NASA TECHNICAL NOTE

NASA TN D-8009

NASA TN D-8009

AN ALGORITHM AND COMPUTER PROGRAM TO LOCATE REAL ZEROS OF REAL POLYNOMIALS

David R. Hedgley, Jr.

Flight Research Center
Edwards, Calif. 93523

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JUNE 1975

The computer program and subroutines listed in the APPENDIX of this report have been punched on cards and are available in Bldg 413, Room 229 (Mr Havens / 264-9852). However, we have been unable to execute the program successfully. and we have not yet located the error(s).

1. Report No.
NASA TN D-8009

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle

AN ALGORITHM AND COMPUTER PROGRAM TO LOCATE REAL ZEROS OF REAL POLYNOMIALS

5. Report Date
June 1975

6. Performing Organization Code
H-855

7. Author(s)

David R. Hedgley, Jr.

8. Performing Organization Report No.

9. Performing Organization Name and Address

NASA Flight Research Center
P. O. Box 273
Edwards, California 93523

10. Work Unit No.
970-43-10

11. Contract or Grant No.

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, D. C. 20546

13. Type of Report and Period Covered
Technical Note

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A method for reliably extracting real zeros of real polynomials using an expanded two-point secant and bisection method is formed into an algorithm for a digital computer, and a computer program based on this algorithm is presented. The results obtained with the program show that the proposed method compares favorably with the Laguerre, Newton-Raphson, and Jenkins-Traub methods when the polynomial has all real zeros, and is more efficient when the polynomial has complex zeros.

17. Key Words (Suggested by Author(s))

Real zeros
Polynomials

18. Distribution Statement

Unclassified - Unlimited

Category: 64

19. Security Classif. (of this report)
Unclassified

20. Security Classif. (of this page)
Unclassified

21. No. of Pages
24

22. Price*
$3.25

*For sale by the National Technical Information Service, Springfield, Virginia 22151

AN ALGORITHM AND COMPUTER PROGRAM TO LOCATE REAL ZEROS OF REAL POLYNOMIALS

David R. Hedgley, Jr.
Flight Research Center

INTRODUCTION

Finding the real zeros of polynomials is a classical problem in almost every technical discipline (ref. 1). It has assumed major importance in the last two decades in the treatment of the masses of data that have accompanied the growth of technology.

Many methods for locating the zeros of real polynomials are being used. However, all of these methods, which locate the zeros of a real polynomial with no prior information regarding the location of the zeros, find both the real and the complex zeros. Moreover, many of the methods have inherent weaknesses. For example, the polynomial $x^{20} - 1$ causes the Newton-Raphson approach to diverge near 1 or -1. Bairstow's method requires close approximations to a zero; otherwise the results may be erroneous. Laguerre's method is satisfactory if the polynomial has all real zeros; however, if it has complex zeros, little can be said about its behavior. Reference 2 discusses these methods in more detail. Finally, the Jenkins-Traub algorithm, which is considered the most advanced method, has difficulty with zeros which form a cluster.

In addition to these anomalies, the methods are inefficient when only real zeros are desired. Furthermore, because of possible computational inaccuracies, which are to a large degree a function of word size limitations of computers, real zeros can be mistaken for complex zeros when the imaginary part of the zero is small in absolute value.

The intent of this paper is to present an algorithm which (1) presupposes no knowledge of the location or number of real zeros and (2) compares favorably with the standard methods when a polynomial has all real zeros but (3) demonstrates a pronounced superiority in efficiency when the polynomial has complex zeros.

A computer program to implement the algorithm is presented, and results from the Laguerre, Newton-Raphson, and Jenkins-Traub methods are compared with results obtained from the proposed method.

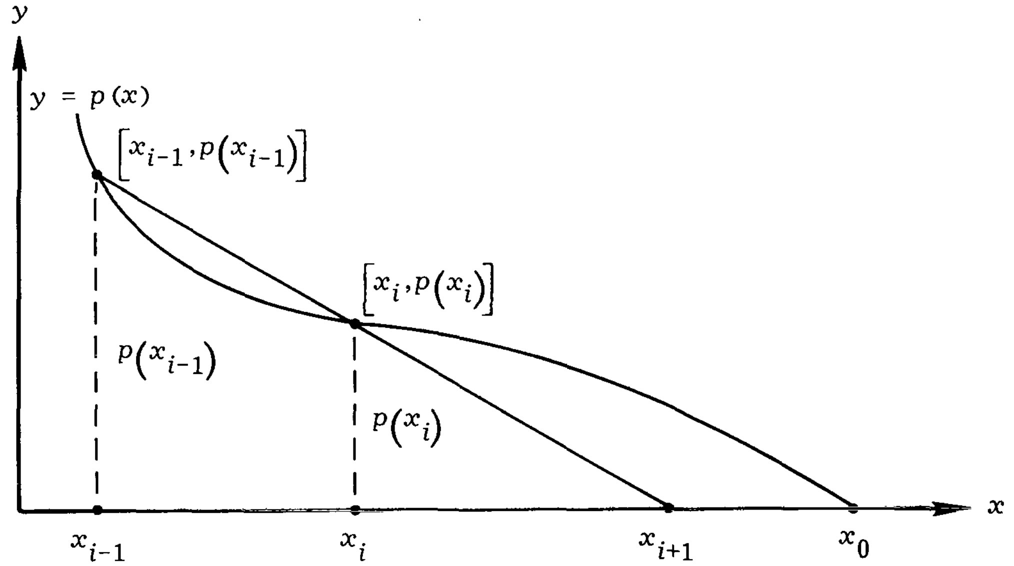
BACKGROUND

Three significant criteria for evaluating a technique which locates real zeros of a real polynomial are: its inherent rate of convergence, the computational time required for each iteration, and the probability of convergence. The two-point secant method for locating real zeros is given by the equation

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{p(x_i) - p(x_{i-1})} p(x_i)$$

in which x_i is an arbitrary variable on the real axis, x_{i+1} is the next iterate determined by x_i and x_{i-1}, and $p(x_i)$ is the value of a real polynomial, $p(x)$, at x_i. This method has an excellent rate of convergence, $(1 + \sqrt{5})/2$ (ref. 2). The computational time per iteration is small because only $p(x_{i+1})$ need be computed after the first iteration. Furthermore, this formulation of the secant method enhances numerical stability, since only a few significant digits are required as convergence is neared (ref. 2). Therefore, because this method satisfies the convergence and computational time criteria, it was selected for further development of the proposed algorithm.

The following sketch is a geometrical representation of the iterative process of the two-point secant method:



In the equation for the secant method, initially some estimates for x_i and x_{i-1} are made. Geometrically, x_{i+1} is the x-intercept of a straight line determined by

$[x_i, p(x_i)]$ and $[x_{i-1}, p(x_{i-1})]$, which is a secant through the curve $p(x)$. Having determined x_{i+1}, x_i and x_{i+1} are used to generate x_{i+2}, and so on.

