AN ABSTRACT OF THE THESIS OF

Leesa Kaye Pohl for the Master of Science

in Mathematics presented in July 1982

Title: Limitations of Solving Polynomial Equations on the Microcomputer

Abstract approved: John W. Carlson

The purpose of this thesis is to point out some of the problems which may occur while attempting to solve a polynomial equation on the microcomputer.

Specifically, programs are given which will solve polynomial equations of degree four or less using the formulas. Since it is not possible to solve a polynomial equation of degree five or more using a formula, programs are also given for Newton's, the secant, and the bisection methods.

Solutions obtained by using these programs are given. Illustrations of some of the things which may cause problems are also given. Specifically, these include multiple roots, reducing the polynomial, and the order in which the roots are found. Problems encountered in getting the program itself to work are also discussed.

LIMITATIONS OF SOLVING POLYNOMIAL EQUATIONS ON THE MICROCOMPUTER

A Thesis

Presented to

the Department of Mathematics

Emporia State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Leesa Kaye Pohl

July 1982

John W. Carley
Approved for the Major Department

Harold E. Durst
Approved for the Graduate Council

ACKNOWLEDGMENT

I would like to thank Dr. John W. Carlson for his assistance on this thesis. I would also like to thank the members of the mathematics faculty for their understanding and guidance during my studies.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Polynomial equations and their roots have been of interest to mathematicians for centuries. Through the years, many diverse methods for finding the roots of polynomial equations have been developed. While newer methods are constantly being sought, older methods are also being adapted to take advantage of the latest tools acquired through the technological advances of society.

One of the latest tools to be thus acquired is the microcomputer. The microcomputer can make calculations exceedingly fast. For this reason it is especially useful in those methods which require many complicated or repetitive calculations.

However, the microcomputer is not without disadvantages. Before it can be used to find the roots of a polynomial equation, a program must be written. It must also be checked as to the accuracy of the solutions obtained thereby. But this is only the beginning of the problems which may be encountered when working with the microcomputer.

Representation error will occur whenever a repeating decimal or irrational number is used. Round-off will also occur frequently in any method used on the microcomputer. Additional representation error will be incurred by the microcomputer when the machine changes the base ten number entered and displayed to the base two number it performs the calculations with, and back again. This error will be especially noticeable when working with decimals.

However, before discussing any specific method for solving polynomial equations, a brief review of several fundamental concepts essential for any method used to find the roots will be outlined.

The basis for solving any polynomial equation is the Fundamental Theorem of Algebra. This theorem was first proven by Gauss, circa 1800 (for a proof see [8, p. 414]). The Fundamental Theorem of Algebra states that every polynomial equation, $P(x) = 0$, of degree $n \geq 1$ has at least one root. While this theorem does not solve the equation, it does guarantee that the polynomial equation has a root.

Using the Fundamental Theorem, it can easily be shown that a polynomial equation of degree n will have exactly n roots. By the Fundamental Theorem, the polynomial equation will have at least one root. Let this root be r_1. If r_1 is a root, then $x-r_1$ is a factor of the polynomial. A reduced equation may be obtained by dividing the polynomial by $x-r_1$. Then, according to the Fundamental Theorem, this reduced equation must also have at least one root. By repeating the above process, the existence of n roots can be shown.

Polynomial equations of degree four or less may be solved by using a formula. However, no formula exists for solving a polynomial equation of degree five or more. Furthermore, it is impossible to find a general formula for solving these polynomial equations. This conjecture was finally proven by Abel in 1824, [3, p. 555].

The above fundamental concepts, then, are the basis for solving polynomial equations of degree five or more. Many methods are available for use in solving these equations. Newton's method, the bisection method, and the secant method will be discussed in this thesis. These methods are all iterative in nature. Thus, the accuracy of the answer is highly dependent upon the accuracy of the machine. While some methods are self-correcting, there is a limit to how much error can be compensated for. A small error introduced at the beginning, such as entering one of the coefficients incorrectly, or early in the procedure, such as reducing an equation by an inaccurate representation of a root, may greatly affect the roots of the polynomial equation. It may, in fact, change the equation so much that the roots of the original equation and of the inaccurate equation are entirely different.

The purpose of this thesis, then, is not to offer methods for solving polynomial equations, but to point out some of the problems which may occur when adapting commonly used methods to the microcomputer. By knowing where to look for error, it is hoped the reader will be more conscientious when using any method on the microcomputer. Some suggestions for circumventing these problems will also be offered.

Programs written in conjunction with this paper are in Applesoft, and are designed to execute on an Apple II Plus microcomputer. Some minor modifications may be necessary for them to execute on a different microcomputer.

For the purposes of this thesis, a polynomial will be denoted as:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 \qquad (a_n \neq 0)$$

where the coefficients are known real numbers, and n is the degree of the equation.

CHAPTER II

POLYNOMIAL EQUATIONS OF DEGREE ≤ 4

Polynomial equations of degree one are also called linear equations. These equations have the general form of $a_1x + a_0 = 0$, and are trivial to solve. Program 2.1 will solve equations of this type.

<u>PROGRAM 2.1</u>

```
10  REM  *** DEGREE 1 ***
20  HOME
30  PRINT "THIS PROGRAM WILL FIND THE ROOT OF A"
40  PRINT "POLYNOMIAL EQUATION OF DEGREE 1"
50  PRINT
60  PRINT "THE GENERAL FORM OF THIS TYPE OF"
70  PRINT "EQUATION IS A1X + A0 = 0"
80  PRINT
90  PRINT "****************************************"
100  PRINT
110  INPUT "ENTER A1 ";A1
120  INPUT "ENTER A0 ";A0
130  HOME
140 R =  - A0 / A1
150  PRINT "THE ROOT OF THE EQUATION "A1"X + "A0" = 0
     IS "
160  PRINT R
170  END
```

Polynomial equations of degree two are more commonly referred to as quadratic equations. These equations have the general form of $a_2x^2 + a_1x + a_0 = 0$. Quadratic equations are routinely solved by using the quadratic formula, which is as follows:

$$x = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2a_0}}{2a_2}$$

There are, however, two forms of the quadratic formula. Form 1 is shown above. Form 2 is obtained from Form 1 by multiplying the numerator and the denominator of Form 1 by the conjugate of the numerator. Form 2 is given below:

$$x = \frac{2a_0}{-a_1 \pm \sqrt{a_1^2 - 4a_2a_0}}$$

The two forms of the quadratic formula, therefore, are equivalent and should yield the same roots. Unfortunately, this is not the case, as is illustrated by Table 2.1.

TABLE 2.1

Equation	Form 1	Form 2
$x^2 + 40x + 400$	-20 -20	-20 -20
$x^2 + 5x - 1000$	29.2214439 -34.2214439	29.2214439 -34.2214438
$x^2 + 400x + 400$	-1.00251248 -398.997488	-1.00251258 -398.997526
$10x^2 - 100x + 10$	9.89897949 0.101020513	9.89897959 0.101020514
$x^2 + 500x + 500$	-1.00200787 -498.997992	-1.00200804 -498.998078
$x^2 + 1000x + 1$	-9.99701675 E-4 -999.999	-1.000001 E-3 -1000.29841
$x^2 + 10000x + 1$	-9.78186727 E-5 -9999.9999	-1.00000001 E-4 -10222.997
$x^2 + 10000x - 1$	1.0189414 E-4 -10000.0001	9.9999999 E-5 -9814.10705
$x^2 + 1000x + .001$	-7.23637641 E-7 -1000	-1 E-6 -1381.90711
$(1.E-5)x^2 + (1.E+6)x + .025$	51.0215759 -1 E+11	48.9988785 -2.5 E-8

The roots generated from the last equation are not even close to being the same, especially when one notices the positive 11 exponent obtained by Form 1 and the negative 8 exponent obtained by Form 2.

It is therefore necessary to know which of the two methods will yield the more accurate answer. Table 2.2 lists the actual roots, and the roots obtained by using both Form 1 and Form 2.

TABLE 2.2

Actual Roots	Form 1	Form 2
.5, -1000	.500000267, -1000	.5, -999.999467
-.5, -1000	-.49999969, -1000	-.5, -1000.00062
-.01, -10000	-9.99775529 E-3, -10000	-.01, -10002.2452
-1000, -1000	-1000, -1000	-1000, -1000
-1 E-5, -1 E+5	5.23924828 E-5, -1 E+5	-1 E-5, 19086.7076
-1 E-6, -1 E+6	5.102157596 E-4, -1 E+6	-1 E-6, 1959.95514
10000, -1 E-4	10000, -1.01752579 E-4	9827.76071, -1 E-4
1000, 1	1000, .999999654	1000.00035, 1
1000, -.5	1000, -.500000267	999.997467, -.5

Both forms appear to yield results with approximately the same degree of accuracy. However, it can be observed that each method will yield a more accurate result for one root than for the other root. It can also be observed that the more accurate root for Form 1 is not the more accurate root for Form 2, and vice versa. Further investigation reveals the more accurate root is obtained when $-a_1$ and $\sqrt{a_1^2 - 4a_2a_0}$ have the same sign. Thus, by checking the sign of a_1, part of each form may be used to improve the accuracy of the answer. In the case of complex roots, only Form 1 is used since it is much easier to work with. Program 2.2 will calculate the roots of a quadratic equation. Sample output from this program is given in Table 2.3.

PROGRAM 2.2

```
10  REM  *** DEGREE 2 ***
20  HOME
30  PRINT "THIS PROGRAM WILL FIND THE ROOTS OF A"
40  PRINT "POLYNOMIAL EQUATION OF DEGREE 2"
50  PRINT
60  PRINT "THE GENERAL FORM OF THIS TYPE OF"
70  PRINT "EQUATION IS A2X^2 + A1X + A0 = 0"
80  PRINT
90  PRINT "****************************************"
100  PRINT
110  INPUT "ENTER A2 ";A2
120  INPUT "ENTER A1 ";A1
130  INPUT "ENTER A0 ";A0
140  HOME
150  REM  *** EVALUATE THE DISCRIMINATE
160 D = A1 * A1 - 4 * A2 * A0
170  IF D < 0 THEN 340
180  REM  *** CALCULATE REAL ROOTS
190 D =  SQR (D)
200  REM  *** CHECK IF A1 IS POSITIVE OR NEGATIVE
210  IF A1 > 0 THEN 270
220  REM  *** A1 IS NEGATIVE
230 R1 = ( - A1 + D) / (2 * A2)
240 R2 = 2 * A0 / ( - A1 + D)
250  GOTO 290
260  REM  *** A1 IS POSITIVE
270 R1 = ( - A1 - D) / (2 * A2)
280 R2 = 2 * A0 / ( - A1 - D)
290  PRINT "THE ROOTS OF THE EQUATION "
300  PRINT A2"X^2 + "A1"X + "A0" = 0 ARE"
310  PRINT R1;" AND ";R2
320  GOTO 420
330  REM  *** CALCULATE COMPLEX ROOTS
340 D =  SQR ( - D)
350 R3 =  - A1 / (2 * A2)
360 R4 = D / (2 * A2)
370  PRINT "THERE ARE NO REAL ROOTS TO THE EQUATION"
380  PRINT A2"X^2 + "A1"X + "A0" = 0"
390  PRINT
400  PRINT "THE COMPLEX ROOTS ARE ";R3;" + ";R4;"I"
410  PRINT "AND ";R3;" - ";R4;"I"
420  END
```

TABLE 2.3

Equation	Roots	
$x^2 + 2x + 1$	-1	-1
$x^2 + 10000x + 1$	-9999.9999	-1.00000001 E-4
$x^2 + 5x - 1000$	-34.2214439	29.2214439
$x^2 + 100000x - 1$	-100000	1 E-5
$x^2 + 1000x + 0.001$	-1000	-1 E-6
$(1.E-5)x^2 + (1.E+6)x + .025$	-1 E+11	-2.5 E-8

No improvement was obtained in the case of the last equation listed in Table 2.3. However, it should be noted that neither Form 1 nor Form 2 yielded satisfactory answers when solving this equation. Nor will satisfactory results be obtained whenever $a_1^2 >> 4a_2a_0$, due to round-off error in the machine.

Polynomial equations of degree three are also called cubic equations. These equations have the general form of $a_3x^3 + a_2x^2 + a_1x + a_0 = 0$. Cubic equations can be solved by using the cubic formula. This formula is not as well known as the quadratic formula. The three roots can be obtained by substituting the values for $y = A + B$, $y = -\frac{A+B}{2} + \frac{A-B}{2}\sqrt{-3}$, and $y = -\frac{A+B}{2} - \frac{A-B}{2}\sqrt{-3}$ into $x = y - \frac{a_2}{3a_3}$. In order to solve for y, let $A = \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$ and $B = \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$ where $p = \frac{a_1}{a_3} - \frac{a_2^2}{3a_3^2}$ and $q = \frac{2a_2^3}{27a_3^3} - \frac{a_2a_1}{3a_3^2} + \frac{a_0}{a_3}$.

A development of these formulas may be found in [2, pp. 115-127].

Program 2.3 will find the roots of a cubic equation. Table 2.4 follows the program, and summarizes some sample output.

