF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* THE QUADRATIC CLOSEST TO X(N). THIS ZERO IS THE NEW APPROXIMATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* XIN+1 TO THE ZERO OF THE POLYNOMIAL.</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION ARG1,ARG2</td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UPX1,VX1,UX2,UX3,VX3,UUX1,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VX3,UPX1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UH3+VH3,UH2+VH2,UH1+VH1,UH0+VH0,UH-1+VH-1,UH-2+VH-2,UH-3+VH-3</td>
</tr>
<tr>
<td>0004</td>
<td>UDH3=1.0+UH3</td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>COMMON EPSRT,EPSO,EP8,I2O,MAX</td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>UX3=UX3-UX2</td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>VX3+VX3-VX2</td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>UH2=UX2-UX1</td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>VX2=VX2-VX1</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>BBB=UH2<em>UH2+VH2</em>VH2</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>UQ3=(UH3<em>UH3+VH3</em>VH3)*BBB</td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td>VG3=(VH3<em>VH3+UH3</em>UH3)*BBB</td>
<td></td>
</tr>
<tr>
<td>0013</td>
<td>VDDD=1.0+VQ3</td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td>VDDD=VG3</td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td>UD=1(UPX3-(UDDD<em>UPX2+VDDD</em>VPX2)+((UQ3+UPX1-VQ3*VPX1)</td>
<td></td>
</tr>
<tr>
<td>0016</td>
<td>VD=(UPX3-(VDDD<em>UPX2+UDDD</em>VPX2)+(VQ3<em>UPX1+VQ3</em>VPX1)</td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td>UAAA=2.0*VQ3</td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td>UAAA=UAAA+1.0</td>
<td></td>
</tr>
<tr>
<td>0019</td>
<td>UBBB=UDDD<em>UDDD</em>VDDD</td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>VBVB=VDDD<em>VDDD</em>VDDD</td>
<td></td>
</tr>
<tr>
<td>0021</td>
<td>UCCC=UQ3<em>UQ3</em>VQ3</td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td>VC=VCCC<em>VCCC</em>VCCC</td>
<td></td>
</tr>
<tr>
<td>0023</td>
<td>UC**=UAAA<em>UPX3-UAAA</em>VPX3-(UBBB<em>VX3-UBBB</em>VX3)+(VCCC<em>UPX1+VCCC</em>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PX1)</td>
</tr>
<tr>
<td>0024</td>
<td>VB=(UAAA<em>UPX3+UAAA</em>VPX3)-(VBBB<em>UPX2+VBBB</em>VPX2)+(VCCC<em>UPX1+VCCC</em>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PX1)</td>
</tr>
<tr>
<td>0025</td>
<td>UC**=UDDD<em>UPX3+VDDD</em>VPX3</td>
<td></td>
</tr>
<tr>
<td>0026</td>
<td>VC=VDDD<em>UPX3</em>UDDD*VPX3</td>
<td></td>
</tr>
<tr>
<td>0027</td>
<td>UDISC=UR<em>UR</em>VR<em>VR-(UDDD</em>VDDD<em>VDDD</em>VDDD)+(VCCC<em>VCCC</em>VCCC)</td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td>VOISC=(2.0<em>VU</em>U1)-(4.0*(VDDD<em>VDDD</em>VDDD))</td>
<td></td>
</tr>
<tr>
<td>0029</td>
<td>AAAA=UDDORT(UDDORT<em>VDDORT</em>VDDORT)</td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td>IF(AAAA.EQ.0.0) GO TO 5</td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>GO TO 7</td>
<td></td>
</tr>
<tr>
<td>0032</td>
<td>SUBROUTINE DATA(DISO,DISC,UDISC)</td>
<td></td>
</tr>
<tr>
<td>0033</td>
<td>SUBROUTINE DATA(VDISC,UDISC)</td>
<td></td>
</tr>
<tr>
<td>0034</td>
<td>ANGLE=THETA+2.0</td>
<td></td>
</tr>
<tr>
<td>0035</td>
<td>THETA=DATA2(VDISC,UDISC)</td>
<td></td>
</tr>
<tr>
<td>0036</td>
<td>RAD=SUOR(aAAA</td>
<td></td>
</tr>
<tr>
<td>0037</td>
<td>ANGLE=THETA/2.0</td>
<td></td>
</tr>
<tr>
<td>0038</td>
<td>UTEST=RAYCOS(ANGLE)</td>
<td></td>
</tr>
<tr>
<td>0039</td>
<td>VTEST=RAD*DSIN(ANGLE)</td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td>UDDEN=UB+UTEST</td>
<td></td>
</tr>
<tr>
<td>0041</td>
<td>VDDEN=UB+VTEST</td>
<td></td>
</tr>
<tr>
<td>0042</td>
<td>UDDEN=UB+UTEST</td>
<td></td>
</tr>
<tr>
<td>0043</td>
<td>VDDEN=UB+VTEST</td>
<td></td>
</tr>
<tr>
<td>0044</td>
<td>ARG1=UDDEN<em>UDDEN+VDDEN</em>VDDEN</td>
<td></td>
</tr>
</tbody>
</table>
TABLE C.V (Continued)

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

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

C SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C IMATIONS BY TESTING THE EXPRESSION
C ABSOLUTE VALUE OF |x(n+1)-x(n)|/ABSOLUTE VALUE OF |x(n)|.
C WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C
C******************************************************************************
C                                                                                   
C IDENOM
C LOGICAL CONV
C COMMON EPSR, EPS, EPS, 102, MAX
C UDOMMY=UX4-UX3
C VDOMMY=VX4-VX3
C AAA=DSORT(UDOMMY+UDOMMY+UDOMMY+UDOMMY)
C DENOM=DSORT(UX4+UX4+UX4+UX4)
C IF(DENOM.EQ.EPS0) GO TO 20
C IF(AAA.GE.DENOM.LT.EPS) GO TO 10
C 5 CONV=.FALSE.
C 10 CONV=.TRUE.
C GO TO 100
C 100 RETURN
C 100 RETURN
C END

0001 SUBROUTINE HORNERR(MA,VA,UX,UX,UB,UB,UPX,VPX)

C HORNERR'S METHOD Computes the value of the polynomial pix) at a point x.
C SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE
C FACTOR (x-D).
C
C******************************************************************************
C                                                                                   
C DOUBLE PRECISION UX,UX,UPX,VPX,UB,UB,MA,VA
C DIMENSION UX(26),VX(26),UB(26),VB(26)
C UX(1)=UA(1)
C VB(1)=VA(1)
C NUM=NA+1
C GO TO 100 NUM
C UX(1)=UA(1)+(UB(1)-1)*UX-UX(1)+1*UX
C 10 VB(1)=VA(1)+(VB(1)-1)*UX+UB(1-1)*UX
C UPX=UB(NUM)
C VPX=VB(NUM)
C RETURN
C END
SUBROUTINE QUAD(UA, VA, UROOT, VROOT, NROOT, MULTI, EPST)

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

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

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

DOUBLE PRECISION UA, VA, UROOT, VROOT, BBB, UAAA, VAAA, UDISC, VDISC, UDUMMY

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

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

UROOT(NROOT+1) = 0.0
VROOT(NROOT+1) = 0.0
MULTI(NROOT+1) = 1
NROOT = NROOT + 1

GO TO 50

5 BBB = UA(1) * UA(1) + VA(1) * VA(1)

UROOT(NROOT+1) = (UA(1) - UA(2) * VA(1) / BBB)

VROOT(NROOT+1) = (VA(1) - UA(1) * UA(1) / BBB)

MULTI(NROOT+1) = 1
NROOT = NROOT + 1

GO TO 50

7 UDISC = UA(2) * UA(2) - VA(2) * VA(2)

VDISC = VA(2) * VA(2) - UA(2) * UA(2)

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

IF(BBB.LT.EPST) GO TO 10

CALL COMSOT(UDISC, VDISC, UDUMMY, VDUMMY)

UAAA = VA(1)

VAAA = UA(1)

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

UROOT(NROOT+1) = (UA(2) * UAAA - VA(2) * VAAA) / BBB

VROOT(NROOT+1) = (VA(2) * UAAA + UA(2) * VAAA) / BBB

MULTI(NROOT+1) = 1
NROOT = NROOT + 1

GO TO 50

10 UAAA = 2.0 * UA(1)

VAAA = 2.0 * VA(1)

BBB = UAAA * UAAA + VAAA * VAAA

UROOT(NROOT+1) = (UBBB * UAAA - VAAA * VAAA) / BBB

VROOT(NROOT+1) = (VAAA * UAAA - UAAA * VAAA) / BBB

MULTI(NROOT+1) = 2
NROOT = NROOT + 1

GO TO 50

50 RETURN

END
TABLE C.V (Continued)

SUBROUTINE COMSQT(UK,VX,UY,VY)
C
**************************************************************************
C
TH\n
* THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
C
**************************************************************************
C
DOUBLE PRECISION UX, UX, VX, VX, VY, VY, DUMMY, R, A, B

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

IF (UX) 10, 20, 30
10 UY = AAA

O 20 IF (UX) 40, 50, 60
40 DUMMY = DABS(UX)
50 UY = DSQRT(DUMMY)

100 RETURN
END
APPENDIX D

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

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

1. Generating Approximations

If the user does not have initial approximations available, subroutine GENAPP can systematically generate, for an \( N \)th degree polynomial, \( N \) initial approximations of increasing magnitude, beginning with the magnitude specified by XSTART. If XSTART is 0., XSTART is automatically initialized to 0.5 to avoid the approximation 0. + 0.1. The approximations are generated according to the formula:

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

where

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

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

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

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

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

Example D.2 shows a portion of a control card which generates initial approximations beginning at a magnitude of 25.0 for a sixth degree polynomial.
Note that if the approximations are generated beginning at the origin, the order in which the roots are found will probably be of increasing magnitude. Roots obtained in this way are usually more accurate.

2. Altering Approximations

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

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

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

where

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

If the number of the alteration is even: $(j = 0, 2, 4)$
\[ x_{j+1} = -x_j \]

Each altered approximation is then taken as a starting approximation. If none of the six starting approximations produce convergence, the next initial approximation is taken, and the process repeated. The six approximations are spaced 60 degrees apart on a circle of radius \( |x_0| \) centered at the origin as illustrated in Figure A.2.

3. Searching the Complex Plane

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

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

By generating initial approximations internally, the program can search an annulus centered at the origin of inner radius XSTART and outer radius XEND. Values for XSTART and XEND are supplied on the control card by the user. Example D.3 shows a portion of a control card to search the above annulus of inner radius 20.0 and outer radius 40.0.
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-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 \( \text{NAPP}, \text{XSTART}, \) and \( \text{XEND} \), 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>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>O</td>
<td>P</td>
<td>L</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XSTART</td>
</tr>
<tr>
<td>N</td>
<td>P</td>
<td>N</td>
<td>A</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>3</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.

```
1 2 4 5 7 8 6 7 7 8
N N XSTART
O P A XEND
L Y
1 7 2
```

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

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

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

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

**Control Information**

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

**TABLE E.II**

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

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOPOLY</td>
<td>c.c. 1-2</td>
<td>Number of the polynomial. Integer. Right justified.</td>
</tr>
<tr>
<td>NP</td>
<td>c.c. 4-5</td>
<td>Degree of the polynomial. Integer. Right justified.</td>
</tr>
<tr>
<td>NAPP</td>
<td>c.c. 7-8</td>
<td>Number of initial approximations to be read. Integer. Right justified. If no initial approximations are given, leave blank.</td>
</tr>
<tr>
<td>MAX</td>
<td>c.c. 19-21</td>
<td>Maximum number of iterations. Integer. Right justified. 200 is recommended.</td>
</tr>
<tr>
<td>EPS1</td>
<td>c.c. 23-28</td>
<td>Test for zero in subroutine GCD. Double precision. Right justify. 1.D-03 is recommended.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Card Columns</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| EPS2          | c.c. 30-35   | Convergence requirement.  
Double precision.  
Right justify.  
1.D-10 is recommended. |
| EPS3          | c.c. 37-42   | Test for zero in subroutine QUAD.  
Double precision.  
Right justify.  
1.D-20 is recommended. |
| EPS4          | c.c. 44-49   | Multiplicity requirement.  
Double precision.  
Right justify.  
1.D-02 is recommended. |
| XSTART        | c.c. 64-70   | Magnitude at which to begin generating initial approximations.  
Double precision.  
Right justify.  
This is a special feature of the program and may be omitted. |
| XEND          | c.c. 72-78   | Magnitude at which to end the generating of initial approximations.  
Double precision.  
Right justify.  
This is a special feature of the program and may be omitted. |
| KCHECK        | c.c. 80      | This should be left blank. |

**Coefficients of the Polynomial**

The coefficient cards follow the control card. For an \( N \)th degree polynomial, \( N+1 \) coefficients must be entered one per card. The coefficient of the highest degree term is entered first; that is, the leading coefficient is entered first. For example, if the polynomial \( X^5 + 3X^4 + 2X + 5 \) were to be solved for its zeros, the order in which
the coefficients would be entered is: 1, 3, 0, 0, 2, 5. Each real or complex coefficient is entered, one per card, as described in Table E.III and illustrated in Figure E.3.

<table>
<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
### Variable Name

<table>
<thead>
<tr>
<th>Card Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000011111111122222222333333333334444445555555555566666677777777778</td>
</tr>
<tr>
<td>1234567890123456789012345678901234567890123456789012345678901234567890</td>
</tr>
</tbody>
</table>

### EPS1

<table>
<thead>
<tr>
<th>EPS2</th>
<th>EPS3</th>
<th>EPS4</th>
<th>XSTART</th>
<th>XEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0D-03</td>
<td>1.0D-10</td>
<td>1.0D-20</td>
<td>1.0D+01</td>
<td>5.0D+02</td>
</tr>
</tbody>
</table>

### Example

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

### Variable Name

<table>
<thead>
<tr>
<th>Card Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000011111111122222222333333333334444445555555555566666677777777778</td>
</tr>
<tr>
<td>1234567890123456789012345678901234567890123456789012345678901234567890</td>
</tr>
</tbody>
</table>

### UP VP

<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.3. Coefficient Card for G.C.D. - Newton's Method**
Figure E.4. Initial Approximation Card for G.C.D. - Newton's Method

Figure E.5. End Card for G.C.D. - Newton's Method
### TABLE E.VI

VARIABLES USED IN G.C.D. - NEWTON'S METHOD

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

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NQ</td>
<td>I</td>
<td>NQ</td>
<td>I</td>
<td>Degree of quotient polynomial Q(X)</td>
</tr>
<tr>
<td>ZRO</td>
<td>C</td>
<td>UZRO, VZRO</td>
<td>D</td>
<td>Value at which to evaluate or deflate polynomial</td>
</tr>
<tr>
<td>DUMMY</td>
<td>C</td>
<td>UDUMMY, VDUMMY</td>
<td>D</td>
<td>Dummy variable</td>
</tr>
<tr>
<td>QQ</td>
<td>C</td>
<td>UQQ, VQQ</td>
<td>D</td>
<td>Working array of coefficients of current polynomial</td>
</tr>
<tr>
<td>NQQ</td>
<td>I</td>
<td>NQQ</td>
<td>I</td>
<td>Degree of current polynomial, QQ(X)</td>
</tr>
<tr>
<td>IALTER</td>
<td>I</td>
<td>IALTER</td>
<td>I</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>CONV = TRUE implies convergence to a root</td>
</tr>
<tr>
<td>EPS4</td>
<td>R</td>
<td>EPS4</td>
<td>D</td>
<td>Tolerance for checking multiplicities</td>
</tr>
<tr>
<td>AP</td>
<td>C</td>
<td>UAP, VAP</td>
<td>D</td>
<td>Array of approximations (initial or altered) producing convergence</td>
</tr>
<tr>
<td>QD</td>
<td>C</td>
<td>UQD, VQD</td>
<td>D</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>Number of distinct roots found by iterative process</td>
</tr>
<tr>
<td>JI</td>
<td>I</td>
<td>JI</td>
<td>I</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>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>Array of improved roots</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Program control. NEWT = TRUE implies that Newton's method was used instead of Subroutine QUAD</td>
</tr>
</tbody>
</table>

**Subroutine NEWTON**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>C</td>
<td>Starting approximation (initial or altered)</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>Degree of current polynomial</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>Array of coefficients of current polynomial</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>Maximum number of iterations</td>
</tr>
<tr>
<td>EPSLON</td>
<td>R</td>
<td>Tolerance for checking convergence</td>
</tr>
<tr>
<td>XO</td>
<td>C</td>
<td>Current approximation to root</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>Array of coefficients of newly deflated polynomial</td>
</tr>
<tr>
<td>DPXO</td>
<td>C</td>
<td>Derivative of the polynomial at XO</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Single Precision Variable Type</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>DIFF</td>
<td>C</td>
<td><strong>UDIFF, VDIFF</strong></td>
</tr>
<tr>
<td>PXO</td>
<td>C</td>
<td><strong>UPXO, VPXO</strong></td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
</tr>
</tbody>
</table>

**Subroutine HORNER**

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

**Subroutine QUAD**

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

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

Subroutine GCD

<table>
<thead>
<tr>
<th>Single Precision</th>
<th>Double Precision</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
<td>Variable Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N I N</td>
<td>P C UP,VP</td>
<td>E Array of coefficients of original polynomial, P(X)</td>
<td></td>
</tr>
<tr>
<td>J I J</td>
<td>I J</td>
<td>E Number of distinct roots of P(X)</td>
<td></td>
</tr>
<tr>
<td>ROOT C UROOT,VROOT</td>
<td></td>
<td>E Array of distinct roots of P(X)</td>
<td></td>
</tr>
<tr>
<td>A C UA,VA</td>
<td>M I M</td>
<td>Working array of coefficients of current polynomial</td>
<td></td>
</tr>
<tr>
<td>M I M</td>
<td>MULT I MULT</td>
<td>R Array of multiplicities of the roots</td>
<td></td>
</tr>
<tr>
<td>I02 I I02</td>
<td>B C UB,VB</td>
<td>C Unit number of output device</td>
<td></td>
</tr>
<tr>
<td>C C UC,VC</td>
<td>E C EPSLON</td>
<td>C Array of coefficients of newly deflated polynomial</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Derivative of polynomial at \text{ROOT}_4</td>
</tr>
<tr>
<td>EPSLON R EPSLON</td>
<td></td>
<td>C</td>
<td>Tolerance for checking multiplicities</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Double Precision Variable</td>
<td>Type</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>----------------------------</td>
<td>------</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>UP,VP</td>
<td>D</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA,VA</td>
<td>D</td>
</tr>
<tr>
<td>M</td>
<td>I</td>
<td>M</td>
<td>I</td>
</tr>
</tbody>
</table>

**Subroutine DERIV**

**Subroutine DIVIDE**

| P                         | C    | UP,VP                      | D    | E                       | Array of coefficients of dividend polynomial |
| N                         | I    | N                          | I    | E                       | Degree of dividend polynomial |
| D                         | C    | UD,VD                      | D    | E                       | Array of coefficients of divisor polynomial |
| M                         | I    | M                          | I    | E                       | Degree of divisor polynomial |
| Q                         | C    | UQ,VQ                      | D    | R                       | Array of coefficients of quotient polynomial P(X)/D(X) |
| K                         | I    | K                          | I    | R                       | Degree of quotient polynomial, Q(X) |
| J                         | I    | J                          | I    | Counter                 |
| TERM                      | C    | UTERM,VTERM                | D    | Dummy variable used for temporary storage of products |
| KK                        | I    | KK                         | I    | Number of coefficients of quotient polynomial, Q(X) |

**Subroutine GENAPP**

| APP                       | C    | APPR,APPI                  | D    | R                       | Array containing initial approximations |
| NAPP                      | I    | NAPP                       | I    | E                       | Number of initial approximations to be generated |
| XSTART                    | R    | XSTART                     | D    | ECR                     | Magnitude at which to begin generating approximations; also magnitude of the approximation being generated |
| BETA                      | R    | BETA                       | D    | Argument of complex approximation being generated |
| U                         | R    | APPR(I)                    | D    | Real part of complex approximation |
| V                         | R    | APPI(I)                    | D    | Imaginary part of complex approximation |
### Table E.VI (Continued)

<table>
<thead>
<tr>
<th>Single Precision</th>
<th>Double Precision</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Variable</td>
<td>Type</td>
</tr>
<tr>
<td>XOLD</td>
<td>C</td>
<td>XOLDR,XOLDI</td>
<td>D</td>
</tr>
<tr>
<td>NALTER</td>
<td>I</td>
<td>NALTER</td>
<td>I</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
</tr>
<tr>
<td>Y</td>
<td>R</td>
<td>XOLDI</td>
<td>D</td>
</tr>
<tr>
<td>X</td>
<td>R</td>
<td>XOLDR</td>
<td>D</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>ABXOLD</td>
<td>D</td>
</tr>
<tr>
<td>BETA</td>
<td>R</td>
<td>BETA</td>
<td>D</td>
</tr>
<tr>
<td>XOLDR</td>
<td>R</td>
<td>XOLDR</td>
<td>D</td>
</tr>
<tr>
<td>XOLDI</td>
<td>R</td>
<td>XOLDI</td>
<td>D</td>
</tr>
<tr>
<td>IO2</td>
<td>I</td>
<td>IO2</td>
<td>I</td>
</tr>
</tbody>
</table>

Subroutine ALTER

Subroutine COMSQT

<table>
<thead>
<tr>
<th>UX, VX</th>
<th>D</th>
<th>E</th>
<th>Complex number for which the square root is desired</th>
</tr>
</thead>
<tbody>
<tr>
<td>UY, VY</td>
<td>D</td>
<td>R</td>
<td>Square root of the complex number</td>
</tr>
</tbody>
</table>
4. Description of Program Output

The output from G.C.D. - Newton's method consists of the following information.

The heading is "GREATEST COMMON DIVISOR METHOD USED WITH NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS NUMBER XX." XX represents the number of the polynomial.

As an aid to ensure that the control information is correct, the number of initial approximations given, maximum number of iterations, test for zero in subroutine GCD, test for convergence, test for zero in subroutine QUAD, test for multiplicities, radius to start search, and radius to end search are printed as read from the control card.

The coefficients of the polynomial are printed under the heading "THE DEGREE OF P(X) IS XX THE COEFFICIENTS ARE." XX represents the degree of the polynomial. The coefficient of the highest degree term is printed first.

The polynomial obtained after dividing the original polynomial, P(X), by the greatest common divisor of P(X) and its derivative, P'(X), is printed under the heading "Q(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE DISTINCT ROOTS OF P(X). THE DEGREE OF Q(X) IS XX THE COEFFICIENTS ARE." XX represents the degree of this polynomial. This polynomial contains all distinct roots and is solved by Newton's method. The coefficient of the highest degree term is printed first; that is, the leading coefficient is printed first.

The zeros found before the attempt to improve accuracy are printed under the heading "ROOTS OF Q(X)."

The initial approximation producing convergence to a root is
printed to the right of the corresponding root and headed by "INITIAL APPROXIMATION." The initial approximations may be those supplied by the user, or generated by the program or a combination of both. The message "RESULTS OF SUBROUTINE QUAD" indicates that the corresponding root was obtained by subroutine QUAD. See Appendix D, § 5.

The zeros found after the attempt to improve accuracy are printed under the heading "ROOTS OF P(X)." The corresponding initial approximation producing convergence is printed as described above.

The multiplicity of each zero is given under the title "MULTIPLICITIES."

5. Informative Messages and Error Messages

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

If not all roots of a polynomial were found before the attempt to improve accuracy, the remaining unsolved polynomial will be printed, with the leading coefficient first, under the heading "COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." See Appendix D, § 6.

