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
Various classical methods exist for extracting the zeros of a polynomial
\[ P(X) = a_1X^N + a_2X^{N-1} + \ldots + a_{N+1} \]
where \( a_1 \neq 0 \) and \( a_1, a_2, \ldots, a_{N+1} \) are complex numbers, when \( N=1,2,3,4 \).
For polynomials of higher degree, iterative numerical methods must be used. In this material four iterative methods are presented for approximating the zeros of a polynomial using a digital computer. Newton's method and Muller's method are two well known iterative methods which are presented. They extract the zeros of a polynomial by generating a sequence of approximations converging to each zero. However, both of these methods are very unstable when used on a polynomial which has multiple zeros. That is, either they fail to converge to some or all of the zeros, or they converge to very bad approximations of the polynomial's zeros.

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

The above methods were all programmed in FORTRAN IV and comparisons in time and accuracy are given. These programs were executed on the
IBM 360/50 computer as well as the UNIVAC 1108 and the CDC 6600 computer.

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

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

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

<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>APPENDIX B - NEWTON'S METHOD</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>APPENDIX C - MULLER'S METHOD</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>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>APPENDIX E - G.C.D. - NEWTON'S METHOD</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 Repeated 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 rechecked and their accuracy possibly improved by using them as initial approximations with Newton's process applied to the full (undeflated) polynomial.

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

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

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

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

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

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

NEWTON'S METHOD

1. Derivation of the Algorithm

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

Consider the polynomial

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

(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

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

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

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

results. Rearranging terms produces
followed by

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

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

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

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

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

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

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

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

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

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

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

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, \( \alpha \), 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

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

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

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

\[
P(X) = (X - a) P_1(X) + R
\]

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

\[
P(X) = (X - a) P_1(X).
\]

Now \( P_1(a_1) = 0 \) implies that \( P(a_1) = 0 \). Hence, \( a_1 \) is a zero of \( P(X) \).

By the process of root finding and successive deflations, zeros \( a_0, a_1, \ldots, a_{N-1} \) of the deflated polynomials

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

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

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

2. Convergence of Newton's Method

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

**Theorem 2.4.** Let \( P(X) \) be a polynomial and let the following conditions be satisfied on the closed interval \([a, b] \):
1. \( P(a) P(b) < 0 \)
2. \( P'(X) \neq 0, X \in [a,b] \).
3. \( P''(X) \) is either \( \geq 0 \) or \( \leq 0 \) for all \( X \in [a,b] \).
4. If \( c \) denotes the endpoint of \([a,b]\) at which \( |P'(X)| \) is smaller, then \( |P(c)/P'(c)| < b - a \).

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 convergence can be found in [L, PP. 31-33].

3. Procedure for Newton's Method

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

1. Calculate a new approximation \( X_{n+1} \) by
   \[
   X_{n+1} = X_n - \frac{P(X_n)}{P'(X_n)}.\]
2. Test for convergence; that is, test
   \[
   \left|\frac{P(X_n)}{P'(X_n)}\right| |X_{n+1}| < \epsilon
   \]
   for some \( \epsilon \) chosen as small as desired.
3. If convergence is obtained, perform the following:
a. Save $X_{n+1}$ as the desired approximation to a zero of $P(X)$.
b. Deflate $P(X)$ using $X_{n+1}$.
c. Replace $P(X)$ by the deflated polynomial.
d. Return to step 1 with a new initial approximation.

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

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

4. Geometrical Interpretation of Newton's Method

A geometrical interpretation of Newton's method is given in Figure 2.1. $X_1$ is an approximation to the zero, $a$. $P'(X_1)$ is the slope of the line tangent to $P(X)$ at $X_1$. $X_{i+1}$ is the intersection of the tangent line with the x axis.

5. Determining Multiple Roots

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

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

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

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

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 \( P_1(\alpha) < \epsilon \) for some small number \( \epsilon \), \( \alpha \) is a root of \( P_1(X) \) and thus has multiplicity at least equal to two. \( P_1(X) \) is then deflated giving \( P_2(X) \). If \( P_2(\alpha) < \epsilon \), \( \alpha \) is of multiplicity at least three. This process is continued until a deflated polynomial \( P_k(X) \) is encountered such that either \( \text{deg. } P_k(X) = 0 \) or \( P_k(\alpha) > \epsilon \). \( \alpha \) is then a zero of multiplicity \( k+1 \).

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

MULLER'S METHOD

1. Derivation of the Algorithm

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

Consider the polynomial

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

(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, \( a \), 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 \( a \). 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

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

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

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

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

In order to simplify this expression, introduce the quantities

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

Then

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

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

(3-3)

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

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

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

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 \( n^{th} \) iteration. However, no general proof of convergence has been obtained for Muller's method. It has produced convergence in the majority of the cases tested.

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

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

2. Procedure for Muller's Method

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

1. Compute \( h_n, q_n, D_n, B_n, C_n, q_{n+1} \) as defined previously.
2. Compute the next approximation \( X_{n+1} \) by
   \[
   X_{n+1} = X_n + h_n q_{n+1}.
   \]
3. Test for convergence; that is, test
   \[
   \left| X_{n+1} - X_n \right| / \left| X_{n+1} \right| < \epsilon
   \]
   for some suitably small number \( \epsilon \).

4. If the test fails, return to step 1 with the last three approximations \( X_{n+1}, X_n, X_{n-1} \).
5. If the test passes, do the following:
   a. Save $X_{n+1}$ as the desired approximation to a zero.
   b. Deflate the current polynomial using $X_{n+1}$.
   c. Replace the current polynomial by the deflated polynomial.
   d. Return to step 1 with a new set of initial approximations.

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

3. Geometrical Interpretation of Muller's Method

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

4. Determining Multiple Roots

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

GREATEST COMMON DIVISOR METHOD

1. Derivation of the Algorithm

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

Consider the Nth degree polynomial

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

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

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

(4-1)

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

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

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

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

**Theorem 4.1.** Let \( P(X) \) be an \( N \)th degree polynomial having \( m \) distinct zeros \( a_1, a_2, \ldots, a_m \) of multiplicity \( e_1, e_2, \ldots, e_m \) respectively. Then the polynomial

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

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

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

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

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

Consider the polynomial \( H(X) \) obtained by dividing \( P(X) \) by its monic g.c.d., \( D(X) \).
\[ H(X) = \frac{P(X)}{D(X)} \]

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

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

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

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

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

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

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

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

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

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

Since a polynomial cannot have degree less than zero, the above process, in a finite number of steps (at most \( n_1 \)), results in polynomials \( R_{k-1}(X), S_k(X) \) and \( R_k(X) \) with \( \deg R_k(X) < \deg R_{k-1}(X) \) such that

\[ R_{k-1}(X) = R_k(X) S_k(X) + R_{k+1}(X) \]

and \( R_{k+1}(X) = 0 \).

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

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

The above theorem tells how to obtain the greatest common divisor of two polynomials. A machine oriented method is now developed for computing the sequence of \( R_j(X) \)'s. Beginning the sequence with \( R_0(X) \) and \( R_1(X) \), the polynomial \( R_{i+1}(X) \) of the sequence is derived from \( R_i(X) \)
and \( R_{i-1}(X) \) as follows: Let \( R_{i-1}(X) \) of degree \( N_{i-1} \) be given by

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

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

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

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

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

Then define \( T_1(X) \) by

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

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

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

\[
+ \ldots
\]

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

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

We consider three cases.

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

2. If \( T_1(X) \neq 0 \) and \( \text{deg. } T_1(X) < N_i \), then \( R_{i+1}(X) = T_1(X) \).
(3) If $T_1(X) \neq 0$ and deg. $T_1(X) = M_1 > N_1$, then define $U_2(X)$ by

$$U_2(X) = \left( \frac{t_{1,1}}{r_{i,1}} \right)^{M_1-N_1} X \cdot$$

where

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

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

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

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

$$+ \cdots$$

$$+ \left[ t_{1,M_1+1} - \left( \frac{t_{1,1}}{r_{i,1}} \right) r_{i,M_1+1} \right]$$

where $r_{i,j} = 0$ for $j > N_1+1$. We again consider the following three cases.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

REPEATED GREATEST COMMON DIVISOR METHOD

1. Derivation of the Algorithm

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

Let

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

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

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

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

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

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

Therefore, it follows that

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

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

It follows that

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

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

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

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

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

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

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

1. Given a polynomial, $P(X)$, in the form

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

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

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

$$D'_0(X) = b_1X^{M-1} + b_2X^{M-2} + \ldots + b_M.$$
where \( \deg D_0(X) = M, D_0(X) = d_1X^M + \ldots + d_{M+1}, \)
and \( b_1 = M d_1, b_2 = (M-1)d_2, \ldots, b_M = d_M. \)

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.

34
Muller's method also produces good results on this class of polynomials. The rate of convergence is, however, somewhat slower than Newton's method. This fact is especially significant when working with polynomials of high degree. The accuracy obtained by Muller's method is comparable to, but does not exceed that of Newton's method. In most cases, the accuracy of the two methods does not differ by more than one or two decimal places. Exhibit 6.2 shows results of Muller's method for the polynomial of Exhibit 6.1. As in Newton's method, convergence is produced for almost any initial approximation given.

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

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

2. Polynomials With Multiple Zeros

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

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

Note the relationship between polynomials #1 and #3.

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

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

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

1. Roots #2 and #3 are greatly improved by iterating on the original polynomial. Distinct roots are usually improved in this manner.
2. The time taken to solve this 6th degree equation with multiple roots is greater than the time taken by the same program to solve a 15th degree polynomial with all distinct roots (Exhibit 6.1).

3. Root #2 did not pass the convergence test after 200 iterations even though it was improved. This is probably due to the fact that the polynomial from which root 2 was extracted had only one multiple root but the original polynomial from which it was extracted the second time had two multiple roots; that is, the original polynomial is more ill-conditioned.

4. The accuracy of the roots before the attempt to improve accuracy is very poor. Root #2 is accurate to only three decimal places as compared to the 15 decimal places in Exhibit 6.1 for distinct roots. Root #3 is especially bad, the imaginary part being accurate to only one decimal place.

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

In many cases, Newton's method fails to converge altogether. Polynomial #3 could not be solved using Newton's method with a maximum
number of 200 iterations and a convergence requirement of $10^{-9}$.

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

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

The g.c.d. method is perfectly suited for polynomials with multiple zeros. All multiple zeros are removed leaving only a polynomial of class 1 (all distinct roots) to be solved. This indicates that best results should be obtained by using Newton's method as the supporting method, since Newton's method enjoys the advantage of speed over Muller's method for distinct zeros. This has indeed proved to be true. The accuracy of the zeros obtained decreases, somewhat, when the number of multiple zeros is increased. This is due to accuracy lost in computing the g.c.d. and the quotient polynomial and not as a result of the supporting method. It is easy to see that the accuracy of the g.c.d. method is best when the degree of the greatest common divisor of
P(X) and P'(X) is maximum. This is due to the fact that the error in the greatest common divisor is minimized in this case. The accuracy obtained using Newton's method and Muller's method as supporting methods is about the same. This is verified by Exhibits 6.11 - 6.14 (g.c.d. method with Newton) and Exhibits 6.15 - 6.18 (g.c.d. method with Muller).

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

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

Table 6.1 gives a comparison of the execution times of the six methods for polynomials #1 - #4.

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

It is clear from Table 6.1 that the g.c.d. and the repeated g.c.d. methods are much faster than both Newton's and Muller's method on

*These times are from execution runs on the IBM 360/50 WATFOR system.
polynomials with multiple zeros. Therefore, for polynomials with multiple zeros, the order in which the methods are recommended is as follows.

1. G.C.D. with Newton.
2. G.C.D. with Muller.
4. Repeated G.C.D. with Muller.
5. Muller.
THE COEFFICIENTS OF P(X) ARE

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.30000000000000000</td>
<td>-0.30000000000000000</td>
<td>0.30000000000000000</td>
<td>-0.30000000000000000</td>
<td>0.30000000000000000</td>
<td>-0.30000000000000000</td>
<td>0.30000000000000000</td>
<td>-0.30000000000000000</td>
<td>0.30000000000000000</td>
<td>-0.30000000000000000</td>
<td>0.30000000000000000</td>
<td>-0.30000000000000000</td>
<td>0.30000000000000000</td>
<td>-0.30000000000000000</td>
<td>0.30000000000000000</td>
<td>-0.30000000000000000</td>
</tr>
</tbody>
</table>

MULTIPLICATIONS

<table>
<thead>
<tr>
<th>ROOT(1)</th>
<th>ROOT(2)</th>
<th>ROOT(3)</th>
<th>ROOT(4)</th>
<th>ROOT(5)</th>
<th>ROOT(6)</th>
<th>ROOT(7)</th>
<th>ROOT(8)</th>
<th>ROOT(9)</th>
<th>ROOT(10)</th>
<th>ROOT(11)</th>
<th>ROOT(12)</th>
<th>ROOT(13)</th>
<th>ROOT(14)</th>
<th>ROOT(15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10000000000000000</td>
<td>-0.20000000000000000</td>
<td>0.30000000000000000</td>
<td>-0.40000000000000000</td>
<td>0.50000000000000000</td>
<td>-0.60000000000000000</td>
<td>0.70000000000000000</td>
<td>-0.80000000000000000</td>
<td>0.90000000000000000</td>
<td>-1.00000000000000000</td>
<td>1.10000000000000000</td>
<td>-1.20000000000000000</td>
<td>1.30000000000000000</td>
<td>-1.40000000000000000</td>
<td>1.50000000000000000</td>
</tr>
</tbody>
</table>

INITIAL APPROXIMATION

<table>
<thead>
<tr>
<th>ROOT(1)</th>
<th>ROOT(2)</th>
<th>ROOT(3)</th>
<th>ROOT(4)</th>
<th>ROOT(5)</th>
<th>ROOT(6)</th>
<th>ROOT(7)</th>
<th>ROOT(8)</th>
<th>ROOT(9)</th>
<th>ROOT(10)</th>
<th>ROOT(11)</th>
<th>ROOT(12)</th>
<th>ROOT(13)</th>
<th>ROOT(14)</th>
<th>ROOT(15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.12000000000000000</td>
<td>-0.24000000000000000</td>
<td>0.36000000000000000</td>
<td>-0.48000000000000000</td>
<td>0.60000000000000000</td>
<td>-0.72000000000000000</td>
<td>0.84000000000000000</td>
<td>-0.96000000000000000</td>
<td>1.08000000000000000</td>
<td>-1.20000000000000000</td>
<td>1.32000000000000000</td>
<td>-1.44000000000000000</td>
<td>1.56000000000000000</td>
<td>-1.68000000000000000</td>
<td>1.80000000000000000</td>
</tr>
</tbody>
</table>

EXHIBIT 6.1.
AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF \( p(x) \) ARE ROOTS OF \( p(x) \) MULTIPlicITIES INITIAL APPROXIMATION

| ROOT(1) | 0.3000000000000000D00 | 1 | 0.48294291158642790D00 + 0.1294095284438180D00 I |
| ROOT(2) | 0.2000000000000000D00 | 1 | 0.70710678118654760D00 + 0.70710678118654760D00 I |
| ROOT(3) | -0.2000000000000000D00 | 1 | 0.38890862677698160D00 - 0.38890862677698160D00 I |
| ROOT(4) | -0.1000000000000000D00 | 1 | 0.1000000000000000D00 + 0.1000000000000000D00 I |
| ROOT(5) | -0.1000000000000000D00 | 1 | 0.3333333333333333D00 + 0.3333333333333333D00 I |
| ROOT(6) | 0.1000000000000000D00 | 1 | 0.1000000000000000D00 - 0.1000000000000000D00 I |
| ROOT(7) | 0.1000000000000000D00 | 1 | 0.2000000000000000D00 - 0.2000000000000000D00 I |
| ROOT(8) | 0.1000000000000000D00 | 1 | 0.2000000000000000D00 + 0.2000000000000000D00 I |
| ROOT(9) | 0.1000000000000000D00 | 1 | 0.2000000000000000D00 - 0.2000000000000000D00 I |
| ROOT(10) | 0.2000000000000000D00 | 1 | 0.2000000000000000D00 - 0.2000000000000000D00 I |
| ROOT(11) | 0.3000000000000000D00 | 1 | 0.3000000000000000D00 + 0.3000000000000000D00 I |
| ROOT(12) | 0.4000000000000000D00 | 1 | 0.4000000000000000D00 - 0.4000000000000000D00 I |
| ROOT(13) | 0.5000000000000000D00 | 1 | 0.5000000000000000D00 - 0.5000000000000000D00 I |
| ROOT(14) | 0.6000000000000000D00 | 1 | 0.6000000000000000D00 - 0.6000000000000000D00 I |

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 $p(x)$ are

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3095</td>
<td>0.3561</td>
<td>0.2101</td>
<td>-0.6576</td>
<td>0.2836</td>
<td>-0.5118</td>
<td>0.7224</td>
<td>-0.1276</td>
<td>0.1167</td>
<td>0.3390</td>
<td>0.2203</td>
<td>0.0460</td>
<td>0.2436</td>
<td>0.0730</td>
<td>-0.1077</td>
<td>-0.3496</td>
<td></td>
</tr>
</tbody>
</table>

**Number of Initial Approximations Given:** 0

**Maximum Number of Iterations:** 200

**Test for Convergence:** 0.100-09

**Test for Multiplicities:** 0.100-01

**Radius to Start Search:** 0.000 00

**Radius to End Search:** 0.000 00

Before attempt to improve accuracy

**Roots of $p(x)$**

<table>
<thead>
<tr>
<th>Root</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3095</td>
<td>3</td>
<td>$\pm 0.9836987517559590$</td>
</tr>
<tr>
<td>0.3561</td>
<td>1</td>
<td>$\pm 0.781999999999500$</td>
</tr>
<tr>
<td>0.2101</td>
<td>1</td>
<td>$\pm 0.599999999999950$</td>
</tr>
<tr>
<td>-0.6576</td>
<td>1</td>
<td>$\pm 0.4106987517559590$</td>
</tr>
<tr>
<td>0.2836</td>
<td>1</td>
<td>$\pm 0.251699999999950$</td>
</tr>
<tr>
<td>-0.5118</td>
<td>1</td>
<td>$\pm 0.157799999999950$</td>
</tr>
<tr>
<td>0.7224</td>
<td>1</td>
<td>$\pm 0.076799999999950$</td>
</tr>
<tr>
<td>-0.1276</td>
<td>1</td>
<td>$\pm 0.046799999999950$</td>
</tr>
<tr>
<td>0.1167</td>
<td>1</td>
<td>$\pm 0.027799999999950$</td>
</tr>
<tr>
<td>0.3390</td>
<td>1</td>
<td>$\pm 0.019899999999950$</td>
</tr>
<tr>
<td>0.2203</td>
<td>1</td>
<td>$\pm 0.012799999999950$</td>
</tr>
<tr>
<td>-0.1077</td>
<td>1</td>
<td>$\pm 0.004699999999950$</td>
</tr>
<tr>
<td>-0.3496</td>
<td>1</td>
<td>$\pm 0.002799999999950$</td>
</tr>
</tbody>
</table>

**Exhibit 6.2.**
**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

\[
\begin{align*}
P_1 &= 0.1000000000000000D 01 + 0.D0000000000000000D 00 1 \\
P_2 &= -0.7000000000000001D 01 - 0.1050000000000000D 02 1 \\
P_3 &= -0.2200000000000000D 02 + 0.5800000000000001D 02 1 \\
P_4 &= 0.1710000000000000D 03 + 0.1500000000000000D 01 1 \\
P_5 &= -0.7300000000000000D 02 - 0.2510000000000000D 03 1 \\
P_6 &= -0.2280000000000000D 03 + 0.1040000000000000D 02 1 \\
P_7 &= 0.72000000000000010D 02 + 0.1040000000000000D 03 1
\end{align*}
\]

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 \( P(x) \)  

\[
\begin{align*}
\text{ROOT}(1) &= 0.9999998836125019D 00 + 0.2000000000000000D 01 1 \\
\text{ROOT}(2) &= 0.19999982582258460D 01 + 0.1995258258248460D 01 1 \\
\text{ROOT}(3) &= -0.9999999999999999D 00 \quad \text{SOLVED BY DIRECT METHOD}
\end{align*}
\]

IN THE ATTEMPT TO IMPROVE ACCURACY, \( \text{ROOT}(2) \) \( \text{DID NOT CONVERGE.} \)
THE PRESENT APPROXIMATION AFTER 200 ITERATIONS IS PRINTED BELOW.

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

ROOTS OF \( P(x) \)  

\[
\begin{align*}
\text{ROOT}(1) &= 0.9999998836125019D 00 + 0.2000000000000000D 01 1 \quad 2 0.4999998836125019D 00 + 0.1299999999999999D 01 1 \\
\text{ROOT}(2) &= 0.19999982582258460D 01 + 0.1995258258248460D 01 1 \quad 3 0.7071067853934460D 00 + 0.7071067853934460D 00 1 \\
\text{ROOT}(3) &= -0.9999999999999999D 00 \quad \text{SOLVED BY DIRECT METHOD}
\end{align*}
\]

Exhibit 6.3. Roots Are: \(2+2i \) (3), \(1+2i \) (2), \(-1+5i \)
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 2 OF DEGREE 15

THE COEFFICIENTS OF PI(1) ARE

| Pi | 1  | = 0.84000000000000000000 02 + 0.00000000000000000000 00 I |
| Pi | 2  | = 0.25571200000000000000 03 + 0.38040000000000000000 00 I |
| Pi | 3  | = 0.73235960000000000000 02 + 0.21876400000000000000 00 I |
| Pi | 4  | = 0.38555659460000000000 04 + 0.69668154650000000000 04 I |
| Pi | 5  | = 0.17393444440000000000 05 + 0.14204329728000000000 05 I |
| Pi | 6  | = 0.45479897200000000000 05 + 0.17668576400000000000 05 I |
| Pi | 7  | = 0.10223943213000000000 06 + 0.60306444232000000000 06 I |
| Pi | 8  | = 0.16427442005600000000 05 + 0.41376434000000000000 05 I |
| Pi | 9  | = 0.20366258842000000000 06 + 0.10359922787000000000 06 I |
| Pi10 | = 0.19710557000000000000 06 + 0.19299494030000000000 06 I |
| Pi11 | = 0.12749777105000000000 06 + 0.21713412274000000000 06 I |
| Pi12 | = 0.26149271060000000000 05 + 0.13258497256000000000 06 I |
| Pi13 | = 0.13294343439000000000 05 + 0.10351302695000000000 06 I |
| Pi14 | = 0.96590677474000000000 06 + 0.29985413160000000000 06 I |
| Pi15 | = 0.18358992000000000000 03 + 0.18275216600000000000 03 I |
| Pi16 | = 0.27556260000000000000 00 + 0.27556260000000000000 00 I |

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

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

<table>
<thead>
<tr>
<th>ROOTS OF PI(X)</th>
<th>MULTIPLEITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT 11</td>
<td>2</td>
<td>0.82839709115652670600 00 + 0.1959912843181700 00 I</td>
</tr>
<tr>
<td>ROOT 12</td>
<td>2</td>
<td>0.70710677552049750000 00 + 0.07071067755204975000 00 I</td>
</tr>
<tr>
<td>ROOT 13</td>
<td>1</td>
<td>0.14913452477563900000 00 + 0.14913452477563900000 00 I</td>
</tr>
<tr>
<td>ROOT 14</td>
<td>1</td>
<td>0.12046903490986300000 01 + 0.12046903490986300000 01 I</td>
</tr>
<tr>
<td>ROOT 15</td>
<td>1</td>
<td>0.33256756376700000000 00 + 0.33256756376700000000 00 I</td>
</tr>
<tr>
<td>ROOT 16</td>
<td>1</td>
<td>0.31345368639942000000 00 + 0.31345368639942000000 00 I</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 pi are:

<table>
<thead>
<tr>
<th>Pi 11</th>
<th>Pi 21</th>
<th>Pi 31</th>
<th>Pi 41</th>
<th>Pi 51</th>
<th>Pi 61</th>
<th>Pi 71</th>
<th>Pi 81</th>
<th>Pi 91</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.16000000000000000 01</td>
<td>-0.30000000000000000 02</td>
<td>-0.51700000000000000 03</td>
<td>0.15720000000000000 04</td>
<td>0.30750000000000000 05</td>
<td>-0.49320000000000000 06</td>
<td>-0.53575000000000000 07</td>
<td>0.16100000000000000 08</td>
<td>0.19800000000000000 09</td>
</tr>
</tbody>
</table>

Number of initial approximations given: 0
Maximum number of iterations: 200
Test for convergence: 0.100-05
Test for multiplicities: 0.100-01
Radius to start search: 0.300 00
Radius to end search: 0.000 00

Before the attempt to improve accuracy, the zeros of pi are:

<table>
<thead>
<tr>
<th>Root 11</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.499999926490441670 00</td>
<td>2</td>
<td>0.4829929011156562800 00 + 0.1299095284381870 00</td>
</tr>
<tr>
<td>Root 12</td>
<td>-0.49999687027753200 00</td>
<td>2</td>
</tr>
<tr>
<td>Root 31</td>
<td>0.19166664822744220 01</td>
<td>1</td>
</tr>
<tr>
<td>Root 41</td>
<td>0.2031726866161618200 01</td>
<td>1</td>
</tr>
<tr>
<td>Root 51</td>
<td>0.23569406410039210 01</td>
<td>1</td>
</tr>
<tr>
<td>Root 61</td>
<td>-0.101544276975570 01</td>
<td>1</td>
</tr>
</tbody>
</table>

After the attempt to improve accuracy, the zeros of pi are:

<table>
<thead>
<tr>
<th>Root 11</th>
<th>Multiplicities</th>
<th>Initial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.49999992639397710 00</td>
<td>2</td>
<td>0.4829929011156562800 00 + 0.1299095284381870 00</td>
</tr>
<tr>
<td>Root 21</td>
<td>-0.49999687017753200 00</td>
<td>2</td>
</tr>
<tr>
<td>Root 31</td>
<td>0.19166664822744220 01</td>
<td>1</td>
</tr>
<tr>
<td>Root 41</td>
<td>0.2031726866161618200 01</td>
<td>1</td>
</tr>
<tr>
<td>Root 51</td>
<td>0.23569406410039210 01</td>
<td>1</td>
</tr>
<tr>
<td>Root 61</td>
<td>-0.101544276975570 01</td>
<td>1</td>
</tr>
</tbody>
</table>

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 PI(X) ARE

\[ P_1(x) = 0.10000000000000000 \]
\[ P_2(x) = -0.12000000000000000 \]
\[ P_3(x) = 0.70000000000000000 \]
\[ P_4(x) = -0.26000000000000000 \]
\[ P_5(x) = -0.78000000000000000 \]
\[ P_6(x) = 0.72000000000000000 \]
\[ P_7(x) = -0.28000000000000000 \]
\[ P_8(x) = 0.78000000000000000 \]
\[ P_9(x) = -0.16320000000000000 \]
\[ P_{10}(x) = 0.26240000000000000 \]
\[ P_{11}(x) = -0.32640000000000000 \]
\[ P_{12}(x) = 0.31200000000000000 \]
\[ P_{13}(x) = -0.22400000000000000 \]
\[ P_{14}(x) = 0.11520000000000000 \]
\[ P_{15}(x) = -0.38400000000000000 \]
\[ P_{16}(x) = 0.64000000000000000 \]

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.100-03
TEST FOR MULTIPLEITIES: 0.100-01
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

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

ROOT(1) = 0.99940798033691520 + 0.9951391489643941D
ROOT(2) = -0.54206461303878890 + 0.70710675530463460 + 0.70710680706845950
ROOT(3) = -0.38822847926540560 + 0.14488887631171930
ROOT(4) = -0.79497020137361440 + 0.51763825519667240 + 0.19318516083687550
ROOT(5) = 0.13956979981330380 + 0.1767717747788520150
ROOT(6) = -0.82719243111765190 + 0.77645674639870700
ROOT(7) = 0.48296291156562790 + 0.1294095284438187D

ROOT(8) = 0.13306201994285140 + -0.12320934962305410


IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(9) = 0.98514220474234100 + -0.13956979981330380 I DID NOT CONVERGE.

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

Exhibit 6.6.
<table>
<thead>
<tr>
<th>ROOT</th>
<th>INITIAL APPROXIMATION</th>
<th>MULTIPLECTIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.999481137951467600D+00 + 0.99507175806017440D+00</td>
<td>5</td>
<td>0.94629421196362770D+00 + 0.1294995244438107D+00</td>
</tr>
<tr>
<td>2</td>
<td>0.995285315311717400D+00 - 0.10031801886766580D+00</td>
<td>1</td>
<td>0.70710667953046340D+00 + 0.27718080768645950D+00</td>
</tr>
<tr>
<td>3</td>
<td>0.995687671786784000D+00 + 0.10033567459321000D+00</td>
<td>1</td>
<td>0.380233847292954000D+00 + 0.54446889703171930D+00</td>
</tr>
<tr>
<td>4</td>
<td>0.999315959028364000D+00 - 0.995231048523120000D+00</td>
<td>1</td>
<td>0.3075382551967240D+00 + 0.81931851038368775D+00</td>
</tr>
<tr>
<td>5</td>
<td>0.99592313317176770D+00 + 0.10031671344932110D+00</td>
<td>1</td>
<td>0.70710667953046340D+00 + 0.27718080768645950D+00</td>
</tr>
<tr>
<td>6</td>
<td>0.99585780717884340D+00 + 0.10033567459321000D+00</td>
<td>1</td>
<td>0.380233847292954000D+00 + 0.54446889703171930D+00</td>
</tr>
<tr>
<td>7</td>
<td>0.99991569963089460D+00 + 0.99507175806017440D+00</td>
<td>1</td>
<td>0.1294995244438107D+00 + 0.94629421196362770D+00</td>
</tr>
<tr>
<td>8</td>
<td>0.9939104218906621D+00 - 0.99507175806017440D+00</td>
<td>1</td>
<td>0.27718080768645950D+00 + 0.70710667953046340D+00</td>
</tr>
</tbody>
</table>

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

THE COEFFICIENTS OF PI(X) ARE

\[ P_1(x) = 0.1000000000000000D+01 + 0.0000000000000000D+00 \]
\[ P_2(x) = -0.7000000000000000D+00 + 0.1050000000000000D+00 \]
\[ P_3(x) = -0.2800000000000000D+00 + 0.5800000000000000D+00 \]
\[ P_4(x) = -0.1710000000000000D+00 + 0.1500000000000000D+00 \]
\[ P_5(x) = -0.7500000000000000D+00 + 0.2510000000000000D+00 \]
\[ P_6(x) = -0.2230000000000000D+00 + 0.1040000000000000D+00 \]
\[ P_7(x) = 0.1200000000000000D+00 + 0.1040000000000000D+00 \]

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

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF PI(X)  

ROOT(1) = 0.1999954888749810D+01 + 0.2000017733363912D+01 3 0.4829629115652790D+00 + 0.12940952844381870D+00
ROOT(2) = 0.1013502627672869D+01 + 0.1996163083160882D+01 1 0.70710675530463460D+00 + 0.70710680706845950D+00
ROOT(3) = 0.9999998306711213D+00 + 0.20000000716850890D+01 1 SOLVED BY DIRECT METHOD
ROOT(4) = -0.1000002106173084D+01 -0.4999768342186059D+00 I

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF PI(X)  

ROOT(1) = 0.1999954888749810D+01 + 0.2000017733363912D+01 3 0.4829629115652790D+00 + 0.12940952844381870D+00
ROOT(2) = 0.1000000159779278D+01 + 0.19999999796000450D+01 1 0.70710675530463460D+00 + 0.70710680706845950D+00
ROOT(3) = 0.9999998306711213D+00 + 0.20000000716850890D+01 1 SOLVED BY DIRECT METHOD
ROOT(4) = -0.99999999999999998D+00 + 0.5000000000000000D+00 I

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

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

\[
\begin{align*}
P(1) &= 0.48000000000000000000 \\
P(2) &= 0.25571200000000000000 \\
P(3) &= -0.72735568000000000000 \\
P(4) &= -0.38595656490000000000 \\
P(5) &= 0.17531066490000000000 \\
P(6) &= -0.49074987200000000000 \\
P(7) &= -0.10227945221300000000 \\
P(8) &= 0.16427420205600000000 \\
P(9) &= -0.20366258884200000000 \\
P(10) &= -0.18712557800100000000 \\
P(11) &= -0.12749972985900000000 \\
P(12) &= -0.28146927168000000000 \\
P(13) &= 0.13294344348000000000 \\
P(14) &= 0.30539007747000000000 \\
P(15) &= -0.18359909200000000000 \\
P(16) &= 0.27556200000000000000
\end{align*}
\]

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. 0.0000
RADIUS TO END SEARCH. 0.0000

BEFORE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF \( P(x) \)  

<table>
<thead>
<tr>
<th>ROOT</th>
<th>MULTIPlicITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root 1</td>
<td>10</td>
<td>0.30000000000000000000</td>
</tr>
<tr>
<td>Root 2</td>
<td>2</td>
<td>0.16647296105786000000</td>
</tr>
<tr>
<td>Root 3</td>
<td>1</td>
<td>0.12027316109690000000</td>
</tr>
<tr>
<td>Root 4</td>
<td>1</td>
<td>0.60108781241761000000</td>
</tr>
<tr>
<td>Root 5</td>
<td>1</td>
<td>0.16648659311784000000</td>
</tr>
<tr>
<td>Root 6</td>
<td>1</td>
<td>-0.23333333333333000000</td>
</tr>
<tr>
<td>Root 7</td>
<td>2</td>
<td>-0.10013777319814000000</td>
</tr>
<tr>
<td>Root 8</td>
<td>1</td>
<td>0.32549055271964000000</td>
</tr>
<tr>
<td>Root 9</td>
<td>1</td>
<td>-0.99651329325563000000</td>
</tr>
<tr>
<td>Root 10</td>
<td>1</td>
<td>0.33444081144049000000</td>
</tr>
<tr>
<td>Root 11</td>
<td>1</td>
<td>-0.11522209724821500000</td>
</tr>
<tr>
<td>Root 12</td>
<td>1</td>
<td>0.11130840245770000000</td>
</tr>
<tr>
<td>Root 13</td>
<td>1</td>
<td>-0.12769927775344400000</td>
</tr>
</tbody>
</table>

AFTER THE 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>( R_{3T1} ) = 0.3000000056258190-02 + 0.82793710251688E-12</td>
<td>2</td>
<td>0.48294291556562790D-07 + 0.12940592844381870D-01</td>
</tr>
<tr>
<td>( R_{3T2} ) = 0.70710675304643460D00 + 0.3000000056258190-02</td>
<td>1</td>
<td>0.70710675304643460D00 + 0.3000000056258190-02</td>
</tr>
<tr>
<td>( R_{3T3} ) = 0.1300000056258190-02</td>
<td>1</td>
<td>0.1300000056258190-02</td>
</tr>
<tr>
<td>( R_{3T4} ) = 0.237512314523230D-04 + 0.3000000056258190-02</td>
<td>1</td>
<td>0.3000000056258190-02</td>
</tr>
<tr>
<td>( R_{3T5} ) = 0.307663351038230D-04 + 0.3000000056258190-02</td>
<td>1</td>
<td>0.3000000056258190-02</td>
</tr>
<tr>
<td>( R_{3T6} ) = 0.307663351038230D-04 + 0.3000000056258190-02</td>
<td>1</td>
<td>0.3000000056258190-02</td>
</tr>
<tr>
<td>( R_{3T7} ) = 0.4684849625640560D00 + 0.12940592844381870D-01</td>
<td>1</td>
<td>0.12940592844381870D-01</td>
</tr>
<tr>
<td>( R_{3T8} ) = 0.2397327410073823D-04 + 0.3000000056258190-02</td>
<td>1</td>
<td>0.3000000056258190-02</td>
</tr>
<tr>
<td>( R_{3T9} ) = 0.3973984540428445D-07 + 0.1000000042454684D01</td>
<td>1</td>
<td>0.1000000042454684D01</td>
</tr>
<tr>
<td>( R_{3T10} ) = 0.61890537903191320-07 + 0.1500000140768000D01</td>
<td>1</td>
<td>0.1500000140768000D01</td>
</tr>
<tr>
<td>( R_{3T11} ) = 0.2397327410073823D-04 + 0.3000000056258190-02</td>
<td>1</td>
<td>0.3000000056258190-02</td>
</tr>
<tr>
<td>( R_{3T12} ) = 0.2397327410073823D-04 + 0.3000000056258190-02</td>
<td>1</td>
<td>0.3000000056258190-02</td>
</tr>
</tbody>
</table>

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

THE COEFFICIENTS OF $P(x)$ ARE

$$P(1) = 0.1000000000000000 \times 0.0000000000000000 \ 01$$
$$P(1) = -0.5175000000000000 \times 0.1150000000000000 \ 02$$
$$P(1) = 0.0375000000000000 \times -0.3475000000000000 \ 03$$
$$P(1) = -0.4750000000000000 \times 0.4462500000000000 \ 03$$
$$P(1) = 0.9500000000000000 \times 0.4267500000000000 \ 03$$
$$P(1) = 0.1810000000000000 \times 0.4267500000000000 \ 03$$

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR CONVERGENCE: 0.100-10
TEST FOR MULTIPLEITIES: 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 1</th>
<th>MULTIPICITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.200047027873018D 01</td>
<td>3</td>
<td>0.4829629115656279D 00 + 0.1294095284438187D 00</td>
</tr>
<tr>
<td>0.100056380690988D 01</td>
<td>1</td>
<td>0.7071067539864340D 00 + 0.7071068076649595D 00</td>
</tr>
<tr>
<td>-0.106352841874984D 01</td>
<td>1</td>
<td>0.3982269265405660D 00 + 0.3982269265405600D 00</td>
</tr>
<tr>
<td>-0.972212251686926D 00</td>
<td>1</td>
<td>0.5603769757070328D 00 + 0.5603769757070328D 00</td>
</tr>
<tr>
<td>0.9999637866056641D 00</td>
<td>1</td>
<td>0.204909426092692D 00 + 0.204909426092692D 00</td>
</tr>
<tr>
<td>-0.964386813184185D 00</td>
<td>1</td>
<td>0.4956395479086550D 00 + 0.4956395479086550D 00</td>
</tr>
</tbody>
</table>

AFTER THE ATTEMPT TO IMPROVE ACCURACY

ROOTS OF $P(x)$

<table>
<thead>
<tr>
<th>ROOT 1</th>
<th>MULTIPICITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.200047027873018D 01</td>
<td>3</td>
<td>0.4829629115656279D 00 + 0.1294095284438187D 00</td>
</tr>
<tr>
<td>0.100056380690988D 01</td>
<td>1</td>
<td>0.7071067539864340D 00 + 0.7071068076649595D 00</td>
</tr>
<tr>
<td>-0.106352841874984D 01</td>
<td>1</td>
<td>0.3982269265405660D 00 + 0.3982269265405600D 00</td>
</tr>
<tr>
<td>-0.972212251686926D 00</td>
<td>1</td>
<td>0.5603769757070328D 00 + 0.5603769757070328D 00</td>
</tr>
<tr>
<td>0.9999637866056641D 00</td>
<td>1</td>
<td>0.204909426092692D 00 + 0.204909426092692D 00</td>
</tr>
<tr>
<td>-0.964386813184185D 00</td>
<td>1</td>
<td>0.4956395479086550D 00 + 0.4956395479086550D 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

<table>
<thead>
<tr>
<th>P(x)</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(0)</td>
<td>0.10000000000000000</td>
</tr>
<tr>
<td>P(1)</td>
<td>-0.12000000000000000</td>
</tr>
<tr>
<td>P(2)</td>
<td>-0.20000000000000000</td>
</tr>
<tr>
<td>P(3)</td>
<td>0.72000000000000000</td>
</tr>
<tr>
<td>P(4)</td>
<td>-0.16320000000000000</td>
</tr>
<tr>
<td>P(5)</td>
<td>0.32640000000000000</td>
</tr>
<tr>
<td>P(6)</td>
<td>-0.32640000000000000</td>
</tr>
<tr>
<td>P(7)</td>
<td>0.31200000000000000</td>
</tr>
<tr>
<td>P(8)</td>
<td>-0.22400000000000000</td>
</tr>
<tr>
<td>P(9)</td>
<td>0.11520000000000000</td>
</tr>
<tr>
<td>P(10)</td>
<td>-0.38400000000000000</td>
</tr>
<tr>
<td>P(11)</td>
<td>0.64000000000000000</td>
</tr>
<tr>
<td>P(12)</td>
<td>0.48296291156562790</td>
</tr>
<tr>
<td>P(13)</td>
<td>0.12940952844381870</td>
</tr>
</tbody>
</table>

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-09
TEST FOR MULTIPLICITIES. 0.10D-01
RADIUS TO START SEARCH. 0.00D-00
RADIUS TO END SEARCH. 0.000-00

BEFORE ATTEMPT TO IMPROVE ACCURACY

<table>
<thead>
<tr>
<th>ROOTI</th>
<th>MULTIPLECIIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT1</td>
<td>4</td>
<td>0.48296291156562790 0 0</td>
</tr>
<tr>
<td>ROOT2</td>
<td>1</td>
<td>0.70710675530463460 0</td>
</tr>
<tr>
<td>ROOT3</td>
<td>1</td>
<td>0.70710680706845950 0</td>
</tr>
<tr>
<td>ROOT4</td>
<td>1</td>
<td>0.19318516083687550</td>
</tr>
<tr>
<td>ROOT5</td>
<td>1</td>
<td>0.17677667465030570 0</td>
</tr>
<tr>
<td>ROOT6</td>
<td>1</td>
<td>0.77645674639870700</td>
</tr>
<tr>
<td>ROOT7</td>
<td>1</td>
<td>0.17677667465030570 0</td>
</tr>
<tr>
<td>ROOT8</td>
<td>1</td>
<td>0.17677667465030570 0</td>
</tr>
</tbody>
</table>