PROGRAM 2.3

```
10  REM  *** DEGREE 3 ***
20  HOME
30  PRINT "THIS PROGRAM WILL FIND THE ROOTS OF A"
40  PRINT "POLYNOMIAL EQUATION OF DEGREE 3"
50  PRINT
60  PRINT "THE GENERAL FORM OF THIS TYPE OF "
70  PRINT "EQUATION IS A3X^3 + A2X^2 + A1X + A0 = 0"
80  PRINT
90  PRINT "****************************************"
100  PRINT
110  INPUT "ENTER A3 ";A3
120  INPUT "ENTER A2 ";A2
130  INPUT "ENTER A1 ";A1
140  INPUT "ENTER A0 ";A0
150  HOME
160 P = A1 / A3 - (A2 * A2) / (3 * A3 * A3)
170 Q = 2 * A2 * A2 * A2 / (27 * A3 * A3 * A3) - A2 *
     A1 / (3 * A3 * A3) + A0 / A3
180  REM  *** CALCULATE THE DISCRIMINATE
190 D = Q * Q / 4 + P * P * P / 27
200  IF  ABS (D) < 1E - 10 THEN D = 0
210  REM  *** CHECK FOR 3 REAL,2 EQUAL,OR 2 COMPLEX RO
     OTS
220  ON ( SGN (D) + 2) GOTO 250,550,780
230  REM  *** 3 REAL UNEQUAL ROOTS
240  REM  *** D<0
250  REM  *** COMPUTE SQUARE ROOT OF DISCRIMINATE
260 DS =  SQR ( - D)
270  REM  *** COMPUTE A AND B
280  REM  *** FIRST CHECK IF Q=0
290  IF Q <  > 0 THEN 390
300 A = DS ^ (1 / 3)
310 B =  - A
320  REM  *** CALCULATE THREE Y'S
330 Y1 = 0
340 Y2 = (A - B) *  SQR (3) / 2
350 Y3 =  - Y2
360  GOTO 650
370  REM  *** Q<>0
380  REM  *** CALCULATE A AND B USING DE MOIVRE'S THEO
     REM
390 Q2 =  - Q / 2
400 Q3 =  SQR (Q2 * Q2 + DS * DS)
410 C = Q2 / Q3
420 T =  -  ATN (C /  SQR ( - C * C + 1)) + 1.5708
430 C0 = T / 3
440 C3 = C0 + 2.09439513
```

PROGRAM 2.3 (continued)

```
450 C7 = C0 + 4.18879026
460 Q3 = Q3 ^ (1 / 3)
470  REM  *** CALCULATE 3 Y'S
480 Y1 = 2 * Q3 *  COS (C0)
490 Y2 = 2 * Q3 *  COS (C3)
500 Y3 = 2 * Q3 *  COS (C7)
510  GOTO 650
520  REM  *** 2 EQUAL ROOTS
530  REM  *** D=0
540  REM  *** CALCULATE A AND B
550 Q2 =  - Q / 2
560 Q3 =  ABS (Q2)
570 A = Q3 ^ (1 / 3)
580  IF Q2 =  - Q3 THEN A =  - A
590 B = A
600  REM  *** CALCULATE 3 Y'S
610 Y1 = A + B
620 Y2 =  - (A + B) / 2
630 Y3 = Y2
640  REM  *** CALCULATE THREE REAL ROOTS
650 R1 = Y1 - A2 / (3 * A3)
660 R2 = Y2 - A2 / (3 * A3)
670 R3 = Y3 - A2 / (3 * A3)
680  REM  *** PRINT RESULTS
690  PRINT "THE ROOTS OF THE EQUATION"
700  PRINT
710  PRINT A3"X^3 + "A2"X^2 + "A1"X + "A0" = 0"
720  PRINT
730  PRINT "ARE ";R1;", ";R2;", AND ";R3
740  GOTO 1020
750  REM  *** 2 COMPLEX ROOTS
760  REM  *** D>0
770  REM  *** CALCULATE A AND B
780 DS =  SQR (D)
790 Q2 =  - Q / 2 + DS
800 Q4 =  - Q / 2 - DS
810 Q3 =  ABS (Q2)
820 Q5 =  ABS (Q4)
830 A = Q3 ^ (1 / 3)
840 B = Q5 ^ (1 / 3)
850  IF Q2 =  - Q3 THEN A =  - A
860  IF Q4 =  - Q5 THEN B =  - B
870  REM  *** CALCULATE 3 Y'S
880 Y1 = A + B
890 Y2 =  - (A + B) / 2
900 Y3 = (A - B) *  SQR (3) / 2
910  REM  *** CALCULATE REAL PART
920 R1 = Y1 - A2 / (3 * A3)
930 R2 = Y2 - A2 / (3 * A3)
940  REM  *** PRINT ANSWERS
950  PRINT "THE EQUATION "A3"X^3 + "A2"X^2 + "A1"X + "
     A0" = 0"
```

PROGRAM 2.3 (continued)

```
960  PRINT "HAS TWO IMAGINARY ROOTS"
970  PRINT
980  PRINT "ITS ONE REAL ROOT IS ";R1
990  PRINT
1000  PRINT "ITS TWO COMPLEX ROOTS ARE"
1010  PRINT R2" + "Y3"I AND "R2" - "Y3"I"
1020  END
```

TABLE 2.4

Equation	Actual Roots	Computed Roots
$x^3 + 4x^2 + 5x + 2$	-2	-2
	-1	-0.999999999
	-1	-0.999999999
$x^3 + 6x^2 + 11x + 6$	-2	-2
	-1	-1
	-3	-3
$x^3 + 6x^2 + 12x + 8$	-2	-2
	-2	-2
	-2	-2
$x^3 + 2x^2 - 5x - 6$	-2	-1.99999859
	-3	-3.00000217
	-1	-0.999996304
$x^3 - 2x^2 - x + 2$	2	1.99999859
	-1	-1.00000072
	1	1.00000222
$x^3 + x^2 + x + 1$	-1	-1
	i	i
	-i	-i

Polynomial equations of degree four are also called quartic, or biquadratic, equations. These equations have the general form of $a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 = 0$. Quartic equations may be solved by using the quartic formula. The four roots of the quartic equation may be found from the following equations: $x = -\frac{a_3}{4} + \frac{R}{2} \pm \frac{D}{2}$ and $x = -\frac{a_3}{4} - \frac{R}{2} \pm \frac{E}{2}$ where $R = \sqrt{\frac{a_3^2}{4} - a_2 + y}$. If $R = 0$ then $D = \sqrt{\frac{3a_3^2}{4} - 2a_2 + 2\sqrt{y^2 - 4a_0}}$ and $E = \sqrt{\frac{3a_3^2}{4} - 2a_2 - 2\sqrt{y^2 - 4a_0}}$.

If $R \neq 0$ then $D = \sqrt{\frac{3a_3^2}{4} - R^2 - 2a_2 + \frac{4a_3a_2 - 8a_1 - a_3^3}{4R}}$ and

$E = \sqrt{\frac{3a_3^2}{4} - R^2 - 2a_2 - \frac{4a_3a_2 - 8a_1 - a_3^3}{4R}}$. In both cases, y is any root of the following resolvent cubic equation.

$$y^3 - a_2y^2 + (a_3a_1 - 4a_0)y - a_3^2a_0 + 4a_2a_0 - a_1^2 = 0$$

A development of these formulas may be found in [2, pp. 128-131].

Program 2.4 will find the roots of a quartic equation. Table 2.5 follows the program, and summarizes some sample output.

PROGRAM 2.4

```
10  REM  *** DEGREE 4 ***
20  HOME
30  PRINT "THIS PROGRAM WILL FIND THE ROOTS OF A"
40  PRINT "POLYNOMIAL EQUATION OF DEGREE 4"
50  PRINT
60  PRINT "THE GENERAL FORM OF THIS TYPE "
70  PRINT "OF EQUATION IS"
80  PRINT "A4X^4 + A3X^3 + A2X^2 + A1X + A0 = 0"
90  PRINT
100  PRINT "***************************************"
110  PRINT
120  INPUT "ENTER A4 ";A
130  INPUT "ENTER A3 ";B
```

PROGRAM 2.4 (continued)

```
140  INPUT "ENTER A2 ";C
150  INPUT "ENTER A1 ";D
160  INPUT "ENTER A0 ";E
170  HOME
180 A9 = B / A
190 B9 = C / A
200 C9 = D / A
210 D9 = E / A
220  REM  *** CALCULATE P,Q,R OF RESOLVENT CUBIC
230 P =  - B9
240 Q = A9 * C9 - 4 * D9
250 R =  - A9 * A9 * D9 + 4 * B9 * D9 - C9 * C9
260  REM  *** SOLVE RESOLVENT CUBIC
270 A1 = Q - P * P / 3
280 B1 = (2 * P * P * P - 9 * P * Q + 27 * R) / 27
290 D1 = B1 * B1 / 4 + A1 * A1 * A1 / 27
300  IF  ABS (D1) < 1E - 10 THEN D1 = 0
310 X =  SGN (D1) + 2
320  ON X GOTO 330,480,550
330  IF B1 <  > 0 THEN 390
340 A3 =  SQR ( - D1)
350 A2 = A3 ^ (1 / 3)
360 B2 =  - A2
370 X1 = 0
380  GOTO 650
390 A3 =  - B1 / 2
400 B3 =  SQR ( - D1)
410 R4 =  SQR (A3 * A3 + B3 * B3)
420 C1 = A3 / R4
430 T =  -  ATN (C1 /  SQR ( - C1 * C1 + 1)) + 1.5708
440 C0 = T / 3
450 R4 = R4 ^ (1 / 3)
460 X1 = 2 * R4 *  COS (C0)
470  GOTO 650
480 A3 =  - B1 / 2
490 A4 =  ABS (A3)
500 A2 = A4 ^ (1 / 3)
510  IF A4 =  - A3 THEN A2 =  - A2
520 B2 = A2
530 X1 = A2 + B2
540  GOTO 650
550 A3 =  - B1 / 2 +  SQR (D1)
560 B3 =  - B1 / 2 -  SQR (D1)
570 A4 =  ABS (A3)
580 A2 = A4 ^ (1 / 3)
590  IF A3 =  - A4 THEN A2 =  - A2
600 B4 =  ABS (B3)
610 B2 = B4 ^ (1 / 3)
620  IF B3 =  - B4 THEN B2 =  - B2
630 X1 = A2 + B2
640  REM  *** Y IS ROOT OF RESOLVENT CUBIC
650 Y = X1 - P / 3
```

PROGRAM 2.4 (continued)

```
660  REM  *** CALCULATE VALUE OF R
670 R9 = A9 * A9 / 4 - B9 + Y
680  IF  ABS (R9) < 1E - 5 THEN R9 = 0
690 R9 =  SQR (R9)
700  REM  *** PRINT HEADINGS
710  PRINT "THE ROOTS OF THE EQUATION"
720  PRINT A;"X^4 + ";B;"X^3 + ";C;"X^2 + ";D;"X + ";E
     ;" = 0"
730  PRINT "ARE ";
740  REM  *** CHECK IF R=0
750  IF R9 = 0 THEN 1080
760  REM  *** R<>0
770  REM  *** CALCULATE D AND E
780 D1 = 3 * A9 * A9 / 4 - R9 * R9 - 2 * B9 + (4 * A9 *
     B9 - 8 * C9 - A9 * A9 * A9) / (4 * R9)
790 E1 = 3 * A9 * A9 / 4 - R9 * R9 - 2 * B9 - (4 * A9 *
     B9 - 8 * C9 - A9 * A9 * A9) / (4 * R9)
800  IF D1 < 0 THEN 880
810 D1 =  SQR (D1)
820  REM  *** CALCULATE 2 REAL ROOTS
830 R1 =  - A9 / 4 + R9 / 2 + D1 / 2
840 R2 =  - A9 / 4 + R9 / 2 - D1 / 2
850  PRINT R1;", ";R2;","
860  GOTO 930
870  REM  *** CALCULATE 2 COMPLEX ROOTS
880 D1 =  SQR ( - D1)
890 R1 =  - A9 / 4 + R9 / 2
900 R2 = D1 / 2
910  PRINT R1;" + ";R2;"I,"
920  PRINT R1;" - ";R2;"I,"
930  IF E1 < 0 THEN 1010
940 E1 =  SQR (E1)
950  REM  *** CALCULATE 2 MORE REAL ROOTS
960 R3 =  - A9 / 4 - R9 / 2 + E1 / 2
970 R4 =  - A9 / 4 - R9 / 2 - E1 / 2
980  PRINT R3;", AND ";R4
990  GOTO 1620
1000  REM  *** CALCULATE 2 MORE COMPLEX ROOTS
1010 E1 =  SQR ( - E1)
1020 R3 =  - A9 / 4 - R9 / 2
1030 R4 = E1 / 2
1040  PRINT R3;" + ";R4;"I, AND"
1050  PRINT R3;" - ";R4;"I"
1060  GOTO 1620
1070  REM  *** R=0
1080  IF Y * Y - 4 * D9 < 0 THEN 1410
1090  REM  *** CALCULATE D AND E
1100 D1 = 3 * A9 * A9 / 4 - 2 * B9 + 2 *  SQR (Y * Y -
     4 * D9)
1110 E1 = 3 * A9 * A9 / 4 - 2 * B9 - 2 *  SQR (Y * Y -
     4 * D9)
1120  IF D1 < 0 THEN 1200
```

PROGRAM 2.4 (continued)

```
1130  REM  *** CALCULATE 2 REAL ROOTS
1140 D1 =  SQR (D1)
1150 R1 =  - A9 / 4 + D1 / 2
1160 R2 =  - A9 / 4 - D1 / 2
1170  PRINT R1;", ";R2
1180  GOTO 1250
1190  REM  *** CALCULATE 2 COMPLEX ROOTS
1200 D1 =  SQR ( - D1)
1210 R1 =  - A9 / 4
1220 R2 = D1 / 2
1230  PRINT R1;" + ";R2;"I,"
1240  PRINT R1;" - ";R2;"I,"
1250  IF E1 < 0 THEN 1330
1260  REM  *** CALCULATE 2 MORE REAL ROOTS
1270 E1 =  SQR (E1)
1280 R3 =  - A9 / 4 + E1 / 2
1290 R4 =  - A9 / 4 - E1 / 2
1300  PRINT R3;", AND ";R4
1310  GOTO 1620
1320  REM  *** CALCULATE 2 MORE COMPLEX ROOTS
1330 E1 =  SQR ( - E1)
1340 R3 =  - A9 / 4
1350 R4 = E1 / 2
1360  PRINT R3;" + ";R4;"I, AND"
1370  PRINT R3;" - ";R4;"I"
1380  GOTO 1620
1390  REM  *** CALCULATE 4 COMPLEX ROOTS
1400  REM  *** USE DE MOIVRE'S THEOREM
1410 D1 = 3 * A9 * A9 / 4 - 2 * B9
1420 E1 = 2 *  SQR ( - Y * Y + 4 * D9)
1430 R5 =  SQR (D1 * D1 + E1 * E1)
1440 C1 = D1 / R5
1450 T1 =  -  ATN (C1 /  SQR ( - C1 * C1 + 1)) + 1.570
     8
1460 T2 = 6.2831853 - T1
1470 C2 = T1 / 2
1480 C3 = T2 / 2
1490 R5 =  SQR (R5)
1500 D1 = R5 *  COS (C2)
1510 D2 = R5 *  SIN (C2)
1520 E1 = R5 *  COS (C3)
1530 E2 = R5 *  SIN (C3)
1540 R1 =  - A9 / 4 + D1 / 2
1550 R2 =  - A9 / 4 - D1 / 2
1560 R3 =  - A9 / 4 + E1 / 2
1570 R4 =  - A9 / 4 - E1 / 2
1580  PRINT R1;" + ";D2 / 2;"I,"
1590  PRINT R2;" - ";D2 / 2;"I,"
1600  PRINT R3;" + ";E2 / 2;"I, AND"
1610  PRINT R4;" - ";E2 / 2;"I"
1620  END
```