"NO ROOTS FOR INITIAL APPROXIMATION ROOT XX = YYY." This message is printed if a root fails to produce convergence when trying to improve accuracy. XX represents the number of the root and YYY represents the value of the root before the attempt to improve accuracy.

"NO ROOTS FOR THE POLYNOMIAL Q(X) OF DEGREE XX WITH GENERATED INITIAL APPROXIMATIONS." XX represents the degree of the polynomial Q(X). This message is printed if none of the roots produce convergence in the attempt to improve accuracy.
"THE EPSILON (XXX) CHECK IN SUBROUTINE MULTI INDICATES THAT ROOT YY = ZZZ IS NOT CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIPLICITY 0." XXX represents the multiplicity requirement (EPS4 on the control card), YY represents the number of the root, and ZZZ represents the value of the root after the attempt to improve accuracy. The message indicates that this root does not meet the requirement for multiplicities. It is, however, usually a good approximation to the true root since convergence was obtained both before and after the attempt to improve accuracy.
Figure E.6. Flow Charts for 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)
COMSQT

Figure E.6. (Continued)
Figure E.6. (Continued)
TABLE E.VII

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

******************************************************************************
C
C * DOUBLE PRECISION PROGRAM FOR G.C.D. - NEWTON'S METHOD
C
C * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPlicITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY
C * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL
C * AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED
C * AND THEIR MULTIPlicILITIES DETERMINED.
C
******************************************************************************
0001  DOUBLE PRECISION UP,VP,UAPP,YAPP,URoot,VRoot1,UDP,VDP,UD,VD,U2D,VD2
0002  1K0,UD,VD,UOUMMY,VUOUMMY,VOO,VAPP,VOAP,VAPP,VOAP,VOO,URoots,VRoots, EPS1,
0003  2EPS2, EPS3, EPS4.
0004  DOUBLE PRECISION XSTART
0005  DIMENSION UP(261),VP(261),UAPP(251),YAPP(251),URoot(251),VRoot(251),MULT
0006  1(251),UDP(261),VDP(261),UD(261),VD(261),UO(261),VO(261),U2D(261),VD2(261),U
0007  2APP(251),VP(251),URoots(251), VRoots(251), VRoots(251), EPS1,
0008  COMMON EPS1, EPS2, EPS3, EPS4, 1O2, MAX
0009  INTEGER NEWT, CONV
0010  DATA Pharea, OName = OName/2HIP, 2HIP, 3HIP/F
0011  DATA ENTRY/(1H,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,2H12,2H13
0012  1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
0013  DATA AName(1), AName(1) /4HNEWT, 4HDNS /
0014  10 01
0015  10 10
0016  10 J=0
0017  10 ITIME = 0
0018  READ(101, 1000) NOPDOLY, NP, NAPP, MAX, EPS1, EPS2, EPS3, EPS4, XSTART, XEND, ICHECK
0019  IF (ICHECK. EQ. 11) STOP
0020  WRITE(102, 10201) AName(1), AName(1), NOPDOLY
0021  WRITE(102, 2003) NAPP
0022  WRITE(102, 2010) MAX
0023  WRITE(102, 2001) EPS1
0024  WRITE(102, 2070) EPS2
0025  WRITE(102, 2040) EPS3
0026  WRITE(102, 2030) EPS4
0027  WRITE(102, 2040) XSTART
0028  WRITE(102, 2050) XEND
0029  WRITE(102, 2060) 
0030  KKK = NP + 1
0031  NN = KKK + 1
0032  DO 20 I = 1, KKK
0033  JJJ = NN
0034  20 READ(101, 1010) UP(IJJJ), VP(IJJJ)
0035  IF(NAPP. EQ. 01) GO TO 22
0036  NAPP = NP
0037  CALL GENAPP(UAPP, YAPP, NAPP, XSTART)
0038  GO TO 23
0039  22 READ(101, 1015) (UAPP(I), YAPP(I), I = 1, NAPP)
0040  23 WRITE(102, 1030) NP
0041  KKK = NP + 1
0042  NN = KKK + 1
0043  DO 25 I = 1, KKK
0044  25
TABLE E.VII (Continued)

```plaintext
0040      JJJ=NNN-1
0041      WRITE(02,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
0042      IF(NP.GE.3) GO TO 30
0043      J=1
0044      CALL QUAD(NP,UP,VP,J,UROOT,VROOT,MULT)
0045      WRITE(02,1070)
0046      WRITE(02,1165) [UROOT,J],VROOT,MULT,1=1,J
0047      GO TO 10
0048  30 CALL DERIV(NP,UP,VP,NP,UP,VP)
0049      CALL GCDF(NP,VP,NP,VP,NP,VP,NP,VP)
0050      IF(NP.NE.3) GO TO 60
0051      NP=NP+1
0052      UDDUMM=UDI(1)*UDI(2)+UDI(3)*UDI(4)
0053      UZRO=UDI(1)*UDI(2)-UDI(3)*UDI(4)/UDDUMM
0054      VZRO=UDI(1)*UDI(2)-UDI(3)*UDI(4)/UDDUMM
0055      CALL HORNER(UZRO,VZRO,UP,VP,VP,VP,UDDUMM)
0056      NQ=NP+1
0057      DO 60 I=1,NP
0058      UQ(I)=UQ(I+1)
0059      VQ(I)=VQ(I+1)
0060      GO TO 80
0061  65 KKK=NP+1
0062      DO 66 I=1,KKK
0063      UQ(I)=UQ(I)
0064      VQ(I)=VQ(I)
0065      NQ=NP
0066      GO TO 80
0067  70 CALL DIVIDE(NP,UP,VP,NP,UP,VP)
0068      WRITE(02,1210) NQ
0069      KKK=NP+1
0070      NNN=KKK+1
0071      DO 83 I=1,NNN
0072      JJJ=NNN-1
0073      WRITE(02,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
0074      IF(NP.GE.3) GO TO 85
0075      GO TO 110
0076  85 KKK=NP+1
0077      DO 90 I=1,KKK
0078      UQ(I)=UQ(I)
0079      VQ(I)=VQ(I)
0080      GO TO 83
0081  90 UQ(I)=UQ(I)
0082      GO TO 120
0083  110 CALL QUAD(NP,UP,VP,J,UROOT,VROOT,MULT)
0084      NEWT=.FALSE.
0085      GO TO 310
0086  120 DO 200 I=1,NAPP
0087      IALTER=0
0088      CALL NEWTON(UAPP(I),VAPP(I),NQQ,QQQ,VQQ,UZRO,VZRO,CONV)
0089      IF(CONV) GO TO 160
0090      CALL ALTER(UAPP(I),VAPP(I),1,1,1)
0091      IF(IALTER.GT.5) GO TO 200
0092      GO TO 130
0093  130 J=J+1
0094      UROOT(J)=UZRO
0095      VROOT(J)=VZRO
0096      UAPP(J)=UAPP(J)
0097      VAPP(J)=VAPP(J)
```

TABLE E.VII (Continued)

```
0098      DO 100 I=1,NQQ
0099      UQQ(I)=UQQ(I)+1
0100 100  VOQ(I)=VOQ(I)+1
0101      NQQ=NQQ-1
0102      IF(NQQ.LT.3) GO TO 220
0103 200 CONTINUE
0104      IF(JJ.EQ.NQ) GO TO 205
0105      IF(XEND.EQ.0) GO TO 205
0106      IF(XSTART.GT.XEND) GO TO 205
0107      NAPP=NQ
0108      CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0109      GO TO 120
0110 205  IF(NQQ.LE.2) GO TO 210
0111      WRITE(102,1200)
0112      KKK=NQQ+1
0113      NNN=KKK+1
0114      DD 157 L=1,KKK
0115      JJJ=NNN-L
0116      157 WRITE(102,1100) QQNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0117      210 IF(JJ.EQ.0) GO TO 10
0118      JAP=J
0119      GO TO 230
0120      220 JAP=J
0121      CALL QUADINGQ(UQQ,VQQ,J,UROOT,MULT)
0122      230 WRITE(102,1133) (I,UROOT(I),VROOT(I),UAP(I),VAP(I),I=1,JAP)
0123      IF(JJ.EQ.NQ) GO TO 235
0124      GO TO 240
0125      235 KKK=JAP+1
0126      WRITE(102,1134) (I,UROOT(I),VROOT(I),I=KKK,J)
0127      240 JJ=0
0128      DO 300 I=1,J
0129      300 DD 300 I=1,J
0130      CALL NEWTON(UROOT(I),VROOT(I),UQ,VQ,UZRO,VZRO,CONV)
0131      IF(0) GO TO 280
0132      WRITE(102,1140)(I,UROOT(I),VROOT(I))
0133      IF(JJ+1.EQ.JAP) GO TO 241
0134      IF(JJ.EQ.JAP) GO TO 250
0135      GO TO 300
0136      241 KKK=JAP+1
0137      DD 245 I=1,KKK
0138      UAP(I)=UAP(I)+1
0139      245 VAP(I)=VAP(I)+1
0140      050 JAP=JAP+1
0141      GO TO 300
0142      280 JJ=J+1
0143      UROOT(JJ)=UZRO
0144      VROOT(JJ)=VZRO
0145      300 CONTINUE
0146      IF(JJ.EQ.0) GO TO 305
0147      JJ=J
0148      DO 303 I=1,J
0149      303 UROOT(I)=UROOT(I)
0150      305 WRITE(102,1150) NQ
0151      KKK=NQ+1
0152      NNN=KKK+1
0153      DD 306 I=1,KKK
```
TABLE E.VII (Continued)

0156  JJJ=NNN-L
0157  306 WRITE(IO,1040) QNAME,ENTRYJJJJJ,JD(JJJJ),YQ(JJJJ)
0158  GO TO 10
0159  307 NEWT=.TRUE.
0160  310 CALL MULTI(NPUPeVPeJUROOTVROOT.MULTI)
0161  IF(NEWT) GO TO 330
0162  WRITE(IO,1070)
0163  WRITE(IO,1165) (L,UROOT(L),VROOT(L),MULTIL),L=1,J
0164  GO TO 10
0165  330 WRITE(IO,1160)
0166  WRITE(IO,1190) (L,UROOT(L),VROOT(L),MULTIL,UAPIL,VAPIL),L=1,JAP
0167  KKK=JAP+1
0168  IF(JAP.LT.J) WRITE(IO,165)
0169  GO TO 10
0170  1000 FORMAT(3(I2,1X,x,13X(06.E0,0,1X,13X(07.,0,1X,11)
0171  1010 FORMAT(2D30.0)
0172  1015 FORMAT(2030.0)
0173  1020 FORMAT(I3,2,1X),9X,I3,1X,406.0,X111)
0174  1030 FORMAT(I3,2,1X),9X,I3,1X,21A4)
0175  1040 FORMAT(I3,2,1X),9X,I3,1X,4H)
0176  1070 FORMAT(I3,2,1X),13HROOTS OF P(X),52X,14HMULTIPLICITIES//
0177  1080 FORMAT(I3,2,1X),13HROOT(912,4H)
0178  1100 FORMAT(I3,2,1X),13HROOT(12,94HI)
0179  1120 FORMAT//IX,7OHCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH
0180  1130 FORMAT//IX,7OHNO ROOTS FOR THE POLYNOMIAL
0181  1140 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0182  1150 FORMAT//IX,7OHCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH
0183  1160 FORMAT//IX,7OHNO ROOTS FOR INITIALIZED APPROXIMATION ROOT
0184  1170 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0185  1180 FORMAT//IX,7OHNO ROOTS FOR THE POLYNOMIAL
0186  1190 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0187  1200 FORMAT//IX,7OHNO ROOTS FOR INITIAL APPROXIMATION ROOT
0188  1210 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0189  1220 FORMAT//IX,7OHNO ROOTS FOR THE POLYNOMIAL
0190  1230 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0191  1240 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0192  1250 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0193  1260 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0194  1270 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0195  1280 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0196  1290 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0197  1300 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0198  1310 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0199  1320 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0200  1330 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0201  1340 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0202  1350 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0203  1360 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0204  1370 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0205  1380 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0206  1390 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0207  1400 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0208  1410 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0209  1420 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0210  1430 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0211  1440 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0212  1450 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0213  1460 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0214  1470 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0215  1480 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0216  1490 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0217  1500 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0218  1510 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0219  1520 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0220  1530 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0221  1540 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0222  1550 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0223  1560 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0224  1570 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0225  1580 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0226  1590 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0227  1600 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0228  1610 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0229  1620 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0230  1630 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0231  1640 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0232  1650 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0233  1660 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0234  1670 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0235  1680 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0236  1690 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0237  1700 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0238  1710 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0239  1720 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0240  1730 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0241  1740 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
0242  1750 FORMAT//IX,7OHROOTS OF P(X),/IX,22HTHE DEGREE OF
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 ******************************************************************************************
0002 DOUBLE PRECISION APPR,APP1,XSTART,BETA, EPS1, EPS2, EPS3, EPS4
0003 DIMENSION APPR(25), APP1(25)
0004 COMMON EPS1, EPS2, EPS3, EPS4, I02, MAX
0005 IF(XSTART.EQ.0.0) XSTART=0.5
0006 BETA=0.2617994
0007 DO 10 I=1,NAPP
0008 APPRI|=XSTART*DCOS(BETA)
0009 APP1(I)=XSTART*DSIN(BETA)
0010 BETA=BETA+0.5235988
10 RETURN
0011 XSTART=XSTART+0.5
0012 RETURN
0013 END
```
TABLE E.VII (Continued)

0001 SUBROUTINE ALTER(KOLDR,XOLDI,NALTER, ITIME)
       C
       C E
       C *
       C E SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO
       C E CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT.
       C
       C
       C ************************************************************************************

0002      DOUBLE PRECISION KOLDR,XOLDI,ABXOLD, BETA, EPS1, EPS2, EPS3, EPS4

0003 COMMON EPS1,EPS2,EPS3,EPS4,ITIME,NALTER

0004      IF( ITIME.NE.00 ) GO TO 5

0005      ITIME = ITIME +1

0006      WRITE(I02,1010) MAX

0007      5 IF(NALTER.EQ.01) GO TO 10

0008      WRITE(I02,1000) XOLDR,XOLDI

0009      GO TO 20

0010      10 ABXOLD=DSQRT(KOLDR*KOLDR)+(XOLDI*KOLDI)

0011      BETA=DATAN2(KOLDI,KOLDR)

0012      WRITE(I02,1020) XOLDR,XOLDI

0013      20 NALTER=NALTER+1

0014      IF(NALTER.LT.5) RETURN

0015      GO TO (30,40,30,40,30,1,NALTER)

0016      30 XOLDR--KOLDR

0017      XOLDI=KOLDI

0018      GO TO 50

0019      40 BETA=BETA+1.0847976

0020      XOLDR=ABXOLD+DCOS(BETA)

0021      XOLDI=ABXOLD+DSIN(BETA)

0022      50 RETURN

0023      1000 FORMAT(IX,023.16,5H + .023.16, 2H 1,10X,21HALTED APPROXIMATION)

0024      1010 FORMAT(///IX,5H NON CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF

0025      ITER, IT,124 ITERATIONS.//)

0026      1020 FORMAT(///IX,023.16,5H + .023.16, 2H 1,10X,21INITIAL APPROXIMATION)

0027      END
SUBROUTINE GCD(IN, UR, VR, US, VS, Ml, MPS, VSS)

C

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

C

* GIVEN POLYNOMIALS P(X) AND D(PX) WHERE DEG. D(PX) IS LESS THAN DEG.

C

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

C

* D(PX).

C

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

DOUBLE PRECISION USSSSS, VSSS

DOUBLE PRECISION UR, VR, US, VS, USS, URR, UVR, UD, UT, VT, EPSLON, EP

DIMENSION UR(26), VR(26), US(26), VS(26), USS(26), VSS(26), URR(26), VRR(26)

COMMON EPSLON, EPS2, EPS3, EPS4, IO2, MAX

N1 = N

M1 = M

KKK = N + 1

DO 10 I = 1, KKK

10 VRR(I) = VR(I)

KKK = M + 1

DO 25 I = 1, KKK

25 USSII) = USII)

25 VSSII) = VS(I)

BBB = USSIMI +1) * USSIMI +1) + VSSIMI +1) * VSSIMI +1)

UD = URR(N1 +1) * USSIMI +1) + VRR(N1 +1) * VSSIMI +1) / BBB

VD = URR(N1 +1) * VRR(N1 +1) - URR(N1 +1) * VSSIMI +1) / BBB

KKK = N1 +1 - M1

DO 40 I = KKK, N1

40 UT(I) = URR(I)

40 VT(I) = VRR(I)

70 DO 90 I = 1, M1

90 BBB = USS(II) * USS(MI +1) + VSS(II) * VSS(MI +1) / BBB

VSSS = (USS(II) * VSS(MI +1) - USS(MI +1) * VSS(II)) / BBB

USS(I) = USSS

VSS(I) = VSSS

USS(MI +1) = 1.0

VSS(MI +1) = 0.0

GO TO 200

100 K = N1 - I

100 IF(K.EQ.0) GO TO 170

104 IF(K.LT. M1) GO TO 140

K = K +1

GO 130 J = 1, KKK

130 URR(J) = UT(J)

GO 30
TABLE E.VII (Continued)

140 KKK=K+1
DO 150 J=1,KK
URR(J)=USS(J)
VRR(J)=VSS(J)
USS(J)=U(J)
VSS(J)=V(J)
150 KKK=KK+2
NNN=M1+1
DO 160 J=KKK,NNN
URR(J)=USS(J)
VRR(J)=VSS(J)
160 M1=M1
M1=K
GO TO 30
170 USS(J)=1.0
VSS(J)=0.0
M1=0
200 RETURN
END
SUBROUTINE QUAD(NUAVAJUROOTVROOTMULT)

DOUBLE PRECISION UAVAtUROOTVROOTUDISCVDISCUTEMPVTEMPUDOVDOE.

DIMENSION UAVAI26|,VA26, UROOTI25)VROOT(25),MULT(25)

COMMON EPS1,EPS2EPSLONEPS4,102,MAX

IF(N.GT.1) GO TO 60

IF(JJ.LT.0) GO TO 40

J=J+1

GO TO 50

40 MULTII=I

J=1

50 BBB=UA(2I)*UA(2)+VA(I)*VA(I)

UROOT(J)=(UA(II)*UA(2)-VA(I)*VA(I))/(BBB)

VROOT(J)=(VA(I)*UA(2)-UA(I)*VA(I))/(BBB)

GO TO 200

60 VDISC=(UA(I)*UA(2)-VA(I)*VA(I)-4.0*UA(I)*VA(I)*VA(I))/(UA(I)*UA(I))

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

IF(BBB.LE.EPSLON) GO TO 100

IF(J.GE.O) GO TO 80

MULT(2)=1

80 CALL COMSQTIUDISC.VDISCUTEMPVTEMP)

UD=2.0*UA(3)

VD=2.0*VA(3)

BBB=UD*UD+VD*VD

UROOT(J+1)=((-UA(I)*UTEMP)*UD+-VA(I)*VTEMP)*VDI/BBB

VROOT(J+1)=(-UA(I)*VTEMP)*UD+-VA(I)*UTEMP)*VDI/BBB

UROOT(J+2)=((-VA(I)*UTEMP)*UD+-UA(I)*VTEMP)*VDI/BBB

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

J=J+2

GO TO 200

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

J=J+1

110 MULTII=2

130 UD=2.0*UA(3)

VD=2.0*VA(3)

BBB=UD*UD+VD*VD

UROOT(J1)=((-UA(I)*UTEMP)*UD+-VA(I)*VTEMP)*VDI/BBB

VROOT(J1)=(-UA(I)*VTEMP)*UD+-VA(I)*UTEMP)*VDI/BBB

200 RETURN

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

```fortran
SUBROUTINE NEWTON(UX, VX, N, UP, VP, UXO, VXO, CONV)
C
C
C THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-
C INATION BY USING THE ITERATION FORMULA
C
C X(N+1) = X(N) - P(X(N))/P'(X(N)).
C
C
******************************************************************************
DOUBLE PRECISION UX, VX, UP, VP, UXO, VXO, UB, VPXO, VDPXO, U, VPXO, VDPXO, U
IDIFF, VDIFF, EPS1, EPSLON, EPS3, EPS4, AAA, BBB
DOUBLE PRECISION DDD
DIMENSION UP(26), VP(26), UB(26), VPXO(26), VDPXO(26)
COMMON EPS1, EPSLON, EPS3, EPS4, IO2, MAX
LOGICAL CONV
UXO = UX
VXO = VX
DO 10 I = 1, MAX
CALL HORNER(UXO, VXO, N, UP, VP, UB, VPXO, VDPXO)
10 CONTINUE
IF (AAA .LE. EPSLON) GO TO 20
15 CONV = .FALSE.
RETURN
20 CONV = .TRUE.
RETURN
END
```
TABLE E.VII (Continued)

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

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

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

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

K=N-M

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

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

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

IF(K.EQ.0) GO TO 100

J=1

DO 50 J=1,K

UTERM=UP(N-J)

VTERM=VP(N-J)

KK=K+1

NN=N-J

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

UTERM=UTERM-UQ(KK)*UD(M+1)-VQ(KK)*VD(M+1)

VTERM=VTERM-UQ(KK)*VD(M+1)+VQ(KK)*UD(M+1)

50 T0.10

GO TO 45

10 IF(K.LT.1) GO TO 20

GO TO 40

20 UTERMN=UTERM-UQ(KK)*UD(M+1)-VQ(KK)*VD(M+1)

VTERN=VTERM-UQ(KK)*VD(M+1)+VQ(KK)*UD(M+1)

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

UQ(K+1)=UTERN*UD(M+1)+VTERN*VD(M+1)/UDUMMY

VQ(K+1)=UTERN*VD(M+1)-VTERN*UD(M+1)/UDUMMY

50 T0.10

100 RETURN

END
TABLE E.VII (Continued)

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

0001 SUBROUTINE DERIV(N, UP, VP, M, UA, VA)
C
C * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV Computes THE COEFFICIENTS OF ITS DERIVATIVE P'(X).
C
C
0002 DOUBLE PRECISION UP, VP, UA, VA, AAA
0003 DIMENSION UP(26), VP(26), UA(26), VA(26)
0004 KKK=N+1
0005 DO 10 I=2, KKK
0006 AAA=1
0007 UA(I-1)=AAA*UP(I)
0008 VA(I-1)=AAA*VP(I)
0009 M=N+1
0010 RETURN
0011 END
### TABLE E.VII (Continued)