Although this process is efficient when it converges, it may not always converge to a zero. If the curve is peculiar, it may diverge, that is, $\left|\frac{p(x_{i+1})}{p(x_i)}\right| \geq 1$.

If the zero, r, is simple, the secant method will converge to r for x_i and x_{i-1} sufficiently "close" to r, that is, $|p(x_i)| < t$, where t is an arbitrary tolerance limit (ref. 2). However, if r is of multiplicity greater than 1, it may not converge for any choice of step size, Δ_j, such that $|x_i - r| < \Delta_j$ and $|x_{i-1} - r| < \Delta_j$. In fact, it may happen that for $x_i \neq x_{i-1}$, $p(x_i) = p(x_{i-1})$. This condition obviously leads to divergence.

On the basis of this analysis of the strengths and weaknesses of the secant method, the proposed algorithm was developed, as discussed in the next section.

ALGORITHM DEVELOPMENT

Increasing the probability of convergence of the secant method requires a theorem on the bounds of real zeros of a real polynomial, $p(x)$. The following theorem was adapted from reference 3:

Theorem 1. If $p(x) = C_n x^n + C_{n-1}x^{n-1} \ldots + C_0$, $(C_n > 0)$ is a real polynomial (where $C_i, i = 0, n$ are the coefficients of $p(x)$) and if the first negative coefficients are preceded by k coefficients which are positive or zero, and if g denotes the greatest number in absolute value of the negative coefficients, then each positive real zero is less than the quantity $1 + \sqrt[k]{g/C_n}$.

This theorem makes it possible to find an interval, I, which contains all the real zeros of $p(x)$, for, by the theorem, the lower bound for the real zeros of $p(x)$ is the negative of the upper bound for $p(-x)$.

We now define Δ_j in the following way:

$$\Delta_j = \frac{\pm I}{2^j} \qquad j = 1, 2, 3, \ldots, n$$

where n = greatest integer less than or equal to $\log_2 I/\varepsilon$ in which ε is an arbitrarily small constant less than 1, and $I = \max(|UB|, |LB|)$ in which max is the larger of the upper bound, *UB*, or the lower bound, *LB*, value of the real zeros of $p(x)$. The ± indicates that the iteration proceeds within the bounded interval in both the positive and the negative direction and in an alternating manner. This scheme usually permits the extraction of real zeros in increasing magnitude, thus preserving accuracy (ref. 2).

Further, we define x_i and x_{i-1}, the initial choices, with step size as follows:

$$\left.\begin{aligned} x_i &= L\Delta_j \\ x_{i-1} &= (L - 1)\Delta_j \end{aligned}\right\} \qquad L = 1, 2, 3, \ldots, k;\ k = 2^j$$

where j is held fixed.

The process is completed if for $\Delta_j, j = 1$ and $L = 1$, x_i and x_{i-1} lead to convergence. If not, $L = 2$ is selected, keeping Δ fixed, and the process is repeated. If, however, no choice of L for $j = 1$ leads to convergence, step size Δ_2 is chosen where $\Delta_2 = \Delta_1/2$ by the preceding definition, and the previous steps are repeated. That is, every L for each $\Delta_j, j = 1, 2, \ldots, n$ has the potential for convergence where $\Delta_n < \Delta_{n-1} < \Delta_{n-2} \ldots < \Delta_1$.

This iterative scheme for choosing initial values x_i and x_{i-1} increases the probability that the secant method will converge, provided $p(x)$ has a real zero, for we know that the secant method will converge to r, a simple zero, for x_i, and x_{i-1} close to r. In fact, it may converge when r is not simple and x_i and x_{i-1} are not close to r. Clearly, if x_i and x_{i-1} are assured of being close to r by decreasing Δ_j, the probability of convergence is greater because of the large number of available choices as well as the fact that at least one pair will be close to a real zero.

Finally, if this scheme is not successful and if the existence of at least one real zero is assumed, it is highly probable that r is not a simple zero. Then consider the following reasoning. Let ε be such that when $\Delta_j < \varepsilon$ the secant method is no longer considered fruitful. If for $\Delta_j < \varepsilon$, $p\left(x_i\right)$ does not converge, find b such that $|p(b)| = \min\left|p\left(x_i\right)\right|$, where min is the minimum value and x_i is any value chosen or computed for which $\Delta_j < \varepsilon$ and divergence occurred for every $\Delta_j > \varepsilon$. From the step size and because of the continuity of $p(x)$, we shall assume that a real zero, r, exists such that $|b - r| \leq \Delta_j$. Consider the closed interval $\left[b - \Delta_j, b + \Delta_j\right]$. Using b as a center and subjecting this interval to the bisection method whose direction of seek is governed by the absolute value of the end points at every subsequent subinterval, it is again probable that $p\left(x_i\right)$ will converge to $p(r)$.

Although the combination of the preceding schemes is efficient in locating a real zero, its efficiency and validity in determining if all the real zeros have been located or if any real zeros exist can be poor, particularly in higher degree polynomials with complex zeros. Sturm's theorem (ref. 2) removes this difficulty. The theorem is stated as follows:

Theorem 2. Let $\{p_i\}(1 \leq i \leq n + 1)$ be a sequence of polynomials related to $p(x)$ of degree n in the following way:

$$p_1(x) = p(x) \qquad\qquad p_2(x) = p'(x)$$

$$p_{i-1}(x) = q_{i-1}(x)p_i(x) - p_{i+1}(x) \qquad\qquad i = 2, 3, \ldots, m - 1 \leq n$$

$$p_{m-1}(x) = q_{m-1}(x)p_m(x)$$

where $q_{i-1}(x)$ is the quotient when $p_{i-1}(x)$ is divided by $p_i(x)$, and $p_{i+1}(x)$ is the negative of the remainder. If $[e,f]$ is any interval on the real axis such that $p(e)$ and $p(f) \neq 0$, then $\nu(e) - \nu(f)$ is the number of distinct real zeros in $[e,f]$ where $\nu(e)$ and $\nu(f)$ represent the number of variations of sign of $\{p_i(x)\}$ evaluated at e and f, respectively.

Since at $x = \pm I$, $p(x)$ does not vanish, the implication is that the number of distinct real zeros of $p(x)$ can be determined by using Theorem 2. Moreover, once a zero is located and the polynomial is deflated to give, for example, $d(x)$, a new I is determined using Theorem 1 and, hence, the number of distinct real zeros, if any, for $d(x)$ can be determined, and so forth.

Thus, for any real polynomial, $p(x)$, the status of completion with respect to all the real zeros including multiplicities can be ascertained efficiently and accurately with Theorems 1 and 2 by considering subsequent deflated polynomials and their corresponding interval of bounds in the same way.