TABLE 2.5

Equation	Actual Roots	Computed Roots
$x^4 + 4x^3 + 6x^2 + 4x + 1$	-1	-1
	-1	-1
	-1	-1
	-1	-1
$x^4 + 8x^3 + 24x^2 + 32x + 16$	-2	-2
	-2	-2
	-2	-2
	-2	-2
$x^4 + 5x^3 + 9x^2 + 7x + 2$	-1	-1
	-1	-1
	-1	-1
	-2	-2
$x^4 + 10x^3 + 35x^2 + 50x + 24$	-1	-0.99999991
	-2	-2.00000027
	-3	-2.99999973
	-4	-4.00000009
$x^4 - 5x^2 + 4$	2	2.00000012
	-2	-2.00000012
	1	0.999999768
	-1	-0.999999768
$x^4 - 1$	1	1
	-1	-1
	i	i
	-i	-i

CHAPTER III

POLYNOMIAL EQUATIONS OF DEGREE ≥ 5

As stated previously, no formula exists for solving polynomials of degree five or more. The roots of these polynomial equations are often found through the use of some iterative method. When using any iterative method, it will be necessary to frequently evaluate the polynomial. There are at least four possible ways to evaluate a polynomial; evaluation with exponents, evaluation without exponents, factored form, and synthetic substitution. Thus, to determine the best way to evaluate a polynomial, several polynomials were evaluated using these four methods, and their results were compared.

Except for the cases in which multiple roots were involved, all methods gave similar results. In the cases of multiple roots, evaluation of the polynomial in factored form was clearly better. However, this is not a viable choice. The next best method was a tie between evaluation without exponents and synthetic substitution. Synthetic substitution was chosen because it requires fewer multiplications for the same accuracy of the evaluation. Program 3.1 will evaluate a polynomial using synthetic substitution.

The n roots of the n-th degree polynomial equation may be real or complex in nature. Complex roots, however, will always occur in pairs. Thus, if a + bi is a root, then its conjugate, a - bi, is also a root. Therefore, any odd degree polynomial equation will contain at least one real root. Since an even degree polynomial may contain no

PROGRAM 3.1

```
10  REM  *** SYNTHETIC SUBSTITUTION ***
20  REM  *** A(X) = COEFFICIENTS OF POLYNOMIAL
30  REM  *** Y = VALUE OF POLYNOMIAL
40  REM  *** R = VALUE POLYNOMIAL IS BEING EVALUATED A
     T
50  HOME
60  PRINT "THIS PROGRAM USES SYNTHETIC SUBSTITUION"
70  PRINT "TO EVALUATE A POLYNOMIAL"
80  PRINT
90  PRINT "ENTER THE DEGREE OF THE EQUATION"
100  INPUT "(MAXIMUM DEGREE IS 10) ";N
110  PRINT
120  REM  *** ENTER THE COEFFICIENTS
130  PRINT "THE COEFFICIENT OF THE X^N TERM IS A(N)"
140  FOR X = N TO 0 STEP  - 1
150  PRINT "ENTER A("X")";
160  INPUT " ";A(X)
170  NEXT X
180  PRINT
190  INPUT "ENTER THE VALUE TO BE SUBSTITUTED ";R
200  REM  *** EVALUATE THE POLYNOMIAL
210 Y = A(N) * R
220  FOR X = N - 1 TO 1 STEP  - 1
230 Y = (Y + A(X)) * R
240  NEXT X
250 Y = Y + A(0)
260  PRINT
270  PRINT "F("R") = "Y
280  END
```

real root, it is most beneficial to know some additional information about the nature of the roots. Descartes' Rule of Signs will provide this information.

Descartes' Rule of Signs may be used to determine the maximum number of positive and negative real roots. The maximum number of positive real roots of the polynomial equation, $P(x) = 0$, may not exceed the number of variations in sign of the polynomial. Likewise, the maximum number of negative real roots may not exceed the number of variations in sign of $P(-x)$.

A polynomial equation with $a_0 = 0$ will have zero as one of its roots. The minimum number of complex roots of a polynomial equation may be determined by subtracting the maximum number of positive real roots, the maximum number of negative real roots, and the number of zeroes from the degree n of the polynomial. Program 3.2 will determine the nature of the roots using Descartes' Rule of Signs.

PROGRAM 3.2

```
10  REM  *** RULE OF SIGNS ***
20  REM  *** A(X) = COEFFICIENTS OF POLYNOMIAL
30  REM  *** RP = NUMBER OF POSITIVE ROOTS
40  REM  *** RN = NUMBER OF NEGATIVE ROOTS
50  REM  *** RZ = NUMBER OF ZERO ROOTS
60  REM  *** S = SIGN OF THE TERM
70  HOME
80  PRINT "THIS PROGRAM USES DESCARTES' RULE OF"
90  PRINT "SIGNS TO DETERMINE THE MAXIMUM NUMBER"
100  PRINT "OF POSITIVE, NEGATIVE, AND ZERO ROOTS"
110  PRINT
120  PRINT "ENTER THE DEGREE OF THE POLYNMIAL"
130  INPUT "(MAXIMUM DEGREE IS 10) ";N
140  PRINT
150  PRINT "THE COEFFICIENT OF THE X^N TERM IS A(N)"
160  PRINT
170  REM  *** ENTER THE COEFFICIENTS
180  FOR X = N TO 0 STEP  - 1
190  PRINT "ENTER A(";X;")";
200  INPUT " ";A(X)
210  NEXT X
220  REM  *** CALCULATE NUMBER OF POSITIVE ROOTS
230 RP = 0
240  REM  *** DETERMINE SIGN OF FIRST TERM
250 S =  SGN (A(N))
260  FOR X = N - 1 TO 0 STEP  - 1
270  REM  *** CHECK TO SEE IF COEFFICIENT IS ZERO
280  IF  SGN (A(X)) = 0 THEN 350
290  REM  *** CHECK TO SEE IF THE SIGN OF THE COEFFICI
     ENT IS DIFFERENT FROM PREVIOUS TERM
300  IF S =  SGN (A(X)) THEN 350
310  REM  *** IF SIGN IS DIFFERENT, ADD 1 TO NUMBER OF
      POSITIVE ROOTS
320 RP = RP + 1
330  REM  *** RESET VALUE OF SIGN
340 S =  SGN (A(X))
350  NEXT X
360 RN = 0
370  REM  *** CHECK IF ODD OR EVEN DEGREE POLYNOMIAL
```

PROGRAM 3.2 (continued)

```
380  IF (N / 2) =  INT (N / 2) THEN 420
390  REM  *** IF N IS ODD, CHANGE SIGN OF FIRST TERM
400 S =  SGN ( - A(N))
410  GOTO 430
420 S =  SGN (A(N))
430  FOR X = (N - 1) TO 0 STEP  - 1
440  IF  SGN (A(X)) = 0 THEN 560
450  REM  *** CHECK IF EXPONENT IS EVEN OR ODD
460  IF (X / 2) =  INT (X / 2) THEN 530
470  REM  *** IF EXPONENT IS ODD, CHANGE SIGN OF TERM
480  IF S =  SGN ( - A(X)) THEN 560
490  REM  *** IF SIGN IS DIFFERENT FROM PREVIOUS TERM,
      ADD 1 TO NEGATIVE ROOTS
500 RN = RN + 1
510 S =  SGN ( - A(X))
520  GOTO 560
530  IF S =  SGN (A(X)) THEN 560
540 RN = RN + 1
550 S =  SGN (A(X))
560  NEXT X
570  REM  *** CALCULATE NUMBER OF ZERO ROOTS
580 RZ = 0
590  FOR X = 0 TO N
600  IF A(X) <  > 0 THEN 640
610 RZ = RZ + 1
620  NEXT X
630  REM  *** PRINT OUT RESULTS
640  PRINT
650  PRINT "THE MAXIMUM NUMBER OF POSITIVE ROOTS IS "R
     P
660  PRINT
670  PRINT "THE MAXIMUM NUMBER OF NEGATIVE ROOTS IS "R
     N
680  PRINT
690  PRINT "THERE ARE "RZ" ZERO ROOTS"
700  END
```

Most of the iterative methods require that an initial approximation be supplied. This requires some knowledge as to the graph of the polynomial. From this graph an approximation to the root can be made.

There are methods for calculating the upper and lower bounds of the interval which contains the real roots of the polynomial equation. By knowing the bounds of this region, the section of the graph which should be studied will also be known, and thus an approximation of the root can be made. One such method, described in [7, p. 300], for finding the upper and lower bounds (UB and LB, respectively) is as follows:

$$UB = \frac{|a_n| + |a_{n-1}| + \dots + |a_0|}{|a_n|}$$

$$LB = -UB$$

If, according to Descartes' Rule of Signs, there are no positive real roots, then UB = 0. Likewise, if there are no negative real roots, then LB = 0. Program 3.3 uses this method to calculate the upper and lower bounds.

<u>PROGRAM 3.3</u>

```
10  REM  *** LIMITS ***
20  REM  *** A(X) = COEFFICIENTS OF POLYNOMIAL
30  REM  *** RP = NUMBER OF POSITIVE ROOTS
40  REM  *** RN = NUMBER OF NEGATIVE ROOTS
50  REM  *** LL = LOWER LIMIT
60  REM  *** UL = UPPER LIMIT
70  HOME
80  PRINT "THIS PROGRAM FINDS THE INTERVAL WHICH"
90  PRINT "CONTAINS ALL POSSIBLE REAL ROOTS OF THE"
100  PRINT "POLYNOMIAL"
110  PRINT
120  PRINT "ENTER THE DEGREE OF THE POLYNMIAL"
130  INPUT "(MAXIMUM DEGREE IS 10) ";N
140  PRINT
150  PRINT "THE COEFFICIENT OF THE X^NTH TERM"
160  PRINT "IS A(N)"
170  PRINT
180  REM  *** ENTER COEFFICIENTS
190  FOR X = N TO 0 STEP  - 1
200  PRINT "ENTER A(";X;")";
210  INPUT " ";A(X)
220  NEXT X
230  REM  *** DETERMINE RP
240 S =  SGN (A(N))
250 RP = 0
260  FOR X = (N - 1) TO 0 STEP  - 1
270  IF  SGN (A(X)) = 0 THEN 310
280  IF S =  SGN (A(X)) THEN 310
290 RP = RP + 1
300 S =  SGN (A(X))
310  NEXT X
```

PROGRAM 3.3 (continued)

```
320  REM  *** DETERMINE UPPER LIMIT
330 UL = 0
340  IF RP = 0 THEN 400
350  FOR X = 0 TO N
360 UL = UL +  ABS (A(X))
370  NEXT X
380 UL = UL / A(N)
390  REM  *** DETERMINE RN
400 RN = 0
410  IF (N / 2) =  INT (N / 2) THEN 440
420 S =  SGN ( - A(N))
430  GOTO 450
440 S =  SGN (A(N))
450  FOR X = (N - 1) TO 0 STEP  - 1
460  IF  SGN (A(X)) = 0 THEN 550
470  IF (X / 2) =  INT (X / 2) THEN 520
480  IF S =  SGN ( - A(X)) THEN 550
490 RN = RN + 1
500 S =  SGN ( - A(X))
510  GOTO 550
520  IF S =  SGN (A(X)) THEN 550
530 RN = RN + 1
540 S =  SGN (A(X))
550  NEXT X
560  REM  *** DETERMINE LOWER LIMIT
570 LL = 0
580  IF RN = 0 THEN 580
590  FOR X = 0 TO N
600 LL = LL +  ABS (A(X))
610  NEXT X
620 LL =  - LL / A(N)
630  REM  *** ROUND LIMITS TO NEXT INTEGER
640  IF UL =  INT (UL) THEN 660
650 UL =  INT (UL) + 1
660 LL =  INT (LL)
670  REM  *** PRINT RESULTS
680  PRINT "THE LOWER LIMIT IS ";LL
690  PRINT "THE UPPER LIMIT IS ";UL
700  END
```

The roots--especially the complex roots--of a polynomial of degree five or more may be found by repeatedly approximating the root and reducing the polynomial until a polynomial of degree four is obtained. This polynomial may then be solved by using the quartic formula. Program 3.4 will reduce a given polynomial by using synthetic division.

PROGRAM 3.4

```
10  REM  *** SYNTHETIC DIVISION ***
20  REM  *** A(X) = COEFFICIENTS OF POLYNOMIAL
30  REM  *** B(X) = COEFFICIENTS OF REDUCED POLYNOMIAL
40  REM  *** R = POLYNOMIAL WILL BE DIVIDED BY (X-R)
50  REM  *** R1 = REMAINDER
60  HOME
70  PRINT "ENTER THE DEGREE OF THE EQUATION TO BE"
80  INPUT "REDUCED (MAXIMUM 10) ";N
90  PRINT
100  REM  *** ENTER COEFFICIENTS OF POLYNOMIAL
110  PRINT "THE COEFFICIENT OF THE X^N TERM IS A(N)"
120  PRINT
130  FOR X = N TO 0 STEP  - 1
140  PRINT "ENTER A("X")";
150  INPUT " ";A(X)
160  NEXT X
170  PRINT
180  PRINT "THE POLYNOMIAL IS TO BE DIVIDED BY (X-R)"
190  INPUT "ENTER R ";R
200 R1 = 0
210  REM  *** REDUCE THE POLYNOMIAL
220 B(N - 1) = A(N)
230  FOR X = N - 1 TO 1 STEP  - 1
240 B(X - 1) = B(X) * R + A(X)
250  NEXT X
260  REM  *** CALCULATE REMAINDER
270 R1 = B(0) * R + A(0)
280  REM  *** PRINT OUT REDUCED POLYNOMIAL
290  PRINT
300  PRINT "THE REDUCED POLYNOMIAL IS"
310  PRINT
320  IF N = 2 THEN 370
330  IF N = 1 THEN 380
340  FOR X = N - 1 TO 2 STEP  - 1
350  PRINT B(X);"X^";X;" + ";
360  NEXT X
370  PRINT B(1)"X + ";
380  PRINT B(0)
390  REM  *** CHECK TO SEE IF THERE IS A REMAINDER
400  IF R1 = 0 THEN 430
410  PRINT
420  PRINT "THERE IS A REMAINDER OF ";R1
430  END
```

Newton's method requires the use of the first derivative. Program 3.5 will compute the coefficients of the first derivative.