```plaintext
0001 SUBROUTINE MULTI(N,UP,VP,J,UROOT,VROOT,MULT)
0002 ************
0003 * GIVEN N ZEROS OF A POLYNOMIAL, SUBROUTINE MULTI COMPUTES THEIR
0004 * MULTICITIES.
0005 ************
0006 DOUBLE PRECISION UP,VP,UROOT,VROOT,UA,VA,UB,VB,VC,EPS1,EPS2,EPS
0007 DIMENSION UP(26),VP(26),UROOT(25),VROOT(25),UA(26),VA(26),UB(26),V
0008 COMMON EPS1,EPS2,EPS3,EPSLN,IO2,MAX
0009 DO 100 I=1,J
0010 M=N
0011 MULT(I)=0
0012 20 CALL HORNERT(UROOT(I),VROOT(I),M,UA,VA,UB,VB,VC)
0013 BB=(U(I)+V(I)+UB(I)+VB(I))
0014 IF(BBB.LT.EPSLN) GO TO 50
0015 IF(MULT(1).EQ.0) GO TO 40
0016 GO TO 100
0017 40 WRITE(102,1000) EPSLN,1,UROOT(I),VROOT(I)
0018 GO TO 100
0019 50 MULT(I)=MULT(I)+1
0020 IF(M.GT.1) GO TO 60
0021 GO TO 100
0022 60 GO TO K+1,M
0023 UA(K)=UB(K)
0024 VB(K)=V(K)
0025 M=M-1
0026 GO TO 20
0027 100 CONTINUE
0028 RETURN
0029 1000 FORMAT(115H THE EPSILON 1.010,3,48H CHECK IN SUBROUTINE MULTI
0030 1NDICATES THAT ROOT(I),2,4H = ~023,16,3H + ~023,16+2H 1/80H IS NO
0031 2F CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP
0032 LICITY 0/1)
0033 END
```
TABLE E.VII (Continued)

0001  SUBROUTINE CONSQRT(UX, VX, UY, VY)
0002  !-----------------------------------------------------------
0003  ! THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
0004  !-----------------------------------------------------------
0005  DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, AAA, BBB
0006  R = DSQRT (UX**2 + VX**2)
0007  AAA = DSQRT (DABS ((R + UX)/2.0))
0008  BBB = DSQRT (DABS ((R - UX)/2.0))
0009  IF (UX) 10, 20, 30
0010  10  UY = AAA
0011  20  VY = -1.0 * BBB
0012  30  GO TO 100
0013  40  IF (VX) 40, 50, 60
0014  50  UY = 0.0
0015  60  VY = DSQRT (DUMMY)
0016  70  GO TO 100
0017  80  UY = 0.0
0018  90  VY = 0.0
0019  100  RETURN
0020  END
APPENDIX F

G.C.D. - MULLER'S METHOD

1. Use of the Program

A double precision FORTRAN IV program using the G.C.D. method with Muller's method as a supporting method is presented here. Flow charts for this program are given in Figure F.1 while Table F.III gives a FORTRAN IV listing of this program. Single precision variables are listed in Table F.II. The single precision variables are used in the flow charts and the corresponding double precision variables can be obtained from Table F.II.

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

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

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

Main Program

Data Entry/1H1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1
URAPP(N,3), VRAPP(N,3)
UAPP(N,3), VAPP(N,3)
UP(N+1), VP(N+1)
UROOT(N), VROOT(N)
MULT(N)
UDP(N+1), VDP(N+1)
UD(N+1),VD(N+1)
UQ(N+1), VQ(N+1)
UQQ(N+1), VQQ(N+1)
UB(N+1), VB(N+1)
ENTRY(N+1)

Subroutines MULTI, DIVIDE, DERIV, GCD, and QUAD

See corresponding subroutines in Table E.I.

Subroutine MULLER

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

Subroutine BETTER

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

Subroutine GENAPP

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

Subroutine HORNER

UA(N+1), VA(N+1)
UB(N+1), VB(N+1)
2. Input Data for G.C.D. - Muller's Method

The input data for G.C.D. - Muller's method is prepared exactly as described in Appendix E, § 2 for G.C.D. - Newton's method.

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

The main variables used in G.C.D. - Muller's method are given in Table F.II. The symbols used to indicate type and disposition are described in Appendix E, § 3. For variables not listed in Table F.II, see the main program or corresponding subprogram of Table E.VI.

4. Description of Program Output

The output from G.C.D. - Muller's method is identical to that for G.C.D. - Newton's method as described in 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 \(.9x_0\) 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 F.II
VARIABLES USED IN G.C.D. - MULLER'S METHOD

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Double Precision Variable</th>
<th>Disposition of Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Precision Type</td>
<td>Double Precision Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>NP</td>
<td>E</td>
<td>Degree of polynomial P(X)</td>
</tr>
<tr>
<td>NROOT</td>
<td>NROOT</td>
<td>R</td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>NOMULT</td>
<td>NOMULT</td>
<td>I</td>
<td>Number of roots (counting multiplicities)</td>
</tr>
<tr>
<td>ROOT</td>
<td>UROOT, VROOT</td>
<td>D</td>
<td>Array containing the roots</td>
</tr>
<tr>
<td>NAPP</td>
<td>UAPP, VAPP</td>
<td>D</td>
<td>Number of initial approximations to be read in</td>
</tr>
<tr>
<td>APP</td>
<td>UAPP, VAPP</td>
<td>E</td>
<td>Array of initial approximations</td>
</tr>
<tr>
<td>WORK</td>
<td>UWORK, VWORK</td>
<td>D</td>
<td>Working array containing coefficients of current polynomial</td>
</tr>
<tr>
<td>B</td>
<td>UB, VB</td>
<td>D</td>
<td>Array containing coefficients of deflated polynomial</td>
</tr>
<tr>
<td>A</td>
<td>UA, VA</td>
<td>E</td>
<td>Array containing coefficients of original polynomial, P(X)</td>
</tr>
<tr>
<td>RAPP</td>
<td>URAPP, VRAPP</td>
<td>D</td>
<td>Array of initial or altered approximation for which convergence was obtained</td>
</tr>
<tr>
<td>X1</td>
<td>UX1, VX1</td>
<td>D</td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X2</td>
<td>UX2, VX2</td>
<td>D</td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X3</td>
<td>UX3, VX3</td>
<td>D</td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>PX1</td>
<td>UPX1, VPX1</td>
<td>D</td>
<td>Value of polynomial P(X) at X1</td>
</tr>
<tr>
<td>PX2</td>
<td>UPX2, VPX2</td>
<td>D</td>
<td>Value of polynomial P(X) at X2</td>
</tr>
<tr>
<td>PX3</td>
<td>UPX3, VPX3</td>
<td>D</td>
<td>Value of polynomial P(X) at X3</td>
</tr>
<tr>
<td>X4</td>
<td>UX4, VX4</td>
<td>D</td>
<td>Newest approximation (X_{n+1}) to root</td>
</tr>
<tr>
<td>PX4</td>
<td>UPX4, VPX4</td>
<td>D</td>
<td>Value of polynomial P(X) at X4</td>
</tr>
<tr>
<td>MULT</td>
<td>MULT</td>
<td>I</td>
<td>Array containing the multiplicities of each root found</td>
</tr>
<tr>
<td>ITER</td>
<td>ITER</td>
<td>I</td>
<td>Counter for iterations</td>
</tr>
<tr>
<td>IO1</td>
<td>IO1</td>
<td>I</td>
<td>Unit number of input device</td>
</tr>
<tr>
<td>IO2</td>
<td>IO2</td>
<td>I</td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>EPSRT</td>
<td>EPSRT</td>
<td>D</td>
<td>Number used in subroutine BETTER to generate two approximations from the one given</td>
</tr>
<tr>
<td>NOPOLY</td>
<td>NOPOLY</td>
<td>I</td>
<td>Number of the polynomial</td>
</tr>
<tr>
<td>Single Precision</td>
<td>Double Precision</td>
<td>Disposition of Argument</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Variable</td>
<td>Type</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
<td>D</td>
</tr>
<tr>
<td>EPSM</td>
<td>R</td>
<td>EPSM</td>
<td>D</td>
</tr>
<tr>
<td>KCHECK</td>
<td>I</td>
<td>KCHECK</td>
<td>I</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>NWORK</td>
<td>I</td>
<td>NWORK</td>
<td>I</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
</tr>
<tr>
<td>NALTER</td>
<td>I</td>
<td>NALTER</td>
<td>I</td>
</tr>
<tr>
<td>IAPP</td>
<td>I</td>
<td>IAPP</td>
<td>I</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</td>
<td>I</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA,VA</td>
<td>D</td>
</tr>
<tr>
<td>NA</td>
<td>I</td>
<td>NA</td>
<td>I</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>UX,VX</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB,VB</td>
<td>D</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>UPX,VPX</td>
<td>D</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
<td>I</td>
</tr>
</tbody>
</table>

Subroutine HORNERS
### TABLE F.II (Continued)

<table>
<thead>
<tr>
<th>Single Precision</th>
<th>Double Precision</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
<td>Variable Type</td>
<td>of Argument</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subroutine TEST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3,VX3</td>
<td>D</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4,VX4</td>
<td>D</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
<td>D</td>
</tr>
<tr>
<td>DENOM</td>
<td>R</td>
<td>DENOM</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subroutine BETTER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA,VA</td>
<td>D</td>
</tr>
<tr>
<td>NP</td>
<td>I</td>
<td>NP</td>
<td>I</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT,VROOT</td>
<td>D</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
<td>I</td>
</tr>
<tr>
<td>BAPP</td>
<td>C</td>
<td>UBAPP,VBAPP</td>
<td>D</td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</td>
<td>I</td>
</tr>
<tr>
<td>ROOTS</td>
<td>C</td>
<td>UROOTS,VROOTS</td>
<td>D</td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>L</td>
<td>I</td>
</tr>
<tr>
<td>EPSRT</td>
<td>R</td>
<td>EPSRT</td>
<td>D</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB,VB</td>
<td>D</td>
</tr>
<tr>
<td>X1</td>
<td>C</td>
<td>UX1,VX1</td>
<td>D</td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>UX2,VX2</td>
<td>D</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3,VX3</td>
<td>D</td>
</tr>
<tr>
<td>PX1</td>
<td>C</td>
<td>UPX1,VPX1</td>
<td>D</td>
</tr>
<tr>
<td>PX2</td>
<td>C</td>
<td>UPX2,VPX2</td>
<td>D</td>
</tr>
<tr>
<td>PX3</td>
<td>C</td>
<td>UPX3,VPX3</td>
<td>D</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Double Precision Variable</td>
<td>Type</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>---------------------------</td>
<td>------</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>VX4, VX4</td>
<td>D</td>
</tr>
<tr>
<td>J</td>
<td>I</td>
<td>J</td>
<td>I</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
</tr>
</tbody>
</table>

**CONV** = true implies convergence has been obtained

Newest approximation to root

Program control - counts the number of roots used as initial approximations

Maximum number of iterations permitted

Unit number of output device

**Subroutine ALTER**

- **X1**
  - C
  - X1R, X1I
  - D
  - ECR
  - One of the three approximations to be altered

- **X2**
  - C
  - X2R, X2I
  - D
  - ECR
  - One of the three approximations to be altered

- **X3**
  - C
  - X3R, X3I
  - D
  - ECR
  - One of the three approximations to be altered

- **X2R**
  - R
  - X2R
  - D
  - ECR
  - Real part of complex approximation

- **X2I**
  - R
  - X2I
  - D
  - ECR
  - Imaginary part of complex approximation

**Subroutine CALC**

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

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

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

```plaintext
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 MULTICIPlicITIES DETERMINED.
C *
DOUBLE PRECISION URAPP, VRAPP
DIMENSION URAPP(25, 3), VRAPP(25, 3), UAPP(25), VAPP(25),
UROOT(25), VROOT(25), MULT(25), UDP(26), VDP(26), UZROOT(26), VZROOT(26),
ANAME(212), ENTRY(261)
DOUBLE PRECISION EPSRT
COMMON EPSRT, EPS1, EPS2, EPS3, EPS4, I2, MAX
DATA PNAME, QNAME, QQNAME / 2HP(, 2HQ(, 3HQQ) /
DATA ENTRY / 1HIH2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9, 2H10, 2H11, 2H12, 2H13,
1H14, 2H15, 2H16, 2H17, 2H18, 2H19, 2H20, 2H21, 2H22, 2H23, 2H24, 2H25, 2H26/
DATA ANAME(1), ANAME(2) / 4HMULL, 4HERS /
LOGICAL NEWTT
10 J=0
018 IF (XCHECK .EQ. 1) STOP
019 WRITE(101, 1000) NOPOLY, NP, NAPP, MAX, EPS1, EPS2, EPS3, EPS4, XSTART, XEND,
020 KCHECK
021 IF (XCHECK .EQ. 0) STOP
022 WRITE(102, 1020) ANAME(1), ANAME(2), NOPOLY
023 WRITE(102, 1020) NAPP
024 WRITE(102, 1020) MAX
025 WRITE(102, 1020) EPS1
026 WRITE(102, 1020) EPS2
027 WRITE(102, 1020) EPS3
028 WRITE(102, 1020) EPS4
029 WRITE(102, 1020) XSTART
030 WRITE(102, 1020) XEND
031 WRITE(102, 1020) KKK = NPP+1
032 NNN = KKK+1
033 DO 20 I=1, KKK
035 NAPP = NP
036 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
037 GO TO 22
038 22 READ(101, 1015) UPI, VP, UPI, VP
039 IF (NAPP .NE. 0) GO TO 22
040 NAPP = NP
041 NNN = KKK+1
```
TABLE F.III (Continued)

      DO 25 I=1,KKK
      JJJ=NNN-1
      WRITE(102,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
      IF(NP.GE.3) GO TO 30
      J=1
      CALL QUAD(NP,UP,VP,J,UROOT,VROOT,MULT)
      WRITE(112,1070)
      WRITE(102,1165) (1,UROOT(I),VROOT(I),MULT(I),I=1,J)
      GO TO 10
      25  CALL QERU(NP,UP,VP,HDP,UDP,VDP)
      CALL CCDINP,UP,VP,HDP,UDP,VDP,NO,UD,VD)
      IFNO.GT.1) GO TO 70
      IF(NP.EQ.0) GO TO 65
      UDUMMY=UD1*UD2+VD2*VD2
      UZRO=UD1*VD2+VD1*UD2/UDUMMY
      VZRO=-UD2*VD1+VD2*UD1/UDUMMY
      KKK=NP+1
      DO 65 I=1,KKK
      UQQ(I)=UP(I)
      VQQ(I)=VP(I)
      NQQ=NP
      GO TO 110
      65  KKK=NP+1
      DO 66 I=1,KKK
      UQ(I)=UP(I+1)
      VQ(I)=VP(I+1)
      NQ=NP
      GO TO 80
      66  KKK=NP+1
      DO 66 I=1,KKK
      UQ(I)=UP(I)
      VQ(I)=VP(I)
      NQ=NP
      GO TO 80
      70  CALL DIVIDE(NP,UP,VP,NO,UD,VD,NO,UD,VD)
      WRITE(102,1120) NO
      KKK=NO+1
      NNN=KKK+1
      DO 83 I=1,NNN
      JJJ=NNN-1
      WRITE(102,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
      IF(NP.GE.3) GO TO 85
      83  WRITE(102,1165) (1,UROOT(I),VROOT(I),MULT(I),I=1,J)
      GO TO 170
      85  KKK=NO+1
      DO 86 I=1,NNN
      UQQ(I)=UQ(I)
      VQQ(I)=VQ(I)
      NQ=NO
      GO TO 90
      86  KKK=NO+1
      DO 86 I=1,NNN
      UQQ(I)=UQ(I)
      VQQ(I)=VQ(I)
      NQ=NO
      GO TO 90
      90  CALL QUAD(NQ,UQ,VQ,J,UROOT,VROOT,MULT)
      WRITE(102,1100) QNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
      IF(NP.GE.3) GO TO 85
      100  KKK=NO+1
      DO 100 I=1,NNN
      UQQ(I)=UQ(I)
      VQQ(I)=VQ(I)
      NQ=NO
      GO TO 120
      100  CALL QERU(NQ,UQ,VQ,HDP,UDP,VDP)
      CALL CCDINP,UP,VP,HDP,UDP,VDP,NO,UD,VD)
      IF(NP.GE.3) GO TO 110
      IF(NP.GE.3) GO TO 110
      WRITE(102,1070)
      WRITE(102,1165) (1,UROOT(I),VROOT(I),MULT(I),I=1,J)
      }
### TABLE F.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0099</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0100</td>
<td>WRITE(102,1180)</td>
</tr>
<tr>
<td>0101</td>
<td>D0 350 L=1,JAP</td>
</tr>
<tr>
<td>0102</td>
<td>WRITE(102,1190) L,UROOT(L),VRoot(L),MULTIL,UAPP(L),VRAPP(L)</td>
</tr>
<tr>
<td>0103</td>
<td>KKK=JAP+1</td>
</tr>
<tr>
<td>0104</td>
<td>T=JAP,L, J WRITE(102,1165) L,UROOT(L),VRoot(L),MULTIL,L=KKK,J</td>
</tr>
<tr>
<td>0105</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0106</td>
<td>1000 FORMAT(12,1X,9X,13,1X,4(D6.0,1X),13X,2(D7.0,1X),11)</td>
</tr>
<tr>
<td>0107</td>
<td>1010 FORMAT(2030.0)</td>
</tr>
<tr>
<td>0108</td>
<td>1015 FORMAT(2D30.0)</td>
</tr>
<tr>
<td>0109</td>
<td>1020 FORMAT(1H,1OX,4H GREATEST COMMON DIVISOR METHOD USED WITH .2(A4),</td>
</tr>
<tr>
<td>0110</td>
<td>1030 FORMAT(1X,2H THE DEGREE OF P(X) IS</td>
</tr>
<tr>
<td>0111</td>
<td>1040 FORMAT(2X,2A2,A2,4H) = .023.16,3H + .023.16,2H 11</td>
</tr>
<tr>
<td>0112</td>
<td>1070 FORMAT:///1X,13H ROOTS OF P(X),52X,14HMULTIPICITIES//</td>
</tr>
<tr>
<td>0113</td>
<td>1080 FORMAT(2X,5HROOT(L),12,4H) = .023.16,3H + .023.16,2H 11,10X,121</td>
</tr>
<tr>
<td>0114</td>
<td>1100 FORMAT(2X,43,A2,4H) = .023.16,3H + .023.16,2H 11</td>
</tr>
<tr>
<td>0115</td>
<td>1120 FORMAT:///1X,73H(X) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE</td>
</tr>
<tr>
<td>0116</td>
<td>1165 FORMAT(2X,5HROOT(L),12,4H) = .023.16,3H + .023.16,2H 1,7X,12,10X,26H</td>
</tr>
<tr>
<td>0117</td>
<td>1180 FORMAT:///1X,13H ROOTS OF P(X),52X,14HMULTIPICITIES,17X,21H INITIAL</td>
</tr>
<tr>
<td>0118</td>
<td>1190 FORMAT(2X,5HROOT(L),12,4H) = .023.16,3H + .023.16,2H 1,7X,12,9X,023.</td>
</tr>
<tr>
<td>0119</td>
<td>1163H + .023.16,2H 11</td>
</tr>
<tr>
<td>0120</td>
<td>2000 FORMAT(1X,4H NUMBER OF INITIAL APPROXIMATIONS GIVEN.</td>
</tr>
<tr>
<td>0121</td>
<td>2010 FORMAT(1X,2H MAXIMUM NUMBER OF ITERATIONS.,11X,13)</td>
</tr>
<tr>
<td>0122</td>
<td>2020 FORMAT(1X,2H TEST FOR CONVERGENCE.,13X,09.2)</td>
</tr>
<tr>
<td>0123</td>
<td>2090 FORMAT(1X,2H TEST FOR MULTIPLICITIES.,10X,09.2)</td>
</tr>
<tr>
<td>0124</td>
<td>2050 FORMAT(1X,2H RADIUS TO START SEARCH.,11X,09.2)</td>
</tr>
<tr>
<td>0125</td>
<td>2000 FORMAT(1X,2H RADIUS TO END SEARCH.,13X,09.2)</td>
</tr>
<tr>
<td>0126</td>
<td>2070 FORMAT(1X,3H TEST FOR ZERO IN SUBROUTINE GCD.,10X,09.2)</td>
</tr>
<tr>
<td>0127</td>
<td>2080 FORMAT(1X,3H TEST FOR ZERO IN SUBROUTINE QUAD.,10X,09.2)</td>
</tr>
<tr>
<td>0128</td>
<td>END</td>
</tr>
</tbody>
</table>
### TABLE F. III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE MULTI(N, UP, VP, J, UROOT, VR0OT, MULT)</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION UP, VP, UROOT, VR0OT, UA, UB, VB, VC, EPS1, EPS2, EPS3, EPS4</td>
</tr>
<tr>
<td>0003</td>
<td>DIMENSION UP(26), VP(26), UROOT(25), VR0OT(25), UA(26), VB(26), VC(26), MULT(25)</td>
</tr>
<tr>
<td>0004</td>
<td>DOUBLE PRECISION EPSRT, EPS1, EPS2, EPS3, EPS4, EPSLON, IO2, MAX</td>
</tr>
<tr>
<td>0005</td>
<td>COMMON EPSRT, EPS1, EPS2, EPS3, EPS4, EPSLON, IO2, MAX</td>
</tr>
<tr>
<td>0006</td>
<td>DO I = 1, J</td>
</tr>
<tr>
<td>0007</td>
<td>K = N + I</td>
</tr>
<tr>
<td>0008</td>
<td>DO K = 1, KKK</td>
</tr>
<tr>
<td>0009</td>
<td>UA(K) = UP(KKK + I - K)</td>
</tr>
<tr>
<td>0010</td>
<td>VA(K) = VP(KKK + I - K)</td>
</tr>
<tr>
<td>0011</td>
<td>MULT(I) = 0</td>
</tr>
<tr>
<td>0012</td>
<td>CALL HORNER(m, UA, VB, UROOT(11), VR0OT(11), UB, VB, VC)</td>
</tr>
<tr>
<td>0013</td>
<td>IF(MULT(I) .EQ. 0) GO TO 40</td>
</tr>
<tr>
<td>0014</td>
<td>EPSRT = SQRT(UA<em>UA + VA</em>VA)</td>
</tr>
<tr>
<td>0015</td>
<td>IF(EPSRT .GT. EPSLON) GO TO 50</td>
</tr>
<tr>
<td>0016</td>
<td>IF(EPSRT .LT. EPSLON) GO TO 50</td>
</tr>
<tr>
<td>0017</td>
<td>IF(MULT(I) .GT. 1) GO TO 60</td>
</tr>
<tr>
<td>0018</td>
<td>WRITE(102, IO2) EPSLON, I, UROOT(I), VR0OT(I)</td>
</tr>
<tr>
<td>0019</td>
<td>M = N</td>
</tr>
<tr>
<td>0020</td>
<td>MULT(I) = 1</td>
</tr>
<tr>
<td>0021</td>
<td>IF(MULT(I) .GT. 1) GO TO 40</td>
</tr>
<tr>
<td>0022</td>
<td>IF(MULT(I) .LT. 1) GO TO 40</td>
</tr>
<tr>
<td>0023</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0024</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0025</td>
<td>100 FORMAT(//15H THE EPSILON (+010, 3, 4H CHECK IN SUBROUTINE MULTI INDICATES THAT ROOT(I,12,4H) = .0231.6, 34 + .0231.6, 2H I, 180H IS NO 27 CLOSE ENOUGH TO BE A TRUE ROOT, IT IS PRINTED BELOW WITH MULTIPlicity 0//)</td>
</tr>
<tr>
<td>0031</td>
<td>END</td>
</tr>
</tbody>
</table>
TABLE F.III (Continued)

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

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

K=N+M

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

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

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

IF(K.EQ.0) GO TO 100

J=J+1

DO 50 I=1,K

UTERM=UP(N-I)

VTERM=VP(N-I)

KK=K/I

KNN=M-J

DO 40 MI=KNN, M

IF(KK.GT.1) GO TO 10

GO TO 45

10 IF(M).GE.1 GO TO 20

GO TO 40

20 UTERM=UTERM+(UQ(KK)+UD(M+1)-VQ(KK)+VD(M+1))

VTERM=VTERM+(UQ(KK)+VD(M+1)-UQ(KK)+UD(M+1))

40 IF(KK>1) GO TO 30

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

VQ(K+1)=VQ(K+1)-UTERM*UD(M+1)+VTERM*VD(M+1)/UDUMMY

50 VQ(K+1)=VQ(K+1)-UQ(KK)+UTERM*VD(M+1)

100 RETURN

END

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

DOUBLE PRECISION UP,VP,UA,VA,AAA

KK=M

DO 10 I=2, KK

AAA=AA*(I-1)

10 VA(I-1)=AAA*UP(I)

RETURN

END
TABLE F. III (Continued)

SUBROUTINE GCDIN, UR, VR, M, US, VS, ML, USS, VSS

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

DOUBLE PRECISION EPSRT
0003 DOUBLE PRECISION US5SSS, VS5SSS
0004 DOUBLE PRECISION UR, VR, M, US, VS, USS, VSS, URR, UOT, UT, VT, EPSLON, EP
0005 DIMENSION UR(26), VR(26), M(26), US(26), VS(26), USS(26), VSS(26), URR(26), VRR(26)

COMMON EPSRT, EPSLON, EPS2, EPS3, EPS4, IO2, MAX
0007 NJ = N
0008 M1 = M
0009 KKK = N + 1
0010 DO 20 L = 1, KKK
0011 URR(1) = UR(1)
0012 20 VAR(1) = VR(1)
0013 KKK = M + 1
0014 DO 25 L = 1, KKK
0015 US11 = US(1)
0016 25 VSS(1) = VSS(1)
0017 30 BBB = USS(ML1) + USS(ML1) + VSS(ML1) + VSS(ML1)
0018 UD = (USS(N1) + USS(N1) + VSS(N1) + VSS(N1))/BBB
0019 VD = (USS(N1) + VSS(N1) - URR(N1) - URR(N1))/BBB
0020 KKK = N1 - 1
0021 DO 40 L = KKK, N1
0022 UT(1) = URR(1) - (USS(N1 - N1 + M1) + VSS(N1 - N1 + M1))
0023 40 UT(1) = UT(1) - VSS(1 - N1 + M1) + VSS(1 - N1 + M1)
0024 IF (M1 .LE. NL) GO TO 70
0025 KKK = N1 - N1
0026 DO 60 L = 1, KKK
0027 UT(1) = URR(1)
0028 60 VT(1) = VR(1)
0029 70 DO 40 = 1, N1
0030 BBB = OSGK (UT(N1) + 1 - 1 + UT(N1) + 1 - VT(N1) + 1 - VT(N1) + 1 - 1)
0031 IF (BBB .GT. EPSLON) GO TO 100
0032 90 CONTINUE
0033 DO 95 L = 95, 1, M1
0034 BBB = US5SSS(M1) + USS(M1) + VSS(M1) + VSS(M1)
0035 US5SSS = USS(1) + USS(1) + VSS(1) + VSS(1) + BBB
0036 VSS5SS = VSS(1) + VSS(M1) + VSS(M1) + BBB
0037 USS5SS = USS(1) + USS(1) + BBB
0038 95 VSS(1) = VSS5SS
0039 US5SSS = USS(1) + 1; 0
0040 VSS5SS = 0; 0
0041 GO TO 200
0042 100 KNN = 1
0043 IF (KNN .EQ. 0) GO TO 170
0044 IF (KNN .EQ. 1) GO TO 140
0045 KKK = K + 1
0046 DO 130 = 1, KKK
0047 URR(1) = UT(1)
0048 130 VR(1) = VT(1)
0049 NL = K
TABLE F.III (Continued)