IMPLEMENTATION AND RESULTS

The proposed algorithm was implemented by using an assembly of computer programs. Listings for the programs, together with brief descriptions and flow charts, are presented in the appendix. The programs were run on a Control Data Corporation Cyber 73-28 computer. The algorithm was applied to five polynomials: a fourth-order polynomial with a non-simple zero; a fourth-order polynomial with both simple and non-simple zeros; a thirteenth-order polynomial with all distinct real zeros which form a cluster; a fifteenth-order polynomial with complex zeros; and a twenty-fifth-order polynomial with complex zeros. The results are compared in tables 1 to 5 with results obtained on the CDC Cyber 73-28 computer for the Laguerre, Newton-Raphson, and Jenkins-Traub methods. A subroutine called ZPOLYR, obtained from International Mathematical and Statistical Laboratories, Inc., was used to implement Laguerre's method. The Newton-Raphson method was implemented by using a subroutine called POLRT from the IBM Scientific Subroutine

Package. The Jenkins-Traub method was implemented by using the subroutine ZRPOLY, also taken from the International Mathematical and Statistical Laboratories, Inc. The zeros located by each of the methods are listed in the order found.

TABLE 1.—COMPARISON OF RESULTS OBTAINED FOR A FOURTH-ORDER POLYNOMIAL WITH A NON-SIMPLE ZERO

Polynomial coefficients:

.10000E+01
-.40000E+01
.60000E+01
-.40000E+01
.10000E+01

Laguerre method		Newton-Raphson method		Jenkins-Traub method		Proposed method
Zeros		Zeros		Zeros		Real zeros
Real	Imaginary	Real	Imaginary	Real	Imaginary	
.10000E+01	.80831E-07	.99998E+00	0.	.10000E+01	0.	.10000E+01
.10000E+01	-.80831E-07	.10000E+01	0.	.10000E+01	0.	.99996E+00
.10000E+01	0.	.10000E+01	0.	.10000E+01	0.	.10000E+01
.10000E+01	0.	.10000E+01	0.	.10000E+01	0.	.99998E+00
Execution time, sec						
0.030		0.195		0.030		0.064

TABLE 2.—COMPARISON OF RESULTS OBTAINED FOR A FOURTH-ORDER POLYNOMIAL WITH ALL REAL ZEROS

Polynomial coefficients:

.58500E+02
-.18000E+02
-.17500E+02
.40000E+01
.10000E+01

Laguerre method		Newton-Raphson method		Jenkins-Traub method		Proposed method
Zeros		Zeros		Zeros		Real zeros
Real	Imaginary	Real	Imaginary	Real	Imaginary	
.21231E+01	0.	.21231E+01	0.	.21008E+01	0.	-.21213E+01
-.61231E+01	0.	-.61231E+01	0.	-.21213E+01	0.	.21231E+01
-.21213E+01	0.	-.21213E+01	0.	.21436E+01	0.	.21213E+01
.21213E+01	0.	.21213E+01	0.	-.61231E+01	0.	-.61231E+01
Execution time, sec						
0.042		0.089		0.031		0.047

TABLE 3.—COMPARISON OF RESULTS FOR A THIRTEENTH-ORDER POLYNOMIAL WITH ALL DISTINCT REAL ZEROS WHICH FORM A CLUSTER

Polynomial coefficients:

.30974E+03
.26695E+04
.10563E+05
.25410E+05
.41464E+05
.48468E+05
.41758E+05
.26849E+05
.12884E+05
.45580E+04
.11554E+04
.19877E+03
.20800E+02
.10000E+01

Laguerre method		Newton-Raphson method		Jenkins-Traub method		Proposed method
Zeros		Zeros		Zeros		Real zeros
Real	Imaginary	Real	Imaginary	Real	Imaginary	
-.20997E+01	0.	-.10000E+01	0.	-.11005E+01	0.	-.10000E+01
-.22000E+01	0.	-.11000E+01	0.	-.99996E+00	0.	-.11000E+01
-.20011E+01	0.	-.12001E+01	0.	-.13091E+01	0.	-.12000E+01
-.18970E+01	0.	-.12995E+01	0.	-.11976E+01	0.	-.13000E+01
-.18051E+01	0.	-.14017E+01	0.	-.15510E+01	0.	-.14000E+01
-.16936E+01	0.	-.14965E+01	0.	-.15463E+01	0.	-.15003E+01
-.16057E+01	0.	-.16057E+01	0.	-.13829E+01	0.	-.15995E+01
-.14965E+01	0.	-.16936E+01	0.	-.17802E+01	0.	-.17005E+01
-.14017E+01	0.	-.18051E+01	0.	-.19065E+01	0.	-.17997E+01
-.12995E+01	0.	-.18970E+01	0.	-.17277E+01	0.	-.19000E+01
-.12001E+01	0.	-.20011E+01	0.	-.21003E+01	0.	-.20001E+01
-.11000E+01	0.	-.20997E+01	0.	-.19980E+01	0.	-.21000E+01
-.10000E+01	0.	-.22000E+01	0.	-.22000E+01	0.	-.22000E+01
Execution time, sec						
0.220		0.950		0.285		0.400

TABLE 4.—COMPARISON OF RESULTS FOR A FIFTEENTH-ORDER POLYNOMIAL WITH COMPLEX ZEROS

Polynomial coefficients:

-.10000E+01
.20000E+01
-.30000E+01
.40000E+01
-.50000E+01
.60000E+01
-.70000E+01
.80000E+01
-.90000E+01
.10000E+02
-.11000E+02
.12000E+02
-.13000E+02
.14000E+02
-.15000E+02
.16000E+02

Laguerre method		Newton-Raphson method		Jenkins-Traub method		Proposed method
Zeros		Zeros		Zeros		Real zeros
Real	Imaginary	Real	Imaginary	Real	Imaginary	
-.56747E+00	.63594E+00	.80860E+00	0.	.80860E+00	0.	.80860E+00
-.56747E+00	-.63594E+00	.75103E+00	-.30205E+00	-.56747E+00	.63594E+00	
.58601E+00	-.56241E+00	.75103E+00	.30205E+00	-.56747E+00	-.63594E+00	
.58601E+00	.56241E+00	-.79267E+00	-.38555E+00	-.28130E+00	.78671E+00	
.80860E+00	0.	-.79267E+00	.38555E+00	-.28130E+00	-.78671E+00	
-.28130E+00	-.78671E+00	.33562E+00	-.74495E+00	.33562E+00	.74495E+00	
-.28130E+00	-.78671E+00	.33562E+00	.74495E+00	.33562E+00	-.74495E+00	
-.79267E+00	-.38555E+00	.33212E-01	-.82381E+00	.33212E-01	.82381E+00	
-.79267E+00	.38555E+00	.33212E-01	.82381E+00	.33212E-01	-.82381E+00	
.33212E-01	-.82381E+00	-.56747E+00	-.63594E+00	-.79267E+00	.38555E+00	
.33212E-01	.82381E+00	-.56747E+00	.63594E+00	-.79267E+00	-.38555E+00	
.75103E+00	-.30205E+00	.58601E+00	-.56241E+00	.58601E+00	.56241E+00	
.75103E+00	.30205E+00	.58601E+00	.56241E+00	.58601E+00	-.56241E+00	
.33562E+00	.74495E+00	-.28130E+00	-.78671E+00	.75103E+00	.30205E+00	
.33562E+00	-.74495E+00	-.28130E+00	.78671E+00	.75103E+00	-.30205E+00	
Execution time, sec						
19.910		0.466		0.208		0.068