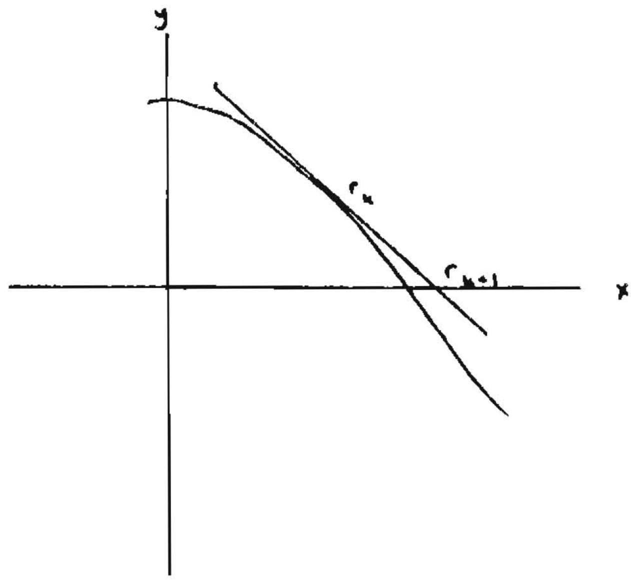
PROGRAM 3.5

```
10  REM  *** DERIVATIVE ***
20  REM  *** A(X) = COEFFICIENTS OF POLYNOMIAL
30  REM  *** B(X) = COEFFICIENTS OF DERIVATIVE
40  HOME
50  PRINT "THIS PROGRAM WILL FIND THE FIRST"
60  PRINT "DERIVATIVE"
70  PRINT
80  PRINT "ENTER THE DEGREE OF THE POLYNOMIAL"
90  INPUT "(MAXIMUM DEGREE IS 10) ";N
100  PRINT
110  PRINT "THE COEFFICIENT OF THE X^N TERM IS A(N)"
120  PRINT
130  REM  *** ENTER COEFFICIENTS
140  FOR X = N TO 0 STEP  - 1
150  PRINT "ENTER A("X")";
160  INPUT " ";A(X)
170  NEXT X
180  REM  *** COMPUTE THE COEFFICIENTS OF THE DERIVATI
     VE
190  FOR X = N TO 1 STEP  - 1
200 B(X - 1) = A(X) * X
210  NEXT X
220  REM  *** PRINT OUT THE FIRST DERIVATIVE
230  PRINT
240  PRINT "THE DERIVATIVE IS "
250  PRINT
260  IF N = 2 THEN 320
270  IF N = 1 THEN 330
280  IF N = 0 THEN 340
290  FOR X = N - 1 TO 2 STEP  - 1
300  PRINT B(X)"X^"X" + ";
310  NEXT X
320  PRINT B(1)"X + ";
330  PRINT B(0)
340  END
```

CHAPTER IV

NEWTON'S METHOD

Newton's method is an iterative method for solving non-linear equations. In order to find the root of a polynomial equation, $P(x) = 0$, it is necessary to find a value r such that $P(r) = 0$. This is done by approximating the function P with the tangent line of the function at $x = r_k$. The point, r_{k+1}, where the tangent line intersects the x-axis is used as the next approximation of the root r of P.



When Newton's method is applied to polynomials, it yields the following formula for calculating successive approximations to the root:

$$r_{k+1} = r_k - P(r_k)/P'(r_k)$$

where r_k is the current approximation, and r_{k+1} is the successive approximation.

The following theorem, given in [1, p. 54], is often given in conjunction with Newton's method.

THEOREM

Assume $f(x)$, $f'(x)$, and $f''(x)$ are continuous for all x in some neighborhood of r, and assume $f(r) = 0$, $f'(r) \neq 0$. Then if r_0 is chosen sufficiently close to r, the iterates r_k, $k \geq 0$, will converge to r.

Moreover,

$$\underset{k \to \infty}{\text{Limit}} \frac{r-r_{k+1}}{(r-r_k)^2} = \frac{f''(r)}{2f'(r)}$$

proving that the iterates are quadratically convergent.

One disadvantage to using Newton's method is that it requires the use of the first derivative. However, for a polynomial the first derivative is easy to evaluate.

One advantage to Newton's method is that, once r_k becomes sufficiently close to r, it is quadratically convergent. However, in the case of multiple roots, that is, when $f'(r) = 0$, this is not necessarily true. In general, it is not known a priori if multiple roots are present. Some of the problems that may occur when multiple roots are present will, therefore, be illustrated later in this chapter.

But no matter what the nature of the roots, it should be pointed out that the real numbers are continuous; that is, between any two real numbers there exists another real number. The floating-point numbers used by the microcomputer, however, are granular; that is, between any two floating-point numbers there does not necessarily exist another floating-point number. Therefore, as soon as the error in Newton's method approaches the distance between nearby floating-point numbers, the granular structure of the floating-point number system prevents the continued use of the algorithm [5, p. 158].

The formula used for finding the successive approximations to the root is used repeatedly, then, until r_{k+1} becomes sufficiently close to the root. The usual point of termination is when $|r_{k+1} - r_k| < \epsilon$. There may exist, however, cases where $|r_{k+1} - r_k| < \epsilon$ but $|r_{k+1} - r| \not< \epsilon$.

Program 4.1 will find a root of a polynomial equation using Newton's method. Table 4.1 summarizes some of the results obtained by using Program 4.1.

PROGRAM 4.1

```
10  REM  *** NEWTON ***
20  REM  *** R = CURRENT APPROXIMATION
30  REM  *** R1 = F(X)
40  REM  *** R2 = F'(X)
50  REM  *** R3 = NEW APPROXIMATION
60  REM  *** A(X) = F(X)
70  REM  *** B(X) = F'(X)
80  REM  *** I = ITERATION NUMBER
90  HOME
100  PRINT "THIS PROGRAM WILL FIND THE ROOT OF A"
110  PRINT "POLYNOMIAL USING NEWTON'S METHOD"
120  PRINT
130  PRINT "ENTER THE DEGREE OF THE POLYNOMIAL"
140  INPUT "(MAXIMUM DEGREE IS 10) ";N
150  PRINT
160  REM  *** ENTER THE COEFFICIENTS
170  PRINT "THE COEFFICIENT OF THE X^NTH TERM"
180  PRINT "IS A(N)"
190  PRINT
200  FOR X = N TO 0 STEP  - 1
210  PRINT "ENTER A("X")";
220  INPUT " ";A(X)
230  NEXT X
240  PRINT
250  INPUT "ENTER THE INITIAL GUESS ";R
260  REM  *** SEND OUTPUT TO PRINTER
270  PR# 1
280  REM  *** PRINT EQUATION AND HEADINGS
290  PRINT  TAB( 10);" ";
300  FOR X = N TO 2 STEP  - 1
310  PRINT A(X);"X^";X;" + ";
320  NEXT X
330  PRINT A(1);"X + ";A(0)
```

PROGRAM 4.1 (continued)

```
340  PRINT
350  PRINT "ITT NO.","    ROOT","    F(X)"
360  PRINT
370 I = 1
380  REM  *** CALCULATE FIRST DERIVATIVE
390  GOSUB 2000
400  REM  *** CALCULATE F(X)
410  GOSUB 2500
420 R1 = Y
430  REM  *** CALCULATE F'(X)
440  GOSUB 3000
450 R2 = Y
460  REM  *** CALCULATE NEW APPROXIMATION
470 R3 = R - R1 / R2
480  REM  *** CALCULATE CLOSENESS OF ANSWER
490  IF  ABS (R3 - R) < 1E - 6 THEN 560
500  REM  *** CALCULATE F(X) OF NEW APPROXIMATION
510 R = R3
520  GOSUB 2500
530  PRINT "  "I,R,Y
540 I = I + 1
550  GOTO 420
560  PRINT
570  PRINT "THE ROOT IS ";R3
580  PR# 0
1999  END
2000  REM  *** CALCULATE FIRST DERIVATIVE
2010  FOR X = N TO 1 STEP  - 1
2020 B(X - 1) = A(X) * X
2030  NEXT X
2040  RETURN
2500  REM  *** CALCULATE F(X)
2510 Y = A(N) * R
2520  FOR X1 = N - 1 TO 1 STEP  - 1
2530 Y = (Y + A(X1)) * R
2540  NEXT X1
2550 Y = Y + A(0)
2560  RETURN
3000  REM  *** CALCULATE F'(X)
3010  IF N = 1 THEN 3070
3020 Y = B(N - 1) * R
3030  IF N = 2 THEN 3070
3040  FOR X1 = N - 2 TO 1 STEP  - 1
3050 Y = (Y + B(X1)) * R
3060  NEXT X1
3070 Y = Y + B(0)
3080  RETURN
```

TABLE 4.1

Equation	Init. Approx.	No. of Iter.	Actual Root	Computed Root
$x^3 - 2x^2 - x + 2$	0	1	2	2
$x^3 + 2x^2 - 5x - 6$	0	4	-1	-1
$4x^3 - 3x$	-1	4	-0.8660254	-0.866025404
$x^3 + 6x^2 + 11x + 6$	0	6	-1	-1
$x^3 + 4x^2 + 5x + 2$	0	17	-1	-1.00000405
$x^3 + 3x^2 + 3x + 1$	0	37	-1	-1.00040349
$x^4 + 10x^3 + 35x^2 + 50x + 24$	0	6	-1	-1
$8x^4 - 8x^2 + 1$	-1	3	-0.9238795	-0.923879532
$x^4 + 4x^3 + 6x^2 + 4x + 1$	0	19	-1	-0.995481137
$16x^5 - 20x^3 + 5x$	-1	3	-0.9510565	-0.951056516
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	0	6	-1	-1
$x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$	0	32	-1	-1.00110733

While Newton's method works quite well when the roots are distinct, the number of iterations required when multiple roots are present increases dramatically. The accuracy of the answer also diminishes considerably. In order to learn more about why this occurs, the value of each approximation and of the function after each iteration was studied. Table 4.2 summarizes the results for $P(x) = (x+1)^3$, $P(x) = (x+1)^4$, and $P(x) = (x+1)^5$.

TABLE 4.2

$P(x) = (x+1)^3$

Iteration No.	Root	P(x)
1	-0.333333333	0.296296296
2	-0.555555556	0.0877914955
3	-0.703703704	0.0260122947
⋮	⋮	⋮
18	-0.999380235	6.98491931 E-10
19	-0.999986051	4.65661287 E-10
20	-1.49998605	-0.124989538
21	-1.33332404	-0.0370339374
⋮	⋮	⋮
35	-1.00109883	-9.31322575 E-10
36	-1.0008417	-9.31322575 E-10
37	-1.00040349	0

$P(x) = (x+1)^4$

Iteration No.	Root	P(x)
1	-0.25	0.31640625
2	-0.4375	0.100112915
3	-0.578125	0.031676352
⋮	⋮	⋮
17	-0.99237465	3.02679837 E-9
18	-0.994079792	1.16415322 E-9
19	-0.995481137	0

TABLE 4.2 (continued)

$P(x) = (x+1)^5$

Iteration No.	Root	P(x)
1	-0.2	0.32768
2	-0.36	0.107374183
3	-0.488	0.0351843718
⋮	⋮	⋮
19	-0.984748639	1.39698386 E-9
20	-0.989921053	-2.32830644 E-9
21	-0.945278196	4.9173832 E-7
22	-0.956246206	1.5925616 E-7
⋮	⋮	⋮
30	-0.986248407	6.98491931 E-10
31	-0.990237769	4.65661287 E-10
32	-1.00110733	0

A closer look at the approximations reveals a "jump" at approximately the 20th iteration. This is especially noticeable for odd-degree polynomials. In order to understand more clearly what was happening at this point, the values of P'(x) were also printed. Table 4.3 summarizes the results for $P(x) = (x+1)^3$, $P(x) = (x+1)^4$, and $P(x) = (x+1)^5$.