AFTER THE ATTEMPT TO IMPROVE ACCURACY

<table>
<thead>
<tr>
<th>ROOTI</th>
<th>MULTIPLECIIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT1</td>
<td>4</td>
<td>0.48296291156562790 0 0</td>
</tr>
<tr>
<td>ROOT2</td>
<td>1</td>
<td>0.70710675530463460 0</td>
</tr>
<tr>
<td>ROOT3</td>
<td>1</td>
<td>0.70710680706845950 0</td>
</tr>
<tr>
<td>ROOT4</td>
<td>1</td>
<td>0.19318516083687550</td>
</tr>
<tr>
<td>ROOT5</td>
<td>1</td>
<td>0.17677667465030570 0</td>
</tr>
<tr>
<td>ROOT6</td>
<td>1</td>
<td>0.77645674639870700</td>
</tr>
<tr>
<td>ROOT7</td>
<td>1</td>
<td>0.17677667465030570 0</td>
</tr>
<tr>
<td>ROOT8</td>
<td>1</td>
<td>0.17677667465030570 0</td>
</tr>
</tbody>
</table>

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 MULTIPLEMENTS.  0.10D-01
RADIUS TO START SEARCH.  0.00D 00
RADIUS TO END SEARCH.  0.00D 00

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

<table>
<thead>
<tr>
<th>Term</th>
<th>Coefficient</th>
<th>Term</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(7)</td>
<td>0.1000000000000000D+01</td>
<td>P(6)</td>
<td>-0.7000000000000001D+01</td>
</tr>
<tr>
<td>P(5)</td>
<td>-0.28000000000000000D+02</td>
<td>P(4)</td>
<td>0.1710000000000000D+03</td>
</tr>
<tr>
<td>P(3)</td>
<td>0.1710000000000000D+03</td>
<td>P(2)</td>
<td>-0.2280000000000000D+03</td>
</tr>
<tr>
<td>P(1)</td>
<td>0.7200000000000001D+02</td>
<td></td>
<td></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 3 THE COEFFICIENTS ARE

<table>
<thead>
<tr>
<th>Term</th>
<th>Coefficient</th>
<th>Term</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q(4)</td>
<td>0.1000000000000000D+01</td>
<td>Q(3)</td>
<td>-0.2000000000000000D+01</td>
</tr>
<tr>
<td>Q(2)</td>
<td>-0.7000000000000000D+01</td>
<td>Q(1)</td>
<td>0.9999999999997762D+00</td>
</tr>
</tbody>
</table>

ROOTS OF Q(X)

<table>
<thead>
<tr>
<th>Root</th>
<th>Coefficient</th>
<th>根数</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(1) 1</td>
<td>0.9999999999999999D+00</td>
<td>1</td>
<td>0.1999999999999999D+01</td>
</tr>
<tr>
<td>ROOT(1) 2</td>
<td>0.2000000000000000D+00</td>
<td>1</td>
<td>0.2000000000000000D+01</td>
</tr>
<tr>
<td>ROOT(1) 3</td>
<td>-0.9999999999999999D+00</td>
<td>1</td>
<td>0.5000000000000000D+01</td>
</tr>
</tbody>
</table>

ROOTS OF P(X)

<table>
<thead>
<tr>
<th>Root</th>
<th>Coefficient</th>
<th>根数</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(1) 1</td>
<td>0.9999999999999999D+00</td>
<td>2</td>
<td>0.1999999999999999D+01</td>
</tr>
<tr>
<td>ROOT(1) 2</td>
<td>0.2000000000000000D+00</td>
<td>3</td>
<td>0.2000000000000000D+01</td>
</tr>
<tr>
<td>ROOT(1) 3</td>
<td>-0.9999999999999999D+00</td>
<td>1</td>
<td>0.5000000000000000D+01</td>
</tr>
</tbody>
</table>

Exhibit 6.11. Roots Are: 2+2i (3), 1+2i (2), -1+.5i
GREATEST COMMON DIVISOR METHOD USED WITH NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 2

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.100-02
TEST FOR CONVERGENCE: 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.100-19
TEST FOR MULTIPlicITIES: 0.100-01
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

THE DEGREE OF PI X I IS 15 THE COEFFICIENTS ARE

PI (1) = 0.400000000000000000000 02 + 0.000000000000000000000 00 I
PI (15) = 0.255717000000000000000 03 + 0.386000000000000000000 03 I
PI (14) = 0.179353480000000000000 01 + 0.218994600000000000000 04 I
PI (13) = 0.395955659000000000000 04 + 0.894085140500000000000 01 I
PI (12) = 0.173838666000000000000 05 + 0.143232979000000000000 05 I
PI (11) = 0.496790927000000000000 09 + 0.176595746000000000000 05 I
PI (10) = 0.102239552100000000000 06 + 0.603064227000000000000 01 I
PI (9) = 0.144582212600000000000 04 + 0.137642320000000000000 05 I
PI (8) = 0.203626588200000000000 06 + 0.109384927600000000000 06 I
PI (7) = 0.101295330100000000000 06 + 0.192820004300000000000 06 I
PI (6) = 0.127497928500000000000 06 + 0.217314227400000000000 06 I
PI (5) = 0.261492716000000000000 05 + 0.193469727900000000000 06 I
PI (4) = 0.132963436000000000000 05 + 0.103815622500000000000 06 I
PI (3) = 0.305590074000000000000 06 + 0.299889931000000000000 06 I
PI (2) = 0.189889930200000000000 03 + 0.182732560000000000000 03 I
PI (1) = 0.279567000000000000000 00 + 0.275840000000000000000 00 I

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

Q(1) = 0.400000000000000000000 02 + 0.000000000000000000000 00 I
Q(2) = 0.196859999999999999999 03 + 0.144366600000000000000 03 I
Q(3) = 0.267019999999999999999 03 + 0.527589999999999999999 03 I
Q(4) = 0.327199999999999999999 04 + 0.914469999999999999999 04 I
Q(5) = 0.231599999999999999999 05 + 0.192175200000000000000 05 I
Q(6) = 0.702720000000000000000 06 + 0.132749999999999999999 06 I
Q(7) = 0.755740000000000000000 03 + 0.752000000000000000000 03 I
Q(8) = 0.226801000000000000000 01 + 0.228766666666666666666 01 I

ROTS OF Q(I)

ROOT 11 = 0.320000003065599270-02 -0.1370623176384330-09 I
ROOT 12 = 0.707106707068455000 00 + 0.300000003864599700 00 I
ROOT 33 = 0.144380355435909000 00 + 0.144380355435909000 00 I
ROOT 41 = 0.1187072219483990-10 + 0.156000000000000000000 01 I

Exhibit 6.12
### Exhibit 6.12

**Roots Are:** 
-2.33, 0.003 (2), i(2), 1.5i (2), 
-1.5i (2) 3i (3), -1-i (3)
GREATEST COMMON DIVISOR METHOD USED WITH NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 3

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.10D-02
TEST FOR CONVERGENCE: 0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.10D-19
TEST FOR MULTPLICITIES: 0.10D-11
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) = 0.100000000000000000000000 01 + 0.000000000000000000000000 00 I
P(1) = -0.500000000000000000000000 01 + 0.115000000000000000000000 02 I
P(2) = 0.517500000000000000000000 02 + 0.430000000000000000000000 03 I
P(3) = 0.585750000000000000000000 03 + 0.347500000000000000000000 04 I
P(4) = 0.307500000000000000000000 04 + 0.347500000000000000000000 05 I
P(5) = 0.157250000000000000000000 05 + 0.144625000000000000000000 06 I
P(6) = 0.181000000000000000000000 06 + 0.142000000000000000000000 07 I
P(7) = 0.158000000000000000000000 07 + 0.600000000000000000000000 01 I
P(8) = 0.100000000000000000000000 01 + 0.000000000000000000000000 00 I
P(9) = -0.2000000000000273D 01 + -0.4500000000000275D 01 I
P(10) = -0.7000000000000935D 01 + 0.3500000000000469D 01 I
P(11) = 0.9999999999994173D 00 + 0.7000000000000498D 01 I

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

Q(0) = 0.100000000000000000000000 01 + 0.000000000000000000000000 00 I
Q(1) = -0.2000000000000273D 01 + -0.4500000000000275D 01 I
Q(2) = -0.7000000000000935D 01 + 0.3500000000000469D 01 I
Q(3) = 0.9999999999994173D 00 + 0.7000000000000498D 01 I

ROOTS OF Q(X) INITIAL APPROXIMATION
ROOT(1) = 0.9999999999995483D 00 + 0.1999999999997559D 01 I
ROOT(2) = 0.2000000000000273D 01 + 0.299999999999599D 01 I
ROOT(3) = 0.4999999999994173D 00 I

ROOTS OF P(X) MULTIPLICITIES
ROOT(1) = 0.9999999999995483D 00 + 0.1999999999997559D 01 I
ROOT(2) = 0.2000000000000273D 01 + 0.299999999999599D 01 I
ROOT(3) = 0.4999999999994173D 00 I

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

POLYNOMIAL NUMBER 4

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-12
TEST FOR MULTIPlicITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

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

P(12) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
P(11) = 0.1200000000000000D 02 + 0.0000000000000000D 00 I
P(10) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
P(9)  = 0.3800000000000000D 03 + 0.0000000000000000D 00 I
P(8)  = 0.1630000000000000D 04 + 0.0000000000000000D 00 I
P(7)  = 0.2640000000000000D 04 + 0.0000000000000000D 00 I
P(6)  = 0.3240000000000000D 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)  = 0.1999999999999999D 01 + 0.0000000000000000D 00 I
P(2)  = 0.3800000000000000D 03 + 0.0000000000000000D 00 I
P(1)  = 0.6400000000000001D 02 + 0.0000000000000000D 00 I
P(0)  = 0.0000000000000000D 00

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(2) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
Q(1) = 0.1999999999999999D 01 + 0.0000000000000000D 00 I
Q(0) = 0.1999999999999999D 01 + 0.0000000000000000D 00 I

ROOTS OF P(X)

MULTIPlicITIES

ROOTS 11 = 0.1000000000000001D 01 + 0.4999999999999999D 740 00 I 6
ROOTS 21 = 0.1000000000000001D 01 - 0.4999999999999999D 740 00 I 6

RESULTS OF SUBROUTINE QUAD

Exhibit 6.14. Roots Are: 1+i (6), 1-i (6)
GREATEST COMMON DIVISOR METHOD) USED WITH MULTIERS METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 1

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GEO. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE GEO. 0.100-19
TEST FOR MULTIPlicITIES. 0.100-01
RADIUS TO START SEARCH. 0.0000 00
RADIUS TO END SEARCH. 0.0000 00

THE DEGREE OF P1) IS 6 THE COEFFICIENTS ARE

P17 = 0.1000000000000000D+01 + 0.0000000000000000D+00 01
P16 = -0.7000000000000000D+00 - 0.1050000000000000D+00 02 1
P15 = 0.1710000000000000D-01 + 0.5800000000000000D+00 02 1
P14 = 0.2800000000000000D-02 + 0.1710000000000000D+00 03 1
P13 = 0.7200000000000000D+00 - 0.1040000000000000D+00 03 1
P12 = 0.4100000000000000D+00 + 0.1500000000000000D+00 03 1
P11 = 0.1000000000000000D+00 + 0.1040000000000000D+00 03 1

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

THE DEGREE OF Q1) IS 3 THE COEFFICIENTS ARE

Q4 = 0.1000000000000000D+00 + 0.0000000000000000D+00 01
Q3 = -0.7000000000000000D+00 - 0.4500000000000000D+00 01 1
Q2 = -0.1000000000000000D+00 + 0.3500000000000000D+00 01 1
Q1 = 0.99999999999977620 00 + 0.6999999998120 00 01 1

ROOTS OF Q1) INITIAL APPROXIMATION

ROOT 1 = 0.99999999999975650 00 + 0.199999999999740 01 1
ROOT 2 = 0.2000000000000000D+00 + 0.199999999999740 01 1

ROOTS OF P(X) MULTIPlicITIES INITIAL APPROXIMATION

ROOT 1 = 0.99999999999975650 00 + 0.199999999999740 01 1
ROOT 2 = 0.2000000000000000D+00 + 0.199999999999740 01 1
ROOT 3 = -0.49999999999975650 00 + 0.5000000000000000D+00 01 1

Exhibit 6.15. Roots Are: 2+2i (3), 1+2i (2), -1+5i
GREATEST 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. 0.10D-02
TEST FOR CONVERGENCE. 0.10D-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.10D-19
TEST FOR MULTIPLEILITIES. 0.10D-01
RADIUS TO START SEARCH. 0.00D 00
RADIUS TO END SEARCH. 0.00D 00

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

P(161) = 0.48000000000000000D 00 + 0.00000000000000000 00 I
P(15) = -0.35684664680000000D 00 00 00 I
P(143) = -0.35585686960000000D 00 00 00 I
P(12) = -0.17339464230000000D 00 00 00 I
P(11) = -0.10239464230000000D 00 00 00 I
P(6) = 0.20366258840000000D 00 00 00 I
P(5) = 0.18713657800000000D 00 00 00 I
P(4) = 0.13479472895900000D 00 00 00 I
P(3) = 0.30539007747000000D 00 00 00 I
P(2) = 0.30539007747000000D 00 00 00 I
P(1) = 0.27557462000000000D 00 00 00 I

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

THE DEGREE OF Q(X) IS 7
THE COEFFICIENTS ARE

Q(8) = 0.48000000000000000D 00 + 0.00000000000000000 00 I
Q(7) = -0.1598559999959083D 03 00 00 I
Q(6) = 0.2675199999866287D 03 00 00 I
Q(5) = 0.3271999999781904D 03 00 00 I
Q(4) = 0.2301599999604455D 02 00 00 I
Q(3) = -0.7207200000479990D 02 00 00 I
Q(2) = -0.7557840000541563D 02 00 00 I
Q(1) = 0.22600000000000000D 01 00 00 I

ROOTS OF P(X)

ROOT( 1) = 0.30000000000000000D 00 00 I
ROOT( 2) = 0.30000000000000000D 00 00 I
ROOT( 3) = 0.30000000000000000D 00 00 I

ROOTS OF Q(X)

ROOT( 1) = 0.30000000000000000D 00 00 I
ROOT( 2) = 0.30000000000000000D 00 00 I
ROOT( 3) = 0.30000000000000000D 00 00 I
ROOT( 4) = 0.30000000000000000D 00 00 I

INITIAL APPROXIMATIONS

0.12940952844381870 00 00 I
0.12940952844381870 00 00 I
0.12940952844381870 00 00 I
0.12940952844381870 00 00 I

Exhibit 6.16.
ROOT( 5) = -0.23333333333408840 01 + 0.4292669751099780D-11 I
SOLVED BY DIRECT METHOD

ROOT( 6) = -0.11677255178139090-12 + 0.1900000000000000D0 01 I
SOLVED BY DIRECT METHOD

ROOT( 7) = -0.1000000000000000D0 01 - 0.9999999999999570D0 01 I

ROOTS OF P(X)

MULTIPlicITIES INITIAL APPROXIMATION

ROOT( 1) = 0.3000000038645997D-02 + 0.1370543177818745D-09 01 2
ROOT( 2) = 0.14666360907717090-09 + 0.1000000000302556D 01 I 2
ROOT( 3) = 0.1628298116869740-09 + 0.1499999999998079D 01 I 2
ROOT( 4) = 0.2991402714880148D-11 I 1
ROOT( 5) = -0.1767767470087010 01 + 0.70710675530463460 00 I 1
ROOT( 6) = -0.11677255178139090-10 + 0.1500000000000000D0 01 I 2
ROOT( 7) = -0.1000000000000000D0 01 - 0.4999999999985570D0 00 I 3
RESULTS OF SUBROUTINE QUAD

Exhibit 6.16. Roots Are: -2.33, .003 (2), i (2), 1.5i (2),
-1.5i (2) 3i (3), -1-i (3)
GREATEST COMMON DIVISOR METHOD USED WITH MULLERS METHOD TO FIND ZEROS OF POLYNOMIALS
POLYNOMIAL NUMBER 3

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 0
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR ZERO IN SUBROUTINE GCD. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
TEST FOR MULTIPLE. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE DEGREE OF PIX) IS 8 THE COEFFICIENTS ARE

P(9) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
P(8) = -0.5000000000000000D 02 + 0.0000000000000000D 01 I
P(7) = -0.5175000000000000D 02 + 0.4300000000000000D 02 I
P(6) = 0.1075000000000000D 03 + 0.4463500000000000D 03 I
P(5) = 0.3475000000000000D 03 + 0.4420000000000000D 03 I
P(4) = 0.1400000000000000D 03 + 0.4200000000000000D 03 I
P(3) = 0.1572500000000000D 03 + 0.4247500000000001D 03 I
P(2) = 0.1810000000000000D 03 + 0.4420000000000000D 03 I
P(1) = 0.1580000000000000D 03 + 0.6000000000000001D 01 I

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

Q(4) = 0.1000000000000000D 01 + 0.0000000000000000D 00 I
Q(3) = -0.4950000000000000D 01 + 0.4948750000000000D 01 I
Q(2) = -0.9999999999999998D 00 + 0.7000000000000000D 01 I
Q(1) = 0.9999999999999999D 00 + 0.7000000000000000D 01 I

ROOTS OF Q(X)

ROOT( 1) = 0.9999999999999992D 00 + 0.1999999999999995D 01 I
ROOT( 2) = 0.2000000000000000D 01 + 0.4999999999999995D 01 I
ROOT( 3) = -0.9999999999999992D 00 + 0.4999999999999995D 01 I

ROOTS OF P(X)

ROOT( 1) = 0.9999999999999992D 00 + 0.1999999999999995D 01 I
ROOT( 2) = 0.2000000000000000D 01 + 0.4999999999999995D 01 I
ROOT( 3) = -0.9999999999999992D 00 + 0.4999999999999995D 01 I

MULTIPLES

INITIAL APPROXIMATION

SOLVED BY DIRECT METHOD

SOLVED BY DIRECT METHOD

RESULTS OF SUBROUTINE QUAD

RESULTS OF SUBROUTINE QUAD

Exhibit 6.17. Roots Are: 2+2i (3), 1+2i (2), -1+.5i (3)
GREATEST COMMON DIVISOR METHOD USED WITH MILLER'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. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD. 0.100-19
TEST FOR MULTIPlicITIES. 0.100-01
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE DEGREE OF $P(x)$ IS 12
THE COEFFICIENTS ARE

$P(13) = 0.1000000000000000000 01 + 0.0000000000000000000 00 01$
$P(12) = -0.1200000000000000000 00 + -0.0000000000000000000 00 00$
$P(11) = 0.7200000000000000000 02 + 0.0000000000000000000 00 01$
$P(10) = -0.2800000000000000000 03 + -0.0000000000000000000 00 00$
$P(9) = 0.7800000000000000000 03 + 0.0000000000000000000 00 00$
$P(8) = -0.1632000000000000000 04 + -0.0000000000000000000 00 00$
$P(7) = 0.2624000000000000000 04 + 0.0000000000000000000 00 00$
$P(6) = -0.3264000000000000000 04 + -0.0000000000000000000 00 00$
$P(5) = 0.3120000000000000000 04 + 0.0000000000000000000 00 00$
$P(4) = -0.2240000000000000000 04 + -0.0000000000000000000 00 00$
$P(3) = 0.1152000000000000000 04 + 0.0000000000000000000 00 00$
$P(2) = -0.3840000000000000000 03 + -0.0000000000000000000 00 00$
$P(1) = 0.6400000000000000000 02 + 0.0000000000000000000 00 00$
$P(0) = 0.1000000000000000000 01 + 0.0000000000000000000 00 00

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

$Q(3) = 0.1000000000000000000 01 + 0.0000000000000000000 00 01$
$Q(2) = -0.2000000000000000000 01 + -0.0000000000000000000 00 00$
$Q(1) = 0.199999999999830 00 01 + 0.0000000000000000000 00 00

ROOTS OF $P(x)$

<table>
<thead>
<tr>
<th>ROOT</th>
<th>MULTIPlicITIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 i</td>
<td>6</td>
</tr>
<tr>
<td>1 i + (6)</td>
<td>6</td>
</tr>
</tbody>
</table>

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 MULTICILITIES OF POLYNOMIALS
POLYNOMIAL NUMBER 1

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.250-02
TEST FOR CONVERGENCE: 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAAD: 0.100-39
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

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

P(1) = 0.1000000000000000 01 + 0.0000000000000000 00 01
P(2) = -0.7000000000000010 01 + 0.1050000000000000 02 01
P(3) = -0.2890000000000000 02 + 0.5800000000000000 02 02
P(4) = 0.1710000000000000 03 + 0.1500000000000000 03 03
P(5) = -0.7390000000000000 02 + 0.2910000000000000 03 02
P(6) = -0.2280000000000000 03 + 0.1040000000000000 03 03
P(7) = 0.7200000000000000 02 + 0.1040000000000000 03 03

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

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF PI(X) WHICH HAVE MULTICITY 1

G(1) = 0.1000000000000000 01 + 0.0000000000000000 00 01
G(2) = 0.9999999999997350 00 + 0.5000000000000125 00 00

ROOTS OF PI(X) MULTICILITIES INITIAL APPROXIMATION

ROOT(1) = -0.99999999973500 00 + 0.5000000000001250 00 01 1
NO INITIAL APPROXIMATIONS

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

THE FOLLOWING POLYNOMIAL, G(X), CONTAINS ALL THE ROOTS OF PI(X) WHICH HAVE MULTICITY 2

G(1) = 0.1000000000000000 01 + 0.0000000000000000 00 01
G(2) = -0.9999999999997170 00 + 0.9999999999998430 01 01

Exhibit 6.19.
<table>
<thead>
<tr>
<th>ROOTS OF P((x))</th>
<th>MULTIPlicITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x = 0.9999999999999179) 01</td>
<td>2</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
</tbody>
</table>

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

\[
2 \mid 0.1999999999999996\times 10^{-11} 01 + 0.000000000000003 00 1 \\
1 \mid -0.1999999999999670 01 - 0.2000000000000192 01 1
\]

<table>
<thead>
<tr>
<th>ROOTS OF P((x))</th>
<th>MULTIPlicITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x = 0.1999999999999670) 01</td>
<td>3</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
</tbody>
</table>

Exhibit 6.19. Roots Are: 2+2i (3), 1+2i (2), -1+.5i
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTONS METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS

POLYNOMIAL NUMBER 2

NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0
MAXIMUM NUMBER OF ITERATIONS: 200
TEST FOR ZERO IN SUBROUTINE GCD: 0.100-02
TEST FOR CONVERGENCE: 0.100-09
TEST FOR ZERO IN SUBROUTINE QUAD: 0.100-19
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

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

P(1) = 0.48000000000000000D02
P(2) = 0.25671200000000000D00
P(3) = 0.00000000000000000D00

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

G(1) = 0.48000000000000000D02
G(2) = 0.11199999928960310-03

ROOTS OF P(X)
MULTICILITIES
INITIAL APPROXIMATIONS

ROOT(1) = -0.2333333318533970 C1 + 0.177395970570930-07 1
NO INITIAL APPROXIMATIONS

Exhibit 6.20.
THE FOLLOWING POLYNOMIAL, $g(x)$, CONTAINS ALL THE ROOTS OF $p(x)$ WHICH HAVE MULTIPLICITY 2

$g(x) = 0.10000000000000000000 0 + 0.00000000000000000000 0 0 1$
$g(x) = -0.299976653634360 0 0 + 0.499999692627250 0 0 0 1$
$g(x) = 0.174999931177850 0 0 + 0.30000000000000000000 0 0 0 1$
$g(x) = 0.3550109902460760 0 0 - 0.225000120676590 0 0 0 1$
$g(x) = 0.64634945362830 0 0 + 0.6749609382170720 0 0 0 2

ROOTS OF $g(x)$

INITIAL APPROXIMATION

$g(x)$ = 0.4829629115656270 0 0 + 0.12940992964438187 0 0 0 1
RESULTS OF SUBROUTINE CHAOS
RESULTS OF SUBROUTINE CHAOS

ROOTS OF $p(x)$

MULTIPLEITIES

INITIAL APPROXIMATION

$g(x)$ = 0.4829629115656270 0 0 + 0.12940992964438187 0 0 0 1
$g(x)$ = 0.707106875145446 0 0 + 0.707106875145446 0 0 0 1
$g(x)$ = 0.000000000000000 0 0 + 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 + 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 + 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 + 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 + 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 + 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 + 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 + 0.000000000000000 0 0 0 0

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

$g(x) = 0.10000000000000000000 0 1$
$g(x) = 0.499999692627250 0 2$
$g(x) = -0.30000000000000000000 0 0 0 0$

ROOTS OF $p(x)$

MULTIPLEITIES

INITIAL APPROXIMATION

$g(x)$ = 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 0 0
$g(x)$ = 0.000000000000000 0 0 0 0

Exhibit 6.20: Roots Are: -2.33, .003 (2), i (2), 1.5i (2),
-1.5i (2), 3i (3), -1-i (3)
REPEATED USE OF THE GREATEST COMMON DIVISOR AND NEWTON'S METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIALS
POLYNOMIAL NUMBER 3

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

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

\[
P(0) = 0.10000000000000000D 01 + 0.0000000000000000D 00 0 1
P(1) = -0.5000000000000001D 01 + -0.1160000000000000D 02 0 2
P(2) = 0.30000000000000000D 03 + -0.34750000000000000 03
P(3) = -0.49525000000000000D 03 + -0.49487500000000000 03
P(4) = 0.15725000000000000 03
P(5) = -0.5857500000000001D 03 + 0.42475000000000010 03
P(6) = 0.18100000000000000D 03 + 0.4420000000000001D 03
P(7) = 0.15800000000000000D 03 + 0.6000000000000001D 01
\]

NO ROOTS OF MULTIPlicity 1

THE FOLLOWING POLYNOMIAL, Gf2(X), CONTAINS ALL THE ROOTS OF P(X) WHICH HAVE MULTIPlicity 2

\[
G(0) = 0.10000000000000000D 01 + 0.0000000000000000D 00 0 1
G(1) = -0.999999999964980 00 + -0.199999999964890 01 1
\]

ROOTS OF P(X) MULTIPlicITIES INITIAL APPROXIMATION

\[
\text{ROOT: 1} = 0.999999999964980 00 + 0.199999999964890 01 1 2 NO INITIAL APPROXIMATIONS
\]

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

\[
\begin{align*}
G(1) &= 0.1000000000000000D-01 + 0.0000000000000000D-00 \times 1 \\
G(2) &= -0.1000000000000000D-01 + -0.2500000000000162D-01 \times 1 \\
G(3) &= -0.3000000000000000D-01 + -0.1000000000000421D-01 \times 1
\end{align*}
\]

<table>
<thead>
<tr>
<th>ROOT(1)</th>
<th>ROOT(2)</th>
<th>ROOT(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2000000000000000D-01 + 0.2400000000000166D-01 \times 1</td>
<td>-0.9999999999999926D-00 + -0.9999999999999759D-00 \times 1</td>
<td>NO INITIAL APPROXIMATIONS</td>
</tr>
</tbody>
</table>

Exhibit 6.21. Roots Are: \( 2+2i \) (3), \( 1+2i \) (2), \(-1+.5i \) (3)
Number of initial approximations given: 0
Maximum number of iterations: 200
Test for zero in subroutine GCD: 0.100-02
Test for convergence: 0.120-09
Test for zero in subroutine quad: 0.100-19
Radius to start search: 0.000 00
Radius to end search: 0.000 03

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

P[13] = 0.100000000000000 01 + 0.000000000000000 00 0
P[12] = -0.1200000000000000 02 + 0.000000000000000 00 0
P[11] = 0.720000000000000 02 + 0.000000000000000 00 0
P[10] = -0.2800000000000000 03 + 0.000000000000000 00 0
P[9] = 0.780000000000000 03 + 0.000000000000000 00 0
P[8] = -0.1632000000000000 04 + 0.000000000000000 00 0
P[7] = 0.7628000000000000 04 + 0.000000000000000 00 0
P[6] = -0.3240000000000000 04 + 0.000000000000000 00 0
P[5] = 0.3120000000000000 05 + 0.000000000000000 00 0
P[4] = -0.3240000000000000 05 + 0.000000000000000 00 0
P[3] = 0.1152000000000000 06 + 0.000000000000000 00 0
P[2] = -0.3840000000000000 06 + 0.000000000000000 00 0
P[1] = 0.8400000000000000 00 0 + 0.000000000000000 00 0

No roots of multiplicity 1

No roots of multiplicity 2

Exhibit 6.22.
NO ROOTS OF MULTIPlicity 3

NO ROOTS OF MULTIPlicity 4

NO ROOTS OF MULTIPlicity 5

THE FOLLOWING POLYNOMIAL, G(x), CONTAINS ALL THE ROOTS OF P(x) WHICH HAVE MULTIPlicity 6.

G13 = 0.1000000000000000 01 + 0.0000000000000000 00 ( 
G12 = -0.2000000000000000 01 + 0.0000000000000000 00 ( 
G11 = 0.2000000000000000 01 + 0.0000000000000000 00 ( 

ROOTS OF P(x)    MULTIPlicITIES    INITIAL APPROXIMATION
ROOT: 1) = 0.1000000000000000 01 + 0.9999999999999999 00 ( 6        NO INITIAL APPROXIMATIONS
ROOT: 2) = 0.1000000000000000 01 + 0.9999999999999999 00 ( 6        NO INITIAL APPROXIMATIONS

Exhibit 6.22. Roots Are: 1+i (6), 1-i (6)
Repeated use of the greatest common divisor and nulls method to extract roots and multiplicities of polynomials

Polynomial number 1

Number of initial approximations given: 0
Maximum number of iterations: 200
Test for zero in subroutine GCD: 0.100-09
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 pixi is 6, the coefficients are:

P(i) = 0.1000000000000000 00
P(1) = -0.1000000000000000 00
P(2) = 0.1000000000000000 00
P(3) = 0.1000000000000000 00
P(4) = 0.1000000000000000 00
P(5) = 0.1000000000000000 00
P(6) = 0.1000000000000000 00

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

G(i) = 0.1000000000000000 00
G(1) = 0.1000000000000000 00

Roots of g(x)

Root 1: -0.999999999999735000 ± 0.500000000000175410 00

Initial approximation: 0.4829629115656279D 00 ± 0.129409528438187D 00

Roots of pixi

Root 1: -0.999999999999735000 ± 0.500000000000175410 00

Multiplicities: 1

Initial approximation: 0.4829629115656279D 00 ± 0.129409528438187D 00

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

\[
G(2) = 0.1000000000000000D\,01 + 0.0000000000000000D\,00 \quad I
\]
\[
G(1) = -0.9999999999999790D\,00 + -0.1999999999968450D\,01 \quad I
\]

ROOTS OF \( G(x) \)  

\[
\text{ROOT}(1) = 0.9999999999999179D\,00 + 0.19999999999968450D\,00 \quad I
\]

ROOTS OF \( P(x) \)  

\[
\text{ROOT}(1) = 0.9999999999999179D\,00 + 0.19999999999968450D\,00 \quad I
\]

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

\[
2 \, 1 = 0.1000000000000000D\,01 + 0.0000000000000000D\,00 \quad I
\]
\[
1 \, 1 = -0.1999999999999670D\,01 + -0.2000000000001519D\,01 \quad I
\]

ROOTS OF \( P(x) \)  

\[
\text{ROOT}(1) = 0.1999999999999670D\,00 + -0.2000000000001519D\,01 \quad I
\]

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 QAD: 0.100-19
RADIUS TO START SEARCH: 0.000 00
RADIUS TO END SEARCH: 0.000 00

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

\[
\begin{align*}
P(10) &= 0.4000000000000000 \times 0.0000000000000000 00 01 \\
P(11) &= 0.2557123000000000 \times 0.0000000000000000 01 01 \\
P(12) &= -0.2189686400000000 \times 0.0000000000000000 01 01 \\
P(13) &= -0.3330555069000000 \times 0.0000000000000000 01 01 \\
P(14) &= -0.1441684868000000 \times 0.0000000000000000 01 01 \\
P(15) &= -0.4467198270000000 \times 0.0000000000000000 01 01 \\
P(16) &= 0.0101834851000000 \times 0.0000000000000000 01 01 \\
P(17) &= 0.8611022879000000 \times 0.0000000000000000 01 01 \\
P(18) &= 0.2396625083000000 \times 0.0000000000000000 01 01 \\
P(19) &= 0.1251872537800000 \times 0.0000000000000000 01 01 \\
P(20) &= 0.3744972769000000 \times 0.0000000000000000 01 01 \\
P(21) &= 0.2173141276200000 \times 0.0000000000000000 01 01 \\
P(22) &= 0.1244592716000000 \times 0.0000000000000000 01 01 \\
P(23) &= 0.4369343962000000 \times 0.0000000000000000 01 01 \\
P(24) &= 0.3053000737000000 \times 0.0000000000000000 01 01 \\
P(25) &= 0.1835899220000000 \times 0.0000000000000000 01 01 \\
P(26) &= 0.2755920000000000 \times 0.0000000000000000 01 01 \\
\end{align*}
\]

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

\[
\begin{align*}
G(10) &= 0.4000000000000000 \times 0.0000000000000000 00 01 \\
G(11) &= 0.2557123000000000 \times 0.0000000000000000 01 01 \\
G(12) &= -0.2189686400000000 \times 0.0000000000000000 01 01 \\
G(13) &= -0.3330555069000000 \times 0.0000000000000000 01 01 \\
G(14) &= -0.1441684868000000 \times 0.0000000000000000 01 01 \\
G(15) &= -0.4467198270000000 \times 0.0000000000000000 01 01 \\
G(16) &= 0.0101834851000000 \times 0.0000000000000000 01 01 \\
G(17) &= 0.8611022879000000 \times 0.0000000000000000 01 01 \\
G(18) &= 0.2396625083000000 \times 0.0000000000000000 01 01 \\
G(19) &= 0.1251872537800000 \times 0.0000000000000000 01 01 \\
G(20) &= 0.3744972769000000 \times 0.0000000000000000 01 01 \\
G(21) &= 0.2173141276200000 \times 0.0000000000000000 01 01 \\
G(22) &= 0.1244592716000000 \times 0.0000000000000000 01 01 \\
G(23) &= 0.4369343962000000 \times 0.0000000000000000 01 01 \\
G(24) &= 0.3053000737000000 \times 0.0000000000000000 01 01 \\
G(25) &= 0.1835899220000000 \times 0.0000000000000000 01 01 \\
G(26) &= 0.2755920000000000 \times 0.0000000000000000 01 01 \\
\end{align*}
\]

ROOTS OF G(X)

ROOT 11 = -0.2333333189333970 01 + 0.17239559705709395 07 01

INITIAL APPROXIMATION

\[
\begin{align*}
0.4829561565627000 \times 0.12940952846381870 00 01 \\
\end{align*}
\]

Multiplicities

Exhibit 6.24.

78
\begin{verbatim}
\texttt{RODRT (1) = -0.29998232059214170 \pm -0.20569882666885600} \\
\texttt{RODRT (2) = 0.6757818456316560 \pm 0.00000059584018400} \\
\texttt{RODRT (3) = 0.23311724169372120 \pm -0.14999997243172500} \\
\texttt{RODRT (4) = -0.35762548781802720 \pm 0.14999997679166400} \\
\end{verbatim}

\begin{verbatim}
\texttt{G15 \alpha = 0.10000000000000000 \pm 0.00000000000000000} \\
\texttt{G14 \alpha = 0.29999999999999999 \pm 0.09999999999999999} \\
\texttt{G13 \alpha = 0.10000000000000000 \pm 0.00000000000000000} \\
\texttt{G1 \alpha = 0.48296291155656279 \pm 0.12940952684338187} \\
\end{verbatim}

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

\begin{verbatim}
\texttt{INITIAL APPROXIMATION} \\
\texttt{SOLVED BY DIRECT METHOD} \\
\texttt{NO INITIAL APPROXIMATIONS} \\
\texttt{NO INITIAL APPROXIMATIONS} \\
\end{verbatim}

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

\begin{verbatim}
\texttt{INITIAL APPROXIMATIONS} \\
\end{verbatim}

Exhibit 6.24.
Exhibit 6.24. Roots Are: -2.33, .003 (2), i (2), 1.5i (2), -1.5i (2), 3i (3), -1-i (3)
REPEATED USE OF THE GREATEST COMMON DIVISOR AND MILLER'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 ZRO IN SUBROUTINE SNZC. 0.100-02
TEST FOR CONVERGENCE. 0.100-09
TEST FOR ZERO IN SUBROUTINE SQAC. 0.100-19
RADIUS TO START SEARCH. 0.000 00
RADIUS TO END SEARCH. 0.000 00

THE DEGREE OF P(i) IS 8. THE COEFFICIENTS ARE

P(0) = 0.100000000000000000 01 + 0.000000000000000000 00 1
P(1) = -0.500000000000000000 01 + 0.115000000000000000 02 1
P(2) = 0.197264350000000000 03 + 0.144879600000000000 03 1
P(3) = 0.307660000000000000 09 + 0.247600000000000000 09 1
P(4) = -0.492200000000000000 03 + 0.499975600000000000 09 1
P(5) = -0.585790000000000000 03 + 0.424750000000000000 09 1
P(6) = 0.181000000000000000 03 + 0.442000000000000000 09 1
P(7) = 0.158000000000000000 03 + 0.560000000000000000 09 1

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

NO ROOTS OF MULTIPlicity 1

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

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

G(0) = 0.100000000000000000 01 + 0.000000000000000000 00 1
G(1) = 0.999999999999944980 00 + 0.1999999999944680 01 1

ROOTS OF G(i)

ROOT(1) = 0.9999999999944680 00 + 0.1999999999944680 01 1

0.422962915367270 00 + 0.1294395264438187 00 1

Exhibit 6.25.
<table>
<thead>
<tr>
<th>ROOTS OF P(X)</th>
<th>MULTIPlicITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(1)</td>
<td>0.999999999999449800 + 0.199999999999646800i</td>
<td>2 0.482962911565627900 + 0.12940952984438187001</td>
</tr>
</tbody>
</table>

The following polynomial, \(G(X)\), contains all the roots of \(P(X)\) which have multiplicity 3:

\[
G(3) = 0.100000000000000000 + 0.000000000000000000i \\
G(2) = -0.100000000000000000 + 0.000000000000000000i \\
G(1) = -0.100000000000000000 + 0.000000000000000000i
\]

**ROOTS OF G(X)**

<table>
<thead>
<tr>
<th>ROOT(I)</th>
<th>MULTIPlicITIES</th>
<th>INITIAL APPROXIMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT(1)</td>
<td>-0.999999999999449800 + 0.499999999999975900i</td>
<td>3 0.482962911565627900 + 0.12940952984438187001</td>
</tr>
</tbody>
</table>

The roots of \(G(X)\) initial approximation:

\[
\text{ROOT}(2) = 0.200000000000253200 + 0.200000000000165300i \\
\text{ROOT}(1) = 0.200000000000253200 + 0.200000000000165300i \\
\text{SOLVED BY DIRECT METHOD}
\]

In the attempt to improve accuracy, root 2 did not converge after 20 iterations. The present approximation is:

\[
0.200199997425326001 + 0.200199997425326001i
\]

Not all roots of the above polynomial \(G\) were found.

**Exhibit 6.25. Roots Are:** 2+2i (3), 1+2i (2), -1+.5i (3)
**Repeated Use of the Greatest Common Divisor and Muller's Method to Extract Roots and Multiplicities of Polynomials**

**Polynomial Number:** 4

- **Number of Initial Approximations Given:** 0
- **Maximum Number of Iterations:** 200
- **Test for Zero in Subroutine GCD:** 0.100e-07
- **Test for Convergence:** 0.100e-09
- **Test for Zero in Subroutine Quad:** 0.100e-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:

<table>
<thead>
<tr>
<th>x</th>
<th>Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>0.0000000000000000D 01 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>02</td>
<td>0.0000000000000000D 02 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>03</td>
<td>0.0000000000000000D 03 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>04</td>
<td>0.0000000000000000D 04 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>05</td>
<td>0.0000000000000000D 05 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>06</td>
<td>0.0000000000000000D 06 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>07</td>
<td>0.0000000000000000D 07 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>08</td>
<td>0.0000000000000000D 08 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>09</td>
<td>0.0000000000000000D 09 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>10</td>
<td>0.0000000000000000D 10 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>11</td>
<td>0.0000000000000000D 11 + 0.0000000000000000D 00</td>
</tr>
<tr>
<td>12</td>
<td>0.0000000000000000D 12 + 0.0000000000000000D 00</td>
</tr>
</tbody>
</table>

**Exhibit 6.26.**
NO ROOTS OF MULTIPLICITY 3

NO ROOTS OF MULTIPLICITY 4

NO ROOTS OF MULTIPLICITY 5

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

\[ G(X) = 0.100000000000000001 + 0.00000000000000000D 00 1 \]
\[ G(X) = -0.200000000000000000008D 01 + 0.00000000000000000D 00 1 \]

ROOTS OF G(X)

\[ \text{ROOT}(1) = 0.1000000000000033D 01 + 0.9999999999999707D 00 1 \]
\[ \text{ROOT}(2) = 0.1000000000000033D 01 + -0.9999999999999707D 00 1 \]

SOLVED BY DIRECT METHOD

ROOTS OF P(X)

\[ \text{ROOT}(1) = 0.100000000000000001 + 0.9999999999999707D 00 1 \]
\[ \text{ROOT}(2) = 0.100000000000000001 + -0.9999999999999707D 00 1 \]

INITIAL APPROXIMATION

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:

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

where

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

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

*These illustrations are representative of Newton's method in double precision. The control cards for Muller's method are similarly prepared.
columns (7-8) may be left blank. If XSTART is left blank, it is interpreted as 0.