TABLE 5.—COMPARISON OF RESULTS FOR A TWENTY-FIFTH-ORDER POLYNOMIAL WITH COMPLEX ZEROS

Polynomial coefficients:

.10000E+01	.14000E+02
.20000E+01	.15000E+02
.30000E+01	.16000E+02
.40000E+01	.17000E+02
-.44000E+02	.18000E+02
.60000E+01	.19000E+02
.70000E+01	.20000E+02
.80000E+01	.21000E+02
.90000E+01	.22000E+02
.10000E+02	.23000E+02
.11000E+02	.24000E+02
.12000E+02	.25000E+02
.13000E+02	.26000E+02

Laguerre method		Newton-Raphson method		Jenkins-Traub method		Proposed method
Zeros		Zeros		Zeros		Real zeros
Real	Imaginary	Real	Imaginary	Real	Imaginary	
No solution	No solution	-.32799E+00	0.	-.57462E-01	.35331E+00	-.32799E+00
		-.57462E-01	-.35331E+00	-.57462E-01	-.35331E+00	.59100E+00
		-.57462E-01	.35331E+00	-.32799E+00	0.	.73831E+00
		.59100E+00	0.	.59100E+00	0.	
		.73831E+00	0.	.79772E+00	.61254E+00	
		-.10511E+01	-.15500E+00	.79772E+00	-.61254E+00	
		-.10511E+01	.15500E+00	.73831E+00	0.	
		.33707E+00	-.98054E+00	-.54917E+00	.90186E+00	
		.33707E+00	.98054E+00	-.54917E+00	-.90186E+00	
		.91694E+00	-.33985E+00	.40625E-01	.10441E+01	
		.91694E+00	.33985E+00	.40625E-01	-.10441E+01	
		-.54917E+00	-.90186E+00	.59772E+00	.83228E+00	
		-.54917E+00	.90186E+00	.59772E+00	-.83228E+00	
		-.96101E+00	-.45127E+00	.33707E+00	.98054E+00	
		-.96101E+00	.45127E+00	.33707E+00	-.98054E+00	
		-.26412E+00	-.10172E+01	-.78862E+00	.70767E+00	
		-.26412E+00	.10172E+01	-.78862E+00	-.70767E+00	
		.59772E+00	-.83228E+00	-.26412E+00	.10172E+01	
		.59772E+00	.83228E+00	-.26412E+00	-.10172E+01	
		.79772E+00	-.61254E+00	.91694E+00	.33985E+00	
		.79772E+00	.61254E+00	.91694E+00	-.33985E+00	
		-.78862E+00	-.70767E+00	-.96101E+00	.45127E+00	
		-.78862E+00	.70767E+00	-.96101E+00	-.45127E+00	
		.40625E-01	-.10441E+01	-.10511E+01	.15500E+00	
		.40625E-01	.10441E+01	-.10511E+01	-.15500E+00	
Execution time, sec						
-----		1.559		0.438		0.238

These results show that the proposed method compares favorably with the Laguerre, Newton-Raphson, and Jenkins-Traub methods when the polynomial has all real zeros, and is more efficient when the polynomial has complex zeros. Moreover, as shown in table 1, Laguerre's method identifies complex zeros when in fact the zeros are real. This discrepancy is possible with any method that locates real and complex zeros, thus demonstrating the advantage of a method which locates only real zeros.

EVALUATION OF ALGORITHM

The proposed algorithm, by using Theorem 1 and Theorem 2 in combination, significantly reduces the difficulty of determining the number of real zeros of a polynomial and, hence, the status of completion. Additionally, the modified bisection method, which is used when the secant method used iteratively does not lead to convergence for non-simple zeros, appreciably improves the probability of convergence. Since the predominant method is the secant method which has near quadratic convergence and small computational time per iteration, the proposed algorithm satisfies the previously stated criteria on rate and probability of convergence and computational time.

Flight Research Center
National Aeronautics and Space Administration
Edwards, Calif., February 24, 1975

APPENDIX — COMPUTER PROGRAM

PROGRAM DESCRIPTION

The digital computer programs which implement the proposed algorithm were written in FORTRAN IV and occupy approximately 1000 decimal words excluding FORTRAN system routines. A user-written program calls the REALRT subroutine, which calls the remaining subroutines.

SAMPLE OF USER-WRITTEN CALLING PROGRAM

The following program, an example of a user-written program, constructs the polynomial $x^4 - 8000x^3 + 3x^2 + 10x - 30\ 000 = p(x)$ and computes all the real zeros of this polynomial.

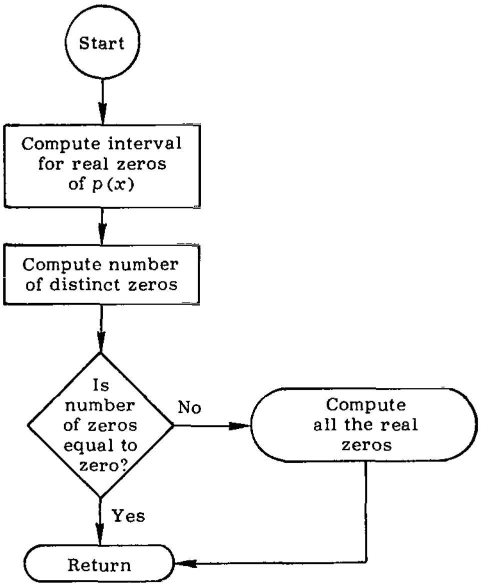
```
      PROGRAM PET(INPUT,OUTPUT,TAPE3=OLTPUT,TAPE1=INPUT)
      DIMENSION A(30),B(30),ROCT(30)
      DIMENSION C(30)
   22 FCRMAT(2F12.4)
      A(1)=-30000
      A(2)=10
      A(3)=1          A(3) = 3
      A(4)=-8000
      A(5)=1
      N=4
      CALL REALRT(A,N,ROCT,NI)
      WRITE(3,7550)
 7550 FCRMAT(1H1)
      NN=N+1
      WRITE(3,24)
      DO 191 J=1,NN
  191 WRITE(3,78)A(J)
      WRITE(3,25)
      DO 190 J=1,NI
      WRITE(3,77)ROOT(J)
  190 CONTINUE
   24 FCRMAT(10X,23HPOLYNOMIAL COEFFICIENTS/)
   25 FORMAT(//10X,20HREAL ZEROES FOR P(X)/)
   78 FORMAT(3X,E15.5)
   77 FORMAT(3X,E15.5)
      STOP
      END
```