TABLE 4.3

$P(x) = (x+1)^3$

Iteration No.	Root	P(x)	P'(x)
1	-0.333333333	1	3
2	-0.555555556	0.296296296	1.33333333
3	-0.703703704	0.0877914955	0.592592592

TABLE 4.3 (continued)

$P(x) = (x+1)^3$ (continued)

Iteration No.	Root	P(x)	P'(x)
⋮	⋮	⋮	⋮
18	-0.999380235	1.16415322 E-9	3.02679837 E-6
19	-0.999986051	6.98491931 E-10	1.15297735 E-6
20	-1.49998605	4.65661287 E-10	9.31322575 E-10
21	-1.33332404	-0.124989538	0.749958155
⋮	⋮	⋮	⋮
35	-1.00109883	-5.58793545 E-9	8.92579556 E-6
36	-1.0008417	-9.31322575 E-10	3.62191349 E-6
37	-1.00040349	-9.31322575 E-10	2.12527812 E-6

$P(x) = (x+1)^4$

Iteration No.	Root	P(x)	P'(x)
1	-0.25	1	4
2	-0.4375	0.31640625	1.6875
3	-0.578125	0.100112915	0.711914063
⋮	⋮	⋮	⋮
17	-0.99237465	9.54605639 E-9	4.00841236 E-6
18	-0.994079792	3.02679837 E-9	1.77510083 E-6
19	-0.995481137	1.16415322 E-9	8.30739737 E-7

TABLE 4.3 (continued)

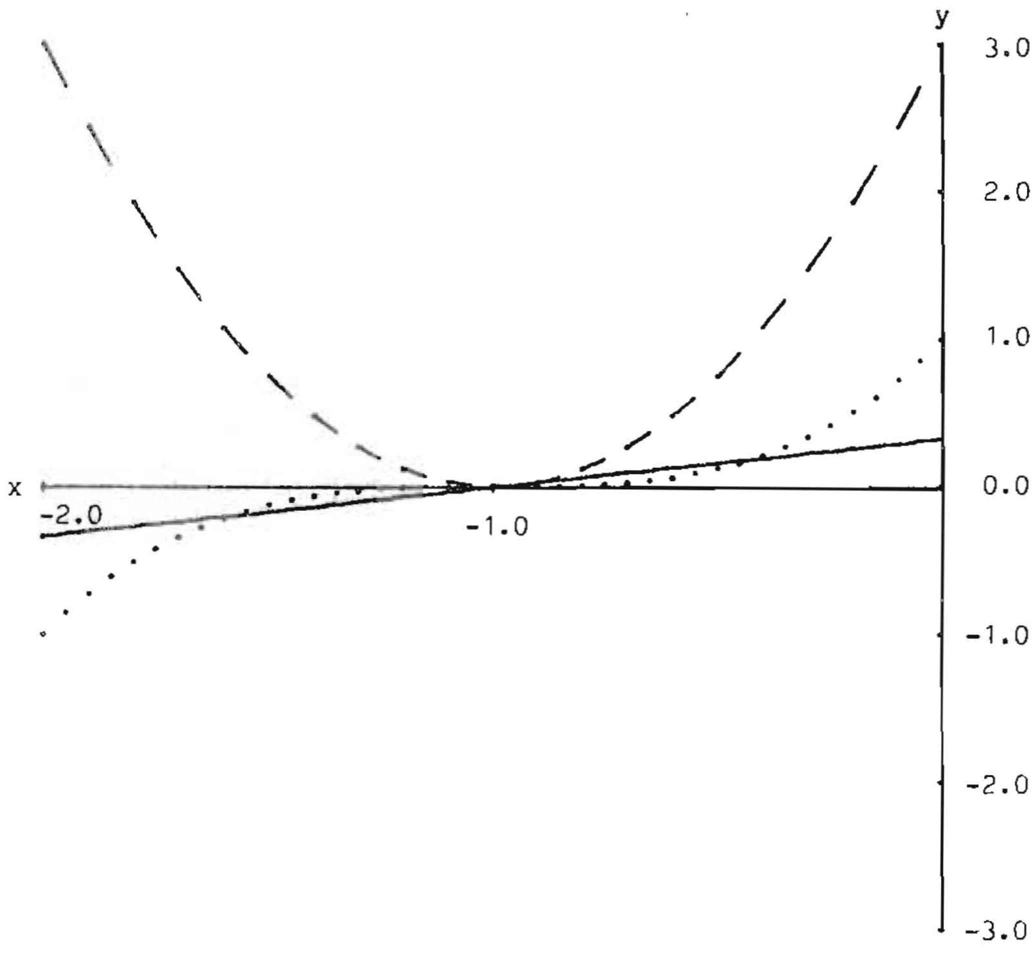
$P(x) = (x+1)^5$

Iteration No.	Root	P(x)	P'(x)
1	-0.2	1	5
2	-0.36	0.32768	2.048
3	-0.488	0.107374183	0.8388608
⋮	⋮	⋮	⋮
19	-0.984748639	1.86264515 E-9	5.81145287 E-7
20	-0.989921053	1.39698386 E-9	2.70083547 E-7
21	-0.945278196	-2.32830644 E-9	5.21540642 E-8
22	-0.956246206	4.9173832 E-7	4.48338688 E-5
⋮	⋮	⋮	⋮
30	-0.986248407	1.86264515 E-9	4.90562449 E-7
31	-0.990237769	6.98491931 E-10	1.75088644 E-7
32	-1.00110733	4.65661287 E-10	4.28408384 E-8

A look at the values for P(x) and P'(x) in the region of the "jump" shows that P(x) and P'(x) are approximately equal to zero. In fact, at the point $x = r$, $P(x) = P'(x) = 0$. Since Newton's method also involves the quotient $P(x)/P'(x)$, round-off error becomes especially important in the region around the root. Graphing P(x), P'(x), and the quotient reveals that in the region around the root, $P(x) \approx 0$, $P'(x) \approx 0$, $P(x)/P'(x) \approx 0$. Thus, the method used to evaluate the functions becomes highly critical as $x \to r$. Graphs 4.1, 4.2, and 4.3 illustrate this fact for $P(x) = (x+1)^3$, $P(x) = (x+1)^4$, and $P(x) = (x+1)^5$.

GRAPH 4.1

$P(x) = (x+1)^3$



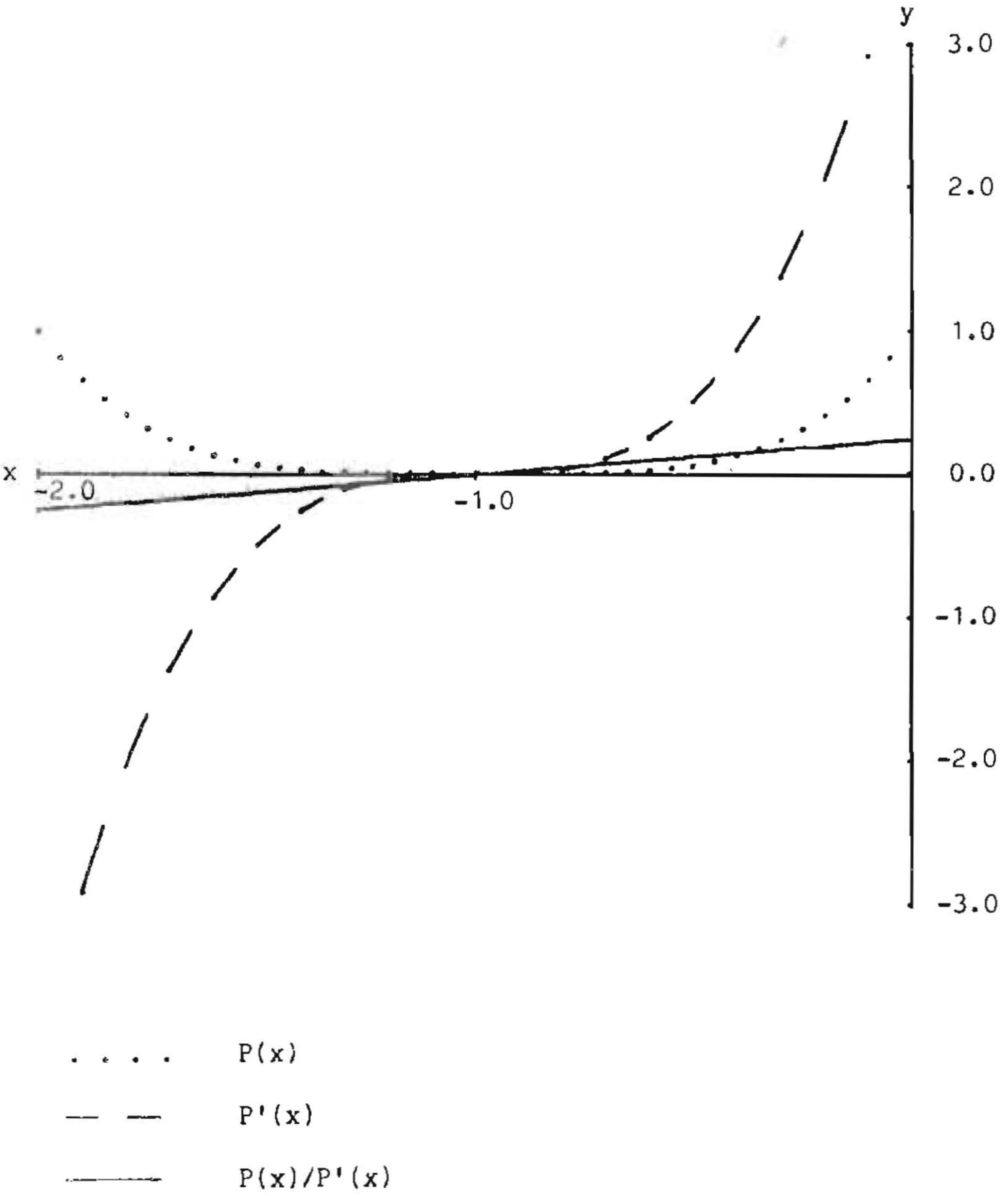
. . . . P(x)

— — P'(x)

——— P(x)/P'(x)

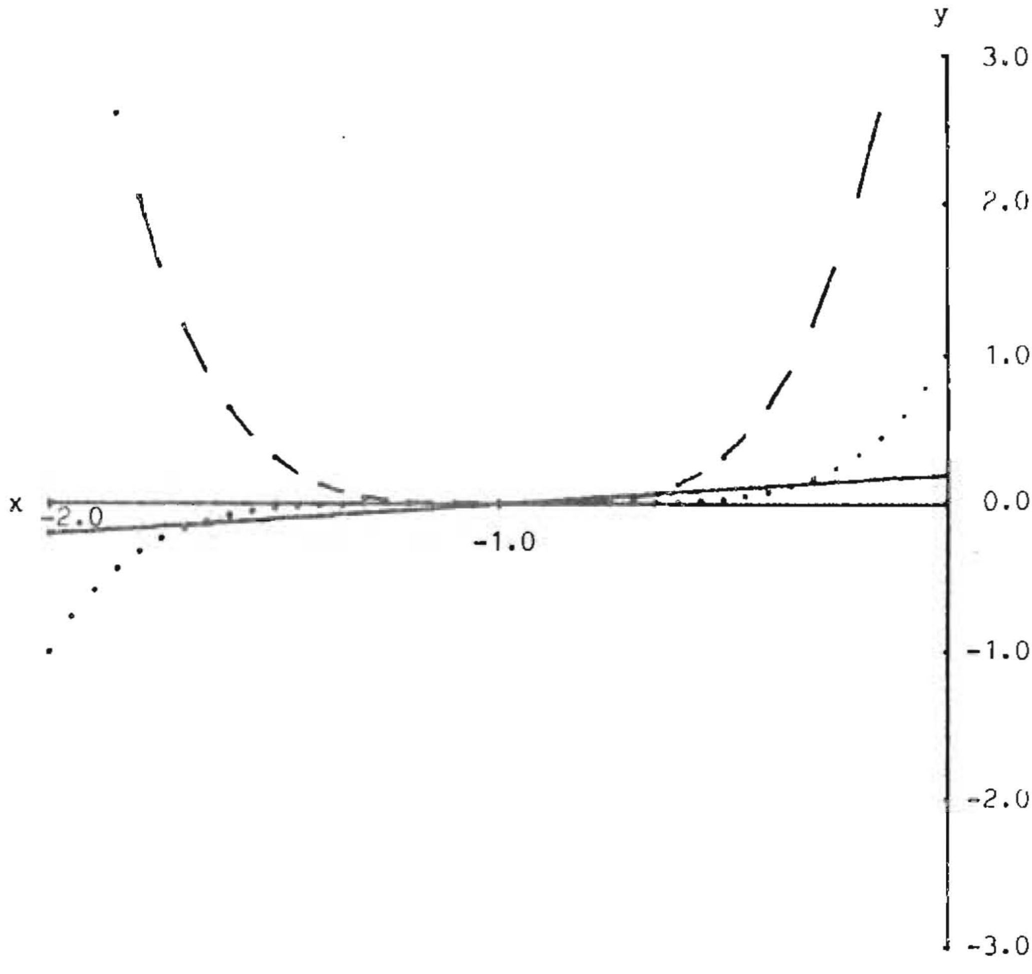
GRAPH 4.2

$P(x) = (x+1)^4$



GRAPH 4.3

$P(x) = (x+1)^5$



. . . . P(x)

— — P'(x)

——— P(x)/P'(x)

Close inspection of the values of P(x) and P'(x) as listed in Table 4.3 reveal that normally $P(x) < P'(x)$. However, in the region of the "jump" $P(x) \approx P'(x)$. In order to accomplish this, the value of P'(x) decreased greatly; that is, $P'(r_{k+1}) \ll P'(r_k)$. Program 4.1 was modified to check for a significant decrease in the value of P'(x). This is especially critical in the case of $P(x) = (x+1)^3$, the value of the root was much closer to the actual root before the "jump," than at the end of the program. Program 4.2 shows the revised version of Program 4.1. Table 4.4 summarizes some sample output from this program.