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

```
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>1 2 4 5 7 8</td>
</tr>
<tr>
<td>O</td>
<td>4 0</td>
</tr>
<tr>
<td>P</td>
<td>7 2</td>
</tr>
<tr>
<td>L</td>
<td>7 8 0</td>
</tr>
<tr>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XSTART</td>
</tr>
</tbody>
</table>

Example A.1
```

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

Example A.2 shows a portion of a control card which generated initial approximations beginning at a magnitude of 25.0 for a sixth degree polynomial.
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: ($j = 1, 3$)

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

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

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

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

3. Searching the Complex Plane

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

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

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

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

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

Example A.4

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

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

5. Solving Quadratic Polynomial

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

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

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 Nth 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>N</th>
<th>O</th>
<th>P</th>
<th>L</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N</th>
<th>O</th>
<th>P</th>
<th>L</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>1 2</th>
<th>4 5</th>
<th>7 8</th>
<th>6 4</th>
<th>7 0</th>
<th>7 2</th>
<th>8 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>N O P O L Y</td>
<td>N</td>
<td>N I A P</td>
<td>XSTART</td>
<td>XEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 7 2</td>
<td></td>
<td></td>
<td></td>
<td>1.5D+01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example A.6

By defining XSTART between 0. and 15. an annulus instead of the circle will be searched (Exhibit A.3).
Figure A.1. Generating Initial Approximations
Figure A.2. Altering Approximations
Figure A.3. Distribution of Approximations
NEWTON'S METHOD TO FIND ZEROS OF POLYNOMIALS

POLYNOMIAL NUMBER 2 OF DEGREE 3

THE COEFFICIENTS OF FIRST ARE

p1 1) = 0.100000000000000D 00
p1 2) = 0.000000000000000D 00
p1 3) = -0.100000000000000D 00
p1 4) = -0.200000000000000D 00

THE NUMBER OF INITIAL APPROXIMATIONS GIVEN: 0

MAXIMUM NUMBER OF ITERATIONS: 3

TEST FOR CONVERGENCE: 0.100-01

TEST FOR MULTIPLE: 0.100-01

RADIUS TO START SEARCH: 0.000 00

RADIUS TO END SEARCH: 0.000 00

NO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AFTER 3 ITERATIONS.

INITIAL APPROXIMATION

ALTERNED APPROXIMATION

COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

p1 1) = 0.100000000000000D 00
p1 2) = 0.200000000000000D 00
p1 3) = -0.300000000000000D 00
p1 4) = -0.400000000000000D 00

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

POLYNOMIAL NUMBER 1 OF DEGREE 7

THE COEFFICIENTS OF P0(X) ARE

| P0 | 0.10000000000000000 | 0.00000000000000000 | 0.21000000000000000 | 0.71000000000000000 | 0.70000000000000000 | 0.14000000000000000 | 0.16000000000000000 | 0.00000000000000000 | 0.16000000000000000 |
| P0 | -0.10000000000000000 | 0.11000000000000000 | 0.59000000000000000 | 0.70000000000000000 | 0.72000000000000000 | 0.16000000000000000 | 0.18000000000000000 | 0.00000000000000000 | 0.18000000000000000 |
| P0 | -0.16240000000000000 | 0.49000000000000000 | 0.18320000000000000 | 0.70000000000000000 | 0.72300000000000000 | 0.16000000000000000 | 0.18000000000000000 | 0.00000000000000000 | 0.18000000000000000 |
| P0 | -0.19220000000000000 | -0.19220000000000000 | 0.19220000000000000 | 0.70000000000000000 | 0.72300000000000000 | 0.16000000000000000 | 0.18000000000000000 | 0.00000000000000000 | 0.18000000000000000 |

NUMBER OF INITIAL APPROXIMATIONS GIVEN. 3
MAXIMUM NUMBER OF ITERATIONS. 200
TEST FOR CONVERGENCE. 0.100-00
TEST FOR MULTIPLEitz. 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 P0(X) ARE

ROOTS OF P0(X) MULTIPLETS INITIAL APPROXIMATION

| ROOT | -0.29000000000000000 | -0.30000000000000000 | -0.39000000000000000 | -0.45000000000000000 |
| ROOT | -0.20000000000000000 | 0.00000000000000000 | 0.00000000000000000 | 0.00000000000000000 |
| ROOT | -0.15000000000000000 | -0.45000000000000000 | 0.00000000000000000 | 0.00000000000000000 |

AFTER THE ATTEMPT TO IMPROVE ACCURACY, THE ZEROS OF P0(X) ARE

ROOTS OF P0(X) MULTIPLETS INITIAL APPROXIMATION

|ROOT | -0.29000000000000000 | -0.30000000000000000 | -0.39000000000000000 | -0.45000000000000000 |
| ROOT | -0.20000000000000000 | 0.00000000000000000 | 0.00000000000000000 | 0.00000000000000000 |
| ROOT | -0.15000000000000000 | -0.45000000000000000 | 0.00000000000000000 | 0.00000000000000000 |

COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND

| D1 | 0.10000000000000000 | 0.00000000000000000 | 0.00000000000000000 | 0.00000000000000000 |
| D1 | -0.29000000000000000 | -0.30000000000000000 | -0.39000000000000000 | -0.45000000000000000 |
| D1 | -0.20000000000000000 | 0.00000000000000000 | 0.00000000000000000 | 0.00000000000000000 |

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

THE COEFFICIENTS OF PL(X) ARE
P1 = 0.1000000000000000D 01
P2 = -0.1000000000000000D 01
P3 = -0.5900000000000000D 01
P4 = 0.1900000000000000D 01
P5 = 0.7000000000000000D 02
P6 = -0.16240000000000000D 02
P7 = 0.1427000000000000D 02
P8 = 0.1500000000000000D 02

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

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

ROOT(1) = -0.2999999999999997D 01
ROOT(2) = 0.2000000000000000D 01
ROOT(3) = 0.40000000000000010D 01
ROOT(4) = -0.9999999999999998D 00
ROOT(5) = -0.20000000000000004D 01

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

ROOT(1) = -0.2999999999999997D 01
ROOT(2) = 0.2000000000000000D 01
ROOT(3) = 0.40000000000000010D 01
ROOT(4) = -0.9999999999999998D 00
ROOT(5) = -0.20000000000000004D 01

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

NEWTON'S METHOD

1. Use of the Program

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

The program is designed to solve polynomials of degree 25 or less. Both the coefficient of the highest degree term and the constant coefficient should be non-zero. In order to solve polynomials of degree $N$, where $N > 25$, certain array dimensions must be changed. These are listed in Table B.I for the main program and subprograms in double precision.
Table B.I lists the system functions used in the program of Newton's method. In the table "d" denotes a double precision variable name.
TABLE B.II

SYSTEM FUNCTIONS USED IN NEWTON'S METHOD

Double Precision

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

2. Input Data for Newton's Method

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

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

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

Control Information

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

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOPOLY</td>
<td>c.c. 1-2</td>
<td>Number of the polynomial. Integer. Right justified.</td>
</tr>
<tr>
<td>N</td>
<td>c.c. 4-5</td>
<td>Degree of the polynomial. Integer. Right justified.</td>
</tr>
<tr>
<td>NIAP</td>
<td>c.c. 7-8</td>
<td>Number of initial approximations to be read. Integer. If no approximations are given, this should be left blank.</td>
</tr>
<tr>
<td>MAX</td>
<td>c.c. 19-21</td>
<td>Maximum number of iterations. Integer. Right justified. 200 is recommended.</td>
</tr>
<tr>
<td>EPSCNV</td>
<td>c.c. 30-35</td>
<td>Convergence requirement. Double precision. 1.D-10 is recommended.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Card Columns</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| EPSQ          | c.c. 37-42   | Tolerance check for zero (0) in subroutine QUAD. Double precision. Right justify.
| EPSMUL        | c.c. 44-49   | Multiplicity requirement. Double precision. Right justify. 1.D-02 is recommended. |
| XSTART        | c.c. 64-70   | Magnitude at which to begin generating initial approximations. Double precision. Right justify. This is a special feature of the program and may be omitted. |
| XEND          | c.c. 72-78   | Magnitude at which to end the generating of initial approximations. Double precision. Right justify. This is a special feature of the program and may be omitted. |
| KCHECK        | c.c. 80      | This should be left blank. |

**Coefficients of the Polynomial**

The coefficient cards follow the control card. For an \( N \)th degree polynomial, \( N+1 \) coefficients must be entered one per card. The coefficient of the highest degree term is entered first. For example, if the polynomial \( x^5 + 3x^4 + 2x + 5 \) were to be solved, the order in which the coefficients would be entered is: 1, 3, 0, 0, 2, 5. Each
coefficient is entered, one per card, as described in Table B.IV and illustrated in Figure B.3.

TABLE B.IV

COEFFICIENT DATA FOR NEWTON'S METHOD

<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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Double precision.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Right justify.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Double precision.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Right justify.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If none, leave blank or enter 0.0D00.</td>
</tr>
</tbody>
</table>

Initial Approximations

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

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

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

**End Card**

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

**TABLE B.VI**

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

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

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

- R - real variable
- I - integer variable
- C - complex variable
- D - double precision
- L - logical variable
- A - alphanumerical variable

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

- E - entered
- R - returned
- ECR - entered, changed, and returned
- C - variable in common

4. Description of Program Output

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

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

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

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

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

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

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

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

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

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

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

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

This message is printed as a result of the value of the derivative of the original polynomial at an approximation, XXX, being zero (0). It occurred in the attempt to improve the accuracy of a zero. The previous message is then printed.
Figure B.1. Sequence of Input Data for Newton's Method
<table>
<thead>
<tr>
<th>NOPOLY</th>
<th>NNIAP</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>1.0D+01</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>EPSCNV</td>
<td>R</td>
<td>EPSCNV</td>
<td>D</td>
<td></td>
<td>Tolerance check for convergence</td>
</tr>
<tr>
<td>EPSMUL</td>
<td>R</td>
<td>EPSMUL</td>
<td>D</td>
<td></td>
<td>Tolerance check for multiplicities</td>
</tr>
<tr>
<td>EPSQ</td>
<td>R</td>
<td>EPSQ</td>
<td>D'</td>
<td></td>
<td>Tolerance check for zero in subroutine QUAD</td>
</tr>
<tr>
<td>XSTART</td>
<td>R</td>
<td>XSTART</td>
<td>D</td>
<td></td>
<td>Magnitude from which to begin the search for zeros</td>
</tr>
<tr>
<td>XEND</td>
<td>R</td>
<td>XEND</td>
<td>D</td>
<td></td>
<td>Magnitude to end the search for zeros</td>
</tr>
<tr>
<td>KCHECK</td>
<td>I</td>
<td>KCHECK</td>
<td>I</td>
<td></td>
<td>Program Control. When KCHECK = 1, program will terminate execution.</td>
</tr>
<tr>
<td>NA</td>
<td>I</td>
<td>NA</td>
<td>I</td>
<td></td>
<td>Number of coefficients or original polynomial</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>RA,VA</td>
<td>D</td>
<td></td>
<td>Array containing the coefficients of original polynomial P(X)</td>
</tr>
<tr>
<td>NDEF</td>
<td>I</td>
<td>NDEF</td>
<td>I</td>
<td></td>
<td>Degree of current deflated polynomial</td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>L</td>
<td>I</td>
<td></td>
<td>Counter for number of initial approximations used</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td></td>
<td>Counter for number of iterations</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
<td>I</td>
<td></td>
<td>Counter for number of roots found (counting multiplicities)</td>
</tr>
<tr>
<td>IALTER</td>
<td>I</td>
<td>IALTER</td>
<td>I</td>
<td></td>
<td>Counter for number of alterations of each initial approximation</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
<td></td>
<td>Program control</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td></td>
<td>Counter for number of distinct roots found</td>
</tr>
<tr>
<td>ND</td>
<td>I</td>
<td>ND</td>
<td>I</td>
<td></td>
<td>Program control &amp; number of coefficient of deflated polynomial for which no zeros were found</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Double Precision Variable</td>
<td>Type</td>
<td>Disposition of Argument</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>---------------------------</td>
<td>------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>XO</td>
<td>C</td>
<td>RXO, VXO</td>
<td>D</td>
<td></td>
<td>Current approximation (X_n) to root</td>
</tr>
<tr>
<td>COEF</td>
<td>C</td>
<td>RCOEF, VCOEF</td>
<td>D</td>
<td></td>
<td>Working array containing coefficients of current deflated polynomial</td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX, VDPX</td>
<td>D</td>
<td></td>
<td>Derivative of (P(X)) at some value (X)</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>RPX, VPX</td>
<td>D</td>
<td></td>
<td>Value of (P(X)) at some point (X)</td>
</tr>
<tr>
<td>XZERO</td>
<td>C</td>
<td>RXZERO, VXZERO</td>
<td>D</td>
<td></td>
<td>Array containing the initial approximations</td>
</tr>
<tr>
<td>XNEW</td>
<td>C</td>
<td>RXNEW, VXNEW</td>
<td>D</td>
<td></td>
<td>New approximation (X_{n+1}) obtained from old approximation (X_n) by Newton's Algorithm</td>
</tr>
<tr>
<td>KANS</td>
<td>I</td>
<td>KANS</td>
<td>I</td>
<td></td>
<td>(KANS = 1) implies convergence, (KANS = 0) implies no convergence</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td></td>
<td>Array containing the number of multiplicities of each root</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>RX, VX</td>
<td>D</td>
<td></td>
<td>Array containing the zeros of (P(X))</td>
</tr>
<tr>
<td>XINIT</td>
<td>C</td>
<td>RXINIT, VXINIT</td>
<td>D</td>
<td></td>
<td>Array containing the initial or altered approximations which produced convergence to each root</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
<td>I</td>
<td></td>
<td>Number of coefficients of current deflated polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>RB, VB</td>
<td>D</td>
<td></td>
<td>Array containing the coefficients of newly deflated polynomial</td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</td>
<td>I</td>
<td></td>
<td>Number of distinct roots found by Newton's method, i.e. not solved for directly by subroutine QUAD</td>
</tr>
<tr>
<td>EPSCHK</td>
<td>R</td>
<td>EPSCHK</td>
<td>D</td>
<td></td>
<td>Array containing the coefficients of deflated polynomial for which no zeros were found</td>
</tr>
<tr>
<td>IO1</td>
<td>I</td>
<td>IO1</td>
<td>I</td>
<td></td>
<td>Unit number of input device</td>
</tr>
<tr>
<td>IO2</td>
<td>I</td>
<td>IO2</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>Current tolerance for checking convergence or multiplicity</td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
<td>----------------------------</td>
<td>------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>RA,VA</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>RB,VB</td>
<td>D</td>
<td>R</td>
<td>Array of coefficients of deflated polynomial</td>
</tr>
<tr>
<td>NDEF</td>
<td>I</td>
<td>NDEF</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</td>
<td>I</td>
<td>E</td>
<td>Number of coefficients of polynomial</td>
</tr>
<tr>
<td>XO</td>
<td>C</td>
<td>RXO, VXO</td>
<td>D</td>
<td>E</td>
<td>Point ( (X_n) ) at which to evaluate the polynomial and its derivative. Also current approximation ( (X_{n+1}) ) used to deflate the polynomial</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>RPX, VPX</td>
<td>D</td>
<td>R</td>
<td>Value of polynomial at ( X_n )</td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX, VDPX</td>
<td>D</td>
<td>R</td>
<td>Value of the derivative of polynomial at ( X_n )</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>RC, VC</td>
<td>D</td>
<td>R</td>
<td>Array of containing sequence of values leading to the derivative</td>
</tr>
<tr>
<td>EPSLON</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
<td>C</td>
<td>Tolerance for convergence or multiplicity check</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>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX, VDPX</td>
<td>D</td>
<td>E</td>
<td>Derivative of ( P(X) ) at ( X_n )</td>
</tr>
<tr>
<td>XO</td>
<td>C</td>
<td>RXO, VXO</td>
<td>D</td>
<td>E</td>
<td>Current approximation ( (X_n) ) to root</td>
</tr>
<tr>
<td>IO2</td>
<td>I</td>
<td>IO2</td>
<td>I</td>
<td>C</td>
<td>New approximation ( (X_{n+1}) ) to root</td>
</tr>
<tr>
<td>KANS</td>
<td>I</td>
<td>KANS</td>
<td>I</td>
<td>R</td>
<td>KANS = 1 implies convergence, KANS = 0 implies no convergence</td>
</tr>
<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>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td>C</td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>XZERO</td>
<td>C</td>
<td>RXZERO, VXZERO</td>
<td>D</td>
<td>E</td>
<td>Array of approximations</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>RX, VX</td>
<td>D</td>
<td>ECR</td>
<td>Array of roots</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>RA, VA</td>
<td>D</td>
<td>E</td>
<td>Coefficients of original (undeflated) polynomial, P(X)</td>
</tr>
<tr>
<td>COEF</td>
<td>C</td>
<td>RCOEF, VCOEF</td>
<td>D</td>
<td>E</td>
<td>Working array for coefficients of polynomial</td>
</tr>
<tr>
<td>NA</td>
<td>I</td>
<td>NA</td>
<td>I</td>
<td>E</td>
<td>Number of coefficients of original polynomial</td>
</tr>
<tr>
<td>X0</td>
<td>C</td>
<td>RX0, VX0</td>
<td>D</td>
<td>E</td>
<td>Current approximation (X_n) to root</td>
</tr>
<tr>
<td>DPX</td>
<td>C</td>
<td>RDPX, VDPX</td>
<td>D</td>
<td>E</td>
<td>Derivative of P(X) at X_n</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>
</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>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td></td>
<td>Counter for number of iterations</td>
</tr>
<tr>
<td>XNEW</td>
<td>C</td>
<td>RXNEW, VXNEW</td>
<td>D</td>
<td>E</td>
<td>New approximation (X_{n+1}) to root</td>
</tr>
<tr>
<td>NN</td>
<td>I</td>
<td>NN</td>
<td>I</td>
<td></td>
<td>Degree of polynomial</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>RC, VC</td>
<td>D</td>
<td>E</td>
<td>Array containing the sequence of values leading to the derivative</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td>E</td>
<td>Number of distinct roots of P(X) found</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
<td>Degree of polynomial P(X)</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>RB, VB</td>
<td>D</td>
<td>E</td>
<td>Array of coefficients of deflated polynomial</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td>C</td>
<td>Maximum number of iterations permitted</td>
</tr>
<tr>
<td>EPSCHK</td>
<td>R</td>
<td>EPS</td>
<td>D</td>
<td></td>
<td>Tolerance for checking convergence</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>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
<td>E</td>
<td>Number of initial approximations to be generated</td>
</tr>
</tbody>
</table>
### TABLE B. VII (Continued)

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

Subroutine ALTER

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

Subroutine COMSQT

| UX, VX                    | D    | E                       | Complex number for which the square root is desired |
| UY, VY                    | D    | R                       | Square root of the complex number |
Figure 8.6. Flow Charts for Newton's Method
Figure B.6. (Continued)
Figure B.6. (Continued)
Figure B.6. (Continued)
Figure 3.5. (Continued)
Figure B.6. (Continued)
TABLE B. VIII

PROGRAM FOR NEWTON'S METHOD

C **********************************************************************
C * DOUBLE PRECISION PROGRAM FOR NEWTON'S METHOD *
C * NEWTON'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A *
C * POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX- *
C * IMATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION *
C * FORMULA *
C * X(N+1) = X(N) - P(X(N))/P'(X(N)). *
C **********************************************************************
C**************************************************************************
C 0001 DOUBLE PRECISION RA,VA,RXZERO,VXZERO,RA,V,RKOEFC,VCOEF,RK,RX,RXINI
C 0002 DIMENSION RA(26),VA(26),RXZERO(25),VXZERO(25),RX(25),VX(25)
C 0003 COMMON EPSCHK,MAX,102
C 0004 101=5
C 0005 102=6
C 0006 READ(101,10001) N,POLYN,N,NIAP,MAX,EPSCNV,EPSPMUL,XSTART,XEND,KC
C 0007 IF(KCHECK.EQ.1) STOP
C 0008 NA=N+1
C 0009 CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
C 0010 READ(I01,1010) (RXZERO(I),VXZERO(I),I=1,NIAP)
C 0011 WRITE(I02,1020) (I,RA(I),VA(I),=1,NA)
C 0012 WRITE(I02,1030) (I,RA(I),VA(I),I=1,NA)
C 0013 WRITE(I02,2060) NIAP
C 0014 WRITE(I02,2000) MAX
C 0015 WRITE(I02,2020) EPSCNV
C 0016 WRITE(I02,2030) EPSMUL
C 0017 WRITE(I02,2040) XSTART
C 0018 WRITE(I02,2050) XEND
C 0019 IF(NIAP.NE.0) GO TO 3
C 0020 NIAP=0
C 0021 IF(NIAP.NE.0) GO TO 3
C 0022 GO TO 4
C 0023 3 READ(I01,1020) (RXZERO(I),VXZERO(I),I=1,NIAP)
C 0024 4 NIAP=0
C 0025 L=1
C 0026 ITER=0
C 0027 NR=0
C 0028 IROOT=0
C 0029 ITIME=0
C 0030 ND=0
C 0031 IALTER=0
C 0032 K=0
C 0033 RX=RXZERO(L)
C 0034 VX=VXZERO(L)
C 0035 DO 5=1,NIAP
C 0036 5 R=HORNER(RX,VX)
C 0037 VX=RX
C 0038 10 CALL HORNERR(RKOEFC,VCOEF,RX,VX,NDEF,RK,V,RKOEFC,VCOEF,RX,VX,NDEF,RK,V)
C 0039 IF(NIAP.NE.0) GO TO 3
C 0040 IF(NIAP.NE.0) GO TO 3
C**************************************************************************
C**************************************************************************
TABLE B. VIII (Continued)

```
0041 IF(ABOPX NE.0.0) GO TO 20
0042 IF(ABPX.EQ.0.0) GO TO 70
0043 GO TO 110
0044 20 CALL NEWTON(RPX,VPX,RPDX,VPDX,RPXO,VPXO,RPXNEW,VPXNEW)
0045 ITER=ITER+1
0046 RXO=RXNEW
0047 VXO=VXNEW
0048 EPSCHK=EPSCHV
0049 CALL CHECK(RPX,VPX,RPDX,VPDX,RPXO,VPXO,KANS)
0050 IF(KANS.EQ.1) GO TO 70
0051 IF(ITER.GE.MAX) GO TO 40
0052 GO TO 10
0053 40 CALL ALTER(RXZERO,L),VXZERO(L),IALTER,ITIME)
0054 IF(IALTER.GT.MAX) GO TO 110
0055 RXD=RXZERO(L)
0056 VX=VXZERO(L)
0057 ITER=0
0058 GO TO 10
0059 60 NDEF=1
0060 GO 65 J=1,N
0061 RD(J)=RCOEF(J)
0062 65 VD(J)=VCORF(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)=RXZERO(L)
0070 VXINIT(K)=VXZERO(L)
0071 CALL HORNER(RCOEF,VCORF,RXO,VPX,NDEF,R8,RB,RC,RPX,VPX,RPDX,VPDX)
0072 80 IF(NROOT.GE.N) GO TO 147
0073 NDEF=NDEF-1
0074 NUM=NUM+1
0075 DO 105 I=1,NUM
0076 RCOEF(I)=RB(I)
0077 105 NUM=NUM-1
0078 CALL HORNER(RCOEF,VCORF,RXO,VPX,NDEF,RPX,VPX,RPDX,VPDX)
0079 11 ABOPX=DSQRT(RPX*RPX+VPX*VPX)
0080 ABPX=DSQRT(RPX*RPX+VPX*VPX)
0081 IF(ABPX.EQ.0.0) GO TO 107
0082 IF(ABOPX.EQ.0.0) GO TO 130
0083 GO TO 110
0084 107 CONTINUE
0085 EPSCHV=EPSCHV
0086 CALL CHECK(RPX,VPX,RPX,VPX,RPXO,VPXO,KANS)
0087 IF(KANS.EQ.11) GO TO 130
0088 110 IF(NDEF.GT.2) GO TO 113
0089 IKOUT=K
0090 CALL QUAD(RCOEF,VCORF,NDEF,RPX,VPX,K,MULT,EPSON)
0091 GO TO 150
0092 113 IF(LALT,NIAP) GO TO 115
0093 IF(LXEND.EQ.0.0) GO TO 60
0094 IF(LSTART.EQ.NXEND) GO TO 60
0095 NIAP=NIAP
0096 CALL GENAPP(RXZERO,VXZERO,NIAP,XSTART)
```
TABLE B. VIII (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0097</td>
<td>L=0</td>
</tr>
<tr>
<td>0098</td>
<td>L=L+1</td>
</tr>
<tr>
<td>0099</td>
<td>RX=RXZERO(L)</td>
</tr>
<tr>
<td>0100</td>
<td>VX=VXZERO(L)</td>
</tr>
<tr>
<td>0101</td>
<td>ITER=0</td>
</tr>
<tr>
<td>0102</td>
<td>IALTER=0</td>
</tr>
<tr>
<td>0103</td>
<td>GO TO 10</td>
</tr>
<tr>
<td>0104</td>
<td>130 MULT(K)=MULT(K)+1</td>
</tr>
<tr>
<td>0105</td>
<td>NROOT=NROOT+1</td>
</tr>
<tr>
<td>0106</td>
<td>GO TO 80</td>
</tr>
<tr>
<td>0107</td>
<td>IF(P(K,EQ.0)) GO TO 160</td>
</tr>
<tr>
<td>0108</td>
<td>IK=IROOT</td>
</tr>
<tr>
<td>0109</td>
<td>WRITE(I02,1025)</td>
</tr>
<tr>
<td>0110</td>
<td>WRITE(I02,1050)</td>
</tr>
<tr>
<td>0111</td>
<td>WRITE(I02,1060)</td>
</tr>
<tr>
<td>0112</td>
<td>RXVX=RXZERO(L),VR=VXZERO(L)</td>
</tr>
<tr>
<td>0113</td>
<td>VX=VXZERO(L),VR=VXZERO(L)</td>
</tr>
<tr>
<td>0114</td>
<td>ITER=0</td>
</tr>
<tr>
<td>0115</td>
<td>IALTER=0</td>
</tr>
<tr>
<td>0116</td>
<td>GO TO 80</td>
</tr>
<tr>
<td>0117</td>
<td>IF(P(K,EQ.0)) GO TO 160</td>
</tr>
<tr>
<td>0118</td>
<td>RX=RXZERO(L),VR=VXZERO(L)</td>
</tr>
<tr>
<td>0119</td>
<td>VX=VXZERO(L),VR=VXZERO(L)</td>
</tr>
<tr>
<td>0120</td>
<td>ITER=0</td>
</tr>
<tr>
<td>0121</td>
<td>IALTER=0</td>
</tr>
<tr>
<td>0122</td>
<td>GO TO 80</td>
</tr>
<tr>
<td>0123</td>
<td>IF(P(K,EQ.0)) GO TO 160</td>
</tr>
<tr>
<td>0124</td>
<td>RX=RXZERO(L),VR=VXZERO(L)</td>
</tr>
<tr>
<td>0125</td>
<td>VX=VXZERO(L),VR=VXZERO(L)</td>
</tr>
<tr>
<td>0126</td>
<td>ITER=0</td>
</tr>
<tr>
<td>0127</td>
<td>IALTER=0</td>
</tr>
<tr>
<td>0128</td>
<td>GO TO 80</td>
</tr>
<tr>
<td>0129</td>
<td>1000 FORMAT(3I12,(13.16,3H+D23.16,2H)</td>
</tr>
<tr>
<td>0130</td>
<td>1010 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0131</td>
<td>1020 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0132</td>
<td>1030 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0133</td>
<td>1040 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0134</td>
<td>1050 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0135</td>
<td>1060 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0136</td>
<td>1070 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0137</td>
<td>1080 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0138</td>
<td>1090 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0139</td>
<td>1100 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0140</td>
<td>1110 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0141</td>
<td>1120 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0142</td>
<td>1130 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0143</td>
<td>1140 FORMAT(1H,12,4H)</td>
</tr>
<tr>
<td>0144</td>
<td>1150 FORMAT(1H,12,4H)</td>
</tr>
</tbody>
</table>

END
TABLE B. VIII (Continued)

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

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

 0002 DOUBLE PRECISION APP1,APP2,XSTART,DUMMY,BETA

 0003 DIMENSION APP1(25),APP2(25)

 0004 COMMON DUMMY,MAX,102

 0005 IF (XSTART.EQ.0.0) XSTART=0.5

 0006 BETA=0.2617994

 0007 DO 10 =1,NAPP

 0008 APP1(I)=XSTART*COS(BETA)

 0009 APP2(I)=XSTART*SIN(BETA)

 0010 BETA=BETA+0.5235988

 0011 10 XSTART=XSTART+0.5

 0012 RETURN

 0013 END

0001 SUBROUTINE ALTER(XOLDI,XOLDI1,NALTER,TIME)

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

 0002 DOUBLE PRECISION XOLDI,XOLDI1,DUMMY,ABXOLD,BETA

 0003 COMMON DUMMY,MAX,102

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

 0005 TIME =1

 0006 WRITE(102,1010) MAX

 0007 5 IF (NALTER.EQ.0) GO TO 10

 0008 WRITE(102,1000) XOLDI,XOLDI1

 0009 GO TO 20

 0010 10 ABXOLD=DSQRT(XOLDI*XOLDI1)

 0011 BETA=DAYANZ(XOLDI1,XOLDI)

 0012 WRITE(102,1020) XOLDI1,XOLDI

 0013 20 NALTER=NALTER+1

 0014 IF (NALTER.GT.5) RETURN

 0015 GO TO 30

 0016 30 XOLDI=XOLDI1

 0017 XOLDI1=XOLDI

 0018 GO TO 50

 0019 40 BETA=BETA+1.0471976

 0020 XOLDI=ABXOLD*COS(BETA)

 0021 XOLDI1=ABXOLD*SIN(BETA)

 0022 50 RETURN

 0023 1000 FORMAT(1X,23.16,3H + ,023.16,2H 1,10X,21HALTERED APPROXIMATION)

 0024 1010 FORMAT(1X,64H AND CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF

 0025 1TER =13,12H ITERATIONS//)

 0026 1020 FORMAT(1X,23.16,3H + ,023.16,2H 1,10X,21INITIAL APPROXIMATION)

 0026 END
SUBROUTINE QUAD(UA, VA, UROOT, VROOT, NROOT, MULTI, EPST)

DOUBLE PRECISION UA, VA, UROOT, VROOT, BBB, UAAA, VAAA, UDISC, VDISC, UDummy
DIMENSION UA(26), VA(26), UROOT(25), VROOT(25), MULTI(25)

IF(INA.EQ.2) GO TO 7
IF(NA.EQ.1) GO TO 5
UROOT(1) = 0.0
VROOT(1) = 0.0
MULTI(1) = 1
NROOT = NROOT + 1
GO TO 50

588 = UA(1) * UA(1) + VA(1) * VA(1)
UROOT(NROOT) = 1 - 588 * UA(1) * UA(1) / 588
VROOT(NROOT) = 1 - VA(1) * VA(1) / 588
MULTI(NROOT) = 1
NROOT = NROOT + 1
GO TO 50

UDISC = (UA(2) * UA(1) + VA(2) * VA(1))
VDISC = (UA(1) * UA(1) + VA(1) * VA(1))
BBB = DSQRT(UDISC * UDISC + VDISC * VDISC)
IF(OBB.LT.EPST) GO TO 10
CALL COMSQT(UDISC, VDISC, UDummy, VDummy)
UBB = UA(1) + UDummy
VBB = VA(1) + VDummy
RDummy = -UA(1)*UA(1) - UDummy
SDummy = -VA(1)*VA(1) - VDummy
UAAA = 2.0 * UA(1)
VAAA = 2.0 * VA(1)
BBB = UAAA * UAAA + VAAA * VAAA
UBB = UBB * UAAA + VBB * VAAA / UBB
VRDOT(NROOT) = 2 = (RDummy*UAAA + SDummy*VAAA) / UBB
MULTI(NROOT) = 2
NROOT = NROOT + 1
GO TO 50

UBBE = UA(1) + UDummy
VBBE = VA(1) + VDummy
ROUMMY = UA(1) - UDummy
SLOUMMY = VA(1) - VDummy
UAAA = 2.0 * UA(1)
VAAA = 2.0 * VA(1)
BBB = UAAA * UAAA + VAAA * VAAA
UBB = UBB * UAAA + VBB * VAAA / UBB
VRDOT(NROOT) = 2 = (UDummy*UAAA + VDummy*VAAA) / UBB
MULTI(NROOT) = 1
NROOT = NROOT + 1
GO TO 50

UBBE = UA(1) + UDummy
VBBE = VA(1) + VDummy
ROUMMY = UA(1) - UDummy
SLOUMMY = VA(1) - VDummy
UAAA = 2.0 * UA(1)
VAAA = 2.0 * VA(1)
BBB = UAAA * UAAA + VAAA * VAAA
UBB = UBB * UAAA + VBB * VAAA / UBB
VRDOT(NROOT) = 2 = (SLOummy*UAAA + ROUMMY*VAAA) / UBB
MULTI(NROOT) = 1
NROOT = NROOT + 1
GO TO 50

RETURN
END
TABLE B. VIII (Continued)

SUBROUTINE COMSQTIUXtVXiUYsVYI

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

DOUBLE PRECISION UXVXeUYeVYDUMMYR,AAA, BBB
R=DSQRTIUX*UX+VX*VX)
AAA=DSQRT(DABS((IRUXI/2.01
BBB=OSQRTIDABS4R-UX/2.O|)
IFIVXI
10,20,30
UY=AAA
VY--t.O*BBB
GO TO 100
20 IF(UXI
40,50,60
UY=AAA
VY=BBB
GO TO 100
40 DUMMY=DABS(UX)
UY=O.O
VY=DSQRT(DUMMY)
GO TO 100
50 UY=O.O
GO TO 100
60 DUMMY=DABS(UX)
UY=DSQRT(DUMMY)
100 RETURN
END
TABLE B. VIII (Continued)

0001 SUBROUTINE HORNERRA,RXO,VXO,NDEF,RB,VB,VC,RPX,VDPX,RDPX,VPX
C *********************************************************************
C * HORNERS METHOD COMPUTES THE VALUE OF A POLYNOMIAL PIXI AT A POINT D AND *
C * ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO DEFLATE THE *
C * POLYNOMIAL BY DIVIDING OUT THE FACTOR (X-D). *
C *********************************************************************
C DO 10 I=2,NDEF
C 10 RXNEW=RXO+(RXNEW-RXO)*VXO/NUM
C RETURN
END

0002 SUBROUTINE NEWTON(RPX,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW)
C *********************************************************************
C * THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-
C * MATION BY USING THE ITERATION FORMULA *
C * X(N+1) = X(N) - PI(X(N))/P'(X(N)). *
C *********************************************************************
C DO 10 I=2,NDEF
C 10 RXNEW=RXO-(RPX*RDPX+VPX*VDPX)/ARG
C RETURN
END

0003 DOUBLE PRECISION RXO,VPX,RDPX,VDPX,RXO,VXO,RXNEW,VXNEW,ARG
C *********************************************************************
C DO 10 I=2,NDEF
C 10 RXNEW=RXO-(RPX*RDPX+VPX*VDPX)/ARG
C RETURN
END

0004 DOUBLE PRECISION RXO,VPX,RDPX,VDPX
C *********************************************************************
C DO 10 I=2,NDEF
C 10 RXNEW=RXO-(RPX*RDPX+VPX*VDPX)/ARG
C RETURN
END

0005 DOUBLE PRECISION RXO,VPX,RDPX,VDPX
C *********************************************************************
C DO 10 I=2,NDEF
C 10 RXNEW=RXO-(RPX*RDPX+VPX*VDPX)/ARG
C RETURN
END

0006 DOUBLE PRECISION RXO,VPX,RDPX,VDPX
C *********************************************************************
C DO 10 I=2,NDEF
C 10 RXNEW=RXO-(RPX*RDPX+VPX*VDPX)/ARG
C RETURN
END

0007 DOUBLE PRECISION RXO,VPX,RDPX,VDPX
C *********************************************************************
C DO 10 I=2,NDEF
C 10 RXNEW=RXO-(RPX*RDPX+VPX*VDPX)/ARG
C RETURN
END

0008 DOUBLE PRECISION RXO,VPX,RDPX,VDPX
C *********************************************************************
C DO 10 I=2,NDEF
C 10 RXNEW=RXO-(RPX*RDPX+VPX*VDPX)/ARG
C RETURN
END

0009 DOUBLE PRECISION RXO,VPX,RDPX,VDPX
C *********************************************************************
C DO 10 I=2,NDEF
C 10 RXNEW=RXO-(RPX*RDPX+VPX*VDPX)/ARG
C RETURN
END
TABLE E. VIII (Continued)

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

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

DOUBLE PRECISION RPX, VPX, RDPX, VDPX, RXO, VXO, ABSXO, ABSQUO, RDUMMY, YOU
IMAY, EPS
DOUBLE PRECISION ARG
COMMON EPS, MAX, K02
ABSXO=DSQRT(RXO*RXO+VXO*VXO)
IF(ABSXO.EQ.0.) GO TO 25
ARG=RPX*RPX+VDPX*VDPX
DOD=DSQRT(ARG)
IF(DOD.EQ.0.) GO TO 25
RDUMMY=(RPX*RDPX+VPX*VDPX)/ARG
VODMMY=(VPX*RDPX-RPX*VDPX)/ARG
ABSQUO=DSQRT(RDUMMY*RDUMMY+VODMMY*VODMMY)
IF(ABSQUO/ABSXO.LT.EPS) GO TO 10
KANS=0
RETURN
10 KANS=1
RETURN
25 KANS=0
RETURN
END
TABLE B. VIII (Continued)

SUBROUTINE BETTER(K, RXZERO, VXZERO, RX, VX, RA, RA, RC, RC, VC, RC, VB, VB)

***************************************************************************
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, RA, RA, RC, RC, VC, RC, VB, VB
DIMENSION RXZERO(25), VXZERO(25), RX(25), VX(25), RA(26), VA(26), RC(26), VC(26), VB(26)
DOUBLE PRECISION EPSMAX
DO 10 I = 1, K
RXZERO(I) = RX(I)
VXZERO(I) = VX(I)
DO 20 I = 1, N
RCOEFF(I) = RA(I)
DO 50 J = 1, K
RXO = RXZERO(J)
VXO = VXZERO(J)
ITER = 0
30 CALL HORNER(RCOEFF, VC, RX, VB, VB, RX, VX, VB, RX, VX, VB, VB)
ABPX = DSQRT(RPX*RPX + VDPX*VDPX)
ABDPPX = DSQRT(VDPX*VDPX + VDPX*VDPX)
IF(ABOPX .NE. 0.0) GO TO 33
IFIABPX.EQ.0.0) GO TO 40
33 CALL NEWTON(RPX, VDPX, RXNEW, VXNEW)
ITER = ITER + 1
RXO = RXNEW
VXO = VXNEW
DO 50 J = 1, K
34 WRITE(102, 1112) RXO, VXO
DO 50 J = 1, K
35 WRITE(102, 1101) J, RXZERO(J), VXZERO(J)
36 CONTINUE
RETURN
1112 FORMAT(36H THE VALUE OF THE DERIVATIVE AT XO = , D2, 16.3H + , D2
13.16.3H = , D2)
100 FORMAT(42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(, D2, 16.3H = , D2
1.16.3H + , D2, 16.2H IN 1.16.1H DID NOT CONVERGE.)
200 FORMAT(33H THE PRESENT APPROXIMATION AFTER , 13.29H ITERATIONS IS P
INTED BELOW.)
END

TABLE B. VIII (Continued)

0001 SUBROUTINE BETTER(K, RXZERO, VXZERO, RX, VX, RA, RA, RC, RC, VC, RC, VB, VB)
0002.constructor.
0003 DOUBLE PRECISION RXZERO, VXZERO, RX, VX, RA, RA, RC, RC, VC, RC, VB, VB
0004 COMMON EPSMAX
0005 DIMENSION RXZERO(25), VXZERO(25), RX(25), VX(25), RA(26), VA(26), RC(26), VC(26), VB(26)
0006 DOUBLE PRECISION EPSMAX
0007 DO 10 I = 1, K
0008 RXZERO(I) = RX(I)
0009 VXZERO(I) = VX(I)
0010 DO 20 I = 1, N
0011 RCOEFF(I) = RA(I)
0012 DO 50 J = 1, K
0013 RXO = RXZERO(J)
0014 VXO = VXZERO(J)
0015 NN = N
0016 ITER = 0
0017 30 CALL HORNER(RCOEFF, VC, RX, VB, NN, RB, VB, RC, RX, VX, VB, VB, VB)
0018 ABPX = DSQRT(RPX*RPX + VDPX*VDPX)
0019 ABDPPX = DSQRT(VDPX*VDPX + VDPX*VDPX)
0020 IF(ABOPX .NE. 0.0) GO TO 33
0021 IF(IFIABPX.EQ.0.0) GO TO 40
0022 33 CALL NEWTON(RPX, VDPX, RXNEW, VXNEW)
0023 ITER = ITER + 1
0024 RXO = RXNEW
0025 VXO = VXNEW
0026 DO 50 J = 1, K
0027 CALL CHECK(RPX, VDPX, RX, VX, RXNO, RXKANS)
0028 IF(KANS .EQ. 11) GO TO 40
0029 IF(ITER .GE. MAX) GO TO 35
0030 GO TO 30
0031 34 WRITE(102, 1112) RXO, VXO
0032 35 WRITE(102, 1101) J, RXZERO(J), VXZERO(J)
0033 36 CONTINUE
0034 40 RXO = RXO
0035 VXO = VXO
0036 50 CONTINUE
0037 RETURN
0038 1112 FORMAT(36H THE VALUE OF THE DERIVATIVE AT XO = , D2, 16.3H + , D2
13.16.3H = , D2)
0039 100 FORMAT(42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(, D2, 16.3H = , D2
1.16.3H + , D2, 16.2H IN 1.16.1H DID NOT CONVERGE.)
0040 200 FORMAT(33H THE PRESENT APPROXIMATION AFTER , 13.29H ITERATIONS IS P
INTED BELOW.)
0041 END
APPENDIX C

MULLER'S METHOD

2. Use of the Program

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

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

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

Double Precision

Main Program

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

Subroutine BETTER

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

Subroutine GENAPP

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

Subroutine HORNER

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

Subroutine QUAD

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

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

SYSTEM FUNCTIONS USED IN MULLER'S METHOD

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

2. Input Data for Muller's Method

The input data for Muller's method is identical to the input data for Newton's method as described in Appendix B, § 2 except for the variable names. The correspondence of input variable names is given in Table C.III. Only one (not three) initial approximation, $X₀$, is given for each root. The other two required by Muller's method are constructed within the program and are $.9X₀$ and $1.1X₀$.