SUBROUTINES

Subroutine REALRT (A, N, ROOT, NI)

Purpose: To locate all the real zeros of a real polynomial of degree less than 30 and greater than 1

Flow chart:


Start
Compute interval for real zeros of p (x)
Compute number of distinct zeros
Is number of zeros equal to zero?
No
Compute all the real zeros
Yes
Return

Subroutine arguments:

A	array of coefficients of $p(x)$
N	degree of $p(x)$
ROOT	array containing the real zeros
NI	number of real zeros in array root

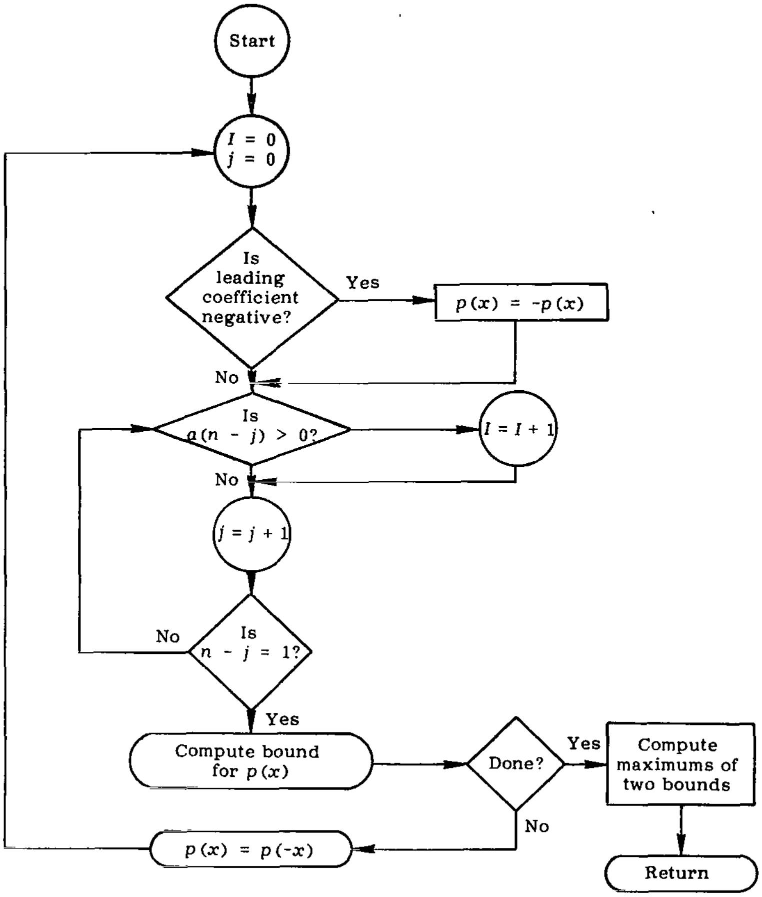
Subroutine listing:

```
      SUBROUTINE REALRT(A,N,ROOT,NI)
C
C     SUBROUTINE REALRT LOCATES ALL REAL ZERCES OF A REAL
C     POLYNOMIAL WHOSE DEGREE IS LESS THAN 30 AND GREATER THAN 1.
C
C     THE 'A' ARRAY  IS THE ARRAY CF COEFFICIENTS ARRANGED IN
C     ASCENDING CRDER.
C     N IS THE DEGREE OF THE POLYNOMIAL
C     THE ROCT VARIABLE IS THE ARRAY WHICH WILL CONTAIN THE REAL
C     RCOTS.
C     NI INDICATES THE NUMBER OF REAL ZEROES FOUND.
C
      DIMENSION A(1),RCOT(1),B(30)
      NI=0
      CALL BCUNE(A,N,BOUND,B)
      CALL CHECK(A,N,BOUND,IER)
      IF(IER.EQ.0)GO TO 120
      NI=IER
      CALL ROOE(A,B,N,BOUND,ROOT,NI)
  120 RETURN
C
      END
```

Subroutine BOUNE (A, N, BOUND, B)

Purpose: To determine an interval on the real axis that contains all the real zeros of $p(x)$

Flow chart:



Subroutine arguments:

A	array of coefficients for $p(x)$
N	degree of $p(x)$
BOUND	closed interval [-BOUND,BOUND] which contains all the real zeros of $p(x)$, an arbitrary polynomial
B	work array