PROGRAM 4.2

```
10  REM  *** NEWTON REVISED ***
20  REM  *** R = VALUE USED IN SUBROUTINES
30  REM  *** R1 = CURRENT APPROXIMATION
40  REM  *** R2 = F(X)
50  REM  *** R3 = F'(X)
60  REM  *** R4 = NEW APPROXIMATION
70  REM  *** R5 = PREVIOUS F(X)
80  REM  *** R6 = PREVIOUS APPROXIMATION
90  REM  *** A(X) = F(X)
100  REM  *** B(X) = F'(X)
110  REM  *** I = ITERATION NUMBER
120  HOME
130  PRINT "THIS PROGRAM WILL FIND THE ROOT OF A"
140  PRINT "POLYNOMIAL USING NEWTON-RAPHSON"
150  PRINT
160  PRINT "ENTER THE DEGREE OF THE POLYNOMIAL"
170  INPUT "(MAXIMUM DEGREE IS 10) ";N
180  PRINT
190  REM  *** ENTER THE COEFFICIENTS
200  PRINT "THE COEFFICIENT OF THE X^NTH TERM"
210  PRINT "IS A(N)"
220  PRINT
230  FOR X = N TO 0 STEP  - 1
240  PRINT "ENTER A("X")";
250  INPUT " ";A(X)
260  NEXT X
270  PRINT
280  INPUT "ENTER THE INITIAL GUESS ";R1
290  REM  *** SEND OUTPUT TO PRINTER
300  PR# 1
310  REM  *** PRINT EQUATION AND HEADINGS
320  PRINT  TAB( 10);" ";
330  FOR X = N TO 2 STEP  - 1
```

PROGRAM 4.2 (continued)

```
340  PRINT A(X);"X^";X;" + ";
350  NEXT X
360  PRINT A(1);"X + ";A(0)
370  PRINT
380  PRINT "ITT NO.","    ROOT","    F(X)"
390  PRINT
400 I = 1
410  REM  *** CALCULATE FIRST DERIVATIVE
420  GOSUB 2000
430  REM  *** CALCULATE F(X)
440 R = R1
450  GOSUB 2500
460 R2 = Y
470  REM  *** CALCULATE F'(X)
480  GOSUB 3000
490 R3 = Y
500  REM  *** CALCULATE NEW APPROXIMATION
510 R4 = R1 - R2 / R3
520  REM  *** CALCULATE CLOSENESS OF ANSWER
530 R = R4
540  IF  ABS (R4 - R1) < 1E - 6 THEN 640
550  IF  ABS (R5 / 100) >  ABS (R3) THEN R = R6: GOTO
     640
560 R5 = R3
570 R6 = R4
580  REM  *** CALCULATE F(X) OF NEW APPROXIMATION
590 R1 = R4
600  GOSUB 2500
610  PRINT "  "I,R,Y
620 I = I + 1
630  GOTO 460
640  PRINT
650  PRINT "THE ROOT IS ";R
660  PR# 0
1000  END
2000  REM  *** CALCULATE FIRST DERIVATIVE
2010  FOR X = N TO 1 STEP  - 1
2020 B(X - 1) = A(X) * X
2030  NEXT X
2040  RETURN
2500  REM  *** CALCULATE F(X)
2510 Y = A(N) * R
2520  FOR X1 = N - 1 TO 1 STEP  - 1
2530 Y = (Y + A(X1)) * R
2540  NEXT X1
2550 Y = Y + A(0)
2560  RETURN
3000  REM  *** CALCULATE F'(X)
3010  IF N = 1 THEN 3070
3020 Y = B(N - 1) * R
3030  IF N = 2 THEN 3070
3040  FOR X1 = N - 2 TO 1 STEP  - 1
```

PROGRAM 4.2 (continued)

```
3050 Y = (Y + B(X1)) * R
3060  NEXT X1
3070 Y = Y + B(0)
3080  RETURN
```

TABLE 4.4

Equation	Init. Approx.	No. of Iter.	Actual Root	Computed Root
$x^3 - 2x^2 - x + 2$	0	1	2	2
$x^3 + 2x^2 - 5x - 6$	0	4	-1	-1
$4x^3 - 3x$	-1	4	-0.8660254	-0.866025404
$x^3 + 6x^2 + 11x + 6$	0	6	-1	-1
$x^3 + 4x^2 + 5x + 2$	0	17	-1	-1.00000405
$x^3 + 3x^2 + 3x + 1$	0	19	-1	-0.999986051
$x^4 + 10x^3 + 35x^2 + 50x + 24$	0	6	-1	-1
$8x^4 - 8x^2 + 1$	-1	3	-0.9238795	-0.923879532
$x^4 + 4x^3 + 6x^2 + 4x + 1$	0	19	-1	-0.995481137
$16x^5 - 20x^3 + 5x$	-1	3	-0.9516565	-0.951056516
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	0	6	-1	-1
$x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$	0	32	-1	-1.00110733

The next step is to reduce the polynomial equation to see if the root found is a multiple root. Program 4.3 shows only the modification made in Program 4.2.

PROGRAM 4.3

```
405  REM  *** R9 IS MULTIPLICITY
406 R9 = 0
650  PRINT "THE ROOT IS ";R
660 R9 = R9 + 1
670  IF R9 = N THEN 800
680  REM  *** REDUCE POLYNOMIAL
690 N1 = N
700  GOSUB 3500
710  REM  *** CHECK TO SEE IF MULTIPLE ROOT
720  GOSUB 3700
730  IF  ABS (Y) > 1E - 9 THEN 790
740  PRINT
750  PRINT R;" IS A MULTIPLE ROOT"
760 R9 = R9 + 1
770  IF R9 = N THEN 800
780  GOTO 700
790  PRINT R;" IS NOT A MULTIPLE ROOT"
800  PR# 0
2200  REM  *** SET UP COEFFICIENTS FOR REDUCED POLYNOM
    IAL
2210  FOR X = 0 TO N
2220 C(X) = A(X)
2230  NEXT X
2240  RETURN
3500  REM  *** REDUCE POLYNOMIAL
3510 D(N1 - 1) = C(N1)
3520  FOR X1 = N1 - 1 TO 1 STEP  - 1
3530 D(X1 - 1) = D(X1) * R + C(X1)
3540  NEXT X1
3550 N1 = N1 - 1
3560  FOR X1 = 0 TO N1
3570 C(X1) = D(X1)
3580  NEXT X1
3590  RETURN
3700  REM  *** CALCULATE F(X) FOR REDUCED POLYNOMIAL
3710 Y = C(N1) * R
3720  IF N1 = 1 THEN 3760
3730  FOR X1 = N1 - 1 TO 1 STEP  - 1
3740 Y = (Y + C(X1)) * R
3750  NEXT X1
3760 Y = Y + C(0)
3770  RETURN
```

Table 4.5 summarizes the results obtained from using this modification. As can be easily observed, the results were anything but spectacular.

TABLE 4.5

Equation	Root	Multiplicity
$(x+1)^3$	-0.999986051	2
$(x+1)^2\ (x+2)$	-1.00000405	1
$(x+1)^4$	-0.995481137	1
$(x+1)^3\ (x+2)$	-0.999161488	1
$(x+1)^5$	-1.00110733	2

Program 4.2 was further modified to first reduce the equation, and then to ask for a new initial approximation. Newton's method is subsequently applied to this reduced equation. Reduction of the polynomial continues until a reduced equation of degree 2 is obtained, whereupon the quadratic formula is used to obtain the remaining two roots. Program 4.4 will find all real roots of the polynomial equation. Since some of the real roots of the original equation may not be roots of the reduced equation due to error introduced while reducing the polynomial, Descartes' Rule of Signs is also incorporated into Program 4.4. After each reduction a check is made to determine whether or not there are still real roots. If there are no longer any real roots, the process is terminated, unless the equation is of degree 2. Table 4.6 follows, and summarizes some of the results obtained from Program 4.4.

PROGRAM 4.4

```
10  REM  *** NEWTON WITH QUAD ***
20  REM  *** R = VALUE USED IN SUBROUTINES
30  REM  *** R1 = CURRENT APPROXIMATION
40  REM  *** R2 = F(X)
50  REM  *** R3 = F'(X)
60  REM  *** R4 = NEW APPROXIMATION
70  REM  *** R5 = PREVIOUS DIFFERENCE
80  REM  *** R9 = NUMBER OF ROOTS FOUND
90  REM  *** A(X) = F(X)
```

PROGRAM 4.4 (continued)

```
100  REM  *** B(X) = F'(X)
110  REM  *** C(X) = REDUCED POLYNOMIAL
120  REM  *** I = ITERATION NUMBER
130  REM  *** RP = NUMBER OF POSITIVE ROOTS
140  REM  *** RN = NUMBER OF NEGATIVE ROOTS
150  REM  *** RZ = NUMBER OF ZERO ROOTS
160  REM  *** S = SIGN OF THE TERM
170  HOME
180  PRINT "THIS PROGRAM WILL FIND THE ROOT OF A"
190  PRINT "POLYNOMIAL USING NEWTON-RAPHSON"
200  PRINT
210  PRINT "IT WILL ALSO REDUCE THE EQUATION BEFORE"
220  PRINT "FINDING THE NEXT ROOT"
230  PRINT
240  PRINT "ENTER THE DEGREE OF THE POLYNOMIAL"
250  INPUT "(MAXIMUM DEGREE IS 10) ";N
260  PRINT
270  REM  *** ENTER THE COEFFICIENTS
280  PRINT "THE COEFFICIENT OF THE X^NTH TERM"
290  PRINT "IS A(N)"
300  PRINT
310  FOR X = N TO 0 STEP  - 1
320  PRINT "ENTER A("X")";
330  INPUT " ";A(X)
340  NEXT X
350  PRINT
360 R9 = 0
370 N1 = N
380  REM  *** SET UP COEFFICIENTS FOR REDUCED POLYNOMI
     AL
390  GOSUB 2200
400  REM  *** CHECK FOR REAL ROOTS
410  GOSUB 4000
420  IF RP + RN + RZ = 0 THEN  PRINT "THERE ARE NO REA
     L ROOTS": GOTO 1999
430  INPUT "ENTER THE INITIAL GUESS ";R1
440  REM  *** SEND OUTPUT TO PRINTER
450  PR# 1
460  REM  *** PRINT EQUATION AND HEADINGS
470  GOSUB 2000
480  REM  *** REMOVE ZERO ROOTS FROM EQUATION
490  IF RZ = 0 THEN 540
500  IF RZ > 0 THEN  GOSUB 3800
510  GOSUB 2000
520  IF N1 = 2 THEN 950
530  GOTO 490
540  PRINT "ITT NO.","    ROOT","    F(X)"
550  PRINT
560 I = 1
570  REM  *** CALCULATE FIRST DERIVATIVE
580  GOSUB 2100
590  REM  *** CALCULATE F(X)
```

PROGRAM 4.4 (continued)

```
600 R = R1
610  GOSUB 2500
620 R2 = Y
630  REM  *** CALCULATE F'(X)
640  GOSUB 3000
650 R3 = Y
660  REM  *** CALCULATE NEW APPROXIMATION
670 R4 = R1 - R2 / R3
680  REM  *** CALCULATE CLOSENESS OF ANSWER
690 R = R4
700  IF  ABS (R4 - R1) < 1E - 6 THEN 800
710  IF  ABS (R5 / 100) >  ABS (R3) THEN R4 = R6: GOTO
     800
720 R5 = R3
730 R6 = R4
740  REM  *** CALCULATE F(X) OF NEW APPROXIMATION
750 R1 = R4
760  GOSUB 2500
770  PRINT "  "I,R,Y
780 I = I + 1
790  GOTO 620
800  PRINT
810  PRINT "THE ROOT IS ";R
820  PRINT
830  PRINT
840  REM  *** REDUCE POLYNOMIAL
850  GOSUB 3500
860  IF N1 = 2 THEN 940
870  REM  *** CHECK FOR NEXT ROOT
880  PR# 0
890  HOME
900  GOSUB 2020
910 R5 = 0
920 R6 = 0
930  GOTO 400
940  GOSUB 2000
950  REM  *** EVALUATE THE DISCRIMINATE
960 D = C(1) * C(1) - 4 * C(2) * C(0)
970  IF D < 0 THEN 1130
980  REM  *** CALCULATE REAL ROOTS
990 D =  SQR (D)
1000  REM  *** CHECK IF A1 IS POSITIVE OR NEGATIVE
1010  IF C(1) > 0 THEN 1070
1020  REM  *** C(1) IS NEGATIVE
1030 R1 = ( - C(1) + D) / (2 * C(2))
1040 R2 = 2 * C(0) / ( - C(1) + D)
1050  GOTO 1090
1060  REM  *** C(1) IS POSITIVE
1070 R1 = ( - C(1) - D) / (2 * C(2))
1080 R2 = 2 * C(0) / ( - C(1) - D)
1090  PRINT "THE ROOTS ARE "R1
1100  PRINT "AND "R2
1110  GOTO 1180
```

PROGRAM 4.4 (continued)

```
1120  REM  *** CALCULATE COMPLEX ROOTS
1130 D =  SQR ( - D)
1140 R3 =  - C(1) / (2 * C(2))
1150 R4 = D / (2 * C(2))
1160  PRINT "THE COMPLEX ROOTS ARE ";R3;" + ";R4;"I"
1170  PRINT "AND ";R3;" - ";R4;"I"
1180  PR# 0
1999  END
2000  REM  *** PRINT EQUATION
2010  PRINT  TAB( 10);" ";
2020  FOR X = N1 TO 2 STEP  - 1
2030  PRINT C(X);"X^";X;" + ";
2040  NEXT X
2050  PRINT C(1);"X + ";C(0)
2060  PRINT
2100  REM  *** CALCULATE FIRST DERIVATIVE
2110  FOR X = N1 TO 1 STEP  - 1
2120 B(X - 1) = C(X) * X
2130  NEXT X
2140  RETURN
2200  REM  *** SET UP COEFFICIENTS FOR REDUCED POLYNOM
     IAL
2210  FOR X = 0 TO N
2220 C(X) = A(X)
2230  NEXT X
2240  RETURN
2500  REM  *** CALCULATE F(X)
2510 Y = C(N) * R
2520  FOR X1 = N1 - 1 TO 1 STEP  - 1
2530 Y = (Y + C(X1)) * R
2540  NEXT X1
2550 Y = Y + C(0)
2560  RETURN
3000  REM  *** CALCULATE F'(X)
3010  IF N1 = 1 THEN 3070
3020 Y = B(N1 - 1) * R
3030  IF N1 = 2 THEN 3070
3040  FOR X1 = N1 - 2 TO 1 STEP  - 1
3050 Y = (Y + B(X1)) * R
3060  NEXT X1
3070 Y = Y + B(0)
3080  RETURN
3500  REM  *** REDUCE POLYNOMIAL
3510 D(N1 - 1) = C(N1)
3520  FOR X1 = N1 - 1 TO 1 STEP  - 1
3530 D(X1 - 1) = D(X1) * R + C(X1)
3540  NEXT X1
3550 N1 = N1 - 1
3560  FOR X1 = 0 TO N1
3570 C(X1) = D(X1)
3580  NEXT X1
3590  RETURN
3600  REM  *** REDUCE IF ZERO ROOTS
```