3. Variables Used in Muller's Method

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

<table>
<thead>
<tr>
<th>Newton's Method</th>
<th>Muller's Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><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 (VAPP)     |

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

4. Description of Program Output

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

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

"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX."

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

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

X is the number of the root before the attempt to improve accuracy, YYY is the value of the root before attempt to improve accuracy, ZZZ is the maximum number of iterations, and AAA is the current approximation after the maximum number of iterations. This message has the same meaning as the corresponding message in Appendix B, § 5.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Type</th>
<th>Disposition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>I</td>
<td>NP</td>
<td>I</td>
<td>Degree of polynomial P(X)</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</td>
<td>I</td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>NOMULT</td>
<td>I</td>
<td>NOMULT</td>
<td>I</td>
<td>Number of roots (counting multiplicities)</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT,VROOT</td>
<td>D</td>
<td>Array containing the roots</td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</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>Array of initial approximations</td>
</tr>
<tr>
<td>WORK</td>
<td>C</td>
<td>UWORK,VWORK</td>
<td>D</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>Array containing coefficients of deflated polynomial</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA,VA</td>
<td>D</td>
<td>Array containing coefficients of original polynomial, P(X)</td>
</tr>
<tr>
<td>RAPP</td>
<td>C</td>
<td>URAPP,VRAAPP</td>
<td>D</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>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>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>One of three current approximations to a root</td>
</tr>
<tr>
<td>PX1</td>
<td>C</td>
<td>UPX1,VPX1</td>
<td>D</td>
<td>Value of polynomial P(X) at X1</td>
</tr>
<tr>
<td>PX2</td>
<td>C</td>
<td>UPX2,VPX2</td>
<td>D</td>
<td>Value of polynomial P(X) at X2</td>
</tr>
<tr>
<td>PX3</td>
<td>C</td>
<td>UPX3,VPX3</td>
<td>D</td>
<td>Value of polynomial P(X) at X3</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4,VX4</td>
<td>D</td>
<td>Newest approximation (Xn+1) to root</td>
</tr>
<tr>
<td>PX4</td>
<td>C</td>
<td>UPX4,VPX4</td>
<td>D</td>
<td>Value of polynomial P(X) at X4</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td>Array containing the multiplicities of each root found</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>ITER</td>
<td>I</td>
<td>Counter for iterations</td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
<td>Unit number of input device</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</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>Type</th>
<th>Double Precision Variable 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>R</td>
<td>Array containing the coefficients of the deflated polynomial</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>U PX, VP X</td>
<td>R</td>
<td>Value of the polynomial at X</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>NUM</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>E</td>
<td>Approximation to Root (old) (Xₙ)</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4, VX4</td>
<td>E</td>
<td>New approximation to root (Xₙ+1)</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</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>C</td>
<td>Tolerance for convergence test</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
<td>C</td>
<td>Tolerance check for zero (0)</td>
</tr>
<tr>
<td>DENOM</td>
<td>R</td>
<td>DENOM</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>Array of multiplicities of each root</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA, VA</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>E</td>
<td>Degree of original polynomial</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT, VROOT</td>
<td>ECR</td>
<td>Array of roots</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>NROOT</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>E</td>
<td>Array of initial approximations (old roots)</td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</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></td>
<td>Temporary storage for new (better) roots</td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>L</td>
<td>I</td>
<td>Number of roots found by BETTER</td>
</tr>
<tr>
<td>EPSRT</td>
<td>R</td>
<td>EPSRT</td>
<td>D</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>C</td>
<td>Counter for number of iterations</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Single Precision Variable</td>
<td>Type</td>
<td>Disposition of Argument</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------</td>
<td>----------------------------</td>
<td>------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB, VB</td>
<td>D</td>
<td>Array containing coefficients of deflated polynomial</td>
</tr>
<tr>
<td>X1</td>
<td>C</td>
<td>UX1, VX1</td>
<td>D</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>One of three approximations to the root</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3, VX3</td>
<td>D</td>
<td>One of three approximations to the root</td>
</tr>
<tr>
<td>PX1</td>
<td>C</td>
<td>UPX1, VPX1</td>
<td>D</td>
<td>Value of polynomial (P(X)) at (X_1)</td>
</tr>
<tr>
<td>PX2</td>
<td>C</td>
<td>UPX2, VPX2</td>
<td>D</td>
<td>Value of polynomial (P(X)) at (X_2)</td>
</tr>
<tr>
<td>PX3</td>
<td>C</td>
<td>UPX3, VPX3</td>
<td>D</td>
<td>Value of polynomial (P(X)) at (X_3)</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
<td>L</td>
<td>(\text{CONV} = \text{true} \Rightarrow \text{convergence has been obtained})</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4, VX4</td>
<td>D</td>
<td>Newest approximation to root</td>
</tr>
<tr>
<td>J</td>
<td>I</td>
<td>J</td>
<td>I</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>Maximum number of iterations permitted</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td>Unit number of output device</td>
</tr>
</tbody>
</table>

Subroutine ALTER

<table>
<thead>
<tr>
<th>variable</th>
<th>type</th>
<th>variable</th>
<th>type</th>
<th>disposition</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>C</td>
<td>XLR, XLI</td>
<td>D</td>
<td>ECR</td>
<td>One of the three approximations to be altered</td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>X2R, X2I</td>
<td>D</td>
<td>ECR</td>
<td>One of the three approximations to be altered</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>X3R, X3I</td>
<td>D</td>
<td>ECR</td>
<td>One of the three approximations to be altered</td>
</tr>
<tr>
<td>X2R</td>
<td>R</td>
<td>X2R</td>
<td>D</td>
<td></td>
<td>Real part of complex approximation</td>
</tr>
<tr>
<td>X2I</td>
<td>R</td>
<td>X2I</td>
<td>D</td>
<td></td>
<td>Imaginary part of complex approximation</td>
</tr>
</tbody>
</table>

Subroutine QUAD

<table>
<thead>
<tr>
<th>variable</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPST</td>
<td>R</td>
<td>Tolerance check for zero (0)</td>
</tr>
</tbody>
</table>

Subroutine CALC

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

Figure C.1. Flow Charts for Muller's Method
Figure C.1. (Continued)
Figure C.1. (Continued)
Figure C.1. (Continued)
Figure C.1. (Continued)
Figure C.1. (Continued)
TABLE C.V.

PROGRAM FOR MULLER'S METHOD

******************************************************************************
** DOUBLE PRECISION PROGRAM FOR MULLER'S METHOD                           **
******************************************************************************
** MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLEITIES OF A          **
** POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE         **
** POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZEROS OF THE QUADRATIC    **
** CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.     **
******************************************************************************

0001 DOUBLET PRECISION UPX3,UPX2,UPX1,UP1,UP4,UP3,UP2,UP1,XV3,XV2,XV1,U3,U2,U1,U0,MAPP,APP
  UPX1=UPX2,UPX2=UPX3,UPX1=UPX2,UPX1=UPX3,UPX2=UPX3,UPX2=UPX3,UPX1=UPX2,UPX1=UPX3,UPX2=UPX3
  EPSF<0.999
  NARGT=0
  IRGOT=0
  IPATH=1
  NOPULT=0
  NALTER=0
  NTIME=0
  NAPP=1
  NTER=1
  READ1(1),1000) NOPOLY,NP,NAPP,MAX,EPS,EPSON,XSTART,XEND,KCHECK
  IF(KCHECK.EQ.11) STOP
  KK=NP+1
  READ1(1),1010) (UA(I),VA(I),I,KK)
  WRITE(102,1020) NOPOLY,NP
  WRITE(102,1035) (PNAME,UA(I),VA(I),I,KK)
  WRITE(102,2060)
  WRITE(102,2000) NAPP
  WRITE(102,2010) MAX
  WRITE(102,2020) EPS
  WRITE(102,2030) EPSM
  WRITE(102,2040) XSTART
  WRITE(102,2050) XEND
  IF(NP.GT.2) GO TO 15
  CALL QUAD(UA,VA,NP,VRGOT,VRGOT,NARGT,MULT,EPSON)
  WRITE(102,1037)
  WRITE(102,1096) (I,VRGOT(I),VRGOT(I),MULT(I),I,NARGT)
  GO TO 10
  IF(NAPP:.NE.0) GO TO 20
  NAPP=N
  CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
  GO TO 27
  READ1(1),1030) (UAPP(I),2),VAPP(I),2),I,NAPP)
  DO 25 I=1,NAPP
  UAPP(I)=UAPP(I)+9*VAPP(I)
  VAPP(I)=0.9*VAPP(I)
  20  }
  VAPP(I)=0.9*VAPP(I)
TABLE C.V (Continued)

0043  UAPP(I,3)=1.1*UAPP(I,2)
0044  VAPP(I,3)=1.1*VAPP(I,2)
0045  27 KKK=NP+1
0046  DO 30 I=1,KKK
0047  UWORK(I)=UA(I)
0048  30 VWORK(I)=VA(I)
0049  00  NWORK=NP
0050  00  40 UI=UAPP(IAPP,I)
0051  00  VK1=VAPP(IAPP,I)
0052  00  UX2=UAPP(IAPP,2)
0053  00  VX2=VAPP(IAPP,2)
0054  00  UX3=UAPP(IAPP,3)
0055  00  VX3=VAPP(IAPP,3)
0056  CALL HORNER(UWORK,VWORK,UX1,VX1,UX2,VX2,UX3,VX3)
0057  CALL HORNER(UWORK,VWORK,UX4,VX4)
0058  CALL HORNER(UWORK,VWORK,UX5,VX5,UX6,VX6)
0059  50 CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UX4,VX4,UX5,VX5,UX6,VX6)
0060  60 CALL HORNER(UWORK,VWORK,UX7,VX7,UX8,VX8)
0061  00  70 CALL TEST(UX3,VX3,UX4,VX4,UX5,VX5,UX6,VX6)
0062  00  IF ICONV GO TO 120
0063  00  IF ITER.LE.10 GO TO 70
0064  00  U34=0.5*UQ4
0065  00  V4=0.5*VQ4
0066  00  UX4=UX3+
0067  00  IF(UH3*UQ4+VH3*VQ4)
0068  00  VX4=VX3+
0069  00  IF(VH3*UQ4+UH3*VQ4)
0070  70 GO TO 60
0071  70 IF CONV GO TO 120
0072  00  IF ITER.LE.100 GO TO 110
0073  00  IF(XEND.EQ.0) GO TO 77
0074  00  CALL ALTER(UAPP,VAPP,NAPP,UX1,UX2,UX3,UX4,UX5,UX6)
0075  00  IF NALTER GT 5 GO TO 75
0076  00  ITER=1
0077  00  GO TO 40
0078  00  75 IF(IAPP.LT.NAPP) GO TO 100
0079  00  IF XEND.EQ.0 GO TO 77
0080  00  IF XSTART.GT.XEND) GO TO 77
0081  00  NAPP=NP
0082  00  CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0083  00  IAPP=0
0084  00  GO TO 100
0085  00  77 WRITE(102,1090)
0086  00  KAP=WORK+1
0087  00  WRITE(102,1035) (ONAME,J,UWORK(I),J=1,KKK)
0088  00  STRIDE=0.1
0089  00  WRITE(102,1060)
0090  00  IF STRIDE.EQ.1 GO TO 42
0091  00  IF(IAPP.LE.0) GO TO 90
0092  00  IF(NROOT.EQ.0) GO TO 85
0093  00  WRITE(102,1080)
0094  00  DO 55 I=1,NROOT
0095  00  55 WRITE(102,1085) (IROOT(I),VRGOT(I),MULT(I),UAPP(I,1),VAPP(I,1))
TABLE C.V. (Continued)

0099 IF(IROOT.LT.NROOT) GO TO 83
0100 GO TO 87
0101 85 KKK=IROOT+1
0102 WRITE(I02,1086) (I,UROOT(I),VROOT(I),MULT(I)=KKK,NROOT)
0103 87 IF(IPATH.EQ.1) GO TO 81
0104 GO TO 10
0105 90 WRITE(I02,1070) NOPOLY
0106 GO TO 10
0107 100 IAPP=IAPP+1
0108 ITER=1
0109 NALTER=0
0110 GO TO 40
0111 120 NROOT=NROOT+1
0112 IROOT=NROOT
0113 MULT(NROOT)=1
0114 NOMULT=NOMULT+1
0115 UROOT(NROOT)=UX4
0116 VROOT(NROOT)=VX4
0117 URAPP(NROOT,1)=UAPP(IAPP,1)
0118 VRAPP(NROOT,1)=VAPP(IAPP,1)
0119 URAPP(NROOT,2)=UAPP(IAPP,2)
0120 VRAPP(NROOT,2)=VAPP(IAPP,2)
0121 URAPP(NROOT,3)=UAPP(IAPP,3)
0122 VRAPP(NROOT,3)=VAPP(IAPP,3)
0123 125 IF(NOMULT.LT.NP) GO TO 130
0124 GO TO 80
0125 130 CALL HORNER(NWORK,UWORK,VWORK,UX4,UX4,UB,UB4,UPX4,VPX4)
0126 NWORK=NWORK-1
0127 KK=NWDRK+1
0128 GO 140 I=1, KK
0129 UWORK(I)=UB(I)
0130 VWORK(I)=VB(I)
0131 CALL HORNER(NWORK,UWORK,VWORK,UX4,UX4,UB,UB4,UPX4,VPX4)
0132 CCC=DSQRT(UPX4*UPX4+VPX4*VPX4)
0133 IFCCC.LT.EPSI GO TO 150
0134 IF(NWORK.GT.2) GO TO 75
0135 IROOT=NROOT
0136 CALL QUAD(UWORK,VWORK,NWORK,UROOT,VROOT,NROOT,MULT,EPSI)
0137 GO TO 80
0138 150 MULT(NROOT)=MULT(NROOT)+1
0139 NOMULT=NOMULT+1
0140 GO TO 125
0141 110 UX1=UX2
0142 VX1=VX2
0143 UX2=UX3
0144 VX2=VX3
0145 UX3=UX4
0146 VX3=VX4
0147 UPX1=UPX2
0148 VPX1=VPX2
0149 UPX2=UPX3
0150 VPX2=VPX3
0151 UPX3=UPX4
0152 VPX3=VPX4
0153 ITER=ITER+1
0154 GO TO 50
0155 1010 FORMAT(2030,0)
0156 1020 FORMAT(IH1,IX,52) MULLERS METHOD FOR FINDING THE ZEROS OF A POLYNOM
TABLE C.V (Continued)

III/IH, IX, 18HPOLYNOMIAL NUMBER, 12/IH, 11H OF DEGREE, 12/IH, 11H, 20H
THE COEFFICIENTS OF PI(X) ARE/

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1030</td>
<td>1090</td>
</tr>
<tr>
<td>1031</td>
<td>1091</td>
</tr>
<tr>
<td>1032</td>
<td>1092</td>
</tr>
<tr>
<td>1033</td>
<td>1093</td>
</tr>
<tr>
<td>1034</td>
<td>1094</td>
</tr>
<tr>
<td>1035</td>
<td>1095</td>
</tr>
<tr>
<td>1036</td>
<td>1096</td>
</tr>
<tr>
<td>1037</td>
<td>1097</td>
</tr>
<tr>
<td>1038</td>
<td>1098</td>
</tr>
<tr>
<td>1039</td>
<td>1099</td>
</tr>
</tbody>
</table>

END
TABLE C.V (Continued)

```plaintext
0001 SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)
0002 C**************************************************************************
0003 C* SUBROUTINE ALTER ALTERS THE INITIAL APPROXIMATIONS WHICH PRODUCE NO *
0004 C* CONVERGENCE TO A ZERO. THIS IS DONE A MAXIMUM OF 5 TIMES FOR EACH ROOT. *
0005 C*******************************************************************************
0006 COMMON EPS,EPS2,EPS3,MAX,ITIME
0007 IF(ITIME.NE.0) GO TO 5
0008 ITIME=1
0009 WRITE(IO21,1000)
0010 10 N=DSQR (X2R*X2R+X2*X2I)
0011 BETA=DATAN2(X2IX2RI)
0012 WRITE(IO21,1020)
0013 20 NALTER=NALTER+1
0014 IF(NALTER.GT.5) RETURN
0015 GO TO (30,40,30,40,30,40,30)
0016 30 X2R=-X2R
0017 X2I=-X2I
0018 GO TO 50
0019 40 BETA=BETA+1.0471976
0020 X2R=DCOS(BETA)
0021 X2I=DSIN(BETA)
0022 50 X1R=0.9*X2R
0023 X1I=0.9*X2I
0024 X3R=1.1*X2R
0025 X3I=1.1*X2I
0026 RETURN
0027 1000 FORMAT(1X,SHX1=,D23.16,3H +,D23.16,2H 1,10X,2HALTERED APPROXIM
0028 1ATIONS/1X,SHX2 =,D23.16,3H +,D23.16,2H 1/1X,SHX3 =,D23.16,3H +
0029 2D23.16,2H 1/1)
0030 1020 FORMAT(1X,SHX1=,D23.16,3H +,D23.16,2H 1/1X,SHX2 =,D23.16,3H +
0031 2D23.16,2H 1/1)
0032 1040 FORMAT(1X,SHX1=,D23.16,3H +,D23.16,2H 1/1X,SHX3 =,D23.16,3H +
0033 2D23.16,2H 1/1)
0034 1060 FORMAT(1X,SHX1=,D23.16,3H +,D23.16,2H 1/1X,SHX3 =,D23.16,3H +
0035 2D23.16,2H 1/1)
0036 1080 FORMAT(1X,SHX1=,D23.16,3H +,D23.16,2H 1/1X,SHX3 =,D23.16,3H +
0037 2D23.16,2H 1/1)
0038 END
```
SUBROUTINE GENAPP(APPR, APP1, NAPP, XSTART)

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

DOUBLE PRECISION APPR, APP1, XSTART, EPS1, EPS2, EPS3, BETA
DIMENSION APPR(25,3), APP1(25,3)
COMMON EPS1, EPS2, EPS3, IO02, MAX
IF (XSTART.EQ.0.0) XSTART=0.5
BETA=0.2817994
DO 10 I = 1, NAPP
APPR(1,2)=XSTART*DOSIN(BETA)
APPR(1,3)=XSTART*DOSIN(BETA)
10 XSTART=XSTART+0.5
DO 20 I = 1, NAPP
APPR(I,1)=0.9*APPR(I,2)
APPR(I,2)=0.9*APPR(I,2)
20 APPR(I,3)=1.1*APPR(I,2)
RETURN
END
TABLE C.V (Continued)

**SUBROUTINE BETTER**

*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**

*DIMENSION UROOT(25),VR0OT(25),U(25),V2(25),UAPP(25,3),VAPP(25,3)**

**COMMON EPSRT,EPSONEPSRT(2),NMAX**

*IF(NROOT.LE.1) RETURN*

*L=0*

**DO I=1,NROOT**

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

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

*UBAPP(I,2)=UROOT(I)*EPSRT*

*VBAPP(I,2)=VR0OT(I)*EPSRT*

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

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

**J=1,NROOT**

*UX1=UBAPP(J,1)*

*VX1=VBAPP(J,1)*

*UX2=UBAPP(J,2)*

*VX2=VBAPP(J,2)*

*UX3=UBAPP(J,3)*

*VX3=VBAPP(J,3)*

**ITER=I**

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

**CALL HORNER(INP,VA,UX2,VX2,US,VB,UPX2,VPX2)**

**CALL HORNER(INP,VA,UX3,VX3,US,VB,UPX2,VPX3)**

**CALL CALC(UX1,VX1,UX2,VX2,UX3,VX3,UPX1,VPX1,UPX2,VPX2,UPX3,VPX3,UX4,VX4,UX5,VX5)**

**IF(CONV) GO TO 50**

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

**WRITE(10,1000) J,UROOT(J),VROOT(J),MAX**

**IF(J.LT.IROOT) GO TO 33**

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

**GO TO 100**

**K=IROOT-1**

**DO K=J,K**

*URAPP(K,1)=URAPP(K+1,1)*

*VRAPP(K,1)=VRAPP(K+1,1)*

*URAPP(K,2)=URAPP(K+1,2)*

*VRAPP(K,2)=VRAPP(K+1,2)*

*URAPP(K,3)=URAPP(K+1,3)*

*VRAPP(K,3)=VRAPP(K+1,3)*

**IROOT=IROOT-1**

**GO TO 100**

**UX1=UX2**
TABLE C.V (Continued)

0046    VX1=VX2
0047    UX2=UX3
0048    VX2=VX3
0049    UX3=UX4
0050    VX3=UX4
0051    UPX1=UPX2
0052    VPX1=VPX2
0053    UPX2=VPX3
0054    VPX2=VPX3
0055    ITER=ITER+1
0056    GO TO 20
0057    50 L=L+1
0058    UROOTS(L)=UX4
0059    VROOTS(L)=VX4
0060    MULT(L)=MULT(I)
0061    100 CONTINUE
0062    IF(L.LE.0) GO TO 120
0063    DO 110 I=1,L
0064    UROOT(I)=UROOTS(I)
0065    110 VROOT(I)=VROOTS(I)
0066    NROOT=L
0067    RETURN
0068    120 NROOT=0
0069    RETURN
0070    1000 FORMAT(///42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(I,12,4H) = ,
0071    1023.16,3H + .D23.16,2H I/24H DID NOT CONVERGE AFTER 13.11H ITERAT
0072    2IONS)
0073    1010 FORMAT(3OH THE PRESENT APPROXIMATION IS .D23.16,3H + .D23.16,2H I/
0074    17)
0075    END
TABLE C.V (Continued)

0001 SUBROUTINE CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VUPX1, UPX2, VUPX2, UPX3, VUPX3)

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

0002 DOUBLE PRECISION ARA1, ARA2
0003 DOUBLE PRECISION UPX3, UPX5, UPX2, UPX1, UX1, UX2, UX3, VX1, VX2, VX3, UPX1, VUPX1, UPX2, VUPX2, UPX3, VUPX3, UX3, VX3, UX2, VX2, UX1, VX1, UH3, VH3, UH2, VH2, UG3, VG3, UG2, VG2, VC, UD, UBD, UBO, UB, UC, UDSC, UDSD, UCC, UC
0004 COMMON EPSRT, EPSO, EPS, E0, MAX
0005 UH3=UX3-UX2
0006 VH3=VX3-VX2
0007 UH2=UX2-UX1
0008 VH2=VX2-VX1
0009 UG3=(UH3=UH2+VH3+VH2)/8
0010 VG3=(VH3=UH2-UH3+VH2)/8
0011 UDDD=1.0+UG3
0012 VDDD=VG3
0013 UD=(UPX3-(UDDD*UPX2+VDDD*VUPX2)+(UG3*UPX1-VG3*VUPX1))
0014 UD=(VPX3-(VDDD*UPX2+VDDD*VUPX2)+(UG3*UPX1-VG3*VUPX1))
0015 VAAA=2.0*VDDD
0016 VAAA=UAAA+1.0
0017 UBBD=UDDD*UDDD*VDDD
0018 VBBB=VDDD*VDDD*VDDD
0019 UCCC=UQ3*UQ3*UG3
0020 VCCC=VQ3*VQ3*VG3
0021 US=(UAAA*UPX3-UAAA*VUPX3)-(UBBB*UPX2-VBBB*VUPX2)+(UCCC*UPX1-VCCC*V)
0022 VPX1)
0023 VB=(VAAA*UPX3+VAAA*VUPX3)-(VBBB*UPX2+VBBB*VUPX2)+(VCCC*UPX1-VCCC*V)
0024 VPX1)
0025 UC=UDDD*UPX3-VDDD*VUPX3
0026 VC=VDDD*UPX3-VDDD*VUPX3
0027 UDSC=(UBBD*UBBD*UBBD)-(4.0*(UBBD*UC-UD*VC))
0028 UDSD=(UBBD*UBBD*UBBD)-(4.0*(UBBD*UC+UD*VC))
0029 AAA+UDSR*UDSC*UDSD*UDSC*UDSD
0030 IF(AAA+EQ.0.0) GO TO 5
0031 GO TO 7
0032 GO TO 9
0035 IF(THA+EQ.0.0) THA+DATA2(YDISC, UDSC, UDSD)
0036 9 RAD+DSRT(AAA)
0037 ANGLE+THA/2.0
0038 UTEST=RAD+DCOS(ANGLE)
0039 VTEST=RAD+DSIN(ANGLE)
0040 UDEN1=UB+UTEST
0041 VDEN1=VB+VTEST
0042 UDEN2=UB-UTE
0043 VDEN2=VB-VTEST
0044 ARGI=UDEN1*YDEN1+YDEN1*YDEN1
TABLE C.V (Continued)

0045 - \[ \text{ARG} = \text{UOEN2} \times \text{UDEN2} + \text{VDEN2} \times \text{VDEN2} \]
0046  \[ \text{AAA} = \text{DSQRT} \times \text{ARG1} \]
0047  \[ \text{BBB} = \text{DSQRT} \times \text{ARG2} \]
0048  IF (\text{AAA} < \text{BBB}) GO TO 10
0049  IF (\text{AAA} = 0.0) GO TO 60
0050  \text{UAAA} = -2.0 \times \text{UC}
0051  \text{VAAA} = -2.0 \times \text{VC}
0052  \text{UQ4} = (\text{UAAA} \times \text{UDEN1} + \text{VAAA} \times \text{VDEN1}) / \text{ARG1}
0053  \text{V04} = (\text{VAAA} \times \text{UDEN1} - \text{UAAA} \times \text{VDEN1}) / \text{ARG1}
0054  GO TO 50
0055  10 IF (\text{BBB} = 0.01) GO TO 60
0056  \text{UAAA} = -2.0 \times \text{UC}
0057  \text{VAAA} = -2.0 \times \text{VC}
0058  \text{UQ4} = (\text{UAAA} \times \text{UDEN2} + \text{VAAA} \times \text{VDEN2}) / \text{ARG2}
0059  \text{VQ4} = (\text{VAAA} \times \text{UDEN2} - \text{UAAA} \times \text{VDEN2}) / \text{ARG2}
0060  GO TO 50
0061  50 \text{UX4} = \text{UX3} + (\text{UH3} \times \text{UQ4} - \text{VH3} \times \text{VQ4})
0062  \text{VX4} = \text{VX3} + (\text{VH3} \times \text{UQ4} + \text{UH3} \times \text{VQ4})
0063  RETURN
0064  60 \text{UQ4} = 1.0
0065  \text{VQ4} = 0.0
0066  GO TO 50
0067  END
TABLE C. V (Continued)

0001 SUBROUTINE TEST(UX3, VX3, UX4, VX4, CONV)
C SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROX-
C IMATIONS BY TESTING THE EXPRESSION
C ABSOLUTE VALUE OF \|x(n+1) - x(n)|/ABSOLUTE VALUE OF x(n).
C WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.
C
C******************************************************************************
C DOUBLE PRECISION UX3, VX3, UX4, VX4, EPSR1, EPSQ, EPS, AAA, UDUMMY, VDUMMY,
IDENOM
LOGICAL CONV
0003 COMMON EPSR1, EPSQ, EPS, 102, MAX
0005 UDUMMY=UX4-UX3
0006 VDUMMY=VX4-VX3
0007 AAA=DSORT(UDUMMY*UDUMMY*UDUMMY*UDUMMY)
0008 DENOM=DSORT(UX4*UX4*UX4*UX4)
0009 IF(IDENOM.LT.EPSQ) GO TO 20
0010 IF(AAA/GENOM.LT.EPSQ) GO TO 10
0011 5 CONV=.FALSE.
0012 GO TO 100
0013 10 CONV=.TRUE.
0014 GO TO 100
0015 20 IF(AAA.LT.EPSQ) GO TO 10
0016 GO TO 5
0017 100 RETURN
0018 END

0001 SUBROUTINE HORNER(NA, UA, VA, UX, VX, UB, VUB, VPX, VPX)
C HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(x) AT A POINT D.
C SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE
C FACTOR (x-D).
C
C******************************************************************************
C DOUBLE PRECISION UX, VX, UPX, VPX, UB, VUB, UA, VA
0004 UB(1)=UA(1)
0005 VUB(1)=VA(1)
0006 NUM=NA+1
0007 GO TO 10+2+NUM
0008 UB(I+1)=UA(I+1)+UB(I+1)*UX+VUB(I)*VX
0009 10 VUB(I+1)=VUB(I)+VUB(I)*UX+UB(I)*VX
0010 UPX=UB(NUM)
0011 VPX=VUB(NUM)
0012 RETURN
0013 END
SUBROUTINE QUAD(UA, VA, UROOT, VROOT, NROOT, MULTI, EPSST)

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

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

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

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(2) * UA(1) + VA(2) * VA(1)) / BBB
VROOT(NROOT+1) = (-VA(2) * UA(1) + UA(2) * VA(1)) / BBB
MULTI(NROOT+1) = 1
NROOT = NROOT + 1
GO TO 50

7 UDISC = (UA(2) * UA(2) - VA(2) * VA(2)) - (UA(1) * UA(1) - VA(1) * VA(1))
VDISC = (UA(2) * UA(2) - VA(2) * VA(2)) - (UA(1) * UA(1) - VA(1) * VA(1))

BBB = SQRT(UDISC*UDISC + VDISC*VDISC)
IF(BBB.LT.EPSST) GO TO 10

CALL COMSOT(UDISC, VDISC, UDDUMM, VDDUMM)
UBBB = -UA(2) + UDDUMM
VBBB = -VA(2) + VDDUMM
ROUMMY = -UA(2) - UDDUMM
SDUMMY = -VA(2) - VDDUMM

UAAA = 2.0 * UA(1)
VAAA = 2.0 * VA(1)
BBB = UAAA * UAAA + VAAA * VAAA

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

10 UAAA = 2.0 * UA(1)
VAAA = 2.0 * VA(1)
BBB = UAAA * UAAA + VAAA * VAAA

UROOT(NROOT+1) = (-UA(2) * UAAA + VA(2) * VAAA) / BBB
VROOT(NROOT+1) = (-VA(2) * UAAA + UA(2) * VAAA) / BBB
MULTI(NROOT+1) = 2
NROOT = NROOT + 1
GO TO 50

50 RETURN
END
TABLE C.V (Continued)

SUBROUTINE COMSRT(UX, UY, VX, VY)

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

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

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

IF(UX) 10, 20, 30

10 UY = AAA
VY = -I*BBB
GO TO 100

20 IF(UX) 40, 50, 60

30 UY = AAA
VY = BBB
GO TO 100

40 DUMMY = |ABS(UX)|
VY = DSQRT(DUMMY)
GO TO 100

50 UY = 0.0
VY = 0.0
GO TO 100

60 DUMMY = |ABS(UX)|
VY = DSQRT(DUMMY)
GO TO 100

70 UY = 0.0
VY = 0.0
GO TO 100

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 \( \text{XSTART} \). If \( \text{XSTART} \) is 0., \( \text{XSTART} \) is automatically initialized to 0.5 to avoid the approximation \( 0. + 0.i \). The approximations are generated according to the formula:

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

where

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

To accomplish this, the user defines the number of initial approximations to be read (\( \text{NAPP} \)) 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.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Card Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 4 5 7 8</td>
</tr>
<tr>
<td>N</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>N</td>
</tr>
<tr>
<td>POLY</td>
<td>N A</td>
</tr>
<tr>
<td></td>
<td>XSTART</td>
</tr>
<tr>
<td></td>
<td>L 7</td>
</tr>
</tbody>
</table>

Example D.1
Example D.2

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

2. Altering Approximations

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

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

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

\[
\beta = \tan^{-1} \frac{\text{Im} \ X_0}{\text{Re} \ X_0} + \frac{K \pi}{3}; \ K = 1 \text{ if } j = 1 \]

\[
K = 2 \text{ if } j = 3.
\]

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 \( \text{XEND} \) but not greater than \( \text{XEND} + .5N \).

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

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

4. Improving Zeros Found

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

5. Solving Quadratic Polynomial

After N-2 roots of an N\textsuperscript{th} degree polynomial have been extracted, the remaining quadratic, \(ax^2 + bx + c\), is solved using the quadratic formula

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

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

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

7. Miscellaneous

By using various combinations of values for NAPP, XSTART, and XEND, the user has several options available as illustrated below. Example D.5 shows the control card for a seventh degree polynomial. Three initial approximations are supplied by the user. At most three roots will be found and the coefficients of the remaining deflated polynomial will be printed.

```
<table>
<thead>
<tr>
<th>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>N</td>
<td>A</td>
</tr>
<tr>
<td>N</td>
<td>P</td>
<td>N</td>
<td>A</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>XSTART</td>
<td></td>
<td>XEND</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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

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

\[
\begin{array}{cccccccc}
1 & 2 & 4 & 5 & 7 & 8 & 6 & 7 & 7 & 7 & 8 & \quad 0 \\
N & N & P & P & N & XSTART & XEND & 1.5D+01 \\
\end{array}
\]

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>
<thead>
<tr>
<th>Table E.I</th>
</tr>
</thead>
</table>

**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>EPSl</td>
<td>c.c. 23-28</td>
<td>Test for zero in subroutine GCD. Double precision. Right justify. 1.D-03 is recommended.</td>
</tr>
</tbody>
</table>
TABLE E.II (Continued)

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

Coefficients of the Polynomial

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

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

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

Initial Approximations

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

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

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

**End Card**

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

### TABLE E.V

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

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

The definitions of the major variables used in G.C.D. - Newton's method are given in Table E.VI. The symbols used to indicate type are:

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

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

- \( E \) - entered
- \( R \) - returned
- \( ECR \) - entered, changed, and returned
- \( C \) - variable in common
Figure E.1. Sequence of Input Data for G.C.D.-Newton's Method
### Variable Name

<table>
<thead>
<tr>
<th>Card Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000011111111222222222333333344444445555555556666666667777777778</td>
</tr>
<tr>
<td>00000000011111111222222222333333344444445555555556666666667777777778</td>
</tr>
</tbody>
</table>

### 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>N OFF N P N MAX EPS1 EPS2 EPS3 EPS4 XSTART XEND K CHECK</td>
</tr>
<tr>
<td>1 7 7 2001.0D-03 1.0D-10 1.0D-20 1.0D-02 1.0D+01 5.0D+02</td>
</tr>
</tbody>
</table>

### Example

- **UP**: +0.125768D+01
- **VP**: -0.37225D+02

### 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>Type</th>
<th>Double Precision Variable</th>
<th>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>Number of distinct roots found</td>
<td></td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>I</td>
<td>Program control</td>
<td></td>
</tr>
<tr>
<td>NOPOLY</td>
<td>I</td>
<td>NOPOLY</td>
<td>I</td>
<td>Number of the polynomial</td>
<td></td>
</tr>
<tr>
<td>NF</td>
<td>I</td>
<td>NF</td>
<td>I</td>
<td>Degree of the original polynomial</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>UP,VP</td>
<td>D</td>
<td>Array of coefficients of original polynomial, ( P(X) )</td>
<td></td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>NAPP</td>
<td>I</td>
<td>Number of initial approximation to be read</td>
<td></td>
</tr>
<tr>
<td>EPS1</td>
<td>R</td>
<td>EPS1</td>
<td>D</td>
<td>Tolerance check for zero (0) in Subroutine GCD</td>
<td></td>
</tr>
<tr>
<td>EPS2</td>
<td>R</td>
<td>EPS2</td>
<td>D</td>
<td>Tolerance check for convergence</td>
<td></td>
</tr>
<tr>
<td>EPS3</td>
<td>R</td>
<td>EPS3</td>
<td>D</td>
<td>Tolerance check for zero (0) in Subroutine QUAD</td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>I</td>
<td>Maximum number of iterations permitted</td>
<td></td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>I01</td>
<td>I</td>
<td>Unit number of input device</td>
<td></td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>I</td>
<td>Unit number of output device</td>
<td></td>
</tr>
<tr>
<td>KCHECK</td>
<td>I</td>
<td>KCHECK</td>
<td>I</td>
<td>Program control, ( \text{KCHECK} = 1 ) implies stop execution</td>
<td></td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>UAPP,VAPP</td>
<td>D</td>
<td>Array of initial approximations</td>
<td></td>
</tr>
<tr>
<td>XSTART</td>
<td>R</td>
<td>XSTART</td>
<td>D</td>
<td>Magnitude at which to start search for roots</td>
<td></td>
</tr>
<tr>
<td>XEND</td>
<td>R</td>
<td>XEND</td>
<td>D</td>
<td>Magnitude at which to end search for roots</td>
<td></td>
</tr>
<tr>
<td>ANAME</td>
<td>A</td>
<td>ANAME</td>
<td>A</td>
<td>Contains name of method used &quot;NEWTONS&quot;</td>
<td></td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT,VROOT</td>
<td>D</td>
<td>Array of roots found</td>
<td></td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td>Array of multiplicities</td>
<td></td>
</tr>
<tr>
<td>DP</td>
<td>C</td>
<td>UDP,VDP</td>
<td>D</td>
<td>Array containing coefficients of the derivative, ( P'(X) ), of ( P(X) )</td>
<td></td>
</tr>
<tr>
<td>NDP</td>
<td>I</td>
<td>NDP</td>
<td>I</td>
<td>Degree of the derivative of original polynomial</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>UD,VD</td>
<td>D</td>
<td>Array of coefficients of the greatest common divisor of ( P(X) ) and ( P'(X) )</td>
<td></td>
</tr>
<tr>
<td>ND</td>
<td>I</td>
<td>ND</td>
<td>I</td>
<td>Degree of g.c.d. of ( P(X) ) and ( P'(X) )</td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>C</td>
<td>UQ,VQ</td>
<td>D</td>
<td>Array of coefficients of quotient polynomial ( P(X)/\text{g.c.d.} )</td>
<td></td>
</tr>
<tr>
<td>Single 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>
</tr>
<tr>
<td>NQ</td>
<td>I</td>
<td></td>
<td>Degree of quotient polynomial Q(X)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZRO</td>
<td>C</td>
<td></td>
<td>Value at which to evaluate or deflate polynomial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUMMY</td>
<td>C</td>
<td></td>
<td>Dummy variable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QQ</td>
<td>C</td>
<td>Working array of coefficients of current polynomial</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NQQ</td>
<td>I</td>
<td>Number of alterations of an initial approximation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IALTER</td>
<td>I</td>
<td>Number of alterations of an initial approximation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td></td>
<td>Tolerance for checking multiplicities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPS4</td>
<td>R</td>
<td></td>
<td>Array of approximations (initial or altered) producing convergence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td>C</td>
<td></td>
<td>Array of coefficients of newly deflated polynomial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QD</td>
<td>C</td>
<td></td>
<td>Number of distinct roots found by iterative process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JAP</td>
<td>I</td>
<td></td>
<td>Number of distinct roots found in the attempt to improve roots</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jl</td>
<td>I</td>
<td></td>
<td>Array of improved roots</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROOTS</td>
<td>C</td>
<td></td>
<td>Program control. NEWT = TRUE implies that Newton's method was used instead of Subroutine QUAD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEWT</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subroutine NEWTON

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Double Precision Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>UX, VX</td>
<td>Starting approximation (initial or altered)</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Degree of current polynomial</td>
</tr>
<tr>
<td>P</td>
<td>UP, VP</td>
<td>Array of coefficients of current polynomial</td>
</tr>
<tr>
<td>MAX</td>
<td>MAX</td>
<td>Maximum number of iterations</td>
</tr>
<tr>
<td>EPSILON</td>
<td>EPSILON</td>
<td>Tolerance for checking convergence</td>
</tr>
<tr>
<td>XO</td>
<td>UXO, VXO</td>
<td>Current approximation to root</td>
</tr>
<tr>
<td>B</td>
<td>UB, VB</td>
<td>Array of coefficients of newly deflated polynomial</td>
</tr>
<tr>
<td>DPXO</td>
<td>UDFXO, VDFXO</td>
<td>Derivative of the polynomial at XO</td>
</tr>
</tbody>
</table>
### TABLE E.VI (Continued)

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

Subroutine HORNER

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>C</td>
<td>UX, VX</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
</tr>
<tr>
<td>P</td>
<td>C</td>
<td>UP, VP</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>UC, VC</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>UB, VB</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

Array of coefficients of polynomial

Array of coefficients of newly deflated polynomial

Subroutine QUAD

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>UA, VA</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>J</td>
<td>I</td>
<td>J</td>
<td>I</td>
<td>ECR</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>UROOT, VROOT</td>
<td>D</td>
<td>ECR</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>MULT</td>
<td>I</td>
<td>ECR</td>
</tr>
<tr>
<td>DISC</td>
<td>C</td>
<td>UDISC, VDISC</td>
<td>D</td>
<td>ECR</td>
</tr>
<tr>
<td>TEMP</td>
<td>C</td>
<td>UTEMP, VTEMP</td>
<td>D</td>
<td>ECR</td>
</tr>
<tr>
<td>EPSLON</td>
<td>R</td>
<td>EPSLON</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>UD, VD</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

Array of roots found

Array of multiplicities

Discriminate of quadratic

D of DISC

Tolerance for zero (0)

Twice leading coefficient of quadratic
TABLE E.VI (Continued)

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

Subroutine GCD

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

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

**Subroutine DERIV**

| 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 DIVIDE**

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

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Single Precision Variable Type</th>
<th>Disposition of Argument</th>
<th>Double Precision Variable Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOLD</td>
<td>C</td>
<td>XOLDR,XOLDI</td>
<td>ECR</td>
<td>D</td>
<td>Old approximation to be altered to new approximation</td>
</tr>
<tr>
<td>NALTER</td>
<td>I</td>
<td>NALTER</td>
<td>ECR</td>
<td>I</td>
<td>Number of alterations performed on an initial approximation</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
<td>E</td>
<td>I</td>
<td>Program control</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
<td>C</td>
<td>I</td>
<td>Maximum number of iterations permitted</td>
</tr>
<tr>
<td>Y</td>
<td>R</td>
<td>XOLDI</td>
<td></td>
<td>D</td>
<td>Imaginary part of original initial approximation (unaltered)</td>
</tr>
<tr>
<td>X</td>
<td>R</td>
<td>XOLDR</td>
<td></td>
<td>D</td>
<td>Real part of original, unaltered initial approximation</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>ABXOLD</td>
<td></td>
<td>D</td>
<td>Magnitude of original unaltered initial approximation</td>
</tr>
<tr>
<td>BETA</td>
<td>R</td>
<td>BETA</td>
<td></td>
<td>D</td>
<td>Argument of new approximation</td>
</tr>
<tr>
<td>XOLDR</td>
<td>R</td>
<td>XOLDR</td>
<td></td>
<td>D</td>
<td>Real part of new approximation</td>
</tr>
<tr>
<td>XOLDI</td>
<td>R</td>
<td>XOLDI</td>
<td></td>
<td>D</td>
<td>Imaginary part of new approximation</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>I02</td>
<td>C</td>
<td>I</td>
<td>Unit number of output device</td>
</tr>
</tbody>
</table>

**Subroutine ALTER**

- UX,VX  D  E  Complex number for which the square root is desired
- UY,VY  D  R  Square root of the complex number
4. Description of Program Output

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

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

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

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

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

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

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

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

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

5. Informative Messages and Error Messages

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

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

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

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

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

C ****************************************************************************** C
C * DOUBLE PRECISION PROGRAM FOR G.C.D.-NEWTON'S METHOD                     C
C * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPlicITIES OF A      C
C * POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY     C
C * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL C
C * AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED C
C * AND THEIR MULTIPlicITIES DETERMINED.                                   C
C ****************************************************************************** C
0001 DOUBLE PRECISION UP,VP,UAPP,VAPP,UROOT,UROOT1,UP,D,U,D,U,UVO,UV0,UVO,UV0, UVO,UV0,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,UVO,
TABLE E.VII (Continued)

0040  JJJ=NNN-1
0041  25 WRITE(IO2,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)
0042  IF(NP.GE.3) GO TO 30
0043  J=1
0044  CALL QUADINP,UP,VP,J,UROOT,VROOT,MULT
0045  WRITE(IO2,1070)
0046  WRITE(IO2,1165) I,UROOT(I),VROOT(I),MULT(I),J=1,J.
0047  GO TO 10
0048  30 CALL DERIV(NP,UP,VP,NP,UD,VD)
0049  CALL GCD(NP,UP,VP,NP,UD,VD)
0050  IF(ND.EQ.0) GO TO 65
0051  UDDUMMY=UD(2)*(UD(2)+VD(2))
0052  UZRO=-UD(1)*(UD(1)+VD(1))/UDDUMMY
0053  VZRO=-(UD(2)*VD(2))/UDDUMMY
0054  CALL HORNER(UZRO,VZRO,NP,UP,VP,UD,VD,UDDUMMY,VDDUMMY)
0055  NQ=NP-1
0056  DO 60 I=1,NP
0057  UQ(I)=UQ(I+1)
0058  DO 66 I=1,ND
0059  VQ(I)=VQ(I+1)
0060  GO TO 80
0061  NQ=NP
0062  GO TO 66
0063  DO 90 I=1,ND
0064  U0Q(I)=UQ(I)
0065  V0Q(I)=VQ(I)
0066  NQ=NP
0067  GO TO 90
0068  DO 80 WRITE(IO2,1120) QNAME,ENTRY(JJJ),UQ(JJJ),VQ(JJJ)
0069  NQ=NP
0070  GO TO 85
0071  NNN=KKK+1
0072  80 DO 120 I=1,ND
0073  WRITE(IO2,1120) QNAME,ENTRY(JJJ),UQ(JJJ),VQ(JJJ)
0074  IF(NQ.GE.3) GO TO 85
0075  GO TO 110
0076  85 NQ=NQ+1
0077  DO 90 I=1,ND
0078  UQQ(I)=UQ(I)
0079  VQQ(I)=VQ(I)
0080  NQ=NQ
0081  GO TO 120
0082  110 CALL QUADINC(UQ,VQ,J,UROOT,VROOT,MULT)
0083  NEWT=.FALSE.
0084  GO TO 310
0085  120 DO 200 I=1,NAPP
0086  130 CALL NEWTON(UAPP(I),VAPP(I),NQ,UQ,VQ,UZRO,VZRO,CONV)
0087  IF(CONV) GO TO 160
0088  140 CALL ALTER(UAPP(I),VAPP(I),IALTER,ITIME)
0089  IF(IALTER.GT.5) GO TO 200
0090  GO TO 130
0091  160 J=J+1
0092  UROOT(J)=UZRO
0093  VROOT(J)=VZRO
0094  UAPP(J)=UAPP(J)
0095  VAPP(J)=VAPP(J)
0096  CALL HORNER(UZRO,VZRO,NQ,UQ,VQ,UQD,UDDUMMY,VDDUMMY)
TABLE E.VII (Continued)

202
TABLE E.VII (Continued)