```
0050  GO TO 30
0051  140  KKK=K+1
0052  DO 150  J=1,KKK
0053    URR(J)=USSI(J)
0054    VRR(J)=VSS(J)
0055  150  USSI(J)=UT(J)
0056  150  VSS(J)=VT(J)
0057  KKK=K+2
0058  NNN=M1+1
0059  DO 160  J=KKK,NNN
0060    URR(J)=USSI(J)
0061    VRR(J)=VSS(J)
0062    M1=M1
0063    M1=K
0064  GO TO 30
0065  170  USSI(1)=1.0
0066  VSS(1)=0.0
0067    M1=0
0068  200  RETURN
0069  END
```
TABLE F.111 (Continued)

**SUBROUTINE QUAD4**, **UA**, **VA**, **J**, **URDOTT**, **VROOT**, **MULT1**

```
0001 C ******************************************************************************
0002 C ******** SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPLEITIES *
0003 C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
0004 C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
0005 C ******************************************************************************
0006 C
0007 DOUBLPRECISION EPSRT
0008 DOUBLPRECISION UA, VA, VRDTT, VROOT, VDISC, VTEMP, UTEMP, UD, VO, E
0010 DIMENSION UA(26), VA(26), UROOT(251), VRDTT(25), MULT(25)
0012 COMMON EPSRT, EPS1, EPS2, EPS3, EPS4, EPS5, EPSLN, BBA
0014 IF(INOT.1) GO TO 60
0016 IF(JJLT.0) GO TO 60
0018 J=J+1
0020 GO TO 50
0022 MULT1=1
0024 50 BBA=UA(2)+UA(2)+VA(2)+VA(2)
0026 VRDOTTJ=(UA(1)+UA(2)+VA(2)+VA(1))/BBA
0028 VRDOTTJ=(UA(1)+UA(2)+VA(2))/BBA
0030 GO TO 200
0032 60 VRDISC=(UA(2)+UA(2)+VA(2)+VA(2)+(UA(2)+UA(2)+VA(2)+VA(1))/BBA)
0034 VRDISC=(UA(2)+UA(2)+VA(2))/BBA
0036 VRDISC=(UA(2)+UA(2)+VA(2)+VA(1)+VA(1)+VA(1))/BBA
0038 VRDISC=(UA(2)+UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0040 VRDISC=(UA(2)+UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0042 VRDISC=(UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0044 VRDISC=(UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0046 CALL COMPDIS(VDISC, VDISC, UTEMP, VTEMP)
0048 UD=2.0*UA(1)
0049 VD=2.0*UA(1)
0050 BBA=UD*UD+VD*VD
0052 VRDOTTJ=VRDOTTJ+(-UA(2)+UTEMP)*UD+(-UA(2)+VTEMP)*VD/BBA
0054 VRDOTTJ=VRDOTTJ+(-UA(2)+UTEMP)*UD+(-UA(2)+VTEMP)*VD/BBA
0056 VRDOTTJ=VRDOTTJ+(-UA(2)+UTEMP)*UD+(-UA(2)+VTEMP)*VD/BBA
0058 VRDOTTJ=VRDOTTJ+(-UA(2)+UTEMP)*UD+(-UA(2)+VTEMP)*VD/BBA
0060 J=J+2
0062 GO TO 200
0064 100 IF(JJLT.0) GO TO 110
0066 J=J+1
0068 GO TO 130
0070 110 MULT1=2
0072 J=J+1
0074 UD=2.0*UA(1)
0076 VD=2.0*UA(1)
0078 BBA=UD*UD+VD*VD
0080 VRDOTTJ=VRDOTTJ+(-UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0082 VRDOTTJ=VRDOTTJ+(-UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0084 VRDOTTJ=VRDOTTJ+(-UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0086 VRDOTTJ=VRDOTTJ+(-UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0088 VRDOTTJ=VRDOTTJ+(-UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0090 VRDOTTJ=VRDOTTJ+(-UA(2)+UA(2)+UA(2)+UA(2)+UA(2))/BBA
0092 200 RETURN
0094 END
```
**TABLE F. III (Continued)**

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Code/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>&quot;SUBROUTINE-MULLER(UA,VA,UP,VP,NAPP,UAPP,VAPP,NAPP,XSTART,XEND,UROOT,VROOT, INROOT,IROOT,UAPP,VRAPP,NOPOLY)&quot;</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,UX1,UAPP,VAPP</td>
</tr>
<tr>
<td></td>
<td>1,UX2,UX2,UWORK,VWORK,UX3,UX3,V3,UB,VB,UX4,UX4,UA,VA,UPX1,VPX1,UAPP,VAPP,V</td>
</tr>
<tr>
<td></td>
<td>ZRAPP,UPX4,VPX4,EPST,EPSD,EPSON,UH3,VPQ4,ABPX4,ABPX3</td>
</tr>
<tr>
<td></td>
<td>3,QQQ,XSTART,XEND</td>
</tr>
<tr>
<td>0003</td>
<td>DIMENSION UROOT(25),VROOT(25),MULTI(25),UAPP(25,3),VAPP(25,3),UWORK</td>
</tr>
<tr>
<td></td>
<td>1(25),VWORK(25),UB(25),VB(25),UA(25),VA(25),URAPP(25,3),VRAPP(25,3)</td>
</tr>
<tr>
<td>0004</td>
<td>LOGICAL CONV</td>
</tr>
<tr>
<td>0005</td>
<td>DOUBLE PRECISION EPS1</td>
</tr>
<tr>
<td>0006</td>
<td>COMMON EPSRT,EPS1,EPSO,EPSM,Z02,AX</td>
</tr>
<tr>
<td>0007</td>
<td>DATA PNAME,DNAME/2HP(,2HD(/</td>
</tr>
<tr>
<td>0008</td>
<td>EPSRT=0.99</td>
</tr>
<tr>
<td>0009</td>
<td>NROOT=0</td>
</tr>
<tr>
<td>0010</td>
<td>IROOT=0</td>
</tr>
<tr>
<td>0011</td>
<td>IPATH=1</td>
</tr>
<tr>
<td>0012</td>
<td>NWORK=0</td>
</tr>
<tr>
<td>0013</td>
<td>NALTER=0</td>
</tr>
<tr>
<td>0014</td>
<td>ITIME=0</td>
</tr>
<tr>
<td>0015</td>
<td>IAPP=1</td>
</tr>
<tr>
<td>0016</td>
<td>ITER=1</td>
</tr>
<tr>
<td>0017</td>
<td>IF(INAPP.NE.0) GO TO 18</td>
</tr>
<tr>
<td>0018</td>
<td>NAPP=NP</td>
</tr>
<tr>
<td>0019</td>
<td>CALL GENAPP(UAPP,VAPP,NAPP,XSTART)</td>
</tr>
<tr>
<td>0020</td>
<td>GO TO 27</td>
</tr>
<tr>
<td>0021</td>
<td>18 DO 25 I=1,NAPP</td>
</tr>
<tr>
<td>0022</td>
<td>UAPP(I,1)=0.9*UAPP(I,2)</td>
</tr>
<tr>
<td>0023</td>
<td>VAPP(I,2)=0.9*VAPP(I,2)</td>
</tr>
<tr>
<td>0024</td>
<td>UAPP(I,3)=1.1*UAPP(I,2)</td>
</tr>
<tr>
<td>0025</td>
<td>VAPP(I,3)=1.1*VAPP(I,2)</td>
</tr>
<tr>
<td>0026</td>
<td>25 DO 27 K=NP+1</td>
</tr>
<tr>
<td>0027</td>
<td>DO 30 I=1,KKK</td>
</tr>
<tr>
<td>0028</td>
<td>UWORK(I)=UA(I)</td>
</tr>
<tr>
<td>0029</td>
<td>30 DO 30 I=VA(1)</td>
</tr>
<tr>
<td>0030</td>
<td>NWORK=NP</td>
</tr>
<tr>
<td>0031</td>
<td>40 UX=UAPP(IAPP,1)</td>
</tr>
<tr>
<td>0032</td>
<td>VX=VAPP(IAPP,1)</td>
</tr>
<tr>
<td>0033</td>
<td>UX2=UAPP(IAPP,2)</td>
</tr>
<tr>
<td>0034</td>
<td>VX2=VAPP(IAPP,2)</td>
</tr>
<tr>
<td>0035</td>
<td>UX3=UAPP(IAPP,3)</td>
</tr>
<tr>
<td>0036</td>
<td>VX3=VAPP(IAPP,3)</td>
</tr>
<tr>
<td>0037</td>
<td>CALL HORNER(NWORK,UXWORK,UXUX1,UX1,UX1,UB,VB,UX1,VPX1)</td>
</tr>
<tr>
<td>0038</td>
<td>CALL HORNER(NWORK,UXWORK,UXUX2,UX2,UX2,UB,VB,UX2,VPX2)</td>
</tr>
<tr>
<td>0039</td>
<td>CALL HORNER(NWORK,UXWORK,UXUX3,UX3,UX3,UB,VB,UX3,VPX3)</td>
</tr>
<tr>
<td>0040</td>
<td>50 CALL CALCX(UX1,UX1,UX2,UX2,UX3,UX3,UX3,UX3,UX1,UX1,UX1,UX1,VPX1,VPX1)</td>
</tr>
<tr>
<td>0041</td>
<td>50 CALL CALCX(UX2,UX2,UX2,UX2,UX2,UX2,UX2,UX2,UX2,UX2,VPX2,VPX2)</td>
</tr>
<tr>
<td>0042</td>
<td>50 CALL CALCX(UX3,UX3,UX3,UX3,UX3,UX3,UX3,UX3,UX3,UX3,VPX3,VPX3)</td>
</tr>
<tr>
<td>0043</td>
<td>50 CALL CALCX(UX4,UX4,UX4,UX4,UX4,UX4,UX4,UX4,UX4,UX4,VPX4,VPX4)</td>
</tr>
<tr>
<td>0044</td>
<td>50 CALL CALCX(UPX3,UPX3,UPX3,UPX3,UPX3,UPX3,UPX3,UPX3,UPX3,UPX3,VPX3)</td>
</tr>
</tbody>
</table>

**MULLER'S METHOD**

- Extracts the zeros and their multiplicities of a polynomial of maximum degree 25 through three given points.
- The polynomial is approximated by a quadratic. The zero of the quadratic closest to the old approximation is taken as the new approximation.
- In this manner a sequence is obtained converging to a zero.
TABLE F.III (Continued)