PROGRAM 4.4 (continued)

```
3810  FOR X2 = 1 TO N1
3820 C(X2 - 1) = C(X2)
3830  NEXT X2
3840 N1 = N1 - 1
3850 RZ = RZ - 1
3860  PRINT "THE ROOT IS 0"
3870  PRINT
3880  RETURN
4000  REM  *** CALCULATE NUMBER OF POSITIVE ROOTS
4010 RP = 0
4020  REM  *** DETERMINE SIGN OF FIRST TERM
4030 S =  SGN (C(N1))
4040  FOR X2 = N1 - 1 TO 0 STEP  - 1
4050  REM  *** CHECK TO SEE IF COEFFICIENT IS ZERO
4060  IF  SGN (C(X2)) = 0 THEN 4130
4070  REM  *** CHECK TO SEE IF THE SIGN OF THE COEFFIC
     IENT IS DIFFERENT FROM PREVIOUS TERM
4080  IF S =  SGN (C(X2)) THEN 4130
4090  REM  *** IF SIGN IS DIFFERENT, ADD 1 TO NUMBER O
     F POSITIVE ROOTS
4100 RP = RP + 1
4110  REM  *** RESET VALUE OF SIGN
4120 S =  SGN (C(X2))
4130  NEXT X2
4140 RN = 0
4150  REM  *** CHECK IF ODD OR EVEN DEGREE POLYNOMIAL
4160  IF (N1 / 2) =  INT (N1 / 2) THEN 4200
4170  REM  *** IF N1 IS ODD, CHANGE SIGN OF FIRST TERM

4180 S =  SGN ( - C(N1))
4190  GOTO 4210
4200 S =  SGN (C(N1))
4210  FOR X2 = N1 - 1 TO 0 STEP  - 1
4220  IF  SGN (C(X2)) = 0 THEN 4340
4230  REM  *** CHECK IF EXPONENT IS EVEN OR ODD
4240  IF (X2 / 2) =  INT (X2 / 2) THEN 4310
4250  REM  *** IF EXPONENT IS ODD, CHANGE SIGN OF TERM

4260  IF S =  SGN ( - C(X2)) THEN 4340
4270  REM  *** IF SIGN IS DIFFERENT FROM PREVIOUS TERM
     , ADD 1 TO NEGATIVE ROOTS
4280 RN = RN + 1
4290 S =  SGN ( - C(X2))
4300  GOTO 4340
4310  IF S =  SGN (C(X2)) THEN 4340
4320 RN = RN + 1
4330 S =  SGN (C(X2))
4340  NEXT X2
4350  REM  *** COUNT NUMBER OF ZERO ROOTS
4360 RZ = 0
4370  FOR X2 = 0 TO N1
4380  IF C(X2) <  > 0 THEN 4410
```

PROGRAM 4.4 (continued)

```
4390 RZ = RZ + 1
4400  NEXT X2
4410  RETURN
```

TABLE 4.6

Equation	Root
1) $x^5 + 2x^4 + x^3$	0
$x^4 + 2x^3 + x^2$	0
$x^3 + 2x^2 + x$	0
$x^2 + 2x + 1$	-1, -1
2) $x^5 - 13x^3 + 36x$	0
$x^4 - 13x^2 + 36$	2
$x^3 + 2x^2 - 9x - 18$	-3
$x^2 - x - 6$	3, -2
3) $x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	-1
$x^4 + 14x^3 + 71x^2 + 154x + 120$	-2.00000001
$x^3 + 12x^2 + 47x + 59.9999998$	-2.99999998
$x^2 + 9.00000003x + 20.0000001$	-5.00000004, -3.99999998

TABLE 4.6 (continued)

Equation	Root
4) $x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$	-1.00110733
$x^4 + 3.99889267x^3 + 5.99667923x^2 + 3.99668045x + 0.998893891$	-1.00010715
$x^3 + 2.99878552x^2 + 2.99757239x + 0.998786873$	-0.999210898
$x^2 + 1.99957462x + 0.99957564$	-0.99978731 + 9.86997169 E-4i -0.99978731 - 9.86997169 E-4i
5) $16x^5 - 20x^3 + 5x$	0
$16x^4 - 20x^2 + 5$	-0.951056516
$16x^3 - 15.2169043x^2 - 5.52786405x + 5.25731112$	-0.587785252
$16x^2 - 24.6214683x + 8.94427191$	0.951056517, 0.587785252

Again, in the cases involving multiple roots, the results were less than ideal. It is therefore suggested that whenever a multiple root is suspected, an alternate method be used. Several alternate methods will be discussed in the next chapter.

Table 4.7 compares the roots obtained by using the depressed equation and the roots obtained by using the original equation each time. The roots are listed in the order found, with the same initial approximation being given to find each root.

TABLE 4.7

Equation	Init. Approx.	Roots-Depressed	Roots-Original
$16x^5 - 20x^3 + 5x$	0	0	0
	-1	-0.951056516	-0.951056516
	-0.7	-0.587785252	-0.587785252
	0.5	0.587785252	0.587785252
	0.9	0.951056517	0.951056516
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	-5.1	-5.0000001	-5.0000001
	-4.1	-3.99999993	-3.99999997
	-3.1	-2.99999997	-3.00000009
	-2.1	-2.00000007	-2
	-1.1	-0.999999969	-1
$x^5 + 5x^4 - 25x^3 - 125x^2 + 144x + 720$	-5.1	-5.0000001	-5.0000001
	-4.1	-3.99999988	-4.00000007
	-3.1	-3.00000012	-3
	2.9	2.99999988	3
	3.9	4.00000011	4.00000012

The roots obtained by using the original equation were only slightly better than those obtained by using the depressed equation. However, the three equations illustrated have five distinct roots. Had multiple roots been involved, the original equation would have produced much better results, especially if the multiple root were found first. Approximately the same number of iterations were required whether using the original equation or the depressed equation. An alternative method might be to combine the two; that is, use the depressed equation until $P(x)$, then finish with the original equation. This might be especially helpful when working with higher degree equations.

Table 4.8 compares the values of the roots computed by using the depressed equation when the roots are found in ascending order and in descending order.

TABLE 4.8

Equation	Init. Approx.	Roots Ascending	Init. Approx.	Roots Descending
$16x^5 = 20x^3 + 5x$	-1	0	1	0
		-0.951056516		-0.951056517
		-0.587785252		-0.587785252
		0.587785252		0.587785252
		0.951056517		0.951056516
$x^5 + 15x^4 + 85x^3 + 225x^2 + 279x + 120$	-6	-5.00000018	0	-5.00000004
		-3.9999995		-3.99999998
		-3.00000088		-2.99999998
		-1.99999937		-2.00000001
		-1.00000016		-1
$x^5 + 5x^4 - 25x^3 - 125x^2 + 144x + 720$	-6	-5	5	-5
		-4.00000009		-4
		-2.99999994		-3.00000001
		2.99999993		3
		4.00000005		4

The values for the roots are comparable. However, the roots found in descending order are slightly more accurate than those found in ascending order. Thus, when using a depressed equation, the order of finding roots should be taken into consideration.

Conte [4, pp. 66-73] provides an alternate algorithm for finding the roots of a polynomial equation using Newton's method. With the coefficients of the polynomial P(x) stored in a_n, a_{n-1}, . . . , a_0, Conte first stores the coefficients obtained through synthetic division by x-z in b_n, b_{n-1}, . . . , b_1. The remainder is stored in b_0. Thus, $P(x) = q(x)(x-z) + b_0$, where q(x) is the quotient polynomial. When $x = z$, $P(z) = b_0$.

The first derivative of P(x) is also required for Newton's method. If $P(x) = q(x)(x-z) + b_0$, then $P'(x) = q(x)(1) + q'(x)(x-z)$. Again, if $x = z$, $P'(z) = q(z)$.

Conte employs the following algorithm:

Let $z = x_m$, $b_n = a_n$, $c_n = b_n$

for $k = n-1$, . . ., 1, do:

Let $b_k = a_k + zb_{k+1}$

Let $c_k = b_k + zc_{k+1}$

Let $b_0 = a_0 + zb_1$

The value of P(z) is now stored in b_0, and the value of P'(z) is stored in c_1. Thus, $x_{m+1} = x_m - b_0/c_1$.

Program 4.5 finds the roots using Conte's algorithm. Table 4.9 summarizes some sample output for Newton's method and Conte's algorithm. It should be noted, however, that the two methods are mathematically equivalent. The only difference is the order in which the calculations are performed.

PROGRAM 4.5

```
10  REM  *** NEWTON (CONTE) ***
20  REM  *** A(X) = F(X)
30  REM  *** B(X) = CONTE'S B(X)
40  REM  *** C(X) = CONTE'S C(X)
50  REM  *** Z = INITIAL APPROXIMATION
60  REM  *** Z1 = SUCCESSIVE APPROXIMATION
70  REM  *** I = ITERATION NUMBER
80  HOME
90  PRINT "THIS PROGRAM WILL FIND THE ROOTS OF A"
100  PRINT "POLYNOMIAL EQUATION BY USING CONTE'S"
110  PRINT "VERSION OF NEWTON'S METHOD"
120  PRINT
130  PRINT "ENTER THE DEGREE OF THE POLYNOMIAL"
140  INPUT "(MAXIMUM DEGREE IS 10) ";N
150  PRINT
160  PRINT "THE COEFFICIENT OF THE X^NTH TERM"
170  PRINT "IS A(N)"
180  PRINT
190  REM  *** ENTER THE COEFFICIENTS
200  FOR X = N TO 0 STEP  - 1
210  PRINT "ENTER A("X")";
220  INPUT " ";A(X)
230  NEXT X
240  PRINT
250 I = 1
260  INPUT "ENTER THE INITIAL GUESS ";Z
270  REM  *** SEND OUTPUT TO PRINTER
280  PR# 1
290  REM  *** PRINT EQUATION
300  PRINT  TAB( 10);" ";
310  FOR X = N TO 2 STEP  - 1
320  PRINT A(X)"X^"X" + ";
330  NEXT X
340  PRINT A(1)"X + "A(0)
350  PRINT
360  PRINT "IT #","   ROOT"
370  PRINT
380  REM  *** CONTE'S ALGORITHM
390 B(N) = A(N)
400 C(N) = B(N)
410  FOR K = N - 1 TO 1 STEP  - 1
420 B(K) = A(K) + Z * B(K + 1)
430 C(K) = B(K) + Z * C(K + 1)
440  NEXT K
450 B(0) = A(0) + Z * B(1)
460 Z1 = Z - B(0) / C(1)
470  IF  ABS (Z1 - Z) < 1E - 6 THEN 520
480  PRINT " "I,Z1
490 I = I + 1
500 Z = Z1
510  GOTO 390
520  PRINT
```

PROGRAM 4.5 (continued)

```
530  PRINT "THE ROOT IS "Z1
540  PR# 0
550  END
```

TABLE 4.9

Equation	No. of Iter.	Newton	No. of Iter.	Conte
$x^3 - 2x^2 - 4 + 2$	1	2	1	2
$x^3 + 2x^2 - 5x - 6$	4	-1	4	-1
$4x^3 - 3x$	4	-0.866025404	4	-0.866025404
$x^3 + 6x^2 + 11x + 6$	6	-1	6	-1
$x^3 + 4x^2 + 5x + 2$	17	-1.00000405	16	-0.999978767
$x^3 + 3x^2 + 3x + 1$	37	-1.00040349	19	-0.999530169
$x^4 + 10x^3 + 35x^2 + 50x + 24$	6	-1	6	-1
$8x^4 - 8x^2 + 1$	3	-0.923879532	3	-0.923879533
$x^4 + 4x^3 + 6x^2 + 4x + 1$	19	-0.995481137	18	-0.994617309
$16x^5 - 20x^3 + 5x$	3	-0.951056516	3	-0.951056516
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	6	-1	6	-0.999999841
$x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$	32	-1.00110733	20	-0.986005838

Except for the cases involving multiple roots, there is no appreciable difference in either the number of iterations or the calculated value of the root. Where multiple roots are involved, Conte's method required fewer iterations, but Newton's method provided the most accurate answer.

CHAPTER V

ALTERNATIVES TO NEWTON'S METHOD

Since Newton's method does not work well when multiple roots are present, alternative methods were sought. One such method is the bisection method. The bisection method is probably one of the oldest iterative methods in existence. Briefly, an interval is found such that $x_1 < x < x_2$, and that $f(x_1)f(x_2) < 0$. That is, at one boundary of the interval f(x) is positive, and at the other boundary f(x) is negative. This interval is then bisected to find x_3, $x_3 = (x_1+x_2)/2$. If $f(x_3) = 0$, then x_3 is a root. Otherwise, the boundaries are changed by moving x_3 to x_1 if $f(x_1)$ has the same sign as $f(x_3)$, or to x_2 if $f(x_2)$ has the same sign as $f(x_3)$. This procedure is repeated until $f(x_3) = 0$, or rather $f(x_3) < \epsilon$. The function f must be continuous on the interval $[x_1, x_2]$.

The bisection method, however, is not without problems. When evaluating the polynomial, if r_3 is sufficiently close to the root, $P(r_3)$ will occasionally have the wrong sign. That is, when $r_3 \approx r$, $P(r_3) \approx 0$. However, due to round-off error incurred while evaluating the polynomial, $P(r_3)$ will be positive rather than negative, or vice versa. This will cause the wrong boundary to be reset. Thus, while $P(r_1)P(r_2) < 0$, $r_1 < r < r_2$ is no longer a true condition. The interval $[r_1, r_2]$ no longer contains the root. It will, therefore, be impossible to obtain a very good approximation to the root. This is illustrated in Table 5.1.