```
JJJ=NNN-L
GO TO 10
NEWT=.TRUE.
CALL MULT(NPUPeVPeJUROOTVROOTMULTI
IFNEWT) GO TO 330
WRITE(IO2,10703
WRITE(IO2,10265) (LeUROOT(LIVRDOTIL),NULTIL)tL=1IJI
GO TO 10
WRITE(102,11801
WRITE(IO2e1t901
(LuROOTILbtVROOTILhiMULTILILVAPILIL=IJAP
KKK=JAP+1
IFIJAP.LT.J)
WRITE(102,165)
(LuROOT(LItVROOT(ILIMULTIL)IL-KKKJI
GO TO 10
1000 FORMAT(3I2,1X),9XI3,
1010 FORMAT(2D30.0)
1015 FORMAT(42030.0)
1020 FORMAT(IH1 1OX,41HGREATEST COMMON DIVISOR METHOD
1030 FORMAT(I2X,5HTHE DEGREE OF
1040 FORMAT(2X,A2,4H)=,023.16,3H
1070 FORMAT(//IXl13HROOTS OF P(XI,52X,14HMULTIPLICITIES//)
1080 FORMAT(2X,5HROOT(912,4H)
1090 FORMAT(///IXI3HROOTS OF QIX)
1132 FORMAT///IX,7OHCOEFFICIENTS
1150 FORMAT///IX,52X,14HMULTIPLICITIES,
1160 FORMAT///IX,7OHROOT(12,94HI)
1200 FORMAT///,1X.40HNO
1210 FORMAT///,1X.23HRADIUS TO START SEARCH.
1220 FORMAT///,1X.23HRADIUS TO END SEARCH.
1230 FORMAT///,1X.23HTEST FOR ZERO IN SUBROUTINE GCD.
1240 FORMAT///,1X.23HTEST FOR ZERO IN SUBROUTINE QUAD.
END
```
**TABLE E.VII (Continued)**

```plaintext
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,BETA, EPS1,EPS2,EPS3,EPS4
DIMENSION APPR(25),APP1(25)
COMMON EPS1,EPS2,EPS3,EPS4,EPS29EPS39EPS4

IFIXSTART.EQ.0.0 XSTART=0.5
BETA=0.2617994

I=1,NAPP
APPRI=XSTART*DCOS(BETA)
APP1(I)=XSTART*DSIN(BETA)
BETA=BETA+0.5235988

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

0001 SUBROUTINE ALTER(KOLDK, XOLDL, NALTER, ITIME)
  COMMON EPS1, EPS2, EPS3, EPS4, IO2, MAX
  IF (ITIME .NE. 0) GO TO 5
  NALTER = NALTER + 1
  GO TO 20
0005 ITIME = ITIME + 1
0006 WRITE(IO2, 1010) XOLDK, XOLDL
0007 5 IF (NALTER.GT.5) RETURN
0008 WRITE(IO2, 1000) XOLDK, XOLDL
0010 10 ABXLO = DSQRT(XOLDK*KOLDK) + (XOLDL*KOLDL)
0011 BETA = DATA2(KOLDL, XOLDK)
0012 WRITE(IO2, 1020) XOLDK, XOLDL
0013 20 NALTER = NALTER + 1
0014 IF (NALTER.GT.5) RETURN
0015 GO TO (30, 40, 30, 40, 30, 301, NALTER)
0016 30 XOLDK = XOLDL
0017 XOLDL = XOLDK
0018 GO TO 50
0019 40 BETA = BETA + 2.047916
0020 XOLDK = ABXLO = COSIBETA
0021 XOLDL = ABXLO = SINIBETA
0022 50 RETURN
0023 1000 FORMAT(1X, D23.16, 5H + .D23.16, 2H 1, D23.21HALTED APPROXIMATION)
0024 1010 FORMAT(1X, 5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
  ITER, 13, 124 ITERATIONS.77)
0026 1020 FORMAT(1X, D23.16, 5H + .D23.16, 2H 1, D23.21INITIAL APPROXIMATION)
0026 END
TABLE E.VII (Continued)

SUBROUTINE GCDIN(UR, VR, US, VS, M, USS, VSS)

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

DOUBLE PRECISION USSSSSSS
DOUBLE PRECISION UR, VR, US, VS, USS, VSS, URR, VRD, UTD, VTD, EPSLON, EPS2, EPS3, EPS4, EPS8, EPS102, MAX
DIMENSION UR(26), VR(26), US(26), VS(26), USS(26), VSS(26), URR(26), VRR(26), UT(26), VT(26)
COMMON EPSLON, EPS2, EPS3, EPS4, EPS8, EPS102, MAX

NI = N
MI = M
KKK = N + 1

DO 20 I = 1, KKK
URR(I) = UR(I)
20 VRR(I) = VR(I)

KKK = M + 1

DO 25 I = 1, KKK
US(I) = US(I)
25 VS(I) = VS(I)

BBB = US(MI + 1)*US(MI + 1) + VS(MI + 1)*VS(MI + 1)

UD = URR(NI - 1)*US(MI + 1) + VRR(NI - 1)*VS(MI + 1)/BBB
VD = URR(NI - 1)*VRR(NI - 1) - URR(MI + 1)*VRR(MI + 1)/BBB

KKK = NI - M

DO 40 J = KKK, NI
UT(J) = URR(J)
40 VT(J) = VRR(J)

IF (M.EQ.N) GO TO 70

KKK = K + 1

DO 120 J = 1, KKK
UT(J) = URR(J)
120 VT(J) = VRR(J)

N1 = K
GO TO 30

100 K = NI - I
IF (K.EQ.0) GO TO 170
IF (K.LT.1) GO TO 140
K = K + 1
DO 130 J = 1, KKK
URR(J) = UT(J)
130 VR(J) = VT(J)

GO TO 30
TABLE E.VII (Continued)

0050 140  KKK+K+1
0051  DO 150  J=M*KK
0052     URRIJI=USSIJI
0053     VRR(J)=VSSI(J)
0054     USS(IJ)=UT(J)
0055     VSSI(J)=VT(J)
0056     KKK+K+2
0057     NNN=M+1
0058     DO 160  J=KKK+NNN
0059     URRIJ=USSI(J)
0060     160 VRR(J)=VSSI(J)
0061     MI=M1
0062     MI=K
0063     GO TO 30
0064 170  USS(IJ)=1.0
0065     VSSI(I)=0.0
0066     MI=0
0067 200 RETURN
0068     END
SUBROUTINE QUAD(UA, VA, J, UROOT, VROOT, MULT)

DOUBLE PRECISION UA, VA, UROOT, VROOT, UDISC, VDISC, UTMP, VDISC, UD, VD, E
COMMON EPS1, EPS2, EPSLON, EPSX, MAX

IF(N.EQ.1) GO TO 60

J=J+1
GO TO 50

MULT(I)=I
GO TO 50

BBB=UA(2)*UA(2)+VA(2)*VA(2)
UROOT(I)=(UA(1)-UA(2)*VA(2))/BBB
VROOT(I)=(VA(1)-UA(2)*UA(1))/BBB
GO TO 200

VDISC=(UA(2)*UA(2)-2.0*UA(1)*VA(2))
BBB=DSQRT(UA(2)*UA(2)-4.0*UA(1)*UA(1))

IF(BBB.LE.EPSLON) GO TO 100

IF(J.GE.0) GO TO 80
MULT(I)=2,
J=1
GO TO 130

100 IF(J.LT.0) GO TO 110
J=J+1
GO TO 130

110 MULT(I)=2
J=1
GO TO 130

130 UD=2.0*UA(3)
VD=2.0*VA(3)
BBB=UD*UD+VD*VD
UROOT(I)=(UA(1)-UA(2)*UA(1)-VA(2)*VDISC)*UD/BBB
VROOT(I)=(UA(2)-UA(2)*VA(2)*VDISC)*VD/BBB
GO TO 200

200 RETURN
END
### TABLE E.VII (Continued)

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

* THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROX-
* INATION BY USING THE ITERATION FORMULA
* \[ x_{n+1} = x_n - \frac{x_n - F(x_n)}{F'(x_n)} \]
  
  0002 DOUBLE PRECISION UX, VX, UP, VP, UXO, VXO, U, V, UDIFF, VDIFF, EPS1, EPSLON, EPS3, EPS4, AAA, BBB
  0003 DOUBLE PRECISION DDD
  0004 COMMON EPS1, EPSLON, EPS3, EPS4, IO2, MAX
  0005 LOGICAL CONV
  0006 DO 10 I = 1, MAX
  0007 UX = UXO
  0008 VX = VXO
  0009 UDIFF = 0.0
  0010 VDIFF = 0.0
  0011 UDIFF = UDIFF * (UXO - UX)
  0012 VDIFF = VDIFF * (VXO - VX)
  0013 IF (ABS(UDIFF).EQ.0.0) GO TO 20
  0014 BBB = ABS(UDIFF) * ABS(VDIFF)
  0015 CONV = .FALSE.
  0016 RETURN
  0017 CONV = .TRUE.
  0018 RETURN
  0019 END
```

---

<table>
<thead>
<tr>
<th>0001</th>
<th>SUBROUTINE NEWTON(UX, VX, N, UP, VP, UXO, VXO, CONV)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>* THIS SUBROUTINE CALCULATES A NEW APPROXIMATION FROM THE OLD APPROXIMATION BY USING THE ITERATION FORMULA [ x_{n+1} = x_n - \frac{x_n - F(x_n)}{F'(x_n)} ]</td>
</tr>
<tr>
<td></td>
<td>DOUBLE PRECISION UX, VX, UP, VP, UXO, VXO, U, V, UDIFF, VDIFF, EPS1, EPSLON, EPS3, EPS4, AAA, BBB</td>
</tr>
<tr>
<td></td>
<td>DOUBLE PRECISION DDD</td>
</tr>
<tr>
<td></td>
<td>COMMON EPS1, EPSLON, EPS3, EPS4, IO2, MAX</td>
</tr>
<tr>
<td></td>
<td>LOGICAL CONV</td>
</tr>
<tr>
<td></td>
<td>DO 10 I = 1, MAX</td>
</tr>
<tr>
<td></td>
<td>UX = UXO</td>
</tr>
<tr>
<td></td>
<td>VX = VXO</td>
</tr>
<tr>
<td></td>
<td>UDIFF = 0.0</td>
</tr>
<tr>
<td></td>
<td>VDIFF = 0.0</td>
</tr>
<tr>
<td></td>
<td>UDIFF = UDIFF * (UXO - UX)</td>
</tr>
<tr>
<td></td>
<td>VDIFF = VDIFF * (VXO - VX)</td>
</tr>
<tr>
<td></td>
<td>IF (ABS(UDIFF).EQ.0.0) GO TO 20</td>
</tr>
<tr>
<td></td>
<td>BBB = ABS(UDIFF) * ABS(VDIFF)</td>
</tr>
<tr>
<td></td>
<td>CONV = .FALSE.</td>
</tr>
<tr>
<td></td>
<td>RETURN</td>
</tr>
<tr>
<td></td>
<td>CONV = .TRUE.</td>
</tr>
<tr>
<td></td>
<td>RETURN</td>
</tr>
<tr>
<td></td>
<td>END</td>
</tr>
</tbody>
</table>

---
TABLE E VII (Continued)

SUBROUTINE DIVIDE(N,UP,VP,UD,M,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-H

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

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

VQ(K+1)=VP(M+1)+UQ(M+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

NN=N-J

DO 40 M=NN,K

IF(M.LT.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)+VQ(KK)*VD(M))

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

40 KK=KK-1

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

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

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

50 GO TO 40

100 RETURN

END
Table E.VII (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td></td>
<td><strong>SUBROUTINE HORNER(UX, VX, N, UP, VP, UB, VB, UC, VC)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT D AND ITS DERIVATIVE AT D. SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X - D).</strong></td>
</tr>
<tr>
<td>0002</td>
<td></td>
<td><strong>DOUBLE PRECISION UX, VX, UP, VP, UB, VB, UC, VC</strong></td>
</tr>
<tr>
<td>0003</td>
<td></td>
<td><strong>DOUBLE PRECISION UDIMMY, VDIMMY</strong></td>
</tr>
<tr>
<td>0004</td>
<td></td>
<td><strong>DIMENSION UP(26), VP(26), UB(26), VB(26)</strong></td>
</tr>
<tr>
<td>0005</td>
<td></td>
<td><strong>UB(N+1) = UP(N+1)</strong></td>
</tr>
<tr>
<td>0006</td>
<td></td>
<td><strong>VB(N+1) = VP(N+1)</strong></td>
</tr>
<tr>
<td>0007</td>
<td></td>
<td><strong>UB(N+1) = UX<em>UB(N+1) - VX</em>VB(N+1) + UP(N)</strong></td>
</tr>
<tr>
<td>0008</td>
<td></td>
<td><strong>VB(N+1) = VX<em>UB(N+1) - UX</em>VB(N+1) + VP(N)</strong></td>
</tr>
<tr>
<td>0009</td>
<td></td>
<td><strong>UC = UB(N+1)</strong></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td><strong>VC = VB(N+1)</strong></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td><strong>KKK = N - 1</strong></td>
</tr>
<tr>
<td>0012</td>
<td></td>
<td><strong>DO 10 = 2, KKK</strong></td>
</tr>
<tr>
<td>0013</td>
<td></td>
<td><strong>AAA = I - 1</strong></td>
</tr>
<tr>
<td>0014</td>
<td></td>
<td><strong>UA(I-1) = AAA*UP(I)</strong></td>
</tr>
<tr>
<td>0015</td>
<td></td>
<td><strong>VA(I-1) = AAA*VP(I)</strong></td>
</tr>
<tr>
<td>0016</td>
<td></td>
<td><strong>M = N - 1</strong></td>
</tr>
<tr>
<td>0017</td>
<td></td>
<td><strong>UC = VDIMMY + UB(KKK+2-1)</strong></td>
</tr>
<tr>
<td>0018</td>
<td></td>
<td><strong>VC = VDIMMY + VB(KKK+2-1)</strong></td>
</tr>
<tr>
<td>0019</td>
<td></td>
<td><strong>RETURN</strong></td>
</tr>
<tr>
<td>0020</td>
<td></td>
<td><strong>END</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td></td>
<td><strong>SUBROUTINE DERIV(N, UP, VP, UA, VA)</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF ITS DERIVATIVE P'(X).</strong></td>
</tr>
<tr>
<td>0002</td>
<td></td>
<td><strong>DOUBLE PRECISION UP, VP, UA, VA, AAA</strong></td>
</tr>
<tr>
<td>0003</td>
<td></td>
<td><strong>DIMENSION UP(26), VP(26), UA(26), VA(26)</strong></td>
</tr>
<tr>
<td>0004</td>
<td></td>
<td><strong>KKK = N</strong></td>
</tr>
<tr>
<td>0005</td>
<td></td>
<td><strong>OD 10 = 2, KKK</strong></td>
</tr>
<tr>
<td>0006</td>
<td></td>
<td><strong>AAA = I - 1</strong></td>
</tr>
<tr>
<td>0007</td>
<td></td>
<td><strong>UA(I-1) = AAA*UP(I)</strong></td>
</tr>
<tr>
<td>0008</td>
<td></td>
<td><strong>VA(I-1) = AAA*VP(I)</strong></td>
</tr>
<tr>
<td>0009</td>
<td></td>
<td><strong>M = N - 1</strong></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td><strong>RETURN</strong></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td><strong>END</strong></td>
</tr>
</tbody>
</table>
TABLE E.VII (Continued)

0001 SUBROUTINE MULT(N,UP,VP,J,URDPT,VRDPT,MULT)
0002 C
0003 C
0004 C
0005 C
0006 C
0007 C
0008 C
0009 C
0010 0 = N
0011 MULT(I) = 0
0012 CALL HORNBER(UORDPT(I),VRDPT(I),M,UP,VP,UB,UB,VC,VC)
0013 BBB = SQRT(UB(1) + UB(1) + VB(1) + VB(1))
0014 IF(BBB .LT. EPSLON) GO TO 50
0015 IF(MULT(I) .EQ. 0) GO TO 40
0016 GO TO 100
0017 40 WRITE(102,1000) EPSLON,I,URDPT(I),VRDPT(I)
0018 GO TO 100
0019 50 MULT(I) = MULT(I) + 1
0020 IF(MI .GT. 1) GO TO 60
0021 GO TO 100
0022 60 GO TO K+1,M
0023 70 UA(K) = UB(K+1)
0024 80 VA(K) = VB(K+1)
0025 M = M-1
0026 GO TO 20
0027 100 CONTINUE
0028 RETURN
0029 1000 FORMAT(/15H THE EPSILON 1.000,3,48H) CHECK IN SUBROUTINE MULT
0030 1INDICATES THAT ROOT(I),42,4H = .023,16,3H + .023,16,3H .1,88H IS NO
0031 2T CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP
0032 3LICITY 0/1
0033 END
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((R+UX/2.0)))
0005 BBB=DSQRT(DABS((R-UX/2.0)))
0006 IF(VX).GT.10,20,30
0007 10 UY=AAA
0008 VY=-1.0*BBB
0009 GO TO 100
0010 20 IF(UX).LT.40,50,60
0011 30 UY=AAA
0012 VY=BBB
0013 GO TO 100
0014 40 DUMMY=DSQRT(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=DSQRT(UX)
0022 UY=DSQRT(DUMMY)
0023 VY=0.0
0024 100 RETURN
0025  END
APPENDIX F

G.C.D. - MULLER'S METHOD

1. Use of the Program

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

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

In this program both the leading coefficient and the constant coefficient are assumed to be non-zero.
TABLE F.1

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

Main Program

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

URAPP(N,3), VRAPP(N,3)
UAPP(N,3), VAPP(N,3)
UP(N+1), VP(N+1)
UROOT(N), VROOT(N)
MULT(N)
UDP(N+1), VDP(N+1)
UD(N+1),VD(N+1)
UQ(N+1), VQ(N+1)
UQQ(N+1), VQQ(N+1)
UB(N+1), VB(N+1)
ENTRY(N+1)

Subroutines MULTI, DIVIDE, DERIV, GCD, and QUAD

See corresponding subroutines in Table E.I.

Subroutine MULLER

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

Subroutine BETTER

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

Subroutine GENAPP

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

Subroutine HORNER

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

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

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

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

4. Description of Program Output

The output from G.C.D. - Muller's method is identical to that for G.C.D. - Newton's method as described in Appendix E, § 4, keeping in mind that Muller's instead of Newton's method is used. The expression "SOLVED BY DIRECT METHOD" is equivalent to "RESULTS OF SUBROUTINE QUAD." Only one initial approximation, \( X_0 \), (not three) is printed. The other two required by Muller's method were \( .9X_0 \) and \( 1.1X_0 \).

5. Informative Messages and Error Messages

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

"THE EPSILON (XXX) CHECK IN SUBROUTINE MULTI INDICATES THAT ROOT \( YY = ZZZ \) IS NOT CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIPlicity 0." This message is described in Appendix E, § 5.

"COEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND." This message is described in Appendix E, § 5.
"NO ZEROS WERE FOUND FOR POLYNOMIAL NUMBER XX." XX represents the number of the polynomial for which no zeros were extracted.

"IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT XX = YYY DID NOT CONVERGE AFTER ZZZ ITERATIONS." This message indicates that a root did not produce convergence during the attempt to improve accuracy. XX represents the number of the root before the attempt to improve accuracy, YYY represents its value, and ZZZ represents the maximum number of iterations. The following message then follows. "THE PRESENT APPROXIMATION IS AAA." AAA represents the present approximation to the root after the maximum number of iterations.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>I</td>
<td>Degree of polynomial P(X)</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>NROOT</td>
<td>I</td>
<td>Number of roots (counting multiplicities)</td>
</tr>
<tr>
<td>NOMULT</td>
<td>I</td>
<td>Array containing the roots</td>
</tr>
<tr>
<td>ROOT</td>
<td>C</td>
<td>Array containing the roots</td>
</tr>
<tr>
<td>NAPP</td>
<td>I</td>
<td>Number of initial approximations to be read in</td>
</tr>
<tr>
<td>APP</td>
<td>C</td>
<td>Array of initial approximations</td>
</tr>
<tr>
<td>WORK</td>
<td>C</td>
<td>Working array containing coefficients of current polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>Array containing coefficients of deflated polynomial</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>Array containing coefficients of original polynomial, P(X)</td>
</tr>
<tr>
<td>RAPP</td>
<td>C</td>
<td>Array of initial or altered approximation for which convergence was obtained</td>
</tr>
<tr>
<td>X1</td>
<td>C</td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X2</td>
<td>C</td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>One of three current approximations to a root</td>
</tr>
<tr>
<td>PX1</td>
<td>C</td>
<td>Value of polynomial P(X) at X1</td>
</tr>
<tr>
<td>PX2</td>
<td>C</td>
<td>Value of polynomial P(X) at X2</td>
</tr>
<tr>
<td>PX3</td>
<td>C</td>
<td>Value of polynomial P(X) at X3</td>
</tr>
<tr>
<td>PX4</td>
<td>C</td>
<td>Newest approximation (X_{n+1}) to root</td>
</tr>
<tr>
<td>MULT</td>
<td>I</td>
<td>Array containing the multiplicities of each root found</td>
</tr>
<tr>
<td>ITER</td>
<td>I</td>
<td>Counter for iterations</td>
</tr>
<tr>
<td>I01</td>
<td>I</td>
<td>Unit number of input device</td>
</tr>
<tr>
<td>I02</td>
<td>I</td>
<td>Unit number of output device</td>
</tr>
<tr>
<td>EPSRT</td>
<td>R</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>Number of the polynomial</td>
</tr>
<tr>
<td>Single Precision Variable</td>
<td>Single Precision Type</td>
<td>Double Precision Variable</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>MAX</td>
<td>I</td>
<td>MAX</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
</tr>
<tr>
<td>EPSM</td>
<td>R</td>
<td>EPSM</td>
</tr>
<tr>
<td>KCHECK</td>
<td>I</td>
<td>KCHECK</td>
</tr>
<tr>
<td>XSTART</td>
<td>R</td>
<td>XSTART</td>
</tr>
<tr>
<td>XEND</td>
<td>R</td>
<td>XEND</td>
</tr>
<tr>
<td>NWORK</td>
<td>I</td>
<td>NWORK</td>
</tr>
<tr>
<td>ITIME</td>
<td>I</td>
<td>ITIME</td>
</tr>
<tr>
<td>NALTER</td>
<td>I</td>
<td>NALTER</td>
</tr>
<tr>
<td>IAPP</td>
<td>I</td>
<td>IAPP</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
</tr>
<tr>
<td>IROOT</td>
<td>I</td>
<td>IROOT</td>
</tr>
</tbody>
</table>

Subroutine HORNER

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>Array of current polynomial coefficients (to be deflated or evaluated)</td>
</tr>
<tr>
<td>NA</td>
<td>I</td>
<td>Degree of polynomial to be deflated or evaluated</td>
</tr>
<tr>
<td>X</td>
<td>C</td>
<td>Approximation at which to evaluate or deflate the polynomial</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>Array containing the coefficients of the deflated polynomial</td>
</tr>
<tr>
<td>PX</td>
<td>C</td>
<td>Value of the polynomial at X</td>
</tr>
<tr>
<td>NUM</td>
<td>I</td>
<td>Number of coefficients of polynomial to be deflated</td>
</tr>
<tr>
<td>Single Precision</td>
<td>Double Precision</td>
<td>Disposition of Argument</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td><strong>Type</strong></td>
<td></td>
</tr>
<tr>
<td>X3</td>
<td>C</td>
<td>UX3, VX3</td>
</tr>
<tr>
<td>X4</td>
<td>C</td>
<td>UX4, VX4</td>
</tr>
<tr>
<td>CONV</td>
<td>L</td>
<td>CONV</td>
</tr>
<tr>
<td>EPS</td>
<td>R</td>
<td>EPS</td>
</tr>
<tr>
<td>EPSO</td>
<td>R</td>
<td>EPSO</td>
</tr>
<tr>
<td>DENOM</td>
<td>R</td>
<td>DENOM</td>
</tr>
</tbody>
</table>

**Subroutine BETTER**

<table>
<thead>
<tr>
<th><strong>Variable</strong></th>
<th><strong>Type</strong></th>
<th><strong>Disposition of Argument</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>
TABLE F.II (Continued)

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

Subroutine ALTER

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

Subroutine CALC

These variables are dummy variables used for temporary storage and thus, are not defined.
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)
TABLE F.III

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

C
C
C * DOUBLE PRECISION PROGRAM FOR G.C.D.-MULLER'S METHOD
C
C * THE G.C.D. METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25. ALL MULTIPLE ROOTS ARE REMOVED BY
C * DIVIDING THE POLYNOMIAL BY THE GREATEST COMMON DIVISOR OF THE POLYNOMIAL
C * AND ITS DERIVATIVE. THE ZEROS OF THE RESULTING POLYNOMIAL ARE EXTRACTED
C * AND THEIR MULTIPLICITIES DETERMINED.
C
C ***********************************************************
DOUBLE PRECISION UAPP, VRAPP
DOUBLE PRECISION UP, VP, UAPP, VAPP, JROOT, YROOT, UDP, VDP, UO, VO, UZRO, VZ
DOUBLE PRECISION UO, VO, UOUMY, YOUMY, UQO, VOQ, U0, VO, EPS1, EPS2, EPS3, EPS4
DIMENSION UAPP(25,11), VRAPP(25,31), UAPPI25,31, VAPP(25,31), VAPP(25,31)
DIMENSION UP(26), VP(26), JROOT(25), YROOT(25), UDP(26), VDP(26), UO(26), VO(26), UZRO(26), VZRO(26), UAPP(26), VAPP(26)
DIMENSION UROOT(25), VROOTI25), MULT(25), UDPI26), VDP(26), UO(26), VO(26), UZRO(26), VZRO(26), UAPP(26), VAPP(26)
DIMENSION UOUMY(26), YOUMY(26), UQO(26), VOQ(26), UQO(26), VOQ(26), UOUMY(26), YOUMY(26)
DIMENSION EPSRT, EPS1, EPS2, EPS3, EPS4
COMMON EPSRTEPS1, EPS2, EPS3, EPS402i2MAX
DATA PNAME, QNAME, QQNAME /2HP(,2HQ(,3HQQI/
DATA ENTRY/IHIH2.H3,1H4,IHSelH6,1H7tIHB,H9,2H10,2HI1,2HI2.2H13
1.2HI4,2H15,2H16,2HI7,2H18,2119,22H02H21,2H22,2H232H24,2H25,2H26/
DATA ANAME(1)IANAME(2)/4HMULL,4HERS
LOGICAL NEWT
101=5
102=6
J=0
READ (101,1000) NOPOLY, NP, NAPP, MAX, EPS1, EPS2, EPS3, EPS4, XSTART, XEND
10 CHECK
IF (KCHECK.EQ.11 STOP
WRITE (IO2,1010) ANAME(Ir,ANAME(I2)NOPOLY
WRITE (102,2010) NAPP
WRITE (102,2020) EPS1
WRITE (102,2030) EPS2
WRITE (102,2040) EPS3
WRITE (102,2050) EPS4
WRITE (102,2060) XSTART
KX=NP+1
NNX=KX+1
DO 20 20=1,KK
CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
WRITE (102,2080) NP
READ (101,1001) UP(IJJJ), VP(IJJJ)
IF (NAPP.NE.0) GO TO 22
NAPP=NP
CALL GENAPP(UAPP, VAPP, NAPP, XSTART)
GO TO 23
22 READ (101,1005) UAPP(I, 21), VAPP(I, 21), I=1, NAPP
23 WRITE (102,2030) NP
KX=NP+1
NNX=KX+1
20
TABLE F. III (Continued)

0042 DO 25 I=1,KKK  
0043 JJJ=NNN-1  
0044 WRITE(02,1040) PNAME,ENTRY(JJJ),UP(JJJ),VP(JJJ)  
0045 IF(NP.GE.3) GO TO 30  
0046 JK=1  
0047 CALL QUAD(NP,UP,VP,J,UROOT,VROOT,MULT)  
0048 WRITE(02,1070)  
0049 WRITE(02,1165) (I,UROOT(I),VROOT(I),MULT(I),I=1,J)  
0050 GO TO 10  
0051 30 CALL DERIV(INP,UP,VP,NDP,UDP,VDP)  
0052 CALL GCD(INP,UP,VP,NDP,UDP,VDP,ND,UD,VD)  
0053 IF(NP.GT.1) GO TO 70  
0054 IF(NP.EQ.1) GO TO 65  
0055 UZROOT=U0(I)+U0(I)+VD(2)*VD(2)  
0056 VZROOT=U0(I)-U0(I)+VD(2)*VD(2)/UDUMMY  
0057 KKK=NP+1  
0058 DO 55 I=1,KKK  
0059 UQQ(I)=UP(KKK+1-I)  
0060 VQQ(I)=VP(KKK+1-I)  
0061 NQQ=NP  
0062 CALL HORNER(NQQ,UQQ,VQQ,UROOT,VROOT,MULT)  
0063 GO TO 80  
0064 M=NP-1  
0065 DO 60 I=1,M  
0066 UQ(I)=UP(I)  
0067 60 VQ(I)=VP(I)  
0068 GO TO 80  
0069 KKK=NP+1  
0070 DO 66 I=1,KKK  
0071 UQ(I)=UP(I)  
0072 66 VQ(I)=VP(I)  
0073 NQ=NP  
0074 GO TO 80  
0075 70 CALL DIVIDE(INP,UP,VP,ND,UD,VD,NQ,UQ,VQ)  
0076 WRITE(02,120) NQ  
0077 KKK=NQ-1  
0078 NNN=KKK+1  
0079 DD 83 I=1,KKK  
0080 JJJ=NNN-1  
0081 WRITE(02,1040) QNAME,ENTRY(JJJ),UQ(JJJ),VQ(JJJ)  
0082 IF(NQ.GE.3) GO TO 85  
0083 GO TO 110  
0084 KKK=NQ+1  
0085 DD 90 I=1,KKK  
0086 UQQ(I)=UQ(KKK+1-I)  
0087 NQQ=NP  
0088 GO TO 70  
0089 110 CALL QUAD(INQ,UQ,VQ,J,UROOT,VROOT,MULT)  
0090 NEW=.FALSE.  
0091 GO TO 310  
0092 120 CALL MULLER(UQ,VQ,NQP,VAPP,NAPP,XSTART,XEND,UROOT,VROOT,J,JUNE,NEW=.FALSE.,1AP,VRAPP,NOPOLY)  
0094 NEW=.TRUE.  
0095 310 CALL MULT(INP,UP,VP,J,UROOT,VROOT,MULT)  
0096 IF(NEW) GO TO 330  
0097 WRITE(02,1070)  
0098 WRITE(02,1165) (I,UROOT(I),VROOT(I),MULT(I),I=1,J)
TABLE F.III (Continued)

0099  GO TO 10
0100  WRITE(102,1180)
0101  DO 350 L=1,JAP
0102  WRITE(102,1190) L,UROOT(L),VRDT(L),MULTIL,VRAPP(L),VRAPP(L)
0103  KK=JAP+1
0104  WRITE(102,1165) L,UROOT(L),VRDT(L),MULTIL,L=KK,J
0105  GO TO 10
0106  1000 FORMAT(1,2(1X),9X,13,1X,41D6.0,IXI,13K,2(DT7.0IXI))
0107  1010 FORMAT(2030.0)
0108  1015 FORMAT(2D30.0)
0109  1020 FORMAT(1H,1OX,41HGREATEST
            COMMON DIVISOR
            METHOD
            USED
            WITH
            POLYNOMIAL
            NUMBER
            1
            22/1
0110  1030 FORMAT(1X,22HTHE DEGREE OF P(X) IS ,12,22H THE COEFFICIENTS ARE//1)
0111  1040 FORMAT(2X,A2,A2,4H) = .023.16,3H + .023.16,2H 11
0112  1050 FORMAT(2X,13HROOTS OF P(X),52X,14HMULTIPLICITIES//1)
0113  1060 FORMAT(2X,5HRROOT(1),12,4H) = .023.16,3H + .023.16,2H 1,10X,121
0114  1100 FORMAT(2X,5HRROOT(1),12,4H) = .023.16,3H + .023.16,2H 11
0115  1120 FORMAT(1X,73H) IS THE POLYNOMIAL WHICH HAS AS ITS ROOTS THE
            10 DISTINCT ROOTS OF P(X).//1X,22HTHE DEGREE OF Q(X) IS ,12,22H THE C
            COEFFICIENTS ARE//1)
0116  1165 FORMAT(2X,5HRROOT(1),12,4H) = .023.16,3H + .023.16,2H 1,7X,12,10X,26H
            RESULTS OF SUBROUTINE QUAD
0117  1180 FORMAT(2X,13HROOTS OF P(X),52X,14HMULTIPLICITIES,17X,21HINITI
            AL 1 APPROXIMATION//1)
0118  1190 FORMAT(2X,5HRROOT(1),12,4H) = .023.16,3H + .023.16,2H 1,7X,12,9X,023.
            116,3H + .023.16,2H 11
0119  2000 FORMTAT(1X,4H)NUMBER OF INITIAL APPROXIMATIONS GIVEN. 121
0120  2010 FORMAT(1X,24HMAXIMUM NUMBER OF ITERTIONS. 11X,13)
0121  2020 FORMAT(1X,24HTEST FOR CONVERGENCE. 11X,13)
0122  2030 FORMAT(1X,24HTEST FOR MULTIPLICITIES. 11X,13)
0123  2040 FORMAT(1X,24HRADIUS TO START SEARCH. 11X,13)
0124  2050 FORMAT(1X,24HRADIUS TO END SEARCH. 11X,13)
0125  2060 FORMAT(1X)
0126  2070 FORMAT(1X,14HTEST FOR ZERO IN SUBROUTINE GCD. 11X,13)
0127  2080 FORMAT(1X,14HTEST FOR ZERO IN SUBROUTINE QUAD. 11X,13)
0128  END
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,VRoot,MULT)</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION UP,VP,URoot,VRoot,UA,VA,UB,VB,VC,EPs1,EPs2,EPs3,EPs</td>
</tr>
<tr>
<td>0003</td>
<td>DIMENSION UP(26),VP(26),URoot(251),VRoot(251),UA(261),VA(261),UB(261),VB(261),VC(261),MULT(25)</td>
</tr>
<tr>
<td>0004</td>
<td>DOUBLE PRECISION EPSRT, EPS1, EPS2, EPS3, EPSLON, IO2, MAX</td>
</tr>
<tr>
<td>0005</td>
<td>COMMON EPSRT, EPS1, EPS2, EPS3, EPSLON, IO2, MAX</td>
</tr>
<tr>
<td>0006</td>
<td>GO TO 100 I=1,J</td>
</tr>
<tr>
<td>0007</td>
<td>KKK=N+I</td>
</tr>
<tr>
<td>0008</td>
<td>GO TO 10 K=1,KKK</td>
</tr>
<tr>
<td>0009</td>
<td>UA(K)=UP(KKK+1-K)</td>
</tr>
<tr>
<td>0010</td>
<td>DO 10 VA(K)=VP(KKK+1-K)</td>
</tr>
<tr>
<td>0011</td>
<td>M=N</td>
</tr>
<tr>
<td>0012</td>
<td>MULT(1)=0</td>
</tr>
<tr>
<td>0013</td>
<td>CALL HORNER(1,UA,VA,URoot(1),VRoot(1),UB,VB,VC)</td>
</tr>
<tr>
<td>0014</td>
<td>B8=DSQRT(UB(UB)+VC(VC))</td>
</tr>
<tr>
<td>0015</td>
<td>IF(MULT(1),EQ,0) GO TO 50</td>
</tr>
<tr>
<td>0016</td>
<td>IF(MULT(1)&lt;EQ,0) GO TO 40</td>
</tr>
<tr>
<td>0017</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0018</td>
<td>WRITE(IO2,1000) EPSLON,1,URoot(1),VRoot(1)</td>
</tr>
<tr>
<td>0019</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0020</td>
<td>MULT(1)=MULT(1)+1</td>
</tr>
<tr>
<td>0021</td>
<td>IF(Mult.GT,01) GO TO 60</td>
</tr>
<tr>
<td>0022</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0023</td>
<td>DO 70 K=1,M</td>
</tr>
<tr>
<td>0024</td>
<td>UA(K)=UB(K)</td>
</tr>
<tr>
<td>0025</td>
<td>DO 70 VA(K)=VB(K)</td>
</tr>
<tr>
<td>0026</td>
<td>M=M-1</td>
</tr>
<tr>
<td>0027</td>
<td>GO TO 20</td>
</tr>
<tr>
<td>0028</td>
<td>100 CONTINUE</td>
</tr>
<tr>
<td>0029</td>
<td>RETURN</td>
</tr>
<tr>
<td>0030</td>
<td>1000 FORMAT(/15H THE EPSLON (/D10.3,4H) CHECK IN SUBROUTINE MULTI INDICATES THAT ROOT(I,12,4H) = .023.16,34 + .023.16,2H 1,80H IS NO</td>
</tr>
<tr>
<td>0031</td>
<td>27 CLOSE ENOUGH TO BE A TRUE ROOT. IT IS PRINTED BELOW WITH MULTIP</td>
</tr>
<tr>
<td>0032</td>
<td>LICITY 0//)</td>
</tr>
</tbody>
</table>

END
### TABLE F. III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td><strong>SUBROUTINE DIVIDE(N, UP, VP, M, UD, VD, K, Q, VQ)</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>-----------------------------------------------------------------------</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>-----------------------------------------------------------------------</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>GIVEN TWO POLYNOMIALS FIXI AND GIXI, SUBROUTINE DIVIDE COMPUTES THE</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>QUOTIENT POLYNOMIAL XI = FIXI/GIXI.</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>-----------------------------------------------------------------------</strong></td>
</tr>
<tr>
<td>0002</td>
<td><strong>DOUBLE PRECISION UP, VP, UD, VD, Q, VQ, UTERM, VTERM, UDUMMY</strong></td>
</tr>
<tr>
<td>0003</td>
<td><strong>DIMENSION UP(26), VP(26), UD(26), VD(26), Q(26), VQ(26)</strong></td>
</tr>
<tr>
<td>0004</td>
<td><strong>K=N-M</strong></td>
</tr>
<tr>
<td>0005</td>
<td>**UDUMMY=UD(M+1)*UD(M+1)<em>VD(M+1)<em>VD(M+1)</em></em></td>
</tr>
<tr>
<td>0006</td>
<td>**VQ(K+1)=VQ(K+1)+VP(K+1)<em>UD(M+1)<em>VD(M+1)</em></em></td>
</tr>
<tr>
<td>0007</td>
<td>**VQ(K+1)=VQ(K+1)<em>UD(M+1)+VP(K+1)<em>UD(M+1)</em></em></td>
</tr>
<tr>
<td>0008</td>
<td><strong>IF(K,EQ.01 GO TO 100</strong></td>
</tr>
<tr>
<td>0009</td>
<td><strong>J=1</strong></td>
</tr>
<tr>
<td>0010</td>
<td><strong>DO 50 I=1,K</strong></td>
</tr>
<tr>
<td>0011</td>
<td><strong>J=J+1</strong></td>
</tr>
<tr>
<td>0012</td>
<td><strong>UTERM=VP(W-J)</strong></td>
</tr>
<tr>
<td>0013</td>
<td><strong>VTERM=VP(W-J)</strong></td>
</tr>
<tr>
<td>0014</td>
<td><strong>KK=K+1</strong></td>
</tr>
<tr>
<td>0015</td>
<td><strong>KNN=M-J</strong></td>
</tr>
<tr>
<td>0016</td>
<td><strong>DO 40 M=MNN-M</strong></td>
</tr>
<tr>
<td>0017</td>
<td><strong>IF(KK.GT.11 GO TO 10</strong></td>
</tr>
<tr>
<td>0018</td>
<td><strong>GO TO 45</strong></td>
</tr>
<tr>
<td>0019</td>
<td><strong>10 IF(KM.GE.11 GO TO 20</strong></td>
</tr>
<tr>
<td>0020</td>
<td><strong>GO TO 40</strong></td>
</tr>
<tr>
<td>0021</td>
<td>**20 UTERM=UTERM-UQ(KK)<em>UD(M+1)<em>VD(M+1)</em></em></td>
</tr>
<tr>
<td>0022</td>
<td>*<em>VTERM=VTERM-UQ(KK)<em>VD(M+1)</em></em></td>
</tr>
<tr>
<td>0023</td>
<td><strong>40 KK=KK-1</strong></td>
</tr>
<tr>
<td>0024</td>
<td>**45 UDUMMY=UD(M+1)<em>UD(M+1)<em>VD(M+1)</em></em></td>
</tr>
<tr>
<td>0025</td>
<td><strong>VQ(K+1)=VQ(K+1)+UTERM+VTERM+VD(M+1)</strong></td>
</tr>
<tr>
<td>0026</td>
<td><strong>50 VQ(K+1)=VQ(K+1)+UTERM+VTERM+UD(M+1)</strong></td>
</tr>
<tr>
<td>0027</td>
<td><strong>100 RETURN</strong></td>
</tr>
<tr>
<td>0028</td>
<td><strong>END</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>SUBROUTINE DERIV(N, UP, V, M, UA, VA)</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>-----------------------------------------------------------------------</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>-----------------------------------------------------------------------</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>ITS DERIVATIVE P'(X),</strong></td>
</tr>
<tr>
<td></td>
<td>C <strong>-----------------------------------------------------------------------</strong></td>
</tr>
<tr>
<td>0002</td>
<td><strong>DOUBLE PRECISION UP, V, UA, VA, AAA</strong></td>
</tr>
<tr>
<td>0003</td>
<td><strong>DIMENSION UP(26), VP(26), UA(26), VA(26)</strong></td>
</tr>
<tr>
<td>0004</td>
<td><strong>KKK=N+1</strong></td>
</tr>
<tr>
<td>0005</td>
<td><strong>DO 10 I=2,KKK</strong></td>
</tr>
<tr>
<td>0006</td>
<td><strong>AAA=1-1</strong></td>
</tr>
<tr>
<td>0007</td>
<td><strong>UA(I-1)=AAA*UP(I)</strong></td>
</tr>
<tr>
<td>0008</td>
<td><strong>10 VA(I-1)=AAA*VP(I)</strong></td>
</tr>
<tr>
<td>0009</td>
<td><strong>M=N-1</strong></td>
</tr>
<tr>
<td>0010</td>
<td><strong>RETURN</strong></td>
</tr>
<tr>
<td>0011</td>
<td><strong>END</strong></td>
</tr>
</tbody>
</table>
TABLE F. III (Continued)

0001 SUBROUTINE GCD(N, UR, VR, M, US, VS, M1, USS, VSS)

0012 DO 25 I=1, KKK

0013 DO 25 N=1, KKK

0016 DO 25 N=1, KKK

0018 DO 25 N=1, KKK

0021 DO 25 N=1, KKK

0022 DO 25 N=1, KKK

0025 DO 25 N=1, KKK

0026 DO 25 N=1, KKK

0027 UT1=U(N1)

0028 UT1=U(N1)

0029 UT1=U(N1)

0031 IF(UBB.GT. EPSLON) GO TO 100

0032 CONTINUE

0033 DO 90 I=1, M1

0034 DO 90 I=1, M1

0035 DO 90 I=1, M1

0036 DO 90 I=1, M1

0037 DO 90 I=1, M1

0038 DO 90 I=1, M1

0039 DO 90 I=1, M1

0040 DO 90 I=1, M1

0041 DO 90 I=1, M1

0042 DO 90 I=1, M1

0043 DO 90 I=1, M1

0044 DO 90 I=1, M1

0045 DO 90 I=1, M1

0046 DO 90 I=1, M1

0047 DO 90 I=1, M1

0048 DO 90 I=1, M1

0049 DO 90 I=1, M1
GO TO 30
KKK=K+1
DO 150 J=1,KKK
URR(J)=USSI(J)
VRR(J)=VSS(J)
USSI(J)=UT(J)
150 VSS(J)=VT(J)
KKK=K+2
NNN=M1+1
DO 160 J=KKK,NNN
URR(J)=USSI(J)
160 VRR(J)=VSS(J)
MI=M1
M1=K
GO TO 30
USSI(1)=1.0
VSS(1)=0.0
MI=0
RETURN
END
TABLE F.III (Continued)

0001 SUBROUTINE QUAD4, UA, VA, J, URODT, VRODT, MULTI

0002 C ***************************************************************
0003 C *
0004 C * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR
0005 C * MULTIPLEITIES
0006 C * OF EITHER A QUADRATIC POLYNOMIAL OR A LINEAR FACTOR. SOLUTION OF THE
0007 C * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.
0008 C *
0009 C ***************************************************************

0010 40 MULTI(1)=1
0011 1=1
0012 50 BBB=UA(2)*UA(2)*VA(Z1)*VA(Z)
0013 URODT(J)=-(VA(1)) UA(2)+VA(3)) UA(1)
0014 VRODT(J)=-(VA(1)) UA(2)+VA(3)) UA(1)
0015 GO TO 200
0016 60 UDSC=UA(2)*UA(3)*VA(Z1)*VA(Z2)*4.0-UA(3)*UA(1)*VA(1))
0017 VDISC=(x2*UA(1)+VA(2))-(4.0-UA(3)*UA(1)*VA(1))
0018 BBB=SQRT(UDDIC*UDIC*VDFC*VDFC)
0019 IF(BBB.LE.EPSLO4) GO TO 100
0020 IF(J.GE.0) GO TO 60
0021 MULTI(1)=1
0022 MULTI(2)=1
0023 IN
0024 80 CALL CONSDF(UDIC, VDISC, UTEMP, VTEMP)
0025 UD=2.0*UA(1)
0026 VD=2.0*VA(1)
0027 BBB=UD*UD+VD*VD
0028 URODT(J)=1.0*VA(2)+TEMP)*UD-1-VA(2)*VA(1))*UD/BBB
0029 VRODT(J)=1.0*VA(2)*VDFC+UD-1-VA(2)*VA(1))*UD/BBB
0030 URODT(J)=2.0*(-UA(2)-VDFC)*UD-1-VA(2)*VA(1))*UD/BBB
0031 VRODT(J)=2.0*(-UA(2)-VDFC)*UD-1-VA(2)*VA(1))*UD/BBB
0032 J=J+2
0033 GO TO 200
0034 100 IF(J.LT.0) GO TO 110
0035 110 MULTI(1)=2
0036 GO TO 130
0037 130 UD=2.0*UA(1)
0038 VD=2.0*VA(1)
0039 BBB=UD*UD+VD*VD
0040 URODT(J)=1.0*UA(2)*UD-1-VA(2)*VA(1))*UD/BBB
0041 VRODT(J)=1.0*UA(2)*UD-1-VA(2)*VA(1))*UD/BBB
0042 IF(J.LT.0) GO TO 110
0043 200 RETURN
0044 END
SUBROUTINE MULLER(UA,V;A;NP,UAPP,VAPP,NAPP,XSTART,XEND,UROOT,VROOT,
INROOT,ROOT,UAPP,VRAPP,NPOLY)

C--------------------------------------------------------------------------
C * MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLICITIES OF A
C * POLYNOMIAL OF MAXIMUM DEGREE 25. THROUGH THREE GIVEN POINTS THE
C * POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE QUADRATIC
C * CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
C * IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
C *--------------------------------------------------------------------------

DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,UROOT,VROOT,UX1,VX1,UAPP,VAPP
1,UX2,VX2,UX3,VX3,UX4,VX4,UX,0,VX,0,UX1,0,VX1,0,UX2,0,VX2,0,UX3,0,VX3,0
2RAPP,UPX4,VPX4,EPST,EPSP,EPSEM,EPSSM,UM3,V3,0,0,0,4,0,ABPX4,ABPX3
3,QQQ,XSTART,XEND

DIMENSION UROOT(25),VROOT(25),MULTIZSUAPP(25,3)VAPPI25,31,UWORK
1(26),VWORK|26.UB(26)VB(26),UA(26|VAI26ItURAPPI25,3)tVRAPP(25,3)

LOGICAL CONV

DOUBLE PRECISION EPS1

COMMON EPSRTEPSIEPSEPSOQEPSM,Z02,.AX

DATA PNAME,DNAME/2HP(,2HD(/

EPSRT=0.999

NROOT=0

IROOT=0

IPATH=1

NOMULT=0

NALTER=0

ITIME=0

IAPP=1

ITER=1

IF(IFAPP.NE.0) GOTO 18

NAPP=NP

CALL GENAPP(UAPP,VAPP,NAPP,XSTART)

GO TO 27

DO 25 I=1,NAPP

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

VAPP(I,2)=0.9*VAPPI,21)

UAPP(I,3)=1.1*UAPPII,2)

VAPP(I,3)=1.1*VAPPII*2

25 KKK=NP+1

DO 30 I=1,KKK

30 UWORK(I)=UA(I)

VWORK(II=VA(I)

DO 40 I=1,NAPP

30 UWORK(I)=VA(I)

40 UX1=UAPP(I,AAPP,1)

VX1=VAPP(IIAPP,1)

UX2=UAPP(I,AAPP,2)

VX2=VAPP(IIAPP,2)

UX3=UAPP(I,AAPP,3)

VX3=VAPP(IIAPP,3)

30 CALL HORNERN(UWORK,UX1,VX1,UX2,VX2,UX3,VX3,UX4,VX4,UX5,VX5)

30 CALL HORNERN(UWORK,UX1,VX1,UX2,VX2,UX3,VX3,UX4,VX4)

30 ABPX4=DSQRT(UPX4*UPX4+VPX4*VPX4)

30 ABPX3=DSQRT(UPX3*UPX3+VPX3*VPX3)

TABLE F.III (Continued)
TABLE F.III (Continued)

0044  IF(ABPX3.EQ.0.0) GO TO 70
0045  EQ=ABPX4/ABPX3
0046  IF(QQQ.LE.10.) GO TO 70
0047  UQ4=0.5*UQ4
0048  VX4=UX3+(UH3*UQ4-VH3*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(UAPP1,VAPP1,APPI1,APPI2,VAPP1,APPI3)
0056  IF(NALTER.GT.5) GO TO 75
0057  ITER=1
0058  GO TO 40
0059  IF(ITER.LT.MAX) GO TO 100
0060  IF(XEND.EQ.0.0) GO TO 77
0061  IF(XSTART.GT.XEND) GO TO 77
0062  CALL GENAPP(UAPP1,VAPP1,APP1,3)
0063  CALL BETTER(UA,VA,APP1,NPOLY,VRAPP1,VRAPP1.MULT)
0064  RETURN
0065  IF(NPOLY.EQ.0) GO TO 90
0066  IF(I1.EQ.0) GO TO 85
0067  WRITE(102,1085) IUROOT,IVROOT,VRAPP1
0068  IF(ITER.LT.NROOT) GO TO 85
0069  GO TO 87
0070  WRITE(102,1086) (IUROOT,IVROOT,VRAPP1)
0071  RETURN
0072  RETURN
0073  WRITE(102,1080)
0074  WRITE(102,1070)
0075  WRITE(102,1090)
0076  WRITE(102,1035) (UWNAME,J,WORK(J),VWORK(J),J=1,KKK)
0077  WRITE(102,1095) IUROOT,IVROOT,VRAPP1
0078  IF(ITER.LT.NROOT) GO TO 85
0079  GO TO 90
0080  WRITE(102,1086) (IUROOT,IVROOT,VRAPP1,I=1,KKK)
0081  RETURN
0082  RETURN
0083  WRITE(102,1085) IUROOT,IVROOT,VRAPP1
0084  WRITE(102,1090) IUROOT,IVROOT,VRAPP1
0085  IF(ITER.LT.NROOT) GO TO 85
0086  GO TO 90
0087  WRITE(102,1080) NOPOLY
0088  RETURN
0089  RETURN
0090  RETURN
0091  RETURN
0092  RETURN
0093  RETURN
0094  RETURN
0095  RETURN
0096  RETURN
0097  RETURN
0098  RETURN
0099  RETURN
0100  RETURN

243
TABLE F.III (Continued)

0101  URAPP(NURO0,3)=UAPP(11,APP,3)
0102  VRAPP(NURO0,3)=VAPP(1,APP,3)
0103  125 IFINMULT.LT.NP) GO TO 130
0104  DO TO 80
0105  130 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB4,VB4,UP4,VP4)
0106  NWORK=NWORK-1
0107  KKK=NWORK+1
0108  DO 140 =1,KKK
0109  UWORK(I)=UB(I)
0110  VWORK(I)=VB(I)
0111  CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB4,VB4,UP4,VP4)
0112  CCC=DSQRT(UP4*UP4+VP4*VP4)
0113  IF(CCC.LT.EPSM) GO TO 150
0114  IF(INWORK.GT.2) GO TO 75
0115  IROOT=NURO0
0116  KKK=NWORK+1
0117  DO 140 =1,KKK
0118  UB(I)=UWORK(KKK+1-I)
0119  VB(I)=VWORK(KKK+1-I)
0120  CALL QUADINWORK,UB,VB,NROOT,UROOT,VROOT,MULT
0121  GO TO 80
0122  150 MULT(NROOT)=MULT(NROOT)+1
0123  NDMULT=NDMULT+1
0124  GO TO 125
0125  110 UX1=UX2
0126  VX1=VX2
0127  UX2=UX3
0128  VX2=VX3
0129  UX3=UX4
0130  VX3=VX4
0131  UPX1=UPX2
0132  VPX1=VPX2
0133  UPX2=UPX3
0134  VPX2=VPX3
0135  UPX3=UPX4
0136  VPX3=VPX4
0137  ITER=ITER+1
0138  GO TO 50
0139  1090 FORMAT(2X,11X,65HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO ZEROS WERE FOUND//)
0140  1090 FORMAT(2X,11X,65HINITIAL APPROXIMATION//)
0141  1070 FORMAT(4H,4H)I
0142  1086 FORMAT(2X,5HROOTF(12,4H)=.023.16,3H+.023.16,2H 1.19X,2H3H
0143  1035 FORMAT(3X,2A12,4H)=.023.16,3H+.023.16,2H 11
0144  1050 FORMAT(18X,.023.16,3H+.023.16,2H 1/.023.16,3H+.023.16,2H 11
0145  1089 FORMAT(2X,5HROOT(12,4H)=.023.16,3H+.023.16,2H 1.18X,2H3H
0146  END
**TABLE F.III (Continued)**