```plaintext
0044 IF (ABPX3.EQ.0.0) GO TO 70
0045 QQ=ABPX4*ABPX3
0046 IF (QQ.LE.10.) GO TO 70
0047 UQ4=0.5*UQ4
0048 VQ4=0.5*VQ4
0049 VX4=VX3+(UH3*UQ4-VH3*VQ4)
0050 VX4=VX3+(VH3*UQ4+UH3*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),VAPP(IAPP),UAPP(IAPP),VAPP(IAPP),UAPP(IAPP),VAPP(IAPP),IAPP,
0056 ITER=1
0057 ITER=1
0058 GO TO 40
0059 75 IF (IAPP.LT.NAPP) GO TO 100
0060 IF (XEND.EQ.0.0) GO TO 77
0061 IF (XSTART.GT.XENO) GO TO 77
0062 NAPP=NAPP+1
0063 CALL GENAPPI(UAPP,VAPP,NAPP,XEND)
0064 IAPP=0
0065 GO TO 100
0066 77 WRITE(102,1035) (DNAME,J),UWORK(J),VWORK(J),J=1,KKK)
0067 WRITE(102,1035) (DNAME,J),UWORK(J),VWORK(J),J=1,KKK)
0068 WRITE(102,1035) (DNAME,J),UWORK(J),VWORK(J),J=1,KKK)
0069 80 IF (NROOT.EQ.0) GO TO 90
0070 IF (IPATH.EQ.1) GO TO 82
0071 IF (IPATH.EQ.1) GO TO 82
0072 CALL BETTER(UA,VAPP,VRAPP,NROOT,UROOT,VROOT,NROOT) RETURN
0073 RETURN
0074 IF (NROOT.EQ.0) GO TO 90
0075 IF (IROOT.EQ.0) GO TO 95
0076 WRITE(102,1080)
0077 DO 55 I=1,IROOT
0078 WRITE(102,1080)
0079 DO 55 I=1,IROOT
0080 WRITE(102,1080)
0081 WRITE(102,1080)
0082 WRITE(102,1080)
0083 WRITE(102,1080)
0084 RETURN
0085 RETURN
0086 RETURN
0087 100 IAPP=1
0088 100 IAPP=1
0089 IAPP=1
0090 IAPP=1
0091 120 NROOT=NROOT+1
0092 IROOT=IROOT+1
0093 MULT=NMULT+1
0094 NDUMULT=NDUMULT+1
0095 UROOT(NROOT)=UX4
0096 VROOT(NROOT)=VX4
0097 UAPP(NROOT,I)=UAPP(IAPP)
0098 VRAPP(NROOT,I)=VAPP(IAPP)
0099 VRAPP(NROOT,I)=VAPP(IAPP)
0100 VRAPP(NROOT,I)=VAPP(IAPP)
```

TABLE F.III (Continued)

0101 URAPP(NROOT,31)=URAPP(IAPP,31)
0102 VRAPP(NROOT,31)=VRAPP(IAPP,31)
0103 IF(NMULT.LT.NPI) GO TO 130
0104 DO TO 80
0105 130 CALL HORNER(NWORK,WORK,WORK,UX4,VX4,UB,V8,UPX4,VPX4)
0106 NWORK=NWORK-1
0107 KKK=NWORK+1
0108 DO 140 I=1,KKK
0109 WORK(I)=UB(I)
0110 140 CALL HORNER(NWORK,WORK,WORK,UX4,VX4,UB,V8,UPX4,VPX4)
0111 CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
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)=UB(I)
0118 VB(I)=V8(I)
0119 145 CALL QUAD(INWORK,UB,V8,NROOT,UROOT,VROOT,MULT)
0120 GO TO 80
0121 150 MULT(NROOT)=MULT(NROOT)+1
0122 NDULT=NDULT+1
0123 GO TO 125
0124 110 UX1=UX2
0125 VX1=VX2
0126 UX2=UX3
0127 VX2=VX3
0128 UX3=UX4
0129 VX3=VX4
0130 UPX1=UPX2
0131 VPX1=VPX2
0132 UPX2=UPX3
0133 VPX2=VPX3
0134 UPX3=UPX4
0135 VPX3=VPX4
0136 ITER=ITER+1
0137 GO TO 50
0138 1090 FORMAT(1X,65COEFFICIENTS OF DEFLECTED POLYNOMIAL FOR WHICH NO
1090 ZEROS WERE FOUND//)
0140 1080 FORMAT(1X,3HINITIAL APPROXIMATION//)
0141 1070 FORMAT(4HNO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,12)
0142 1060 FORMAT(2X,5HROOT(S),12,4H = .023.16,3H + .023.16,2H 1.19X,2HSOLVED
0143 1 BY DIRECT
0144 1035 FORMAT(3X,9A2,12,4H = .023.16,3H + .023.16,2H 1)
0145 1050 FORMAT(12X,9A2,12,4H = .023.16,3H + .023.16,2H 1/82X,023.16,3H + .023.16,2H 1)
0146 1049 FORMAT(2X,5HROOT(S),12,4H = .023.16,3H + .023.16,2H 1.18X,023.16,3H
0147 1 + .023.16,2H 1)
0146 END
SUBROUTINE BETTER(UA, VA, NP, UROOT, VROOT, NROOT, UAPP, VRAPP, IROOT, MUL IT)

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

* 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.

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

DOUBLE PRECISION UROOT, VROOT, UA, VA, UAPP, VRAPP, UX1, VX1, UX2, VX2, UX3
  1, VX3, UPX1, UAPP, UBAPP, VBAPP, UX2, VX2, VPX2, VPX3, VPX3, UB, VB, UROOTS, VROOTS, EPSRT, UK
  2, UBAPP, VRAPP, EPSD, EPS, UQ4, VX4, UH4, VX4, UH4, VX4, UH4

LOGICAL CONV

DIMENSION UROOT(35), VROOT(35), UA(35), VA(35), UBAPP(35), VBAPP(35, 3)

DOUBLE PRECISION EPS1, EPSH

COMMON EPSRT, EPSL, EPS, EPSD, EPSM, IO2, MAX

IF (NROOT.LE.1) RETURN

L = 0

DO 10 I = 1, NROOT

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

VBAPP(I, 1) = VROOT(I) * EPSRT

UBAPP(I, 2) = UROOT(I)

VBAPP(I, 2) = VROOT(I)

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

VBAPP(I, 3) = VROOT(I) * (2.0 - EPSRT)

10    VBAPP(I, 1) = UROOT(I) * (2.0 - EPSRT)

DO 100 J = 1, NROOT

UX1 = UAPP(J, 1)

VX1 = VBAPP(J, 1)

UX2 = UAPP(J, 2)

VX2 = VBAPP(J, 2)

UX3 = UAPP(J, 3)

VX3 = VBAPP(J, 3)

ITER = 1

CALL HORNER(NP, UA, VA, UX1, VX1, UB, VB, UPX1, VPX1)

CALL HORNER(NP, UA, VA, UX2, VX2, UB, VB, UPX2, VPX2)

20 CALL HORNER(NP, UA, VA, UX3, VX3, UB, VB, UPX3, VPX3)

CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UX
  34, VX4, 0.0, VX4, 0.0)

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

IF (CONV) GO TO 50

IF (ITER.LT.MAX) GO TO 40

WRITE(IO2, 1000) UX4, VX4

IF (J.LT.IROOT) GO TO 33

33 K = IROOT - L

IF (J.LE.1) GO TO 35

WRITE(IO2, 100) UX4, VX4

IF (J.LT.IROOT) GO TO 33

34 V = VRAPP(K, 1) + VRAPP(K, 2) + VRAPP(K, 3)

35 IROOT = IROOT - 1

GO TO 100
TABLE F.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0046</td>
<td>40 UX1=UX2</td>
</tr>
<tr>
<td>0047</td>
<td>VX1=VX2</td>
</tr>
<tr>
<td>0048</td>
<td>UX2=UX3</td>
</tr>
<tr>
<td>0049</td>
<td>VX2=VX3</td>
</tr>
<tr>
<td>0050</td>
<td>UX3=UX4</td>
</tr>
<tr>
<td>0051</td>
<td>VX3=VX4</td>
</tr>
<tr>
<td>0052</td>
<td>UX2=UX3</td>
</tr>
<tr>
<td>0053</td>
<td>VX2=VX3</td>
</tr>
<tr>
<td>0054</td>
<td>UX3=UX4</td>
</tr>
<tr>
<td>0055</td>
<td>VX3=VX4</td>
</tr>
<tr>
<td>0056</td>
<td>ITER=ITER+1</td>
</tr>
<tr>
<td>0057</td>
<td>GO TO 20</td>
</tr>
<tr>
<td>0058</td>
<td>50 L=L+1</td>
</tr>
<tr>
<td>0059</td>
<td>UROOTS(L)=UX4</td>
</tr>
<tr>
<td>0060</td>
<td>VROOTS(L)=VX4</td>
</tr>
<tr>
<td>0061</td>
<td>100 CONTINUE</td>
</tr>
<tr>
<td>0062</td>
<td>IF(L.EQ.0) GO TO 120</td>
</tr>
<tr>
<td>0063</td>
<td>DO 110 I=1,L</td>
</tr>
<tr>
<td>0064</td>
<td>UROOTS(I)=UROOTS(I)</td>
</tr>
<tr>
<td>0065</td>
<td>VROOTS(I)=VROOTS(I)</td>
</tr>
<tr>
<td>0066</td>
<td>RETURN</td>
</tr>
<tr>
<td>0067</td>
<td>RETURN</td>
</tr>
<tr>
<td>0068</td>
<td>120 NROOT=0</td>
</tr>
<tr>
<td>0069</td>
<td>RETURN</td>
</tr>
<tr>
<td>0070</td>
<td>1000 FORMAT/42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(I,12,4H) = .</td>
</tr>
<tr>
<td></td>
<td>L023.16,2H + .023.16,2H I/24H DID NOT CONVERGE AFTER 13,11H ITERAT</td>
</tr>
<tr>
<td></td>
<td>ZIONS</td>
</tr>
<tr>
<td>0071</td>
<td>1010 FORMAT13H THE PRESENT APPROXIMATION IS .023.16,2H + .023.16,2H 1/</td>
</tr>
<tr>
<td></td>
<td>1/1</td>
</tr>
<tr>
<td>0072</td>
<td>END</td>
</tr>
</tbody>
</table>
TABLE F.III (Continued)

0001 SUBROUTINE ALTER(X1R, X2R, X3R, X1I, X2I, 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, EPS4, EPS5, EPS6, MAX
0004 IF (ITIME .NE. 0) GO TO 5
0005 ITIME = 1
0006 IF (INALTER .EQ. 0) GO TO 10
0007 WRITE (IO2, 1000) X1R, X1I, X2R, X2I, X3R, X3I
0008 NALTER = NALTER + 1
0009 IF (INALTER .GT. 5) RETURN
0010 GO TO 130

0011 10 R = DSQRT (X2R*X2R + X2I*X2I)
0012 BETA = DATAN2 (X2I, X2R)
0013 WRITE (IO2, 1020) X1R, X1I, X2R, X2I, X3R, X3I
0014 20 NALTER = NALTER + 1
0015 IF (INALTER .GT. 5) RETURN
0016 GO TO 130

0017 30 X2R = -X2R
0018 X2I = -X2I
0019 GO TO 50
0020 40 BETA = BETA + 1.0471976.
0021 XR = R*DCOS (BETA)
0022 XI = R*DSIN (BETA)
0023 50 X1R = XR*0.9*X2R
0024 X1I = 0.4*X2I
0025 X3R = 1.1*XR
0026 X3I = 1.1*X2I
0027 RETURN

0028 1000 FORMAT (1X, 5H1X = , D23.16, 3H + , D23.16, 2H I, 10X, 22HALTERED APPROXIM
0029 IATIONS/1X, 5H1X = , D23.16, 3H + , D23.16, 2H I/1X, 5H1X = , D23.16, 3H +
0030 Z , D23.16, 2H I/I)

0031 1020 FORMAT (1X, 5H1X = , D23.16, 3H + , D23.16, 2H I, 10X, 22INITIAL APPROXI
0032 MATIONS/1X, 5H1X = , D23.16, 3H + , D23.16, 2H I/I, 5H1X = , D23.16, 3H +
0033 Z , D23.16, 2H I/I)

0034 1010 FORMAT (1X, 5H1X NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
0035 TER , 13, 12H ITERATIONS./I)

0036 END
SUBROUTINE GENAPP(APPR, APP1, NAPP, XSTART)

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

DOUBLE PRECISION APPR, APP1, XSTART, EPS1, EPS2, EPS3, EPS4, EPSRT, EPS1, EPS2, EPS3, EPS4
DIMENSION APPRI2(25.3), APP1(125.3), MAX

IF(XSTART.EQ.0.0) XSTART=0.5

BETA=0.2617994
DO 10 I=1, NAPP
APPRI2(I,2)=XSTART*DCOS(BETA)
APP1(I,2)=XSTART*DSIN(BETA)
BETA=BETA+0.5235988
10 XSTART=XSTART+0.5
DO 20 I=1, NAPP
APPRI2(I,1)=0.9*APPRI2(I,2)
APP1(I,1)=0.9*APP1(I,2)
APPRI2(I,3)=1.1*APPRI2(I,2)
APP1(I,3)=1.1*APP1(I,2)
20 RETURN
END
TABLE F.III (Continued)

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

C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
C**************************************************************************9*
<table>
<thead>
<tr>
<th>TABLE F.III (Continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
</tr>
<tr>
<td>0002</td>
</tr>
<tr>
<td>0003</td>
</tr>
<tr>
<td>0005</td>
</tr>
<tr>
<td>0006</td>
</tr>
<tr>
<td>0007</td>
</tr>
<tr>
<td>0008</td>
</tr>
<tr>
<td>0009</td>
</tr>
<tr>
<td>0010</td>
</tr>
<tr>
<td>0011</td>
</tr>
<tr>
<td>0012</td>
</tr>
<tr>
<td>0013</td>
</tr>
<tr>
<td>0014</td>
</tr>
<tr>
<td>0015</td>
</tr>
<tr>
<td>0016</td>
</tr>
<tr>
<td>0017</td>
</tr>
<tr>
<td>0018</td>
</tr>
<tr>
<td>0019</td>
</tr>
<tr>
<td>0020</td>
</tr>
<tr>
<td>0021</td>
</tr>
<tr>
<td>0022</td>
</tr>
<tr>
<td>0023</td>
</tr>
<tr>
<td>0024</td>
</tr>
<tr>
<td>0025</td>
</tr>
<tr>
<td>0026</td>
</tr>
<tr>
<td>0027</td>
</tr>
<tr>
<td>0028</td>
</tr>
<tr>
<td>0029</td>
</tr>
<tr>
<td>0030</td>
</tr>
<tr>
<td>0031</td>
</tr>
<tr>
<td>0032</td>
</tr>
<tr>
<td>0033</td>
</tr>
<tr>
<td>0034</td>
</tr>
<tr>
<td>0035</td>
</tr>
<tr>
<td>0036</td>
</tr>
<tr>
<td>0037</td>
</tr>
<tr>
<td>0038</td>
</tr>
<tr>
<td>0039</td>
</tr>
<tr>
<td>0040</td>
</tr>
<tr>
<td>0041</td>
</tr>
<tr>
<td>0042</td>
</tr>
<tr>
<td>0043</td>
</tr>
<tr>
<td>0044</td>
</tr>
</tbody>
</table>
TABLE F.III (Continued)

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

SUBROUTINE CONSQT(UX, VX, UY, VY)
C
C***********************************************************************
C
** THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER. **
C***********************************************************************
C
DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, AAA, BBB

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

IF(UX) 10, 20, 30
10  UY = AAA
11    GO TO 100
20  IF(UX) 40, 50, 60
21    UY = AAA
22    GO TO 100
23  40  DUMMY = ABS(UX)
24    UY = 0.0
25    GO TO 100
26  50  UY = 0.0
27    GO TO 100
28  60  VY = 0.0
29    GO TO 100
30  100 RETURN
END
APPENDIX G

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

1. Use of the Program

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

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

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

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

Main Program

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

  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)
  UDDL(N+1), VDDL(N+1)
  UDG(N+1), VG(N+1)
  UDG3(2N+1), VDG3(2N+1)
  UDDG(N+1), VDDG(N+1)
  UAP(N), VAP(N)
  UROOT(N), VROOT(N)
  NULT(N)
  ENTRY(N+1)

Subroutine PROD

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

Subroutine ZROS

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

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

See corresponding subroutine in Table E.I.

Subroutine QUAD

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

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

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

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

4. Description of Program Output

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

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

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

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

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

5. Informative Messages and Error Messages

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

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

"QUAD FOUND XXX TO BE A MULTIPLE ROOT." XXX represents the value of the root found as a multiple root by Subroutine QUAD.
### Figure G.1 Control Card for Repeated G.C.D. - Newton's Method

<table>
<thead>
<tr>
<th>POLY</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>0.1D-03</td>
<td>1.D-10</td>
<td>1.D-20</td>
<td>1.0D+01</td>
<td>2.0D+01</td>
</tr>
</tbody>
</table>
TABLE G.II

REPEATED GCD - NEWTON'S METHOD

<table>
<thead>
<tr>
<th>Single Precision 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></td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td></td>
<td>Number of roots found</td>
</tr>
<tr>
<td>Jl</td>
<td>I</td>
<td>Jl</td>
<td>I</td>
<td></td>
<td>Multiplicity of given root</td>
</tr>
<tr>
<td>DO</td>
<td>C</td>
<td>UD0,VD0</td>
<td>D</td>
<td></td>
<td>Array of coefficients of original polynomial</td>
</tr>
<tr>
<td>ND0</td>
<td>I</td>
<td>ND0</td>
<td>I</td>
<td></td>
<td>Degree of original polynomial</td>
</tr>
<tr>
<td>DDO</td>
<td>C</td>
<td>UDDO,VDDO</td>
<td>D</td>
<td></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></td>
<td>Degree of DDO(X)</td>
</tr>
<tr>
<td>D1</td>
<td>C</td>
<td>UD1,VD1</td>
<td>D</td>
<td></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></td>
<td>Degree of D1(X)</td>
</tr>
<tr>
<td>DD1</td>
<td>C</td>
<td>UD1D,VD1D</td>
<td>D</td>
<td></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></td>
<td>Degree of DD1(X)</td>
</tr>
<tr>
<td>D2</td>
<td>C</td>
<td>UD2,VD2</td>
<td>D</td>
<td></td>
<td>Array of coefficients of g.c.d. of D1(X) and DD1(X)</td>
</tr>
<tr>
<td>ND2</td>
<td>I</td>
<td>ND2</td>
<td>I</td>
<td></td>
<td>Degree of D2(X)</td>
</tr>
<tr>
<td>D3</td>
<td>C</td>
<td>UD3,VD3</td>
<td>D</td>
<td></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></td>
<td>Degree of D3(X)</td>
</tr>
<tr>
<td>D4</td>
<td>C</td>
<td>UD4,VD4</td>
<td>D</td>
<td></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></td>
<td>Degree of D4(X)</td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td>UG, VG</td>
<td>D</td>
<td></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></td>
<td>Degree of G(X)</td>
</tr>
<tr>
<td>ZROS</td>
<td>C</td>
<td>UZROS,VZROS</td>
<td>D</td>
<td></td>
<td>Array of roots of G(X)</td>
</tr>
</tbody>
</table>

Main Program

Subroutine ZROS

APROX C UAPROX,VAPROX D R  Starting approximation (initial or altered)
TABLE G.11 (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>M</td>
<td>I</td>
<td>M</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial to be multiplied</td>
</tr>
<tr>
<td>F</td>
<td>C</td>
<td>UF, VF</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of polynomial to be multiplied</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial to be multiplied</td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td>UC, VG</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of polynomial to be multiplied</td>
</tr>
<tr>
<td>MN</td>
<td>I</td>
<td>MN</td>
<td>I</td>
<td>R</td>
<td>Degree of product polynomial H(X)</td>
</tr>
<tr>
<td>H</td>
<td>C</td>
<td>UH, VH</td>
<td>D</td>
<td>R</td>
<td>Array of coefficients of product polynomial</td>
</tr>
<tr>
<td>LIMIT</td>
<td>I</td>
<td>LIMIT</td>
<td>I</td>
<td>Number of coefficients of polynomial F(X)</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td>Counter</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)
TABLE G.III

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

******************************************************************************
0001 DOUBLE PRECISION EPS1,EPS2,EPS3,UP,VP,UAPP,VAPP,UD0,VDO,UD00,VDO0,
1UD1,V01,V02,V0D2,V0D1,V0D11,V0D3,V0D3,UD04,V0D4,U2R05,V2R05,V0R05,VAP,VA
2P,VR01,VR02,VR03
0002 DOUBLE PRECISION XSTART
0003 DOUBLE PRECISION XEND
0004 DIMENSION ANAMEI2),UP(26),VP(26),UAPP(251),VAPP(251),UD0(126),VDO(126)
1UD01261,V0D01261,U0D1261,V0D1261,U0D261,V0D261,U0D01261,V0D01261
2UD0261,V0D261,UD03511,V0D3511,V0D4511,V0D4511,U2R05,VD05,VD05,UE12
351,VAPP251),VAP251),VR01261),VR01261),MULT1251),ENTRY126)
0005 COMMON EPS1,EPS2,EPS3,I02,MAX
0006 DATA ASTER/4H****/
0007 DATA PNAME,GNAME/2HP,2HG(/,
0008 DATA NAME/3HD1(/**
0009 DATA ENTRY/IHI,IH2,LH3,1H4,IH5,IH6,1H71,H8,IH9,2H10,2H11,2H12,2H13
0010 1,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,2H24,2H25,2H26/
0011 DATA ANAME12),ANAME12)/4HNEWT,4HONS
0012 101=5
0013 102=6
0014 READ(IO1,10001 NOPOLYNP,NAPP,MAXEPSI,E2,EPS3,XSTART,XENDKCHEC
0015 KP
0016 IF(KPNEQ.1)
0017 STOP
0018 WRITE(IO2,10201 ANAME1),ANAME1,NOPOLY
0019 WRITE(IO2,20001 NAPP
0020 WRITE(IO2,20501 EPS1
0021 WRITE(IO2,20601 EPS2
0022 WRITE(IO2,20701 EPS3
0023 WRITE(IO2,20801 XSTART
0024 WRITE(IO2,20901 XEND
0025 KK=NK+1
0026 NNN=KKK+1
0027 ON 5 1=1,KK
0028 5 READ(IO1,1010) UP(JJJ),VP(JJJ)
0029 IF(NAPP.NE.0) GO TO 22
0030 NAPP=NAP
0031 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0032 GO TO 23
0033 22 READ(IO1,1015) (UAPP(I),VAPP(I),NAPP=1,NAPP
0034 GO TO 23
0035 23 WRITE(IO2,10300) NP
0036 KK=NP+1
0037 NNN=KNN+1
0038 ON R 1=1,KK
0039 JJJ=NNN-1
0040 WRITE(IO2,10400) NAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
TABLE G.III (Continued)

0041 \( \text{J} = 1 \)
0042 \( \text{KKK} = \text{NP} + 1 \)
0043 \( \text{DO} 10 \ \text{I}=1,\text{KKK} \)
0044 \( \text{UDDI} = \text{UPI} \)
0045 \( \text{10} \ \text{VDDI} = \text{VPI} \)
0046 \( \text{ND} = \text{UP} \)
0047 \( \text{CALL DERIV(ND0,UD0,VD0,ND0,UD0,VD0,ND0,UD0,VD0)} \)
0048 \( \text{CALL GCD(ND0,UD0,VD0,ND0,UD0,VD0,ND0,UD0,VD0)} \)
0049 \( \text{20 WRITE(102,3000) (*)ST,} = 1,33) \)
0050 \( \text{IF(ND1,LE,1)} \text{GO TO 30} \)
0051 \( \text{GO TO 40} \)
0052 \( \text{30 UD2(1)} = 1.0 \)
0053 \( \text{VD2(1)} = 0.0 \)
0054 \( \text{ND2} = 0 \)
0055 \( \text{GO TO 50} \)
0056 \( \text{40 CALL DERIV(ND0,UD0,VD0,ND0,UD0,VD0,ND0,UD0,VD0)} \)
0057 \( \text{CALL GCD(ND0,UD0,VD0,ND0,UD0,VD0,ND0,UD0,VD0)} \)
0058 \( \text{50 IF(ND0,ND2,LE,2*ND1)} \text{GO TO 60} \)
0059 \( \text{GO TO 70} \)
0060 \( \text{60 WRITE(102,1025) J1} \)
0061 \( \text{GO TO 170} \)
0062 \( \text{70 IF(ND1,LE,0)} \text{GO TO 80} \)
0063 \( \text{GO TO 90} \)
0064 \( \text{80 KKK} = \text{ND0} + 1 \)
0065 \( \text{DO 85 I}=1,\text{KKK} \)
0066 \( \text{UG(I)} = \text{UDDI} \)
0067 \( \text{VG(I)} = \text{VDDI} \)
0068 \( \text{NG} = \text{ND0} \)
0069 \( \text{GO TO 110} \)
0070 \( \text{90 IF(ND2,LE,0)} \text{GO TO 115} \)
0071 \( \text{CALL PROD(ND0,UD0,VD0,ND0,UD0,VD0,ND0,UD0,VD0)} \)
0072 \( \text{100 CALL PROD(ND0,UD0,VD0,ND0,UD0,VD0,ND0,UD0,VD0)} \)
0073 \( \text{110 WRITE(102,1035) J1} \)
0074 \( \text{115 KKK} = \text{NG} + 1 \)
0075 \( \text{NNN} = \text{KKK} + 1 \)
0076 \( \text{DO 112 I}=1,\text{KKK} \)
0077 \( \text{J1JJ} = \text{NNN} - 1 \)
0078 \( \text{112 WRITE(102,1040) GNAM,ENTRY(JJJ),UG(JJJ),VG(JJJ)} \)
0079 \( \text{CALL ZEROISING,UG,VG,NAPP,JAPP,JAPP,J,IZROS,IZROS,JAP,} \)
0080 \( \text{UAP,VP,ENTRY} \)
0081 \( \text{1*,START,REND)} \)
0082 \( \text{IF(JJ,LE,0)} \text{GO TO 150} \)
0083 \( \text{WRITE(102,1180)} \)
0084 \( \text{115 KKK} = \text{ND0} + 1 \)
0085 \( \text{120 KKK} = \text{JAP} + 1 \)
0086 \( \text{DO 116 I}=1,\text{KKK} \)
0087 \( \text{116 UD3I} = \text{UD0I} \)
0088 \( \text{VD3I} = \text{VDDI} \)
0089 \( \text{NO3} = \text{ND0} \)
0090 \( \text{GO TO 100} \)
0091 \( \text{120 KKK} = \text{JAP} + 1 \)
0092 \( \text{WRITE(102,1085) (*)UGROS(I),VROS(I),J1I,J1=KKK,J1} \)
0093 \( \text{GO TO 140} \)
0094 \( \text{130 WRITE(102,1190) (*)UGROS(I),VROS(I),J1J,AP,VP,ENTRY,} \)
0095 \( \text{1*,JAP,LE,J1} \text{GO TO 120} \)
0096 \( \text{140 IF(JJ,LE,NG)} \text{GO TO 155} \)
0097 \( \text{150 WRITE(102,1095)} \)
TABLE G.III (Continued)

0078 IF(J.EQ.0 GO TO 170
0079 155 DD 180 I=1,J
0080 UNROOT(KD+1)=VRQDS(I)
0081 VRQDS(KD+1)=VRQDS(I)
0082 160 MULT(KD+1)=J
0083 K=(J*J)+K
0084 KD=KD+J
0085 IF(K.GE.NP) GO TO 1
0086 170 J=J+1
0087 IF(ND.I.EQ.1) GO TO 200
0088 DD 180 I=1,NDI
0089 UN(I)=UD(I)
0100 VD(I)=VD(I)
0101 UNDO(I)=UDI(I)
0102 VD(I)=VDI(I)
0103 UNIND(I)=UDI(I)
0104 VD(I)=VDI(I)
0105 UNIND(I)=UDI(I)
0106 VD(I)=VDI(I)
0107 UNIND(I)=UDI(I)
0108 VD(I)=VDI(I)
0109 UNIND(I)=UDI(I)
0110 VD(I)=VDI(I)
0111 UNIND(I)=UDI(I)
0112 VD(I)=VDI(I)
0113 UNIND(I)=UDI(I)
0114 VD(I)=VDI(I)
0115 ND=NDI
0116 NDD=NDDI
0117 KK=NND+1
0118 DD 190 I=1,KK
0119 UN(I)=UDI(I)
0120 VD(I)=VDI(I)
0121 ND=ND2
0122 GO TO 20
0123 200 IF(ND.I.EQ.0) GO TO 1
0124 KD=KD+1
0125 DENOM=UDI(I)*VD(I)+UDI(I)*VD(I)
0126 VRQDS(KD)=I-UDI(I)*VD(I)/DENOM
0127 UROOT(KD)=I+UDI(I)*VD(I)/DENOM
0128 MULT(KD)=J
0129 WRITE(102,1133)TASER,1,333
0130 WRITE(102,1035)J
0131 K=KK
0132 NNN=KKK+1
0133 DD 210 I=1,KK
0134 JJ=NNN-1
0135 210 WRITE(102,1100)DGNAME,ENTRY(JJJJ,UDIJJJJ),VDI(JJJ)
0136 WRITE(102,1180)
0137 WRITE(102,1085)KD,UROOT(KD),VRQDS(KD),J
0138 GO TO 1
0139 1020 FORMAT(1H1),10X,4HREPEATED USE OF THE GREATEST COMMON DIVISOR AND
0140 1,4,4,4,SIM METHOD TO EXTRACT ROOTS AND MULTIPLECTIES OF POLYNOMIA
0141 215/14,16POLYNOMIAL NUMBER,12/1
0142 1025 FORMAT(//1X,25HROOTS OF MULTIPLECTY 12/1
0143 1035 FORMAT(//1X,25HTHE FOLLOWING POLYNOMIAL, G(I), CONTAINS ALL THE R
0144 1055 FURMAT(2X,5HROT(I),12,4H)
0145 1045 FORMAT(2X,5HROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUND
0146 1050 FORMAT(1H1),10X,25HTHE DEGREE OF P(X) IS 12,22H THE COEFFICIENTS ARE//
0147 1060 FORMAT(1H1),10X,25HROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUND
0148 1070 FORMAT(1H1),10X,25HROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUND
0149 1080 FORMAT(1H1),10X,25HROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUND
TABLE G.III (Continued)

0150 1180 FORMAT(IX,13HROOTS OF PI,2X,14HMULTIPLICITIES,17X,21HINITIAL
1 APPROXIMATION//)
0151 1190 FORMAT(IX,2X,5HROOT(I2,4H)=.D23.16,3H+.D23.16,2H1,7X,12,7X,02).116.3H+.D23.16,2H1)
0152 2000 FORMAT(IX,41HNUMBER OF INITIAL APPROXIMATIONS GIVEN.,12)
0153 2010 FORMAT(IX,13HMAXIMUM NUMBER OF ITERATIONS.,11X,13)
0154 2020 FORMAT(IX,13HTEST FOR CONVERGENCE.,13X,09.2)
0155 2040 FORMAT(IX,23HRAADIUS TO START SEARCH.,11X,09.2)
0156 2050 FORMAT(IX,21HRAADIUS TO END SEARCH.,13X,09.2)
0157 2060 FORMAT(/X)
0158 2070 FORMAT(IX,34HTEST FOR ZERO IN SUBROUTINE GCD.,09.2)
0159 2080 FORMAT(IX,34HTEST FOR ZERO IN SUBROUTINE QUAD.,09.2)
0160 3000 FORMAT(///,IX,A3,A5,A4)
0161 END
TABLE G.III (Continued)

FUNCTIONS R(x) AND S(x), THIS SUBROUTINE COMPUTES THE COEFFICIENTS OF THE PRODUCT POLYNOMIAL T(x) = R(x) * S(x).

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

DIMENSION UH(M+1), VH(N+1), UF(51), VF(51), VG(51)

MN = M + N

K = 1

UH(1) = 0.0

IF (K <= M + 1) GO TO 10
LIMIT = M + 1
GO TO 10
10 LIMIT = I
GO TO 20
20 K = K - 1
CONTINUE
RETURN
END

SUBROUTINE GENAPP(APPR, APP, NAPP, XSTART)

DIMENSION APPR(25), APP(25)
COMMON EPS1, EPS2, EPS3

IF (XSTART .EQ. 0.0) 
XSTART = 0.5
EPS1 = 0.2617994
DO I = 1, NAPP
APPRI = XSTART * COS(BETA)
APP(I) = XSTART * SIN(BETA)
BETA = BETA * 0.5235988
CONTINUE
RETURN
END
### TABLE G.III (Continued)

**SUBROUTINE ALTER** (XOLDR, XOLDI, NALTER, ITIME)

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

C

**DOUBLE PRECISION** XOLDR, XOLDI, ABXOLD, BETA, EPS1, EPS2, EPS3

**COMMON** EPS1, EPS2, EPS3, IO2, MAX

IF (ITIME .NE. 0) GO TO 5

ITIME = 1

WRITE(IO2, 1010) MAX

IF (FINALTER .EQ. 0) GO TO 10

WRITE(IO2, 10001) XOLDR, XOLDI

GO TO 20

10 ABXOLD = DSQRT((XOLDR*XOLDI)*IXOLOI*XOLDII)

BETA = OATAN2(XOLDI, XOLDR)

WRITE(IO2, 10201) XOLDR, XOLDI

NALTER = NALTER + 1

IF (FINALTER .GT. 51) RETURN

GO TO (30, 40, 30, 40, 30), NALTER

30 XOLDR = -XOLDR

XOLDI = -XOLDI

GO TO 50

40 BETA = BETA*1.0471976

XOLDR = ABXOLD*DCOS(BETA)

XOLDI = ABXOLD*DSIN(BETA)

RETURN

1000 FORMAT(Ix, D23.16, 3H + .023.16, 2H I, 10X, 21HALTERED APPROXIMATION)

1010 FORMAT(/IX, 5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF ITER , 13.12H ITERATIONS, //)

1020 FORMAT(Ix, D23.16, 3H + .023.16, 2H I, 10X, 21INITIAL APPROXIMATION)

END
### TABLE G.111 (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE ZEROS(INQ, UQ, VQ, NAPP, VAPP, J, URDOT, VROOT, JAP, UAP, VAP, ENTRY, XSTART, XEND)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>**********************************************************</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>* NEUMANN'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLEITIES OF A</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>* POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>* IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>* FORMULA</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>( x(N+1) = x(N) - \frac{p(x(N))}{p'(x(N))} ).</td>
</tr>
<tr>
<td>C</td>
<td>**********************************************************</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>DOUBLE PRECISION UAPP, UAP, VAPP, URDOT, VROOT, URZRO, VZRO, UQ, VQ, UDUMMY, VDUM</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>JQP, UQ(J), VQ(J), UAP(J), VAP(J), VQ(J), URDOTS, VROOTS, EPS1, EPS2, EPS3, UAPROX, VAP</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td>DOUBLE PRECISION XEND, XSTART</td>
<td></td>
</tr>
<tr>
<td>0004</td>
<td>DIMENSION UAPP(25), VAPP(25), URDOT(25), VROOT(25), UQ(25), VQ(25), UQ(J)</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>0005</td>
<td>COMMON EPS1, EPS2, EPS3, J0, MAX</td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td>DATA QNAME, QNAME/MHQ, M1/SUM, M1/SUM, M1/SUM</td>
<td></td>
</tr>
<tr>
<td>0007</td>
<td>LOGICAL CONV</td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>J=1</td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>ITIME=0</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>IF (INQ.GE.3) GO TO 85</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>GO TO 110</td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td>85 KKK=NO+1</td>
<td></td>
</tr>
<tr>
<td>0013</td>
<td>GO 90 1=1,KKK</td>
<td></td>
</tr>
<tr>
<td>0014</td>
<td>UQ(1)=UQ(1)</td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td>90 VQ(1)=VQ(1)</td>
<td></td>
</tr>
<tr>
<td>0016</td>
<td>NO=NO</td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td>GO TO 120</td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td>110 CALL QUADANG(UQ, VQ, J, URDOT, VROOT)</td>
<td></td>
</tr>
<tr>
<td>0019</td>
<td>JAP=0</td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>GO TO 310</td>
<td></td>
</tr>
<tr>
<td>0021</td>
<td>120 DO 200 I=1, NAPP</td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td>ALTER=0</td>
<td></td>
</tr>
<tr>
<td>0023</td>
<td>UAPROX=UAPP(I)</td>
<td></td>
</tr>
<tr>
<td>0024</td>
<td>VAPROX=VAPP(I)</td>
<td></td>
</tr>
<tr>
<td>0025</td>
<td>130 CALL NEWTON(UAPROX, VAPROX, NOQ, UQ(J), VQ(J), URZRO, VZRO, CONV)</td>
<td></td>
</tr>
<tr>
<td>0026</td>
<td>IF (CONV) GO TO 160</td>
<td></td>
</tr>
<tr>
<td>0027</td>
<td>CALL ALTER(UAPP(I), VAPP(I), ALTER, ITIME)</td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td>IF (ALTER.GT.5) GO TO 200</td>
<td></td>
</tr>
<tr>
<td>0029</td>
<td>UAPROX=UAPP(I)</td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td>VAPROX=VAPP(I)</td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td>GO TO 130</td>
<td></td>
</tr>
<tr>
<td>0032</td>
<td>160 J=J+1</td>
<td></td>
</tr>
<tr>
<td>0033</td>
<td>URDOT(J)=URZRO</td>
<td></td>
</tr>
<tr>
<td>0034</td>
<td>VROOT(J)=VZRO</td>
<td></td>
</tr>
<tr>
<td>0035</td>
<td>UAP(J)=UAPROX</td>
<td></td>
</tr>
<tr>
<td>0036</td>
<td>VAP(J)=VAPROX</td>
<td></td>
</tr>
<tr>
<td>0037</td>
<td>CALL HOPNER(UZRO, VZRO, UQ(J), VQ(J), UQ(J), VQ(J), UDUMMY, VDUM)</td>
<td></td>
</tr>
<tr>
<td>0038</td>
<td>GO 180 I=1, NOQ</td>
<td></td>
</tr>
<tr>
<td>0039</td>
<td>180 UQ(J)=UQ(J)+1</td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td>UQ(J)=UQ(J)+1</td>
<td></td>
</tr>
<tr>
<td>0041</td>
<td>NOQ=NOQ+1</td>
<td></td>
</tr>
<tr>
<td>0042</td>
<td>IF (INQ.GE.3) GO TO 200</td>
<td></td>
</tr>
<tr>
<td>0043</td>
<td>JAP=J</td>
<td></td>
</tr>
<tr>
<td>0044</td>
<td>GO TO 220</td>
<td></td>
</tr>
</tbody>
</table>
### 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.2I) GO TO 210
0053 WRITE(102,1200)
0054 KKK=NO+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.EQ.0) GO TO 230
0060 CALL QUAD(UQQ,VQQ,J,UROOT,VROOT)
0061 GO TO 235
0062 220 CALL QUAD(UQQ,VQQ,J,UROOT,VROOT)
0063 WRITE(102,1131) (UROOT(I),VROOT(I),I=1,JAP)
0064 IF(JAP.LT.J) GO TO 235
0065 GO TO 240
0066 235 KKK=JAP+1
0067 WRITE(102,1134) (UROOT(I),VROOT(I),I=KKK,J)
0068 240 J=0
0070 DO 300 I=1,J
0071 CALL NEWTON(UROOT(I),VROOT(I),NO,UQ,VQ,UZRO,VZRO,CONV)
0072 IF(IVAL) GO TO 280
0073 WRITE(102,1140) (UROOT(I),VROOT(I),MAX,NO
0074 KKK=NO+1
0075 NNN=KKK+1
0076 DO 242 L=1,KKK
0077 JJJ=NNN-L
0078 242 WRITE(102,1040) QNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0079 IF(J.EQ.JAP) GO TO 241
0080 WRITE(102,1100) 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 VAP(I)=VAP(I+1)
0085 245 J=J+1
0086 240 JAP=JAP-1
0087 GO TO 300
0088 280 J=J+1
0089 UROOTS(JIJ)=UZRO
0090 VROOTS(JIJ)=VZRO
0091 300 CONTINUE
0092 J=J+1
0093 IF(J.EQ.0) GO TO 305
0094 DO 303 I=1,J
0095 WRITE(UROOT(I),UROOTS(I)
0096 303 UROOT(I)=UROOTS(I)
0097 GO TO 310
0098 305 WRITE(102,1150) NQ
0099 KKK=NO+1
0100 NNN=KKK+1
0101 DO 306 L=1,KKK
0102 JJJ=NNN-L
```