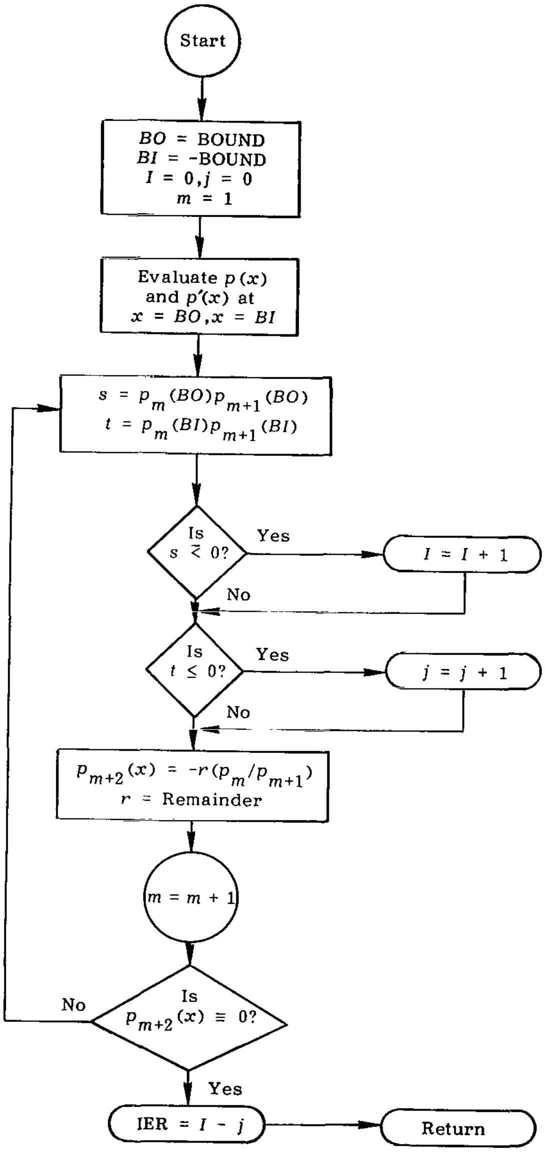
Subroutine listing:

```
            SUBROUTINE BOUNE(A,N,BOUND,B)
      C
      C     THIS ROUTINE DETERMINES THE INTERVAL CONTAINING THE ROCTS,
      C     IF ANY. (REFER TO CORRESPONINC "TNX".)
      C
            DIMENSION E(1),A(1)
            HHH=10.**6 ←  This is never used
            K=N+1
            DO 15 J=1,K
         15 B(J)=A(J)
      C
      C     FIND F(-X) AND SUBSEQUENTLY CCUNT THE NO. OF COEFFICIENTS
      C     PRECEDING THE FIRST NEG COEFFICIENT.
      C
      C     THEN DO LIKEWISE FOR F(X).
      C
            DO 16 J=2,K
            TL=(MOD(J,2))-.5
         16 B(J)=B(J)*(SIGN(1.,TL))
      C
      C     IF LEADING COEFFICIENT IS NEG,THEN CONSIDER -F(X).
      C
            IF(B(K).GT.0)GO TO 190
            DO 22 J=1,K-
         22 B(J)=-B(J)
        190 CONTINUE
        100 I=0
            DO 2000 J=1,K
            IF(B(K-J+1).LT.0)GO TO 1200
            I=I+1
       2000 CONTINUE
      C
      C     COMPUTE MAXIMUM NEG COEFFICIENT OF F(-X).
      C
       1200 R=0
            DO 1800 J=1,K
            IF(B(J).GE.0)GO TO 1800
            S=ABS(B(J))
            R=AMAX1(R,S)
       1800 CONTINUE
            W=1./I
            BT=1.+(R/B(K))**W
      C
      C     COMPUTE MAXIMUM NEG COEFFICIENT CF F(X)
      C
            T=0
            IF(A(K).GT.0)GO TO 10
            T=1
            DO 19 J=1,K
         19 A(J)=-A(J)
         10 I=0
            DO 20 J=1,K
            IF(A(K-J+1).LT.0)GO TO 12
            I=I+1
         20 CONTINUE
         12 R=0
            DO 180 J=1,K
            IF(A(J).GE.0)GO TO 180
            S=ABS(A(J))
            R=AMAX1(R,S)
        180 CONTINUE
            W=1./I
            BOUND=1.+((R/A(K))**W)
      C
      C     DETERMINE THE LARGER OF THE BOLNDS. THIS WILL BE THE INTERVAL
      C     CONTAINING ROOTS.
      C
            BOUND=AMAX1(BOUND,BT)
            IF(T.NE.1)GO TO 200
            DO 11 J=1,K
         11 A(J)=-A(J)
        200 RETURN
            END
```

Subroutine CHECK (A, N, BOUND, IER)

Purpose: To compute the number of distinct real zeros of $p(x)$

Flow chart:



Subroutine arguments:

A	array of coefficients for $p(x)$
N	degree of $p(x)$

BOUND closed interval [-BOUND,BOUND] which contains all the real zeros of $p(x)$, an arbitrary polynomial

IER total number of distinct real zeros of $p(x)$

Subroutine listing:

```
      SUBROUTINE CHECK(A,N,BOUND,IER)
C
C     THIS ROUTINE CONSTUCTS THE STURM SEQUENCE OF POLYNOMIALS
C     HAVING DETERMINED THIS SEQUENCE, IT THEN EVALUATES EACH
C     POLYNOMIAL AT BOUND AND -BOUND AND SUBSEQUENTLY COUNTS THE NUMBER OF
C     VARIATIONS OF SIGNS AT EACH EXTREMITY.  THE DIFFERENCE
C     OF THESE SUMS REPRESENTS THE NUMBER OF REAL DISTINCT ZEROES
C     OF P(X). (I.E. V(A)-V(B))
      DIMENSION A(1),B(30),C(30),D(30)
      IJ=0
      K=N+1
      NN=K-1
      XX=1./ABS(A(1))
      DO 22 JX=1,K
   22 D(JX)=A(JX)*XX
      FF=.1**10
      HH=10.**10
      DO 44 JX=1,K
      GG=ABS(D(JX))
      HH=AMIN1(HH,GG)
   44 CONTINUE
      EPS=HH/4
      EPS=AMIN1(EPS,FF)
      CALL PDER(B,NN,A,K)
      I=0
      J=0
      BO=BOUND
      BI=-BOUND
      CALL PVAL(S,BO,A,K)
      CALL PVAL(SO,BI,A,K)
      S=SIGN(1.,S)
      SC=SIGN(1.,SO)
   12 CALL PCIV(C,NI,D,K,B,NN,EPS,IER)
C
C     IF THERE IS NO REMAINDER, THEN COMPLETE ARGUMENT AND EXIT
C
      IF(K.LE.0)IJ=1
      CALL PVAL(T,BO,B,NN)
      CALL PVAL(TC,BI,B,NN)
      T=SIGN(1.,T)
      TO=SIGN(1.,TO)
      S=S*T
      SO=SO*TO
      IF(S.GT.0.)GO TO 20
      I=I+1
   20 IF(SO.GT.0.)GO TO 21
      J=J+1
   21 CONTINUE
      IF(IJ.EQ.1)GO TO 10
      SC=TO
      S=T
      DO 17 JX=1,NN
   17 C(JX)=B(JX)
      DO 18 JX=1,K
   18 B(JX)=-D(JX)
      DO 19 JX=1,NN
   19 D(JX)=C(JX)
      L=K
      K=NN
      NN=L
      GO TO 12
   10 NG=J-I
      IER=NG
      RETURN
      END
```

Subroutine PDER (Y, IDIMY, X, IDIMX)

Purpose: To find the derivative of a polynomial

Subroutine arguments:

Y array of coefficients, ordered from smallest to largest power for the derivative
IDIMY dimension of Y
X array of coefficients for the original polynomial
IDIMX dimension of X

Subroutine listing:

```
      SUBROUTINE PDER(Y,IDIMY,X,IDIMX)
      DIMENSION X(1),Y(1)
      IF(IDIMX-1)3,3,1
    1 EXPT=0
      DO 2 I=1,IDIMY
      EXPT=EXPT+1.
    2 Y(I)=X(I+1)*EXPT
      GO TO 4
    3 IDIMY=0
    4 RETURN
      END
```

Subroutine PVAL (RES, ARG, X, IDIMX)

Purpose: To evaluate a given polynomial at any given value using nested arithmetic

Subroutine arguments:

RES resultant value (*i.e.*, P(ARG))
ARG given value of the independent variable
X vector of coefficients, ordered from smallest to largest
IDIMX degree + 1

Subroutine listing:

```
      SUBROUTINE PVAL(RES,ARG,X,IDIMX)
C
C     THIS SUBROUTINE EVALUATES A GIVEN POLYNOMIAL AT
C     ANY VALUE USING NESTED ARITHMETIC
C
C
      DIMENSION X(1)
      RES=0
      J=IDIMX
    1 IF(J)3,3,2
    2 RES=RES*ARG+X(J)
      J=J-1
      GO TO 1
    3 RETURN
      END
```

Subroutine PDIV (P, IDIMP, X, IDIMX, Y, IDIM~~R~~Y, TOL, IER)

Purpose: To divide two polynomials

Subroutine arguments:

P resultant vector of integral part
IDIMP dimension of P
X vector of coefficients for dividend polynomial, ordered from smallest to largest; replaced with remainder after division
IDIMX dimension of X
Y vector of coefficients of divisor polynomial ordered from smallest to largest
IDIMY dimension of Y
TOL tolerance value below which coefficients are eliminated
IER error code; 1 denotes zero divisor, 0 denotes normal

Subroutine listing:

```
      SUBROUTINE PDIV(P,IDIMP,X,IDIMX,Y,IDIMY,TOL,IER)
C     THIS ROUTINE DIVIDES TWO POLYNOMIALS RETURNING
C     THE QUOTIENT AND THE REMAINDER
      DIMENSION P(1),X(1),Y(1)
   10 IDIMP=IDIMX-IDIMY+1
      IF(IDIMP)20,30,60
   20 IDIMP=0
   30 IER=0
   40 RETURN
   50 IER=1
      GO TO 40
   60 IDIMX=IDIMY-1
      I=IDIMP
   70 II=I+IDIMX
      P(I)=X(II)/Y(IDIMY)
      DO 80 K=1,IDIMX
      J=K-1+I
   80 X(J)=X(J)-P(I)*Y(K)
      I=I-1
      IF(I)90,90,70
   90 CALL PNORM(X,IDIMX,TOL)
      GO TO 30
      END
```

Subroutine PNORM (X, IDIMX, EPS)

Purpose: To normalize coefficients of a polynomial

Subroutine arguments:

X array of coefficients for $p(x)$
IDIMX dimension of X; replaced by final dimension after normalization
EPS tolerance below which coefficient is eliminated

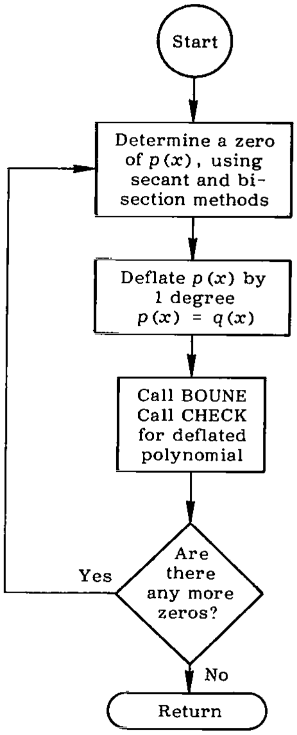
Subroutine listing:

```
      SUBROUTINE PNORM(X,IDIMX,EPS)
      DIMENSION X(1)
    1 IF(IDIMX)4,4,2
    2 IF(ABS(X(IDIMX))-EPS)3,3,4
    3 IDIMX=IDIMX-1
      GO TO 1
    4 RETURN
      END
```

Subroutine ROOE (A, B, N, BOUND, ROOT, NI)

Purpose: To compute all the real zeros of $p(x)$

Flow chart:


Start
Determine a zero of $p(x)$, using secant and bisection methods
Deflate $p(x)$ by 1 degree $p(x) = q(x)$
Call BOUNE
Call CHECK for deflated polynomial
Are there any more zeros?
Yes
No
Return

Subroutine arguments:

A	array of coefficients for $p(x)$
B	work array
N	degree of $p(x)$
BOUND	closed interval [-BOUND,BOUND] which contains all the real zeros of $p(x)$, an arbitrary polynomial
ROOT	array containing the real zeros
NI	number of real zeros in array root