```fortran
SUBROUTINE BETTER(UAVAtNP,UROOT,VROOT),URAPPI,VRAPP,1ROOTIMUL

*************************************************************************
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,UBAPP,VBAPP,UX1,VX1,UX2,VX2,UX3
1,VX3,UP1,VPX1,UPX2,VPX2,UPX3,VPX3,UB,V,UROOTS,VRAPP,EPSRT,UK4,V
2X4,URAPP,VRAPP,EPS0,EPS,UQ4,UX4,UX5,UX3

LOGICAL CONV

DIMENSION UROOT(25),VROOT(25),UA(26),VA(26),UBAPP(25,3),VRAPP(25,3)
1)

DOUBLE PRECISION EPS1,EPSH

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

IF(NROOT.LE.10 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)

ITER=1
CALL HORNER(NPUAVAX,UVX1,VX1,UBPVPX1)
CALL HORNER(NPUAVAX2,VX2,UBAPP2VPX2)
CALL HORNER(NPUAVAX3,VX3,UBAPP3VPX3)
CALL CALLUXIVX12UX2,VX2,UX3,VX3,UPX1VPX1,VPX2,VPX3,VPX3,UX
14,UX4,UX4,UX4,UX4,UX4,UX4

IF(CONV) GO TO 50
IF(ITER.LT.MAX) GO TO 40
WRITE(IO2,1000) UX4,VX4
IF(J.LT.IROOT) GO TO 33
IF(J.EQ.IROOT) GO TO 35
GO TO 100

33 KKK=IROOT-1

34 KKK=KKK

35 IROOT=IROOT-1

GO TO 100
```

---

TABLE F.III (Continued)
TABLE F.III (Continued)

0046  40 UX1=UX2
0047  VX1=VX2
0048  UX2=UX3
0049  VX2=VX3
0050  UX3=UX4
0051  VX3=VX4
0052  UPX1=UPX2
0053  VPX1=VPX2
0054  UPX2=UPX3
0055  VPX2=VPX3
0056  ITER=ITER+1
0057  GO TO 20
0058  50 L=L+1
0059  VROOTS(LI=UX4
0060  VROOTS(LJ=VX4
0061  100 CONTINUE
0062  IF(L.EQ.0) GO TO 120
0063  DD 110 I=1,L
0064  UROOT(I)=UROOTS(I)
0065  VROOT(I)=VROOTS(I)
0066  NROOT=L
0067  RETURN
0068  120 NROOT=0
0069  RETURN
0070  1000 FORMAT(42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(I,12,4H) = ,
0071  123.16,3H + .023.16,2H I/24H DID NOT CONVERGE AFTER ,13,11H ITERAT
0072  210NS)
0073  1010 FORMAT(30H THE PRESENT APPROXIMATION IS ,D23.16,3H + .D23.16,2H 1/
0074  1/1)
0075  END
TABLE F.III (Continued)

0001 SUBROUTINE ALTER(X1R,X1I,X2R,X2I,X3R,X3I,NALTER,ITIME)

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

0002 DOUBLE PRECISION X1R,X1I,X2R,X2I,X3R,X3I,EPS1,EPS2,EPS3,R,BETA
0003 COMMON EPS1,EPS2,EPS3,EPS4,EPS5,MAX
0005 IF(ITIME.NE.0) GO TO 5
0006 ITIME=1
0007 WRITE(IO2,1010) MAX
0008 IF(INALTER.EQ.0) GO TO 10
0009 WRITE(IO2,1000) X1R,X1I,X2R,X2I,X3R,X3I
0010 GO TO 20
0011 10 R=DSQRT(X2R*X2I+X2R*X2I)
0012 BETA=DATAN2(X2R,X2I)
0013 WRITE(IO2,1020) X1R,X1I,X2R,X2I
0014 20 NALTER=NALTER+1
0015 IF(INALTER.GT.5) RETURN
0016 IF(INALTER.GT.5) RETURN
0017 30 X2R=-X2R
0018 X2I=-X2I
0019 GO TO 50
0020 40 BETA=BETA+1.0471976 
0021 X2R=R*DCOS(BETA)
0022 X2I=R*DSIN(BETA)
0023 50 X1R=0.9*X2R
0024 X1I=0.9*X2I
0025 X3R=1.1*X2R
0026 X3I=1.1*X2I
0027 RETURN
0028 1000 FORMAT(I5,5H1X5K1 = D23.16,3H + D23.16,2H 1,10X,22HALTERED APPROXIM *
0029 ATE/5X1K1 = D23.16,3H + D23.16,2H 1/1X,5K1 = D23.16,3H + *
0029 Z + D23.16,2H 1/1)
0029 1020 FORMAT(I5,5H1X5K2 = D23.16,3H + D23.16,2H 1,10X,22HALTERED APPROXIM *
0030 ATE/5X1K2 = D23.16,3H + D23.16,2H 1/1X,5K2 = D23.16,3H + *
0030 Z + D23.16,2H 1/1)
0030 1010 FORMAT(///5X,5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF *
0031 TER/13,12H ITERATIONS///)
0031 END
TABLE F.III (Continued)

SUBROUTINE GENAPP(APPR, APP1, NAPP, XSTART)

**************************************************************************
DOUBLE PRECISION APPR, APP1, XSTART, EPS1, EPS2, EPS3, BETA
COMMON EPSR, EPS1, EPS2, EPS3, EPS4, IO2, MAX
IF(XSTART.EQ.0.0D0) XSTART=0.5
BETA=0.2617994
DO 10 I=1, NAPP
APPRI(I,2)=XSTART*DCOS(I*BETA)
APPPII,2)=XSTART*DSSIN(I*BETA)
BETA=BETA+0.5235988
10 XSTART=XSTART+0.5
DO 20 I=1, NAPP
APPRI(I,1)=0.9*APPR(I,2)
APPPI(I,1)=0.9*APPPIII,2)
APPRI(I,3)=1.1*APPR(I,2)
20 APPPII,3)=1.1*APPPIII,2)
RETURN
END
### TABLE F.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE TEST(UX3, VX3, UX4, VX4, CONV)</td>
</tr>
<tr>
<td>C</td>
<td>* SUBROUTINE TEST CHECKS FOR CONVERGENCE OF THE SEQUENCE OF APPROXIMATIONS</td>
</tr>
<tr>
<td>C</td>
<td>* BY TESTING THE EXPRESSION</td>
</tr>
<tr>
<td>C</td>
<td>* ABSOLUTE VALUE OF (XN+1 - XN)/ABSOLUTE VALUE OF XN*).</td>
</tr>
<tr>
<td>C</td>
<td>* WHEN IT IS AS SMALL AS DESIRED, CONVERGENCE IS OBTAINED.</td>
</tr>
<tr>
<td>C</td>
<td>*</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION UX3, VX3, UX4, VX4, EPSRT, EPSO, EPSA, AAA, UDUMMY, VDUMMY</td>
</tr>
<tr>
<td>0003</td>
<td>IDENOM</td>
</tr>
<tr>
<td>0004</td>
<td>LOGICAL CONV</td>
</tr>
<tr>
<td>0005</td>
<td>DOUBLE PRECISION EPS1, EPSM</td>
</tr>
<tr>
<td>0006</td>
<td>COMMON EPSRT, EPS1, EPS, EPSM, IO2, MAX</td>
</tr>
<tr>
<td>0007</td>
<td>UDUMMY=UX4-UX3</td>
</tr>
<tr>
<td>0008</td>
<td>VDUMMY=VX4-VX3</td>
</tr>
<tr>
<td>0009</td>
<td>AAA=DSQRT(UDUMMY<em>UDUMMY+VDUMMY</em>VDUMMY)</td>
</tr>
<tr>
<td>0010</td>
<td>IF(IO2/IDENOM+LT.EPSO) GO TO 20</td>
</tr>
<tr>
<td>0011</td>
<td>IF(AAA/IDENOM+LT.EPS) GO TO 10</td>
</tr>
<tr>
<td>0012</td>
<td>IF(AAA.LT.EPSO) GO TO 10</td>
</tr>
<tr>
<td>0013</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0014</td>
<td>10 CONV=.FALSE.</td>
</tr>
<tr>
<td>0015</td>
<td>100 RETURN</td>
</tr>
<tr>
<td>0016</td>
<td>END</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE HORNER(NA, UA, VA, UX, VX, UB, VB, UPX, VPX)</td>
</tr>
<tr>
<td>C</td>
<td>* HORNER'S METHOD COMPUTES THE VALUE OF THE POLYNOMIAL P(X) AT A POINT O.</td>
</tr>
<tr>
<td>C</td>
<td>* SYNTHETIC DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE</td>
</tr>
<tr>
<td>C</td>
<td>* FACTOR (X-D).</td>
</tr>
<tr>
<td>C</td>
<td>*</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION UX, VX, UPX, VPX, UB, VB, UA, VA</td>
</tr>
<tr>
<td>0003</td>
<td>DIMENSION UA(26), VA(26), UB(26), VB(26)</td>
</tr>
<tr>
<td>0004</td>
<td>UB(1)=UA(1)</td>
</tr>
<tr>
<td>0005</td>
<td>VB(1)=VA(1)</td>
</tr>
<tr>
<td>0006</td>
<td>NUM=UA-1</td>
</tr>
<tr>
<td>0007</td>
<td>GO TO 10 *(=2, NUM)</td>
</tr>
<tr>
<td>0008</td>
<td>UB(I)=UA(I)+UB(I-1)*UX-UB(I-1)*VX</td>
</tr>
<tr>
<td>0009</td>
<td>10 VB(I)=VA(I)+VB(I-1)*UX+UB(I-1)*VX</td>
</tr>
<tr>
<td>0010</td>
<td>UPX=UB(NUM)</td>
</tr>
<tr>
<td>0011</td>
<td>VPX=VB(NUM)</td>
</tr>
<tr>
<td>0012</td>
<td>RETURN</td>
</tr>
<tr>
<td>0013</td>
<td>END</td>
</tr>
</tbody>
</table>
**TABLE F.III (Continued)**

<table>
<thead>
<tr>
<th>Page 0001</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SUBROUTINE CALC(UX1, VX1, UX2, VX2, UX3, VX3, VPX1, VPX2, VPX3, V</strong></td>
</tr>
<tr>
<td>* GIVEN THREE APPROXIMATIONS XIN-2, XIN-1, AND XIN, SUBROUTINE CALC</td>
</tr>
<tr>
<td>* APPROXIMATES THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF</td>
</tr>
<tr>
<td>* THE QUADRATIC CLOSEST TO XIN. THIS ZERO IS THE NEW APPROXIMATION</td>
</tr>
<tr>
<td>* XIN TO THE ZERO OF THE POLYNOMIAL.</td>
</tr>
<tr>
<td><strong>DOUBLE PRECISION ARG1, ARG2</strong></td>
</tr>
<tr>
<td><strong>DOUBLE PRECISION UPX3, VPX3, UPX2, VPX2, UX1, VX1, UX2, VX2, UX3, VX3, UPX1,</strong></td>
</tr>
<tr>
<td>LVPX1, UH3, VH3, UH2, VH2, UQ3, UQ0, UO, UH, UB, UC, UDISC, VDISC, UCCE, VC</td>
</tr>
<tr>
<td>ZC, UDENG, VDENG, UO, UB, UB, VX3, VX2, VX1, UX2, VU3, VX1, VX2, VX3, VX1, UQ3</td>
</tr>
<tr>
<td><strong>COMMON EPSRT, EPS1, EPS2, EPSM, IO2</strong></td>
</tr>
<tr>
<td><strong>DOUBLE PRECISION THETA, ANGLE, UTEST, VTEST</strong></td>
</tr>
<tr>
<td><strong>O002</strong></td>
</tr>
<tr>
<td><strong>O003</strong></td>
</tr>
<tr>
<td><strong>O004</strong></td>
</tr>
<tr>
<td><strong>O005</strong></td>
</tr>
<tr>
<td><strong>O006</strong></td>
</tr>
<tr>
<td><strong>O007</strong></td>
</tr>
<tr>
<td><strong>O008</strong></td>
</tr>
<tr>
<td><strong>O009</strong></td>
</tr>
<tr>
<td><strong>O010</strong></td>
</tr>
<tr>
<td><strong>O011</strong></td>
</tr>
<tr>
<td><strong>O012</strong></td>
</tr>
<tr>
<td><strong>O013</strong></td>
</tr>
<tr>
<td><strong>O014</strong></td>
</tr>
<tr>
<td><strong>O015</strong></td>
</tr>
<tr>
<td><strong>O016</strong></td>
</tr>
<tr>
<td><strong>O017</strong></td>
</tr>
<tr>
<td><strong>O018</strong></td>
</tr>
<tr>
<td><strong>O019</strong></td>
</tr>
<tr>
<td><strong>O020</strong></td>
</tr>
<tr>
<td><strong>O021</strong></td>
</tr>
<tr>
<td><strong>O022</strong></td>
</tr>
<tr>
<td><strong>O023</strong></td>
</tr>
<tr>
<td><strong>O024</strong></td>
</tr>
<tr>
<td><strong>O025</strong></td>
</tr>
<tr>
<td><strong>O026</strong></td>
</tr>
<tr>
<td><strong>O027</strong></td>
</tr>
<tr>
<td><strong>O028</strong></td>
</tr>
<tr>
<td><strong>O029</strong></td>
</tr>
<tr>
<td><strong>O030</strong></td>
</tr>
<tr>
<td><strong>O031</strong></td>
</tr>
<tr>
<td><strong>O032</strong></td>
</tr>
<tr>
<td><strong>O033</strong></td>
</tr>
<tr>
<td><strong>O034</strong></td>
</tr>
<tr>
<td><strong>O035</strong></td>
</tr>
<tr>
<td><strong>O036</strong></td>
</tr>
<tr>
<td><strong>O037</strong></td>
</tr>
<tr>
<td><strong>O038</strong></td>
</tr>
<tr>
<td><strong>O039</strong></td>
</tr>
<tr>
<td><strong>O040</strong></td>
</tr>
<tr>
<td><strong>O041</strong></td>
</tr>
<tr>
<td><strong>O042</strong></td>
</tr>
<tr>
<td><strong>O043</strong></td>
</tr>
<tr>
<td><strong>O044</strong></td>
</tr>
</tbody>
</table>

**250**
ARG1 = UDEN1*UDEN1 + VDEN1*VDEN1
ARG2 = UDEN2*UDEN2 + VDEN2*VDEN2
AAA = DSQRT(ARG1)
BBB = DSQRT(ARG2)
IF (AAA .LT. BBB) GO TO 10
IF (AAA .EQ. 0.0) GO TO 60
UAAA = -2.0*UC
VAAA = -2.0*VC
UQ4 = (UAAA*UDEN1 + VAAA*VDEN1)/ARG1
VQ4 = (VAAA*UDEN1 - UAAA*VDEN1)/ARG1
GO TO 50
10 IF (BBB .EQ. 0.0) GO TO 60
UAAA = -2.0*UC
VAAA = -2.0*VC
UQ4 = (UAAA*UDEN2 + VAAA*VDEN2)/ARG2
VQ4 = (VAAA*UDEN2 - UAAA*VDEN2)/ARG2
GO TO 50
50 UX4 = UX3 + (VH3*UQ4 - VH3*VQ4)
VX4 = VX3 + (VH3*UQ4 + VH3*VQ4)
RETURN
60 UQ4 = 1.0
VQ4 = 0.0
GO TO 50
END
TABLE F.III (Continued)