TABLE G.III (Continued)

0103 WRITE(102,1040) QNAME,ENTRY(JJJJ),WQ(JJJJ),VQ(JJJJ)
0104 110 RETURN
0105 1200 FORMAT(///,1X,7OHCOEFFICIENTS OF THE DEFLATED POLYNOMIAL FOR WHICH
103 ZEROS WERE FOUND.///)
0106 1132 FORMAT(///,1X,13HROOTS OF G(X),04X,21HINITIAL APPROXIMATION///)
0107 1133 FORMAT(2X,5HR0OT(1,12,4H) = ,D23.16,3H * ,D23.16,2H I,17X,D23.16,3H
1 * ,D23.16,2H I)
0108 1134 FORMAT(2X,5HROOT(1,12,4H) = ,D23.16,3H * ,D23.16,2H I,22X#26HRESULT
15 OF SURROUNQ Quad)
0109 1140 FORMAT(///,1X,4HNO ROOTS FOR INITIAL APPROXIMATION ROOT(1,12,4H) =
1 ,D23.16,3H * ,D23.16,2H I/6H AND 13,40M ITERATIONS ON THE POLYN
OMIAL OF DEGREE 112, 18H WITH COEFFICIENTS///)
0110 1150 FORMAT(///,1X,4HNO ROOTS FOR THE POLYNOMIAL G(X) OF DEGREE = 12,
13H WITH GENERATED INITIAL APPROXIMATIONS///)
0111 1040 FORMAT(2X,A2,A2,4H) = ,D23.16,3H * ,D23.16,2H I)
0112 1100 FORMAT(2X,A3,A2,4H) = ,D23.16,3H * ,D23.16,2H I)
0113 END
TABLE G.III (Continued)

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

* ****
* *
* GIVEN POLYNOMIALS P(X) AND DPIXI WHERE DEG. DPIX IS LESS THAN DEG.
* *
* P(X). SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF PIXI AND
* *
* ***************************************************

DOUBLE PRECISION USSSS,S,VSSSS


DIMENSION UR(26),VR(26),US(26),USS(26),VSS(26),URR(26),VRR(26),UT(26),VT(26)

COMMON EPSLON,EPS2,EPS3,MAX

M1=N

KAK=N+1

DO 25 I=1,KKK

25 VSS(I)=VSS(I)

30 UB=USS(MI+1)*USS(MI+1)+VSS(MI+1)*VSS(MI+1)

31 UD=URR(MI+1)*USS(MI+1)+VRR(N1+1)*VSS(MI+1)

32 UVR=N1-1

33 DO 40 I=KKK,N1

40 UT(I)=URR(I)-UD*USS(I)-VD*VSS(I)

41 IF(MI.EQ.N1)GO TO 70

42 KKK=N1-M1

43 DO 60 I=KKK,N1

60 UT(I)=URR(I)

61 VT(I)=VRR(I)

70 DO 90 I=1,N1

BBB=DSQRT(UT(I)*UT(I)+VT(I)*VT(I))

71 IF(BBB.GT.EPSLON)GO TO 100

72 URR=USS(MI+1)*USS(MI+1)+VSS(MI+1)*VSS(MI+1)

73 US=USS(MI+1)*VSS(MI+1)+VSS(MI+1)*USS(MI+1)

74 VSS=VSS(MI+1)*VSS(MI+1)

75 K=NI-I

76 DO 95 I=1,M1

90 CONTINUE

DO 95 I=1,N1

D1=URR(I)-UD*USS(I)-VD*VSS(I)

91 DO 130 I=MI,N1

130 NN=NI+1

DO 140 I=1,NN

DO TO 200

100 K=NI-I

110 IF(K.EQ.0)GO TO 170

120 IF(K.EQ.1)GO TO 140

130 IF(K.EQ.2)GO TO 200

140 IF(K.EQ.3)GO TO 170

150 IF(K.EQ.4)GO TO 140

160 IF(K.EQ.5)GO TO 200

170 VI=VR(I)

180 VI=VI+USS

190 VI=VI+VSS

200 GO TO 30
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0050</td>
<td>140  KKK=KK+1</td>
</tr>
<tr>
<td>0051</td>
<td>DO 150 J=1, KK</td>
</tr>
<tr>
<td>0052</td>
<td>150 URR(J)=USS(IJ)</td>
</tr>
<tr>
<td>0053</td>
<td>VRR(J)=VSS(IJ)</td>
</tr>
<tr>
<td>0054</td>
<td>USS(IJ)=U(J)</td>
</tr>
<tr>
<td>0055</td>
<td>150 VSS(IJ)=VT(J)</td>
</tr>
<tr>
<td>0056</td>
<td>KKK=KK+2</td>
</tr>
<tr>
<td>0057</td>
<td>NNN=M1+1</td>
</tr>
<tr>
<td>0058</td>
<td>DO 160 J=KK+1, NNN</td>
</tr>
<tr>
<td>0059</td>
<td>160 URR(J)=USS(IJ)</td>
</tr>
<tr>
<td>0060</td>
<td>VRR(J)=VSS(IJ)</td>
</tr>
<tr>
<td>0061</td>
<td>M1=M1</td>
</tr>
<tr>
<td>0062</td>
<td>M1=K</td>
</tr>
<tr>
<td>0063</td>
<td>GO TO 30</td>
</tr>
<tr>
<td>0064</td>
<td>170 USS(I)=1.0</td>
</tr>
<tr>
<td>0065</td>
<td>VSS(I)=0.0</td>
</tr>
<tr>
<td>0066</td>
<td>M1=0</td>
</tr>
<tr>
<td>0067</td>
<td>200 RETURN</td>
</tr>
<tr>
<td>0068</td>
<td>END</td>
</tr>
</tbody>
</table>

TABLE G.III (Continued)
TABLE G.III (Continued)

0001 SUBROUTINE NEWTONIUXVX,N,UPV,UPV,UXO,VXO,CONV)
* *-------------------------------------------------------------------------*
* THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-
* FORMULA USING THE ITERATION FORMULA:
* X(N+1) = X(N) - P(X(N))/P'(X(N)).
* *-------------------------------------------------------------------------*
0002 DOUBLE PRECISION UX,VX,UP,VP,UXO,VXO,UB,VB,UPXO,VPXO,UBP,VPXO,VDPXO,U
0003 DOUBLE PRECISION UDIFF,VDIFF,EPSI,EPSLON,EPS3,AAA,BBB
0004 DOUBLE PRECISION ABPXO
0005 DIMENSION UP(26),VP(26),UB(26),VB(26)
0006 COMMON EPSI,EPSLON,EPS3,102,MAX
0007 LOGICAL CONV
0008 UXO=UX
0009 VXO=VX
0010 DO 10 I=1,MAX
0011 CALL HORNERIUXOVXO.N,UP,VP,UB,VB,UPXO,VPXO,UBP,VPXO,VDPXO
0012 UPXO=UB(1)
0013 VPXO=VB(1)
0014 DDD=DSQRT(UPXO*UPXO+VPXO*VPXO)
0015 IF (ODD.EQ.0) GO TO 5
0016 IF (ABPXO.EQ.0) GO TO 20
0017 GO TO 15
0018 5 BBB=UDPXO*UPXO+VDPXO*VDPXO
0019 SBB=UDPXO*UPXO+VDPXO*VDPXO
0020 BDIFF=(VPXO*UPXO-UPXO*VPXO)*UPXO+VDPXO*VDPXO
0021 VDIFF=(VPXO*UPXO-UPXO*VPXO)*VDPXO
0022 UXX=UXO-UDIFF
0023 VXO=VXO-VDIFF
0024 AAA=DSQRT(UDIFF*UDIFF+VDIFF*VDIFF)
0025 BBB=DSQRT(UXX*UXX+VXX*VXX)
0026 IF (AAA.EQ.0) GO TO 10
0027 IF (BBBB.EQ.0) GO TO 20
0028 10 CONTINUE
0029 15 CONV=.FALSE.
0030 RETURN
0031 20 CONV=.TRUE.
0032 RETURN
0033 END
TABLE G.III (Continued)

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

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

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

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

K = N - M

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

UQ(K + 1) = (UP(N + K) + VP(N + K)) / UDUMMY

VQ(K + 1) = (VP(N + K) * UD(M + 1) - UP(N + K) * 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(MI .GT. 1) GO TO 10

GO TO 45

10 UTERM = UTERM - (UQ(KK) * UD(MI) - VQ(KK) * VD(MI))

VTERM = VTERM + (UQ(KK) * VD(MII) + VQ(KK) * UD(MII))

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

40 KK = KK - 1

100 RETURN

END
TABLE G.111 (Continued)

0001 SUBROUTINE HORNER(UX, VX, N, UP, VP, UB, VB, UC, VC)
C
C * *******************************************************
C * HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A
C * POINT D AND ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO
C * DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X - D).
C * *******************************************************
C
0002 DOUBLE PRECISION UX, VX, UP, VP, UB, VB, UC, VC
0003 DOUBLE PRECISION UDUMMY, VDUMMY
0004 DIMENSION UP(26), VP(26), UB(26), VB(26)
0005 UBIN = UP(N+1)
0006 VB(N+1) = VP(N+1)
0007 UB(N) = UX*UB(N) + VX*VB(N) + UP(N)
0008 VB(N) = UX*VB(N) + VX*UB(N) + VP(N)
0009 UC = UB(N)
0010 VC = VB(N)
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)
0014 VB(KKK+1-I) = UX*VB(KKK+2-I) + VX*UB(KKK+2-I) + VP(KKK+1)
0015 UDUMMY = UC-VC*VX*VC
0016 VDUMMY = UX*VC*VX*UC
0017 UC = UDUMMY + VB(KKK+2-I)
0018 VC = VDUMMY + VB(KKK+2-I)
0019 RETURN
0020 END
TABLE G.III (Continued)

0001 SUBROUTINE QUAD(NUA,VA,J,URoot,VRoot)
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
**************************************************************************
C
DOUBLE PRECISION EPS1,EPS2,EPSLON,URoot,VRoot,UA,VA,UDISC,VDISC,UD
0002 DIMENSION URoot(25),VRoot(25),UA(26),VA(26)
0003 COMMON EPS1,EPS2,EPSLON,IO2,MAX
0004 IF(IN.GT.1) GO TO 10
0005 J=J+1
0006 BB=UA(I)*UA(2)+VA(2)*VA(2)
0007 URoot(J)=-UA(I)*UA(2)+VA(2))/BB
0008 VRoot(J)=-VA(I)*UA(I)*VA(2))/BB
0009 UDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(I)*UA(I)-VA(I)*VA(I))
0010 VDISC=(UA(2)*UA(2)-VA(2)*VA(2))-(4.0*(UA(I)*VA(I)+VA(I)*UA(I))
0011 UD=2.0*UA(I)
0012 VD=2.0*VA(I)
0013 DDD=SQRT(UDISC*UDISC+VDISC*VDISC)
0014 IF(DDD.LT.EPSLON) GO TO 20
0015 CALL COMSQT(UDISC,VDISC,UTEMP,VTEMP)
0016 BB=UD*UD+VD*VD
0017 URoot(J+1)=(-UA(I)+UTEMP)*UD-VA(I)*VD)/BB
0018 VRoot(J+1)=(-VA(I)+VTEMP)*UD-(-UA(I)+VTEMP)*VD)/BB
0019 J=J+2
0020 GO TO 10
0021 J=J+1
0022 BB=UA(I)*UA(I)+VD*VD
0023 URoot(J+1)=(-UA(I)+UTEMP)*UD-VA(I)*VD)/BB
0024 VRoot(J+1)=(-VA(I)+VTEMP)*UD-(-UA(I)+VTEMP)*VD)/BB
0025 WRITE(IO2,1000) URoot(J),VRoot(J)
0026 1000 FORMAT(///Ix,16.16,E15.16,2H TO BE A M
0027 MULTIPLE ROOT/)
0028 100 RETURN
0029 END
TABLE G.III (Continued)

0001 SUBROUTINE DERIV(N,UP,VP,U,UA,VA)
C ***********************************************************************
C * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF
C * ITS DERIVATIVE P'(X).
C ***********************************************************************
0002 DOUBLE PRECISION UP,VP,U,UA,VA
0003 DIMENSION UP(261),VP(261),UA(261),VA(261)
0004 NKK=N+1
0005 DO 10 I=2,NKK
0006 AAA=I-I
0007 UA(I-1)=AAA*UP(I)
0008 VA(I-1)=AAA*VP(I)
0009 M=N-1
0010 RETURN
0011 END

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

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

1. Use of the Program

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

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

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

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

Main Program

Data Entry/1H1, 1H2, ..., 1H9, 2H10, 2H11, ..., 2HXX/where XX = N+1
UAPP(N,3), VAPP(N,3)
URAPP(N,3), URAPP(N,3)
UP(N+1), VP(N+1)
MULT(N)
UDD0(N+1), VDD0(N+1)
UD1(N+1), VD1(N+1)
UDD1(N+1), VDD1(N+1)
UD2(N+1), VD2(N+1)
UG(N+1), VG(N+1)
UD3(2N+1), VD3(2N+1)
UD4(2N+1), VD4(2N+1)
UAP(N+1), VAP(N+1)
UZROS(N), VZROS(N)
UROOT(N), VROOT(N)
UD0(N+1), VDO(N+1)
ENTRY(N+1)

Subroutines PROD, QUAD
See corresponding subroutine in Table G.I.

Subroutines DERIV, GCD, and DIVIDE
See corresponding subroutine in Table E.I.

Subroutines MULLER, GENAPP, BETTER and HORNER
See corresponding subroutine in Table F.I.

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

The input data to the repeated G.C.D. - Muller's method is the same as for the repeated G.C.D. - Newton's method as described in Appendix G, § 2.
3. Variables Used in Repeated G.C.D. – Muller's Method

The variables used in this program are referenced in Table H.II. The notation and symbols used in the referenced tables are described in Appendix E, § 3.

### TABLE H.II

**VARIABLES USED IN REPEATED G.C.D. – MULLER'S METHOD**

Main Program and Subroutine PROD

See Table G.II.

Subroutines QUAD, DERIV, GCD, DIVIDE, and COMSQRT

See corresponding subroutine in Table E.VI.

Subroutines CALC, MULLER, GENAPP, ALTER, BETTER, TEST, and HORNER.

See corresponding subroutine in Table F.II.