Subroutine listing:

```
      SUBROUTINE ROOE(A,B,N.BOUND,RCCT,NI)
      DIMENSION B(1),A(1),ROOT(1)
      DIMENSION Y(30)
      REAL LX
C
C     THIS ROUTINE ACTUALLY COMPUTES THE REAL ROOTS.
C     E,DR,XL,ARE TOLERANCES UPON WHICH CONVERGENCE IS BASED;THEIR
C     ORDER INDICATING THEIR SEVERITY, MORE THAN ONE IS USED TO IMPRCVE
C     PROBABILITY OF CONVERGENCE WHEN ACCURACY IS LOST AND/OR PRESENCE
C     OF MULTIPLE ROOTS.
C
      JJ=N+10
      KB=NI
      E=.1**16
      GK=0
      0=0.
      XL=.1**6
      P=0
      Q=.1**10
      DR=Q
      NI=0
      LK=N+1
      LL=LK
      DO 190 J=1,LK
  190 B(J)=A(J)
      XX=1/B(1)
      VV=XX
      DO 444 J=1,LK
  444 B(J)=B(J)*XX
      I=0
  183 LJ=2
      IJ=KB-I
      ANS=0
      G=0
      IF(IJ.NE.1)GO TO 2566
      JH=ALOG10(BOUNC)/ALOG(2.)
      IM=JH/6
C
C     IF ONLY ONE DISTINCT ZERO REMAINS, THEN ISCLATE IT.
C
      DO 117 J=1,1000
      G=BOUND-(BCUND/(2.**(J*IM)))
      SS=BOUND-G
      IF(SS.LT.100.)GO TO 1000
      CALL CHECK(B,LK-1,G,IER)
      IF(IER.NE.0)GO TO 1000
  117 CONTINUE
 1000 CONTINUE
      G=BOUND-(BOUND/(2.**((J-1)*IM)))
 2566 CONTINUE
      X=BOUND-G
  182 CONTINUE
C
C     DETERMINES STEP SIZE,DEG.
C
      DEG=(2*X)/(2.**(LJ-1))
      LG=2*X/DEG
      T=0
  150 L=2
   22 LX=MOD(L,2)-1
C
C     COMPUTES INITIAL GUESSES WITH STEF SIZE,DEG.
C
      X0=SIGN(G,LX)-(SIGN(DEG,LX)*((L/2)-1))
      X1=SIGN(G,LX)-(SIGN(DEG,LX)*(L/2))
C
      CALL PVAL(Z,X1,B,LK)
      Z=ABS(Z)+10.
      CALL PVAL(R,X0,B,LK)
   20 CONTINUE
C
C     EVALUATES F(X) AT COMPUTED ITERATE
C
      CALL PVAL(S,X1,B,LK)
      IF(IJ.NE.1)GO TO 103
      II=KB-I
      IF(II.GT.1)GO TO 103
      F=SIGN(1.,S)
      H=SIGN(1.,R)
      IF(F*H.GT.C)GO TO 103
      XM=(S-R)/(X1-X0)
      CT=ABS(X1-X0)
      IF(CT.GT..5)GO TO 97
      IF(CT.GT..01*ABS(X1))GO TO 97
      XMM=S-XM*X1
      ANS=-XMM/XM
      GC TO 206
   97 CONTINUE
  103 CONTINUE
C
C     DETERMINES MINIMUM F(X) IF DIVERGENCE OCCURS FOR ALL ITERATES.
C
      IF(T.EQ.0)GO TO 122
      IF(ABS(S).GT.T)GC TO 10
  122 CCNTINUE
      T=ABS(S)
      G1=X1
   10 CONTINUE
C
C     IF F(X) CONVERGES , THEN ISOLATE ZERO AND REDUCE P(X)
C
      IF(ABS(S).LT.E)GO TO 125
      IF((ABS(S).GT.Z).AND.(ABS(S).LT.DR))GO TO 125
      IF(ABS(S).LT.XL)P=X1
C
C     CHECKING FOR DIVERGENCE
C
      IF(ABS(S).GT.Z)GO TO 120
      Z=ABS(S)
      V=X1
      IF(S-R.EQ.0)GO TO 120
      FG=(X1-X0)/(S-R)
      X1=X1-(FG*S)
      IF(ABS(X1).GT.BOUND)X1=SIGN(BOUNC,X1)
      IF(ABS(X1).LT.G)X1=SIGN(G,X1)
      XO=V
      R=S
      GO TO 20
  125 I=I+1
      ANS=0
      P=0
      NI=I
   23 CONTINUE
      ROOT(I)=X1
      IF(I.EQ.N)GO TO 206
      IF(GK.NE.0.)GO TO 105
C
C     COMPUTES DEFLATED POLYNOMIAL,USING SYNTHETIC DIVISION, DETERMINE
C     NEW INTERVAL.
C
  550 CONTINUE
      CALL SYNTH(B,LK,X1)
      IF(LK-1.EQ.2)GO TO 100
      LK=LK-1
C
C     DETERMINE BOUND FOR ZERO AND HENCE THE NO. OF DISTINCT ZEROES
C     FOR THE RECUCED FOLYNOMIAL
C
      CALL BOUNE(B,LK-1,BOUND,Y)
      CALL CHECK(B,LK-1,BOUND,IER)
C
C     INTERRCGATES THE STATUS OF CCMPLETION.
C
      IF(I.LT.KB)GO TO 2300
      IF((IER.EQ.0).AND.(I.GE.KB))GO TO 206
 2300 CONTINUE
      VV=1/B(1)
      DO 355 J=1,LK
  355 B(J)=B(J)*VV
  699 CONTINUE
      GO TO 183
  100 X1=-B(1)/B(2)
      GO TO 125
  120 L=L+1
C
C     COMPUTES THE NEXT INITIAL GUESS IF INTERVAL IS NOT EXHAUSTED.
C
      IF(L.LT.LG)GO TO 22
      IX=IX+1
      IF(P.EQ.0)GO TO 1212
      X1=P
      GO TO 125
 1212 CONTINUE
C
C     IF STEP SIZE IS SMALL ENOUGH,SUBJECT POLYNOMIAL TO BISECTICN,
C     OTHERWISE,CHOOSE A SMALLER  STEP SIZE,ETC.
C
      IF(DEG.LT..1)GO TO 1556
      LJ=LJ+1
      GO TO 182
 1556 CONTINUE
C
C     BEGIN  BISECTION ARGUMENT BASED ON G1 AND DEG.
C
      X1=G1
      D=DEG
      JH=ALOG10(DEG)/ALOG10(2.)
      JH=JH+18
      DO 18 J=1,JH
      XC=X1+D
      XG=X1-D
      CALL PVAL(S,XC,A,LL)
      S=ABS(S)
      CALL PVAL(T,XG,A,LL)
      T=ABS(T)
      X1=X1-(SIGN(D/2,S-T))
      CALL PVAL(H,X1,A,LL)
      H=ABS(H)
      IF(H.LT.XL)GO TO 125
      D=D/2
   18 CCNTINUE
  206 CONTINUE
      IF(ANS.EQ.0)GO TO 105
      X1=ANS
      GK=1
      GO TO 125
  105 CCNTINUE
      RETURN
      END
```

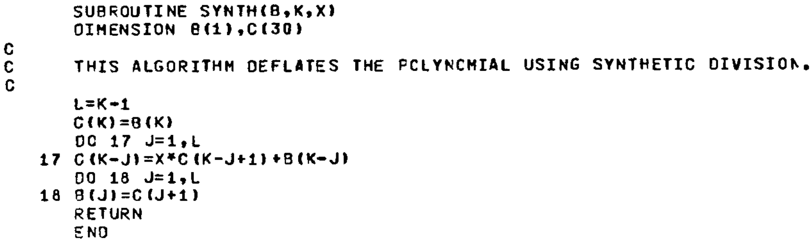
Subroutine SYNTH (B, K, X)

Purpose: To deflate a polynomial by one degree using synthetic division

Subroutine arguments:

B	array of coefficients for a polynomial, $p(x)$
K	degree + 1
X	zero of $p(x)$

Subroutine listing:



```
          SUBROUTINE SYNTH(B,K,X)
          DIMENSION B(1),C(30)
      C
      C   THIS ALGORITHM DEFLATES THE PCLYNCMIAL USING SYNTHETIC DIVISION.
      C
          L=K-1
          C(K)=B(K)
          DO 17 J=1,L
       17 C(K-J)=X*C(K-J+1)+B(K-J)
          DO 18 J=1,L
       18 B(J)=C(J+1)
          RETURN
          END
```

REFERENCES

1. Hamming, R. W.: Numerical Methods for Scientists and Engineers. Second ed. International Series in Pure and Applied Mathematics, McGraw-Hill Book Co., Inc., c.1973, p. 59.

2. Ralston, Anthony: A First Course in Numerical Analysis. International Series in Pure and Applied Mathematics, McGraw-Hill Book Co., Inc., c.1965, pp. 318-381.

3. Dickson, Leonard Eugene: New First Course in the Theory of Equations. John Wiley & Sons, Inc., 1962, pp. 23-25.

NASA-Langley, 1975 H-855

> *"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."*
>
> —NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons. Also includes conference proceedings with either limited or unlimited distribution.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include final reports of major projects, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546