```
0001 "SUBROUTINE CONSQT(UX, VX, UY, VY)
C **********************************************************************************************
C *
C * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
C *
C **********************************************************************************************
C 0002 DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, A, B
C 0003 R=DABS(UX)+DABS(VX)
C 0004 A=DABS((R+UX/2.0)1/2)
C 0005 B=DABS((R-UX/2.0)1/2)
C 0006 IF UX > 10, 20, 30
C 0007 10 UY=A
C 0008 VY=-1.0*B
C 0009 GO TO 100
C 0010 20 IF UX < 40, 50, 60
C 0011 30 UY=A
C 0012 VY=B
C 0013 GO TO 100
C 0014 40 DUMMY=DABS(UX)
C 0015 UY=0.0
C 0016 VY=DABS(DUMMY)
C 0017 GO TO 100
C 0018 50 UY=0.0
C 0019 VY=0.0
C 0020 GO TO 100
C 0021 60 DUMMY=DABS(UX)
C 0022 UY=DABS(DUMMY)
C 0023 VY=0.0
C 0024 100 RETURN
C 0025 END
```
APPENDIX G

REPEATED G.C.D. — NEWTON'S METHOD

1. Use of the Program

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

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

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

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

Main Program

Data Entry/1H1,1H2,...,1H9,2H10,2H11,...,2HXX/where XX = N+1
UP(N+1), VP(N+1)
UAPP(N), VAPP(N)
UDO(N+1), VDO(N+1)
UDDO(N+1), VDDO(N+1)
UD1(N+1), VD1(N+1)
UD2(N+1), VD2(N+1)
UDD1(N+1), VDD1(N+1)
UG(N+1), VG(N+1)
UD3(2N+1), VD3(2N+1)
UD4(2N+1), VD4(2N+1)
UZROS(N), VZROS(N)
UAP(N), VAP(N)
UROOT(N), VROOT(N)
NULT(N)
ENTRY(N+1)

Subroutine PROD

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

Subroutine ZROS

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

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

See corresponding subroutine in Table E.I.

Subroutine QUAD

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

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

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

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

4. Description of Program Output

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

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

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

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

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

5. Informative Messages and Error Messages

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

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

"QUAD FOUND XXX TO BE A MULTIPLE ROOT." XXX represents the value of the root found as a multiple root by Subroutine QUAD.
<table>
<thead>
<tr>
<th>NO</th>
<th>POLY</th>
<th>MAX</th>
<th>EPS1</th>
<th>EPS2</th>
<th>EPS3</th>
<th>XSTART</th>
<th>XEND</th>
<th>CHECK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>200</td>
<td>1.D-03</td>
<td>1.D-10</td>
<td>1.D-20</td>
<td>1.0D+01</td>
<td>2.0D+01</td>
<td></td>
</tr>
</tbody>
</table>

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

**REPEATED GCD – NEWTON'S METHOD**

<table>
<thead>
<tr>
<th>Single Precision</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>Variable Type</td>
<td>Variable Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable Type</td>
<td>Variable Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable Type</td>
<td>Variable Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable Type</td>
<td>Variable Type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Main Program**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD</td>
<td>I</td>
<td>Number of distinct roots found</td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>Number of roots found</td>
</tr>
<tr>
<td>J1</td>
<td>I</td>
<td>Multiplicity of given root</td>
</tr>
<tr>
<td>DO</td>
<td>C</td>
<td>Array of coefficients of original polynomial</td>
</tr>
<tr>
<td>NDO</td>
<td>I</td>
<td>Degree of original polynomial</td>
</tr>
<tr>
<td>DDO</td>
<td>C</td>
<td>Array of coefficients of derivative of DO(X) i.e. DO'(X)</td>
</tr>
<tr>
<td>NDDO</td>
<td>I</td>
<td>Degree of DDO(X)</td>
</tr>
<tr>
<td>D1</td>
<td>C</td>
<td>Array of coefficients of g.c.d. of DO(X) and DDO(X)</td>
</tr>
<tr>
<td>NDI</td>
<td>I</td>
<td>Degree of D1(X)</td>
</tr>
<tr>
<td>DDI</td>
<td>C</td>
<td>Array of coefficients of derivative of D1(X) i.e. D1'(X)</td>
</tr>
<tr>
<td>NDD1</td>
<td>I</td>
<td>Degree of DDI(X)</td>
</tr>
<tr>
<td>D2</td>
<td>C</td>
<td>Array of coefficients of g.c.d. of D1(X) and DDI(X)</td>
</tr>
<tr>
<td>N2</td>
<td>I</td>
<td>Degree of D2(X)</td>
</tr>
<tr>
<td>D3</td>
<td>C</td>
<td>Array of coefficients of the product of DO(X) and D2(X)</td>
</tr>
<tr>
<td>N3</td>
<td>I</td>
<td>Degree of D3(X)</td>
</tr>
<tr>
<td>D4</td>
<td>C</td>
<td>Array of coefficients of the square of D1(X)</td>
</tr>
<tr>
<td>N4</td>
<td>I</td>
<td>Degree of D4(X)</td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td>Array of coefficients of the quotient D3(X)/D4(X)</td>
</tr>
<tr>
<td>NG</td>
<td>I</td>
<td>Degree of G(X)</td>
</tr>
<tr>
<td>ZROS</td>
<td>C</td>
<td>Array of roots of G(X)</td>
</tr>
</tbody>
</table>

**Subroutine ZROS**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APROX</td>
<td>C</td>
<td>Starting approximation (initial or altered)</td>
</tr>
</tbody>
</table>
### TABLE G.II (Continued)

<table>
<thead>
<tr>
<th>Single Precision Variable</th>
<th>Type</th>
<th>Double Precision Variable 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>
</tr>
<tr>
<td>F</td>
<td>C</td>
<td>UF,VF</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>N</td>
<td>I</td>
<td>E</td>
</tr>
<tr>
<td>G</td>
<td>C</td>
<td>UC,VG</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>MN</td>
<td>I</td>
<td>MN</td>
<td>I</td>
<td>R</td>
</tr>
<tr>
<td>H</td>
<td>C</td>
<td>UH,VH</td>
<td>D</td>
<td>R</td>
</tr>
<tr>
<td>LIMIT</td>
<td>I</td>
<td>LIMIT</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>I</td>
<td>K</td>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>

**Subroutine PROD**

- **M**: Degree of polynomial to be multiplied
- **F**: Array of coefficients of polynomial to be multiplied
- **N**: Degree of polynomial to be multiplied
- **G**: Array of coefficients of polynomial to be multiplied
- **MN**: Degree of product polynomial $H(X)$
- **H**: Array of coefficients of product polynomial
- **LIMIT**: Number of coefficients of polynomial $F(X)$
- **K**: Counter
Figure G.2. Flow Charts for Repeated G.C.D.-Newton's Method
Figure G.2. (Continued)
Figure G.2. (Continued)
Figure G.2. (Continued)
Figure G.2. (Continued)
Figure G.2. (Continued)
Figure G.2, (Continued)
Figure G.2. (Continued)
Figure G.2. (Continued)
Figure G.2. (Continued)
## PROGRAM FOR REPEATED G.C.D. – NEWTON’S METHOD

- **DOUBLE PRECISION PROGRAM FOR THE REPEATED G.C.D. – NEWTON’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.**

### Table G.III

<table>
<thead>
<tr>
<th>0001</th>
<th>0002</th>
<th>0003</th>
<th>0004</th>
<th>0005</th>
<th>0006</th>
<th>0007</th>
<th>0008</th>
<th>0009</th>
<th>0010</th>
<th>0011</th>
<th>0012</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOUB</td>
<td>DOUB</td>
<td>DOUB</td>
<td>DIMEN</td>
<td>COMMON</td>
<td>DATA</td>
<td>DATA</td>
<td>DATA</td>
<td>DATA</td>
<td>READ</td>
<td>READ</td>
<td>READ</td>
</tr>
<tr>
<td>LUE</td>
<td>LUE</td>
<td>LUE</td>
<td>SIZE</td>
<td>EPS1, EPS2, EPS3, UP, VP, UAPP, VAPP, UD0, VDO, UD00, VDO0, 1U01, V01, U02, V02, UD01, VD01, UG, VG, UD3, VD3, U04, V04, UZROS, VZROS, UAP, VA</td>
<td>DOUBLE PRECISION EPS1, EPS2, EPS3, UP, VP, UAPP, VAPP, UD0, VDO, UD00, VDO0, 1U01, V01, U02, V02, UD01, VD01, UG, VG, UD3, VD3, U04, V04, UZROS, VZROS, UAP, VA</td>
<td>DOUBLE PRECISION XSTART</td>
<td>DOUBLE PRECISION XEND</td>
<td>ANAME(12), UP(26), VP(26), UAPP(25), VAPP(25), UD01(26), VD01(26), UD0(26), VD0(26), UD00(26), VDO0(26)</td>
<td>ANAME(12), UROGT(26), VROOT(25), MULT(25), ENTRY(26)</td>
<td>EPS1, EPS2, EPS3, I02, MAX</td>
<td>/4H****/</td>
</tr>
<tr>
<td>10001</td>
<td>10002</td>
<td>10003</td>
<td>10004</td>
<td>10005</td>
<td>10006</td>
<td>10007</td>
<td>10008</td>
<td>10009</td>
<td>10100</td>
<td>10110</td>
<td>10120</td>
</tr>
<tr>
<td>IFLK</td>
<td>IFLK</td>
<td>IFLK</td>
<td>COMMON</td>
<td>DATA</td>
<td>DATA</td>
<td>DATA</td>
<td>DATA</td>
<td>READ</td>
<td>READ</td>
<td>READ</td>
<td></td>
</tr>
<tr>
<td>CHECK EQ.1) STOP</td>
<td>WRITE(102,1020) ANAME11, ANAME21, NAPP, MAX</td>
<td>WRITE(102,2000) NAPP</td>
<td>WRITE(102,2010) NAX</td>
<td>WRITE(102,2070) EPS1</td>
<td>WRITE(102,2020) EPS2</td>
<td>WRITE(102,2030) EPS3</td>
<td>WRITE(102,2040) XSTART</td>
<td>WRITE(102,2050) XEND</td>
<td>WRITE(102,2060) X END</td>
<td>WRITE(102,2070) X END</td>
<td>WRITE(102,2080) X E N D</td>
</tr>
<tr>
<td>1K</td>
<td>1K</td>
<td>1K</td>
<td>EPS1, EPS2, EPS3, I02, MAX</td>
<td>DATA PNAME, NAME(2), NAME(2), NAME(3)</td>
<td>DATA PNAME, NAME(2), NAME(3)</td>
<td>DATA PNAME, NAME(2), NAME(3)</td>
<td>DATA PNAME, NAME(2), NAME(3)</td>
<td>READ</td>
<td>READ</td>
<td>READ</td>
<td></td>
</tr>
<tr>
<td>013</td>
<td>014</td>
<td>015</td>
<td>016</td>
<td>017</td>
<td>018</td>
<td>019</td>
<td>020</td>
<td>021</td>
<td>022</td>
<td>023</td>
<td></td>
</tr>
<tr>
<td>IF (KCHECK.EQ.1) STOP</td>
<td>WRITE(102,1020) ANAME11, ANAME21, NAPP, MAX</td>
<td>WRITE(102,2000) NAPP</td>
<td>WRITE(102,2010) NAX</td>
<td>WRITE(102,2070) EPS1</td>
<td>WRITE(102,2020) EPS2</td>
<td>WRITE(102,2030) EPS3</td>
<td>WRITE(102,2040) XSTART</td>
<td>WRITE(102,2050) XEND</td>
<td>WRITE(102,2060) X END</td>
<td>WRITE(102,2070) X END</td>
<td></td>
</tr>
<tr>
<td>024</td>
<td>025</td>
<td>026</td>
<td>027</td>
<td>028</td>
<td>029</td>
<td>030</td>
<td>031</td>
<td>032</td>
<td>033</td>
<td>034</td>
<td></td>
</tr>
<tr>
<td>NN=KK+1</td>
<td>ON 5 I=1, KKK</td>
<td>ON 5 I=1, KKK</td>
<td>5 READ(101, 1010) UP(JJJ), VP(JJJ)</td>
<td>IF (NAPP, NE.01) GO TO 22</td>
<td>NAPP=NAPP</td>
<td>CALL GENAPP(UAPP, VAPP, NAPP, XSTART)</td>
<td>GO TO 23</td>
<td>22 READ(101, 1015) (UAPP(I), VAPP(I), I=1, NAPP)</td>
<td>23 WRITE(102, 1030) NP</td>
<td>035</td>
<td></td>
</tr>
<tr>
<td>036</td>
<td>037</td>
<td>038</td>
<td>039</td>
<td>040</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN=KK+1</td>
<td>ON 5 I=1, KKK</td>
<td>ON 5 I=1, KKK</td>
<td>8 WRITE(102, 1040) PNAME, ENTRY(JJJJ), UI(JJJJ), VP(JJJJ)</td>
<td>K=0</td>
<td>K=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE G.11 (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0041</td>
<td>J1=1</td>
</tr>
<tr>
<td>0042</td>
<td>KKK=NP+1</td>
</tr>
<tr>
<td>0043</td>
<td>DO 10 I=1,KKK</td>
</tr>
<tr>
<td>0044</td>
<td>UDD(I)=UP(I)</td>
</tr>
<tr>
<td>0045</td>
<td>10 VDD(I)=VP(I)</td>
</tr>
<tr>
<td>0046</td>
<td>NDD=UP</td>
</tr>
<tr>
<td>0047</td>
<td>CALL DERIV(NDO,UDO,VDO,NG0,UDP,VD0)</td>
</tr>
<tr>
<td>0048</td>
<td>CALL GCD(NDO,UDO,VDO,NG0,UDP,VD0,ND1,VD1)</td>
</tr>
<tr>
<td>0049</td>
<td>20 WRITE(IO2,1000) (*STRT, I=1,33)</td>
</tr>
<tr>
<td>0050</td>
<td>IF(NDI.LE.1) GO TO 30</td>
</tr>
<tr>
<td>0051</td>
<td>GO TO 40</td>
</tr>
<tr>
<td>0052</td>
<td>30 UO2(I)=1.0</td>
</tr>
<tr>
<td>0053</td>
<td>VD2(I)=0.0</td>
</tr>
<tr>
<td>0054</td>
<td>NDO=0</td>
</tr>
<tr>
<td>0055</td>
<td>GO TO 50</td>
</tr>
<tr>
<td>0056</td>
<td>40 CALL DERIV(ND1,UD1,VD1,NDD,UDP,VDD)</td>
</tr>
<tr>
<td>0057</td>
<td>CALL GCD(ND1,UD1,VD1,NDD,UDP,VDD,ND2,UD2,VD2)</td>
</tr>
<tr>
<td>0058</td>
<td>50 IF(NDO+ND2.LE.2*NG0) GO TO 60</td>
</tr>
<tr>
<td>0059</td>
<td>GO TO 70</td>
</tr>
<tr>
<td>0060</td>
<td>60 WRITE(IO2,1025) J1</td>
</tr>
<tr>
<td>0061</td>
<td>GO TO 170</td>
</tr>
<tr>
<td>0062</td>
<td>70 IF(NDI.EQ.0) GO TO 80</td>
</tr>
<tr>
<td>0063</td>
<td>GO TO 90</td>
</tr>
<tr>
<td>0064</td>
<td>80 KKK=NG0+1</td>
</tr>
<tr>
<td>0065</td>
<td>DO 85 I=1,KKK</td>
</tr>
<tr>
<td>0066</td>
<td>UG(I)=UDD(I)</td>
</tr>
<tr>
<td>0067</td>
<td>85 VG(I)=VDD(I)</td>
</tr>
<tr>
<td>0068</td>
<td>NG0=NDD</td>
</tr>
<tr>
<td>0069</td>
<td>GO TO 110</td>
</tr>
<tr>
<td>0070</td>
<td>90 IF(ND2.EQ.0) GO TO 115</td>
</tr>
<tr>
<td>0071</td>
<td>CALL PROD(NDO,UDO,VDO,NG0,UDP,VD0,ND2,UD2,VD2,ND3,UD3,VD3)</td>
</tr>
<tr>
<td>0072</td>
<td>100 CALL PROD(ND1,UD1,VD1,ND1,UD1,VD1,ND2,UD2,VD2)</td>
</tr>
<tr>
<td>0073</td>
<td>CALL DIVIDE(ND3,UD3,VD3,ND4,UD4,VD4,NG0,UDG,VDG)</td>
</tr>
<tr>
<td>0074</td>
<td>110 WRITE(IO2,1035) J1</td>
</tr>
<tr>
<td>0075</td>
<td>KKK=NG0+1</td>
</tr>
<tr>
<td>0076</td>
<td>NN0=NND+1</td>
</tr>
<tr>
<td>0077</td>
<td>DO 112 I=1,KKK</td>
</tr>
<tr>
<td>0078</td>
<td>JJJ=NN0-1</td>
</tr>
<tr>
<td>0079</td>
<td>112 WRITE(IO2,1040) GNAM,ENTRY(JJJ),UG(JJJ),VG(JJJ)</td>
</tr>
<tr>
<td>0080</td>
<td>CALL ZEROUSING,UG,VG,NAPP,UPAP,JPAP,JUZROS,VZROS,JAP,UPA,VPAP,ENTRY</td>
</tr>
<tr>
<td>0081</td>
<td>1,1STRT,ENDD</td>
</tr>
<tr>
<td>0082</td>
<td>IF(JJ.EQ.0) GO TO 150</td>
</tr>
<tr>
<td>0083</td>
<td>WRITE(IO2,1100)</td>
</tr>
<tr>
<td>0084</td>
<td>IF(JJAP.EQ.0) GO TO 120</td>
</tr>
<tr>
<td>0085</td>
<td>GO TO 130</td>
</tr>
<tr>
<td>0086</td>
<td>115 KKK=NG0+1</td>
</tr>
<tr>
<td>0087</td>
<td>DO 116 I=1,KKK</td>
</tr>
<tr>
<td>0088</td>
<td>UO3(I)=UDD(I)</td>
</tr>
<tr>
<td>0089</td>
<td>116 VD3(I)=VDD(I)</td>
</tr>
<tr>
<td>0090</td>
<td>NO3=NG0</td>
</tr>
<tr>
<td>0091</td>
<td>GO TO 100</td>
</tr>
<tr>
<td>0092</td>
<td>120 KKK=JAP+1</td>
</tr>
<tr>
<td>0093</td>
<td>WRITE(IO2,1085) (1,UZROS(I),VZROS(I),J1=1-KKK)</td>
</tr>
<tr>
<td>0094</td>
<td>GO TO 140</td>
</tr>
<tr>
<td>0095</td>
<td>WRITE(IO2,1190) (1,UZROS(I),VZROS(I),J1=1-JAP)</td>
</tr>
<tr>
<td>0096</td>
<td>IF(JJAP.EQ.JJ) GO TO 120</td>
</tr>
<tr>
<td>0097</td>
<td>140 IF(JJ.EQ.NG) GO TO 155</td>
</tr>
<tr>
<td>0098</td>
<td>150 WRITE(IO2,1095)</td>
</tr>
</tbody>
</table>
TABLE G.III (Continued)

0098 IF(J.EQ.0) GO TO 170
0099 I=1,J
0100 UROOT(KO+II=UZROSiII
0101 VROOTIKD+II=VZROS(I)
0102 MULTIKO+II=J1
0103 K=(J*J)+K
0104 KD=KD+J
0105 IF(K.GE.NP) GO TO 1
0106 J1=J1+1
0107 IF(ND1.LE.1) GO TO 200
0108 DO 180 I=1,ND1
0109 UDO(I)=UDIII)
0110 VDOOII=VDIII)
0111 UDD0II)=UDDOl(I
0112 VDDO(II)=VDDL(I
0113 Ji=J1l1
0114 IF(ND1.LE.1) GO TO 200
0115 KD=KD+1
0116 KD=KD+1
0117 KKK=ND2+1
0118 DO 190 I=1,KKK
0119 WRITE(OO2,30001
0120 WRITE(OO2,10351
0121 WRITE(OO2,1085)
0122 GO TO 1
0123 200 IF(ND1.LE.1) GO TO 1
0124 KD=KD+1
0125 DEMONY=UD12*VDI2*UD12*VDI2
0126 UROOTKI(-=UD21+UD21+VD12+VDI2)/DEMON
0127 VROOTKI(DI=-VDL11+VL11+VDL11+VDI2)/DEMON
0128 MULTKI=J1
0129 WRITE(IO2,10001 FASTER,=I=1,331
0130 WRITE(IO2,10351 J1
0131 KK=KD+1
0132 NND=NND+1
0133 KD=KD+1
0134 JJJ=NNN-1
0135 WRITE(IO2,1100) ONAME,ENTRY(JJJI,UDIIJJJJ,VDI(JJI)
0136 WRITE(IO2,1180)
0137 WRITE(IO2,1085) KD,UROOT(KD),VROOT(KD),J1
0138 GO TO 1
0139 1020 FORMAT(1X,10X,+HAVE REPEATED USE OF THE GREATEST COMMON DIVISOR AND
0140 1,AY,A4,5AH METHOD TO EXTRACT ROOTS AND MULTIPLICITIES OF POLYNOMIA
0141 115/14,1HPOLYNOMIAL NUMBER,12/1)
0142 1025 FORMAT(14,25H,ROOTS OF MULTIPLICITY ,12/1)
0143 1035anke FOLLOWING POLYNOMIAL, GIXI, CONTAINS ALL THE R
0144 1095 FORMAT(14,5H,ROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUN
0145 1000 FORMAT(13(12.1X,9X,1X,3(10.1X,1X),20X,2(12.1X,11)
0146 1010 FORMAT(12X,20.1X)
0147 1030 FORMAT(12X,20.1X)
0148 1040 FORMAT(12X,20.1X)
0149 1100 FORMAT(12X,20.1X)
TABLE G.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0150</td>
<td><strong>FORMAT</strong>/<em>IX,13HROOTS OF PI</em>*/X,14HMULTIPLICITIES,17X,21MINITIAL APPRXIMA**/</td>
</tr>
<tr>
<td>0151</td>
<td><strong>FORMAT</strong>/<em>IX,12,4H</em>* = .023.16,3H + .023.16,2H 1,7X,12,7X,023.16,3H + .023.16,2H 1)</td>
</tr>
<tr>
<td>0152</td>
<td><strong>FORMAT</strong>/<em>IX,4HNUMBER OF INITIAL APPRXIMATIONS GIVEN.</em>*</td>
</tr>
<tr>
<td>0153</td>
<td><strong>FORMAT</strong>/<em>IX,12MAXIMUM NUMBER OF ITERATIONS.</em>*</td>
</tr>
<tr>
<td>0154</td>
<td><strong>FORMAT</strong>/<em>IX,12TEST FOR CONVERGENCE.</em>*</td>
</tr>
<tr>
<td>0155</td>
<td><strong>FORMAT</strong>/*IX,23HNUMBER TO START SEARCH.,11X,09.2)</td>
</tr>
<tr>
<td>0156</td>
<td><strong>FORMAT</strong>/*IX,21NUMBER TO END SEARCH.,13X,09.2</td>
</tr>
<tr>
<td>0157</td>
<td><strong>FORMAT</strong>/*IX,34HTEST FOR ZERO IN SUBROUTINE GCD.,09.2</td>
</tr>
<tr>
<td>0158</td>
<td><strong>FORMAT</strong>/*IX,34HTEST FOR ZERO IN SUBROUTINE QUAD.,09.2</td>
</tr>
<tr>
<td>0159</td>
<td><strong>FORMAT</strong>/*IX,32A32A4</td>
</tr>
<tr>
<td>0160</td>
<td><strong>FORMAT</strong>/*IX,32A4</td>
</tr>
<tr>
<td>0161</td>
<td><strong>END</strong></td>
</tr>
</tbody>
</table>
TABLE G.III (Continued)

SUBROUTINE PROOCM,UF,VF,UG,VG,UM,UH,VH
C
****************.*************************************************************************
C
GIVEN POLYNOMIALS R(X) AND S(X), THIS SUBROUTINE COMPUTES THE
C COEFFICIENTS OF THE PRODUCT POLYNOMIAL T(X) = R(X).S(X).
C
*************************************************************************

DOUBLE PRECISION UM,VH,UF,VF,UG,VG
DIMENSION UM(1),VF(1),UG(1),VG(1)
MN=M+N
KKK=MN+1
10 I=1,KKK
   VH(II)=0.0
   IF(I.LE.M+L) GO TO 10
   LIMIT=M+1
   GO TO 20
20 JAVA LIMIT
   IF(J.GT.N+I) GO TO 50
   IF(J+K.EQ.I+1) GO TO 40
   GO TO 50
40 UH(I)=UH(I)+(UF(I)*UG(K)+VF(I)*VG(K))
   VHI=VHI+(VF(I)*UG(K)+UF(I)*VG(K))
   K=K-1
   10 CONTINUE
RETURN
END

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

DOUBLE PRECISION APPR,APPI,XSTART,BETA, EPS1,EPS2,EPS3
DIMENSION APPR(25),APPI(25)
COMMON EPS1,EPS2,EPS3,102,AX
IFIXSTART.EQ.O.O0
   XSTART=0.5
8ETA=0.2617994
   DO I=1,NAPP
      APPRI=XSTART*COS(BETA)
      APPI(I)=XSTART*DSIN(BETA)
      BETA=BETA*0.5235988
   XSTART=XSTART+0.5
   RETURN
END

SUBROUTINE GENAPP(appr,appl,napp,xstart)
TABLE G.III (Continued)

SUBROUTINE ALTER(XOLDR, XOLDI, NALTER, ITIME)

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

DOUBLE PRECISION XOLDR, XOLDI, ABXOLD, 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

GO TO 20
10 ABXOLD = DSQRT((XOLDR*XOLDRI*IXOLOI*XOLDII)
BETA = OATAN2(XOLDI, XOLDR)
WRITE(IO2, 10201)

NALTER = NALTER + 1
IF (FINALTER .GT. 5) RETURN

XOLDR = -XOLDR
XOLDI = -XOLDI
GO TO 50

40 BETA = BETA + 1.0471976
XOLDR = ABXOLD*DCOS(BETA)
XOLDI = ABXOLD*DSIN(BETA)
RETURN

1000 FORMAT(LX, D23.16, 3H +, D23.16, 2H I, 10X, 21HALTED APPROXIMATION)
1010 FORMAT(//, LX, 5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS OF ITER , 13, 12H ITERATIONS,//)
1020 FORMAT(//, LX, 5HINITIAL APPROXIMATION)

END
TABLE G.11 (Continued)

0001 SUBROUTINE ZEROS(INQ, UQ, VQ, NAPP, VAPP, J, UROOT, VROOT, JAP, UAP, VAP, IENTRY, XSTART, XEND)

0002 C*******************************************************************************

0003 C ** NEWTON'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPLEITIES OF A

0004 C ** POLYNOMIAL OF MAXIMUM DEGREE 25 BY COMPUTING A SEQUENCE OF APPROX-

0005 C ** MATIONS CONVERGING TO A ZERO OF THE POLYNOMIAL USING THE ITERATION

0006 C ** FORMULA

0007 C

0008 C      X(N+1) = X(N) - P(X(N))/P'(X(N)).

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

0010 0002 DOUBLE PRECISION UAPP, VAPP, UROOT, VROOT, UZRO, VZRO, UQQ, VQQ, UDUMMY, VDUM

0011 JY, UQQ, VQQ, UAP, VAP, UGQ, VQQ, UROOT, VROOTS, EPS1, EPS2, EPS3, UAPRX, VAP

0012 PDRX

0013 DOUBLE PRECISION XENON, XSTART

0014 DIMENSION UAPP(25), VAPP(25), UROOT(25), VROOT(25), UGQ(25), VQQ(25), UAPRX(25), VAPRX(25)

0015 COMMON EPS1, EPS2, EPS3, IOZ, MAX

0016 DATA GONAME, GONAME /3HQQI, 2HQQI /

0017 LOGICAL CONV

0018 J=0

0019 ITIME=0

0020 IF(NGQ.GE.3) GO TO 85

0021 GO TO 110

0022 85 KKK=NGQ+1

0023 GO 90 1=1, KKK

0024 UQQ(I)=UQQ(I)

0025 90 VQQ(I)=VQQ(I)

0026 NGQ=NGQ

0027 GO TO 120

0028 110 CALL QUADINO(UQ, VQ, J, UROOT, VROOT)

0029 JAP=0

0030 GO TO 130

0031 120 GO 200 I=1, NAPP

0032 IF(LATER=0)

0033 UAPRX=UAPP(I)

0034 VAPRX=VAPP(I)

0035 130 CALL NEWTON(UAPRX, VAPRX, NGQ, UQQ, VQQ, UZRO, VZRO, CONV)

0036 IF(CONV) GO TO 160

0037 CALL ALTER(UAPRX(I), VAPRX(I), LATER, ITIME)

0038 IF(LATER.GT.5) GO TO 200

0039 UAPRX=UAPP(I)

0040 VAPRX=VAPP(I)

0041 GO TO 130

0042 160 J=J+1

0043 UGQ(J)=UZRO

0044 VQQ(J)=VZRO

0045 170 UAPP(J)=UAPRX

0046 VAPP(J)=VAPRX

0047 CALL HOMER(UZRO, VZRO, UQQ, VQQ, UQQ, VQQ, UDUMMY, VDUM)

0048 GO 180 I=1, NGQ

0049 UQQ(I)=UQQ(I+1)

0050 180 VQQ(I)=VQQ(I+1)

0051 NGQ=NGQ-1

0052 IF(NGQ.GE.3) GO TO 200

0053 JAP=J

0054 GO TO 220
TABLE G.III (Continued)

0045 200 CONTINUE
0046 IF(J.GE.NQ) GO TO 205
0047 IF(I.XEQ.EQ.0) GO TO 205
0048 IF(I.XSTART.GT.XEND) GO TO 205
0049 NAPP=NO
0050 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0051 GO TO 120
0052 205 IF(NQQ.LE.2) GO TO 210
0053 WRITE(102,1200)
0054 KKK=NQQ+1
0055 NNN=KKK+1
0056 DO 157 L=1,KKK
0057 JJ=NNN-L
0058 157 WRITE(102,1100) QNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0059 210 IF(J.EQ.0) GO TO 310
0060 JAP=J
0061 GO TO 230
0062 220 CALL QUAD(UQQ,VQQ,J,UROOT,VROOT)
0063 230 WRITE(102,1132) (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,K)
0068 240 J=0
0069 DO 300 J=1,J
0070 CALL NEWTON(UROOT(I),VROOT(I),NO,UQ,VQ,U2Q,V2Q,CONV)
0071 IF(0.IEQ.0) GO TO 290
0072 WRITE(102,1140) UROOT(I),VROOT(I),MAX,NQ
0073 KKK=NO+1
0074 NNN=KKK+1
0075 DO 242 L=1,KKK
0076 JJ=NNN-L
0077 WRITE(102,1040) QNAME,ENTRY(JJJ),UQQ(JJJ),VQQ(JJJ)
0078 IF(J.LT.J+1) GO TO 241
0079 IF(J.EQ.JAP) GO TO 250
0080 KKK=JAP+1
0081 DO 245 J=1,KKK
0082 WRITE(102,1140) UROOT(I),VROOT(I),MAX,NQ
0083 VAPI(I1)=UAP(I1+1)
0084 245 JAP=JAP+1
0085 GO TO 240
0086 241 KKK=JAP+1
0087 DO 300 J=1,J
0088 IF(J.LT.JAP) UROOTS(JJ)=U2Q
0089 VROOTS(JJ)=V2Q
0090 GO TO 310
0091 300 CONTINUE
0092 J=J+1
0093 IF(J.EQ.0) GO TO 305
0094 DO 303 J=1,J
0095 UROOT(I)=UROOTS(I1)
0096 VROOT(I)=VROOTS(I1)
0097 GO TO 310
0098 303 GO TO 305
0099 KKK=NO+1
0100 NNN=KKK+1
0101 DO 306 L=1,NNN
0102 JJ=NNN-L
TABLE C.III (Continued)

0103 306 WRITE(102,10401) QNAME,ENTRY(JJ+I),WQ(JJJ),VQ(JJJ)
0104 110 RETURN
0105 1200 FORMAT(/X,7O10.0)
0106                    FORMAT(/X,13HROOTS OF G(X),84X,21HINITIAL APPROXIMATION/) 
0107 1133 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H * 1 + ,D23.16,2H * I)
0108 1134 FORMAT(2X,5HROOT(,I2,4H) = ,D23.16,3H + ,D23.16,2H * 1,22X#26RESULT 
0109 1140 FORMAT(/X,1X,40HNO ROOTS FOR INITIAL APPROXIMATION ROOT(,I2,4H) = 
0110 1150 FORMAT(/X,1X,40HNO ROOTS FOR THE POLYNOMIAL QIX) OF DEGREE = ,12, 
13H WITH GENERATED INITIAL APPROXIMATIONS//)
0111 1040 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H * I)
0112 1100 FORMAT(2X,A3,A2,4H) = ,D23.16,3H + ,D23.16,2H * I)
0113 END
TABLE G.III (Continued)

0001 SUBROUTINE GCD(NUR,VR,M,US,VS,MI,USSS,VSST)
0002 *******************************************************************************************************************************
0003 C
0004 * GCD Computes the Greatest Common Divisor of P(X) and DPIX
0005 C
0006 ※ POLYNOMIALS P(X) AND DPIX WHERE DEG. DPIX IS LESS THAN DEG. P(X)
0007 C
0008 ※ P(X)
0009 C
0010 C
0011 *******************************************************************************************************************************
0012 DOUBLE PRECISION USSSSS,VSSTSSS
0013 DOUBLE PRECISION UR,VR,US,VS,USS,URR,VRR,US,VS,USST,VSST,URR,VR,US,VS,USST,VSST
0014 DIMENSION UR(26),VR(26),US(26),VS(26),USS(26),VSS(26),UR(26),VR(26),US(26),VS(26)
0015 COMMON EPSLON,EPS2,EPS3,EPSLON,EP,EP
0016 NI=M
0017 M1=MI
0018 KKK=N+1
0019 DO 20 I=1,KKK
0020 URRII)=UR(II
0021 20 VRR(I)=VR(I)
0022 KKK=M+I
0023 DO 35 I=1,KKK
0024 USSI)=US(II
0025 VSSI)=VS(I)
0026 35 DO 40 I=I,MI
0027 BBB=USS(MI+1)*USS(MI+1)+VSS(MI+1)*VSS(MI+1)
0028 UD=URR(N1+1)*USS(MI+1)+VAR(N1+1)*VSS(MI+1)/BBB
0029 VDO=USS(MI+1)*VRR(N1+1)-URR(N1+1)*VSS(N1+1)/BBB
0030 KKK=NI+I
0031 DO 60 I=KKKNI
0032 UT(I)=URR(I)
0033 60 VTR(I)=VRR(I)
0034 70 DO 80 I=I,NI
0035 BBB=DSQRT(UTINI+1-I)*UTINI+1-I+VT(NI+1-1)*VT(NI+1-1)
0036 IF(BBB.GT.EPSLON) GO TO 100
0037 90 CONTINUE
0038 DO 95 I=MI
0039 BBB=USS(MI+1)*USS(MI+1)+VSS(MI+1)*VSS(MI+1)
0040 USSSS=(USS(I)*USS(MI+1)+VSS(I)*VSS(MI+1))/BBB
0041 VSSSSS=IVSS(I)*USS(MI+1)-USS(I)*VSS(MI+1)/BBB
0042 USS(MI+1)=USSSS
0043 VSS(MI+1)=VSSSSS
0044 DO 95 I=1,MI
0045 UT(I)=URR(I)
0046 VTR(I)=VRR(I)
0047 95 GO TO 30
0048 K=NI-1
0049 IF(K.EQ.0) GO TO 70
0050 DO 94 I=MI
0051 KKK=K
0052 URR(I)=VT(I)
0053 IF(K.EQ.0) GO TO 70
0054 100 CONTINUE
0055 GO TO 30
0056 130 VRR(J)=VT(I)
0057 NI=K
0058 GO TO 30
TABLE G.III (Continued)

0050       140  KKK=K+1
0051       150  DO 150  J=1, KKK
0052       150  URRI(J)=USSI(J)
0053       150  VRR(J)=VSSI(J)
0054       150  USSI(J)=UTI(J)
0055       150  VSSI(J)=VTI(J)
0056       150  KKK=K+2
0057       150  NNN=MJ+1
0058       160  DO 160  J=KKK, NNN
0059       160  URRI(J)=USSI(J)
0060       160  VRR(J)=VSSI(J)
0061       160  MJ=MJ
0062       160  M1=M
0063       160  GO TO 30
0064       170  VSSI(1)=1.0
0065       170  VSSI(1)=0.0
0066       170  MJ=0
0067       200  RETURN
0068       END
TABLE G.11 (Continued)

0001 SUBROUTINE NEWTONIUXVX,N,UP,VP,UXO,VXO,CONV)

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

0002 DOUBLE PRECISION UX, VX, UP, VP, UXO, VXO, UB, VB, UDPXO, VDPXO, UPOXO, VPXO, UDIFF, VDIFF, EPSLI, EPSLON, EPS3, AAA, BBB

0003 DOUBLE PRECISION A

0004 DIMENSION U(26), V(26), UB(26), VB(26)

0005 COMMON EPSLI, EPSLON, EPS3, IO2, MAX

0006 LOGICAL CONV

0007 UXO = UX
    VXO = VX

0010 DO 10 I = 1, MAX

0011 CALL HORNERIUXOVXO.N, UP, VP, UB, VB, UDPXO, VDPXO, UPOXO, VPXO

0012 UPXO = UB(1)
    VPXO = VB(1)

0013 UDIFF = DSQRT(UDPXO*UDPXO+VDPXO*VDPXO)

0014 IF(UDIFF .NE. 0.0) GO TO 5

0015 A = DSQRT(UDPXO*UDPXO+VDPXO*VDPXO)

0016 IF(A .LE. EPSLON) GO TO 20

0017 BBB = UDIFF

0018 GO TO 15

0019 5 BBB = UDIFF

0020 UDIFF = (VPXO*UPXO-UDPXO*VPXO)/ BBB

0021 VDIFF = (UDPXO*UPXO-VPXO*VPXO)/ BBB

0022 UXO = URO - UDIFF

0023 VXO = VRO - VDIFF

0024 AAA = DSQRT(UDIFF*UDIFF+VDIFF*VDIFF)

0025 BBB = DSQRT(URO*UXO+VRO*VXO)

0026 IF(AAA .LE. EPSLON) GO TO 20

0027 IF(UDIFF .LE. EPSLON) GO TO 20

0028 10 CONTINUE

0029 15 CONV = .FALSE.

0030 RETURN

0031 20 CONV = .TRUE.

0032 RETURN

0033 END
TABLE G.III (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>SUBROUTINE DIVIDE(U,P,V,UP,VP,U,Q,VQ)</td>
</tr>
<tr>
<td>0002</td>
<td>DOUBLE PRECISION UP,VP,U,Q,V,UTERM,VTERM,UOQ,VQ</td>
</tr>
<tr>
<td>0003</td>
<td>DIMENSION U(26),VP(26),UD(26),VD(26),UQ(26),VD(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=N-J</td>
</tr>
<tr>
<td>0016</td>
<td>DO 40 M=MNN+M</td>
</tr>
<tr>
<td>0017</td>
<td>IF(KK.GT.1) GO TO 10</td>
</tr>
<tr>
<td>0018</td>
<td>IF(KK.GT.1) GO TO 10</td>
</tr>
<tr>
<td>0019</td>
<td>IF(KK.GT.1) GO TO 40</td>
</tr>
<tr>
<td>0020</td>
<td>UTERM=UTERM-(UQ(KK)*UD(M+1)+VQ(KK)*VD(M+1))/UDUMMY</td>
</tr>
<tr>
<td>0021</td>
<td>VTERM=VTERM-(UQ(KK)*VD(M+1)+VQ(KK)*UD(M+1))/UDUMMY</td>
</tr>
<tr>
<td>0022</td>
<td>KK=KK+1</td>
</tr>
<tr>
<td>0023</td>
<td>40 UDUMMY=UD(M+1)*UD(M+1)+VD(M+1)*VD(M+1)</td>
</tr>
<tr>
<td>0024</td>
<td>45 UQ(KK+1)=UTERM<em>UD(M+1)+VTERM</em>VD(M+1)/UDUMMY</td>
</tr>
<tr>
<td>0025</td>
<td>VQ(KK+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>

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

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 POINT X AND ITS DERIVATIVE AT X. SYNTHETIC
C * DIVISION IS USED TO DEFLATE THE POLYNOMIAL BY DIVIDING OUT THE FACTOR (X - D).                        
C ****************************************************************************************************************
C
DO DIMENSION UP(26), VP(26), UB(26), VB(26)
DO UBIN = IUX * UB(N+I) - VX * VB(N+I) + UP(N)
DO VB(N+1) = IUX * VB(N+1) + VX * UB(N+1) + VP(N)
DO UC = UB(N+1)
DO VC = VB(N+1)
DO KKK = N - 1
DO 10 I = 1, KKK
DO 10 UB(KKK+1-I) = IUX * UB(KKK+2-I) - VX * VB(KKK+2-I) + UP(KKK+1-I)
DO 10 VB(KKK+1-I) = IUX * VB(KKK+2-I) + VX * UB(KKK+2-I) + VP(KKK+1-I)
DO 10 UDUMMY = IUX * UC - VX * VC
DO 10 VC = UDUMMY + VB(KKK+2-I)
10 RETURN
END
TABLE G.III (Continued)

0001 SUBROUTINE QUAD(NUA,VA,J,VRDIT,VRDIT)
0002 ************************************************************************
0003 * SUBROUTINE QUAD SOLVES DIRECTLY FOR THE ZEROS AND THEIR
0004 * MULTIPLE ROOTS OF EITHER A QUADRATIC POLYNOMIAL OR A
0005 * LINEAR FACTOR. SOLUTION OF THE
0006 * QUADRATIC IS DONE USING THE QUADRATIC FORMULA.
0007 ************************************************************************
0008 DOUBLE PRECISION EPI, EPS2, EPSLON, VRDIT, VRDIT, UA, UDISC, VDISC, UD
0009 DIMENSION UROOT(25), VROOT(25), UAI(26), VAI(26)
0010 COMMON EPSI, EPS2, EPSLON, IO2, MAX
0011 IF(IN.GT.1) GO TO 10
0012 J=J+1
0013 BBB=UA(2)*UA(2)+VA(I)*VA(I)
0014 UROOT(J)=-UA(I)*UA(2)+VA(2))/BBB
0015 VROOT(J)=-VA(2)/VA(I)*UA(I)*UA(2))/BBB
0016 IF(J.LE.10) GO TO 20
0017 UDISC=(UA(2)*UA(2)+VA(2)*VA(2)-4.0*(UA(3)*UA(I)+VA(3)*VA(I))
0018 VDISC=2.0*UA(3)
0019 UD=2.0*UA(I)
0020 VD=2.0*VA(I)
0021 DDD=SQRT(UDISC*UDISC+VDISC*VDISC)
0022 IF(DDD.LT.EPSLON) GO TO 20
0023 CALL COMSQT(UDISC, VDISC, UTEMPl, VTEMP)
0024 BBB=UD*UD+VD*VD
0025 UROOT(J+1)=((-UA(I)+UTEMP)*UD-VA(2)+VTEMP)*VD)/BBB
0026 VROOT(J+1)=(-VA(2)+VTEMP*UD-(-UA(I)+UTEMP)*VD)/BBB
0027 UROOT(J+2)=((-UA(I)-UTEMP)*UD+(-VA(2)-VTEMP)*VD)/BBB
0028 VROOT(J+2)=((-VA(2)-VTEMP)*UD+(-UA(I)-UTEMP)*VD)/BBB
0029 J=J+2
0030 GO TO 10
0031 20 J=J+1
0032 BBB=UD*UD+VD*VD
0033 UROOT(J+1)=((-UA(I)+UD-VA(2)+VOD*VOD)
0034 VROOT(J+1)=(-VA(2)+VOD*UA(I)+VOD*VOD)
0035 WRITE(IO2,1000) UROOT(J), VROOT(J)
0036 1000 FORMAT(/I1,10HQUAD FOUND ,1H23.16,3H + ,1H23.16,2H I,2H TO BE A M
0037 MULTIPLE ROOT//)
0038 RETURN
0039 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,AAA
0003 DIMENSION UP(26),VP(26),U(26),VA(26)
0004 KKK=1
0005 DO 10 I=2,KKK
0006 AAA=I-1
0007 U(I)=AAA*UP(I)
0008 VA(I)=AAA*VP(I)
0009 N=1
0010 RETURN
0011 END

0011 SUBROUTINE CONSQT(UX,VX,UY,VY)
C +%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C +% THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER.
C +%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

0012 DOUBLE PRECISION UX,VX,UY,VY,DUMMY,R,AAA,BBB
0013 R=DSQRT(UX*UX+VX*VX)
0014 AAA=DSQRT(R)
0015 BBB=DSQRT((R-UX)/2.0)
0016 IF(UX) 10,20,30
0017 10 UY=AAA
0018 VY=-I.0*BBB
0019 GO TO 100
0020 20 UY=AAA
0021 VY=BBB
0022 GO TO 100
0023 30 UY=0.0
0024 VY=DSQRT(DUMMY)
0025 GO TO 100
0026 40 DUMMY=DSQRT(UX)
0027 UY=0.0
0028 VY=DSQRT(DUMMY)
0029 GO TO 100
0030 50 UY=0.0
0031 VY=0.0
0032 GO TO 100
0033 60 DUMMY=DSQRT(UX)
0034 UY=DSQRT(DUMMY)
0035 VY=0.0
0036 100 RETURN
0037 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.1

**Program Changes Necessary to Solve Polynomials of Degree Greater Than 25 by the Repeated G.C.D. - Muller's Method**

#### Main Program

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

- UAPP(N,3), VAPP(N,3)
- URAPP(N,3), URAPP(N,3)
- UP(N+1), VP(N+1)
- MULT(N)
- UDDO(N+1), VDDO(N+1)
- UD1(N+1), VD1(N+1)
- UDD1(N+1), VDD1(N+1)
- UD2(N+1), VD2(N+1)
- UG(N+1), VG(N+1)
- UD3(2N+1), VD3(2N+1)
- UD4(2N+1), VD4(2N+1)
- UAP(N+1), VAP(N+1)
- UZROS(N), VZROS(N)
- UROOT(N), VROOT(N)
- UD0(N+1), VDO(N+1)
- ENTRY(N+1)

**Subroutines PROD, QUAD**

See corresponding subroutine in Table G.1.

**Subroutines DERIV, GCD, and DIVIDE**

See corresponding subroutine in Table E.1.

**Subroutines MULLER, GENAPP, BETTER and HORNER**

See corresponding subroutine in Table F.1.

---

2. **Input Data for Repeated G.C.D. - Muller's Method**

The input data to the repeated G.C.D. - Muller's method is the same as for the repeated G.C.D. - Newton's method as described in Appendix G, § 2.
3. Variables Used in Repeated G.C.D. - Muller's Method

The variables used in this program are referenced in Table H.II. The notation and symbols used in the referenced tables are described in Appendix E, § 3.

TABLE H.II
VARIABLES USED IN REPEATED G.C.D. - MULLER'S METHOD

Main Program and Subroutine PROD

See Table G.II.

Subroutines QUAD, DERIV, GCD, DIVIDE, and COMSQRT

See corresponding subroutine in Table E.VI.

Subroutines CALC, MULLER, GENAPP, ALTER, BETTER, TEST, and HORNER.

See corresponding subroutine in Table F.II.

4. Description of Program Output

The output for this program is the same as that for repeated G.C.D. - Newton's method as described in Appendix G, § 4. Only one initial approximation, $X_0$, (not three) is printed. The other two required by Muller's method are $0.9X_0$ and $1.1X_0$. The message "SOLVED BY DIRECT METHOD" means that the corresponding root was obtained by Subroutine QUAD.
5. Informative Messages and Error Messages

Descriptions of the informative messages and error messages printed by this program can be found either in Appendix E, § 5, Appendix F, § 5, or Appendix G, § 5.
Figure H.1. Flow Charts for Repeated G.C.D.-Muller's Method
Figure H.1: (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
Figure H.1. (Continued)
TABLE H. III

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

C

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

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

* THIS METHOD REPEATEDLY FINDS THE GREATEST COMMON DIVISOR OF TWO
C

* POLYNOMIALS IN ORDER TO EXTRACT THE ZEROS IN GROUPS ACCORDING TO
C

* MULTIPLICITY USING NEWTON'S METHOD. ALL ZEROS OF MULTIPLICITY 1
C

* ARE EXTRACTED FOLLOWED BY THOSE OF MULTIPLICITY 2, ETC.
C

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

0001  DOUBLE PRECISION EPS1, EPS2, EPS3, UP, VP, UAPP, VAPP, UD0, VOO, UDDO, VDDO,
        UD1, VD1, UD2, VD2, UD01, UD00, UD12, UDD1, VDD1, UD26, VDD26, UD126, VD126

0002  DOUBLE PRECISION XDEND

0003  DOUBLE PRECISION UAPP, VRAPP

0004  DOUBLE PRECISION EPS4

0005  DIMENSION UAPP(25,3), UAPP(25,3), VRAPP(25,3), VRAPP(25,3)

0006  DIMENSION UD0(26), UD0(26), UD0(26), UDD0(26), UDD0(26), UD1(26), UD1(26)

0007  1, UDD1(26), UD1(26), UD1(26), UD2(26), UD2(26), UD3(25), UD3(25),

0008  UDD3(25), UDD3(25), UDD3(25), UDD3(25), UDD3(25), UDD3(25), UDD3(25),

0009  COMMON EPS1, EPS2, EPS3, EPS4, J02, MAX

0010  DATA NAME, GNAME, 2HPI, 2HIP, BLNAME, SHDRT

0011  DATA ASTER\ thoại/****/

0012  DATA ENTRY/1H1, J01, 1H2, 1H3, 1H4, 1H5, 1H6, 1H7, 1H8, 1H9, 2H10,

0013  2H11, 2H12, 2H13, 2H14, 2H15, 2H16, 2H17, 2H18, 2H19, 2H20, 2H21,

0014  2H22, 2H23, 2H24, 2H25, 2H26/

0015  DATA NAME(11), NAME(12), NAME(13), NAME(14), NAME(15)

0016  101 = 5

0017  102 = 6

0018  103 = 7

0019  104 = 8

0020  105 = 9

0021  106 = 10

0022  107 = 11

0023  108 = 12

0024  109 = 13

0025  1203 = 14

0026  1204 = 15

0027  1205 = 16

0028  1206 = 17

0029  1207 = 18

0030  1208 = 19

0031  1209 = 20

0032  1210 = 21

0033  1211 = 22

0034  1212 = 23

0035  1213 = 24

0036  1214 = 25

0037  1215 = 26

0038  1216 = 27

0039  1217 = 28

0040  1218 = 29

0041  1219 = 30
TABLE H.III (Continued)

0041  8 WRITE(102,1040) PNAME,ENTRY(JJJJ),UP(JJJJ),VP(JJJJ)
0042  K=0
0043  KD=0
0044  J=1
0045  KKK=NP+1
0046  DO 10 I=1,KKK
0047  UDD(I)=UP(I)
0048  10 VDD(I)=VP(I)
0049  NDD=NP
0050  CALL DERIV(NDD,UDD,VDD,NDD,UDD,VDD1)
0051  CALL GCD(NDD,UDD,VDD,NDD,UDD,VDD,NDD,UDD1,VDD1)
0052  20 WRITE(102,3000) (ASTER,I=1,133)
0053  IF(NDD.LE.1) GO TO 30
0054  GO TO 40
0055  30 UD2(I)=1.0
0056  VD2(I)=0.0
0057  N2=0
0058  GO TO 50
0059  40 CALL DERIV(NDD1,UDD1,VDD1,NDD1,UDD1,VDD1)
0060  CALL GCD(NDD1,UDD1,VDD1,NDD1,UDD1,VDD1,NDD1,UDD1,VDD1)
0061  50 IF(NDD+NDD1.LE.2+ND) GO TO 60
0062  GO TO 70
0063  60 WRITE(102,1025) J1
0064  GO TO 110
0065  70 IF(NDD.EQ.0) GO TO 80
0066  GO TO 90
0067  80 KKK=NDD+1
0068  DO 85 I=1,KKK
0069  UG(I)=UD(I)
0070  VG(I)=VDD(I)
0071  NG=NDD
0072  GO TO 110
0073  90 IF(NDD.EQ.0) GO TO 115
0074  CALL PROD(NDD,UDD,VDD,NDD1,UDD1,VDD1)
0075  100 CALL PROD(NDD1,UDD1,VDD1,NDD1,UDD1,VDD1)
0076  CALL DIVIDE(NDD1,UDD1,VDD1,NDD1,UDD1,VDD1)
0077  110 WRITE(102,1035) J1
0078  KKK=NG+1
0079  NDD=NDD+1
0080  DO 112 I=1,KKK
0081  JJJ=NNN-I
0082  112 WRITE(102,1040) GNAME,ENTRY(JJJJ),UG(JJJJ),VG(JJJJ)
0083  KKK=NG+1
0084  NO 113 I=1,KKK
0085  UAP(I)=UG(KKK+1-I)
0086  113 VAP(I)=VG(KKK+1-I)
0087  CALL MULLER(NGUAP,VAP,NAPPVAP,NAPPVAP,J,START,OXST,XEND,
0088                             NDNPOLY,URAPP,VRAPP)
0089  IF(I.EQ.0) GO TO 150
0090  WRITE(102,1180)
0091  IF(JAP.EQ.0) GO TO 120
0092  GO TO 130
0093  119 KKK=NDD+1
0094  DO 116 I=1,KKK
0095  UDD(I)=UDD(I)
0096  116 VDD(I)=VDD(I)
0097  NDD=NDD
0098  GO TO 100
TABLE H.III (Continued)