4. Description of Program Output

The output for this program is the same as that for repeated G.C.D. – Newton's method as described in Appendix G, § 4. Only one initial approximation, \( X_0 \), (not three) is printed. The other two required by Muller's method are \( .9X_0 \) and \( 1.1X_0 \). The message "SOLVED BY DIRECT METHOD" means that the corresponding root was obtained by Subroutine QUAD.
5. Informative Messages and Error Messages

Descriptions of the informative messages and error messages printed by this program can be found either in Appendix E, § 5, Appendix F, § 5, or Appendix G, § 5.
Figure H.1. Flow Charts for Repeated G.C.D.-Muller's Method
Figure H.1: (Continued)
Figure H.1. (Continued)
Figure II.1. (Continued)
Figure H.1. (Continued)
DIVIDE

START

K = 0

J = -1

\( q_{n+1} = \frac{q_{n-1}}{q_{n+1}} \)

RETURN

\( q_{n+1} \)

***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. III

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

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

0001 DOUBBLE PRECISION EPS1, EPS2, EPS3, UP, VP, UAPP, VAPP, UDD1, UDDO, VDD0,
        UDD1, UDD2, VDD1, UDD1, UG, VG, UDD3, UDD4, UDD5, VDD5, UAP, VA
        2P, UKOOT, VKOOT, DENOM
0002 DOUBLE PRECISION XSTART
0003 DOUBLE PRECISION XEND
0004 DOUBLE PRECISION UAPP, VRAPP
0005 DOUBLE PRECISION EPS4
0006 DIMENSION UAPP(25, 3), VAPP(25, 3), URAPP(25, 3), VRAPP(25, 3)
0007 DIMENSION UDD(26), VP(26), UDD1(26), VDD1(26), UDD1(26), VDD1(26)
        1, UDD1(26), VDD1(26), UDD1(26), VDD1(26), UDD1(26), VDD1(26)
        1, UDD1(26), VDD1(26), UDD1(26), VDD1(26), UDD1(26), VDD1(26)
        1, UDD1(26), VDD1(26), UDD1(26), VDD1(26), UDD1(26), VDD1(26)
        1, UDD1(26), VDD1(26), UDD1(26), VDD1(26), UDD1(26), VDD1(26)
0008 COMMON EPS1, EPS2, EPS3, EPS4, IOZ, MAX
0009 DATA PNAME, GNAME, ZHP1, 2HGP, BNAME/BKNH/L/
0010 DATA ASTER/4H***/
0011 DATA ENTRY/1H1, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9, 2H10, 2H11, 2H12, 2H13
        1, 2H14, 2H15, 2H16, 2H17, 2H18, 2H19, 2H20, 2H21, 2H22, 2H23, 2H24, 2H25, 2H26/
0012 DATA NAME(11), NAME(2)/4HMSL/4HERS/
0013 I03 = 5
0014 I02 = 6
0015 1 READ(101, 1000) NOPOLY, NP, NAPP, MAX, EPS1, EPS2, EPS3, XSTART, XEND, KCMEC
        1K
0016 IF (KCHECK.EQ.1) STOP
0017 WRITE(102, 1020) NAME(1), NAME(2), NOPOLY
0018 WRITE(102, 2000) NAPP
0019 WRITE(102, 2010) MAX
0020 WRITE(102, 2070) EPS1
0021 WRITE(102, 2020) EPS2
0022 WRITE(102, 2080) EPS3
0023 WRITE(102, 2040) XSTART
0024 WRITE(102, 2050) XEND
0025 WRITE(102, 2060)  
0026 KKK = NP + 1
0027 NNN = KKK + 1
0028 GO TO 5, 1 = 1, KKK
0029 JJJ = NNN - 1
0030 5 READ(101, 1010) UP, VJH, VP(JJJ)
0031 IF (NAPP .NE. 0) GO TO 22
0032 NAPP = NP
0033 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
0034 GO TO 23
0035 22 WRITE(102, 1015) (UAPP(I, 2), VAPP(I, 2), I = 1, NAPP)
0036 23 WRITE(102, 1030) NP
0037 KKK = NP + 1
0038 NNN = KKK + 1
0039 GO TO 8, 1 = 1, KKK
0040 JJJ = NNN - 1
TABLE H.III (Continued)

8 WRITE(IO2,1040) PNAME,ENTRY(JJJJ),UP(JJJJ),VP(JJJJ)

K=0
K0=0
J=1
KKK=NP+1
DO 10 I=1,KKK
UDD(I)=UP(I)
10 VDO(I)=VP(I)
NDO=NP
CALL DERIV(NDO,UD0,VDO,NDD0,UDDD,VDDD)
CALL GCD(NDO,UD0,VDO,NDD0,UDDD,VDDD,N01,UD1,V01)
WRITE(IO2,3000) (ASTER,=1,33)
IF(N01.LE.1) GO TO 30
GO TO 40
10 VDO(I)=V00(I)
NDO=N00
CALL DERIV(NDO,UD0,VDO,NDD0,UDDD,VDDD)
20 WRITE(IO2,3000) (ASTER,=1,33)
IF(N01.LE.1) GO TO 30
GO TO 40
30 UD2(I)=1.0
V02(I)=0.0
ND2=0
GO TO 50
40 CALL DERIV(ND0,UD0,V00,NDD0,UDDD,VDDD)
IF(NDD0.NE.0) GO TO 50
50 IF(NDD0+ND2.LE.2*ND0) GO TO 60
GO TO 70
60 WRITE(IO2,10251) JI
GO TO 170
70 IF(N02.EQ.0) GO TO 80
GO TO 90
80 KKK=ND0+1
DO 85 I=1,KKK
UD(I)=UD0(I)
V0(I)=V00(I)
NG=NDD0
GO TO 110
85 NG=ND0
GO TO 110
90 IF(N02.EQ.0) GO TO 115
CALL PROD(NDO,UD0,V00,NDD0,UDDD,VDDD,N02,UD2,V02)
CALL PROD(ND0,UD0,V00,NDD0,UDDD,VDDD,N02,UD2,V02)
IF(NDD0.NE.0) GO TO 120
WRITE(IO2,1035) J1
Kkk=NG+1
N00=KKK+1
GO TO 116
112 WRITE(IO2,1040) GNAME,ENTRY(JJJJ),UG(JJJJ),VG(JJJJ)
K=NG+1
DO 113 I=1,KKK
113 UAP(I)=UG(KKK+1-I)
VAP(I)=VG(KKK+1-I)
CALL MULLER(NG,UG,UAP,NAP,VPAP,VAPP,UG,VPAP,VAPP,VPAP,VAPP,UG,VPAP)
IF(J.EQ.0) GO TO 150
WRITE(IO2,1180)
IF(JAP.EQ.0) GO TO 120
GO TO 130
115 KKK=NDD0+1
DO 116 I=1,KKK
116 UD3(I)=UD0(I)
N03=NDD0
GO TO 100
TABLE H.111 (Continued)

0098 120  KKK=JAP+1
0099 WRITE(102,1085) (I,UZROS(I),VZROS(I),J1,1=KKK,J)
0100 GO TO 140
0101 130  DO 135 I=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 IF(I.EQ.NG) GO TO 155
0105 WRITE(102,1095)
0106 I.FI.EQ.0) GO TO 170
0107 155 DO 160 I=1,J
0108 UROOT(KD+1)=UZROS(I)
0109 VROOT(KD+1)=VZROS(I)
0110 MULT(KD+1)=J
0111 K=J+1
0112 KD=KD+1
0113 IF(K.LT.NP) GO TO 1
0114 170 J=J+1
0115 IF(ND1.LT.1) GO TO 200
0116 UD(I)=UD1(I)
0117 VD(I)=VD1(I)
0118 UD0(I)=UD1(I)
0119 VD0(I)=VD1(I)
0120 160 VD0(I)=VD1(I)
0121 UD0(ND1+1)=UD1(ND1+1)
0122 VD0(ND1+1)=VD1(ND1+1)
0123 ND0=ND1
0124 ND0=ND1
0125 KKK=ND0+1
0126 DO 190 I=1,KKK
0127 UD(I)=UD2(I)
0128 VD(I)=VD2(I)
0129 ND1=ND0
0130 GO TO 20
0131 200 IF(ND1.LT.0) GO TO 1
0132 KD=KD+1
0133 DENOM=UD1(2)*UD1(2)+VD1(2)*VD1(2)
0134 UR0OT(KD)=-UD1(1)*UD1(1)+VD1(1)*VD1(1))/DENOM
0135 VR0OT(KD)=1-VD1(1)*VD1(1)+UD1(1)*UD1(1))/DENOM
0136 MULT(KD+1)-J
0137 WRITE(102,3000) (ASTER,I=1,33)
0138 WRITE(102,1035) J1
0139 KKK=ND1+1
0140 NN=KKK+1
0141 DO 210 I=1,KKK
0142 J1=NN-I
0143 WRITE(102,1103) OLNAME,ENTRY(I),J1,U1(I),V1(I)
0144 WRITE(102,1101) KO,UR0OT(KD),VR0OT(KD),J1
0145 GO TO 1
0146 1020 FORMAT(1HOLTHI1,10X,4HREPEATED USE OF THE GREATEST COMMON
0147 DIVER AND 1,4X,4S,5H METHOD TO EXTRACT ROOTS AND MULTIPLOCITIES OF POLYNOMIA
0148 21S/11X,1BHPOLYNOMIAL NUMBER 1,12//)
0149 1025 FORMAT(1X,125HHO ROOTS OF MULTIPLOCITY #12//)
0150 1035 FORMAT(1X,874T THE FOLLOWING POLYNOMIAL GEX), CONTAINS ALL THE R
0151 1000S OF PIX WHICH HAVE MULTIPLOCITY #12//)
0152 1065 FORMAT(2X,5HROOTS (1,2,4H) = R23,16,3H + ,R23,16,2H I,8X,12,9K,2SHN
0153 10 INITIAL APPROXIMATIONS)
0154 1095 FORMAT(1X,9HNOT ALL ROOTS OF THE ABOVE POLYNOMIAL,6, WERE FOUN
TABLE H.III (Continued)

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>THE DEGREE OF $p(x)$ IS 12, THE COEFFICIENTS ARE:</td>
</tr>
<tr>
<td>1010</td>
<td>$p_0 = 23.16$, $p_1 = 32.16$</td>
</tr>
<tr>
<td>1030</td>
<td>THE ROOTS OF PIXI, THEIR MULTIPLICITIES, AND INITIAL APPROXIMATIONS</td>
</tr>
<tr>
<td>1100</td>
<td>$r_0 = 23.16$, $r_1 = 32.16$</td>
</tr>
<tr>
<td>1180</td>
<td>THE MAXIMUM NUMBER OF ITERATIONS, TEST FOR CONVERGENCE, AND RADIUS TO START SEARCH</td>
</tr>
<tr>
<td>1200</td>
<td>END</td>
</tr>
</tbody>
</table>

END
TABLE H.III (Continued)

SUBROUTINE PROD(UF, VF, MN, VG, MN, UM, VH)

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

DOUBLE PRECISION UM, VM, UF, VF, VG, UM, VH
DIMENSION UM(5L), VM(5L), UF(26), VF(26), VG(26)

MN=M+N
KK=MN*
DO 100 I=1, KKK
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 20
100 CONTINUE
RETURN
END
TABLE H.III. (Continued)

0001 SUBROUTINE QUAD(UA, VA, J, UROOT, VROOT)
C **************************************************************
C * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR MULTIPlicITIES *
C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE *
C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA. *
C **************************************************************
0002 DOUBLE PRECISION EPS1, EPS2, EPSLON, UROOT, VROOT, UA, VA, UDISC, VDISC, UD
0003 COMMON EPS1, EPS2, EPSLON, EPS4, IO2, MAX
0004 IF(N.GT.1) GO TO 10
0005 J=J+1
0006 BBB=UA(2)*UA(2)+VA(2)*VA(2)
0007 UROOT(J)=-(UA(1)*UA(2)+VA(1)*VA(2)) / BBB
0008 VROOT(J)=-(VA(1)*UA(2)-UA(1)*VA(2)) / BBB
0009 GO TO 100
0010 10 UDISC=UA(2)*UA(2)+VA(2)*VA(2)-4.0*(UA(3)*UA(1)-VA(3)*VA(1))
0011 VDISC=2.0*(UA(3)*UA(1)+VA(3)*VA(1))
0012 UD=2.0*UA(3)
0013 VD=2.0*VA(3)
0014 DDD=DSQRT(UDISC*UDISC+VDISC*VDISC)
0015 IF(DDD.LT.EPSLON) GO TO 20
0016 CALL COMSQRT(UDISC, VDISC, UTEMP, VTEMP)
0017 BBB=UD*UD+VD*VD
0018 UROOT(J)=(-UA(2)+UTEMP)*UD-VA(2)*VD)/BBB
0019 VROOT(J)=(-VA(2)+UD+UA(2)*VTEMP)/BBB
0020 UROOT(J+2)=(1.0-VA(2)+VTEMP)*UD-(-UA(2)+UTEMP)/BBB
0021 VROOT(J+2)=(1.0-UA(2)+UTEMP)*UD-(-VA(2)+VTEMP)/BBB
0022 GO TO 100
0023 20 J=J+1
0024 GO TO 100
0025 100 RETURN
0026 END
TABLE H.III (Continued)

0001 SUBROUTINE DERIV(IN,UP,VP,M,UA,VA)
0002 C ***********************************************
0003 C * GIVEN A POLYNOMIAL PIXI, SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF *
0004 C * ITS DERIVATIVE P'(X). *
0005 C ***********************************************
0006 C DOUBLE PRECISION UP,VP,UA,VA,AAA
0007 DIMENSION UP(26),VP(26),UA(26),VA(26)
0008 KKK=N+1
0009 DO 10 I=2,KKK
0010 AAA=I-1
0011 UA(I-1)=AAA*UP(I)
0012 VA(I-1)=AAA*VP(I)
0013 M=N-1
0014 RETURN
0015 END
SUBROUTINE GCODNUR,VR,MUS,MIoUSStVSSI

C

* GCD COMPUTES THE GREATEST COMMON DIVISOR OF P(X) AND Q(X).

DOUBLE PRECISION USSS,VSSSSS

DOUBLE PRECISION URVRUSVSUSSVSSURRtRRRUDVDUTVTEPSLONEP

COMMON EPSLONEPS2,EPS3,EPS4#102,MAX

MI=M

KKK=N+1

DO I=1,KKK

URR(I)=UR(I)

20 VRR(I)=VR(I)

KKK=M+1

DO I=1,KKK

USSI)=US(I)

25 VSS(I)=VS(I)

BBB=USS(M+1)*USSIM11+IVSS(MI+1)*VSSIMI+L))/BBB

UD=(URRIN1+1)*USSEMI+1)+VRRINI+1)*VSSIMI+L))/BBB

VD=(USS(IMI1)*VRRINI1)I-URR(NI1)*VSS(MI+ll

KKK=N1-MI

DO 40 I=KKK,N1

UTII)=URR(I)-(UDUSSII-NI+M1)-VD*VSSII-N1+MI))

40 VT(II=VRR(I)-IUD*VSSII-NIMIII

IF(MI.EQ.N11 GO TO 70

KKK=N1-MI

DO 60 I=1,KKK

UT(I)=URRI

60 VRR(I)=VRRI

70 DO 90 I=I,Mt

BBB=USS(M1+1)*USS(M1*If+VSS(IMI1)*VSS(IMI+I

USSSSS=IUSS(III)*USSIM1+11+VSS(I3*VSS(M1+1))/BBB

VSSSSS=IVSSII)*USSIM1+11-USS(II*VSSIMItnlIBBB

USS(I)=USSSSS

VSS(I)=VSSSSS

USS(MI1I)=I.O

VSS(MI+1)=O.O

GO TO 200

100 K=NI-1

IF(K,EQ.0) GO TO 170

IF(KLT.M1) GO TO 140

K=K+1

DO 130 J=1,KK

URR(J)=UT(J)

130 VRR(J)=VT(J)

NI=K

GO TO 30
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0050</td>
<td>140 KKK=K+1</td>
</tr>
<tr>
<td>0051</td>
<td>GO 150 J=1,KKK</td>
</tr>
<tr>
<td>0052</td>
<td>URR(J)=USS(J)</td>
</tr>
<tr>
<td>0053</td>
<td>VRR(J)=VSS(J)</td>
</tr>
<tr>
<td>0054</td>
<td>USS(J)=UT(J)</td>
</tr>
<tr>
<td>0055</td>
<td>VSS(J)=VT(J)</td>
</tr>
<tr>
<td>0056</td>
<td>KKK=K+2</td>
</tr>
<tr>
<td>0057</td>
<td>NNN=ML+1</td>
</tr>
<tr>
<td>0058</td>
<td>DO 160 J=KKK,NNN</td>
</tr>
<tr>
<td>0059</td>
<td>URR(J)=USS(J)</td>
</tr>
<tr>
<td>0060</td>
<td>VRR(J)=VSS(J)</td>
</tr>
<tr>
<td>0061</td>
<td>N1=M1</td>
</tr>
<tr>
<td>0062</td>
<td>M1=K</td>
</tr>
<tr>
<td>0063</td>
<td>GO TO 30</td>
</tr>
<tr>
<td>0064</td>
<td>170 USS(1)=1.0</td>
</tr>
<tr>
<td>0065</td>
<td>VSS(1)=0.0</td>
</tr>
<tr>
<td>0066</td>
<td>M1=0</td>
</tr>
<tr>
<td>0067</td>
<td>200 RETURN</td>
</tr>
<tr>
<td>0068</td>
<td>END</td>
</tr>
</tbody>
</table>
**TABLE H.III (Continued)**

```plaintext
SUBROUTINE DIVIDE(N.UP,VP,UD,VO,K,UQ,VQ)  
C ***************************************************************  
C * GIVEN TWO POLYNOMIALS F(X) AND G(X), SUBROUTINE DIVIDE COMPUTES THE  
C * QUOTIENT POLYNOMIAL H(X) = F(X)/G(X).                       
C ***************************************************************  

DOUBLE PRECISION UP,VP,UD,VO,UQ,VQ,UTERM,VTERM,UDUMMY  
DIMENSION UP(26),VP(26),UD(26),VD(26),UQ(26)  

K=N-M  
UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)  
UQ(K+1)=UP(N+1)*UD(M+1)+VP(N+1)*VD(M+1)/UDUMMY  
VQ(K+1)=VP(N+1)*UD(M+1)-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  
GO TO 40  
UQ(KK)=UTERM*UD(M+1)-VQ(KK)*VD(M+1)  
VTERM=VTERM*UQ(KK)+VQ(VTERM)*UD(M+1)  
UDUMMY=UD(M+1)*UD(M+1)+VD(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
```

```markdown
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE DIVIDE(N.UP,VP,UD,VO,K,UQ,VQ)</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION UP,VP,UD,VO,UQ,VQ,UTERM,VTERM,UDUMMY</td>
</tr>
<tr>
<td>0003</td>
<td>DIMENSION UP(26),VP(26),UD(26),VD(26),UQ(26)</td>
</tr>
<tr>
<td>0004</td>
<td>K=N-M</td>
</tr>
<tr>
<td>0005</td>
<td>UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)</td>
</tr>
<tr>
<td>0006</td>
<td>UQ(K+1)=UP(N+1)*UD(M+1)+VP(N+1)*VD(M+1)/UDUMMY</td>
</tr>
<tr>
<td>0007</td>
<td>VQ(K+1)=VP(N+1)*UD(M+1)-UP(N+1)*VD(M+1)/UDUMMY</td>
</tr>
<tr>
<td>0008</td>
<td>IF(K.EQ.0) GO TO 100</td>
</tr>
<tr>
<td>0009</td>
<td>J=-1</td>
</tr>
<tr>
<td>0010</td>
<td>DO 50 I=1,K</td>
</tr>
<tr>
<td>0011</td>
<td>J=J+1</td>
</tr>
<tr>
<td>0012</td>
<td>UTERM=UP(N-J)</td>
</tr>
<tr>
<td>0013</td>
<td>VTERM=VP(N-J)</td>
</tr>
<tr>
<td>0014</td>
<td>KK=K+1</td>
</tr>
<tr>
<td>0015</td>
<td>NNN=M-J</td>
</tr>
<tr>
<td>0016</td>
<td>DO 40 MI=NNN,M</td>
</tr>
<tr>
<td>0017</td>
<td>IF(KK.GT.1) GO TO 10</td>
</tr>
<tr>
<td>0018</td>
<td>GO TO 45</td>
</tr>
<tr>
<td>0019</td>
<td>10 IF(M.GE.11) GO TO 20</td>
</tr>
<tr>
<td>0020</td>
<td>20 UTERM=UTERM-UQ(KK)*UD(M+1)-VQ(KK)*VD(M+1)</td>
</tr>
<tr>
<td>0021</td>
<td>VTERM=VTERM-UQ(VTERM)+VQ(KK)*UD(M+1)</td>
</tr>
<tr>
<td>0022</td>
<td>40 KD=KK-1</td>
</tr>
<tr>
<td>0023</td>
<td>45 UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)</td>
</tr>
<tr>
<td>0024</td>
<td>UQ(K+1)=UTERM<em>UD(M+1)-VTERM</em>VD(M+1)/UDUMMY</td>
</tr>
<tr>
<td>0025</td>
<td>VQ(K+1)=VTERM<em>UD(M+1)-UTERM</em>VD(M+1)/UDUMMY</td>
</tr>
<tr>
<td>0026</td>
<td>100 RETURN</td>
</tr>
<tr>
<td>0027</td>
<td>END</td>
</tr>
</tbody>
</table>
```
SUBROUTINE CONSQTIUXVXUY,VY)