```
0078 120  KKK=JAP+1
0079 WRITE(102,1085) (I,UZROS(I),VZROS(I),J),I=KKK,J
0100 GO TO 140
0101 130 DO 135 I=1,JAP
0102 135 WRITE(102,11900) (I,UZROS(I),VZROS(I),J),I=KKK,J
0103 IF(IAP.LT.J) GO TO 120
0104 140 IF(I.EQ.NG) GO TO 155
0105 150 WRITE(102,1095)
0106 IF(I.EQ.0) GO TO 170
0107 155 DO 160 I=1,J
0108 UROOT(KO+1)=UZROS(I)
0109 VROOT(KO+1)=VZROS(I)
0110 160 VROOT(KO+1)=VROOT(I)
0111 160 KD=KD+1
0112 IF(K.GE.NP) GO TO 1
0113 JD=JD+1
0114 IF(NDE.EQ.0) GO TO 1
0115 ND1=ND1+1
0116 UD1(I)+=UD1(I)
0117 VD1(I)=VD1(I)
0118 170 IF(J.GT.NJ) GO TO 200
0119 DO 180 I=1,ND1
0120 UDDO(I)+=UDDO(I)
0121 VDDO(I)+=VDDO(I)
0122 ND2=ND2+1
0123 KKK=ND2+1
0124 ND3=ND3+1
0125 ND4=ND4+1
0126 ND5=ND5+1
0127 170 IF(J.GT.NJ) GO TO 200
0128 NDE=ND1
0129 DO 200 I=1,NDE
0130 IF(NDE.EQ.0) GO TO 1
0131 ND1=ND1+1
0132 KD=KD+1
0133 DENOM=UD1(1)+UD1(2)*VD1(2)+VD1(1)
0134 UROOT(KD)=(-UD1(1)*UD1(2)-VD1(1)*VD1(2)+UD1(1))/DENOM
0135 VROOT(KD)=(-UD1(1)*UD1(2)+UD1(1)*VD1(2)+VD1(1))/DENOM
0136 MULT(KD)=J
0137 WRITE(102,3000) (ASTER(I=1,331
0138 WRITE(102,1085) KD,UROOT(KD),VROOT(KD),J
0139 GO TO 1
0140 1020 FORMAT(2X,5HROOT(,12,4H)=,D23.16,3H
0141 1025 FORMAT(///IX,25HNO ROOTS OF MULTICLITY ,12//)
0142 1035 FORMAT(///IX,5HTHE FOLLOWING POLYNOMIAL, GEX, CONTAINS ALL THE R
0143 1035 FORMAT(///IX,5H10OFS OF PIX WHICH HAVE MULTICLITY ,12//)
0144 1045 FORMAT(///IX,5HNO ROOTS OF IMMUNITY, 12//)
0145 GO TO 1
0146 GO TO 1
0147 1050 FORMAT(///IX,5HNOT ALL ROOTS OF THE ABOVE POLYNOMIAL, G, WERE FOUN
```
TABLE H.III (Continued)

1000 FORMAT(3(12,1X,8X,13,1X,3(06.0,1X),20X,2(07.0,1X)),11)
1010 FORMAT(2D30.0)
1015 FORMAT(2D30.0)
1020 FORMAT(IX,22HTHE DEGREE OF P(X) IS ,12.2H, THE COEFFICIENTS ARE//)
1030 FORMAT(2X9A2,A2,4H)
1035 FORMAT(2X.5HROOTI,12,4H)
1040 FORMAT(I2X,4THNUMBER OF INITIAL APPROXIMATIONS GIVEN. 9121)
1045 FORMAT(I2X#,12X#)
1050 FORMAT(I2X,21HTEST FOR CONVERGENCE.,13X##)
1055 FORMAT(I2X,34HTEST FOR ZERO IN SUBROUTINE GCD. ,09.2)
1060 FORMAT(I2X,34HTEST FOR ZERO IN SUBROUTINE QUAD. ,09.2)
1065 FORMAT(///1XA3,32A4)
1070 END
TABLE H. III (Continued)

0001

SUBROUTINE PROD(M,UF,VF,N,UG,VG,MN,UM,VH)

C ***************************************************************
C * GIVEN POLYNOMIALS P(X) AND S(X), THIS SUBROUTINE COMPUTES THE
C * COEFFICIENTS OF THE PRODUCT POLYNOMIAL T(X) = P(X).S(X).
C ***************************************************************

0002

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

0003

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

0004

MN=M+N

0005

KKK=MN

0006

DO 100 I=1,KKK

0007

K=I

0008

UH(I)=0.O

0009

VH(I)=0.O

0010

IF(I.LE.M+1) GO TO 10

0011

LIMIT=N+1

0012

GO TO 20

0013

10 LIMIT=I

0014

20 DO 50 J=1,LIMIT

0015

IF(K.GT.N+1) GO TO 50

0016

IF(J.GT.N+1) GO TO 40

0017

GO TO 50

0018

40 UH(I)=UH(I)+UF(J)*UG(K)-VF(J)*VG(K)

0019

VH(I)=VH(I)+VF(J)*UG(K)+UF(J)*VG(K)

0020

50 K=K-1

0021

100 CONTINUE

0022

RETURN

0023

END
SUBROUTINE QUAD(UA, VA, J, UROOT, VROOT)
  DOUBLE PRECISION EPS, EPS2, EPSLON, UROOT, VROOT, UA, VA, UDISC, VDISC, UD
  DOUBLE PRECISION EPS4
  DIMENSION UROOT(25), VROOT(25), UA(26), VA(26)
  COMMON EPS, EPS2, EPSLON, EPS4, IO2, MAX
  IF (N.GT.1) GO TO 10
  J = J + 1
  BBB = UA(1) * UA(2) + VA(1) * VA(2)
  UROOT(J) = -(UA(1) + UA(2)) / BBB
  VROOT(J) = -(VA(1) + VA(2)) / BBB
  GO TO 100
  UDISC = UA(1) * UA(2) - VA(1) * VA(2) - (4.0 * (UA(3) * UA(1) - VA(3) * VA(1))
  VDISC = (2.0 * UA(2) * VA(2) - 4.0 * (UA(3) * VA(1) + VA(3) * UA(1)))
  UD = 2.0 * UA(3)
  VD = 2.0 * VA(3)
  BBB = UD * UD + VD * VD
  UROOT(J) = (-UD + UTEMP) / BBB
  VROOT(J) = (-VD + VTEMP) / BBB
  WRITE(IO2, 1000) UROOT(J), VROOT(J)
  1000 FORMAT(///LX, IIHQUAD FOUND, 023.16, 2H I22H TO BE A MUL-
  TABLE H.III. (Continued)
  100 RETURN
END
TABLE H.III (Continued)

0001 SUBROUTINE DERIV(IN,UP,VP,M,UA,VA)
0002 C **********************************************************************
0003 C * GIVEN A POLYNOMIAL P(X), SUBROUTINE DERIV COMPUTES THE COEFFICIENTS OF 
0004 C * ITS DERIVATIVE P'(X).
0005 C **********************************************************************
0006 C
0007 DOUBLE PRECISION UP,VP,UA,VA,AAA
0008 DIMENSION UP(26),VP(26),UA(26),VA(26)
0009 KKK=N+1
0010 DO 10 I=2,KKK
0011 AAA=I-1
0012 VA(I-1)=AAA*UP(I)
0013 UA(I-1)=AAA*VP(I)
0014 M=N-1
0015 RETURN
0016 END
**TABLE H.III (Continued)**

```plaintext
0001 SUBROUTINE GCDNUR,VR,MUS,MIoUSStVSSI

C ****
C * GIVEN POLYNOMIALS PIX) AND DPIX) WHERE DEG. DPIX) IS LESS THAN DEG. C
C * PIX), SUBROUTINE GCD COMPUTES THE GREATEST COMMON DIVISOR OF PIX) AND
C *
C ****

C 0002 DOUBLE PRECISION U1SSSSS,JSSSSS


0004 DIMENSION U(26),VR(26),US(26),VS(26),URR(26),VR(26)


0006 M1=N

0007 MI=M

0008 KKK=N+1

0009 DO 20 I=1,KKK

0010 URR(I)=UR(I)

0011 VRR(I)=VR(I)

0012 KKK=M+1

0013 DO 25 I=1,KKK

0014 USSII)=US(I)

0015 VSS(I)=VS(I)

0016 BBB=USS(M+1)*USS(M+1)+VSS(M+1)*VSS(M+1)

0017 UD=(URR(NI1)*VRR(NI1)+VRR(NI1)*URR(NI1))/BBB

0018 VD=(URR(NI1)*URR(NI1)+VRR(NI1)*VRR(NI1))/BBB

0019 KKK=N1+M1

0020 DO 40 I=KKK,N1

0021 UT(I)=URR(I)-UD*USS(NI1-MI)-VD*VSS(NI1-MI)

0022 40 VT(I)=VRR(I)-UD*VSS(NI1-MI)+VD*USS(NI1-MI)

0023 IF(M1.EQ.N1) GO TO 70

0024 KKK=N1-M1

0025 DO 60 I=1,KKK

0026 UT(I)=URR(I)

0027 60 VT(I)=VRR(I)

0028 DO 70 I=1,N1

0029 BBB=USS(T(N1)-N1+M1)*VSS(T(N1)-N1+M1)

0030 IF(BBB.GT.EPSLON) GO TO 100

0031 CONTINUE

0032 DO 90 I=1,M1

0033 BBB=USS(M1+1)*USS(M1+1)+VSS(M1+1)*VSS(M1+1)

0034 U1SSSSS=USS(I1)*USS(M1+1)+VSS(I1)*VSS(M1+1)/BBB

0035 V1SSSSS=VSS(I1)*VSS(M1+1)+USS(I1)*USS(M1+1)/BBB

0036 V1SSSSS=USSSSS

0037 95 V1SSSSS=U1SSSSS

0038 VS(M1+1)=0

0039 VS(M1+1)=0

0040 GO TO 200

0041 100 K=N1-I

0042 IF(K.EQ.0) GO TO 170

0043 IF(K.LT.M1) GO TO 140

0044 KKK=K1

0045 DO 140 J=1,KKK

0046 URR(J)=UT(J)

0047 140 VT(R)=VT(J)

0048 NI=K

0049 GO TO 30
```
TABLE H. III (Continued)

0050  140  KKK+K+1
0051  DO 150  J=1,KKK
0052    URR(J)=USS(J)
0053    VRR(J)=VSS(J)
0054    USS(J)=UTJ(J)
0055  150  VSS(J)=VT(J)
0056    KKK=K+2
0057    NNN=M+1
0058  DD 160  J=KKK,NNN
0059    URR(J)=USS(J)
0060  160  VRR(J)=VSS(J)
0061    MI=MI
0062    MI=K
0063  GO TO 30
0064  170  USS(J)=1.0
0065    VSS(J)=0.0
0066    MI=0
0067  200  RETURN
0068  END
SUBROUTINE DIVIDE(NUP,VP,UD,VD,UQ,VQ)

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

DOUBLE PRECISION UP,VP,UD,VD,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)=UQ(N+1)*UD(M+1)+VP(N+1)*VO(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
50 I=,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
45 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
SUBROUTINE CONSQTIUXVXUY, VY)
C **********************************************************************
C * THIS SUBROUTINE COMPUTES THE SQUARE ROOT OF A COMPLEX NUMBER. *
C **********************************************************************
C DOUBLE PRECISION UX, VX, UY, VY, DUMMY, R, A, B, B
C
0002 R = DSQRT(UX*UX + VX*VX)
0003 AAA = DSQRT((R + UX)/2.0)
0004 BBB = DSQRT((R - UX)/2.0)
0005 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(UY)
0022 UY = DSQRT(DUMMY)
0023 VY = 0.0
0024 100 RETURN
0025 END
TABLE II (Continued)

0001  SUBROUTINE CALC(UX1, VX1, UX2, VX2, UX3, VX3, UX4, VX4, UQ4, VQ4, UH3, VH3)

C   **************************************************************************
C   * GIVEN THREE APPROXIMATIONS XIN-2), XIN-1, AND XIN1, SUBROUTINE CALC
C   * APPROXimates THE POLYNOMIAL BY A QUADRATIC AND SOLVES FOR THE ZERO OF
C   * THE QUADRATIC CLOSEST TO XIN1. THIS ZERO IS THE NEW APPROXIMATION
C   **
C   **************************************************************************

0002  DOUBLE PRECISION ARG1, ARG2
0003  DOUBLE PRECISION UX1, VX1, UX2, VX2, UX3, VX3, UX4, VX4, UQ4, VQ4, UH3, VH3

0004  COMMON EPS1, EPS2, EPS3, EPS4, EPS5, EPS6, EPS7, MAX

0005  UH3=UX3-UX2
0006  VH3=VX3-VX2
0007  UH2=UX2-UX1
0008  VH2=VX2-VX1
0009  BBB=UH2*UH2+VH2*VH2
0010  UQ3=(UH3*UH2-UH3*VH2)/BBB
0011  VQ3=(VH3*UH2-VH3*VH2)/BBB
0012  UDD=(UQ3*UH3-VQ3*VH3)/BBB
0013  VDD=(VQ3*UH3+UQ3*VH3)/BBB
0014  UAAA=2.0*UQ3
0015  VAAA=2.0*VQ3
0016  UUBB=UQ3*UQ3-VQ3*VQ3
0017  VUBB=UQ3*VQ3+VQ3*UQ3
0018  UCCC=UQ3*UQ3-VQ3*VQ3
0019  VCCC=VQ3*UQ3+UQ3*VQ3
0020  UH=((UAAA*UX3-VAAA*VX3)-(UDBB*UX2-VUDBB*VX2))/(UCCC*UX1-VCCC*VX1)
0021  VB=(VAAA*UX3+UAAA*VX3)-(VUBB*UX2+VUDBB*VX2)/(VCCC*UX1-VCCC*VX1)
0022  UD=I.0+UC
0023  VD=I.0+VC
0024  UDISC=(UD*UD-VD*VD+4.0*(UD*UC-VD*VC))
0025  VDISC=(2.0*(VD*UD))-(4.0*(VD*UC+UD*VC))
0026  AAA=DSQRT(UDISC*UDISC+VDISC*VDISC)
0027  IF(AAA.EQ.0.0) GO TO 5
0028  GO TO 7
0029  IF(AAA.EQ.0.0) GO TO 5
0030  GO TO 7
0031  UTENT=DATAN2(VD,UDISC,UDISC)
0032  RADCIG=DSQRT(AAA)
0033  IF(VTEST.RD=UTEST) GO TO 9
0034  UTENT=DATAN2(VD,UDISC,UDISC)
0035  RADCIG=DSQRT(AAA)
0036  IF(VTEST.RD=UTEST) GO TO 9

0037  GO TO 5
0038  GO TO 9
0039  UTENT=DATAN2(VD,UDISC,UDISC)
0040  RADCIG=DSQRT(AAA)
0041  IF(VTEST.RD=UTEST) GO TO 9
0042  UTENT=DATAN2(VD,UDISC,UDISC)
0043  RADCIG=DSQRT(AAA)
0044  IF(VTEST.RD=UTEST) GO TO 9
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  IFAAA.LT.BBB GO TO 10
0050  IFAAA.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+IVH3*UQ4+UH3*VQ4
0064  RETURN
0065  60 UQ4=1.0
0066  VQ4=0.0
0067  GO TO 50
0068  END
### TABLE H.11 (Continued)

```plaintext
0001 SUBROUTINE MULLER(NP, UA, VA, NAPP, UAPP, VAPP, NROOT, UROOT, VROOT, IROOT, XSTART, XEND, NORDER, UAPP, VAPP)

******************************************************************************
C ** MULLER'S METHOD EXTRACTS THE ZEROS AND THEIR MULTIPlicITIES OF A
C ** POLYNOMIAL OF MAXIMUM DEGREE 25, THROUGH THREE GIVEN POINTS THE
C ** POLYNOMIAL IS APPROXIMATED BY A QUADRATIC. THE ZERO OF THE
C ** QUADRATIC IS CLOSEST TO THE OLD APPROXIMATION IS TAKEN AS THE NEW APPROXIMATION.
C ** IN THIS MANNER A SEQUENCE IS OBTAINED CONVERGING TO A ZERO.
******************************************************************************

0002 DOUBLE PRECISION UPX3,VPX3,UPX2,VPX2,XROOT,YROOT,U1,X1,UAPP, VAPP
1,X2,VX2,UX2,VWORX,UX3,UX,VB,UX4,UX4,UA,VA,UPX1,VPX1,URAPP, VAPP
2,URAPP,UPX4,VPX4,EPSRT,EPS0,EPS,CSC,EPSEM,UM3,UM4,UM5,UM6,UM4,VB4,ABPR3
3,UM7,UM8,UM9,XSTART,XEND

0003 DIMENSION UROOT(25),VROOT(25),MULTI(251),UAPP(25,3),VAPP(25,3),UWORX
1(26),VWORX(26),UB(26),VB(26),UA(26),VA(26),URAPP(25,3),VAPP(25,3)

0004 LOGICAL CONV

0005 COMMON EPSM,EPS,EPS0,EPSRT,10Z,MAX

0006 DATA PNAME,DONAME/ZHP(1),2HD(1/

0007 EPSM=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(IAPP.NE.0) GO TO 10

0018 NAPP=NP

0019 GO TO 315

0020 CALL GENAPP(UAPP, VAPP, NAPP, XSTART)

0021 GO TO 27

0022 DO 25 I=1,NAPP

0023 UAPP(I,1)=0.9*UAPP(I,2)

0024 VAPP(I,1)=0.9*VAPP(I,2)

0025 APP(I,3)=1.1*APP(I,2)

0026 27 KK=NP+1

0027 GO 30 T=1,KK

0028 UPXK(I)=UA(I)

0029 30 VWORX(I)=VA(I)

0030 NWORX=NP

0031 40 UX1=UAPP(IAPP,1)

0032 VX1=VAPP(IAPP,1)

0033 UX2=UAPP(IAPP,2)

0034 VX2=VAPP(IAPP,2)

0035 UX3=UAPP(IAPP,3)

0036 VX3=VAPP(IAPP,3)

0037 CALL HORNERN(WORK,UX1,UX2,UX3,UX,VB,UX1,VPX1,VPX2)

0038 CALL HORNERN(WORK,UX1,UX2,UX3,UX,VB,UX1,VPX1,VPX2)

0039 CALL HORNERN(WORK,UX1,UX2,UX3,UX,VB,UX1,VPX1,VPX2)

0040 50 CALL CALC(UX1,UX2,UX3,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX)

0041 CALL HORNERN(WORK,UX1,UX2,UX3,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX)

0042 60 CALL HORNERN(WORK,UX1,UX2,UX3,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX,UX)

0043 ABPX=4*DISORT(UPX4*UPX4+VPX4*VPX4)

0044 ABPX3=DSORT(UPX3*UPX3+VPX3*VPX3)
```

314
TABLE H.III (Continued)

0044 IF(IBPX3.EQ.0.0) GO TO 70
0045 QQQ=IBPX4/IBPX3
0046 IF(QQQ.LE.10.) GO TO 70
0047 UQ4=0.5*UQ4
0048 VX4=VX3+(UH3*UQ4+VH3*VQ4)
0051 GO TO 60
0052 CALL TEST(UX3,VX3,UX4,VX4,CONV)
0053 IF(CONV) GO TO 120
0054 IF(ITER.LT.MAX) GO TO 110
0055 CALL ALTER(UAPP(IAPP),VAPP(IAPP,1),VAPP(IAPP,2),UAPP(IAPP,3),VAPP(IAPP,4),NALTER,TIME)
0056 IF(NALTER.GT.5) GO TO 75
0057 ITER=1
0058 GO TO 40
0059 IF(IAPP.LT.NAPP) GO TO 100
0060 IF(XEND.EQ.0.0) GO TO 77
0061 IF(XSTART.GT.XEND) GO TO 77
0062 NAPP=NP
0063 CALL GENAPP(UAPP,VAPP,NAPP,XSTART)
0064 IAPP=0
0065 GO TO 100
0066 77 WRITE(IO2,1090)
0067 KKK=NWORK+1
0068 WRITE(IO2,1035) (DNAME,J,UWORK(I,J,1),VWORK(J,J1,KKK)
0070 IF(ITER.EQ.0) GO TO 62
0071 IF(IPATH.EQ.2) CALL BETTER(UA,VA,NP,URoot,VRoot,NRoot,URAPP,VRAPP,ROOT,MULT)
0072 RETURN
0073 IF(NROOT.EQ.0) GO TO 90
0074 IF(IPATH.EQ.1) WRITE(IO2,1080)
0075 IF(IPATH.EQ.1) GO TO 85
0076 WRITE(IO2,1090)
0077 DO 55 I=1,NRoot
0078 55 WRITE(IO2,1085) (URoot(I),VRoot(I),URAPP(I),VRAPP(I),I)
0080 IF(URoot(I).LT.ZERO) GO TO 85
0081 RETURN
0082 WRITE(IO2,1066) (URoot(I),VRoot(I),I=KKK,NRoot)
0083 IF(IPATH.EQ.1) WRITE(IO2,1080)
0084 RETURN NRoot=0
0085 RETURN
0086 100 IAPP=0
0088 100 RETURN
0089 NALTER=0
0090 GO TO 40
0091 120 NRoot=0
0092 IF(DTOWN) MULT(NRoot)=1
0094 MULT(NRoot)=NMultiplier+1
0095 URoot(NRoot)=UQ4
0096 VRoot(NRoot)=VQ4
0097 URAPP(NRoot,1)=URAPP(IAPP,1)
0098 VRAPP(NRoot,1)=VRAPP(IAPP,1)
0099 URAPP(NRoot,2)=URAPP(IAPP,2)
0100 VRAPP(NRoot,2)=VRAPP(IAPP,2)
TABLE H.III (Continued)

0101 \texttt{URAPP(NROOT,3)=VAPP(IAPP,3)}
0102 \texttt{VRAPP(NROOT,3)=VAPP(IAPP,3)}
0103 \texttt{125 IF(NOMULT.LT.NP) GO TO 130}
0104 \texttt{GO TO 60}
0105 \texttt{130 CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)}
0106 \texttt{NWORK=NWORK+1}
0107 \texttt{KKK=NWORK+1}
0108 \texttt{DO 140 I=1,KKK}
0109 \texttt{UWORK(I)=UB(I)}
0110 \texttt{140 VWORK(I)=VB(I)}
0111 \texttt{CALL HORNER(NWORK,UWORK,VWORK,UX4,VX4,UB,VB,UPX4,VPX4)}
0112 \texttt{CCC=DSORT(UB*UB+VB*VB) GO TO 150}
0113 \texttt{1F(CCC.LT.EPSM) GO TO 150}
0114 \texttt{IF(NWORK.GT.2) GO TO 75}
0115 \texttt{IROOT=NROOT}
0116 \texttt{KKK=NWORK+1}
0117 \texttt{DO 145 I=1,KKK}
0118 \texttt{UB(I)=UWORK(KKK+1-I)}
0119 \texttt{145 VB(I)=VWORK(KKK+1-I)}
0120 \texttt{CALL QUADS(NWORK,UB,VB,NROOT,UROOT,VROOT)}
0121 \texttt{GO TO 80}
0122 \texttt{MULT(NROOT)=MULT(NROOT)+1}
0123 \texttt{NOMULT=NOMULT+1}
0124 \texttt{GO TO 125}
0125 \texttt{110 UX1=UX2}
0126 \texttt{UX2=UX3}
0127 \texttt{UX3=UX4}
0128 \texttt{VX1=VX2}
0129 \texttt{VX2=VX3}
0130 \texttt{VX3=VX4}
0131 \texttt{UPX1=UPX2}
0132 \texttt{UPX2=UPX3}
0133 \texttt{VPX1=VPX2}
0134 \texttt{VPX2=VPX3}
0135 \texttt{UPX3=UPX4}
0136 \texttt{VPX3=VPX4}
0137 \texttt{ITER=ITER+1}
0138 \texttt{GO TO 50}
0139 \texttt{1090 FORMAT(///,5X,5HCOEFFICIENTS OF DEFLATED POLYNOMIAL FOR WHICH NO}}
0140 \texttt{1ZEROS WERE FOUND//)}
0141 \texttt{1030 FORMAT(///,1X,13HROOTS OF G(X),21X,INITIAL APPROXIMATION//)}
0142 \texttt{1035 FORMAT(///,1X,13HROOTS OF G(X),21X,INITIAL APPROXIMATION//)}
0143 \texttt{1036 FORMAT(12X,SHROOT(12,4H),=,010,16,3H,+,012,16,2H,1,19X,23HSOLVED}}
0144 \texttt{1BY DIRECT METHOD)}
0145 \texttt{1036 FORMAT(12X,SHROOT(12,4H),=,010,16,3H,+,012,16,2H,1,18X,23HSOLVED}}
0146 \texttt{END}
TABLE H.III (Continued)

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

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

**DOUBLE PRECISION**

**DOUBLE PRECISION EPS1, EPS2, EPS3, BETA**

**COMMON EPSM, EPS1, EPS2, EPS3, IO2, MAX**

**IF (XSTART.EQ.0.0) XSTART=0.5**

**BETA=0.2617994**

**DO 10 I=1, NAPP**

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

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

**BETA=BETA+0.5235988**

**10 XSTART=XSTART+0.5**

**DO 20 I=1, NAPP**

**APPR(I,3)=0.9*APPR(I,2)**

**APPI(I,3)=0.9*APPI(I,2)**

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

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

**RETURN**

**END**
TABLE H. III (Continued)

```fortran
0001 C SUBROUTINE ALTER(X1R, X1I, X2R, X2I, X3R, X3I, NALTER, TIME)
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
0004 COMMON EPS3, EPS5, EPS2, EPS3, IO2, MAX
0005 IF ITIME.EQ.01 GO TO 5
0006 ITEMS = 1
0007 WRITE(102,1010) MAX
0008 5 IFINALTER.EQ.0) GO TO 10
0009 WRITE(102,1000) X1R, X1I, X2R, X2I, X3R, X3I
0010 GO TO 20
0011 10 R=DSQRT(X1R*X1R+X2R*X2R+X3R*X3R)
0012 BETA=DATAN2(X1I,X2R)
0013 WRITE(102,1020) X1R, X1I, X2R, X2I, X3R, X3I
0014 20 NALTER= NALTER+1
0015 IFINALTER.GT.5) RETURN
0016 GO TO (30,40,50,60,70,80,90,100,110)
0017 30 X2R=-X1R
0018 X2I=-X1I
0019 GO TO 50
0020 40 BETA=BETA+1.0471976
0021 X2R=R*DGOS4(BETA)
0022 X2I=R*DSIN4(BETA)
0023 50 X1R=X1R+0.9*X2R
0024 X1I=X1I+0.9*X2I
0025 X3R=X3R+1.1*X2R
0026 X3I=X3I+1.1*X2I
0027 RETURN
0028 1000 FORMAT(1X,5HXL = ,D23.16,3H + ,D23.16,2H I,10X,22HALTERED APPROXIM
ATIONS/1X,5HXK = ,D23.16,3H + ,D23.16,2H I/1X,5HXK = ,D23.16,3H +
+ Z/D23.16,2H I/)!
0029 1020 FORMAT(1H0,5HXL = ,D23.16,3H + ,D23.16,2H I,10X,22INITIAL APPROXIM
ATIONS/1X,5HXK = ,D23.16,3H + ,D23.16,2H I/1X,5HXK = ,D23.16,3H +
+ Z/D23.16,2H I/)!
0030 1010 FORMAT(1X,5HNO CONVERGENCE FOR THE FOLLOWING APPROXIMATIONS AF
ITER /13,12H ITERATIONS/)
0031 END
```
TABLE H.III (Continued)

<table>
<thead>
<tr>
<th>0001</th>
<th>SUBROUTINE BETTER</th>
<th>(UA, VA, N_P, UROOT, VROOT, NROOT, URAPP, VRAPP, IROOT, MUL</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>SUBROUTINE BETTER ATTEMPTS TO IMPROVE THE ACCURACY OF THE ZEROS FOUND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>BY USING THEM AS INITIAL APPROXIMATIONS WITH MULLER'S METHOD APPLIED TO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>THE FULL, UNDEFLATED POLYNOMIAL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 0002 | DOUBLE PRECISION UROOT, VROOT, UA, VA, URAPP, VRAPP, UX1, UX2, VX1, VX2, UX3, VX3, UPX1, UPX2, UPX3, VPX1, VPX2, VPX3, UB, VB, UROOTS, VROOTS, EPSRT, UX4, VX4, UPX4, VPX4, LH3, VH3 |
| 0003 | DOUBLE PRECISION EPSM |
| 0004 | LOGICAL CONV |
| 0005 | DIMENSION UROOT(253), VROOT(255), UA(26), VA(26), URAPP(253, 3), VRAPP(253, 3), UX1(253), VX1(253), UROOTS(253), VROOTS(253), URAPP(253, 3), VRAPP(253, 3), MULT(253) |

<p>| 0006 | COMMON EPSM, EPS, EPSQ, EPSRT, IO2, MAX |
| 0007 | IF (NROOT.LT.1) RETURN |
| 0008 | L=0 |
| 0009 | GO TO 10 |
| 0010 | UAPP(I,1)=UROOT(I)*EPSRT |
| 0011 | VAPP(I,1)=VROOT(I)*EPSRT |
| 0012 | UAPP(I,2)=UROOT(I) |
| 0013 | VAPP(I,2)=VROOT(I) |
| 0014 | UAPP(I,3)=UROOT(I)<em>12.0</em>EPSRT |
| 0015 | 10 VAPP(I,3)=VROOT(I)<em>12.0</em>EPSRT |
| 0016 | DO 100 J=1, NROOT |
| 0017 | UX1=URAPP(J,1) |
| 0018 | VX1=VRAPP(J,1) |
| 0019 | UX2=URAPP(J,2) |
| 0020 | VX2=VRAPP(J,2) |
| 0021 | UX3=URAPP(J,3) |
| 0022 | VX3=VRAPP(J,3) |
| 0023 | ITER=1 |
| 0024 | CALL HORNER(NP, UA, VA, UX1, VX1, UB, VB, UPX1, VPX1) |
| 0025 | CALL HORNER(NP, UA, VA, UX2, VX2, UB, VB, UPX2, VPX2) |
| 0026 | CALL HORNER(NP, UA, VA, UX3, VX3, UB, VB, UPX3, VPX3) |
| 0027 | CALL CALC(UX1, VX1, UX2, VX2, UX3, VX3, UPX1, VPX1, UPX2, VPX2, UPX3, VPX3, UX4, VX4, UPX4, VPX4, LH3, VH3) |
| 0028 | 30 CALL TEST(U3, VX3, UX4, VX4, CONV) |
| 0029 | IF (CONV) GO TO 50 |
| 0030 | IF (ITER.LT. MAX) GO TO 40 |
| 0031 | WRITE(102, 1000) J, UROOT(I, J), VROOT(I, J), MAX |
| 0032 | WRITE(102, 1010) UX4, VX4 |
| 0033 | IF (J.LT. IROOT) GO TO 33 |
| 0034 | IF (J.GT. IROOT) GO TO 35 |
| 0035 | GO TO 100 |
| 0036 | 33 KKK=IROOT-1 |
| 0037 | DO 34 JJ=KKK |
| 0038 | URAPP(K, JJ)=URAPP(K, JJ+1) |
| 0039 | VRAPP(K, JJ)=VRAPP(K, JJ+1) |
| 0040 | URAPP(K, JJ)=URAPP(K, JJ+1) |
| 0041 | VRAPP(K, JJ)=VRAPP(K, JJ+1) |
| 0042 | URAPP(K, JJ)=URAPP(K, JJ+1) |
| 0043 | 34 VRAPP(K, JJ)=VRAPP(K, JJ+1) |
| 0044 | 35 IROOT=IROOT-1 |
| 0045 | GO TO 100 |</p>
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0046</td>
<td>40</td>
<td>UX1=UX2</td>
</tr>
<tr>
<td>0047</td>
<td>40</td>
<td>VX1=VX2</td>
</tr>
<tr>
<td>0048</td>
<td>40</td>
<td>UX2=UX3</td>
</tr>
<tr>
<td>0049</td>
<td>40</td>
<td>VX2=VX3</td>
</tr>
<tr>
<td>0050</td>
<td>40</td>
<td>UX3=UX4</td>
</tr>
<tr>
<td>0051</td>
<td>40</td>
<td>VX3=VX4</td>
</tr>
<tr>
<td>0052</td>
<td>40</td>
<td>UPX1=UPX2</td>
</tr>
<tr>
<td>0053</td>
<td>40</td>
<td>VPX1=VPX2</td>
</tr>
<tr>
<td>0054</td>
<td>40</td>
<td>UPX2=UPX3</td>
</tr>
<tr>
<td>0055</td>
<td>40</td>
<td>VPX2=VPX3</td>
</tr>
<tr>
<td>0056</td>
<td>40</td>
<td>ITERS=ITERS+1</td>
</tr>
<tr>
<td>0057</td>
<td>50</td>
<td>GO TO 20</td>
</tr>
<tr>
<td>0058</td>
<td>50</td>
<td>L=L+1</td>
</tr>
<tr>
<td>0059</td>
<td>50</td>
<td>UROOTS(L)=UX4</td>
</tr>
<tr>
<td>0060</td>
<td>50</td>
<td>VROOTS(L)=VX4</td>
</tr>
<tr>
<td>0061</td>
<td>100</td>
<td>CONTINUE</td>
</tr>
<tr>
<td>0062</td>
<td>100</td>
<td>IF (L.EQ.0) GO TO 120</td>
</tr>
<tr>
<td>0063</td>
<td>110</td>
<td>DO 110 I=1,L</td>
</tr>
<tr>
<td>0064</td>
<td>110</td>
<td>UROOT(I)=UROOTS(I)</td>
</tr>
<tr>
<td>0065</td>
<td>110</td>
<td>VROOT(I)=VROOTS(I)</td>
</tr>
<tr>
<td>0066</td>
<td>110</td>
<td>NROOT=L</td>
</tr>
<tr>
<td>0067</td>
<td>110</td>
<td>RETURN</td>
</tr>
<tr>
<td>0068</td>
<td>110</td>
<td>NROOT=0</td>
</tr>
<tr>
<td>0069</td>
<td>110</td>
<td>RETURN</td>
</tr>
<tr>
<td>0070</td>
<td>110</td>
<td>FORMAT(/42H IN THE ATTEMPT TO IMPROVE ACCURACY, ROOT(I,12,4H) = , T023.16,2H * ,023.16,2H I/2,H DID NOT CONVERGE AFTER ,13,11H ITERATIONS)</td>
</tr>
<tr>
<td>0071</td>
<td>110</td>
<td>FORMAT(30H THE PRESENT APPROXIMATION IS ,023.16,3H * ,023.16,2H I/ 1/)</td>
</tr>
<tr>
<td>0072</td>
<td>110</td>
<td>END</td>
</tr>
</tbody>
</table>
TABLE E.11I (Continued)

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

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

0012 DOUBLE PRECISION UX3, UX4, VX3, VX4, EPSRT, EPS0, EPS, AAA, VDUMMY, VDUMMY,
  DENOM
0013 DOUBLE PRECISION EPSM
0014 LOGICAL CONV
0015 COMMON EPSM, EPS, EPS0, EPSRT, MAX
0016 VDUMMY=UX4-UX3
0017 VDUMMY=VX4-UX3
0018 AAA=DSQRT(VDUMMY*VDUMMY+VDUMMY*VDUMMY)
0019 DENOM=DSQRT(UX4+UX4+VX4+VX4)
0100 IF (DENOM.LT.EPS0) GO TO 20
0110 IF (AAA/DENOM.LT.EPS1) GO TO 10
0112 5 CONV=.FALSE.
0113 GO TO 100
0114 10 CONV=.TRUE.
0115 GO TO 100
0116 20 IF (AAA.LT.EPS0) GO TO 10
0117 GO TO 5
0118 100 RETURN
0119 END

0001 SUBROUTINE HORNER(UA, UA, UX, VX, UB, VB, UBP, VBP)

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

0002 DOUBLE PRECISION UX, VX, UBP, VBP, UB, UB, UA
0003 DIMENSION UA(26), VA(26), UB(26), VB(26)
0004 UBI=UB(1)
0005 VB(1)=VA(1)
0006 NUM=NU+1
0007 DO 10 I=2, NUM
0008 UBI=UB(I)+U(B[I-1]*UX-DB[I-1]*VX)
0009 10 UBP(1)=VA(I)+VB[I-1]*U/I+UB[I-1]*VX
0010 UXI=-UB(INUM)
0011 VBP=VBP[NUM]
0012 RETURN
0013 END