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

DOUBLE PRECISION UX,VX,UY,VY,DUMMY,AA,AB

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

IF(VX)
10 UY=AAA
VY=-1.0*BBB
GO TO 100

20 IF(UX) 40,50,60
30 UY=AAA
VY=BBB
GO TO 100

40 DUMMY=DABS(UX)
UY=0.0
VY=DSQRT(DUMMY)
GO TO 100

50 UY=0.0
VY=0.0

60 DUMMY=DABS(UX)
UY=DSQRT(DUMMY)

100 RETURN

END
**TABLE H. III (Continued)**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td><strong>SUBROUTINE CALCUX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3.</strong></td>
</tr>
<tr>
<td>0002</td>
<td><strong>DOUBLE PRECISION ARG1,ARG2</strong></td>
</tr>
<tr>
<td>0003</td>
<td><strong>DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,</strong></td>
</tr>
<tr>
<td>0004</td>
<td><strong>U1,V1,U2,V2,U3,V3,UPX1,VPX1,</strong></td>
</tr>
<tr>
<td>0005</td>
<td><strong>U3=UX3-UX2</strong></td>
</tr>
<tr>
<td>0006</td>
<td><strong>V3=VX3-VX2</strong></td>
</tr>
<tr>
<td>0007</td>
<td><strong>U2=UX2-UX1</strong></td>
</tr>
<tr>
<td>0008</td>
<td><strong>V2=VX2-VX1</strong></td>
</tr>
<tr>
<td>0009</td>
<td><strong>BBB=U2<em>U2+V2</em>V2</strong></td>
</tr>
<tr>
<td>0010</td>
<td><strong>U03=(U3<em>U3-U2</em>U2)/BBB</strong></td>
</tr>
<tr>
<td>0011</td>
<td><strong>VQ3=(V3<em>V3-U2</em>U2)/BBB</strong></td>
</tr>
<tr>
<td>0012</td>
<td><strong>UDDD=1.0*U03</strong></td>
</tr>
<tr>
<td>0013</td>
<td><strong>VDDD=1.0*VQ3</strong></td>
</tr>
<tr>
<td>0014</td>
<td><strong>UAAA=2.0*UDDD</strong></td>
</tr>
<tr>
<td>0015</td>
<td><strong>VAAA=2.0*VDDD</strong></td>
</tr>
<tr>
<td>0016</td>
<td><strong>UAAA=UAAA+4.0</strong></td>
</tr>
<tr>
<td>0017</td>
<td><strong>UBBB=UDDD<em>UDDD-VDDD</em>VDDD</strong></td>
</tr>
<tr>
<td>0018</td>
<td><strong>VBBB=VDDD<em>VDDD+UDDD</em>UDDD</strong></td>
</tr>
<tr>
<td>0019</td>
<td><strong>UCCC=UQ3<em>UQ3-VQ3</em>VQ3</strong></td>
</tr>
<tr>
<td>0020</td>
<td><strong>VCCC=VQ3<em>UQ3+UQ3</em>VQ3</strong></td>
</tr>
<tr>
<td>0021</td>
<td><strong>UH=((UAAA<em>UPX3-VAAA</em>VPX3)-(UBBB<em>UPX2-VBBB</em>VPX2)+(UCCC<em>UPX1-VCCC</em>VPX1)</strong></td>
</tr>
<tr>
<td>0022</td>
<td><strong>VB=(VAAA<em>UPX3+UAAA</em>VPX3)-(VBBB<em>UPX2+VBBB</em>VPX2)+(VCCC<em>UPX1+VCCC</em>VPX1)</strong></td>
</tr>
<tr>
<td>0023</td>
<td><strong>UC=UDDD<em>UPX3-VDDD</em>VPX3</strong></td>
</tr>
<tr>
<td>0024</td>
<td><strong>VC=VDDD<em>VPX3+UDDD</em>UPX3</strong></td>
</tr>
<tr>
<td>0025</td>
<td><em><em>UDISC=(UH<em>UH-VB</em>VB-(4.0</em>(UC-VC))</em>*</td>
</tr>
<tr>
<td>0026</td>
<td><em><em>VDISC=(VB<em>VB-UH</em>UH-(4.0</em>(VC-UC))</em>*</td>
</tr>
<tr>
<td>0027</td>
<td><strong>AAA=ABS(THETA)</strong></td>
</tr>
<tr>
<td>0028</td>
<td><strong>BBB=ABS(BBB)</strong></td>
</tr>
<tr>
<td>0029</td>
<td><strong>CC=ABS(CC)</strong></td>
</tr>
<tr>
<td>0030</td>
<td><strong>AAA=ABS(AA)</strong></td>
</tr>
<tr>
<td>0031</td>
<td><strong>BBB=ABS(BB)</strong></td>
</tr>
<tr>
<td>0032</td>
<td><strong>CC=ABS(CC)</strong></td>
</tr>
<tr>
<td>0033</td>
<td><strong>THETA=0.0</strong></td>
</tr>
<tr>
<td>0034</td>
<td><strong>GO TO 9</strong></td>
</tr>
<tr>
<td>0035</td>
<td><strong>THETA=90.0</strong></td>
</tr>
<tr>
<td>0036</td>
<td><strong>GO TO 9</strong></td>
</tr>
<tr>
<td>0037</td>
<td><strong>THETA=90.0</strong></td>
</tr>
<tr>
<td>0038</td>
<td><strong>GO TO 9</strong></td>
</tr>
<tr>
<td>0039</td>
<td><strong>THETA=90.0</strong></td>
</tr>
<tr>
<td>0040</td>
<td><strong>THETA=90.0</strong></td>
</tr>
<tr>
<td>0041</td>
<td><strong>THETA=90.0</strong></td>
</tr>
<tr>
<td>0042</td>
<td><strong>THETA=90.0</strong></td>
</tr>
<tr>
<td>0043</td>
<td><strong>THETA=90.0</strong></td>
</tr>
<tr>
<td>0044</td>
<td><strong>THETA=90.0</strong></td>
</tr>
</tbody>
</table>
TABLE H.III (Continued)

```
0045  ARG1=UDEN1*UDEN1+VDEN1+VDEN1
0046  ARG2=UDEN2*UDEN2+VDEN2*VDEN2
0047  AAA=DSQRT(ARG1)
0048  BBB=DSQRT(ARG2)
0049  IF(AAA.LT.BBB) GO TO 10
0050  IF(AAA.EQ.0.0) GO TO 60
0051  UAAA=-2.0*UC
0052  VAAA=-2.0*VC
0053  UQ4=(UAAA*UDEN1+VAAA*VDEN1)/ARG1
0054  VQ4=(VAAA*UDEN1-UAAA*VDEN1)/ARG1
0055  GO TO 50
0056  10 IF(BBB.EQ.0.0) GO TO 60
0057  UAAA=-2.0*UC
0058  VAAA=-2.0*VC
0059  UQ4=(UAAA*UDEN2+VAAA*VDEN2)/ARG2
0060  VQ4=(VAAA*UDEN2-UAAA*VDEN2)/ARG2
0061  GO TO 50
0062  50 UX4=UX3+(UH3*UQ4-VH3*VQ4)
0063  VX4=VX3+VH3*UQ4+UH3*VQ4
0064  RETURN
0065  60 UQ4=1.0
0066  VQ4=0.0
0067  GO TO 50
0068  END
```
TABLE H. III (Continued)

```
0001 SUBROUTINE MULLER(NP, UAP, VA, NAPP, UAPP, VAPP, NROOT, UROOT, VROOT, IROOT,
  ISTART, XEND, NORDER, URAPP, VRAPP)
C******************************************************************************
C  * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLECITIES OF A
C  * POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE
C  * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE QUADRATIC
C  * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
C  * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
C******************************************************************************
0002 DOUBLE PRECISION UPX1, UPX2, UPX3, UROOT, VRROOT, UX1, VX1, UAPP, VAPP
  , UAPP1, UAPP2, UAPP3, UAPP4, UAPP5, UAPP6, UAPP7, UAPP8, UAPP9,
  VXAPP, VXAPP1, VXAPP2, VXAPP3, VXAPP4, VXAPP5, VXAPP6, VXAPP7,
  VXAPP8, VXAPP9
0003 DIMENSION UROOT(25), VRROOT(25), MULTIPLS(25), UAPP(25, 3), VRAPP(25, 3),
  UWORK(25), VWORK(25), UAPPLH(25, 3), VRAPPLH(25, 3), VRAPP(25, 3)
0004 LOGICAL COWN
0005 COMMON EPSM, EPSD, EPSRT, ITOL, MAX
0006 DATA PNAME, DNAME/2HP1, 2HD1/
0007 EPSD=0.0000  
0008 EPSRT=0.009  
0009 NROOT=0  
0010 IROOT=0  
0011 IPATH=1  
0012 NOMULT=0  
0013 NALTER=0  
0014 ITIME=0  
0015 [APP=1  
0016 ITER=1  
0017 IF (NAPP.NE.0) GO TO 10
0018 NAPP=NP
0019 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
0020 GO TO 27
0021 10 DO 25 I=1,NAPP
0022 UAPP(I, 1)=0.9*UAPP(I, 2)
0023 VAPP(I, 1)=0.9*VAPP(I, 2)
0024 UAPP(I, 3)=1.4*UAPP(I, 2)
0025 VAPP(I, 3)=1.4*VAPP(I, 2)
0026 25 KKK=KKK+1
0027 GO 30 T=1,KKK
0028 UWORK(I)=UAT(I)
0029 30 VWORK(I)=VAT(I)
0030 NWORK=NP
0031 40 UX=UAPP(IAPP, 1)
0032 VX=VAPP(IAPP, 1)
0033 UX2=UAPP(IAPP, 2)
0034 VX2=VAPP(IAPP, 2)
0035 UX3=UAPP(IAPP, 3)
0036 VX3=VAPP(IAPP, 3)
0037 CALL HORNEX(UREAL, UWORK, UX1, VX1, UX2, VX2, UX3, VX3, UAPPLH, VAPPLH)
0038 CALL HORNEX(UREAL, UWORK, UX1, VX1, UX2, VX2, UX3, VX3, VAPPLH)
0039 50 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UX2, VX2, UPX2, VPX2)
0040 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UX2, VX2, UPX2, VPX2)
0041 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UX2, VX2, UPX2, VPX2)
0042 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UX2, VX2, UPX2, VPX2)
0043 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UX2, VX2, UPX2, VPX2)
```

TABLE H.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0044</td>
<td>IF(APX3.EQ.0.0) GO TO 70</td>
</tr>
<tr>
<td>0045</td>
<td>QQQ=APX4/ABPX3</td>
</tr>
<tr>
<td>0046</td>
<td>IF(QQQ.LE.10.) GO TO 70</td>
</tr>
<tr>
<td>0047</td>
<td>UQ4=0.5*UQ4</td>
</tr>
<tr>
<td>0048</td>
<td>VQ4=0.5*VQ4</td>
</tr>
<tr>
<td>0049</td>
<td>UX4=UX3*(UH3<em>UQ4-VH3</em>VQ4)</td>
</tr>
<tr>
<td>0050</td>
<td>VX4=VX3*(UH3<em>UQ4+VH3</em>VQ4)</td>
</tr>
<tr>
<td>0051</td>
<td>GO TO 60</td>
</tr>
<tr>
<td>0052</td>
<td>CALL TEST(UX3,VX3,UX4,VX4,CONV)</td>
</tr>
<tr>
<td>0053</td>
<td>IF(CONV) GO TO 120</td>
</tr>
<tr>
<td>0054</td>
<td>IF(ITER.LT.MAX) GO TO 110</td>
</tr>
<tr>
<td>0055</td>
<td>CALL ALTER(UAPP(IAPP),VAPP(IAPP,1),UROOT,VROOT,NROOT,UAPP,VAPP,RROOT,MULT)</td>
</tr>
<tr>
<td>0056</td>
<td>IF(NALTER.GT.5) GO TO 75</td>
</tr>
<tr>
<td>0057</td>
<td>ITER=1</td>
</tr>
<tr>
<td>0058</td>
<td>GO TO 40</td>
</tr>
<tr>
<td>0059</td>
<td>75 IF(IAPP.LT.NAPP) GO TO 100</td>
</tr>
<tr>
<td>0060</td>
<td>IF(XEND.EQ.0.0) GO TO 77</td>
</tr>
<tr>
<td>0061</td>
<td>IF(XSTART.GT.XEND) GO TO 77</td>
</tr>
<tr>
<td>0062</td>
<td>NAPP=NP</td>
</tr>
<tr>
<td>0063</td>
<td>CALL GENAPP(UAPP,VAPP,NAPP,XSTART)</td>
</tr>
<tr>
<td>0064</td>
<td>IAPP=0</td>
</tr>
<tr>
<td>0065</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0066</td>
<td>77 WRITE(102,1040)</td>
</tr>
<tr>
<td>0067</td>
<td>1040 KKK=NROOT+1</td>
</tr>
<tr>
<td>0068</td>
<td>WRITE(102,1035) (DNAME,J,UWORK(I,J),VWORK(J),J=1,1)</td>
</tr>
<tr>
<td>0069</td>
<td>1035 IF(IPATH.EQ.11) GO TO 62</td>
</tr>
<tr>
<td>0070</td>
<td>61 IF(PATH.EQ.2)</td>
</tr>
<tr>
<td>0071</td>
<td>CALL BETTER(UA,VA,NP,UROOT,VROOT,NROOT,UAPP,VAPP,RROOT,MULT)</td>
</tr>
<tr>
<td>0072</td>
<td>RETURN</td>
</tr>
<tr>
<td>0073</td>
<td>81 IF(NROOT.EQ.0) GO TO 90</td>
</tr>
<tr>
<td>0074</td>
<td>80 IF(NROOT.EQ.0) GO TO 85</td>
</tr>
<tr>
<td>0075</td>
<td>85 WRITE(102,1041)</td>
</tr>
<tr>
<td>0076</td>
<td>1041 DO 55 I=1,1,RROOTD</td>
</tr>
<tr>
<td>0077</td>
<td>55 WRITE(102,1045) (UROOT(I),VROOT(I),UROOT(I),2),VRAPP(I)</td>
</tr>
<tr>
<td>0078</td>
<td>1045 IF(IROOT.LT.NROOT) GO TO 85</td>
</tr>
<tr>
<td>0079</td>
<td>80 GO TO 83</td>
</tr>
<tr>
<td>0080</td>
<td>83 S=1</td>
</tr>
<tr>
<td>0081</td>
<td>90 IF(NROOT.EQ.0) GO TO 81</td>
</tr>
<tr>
<td>0082</td>
<td>81 WRITE(102,1040)</td>
</tr>
<tr>
<td>0083</td>
<td>1040 IF(PATH.EQ.11) GO TO 81</td>
</tr>
<tr>
<td>0084</td>
<td>RETURN</td>
</tr>
<tr>
<td>0085</td>
<td>90 WRITE(102,1040)</td>
</tr>
<tr>
<td>0086</td>
<td>RETURN</td>
</tr>
<tr>
<td>0087</td>
<td>100 IAPP=IAPP+1</td>
</tr>
<tr>
<td>0088</td>
<td>RETURN</td>
</tr>
<tr>
<td>0089</td>
<td>NALTER=0</td>
</tr>
<tr>
<td>0090</td>
<td>GO TO 40</td>
</tr>
<tr>
<td>0091</td>
<td>120 NROOT=NROOT+1</td>
</tr>
<tr>
<td>0092</td>
<td>RETURN</td>
</tr>
<tr>
<td>0093</td>
<td>RROOT=RROOT+1</td>
</tr>
<tr>
<td>0094</td>
<td>NROOT=NROOT+1</td>
</tr>
<tr>
<td>0095</td>
<td>UROOT(UROOT)=UX4</td>
</tr>
<tr>
<td>0096</td>
<td>VROOT(VROOT)=VX4</td>
</tr>
<tr>
<td>0097</td>
<td>URAPP(UROOT,1)=UAPP(IAPP,1)</td>
</tr>
<tr>
<td>0098</td>
<td>VRAPP(VROOT,1)=VAPP(IAPP,1)</td>
</tr>
<tr>
<td>0099</td>
<td>URAPP(UROOT,2)=UAPP(IAPP,2)</td>
</tr>
<tr>
<td>1000</td>
<td>VRAPP(VROOT,2)=VAPP(IAPP,2)</td>
</tr>
</tbody>
</table>
TABLE H.III (Continued)

0101  
0102  VRAPP(NROTT=IAPP,3)=VRAPP(NROTT=IAPP,3)
0103  125 IF(NMULT.LT.NP) GO TO 130
0104  130 CALL HORNER(NWORK,UX4,VX4,UB,UB,VPX4,VPX4)
0105  NWORK=NWORK-1
0106  KKK=NWORK+1
0107  DD 140 (I=1, KKK
0108  UWORK(I)=UB(I)
0109  140 VWORK(I)=VB(I)
0110  CALL HORNER(NWORK,UX4,VX4,UB,UB,VPX4,VPX4)
0111  CCC=DSORT(UPX4*UPX4+VPX4*VPX4)
0112  IF(CCC.LT.EPSM) GO TO 150
0113  IF(NWORK.GT.2) GO TO 75
0114  IROOT=NROOT
0115  KKK=NWORK+1
0116  DO 145 (I=KIKK+1-1)
0117  UB(I)=UWORK(KIKK+1-I)
0118  VB(I)=VWORK(KIKK+1-I)
0119  CALL QUAO(NWORK,UB,VB,NROOT,UROOT,VROOT)
0120  GO TO 80
0121  MULT(NROOT)=MULT(NROOT)+1
0122  NOMULT=NOMULT+1
0123  GO TO 125
0124  110 UX1=UX2
0126  VX1=VX2
0127  UX2=UX3
0128  VX2=VX3
0129  UX3=UX4
0130  VX3=VX4
0131  UX4=UX5
0132  VX4=VX5
0133  UPX1=UPX2
0134  VPX1=VPX2
0135  UPX2=UPX3
0136  VPX2=VPX3
0137  UPX3=UPX4
0138  VPX3=VPX4
0139  ITER=ITER+1
0140  GO TO 50
0141  1090 FORMAT(///,1X,52HYCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO
1ZEROS WERE FOUND//)
0142  1080 FORMAT(///IX,13HROOTS OF G(X),83X,21HINITIAL
1APPROXIMATION//)
0143  1070 FORMAT(/IX,13HROOTS FOR WHICH NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER ,12)
0144  1060 FORMAT(///X,52HZEROS FOUND FOR POLYNOMIAL NUMBER ,12)
0145  1050 FORMAT(///X,13HZEROS FOUND FOR POLYNOMIAL NUMBER ,12)
0146  1040 FORMAT(///X,13HZEROS FOUND FOR POLYNOMIAL NUMBER ,12)
0147  1030 FORMAT(///X,13HZEROS FOUND FOR POLYNOMIAL NUMBER ,12)
0148  1020 FORMAT(///X,13HZEROS FOUND FOR POLYNOMIAL NUMBER ,12)
0149  1010 FORMAT(///X,13HZEROS FOUND FOR POLYNOMIAL NUMBER ,12)
0150  1000 FORMAT(///X,13HZEROS FOUND FOR POLYNOMIAL NUMBER ,12)

END
**TABLE H.III (Continued)**

```
0001 SUBROUTINE GENAPP(APPR, APP1, NAPP, XSTART)
0002 C
0003 C
0004 SUBROUTINE GENAPP GENERATES N INITIAL APPROXIMATIONS, WHERE N IS THE
0005 DEGREE OF THE ORIGINAL POLYNOMIAL.
0006 C
0007 C
0008******************************************************************************
0009 0001 DOUBLE PRECISION APPR, APP1, XSTART, EPS1, EPS2, EPS3, BETA
0010 0002 DOUBLE PRECISION EPSM
0011 0003 DIMENSION APPR(25,31), APP1(25,3)
0012 0004 COMMON EPSM, EPS1, EPS2, EPS3, IO2, MAX
0013 0005 IF (XSTART.EQ.0.0) XSTART=0.5
0014 0006 BETA=0.2617994
0015 0007 DO 10 I=1, NAPP
0016 0008 APPR(I,2)=XSTART*DCOS(BETA)
0017 0009 APP1(I,2)=XSTART*DSIN(BETA)
0018 0010 BETA=BETA+0.5235988
0019 0011 XSTART=XSTART+0.5
0020 10 DO 20 I=1, NAPP
0021 0021 APPR(I,1)=0.9*APPR(I,2)
0022 0022 APP1(I,1)=0.9*APP1(I,2)
0023 0023 APPR(I,3)=1.1*APPR(I,2)
0024 0024 APP1(I,3)=1.1*APP1(I,2)
0025 20 RETURN
0026 END
```
TABLE H. III (Continued)

00001 'SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,TIME)

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

00002 DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA

00004 COMMON EPS5, EPS1, EPS2, EPS3, IO2, MAX

00005 IF(TIME.NE.0) GO TO 5

00006 ITEMS=1

00007 WRITE(IO2,1010) MAX

00008 IF(NALTER.EQ.0) GO TO 10

00009 WRITE(IO2,1000) X1R,X1I,X2R,X2I,X3R,X3I

00100 GO TO 20

00011 10 R=SQRT(X1R*X2R+X2I*X2I)

00012 BETA=DATAN2(X2I,X2R)

00013 WRITE(IO2,1020) X1R,X1I,X2R,X2I,X3R,X3I

00014 20 NALTER=NALTER+1

00015 IF(NALTER.GT.5) RETURN

00016 GO TO (30,40,30,40,30,401,NALTER)

00017 30 X2R=-X2R

00018 X2I=-X2I

00019 GO TO 50

00020 40 BETA=BETA+1.0471976

00021 X2R=R*DCOS(BETA)

00022 X2I=R*DSIN(BETA)

00023 50 X1R=0.9*X2R

00024 X1I=0.9*X2I

00025 X3R=1.1*X2R

00026 X3I=1.1*X2I

00027 RETURN

00028 1000 FORMAT(1X,5HMX1 = D23,16,3H + ,D23,16,2H I,10K,22H ALTERED APPROXIM

00029 IATIONS/X1,5HMX2 = D23,16,3H + ,D23,16,2H I/I/X,5HMX3 = D23,16,3H +

00030 Z,D23,16,2H I/)

00031 1020 FORMAT(1H0,5HMX1 = D23,16,3H + ,D23,16,2H I,10K,22H INITIAL APPROXI

00032 MATIONS/X1,5HMX2 = D23,16,3H + ,D23,16,2H I/I/X,5HMX3 = D23,16,3H +

00033 Z,D23,16,2H I/)

00034 1030 FORMAT(1X,5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF

00035 TER, 15,12H ITERATIONS.//)

00036 END
TABLE H.III (Continued)

0001 SUBROUTINE BETTER(UA, VA, NP, UROOT, VROOT, NROOT, URAPP, VRAPP, IROOT, MUL)

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

0006 *------------------------------------------------------------------------------------------------------------------
0007 DOUBLE PRECISION UROOT, VRoot, UA, VA, URAPP, VRAPP, UX1, VX1, UX2, VX2, UX3
0008 VX3, UX3, VPX1, UPX2, VPX2, UPX3, VPX3, UB, V8, UROOT5, VROOT5, EPSRT, VX4, VX5
0009 UX4, URAPP, VRAPP, EPSQ, EPS, UQ4, VQ4, UH3, VH3

0003 DOUBLE PRECISION EPSM

0004 LOGICAL CONV

0005 DIMENSION UROOT(253), VRoot(253), UA(26), VA(26), URAPP(253, 3), VRAPP(253, 3)
0006 UX(126), VX(126), UROOT5(253), VROOT5(253), URAPP(253, 3), VRAPP(253, 3), MULT
0007 (253)

0008 COMMON EPSM, EPS, EPSQ, EPSRT, IO2, MAX

0009 IF(NROOT.LE.1) RETURN

0010 L=0

0011 GO TO 10 +1, NROOT

0012 URAPP(1, 1) = UROOT(1) * EPSRT

0013 UBAPP(1, 2) = VR apparatus(1)

0014 UBAPP(1, 3) = UROOT(1) * (2, 0 - EPSRT)

0015 10 URAPP(1, 1) = VROOT(1) * (2, 0 - EPSRT)

0016 DO 100 J+1, NROOT

0017 UX1 = URAPP(1, 1)

0018 VX1 = VRAPP(J, 1)

0019 UX2 = VRAPP(J, 2)

0020 VX2 = VRAPP(J, 3)

0021 UX3 = VRAPP(J, 3)

0022 VX3 = VRAPP(J, 3)

0023 IF(J+1) RETURN

0024 CALL HORNER(NP, UA, VA, UX1, VX1, UB, V8, UPX1, VPX1)

0025 CALL HORNER(NP, UA, VA, UX2, VX2, UB, V8, UPX2, VPX2)

0026 20 CALL HORNER(NP, UA, VA, UX3, VX3, UB, V8, UPX3, VPX3)

0027 CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UX

0028 14, VX4, UQ4, VQ4, UH3, VH3)

0029 30 CALL TEST(UX3, VX3, UX4, VX4, CONV)

0030 IF(CONV) GO TO 40

0031 IF(J+1, LT, MAX) GO TO 40

0032 WRITE(102, 1000) J, UROOT(J, 1), UROOT(J, 2), MAX

0033 IF(J+1, LT, IROOT) GO TO 33

0034 IF(J+1, EQ, IROOT) GO TO 35

0035 GO TO 100

0036 33 KKK = IROOT-1

0037 DO 34 K=KKK, KKK

0038 URAPP(K, 1) = URAPP(K+1, 1)

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

0040 URAPP(K, 2) = URAPP(K+1, 2)

0041 VRAPP(K, 2) = VRAPP(K+1, 2)

0042 URAPP(K, 3) = URAPP(K+1, 3)

0043 VRAPP(K, 3) = VRAPP(K+1, 3)

0044 35 IROOT = IROOT-1

0045 GO TO 100
TABLE H.III. (Continued)

`0046  40  UX1=UX2
0047   VX1=VX2
0048   UX2=UX3
0049   VX2=VX3
0050   UX3=UX4
0051   VX3=VX4
0052   UPX1=UPX2
0053   VPX1=VPX2
0054   UPX2=UPX3
0055   VPX2=VPX3
0056   ITER=ITER+1
0057   GO TO 20
0058   50  L=L+1
0059   VROOT5(L)=UX4
0060   VROOT5(L)=VX4
0061   100 CONTINUE
0062   IF(L.LE.0) GO TO 120
0063   DO 110 I=1,L
0064   UROOT(I)=UROOT(I)
0065   VROOT(I)=VROOT(I)
0066   NROOT=L
0067   RETURN
0068   120  NROOT=0
0069   RETURN
0070   1000 FORMAT(/42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT112.4H) = ,
0071       1023.16,3H + .023.16,2H I/24H DID NOT CONVERGE AFTER ,13.11H ITERAT
0072       2IONS)
0073   1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,023.16,3H + ,023.16,2H I/1)
0074   END`
TABLE H. III (Continued)

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

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

0002 DOUBLE PRECISION UX3, VX3, UX4, VX4, EPSRT, EPS0, EPS, AAA, UDUMMY, VDUMMY

0003 DOUBLE PRECISION EPSM

0004 LOGICAL CONV

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

0006 UDUMMY = UX4 - UX3

0007 VDUMMY = VX4 - UX3

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

0009 DENOM = DSQRT(UX4 + UX4 + VX4 + VX4)

0010 IF(DENOM.LT.EPS0) GO TO 20

0011 IF((AAA/DENOM).LT.EPS1) GO TO 10

0012 5 CONV = .FALSE.

0013 GO TO 100

0014 10 CONV = .TRUE.

0015 GO TO 100

0016 20 IF((AAA.LT.EPS0)) GO TO 10

0017 GO TO 5

0018 100 RETURN

0019 END

0021 SUBROUTINE HORNER(NA, UA, VA, UX, VX, UB, VB, UPX, VPX)

* HORNERS METHOD COMPUTES THE VALUE OF THE POLYNOMIAL F(X) AT A POINT D.
* SYNTHETIC DIVISION IS USED TO DEFLECT THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D).

0022 DOUBLE PRECISION UX, VX, UPX, VPX, UB, VB, UA, VA

0023 DIMENSION UA(26), VA(26), UB(26), VD(26)

0024 UB(1) = UA(1)

0025 VB(1) = VA(1)

0026 NUM = NA + 1

0027 DO 10 I=2, NUM

0028 UB(I) = UA(I) + UB(I-1) - UX - VB(I-1) - VX

0029 VB(I) = VA(I) + VB(I-1) - UX + UB(I-1) + VX

0030 UPX = VB(NUM)

0031 VPX = VB(NUM)

0032 RETURN

0033 END