TABLE 5.1

$$P(x) = x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$$

IT #	r_1	r_2	$P(r_1)$	$P(r_2)$
1	-2	0.01	-1	1.05101005
2	-0.995	0.01	-4.65661287 E-10	1.05101005
3	-0.995	-0.4925	-4.65661287 E-10	0.033665125
4	-0.995	-0.74375	-4.65661287 E-10	1.1048899 E-3
5	-0.995	-0.869375	-4.65661287 E-10	3.80305573 E-5
6	-0.995	-0.9321875	-4.65661287 E-10	1.43353827 E-6
7	-0.995	-0.96359375	-4.65661287 E-10	6.44940883 E-8
8	-0.995	-0.979296875	-4.65661287 E-10	3.7252903 E-9
9	-0.995	-0.987148438	-4.65661287 E-10	6.98491931 E-10
10	-0.991074219	-0.987148438	-2.32830644 E-9	6.98491931 E-10
11	-0.989111328	-0.987148438	-9.31322575 E-10	6.98491931 E-10
12	-0.989111328	-0.988129883	-9.31322575 E-10	1.16415322 E-9
13	-0.989111328	-0.988620606	-9.31322575 E-10	4.65661287 E-10
14	-0.988865967	-0.988620606	-4.65661287 E-10	4.65661287 E-10
15	-0.988743287	-0.988620606	-4.65661287 E-10	4.65661287 E-10

In the first iteration, $r_3 = 0.995 \simeq r$. However, $P(r_3) =$ -4.65661287 E-10. Thus, while r_3 is to the right of r, the sign of $P(r_3)$ indicates it should be to the left. The interval used for the second iteration, therefore, does not contain the root. Theoretically, this should not happen. This error can be compensated for by increasing the value of ϵ sufficiently to prevent $P(r_3) \simeq 0$. There is also a corresponding loss in the accuracy of the answer as a result of this compensation. However, the two-place accuracy with the change is better than the one-place accuracy without it. Program 5.1 will find the root of a polynomial equation using the bisection method. Table 5.2 summarizes some of the results obtained from Program 5.1. In all cases, r_1 and r_2 were chosen so that $r_3 \neq r$ on the first iteration.

PROGRAM 5.1

```
10  REM  *** BISECTION ***
20  REM  *** A(X) = COEFFICIENTS OF POLYNOMIAL
30  REM  *** N = DEGREE OF POLYNOMIAL
40  REM  *** LL = LOWER LIMIT
50  REM  *** UL = UPPER LIMIT
60  REM  *** X = MIDDLE VALUE
70  REM  *** LB = F(LL)
80  REM  *** RB = F(UL)
90  REM  *** MB = F(X)
100  REM  *** R = VALUE FOR SUBROUTINE
110  REM  *** I = ITERATION NUMBER
120  HOME
130  PRINT "THIS PROGRAM USES THE BISECTION METHOD"
140  PRINT "TO FIND THE ROOTS OF A POLYNOMIAL"
150  PRINT
160  PRINT "ENTER THE DEGREE OF THE POLYNOMIAL"
170  INPUT "(MAXIMUM DEGREE IS 10) ";N
180  PRINT
190  PRINT "THE COEFFICIENT OF THE X^NTH TERM"
200  PRINT "IS A(N)"
210  PRINT
220  REM  *** ENTER COEFFICIENTS
230  FOR X = N TO 0 STEP  - 1
240  PRINT "ENTER A(";X;")";
250  INPUT " ";A(X)
260  NEXT X
270  PRINT
280  INPUT "ENTER THE LEFT BOUND OF THE INTERVAL    >
     ";LL
290  PRINT
300  INPUT "ENTER THE RIGHT BOUND OF THE INTERVAL   >
     ";UL
310  REM  *** CHECK TO SEE IF INTERVAL CONTAINS ROOT
320 R = LL
330  GOSUB 2000
340 LB = Y
350 R = UL
360  GOSUB 2000
370 RB = Y
380  IF  SGN (LB) =  -  SGN (RB) THEN 430
390  IF  ABS (LB) < 1E - 9 THEN R = LL: GOTO 680
400  IF  ABS (RB) < 1E - 9 THEN R = UL: GOTO 680
410  PRINT "INTERVAL DOES NOT CONTAIN A ROOT"
```

PROGRAM 5.1 (continued)

```
420  GOTO 280
430  REM  *** SEND OUTPUT TO PRINTER
440  PR# 1
450  REM  *** PRINT EQUATION AND HEADINGS
460  PRINT  TAB( 10)" ";
470  FOR X = N TO 2 STEP  - 1
480  PRINT A(X)"X^"X" + ";
490  NEXT X
500  PRINT A(1)"X + "A(0)
510  PRINT
520  PRINT "IT #","    ROOT";"      F(X)"
530  PRINT
540 I = 1
550  REM  *** CALCULATE MIDPOINT OF INTERVAL
560 X = (LL + UL) / 2
570 R = X
580  GOSUB 2000
590 MB = Y
600  REM  *** CHECK FOR CLOSENESS OF ROOT
610  IF  ABS (MB) < 1E - 9 THEN 680
620  REM  *** RESET BOUNDS
630  IF  SGN (LB) =  SGN (MB) THEN LB = MB:LL = X
640  IF  SGN (RB) =  SGN (MB) THEN RB = MB:UL = X
650  PRINT "  "I,R,Y
660 I = I + 1
670  GOTO 560
680  PRINT
690  PRINT "THE ROOT IS ";R
700  PR# 0
999  END
2000  REM  *** EVALUATE F(X)
2010 Y = A(N) * R
2020  FOR X1 = N - 1 TO 1 STEP  - 1
2030 Y = (Y + A(X1)) * R
2040  NEXT X1
2050 Y = Y + A(0)
2060  RETURN
```

TABLE 5.2

Equation	Interval	No. of Iter.	Actual Root	Computed Root
$x^3 - 2x^2 - x + 2$	-1.5, -0.4	26	-1	-1
$x^3 + 2x^2 - 5x - 6$	-1.5, -0.4	29	-1	-1
$4x^3 - 3x$	-1.0, -0.5	29	-1	-1
$x^3 + 6x^2 + 11x + 6$	-1.5, -0.4	30	-0.8660254	-0.866025404
$x^3 + 4x^2 + 5x + 2$	-2.5, -1.4	28	-2	-2
$x^3 + 3x^2 + 3x + 1$	-1.5, -0.4	9	-1	-0.999414063
$x^4 + 10x^3 + 35x^2 + 50x + 24$	-1.5, -0.4	29	-1	-1
$8x^4 - 8x^2 + 1$	-1.0, -0.5	32	-0.9238795	-0.923879533
$x^4 + 4x^3 + 6x^2 + 4x + 1$	(bisection method not appropriate)			
$16x^5 - 20x^3 + 5x$	-1.0, -0.8	27	-0.9510565	-0.951056516
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	-1.5, -0.4	5	-1	-0.984375
$x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$	-1.5, -0.4	29	-1	-1

In general, the bisection method requires more iterations than does Newton's method. It, too, does not work well when multiple roots are present. In fact, it will not work at all in cases such as $(x+1)^4 = P(x)$ where the function only touches, rather than crosses, the x-axis.

As another illustration of the types of problems which may occur as a result of round-off error, the following example is given. The root printed as being used on the 26th, 27th, and 28th iteration in the equation $x^3 + 4x^2 + 5x + 2 = P(x)$ was -2. However, the values printed for P(x) were -1.86264515 E-9, 4.19095159 E-9, and 0, respectively. Similar examples are given in Table 5.3.

TABLE 5.3

Equation	It. No.	Root	P(x)
$x^3 - 2x^2 - x + 2$	26	-1	-5.58793545 E-9
	27	-0.999999994	3.77185643 E-8
	28	-0.999999998	1.25728548 E-8
	29	-1	0
$x^3 + 2x^2 - 5x - 6$	26	-1	5.58793545 E-9
	27	-0.999999994	-3.7252903 E-8
	28	-0.999999998	-1.3038516 E-8
	29	-1	0
$x^4 + 10x^3 + 35x^2 + 50x + 24$	26	-1	-7.4505806 E-9
	27	-0.999999994	3.7252903 E-8
	28	-0.999999998	1.49011612 E-8
	29	-1	0
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	26	-1	-5.96046448 E-8
	27	-0.999999994	1.49011612 E-7
	28	-0.999999994	5.96046448 E-8
	29	-1	0

TABLE 5.3 (continued)

Equation	It. No.	Root	P(x)
$8x^4 - 8x^2 + 1$	28	-0.923879532	-4.19095159 E-9
	29	-0.923879533	5.3551048 E-9
	30	-0.923879533	1.62981451 E-9
	31	-0.923879532	-1.39698386 E-9
	32	-0.923879533	1.62981451 E-9
	⋮	⋮	⋮
$16x^5 - 20x^3 + 5x$	26	-0.951056514	3.01151737 E-8
	27	-0.951056516	8.85740403 E-9
	28	-0.951056516	-3.54296162 E-9
	29	-0.951056516	1.77148081 E-9
	30	-0.951056516	-3.54296162 E-9
	⋮	⋮	⋮

It should be noted, however, that although the final computed root shown for the fourth and fifth degree Chebyshev polynomial equations were the same as those obtained by using Newton's method, the procedure used for the bisection method did not terminate naturally.

The secant method is similar to Newton's method. This method uses the slope of the line drawn between two points on the graph to approximate the slope of the tangent line. The general formula is as follows:

$$r_{k+1} = r_k - P(r_k)/s$$

where s is the slope of the line and $s = [P(r_k) - P(r_{k-1})]/(r_k - r_{k-1})$.

Program 5.2 will compute the root of a polynomial equation using the secant method.

PROGRAM 5.2

```
10  REM  *** SECANT ***
20  REM  *** R1 = FIRST APPROXIMATION
30  REM  *** R2 = SECOND APPROXIMATION
40  REM  *** R3 = NEW APPROXIMATION
50  REM  *** F1 = F(R1)
60  REM  *** F2 = F(R2)
70  REM  *** S = SLOPE
80  HOME
90  PRINT "THIS PROGRAM WILL FIND THE ROOT OF A"
100  PRINT "POLYNOMIAL USING THE SECANT METHOD"
110  PRINT
120  PRINT "ENTER THE DEGREE OF THE POLYNOMIAL"
130  INPUT "(MAXIMUM DEGREE IS 10) ";N
140  PRINT
150  REM  *** ENTER THE COEFFICIENTS
160  PRINT "THE COEFFICIENT OF THE X^NTH TERM "
170  PRINT "IS A(N)"
180  PRINT
190  FOR X = N TO 0 STEP  - 1
200  PRINT "ENTER A("X")";
210  INPUT " ";A(X)
220  NEXT X
230  PRINT
240  PRINT "THE SECANT METHOD REQUIRES TWO INITIAL"
250  PRINT "APPROXIMATIONS TO THE ROOT"
260  PRINT
270  INPUT "ENTER APPROXIMATION #1 ";R1
280  INPUT "ENTER APPROXIMATION #2 ";R2
290 I = 1
300  REM  *** SEND OUTPUT TO PRINTER
310  PR# 1
320  REM  *** PRINT EQUATION AND HEADINGS
330  PRINT  TAB( 10)" ";
340  FOR X = N TO 2 STEP  - 1
350  PRINT A(X)"X^"X" + ";
360  NEXT X
370  PRINT A(1)"X + "A(0)
380  PRINT
390  PRINT "IT #","ROOT 1","ROOT 2"
400  PRINT
410  REM  *** EVALUATE F(R1)
420 R = R1
430  GOSUB 2000
440 F1 = Y
450  REM  *** EVALUATE F(R2)
460 R = R2
470  GOSUB 2000
```

PROGRAM 5.2 (continued)

```
480 F2 = Y
490  REM  *** CALCULATE SLOPE
500 S = (F2 - F1) / (R2 - R1)
510  REM  *** CALCULATE NEW APPROXIMATION
520 R3 = R1 - F1 / S
530  REM  *** CHECK FOR CLOSENESS
540  IF  ABS (R3 - R1) < 1E - 6 THEN 610
550  REM  *** RESET R1 AND R2
560 R2 = R1
570 R1 = R3
580  PRINT "  "I,R1,R2
590 I = I + 1
600  GOTO 420
610  PRINT
620  PRINT "THE ROOT IS ";R3
630  PR# 0
1999  END
2000  REM  *** EVALUATE F(R)
2010 Y = A(N) * R
2020  FOR X = N - 1 TO 1 STEP  - 1
2030 Y = (Y + A(X)) * R
2040  NEXT X
2050 Y = Y + A(0)
2060  RETURN
```

In order to investigate whether the location of the two initial approximations with respect to the root affects the number of iterations required to find the root, various approaches with three representative equations were compared. Table 5.4 summarizes the results.

TABLE 5.4

$$P(x) = x^3 + 6x^2 + 11x + 6$$

Approx. 1	Approx. 2	No. of Iter.	Actual Root	Computed Root
-4	0	1	-2	-2
-2.9	-1.1	2	-2	-1.99999998
0	1	9	-1	-1
-4	-5	9	-3	-3

TABLE 5.4 (continued)

$$P(x) = x^3 + 4x^2 + 5x + 2$$

Approx. 1	Approx. 2	No. of Iter.	Actual Root	Computed Root
-1.4	0	19	-1	-1.00001519
-1.6	0	14	-2	-2
-3	-2.4	8	-2	-2.00000004
-3	0	1	-1	-1
0	1	24	-1	-0.999992713
-3	-4	9	-2	-2

$$P(x) = x^3 + 3x^2 + 3x + 1$$

Approx. 1	Approx. 2	No. of Iter.	Actual Root	Computed Root
-2	1	26	-1	-1.00051558
0	-3	25	-1	-0.999244997
-3	0	26	-1	-0.99964703
-2	0	1	-1	-1
0	1	26	-1	-0.999561218
-3	-2	28	-1	-1.00052643

In general, fewer iterations were required when the two initial approximations surrounded the root. For all practical purposes, however, this is not a viable choice. But for the purposes of comparison, the initial approximations given for use in Program 5.2 were the same as those used in Program 5.1. Table 5.5 summarizes the results obtained from Program 5.2.

TABLE 5.5

Equation	r_1, r_2	No. of Iter.	Actual Root	Computed Root
$x^3 - 2x^2 - x + 2$	-1.5, -0.4	7	-1	-1.00000003
$x^3 + 2x^2 - 5x - 6$	-1.5, -0.4	5	-1	-1.00000003
$4x^3 - 3x$	-1.0, -0.5	7	-0.8660254	-0.866025404
$x^3 + 6x^2 + 11x + 6$	-1.5, -0.4	14	-1	-1
$x^3 + 4x^2 + 5x + 2$	-2.5, -1.4	23	-1	-0.999998388
$x^3 + 3x^2 + 3x + 1$	-1.5, -0.4	20	-1	-1.00063794
$x^4 + 10x^3 + 35x^2 + 50x + 24$	-1.5, -0.4	7	-3	-2.99999999
$8x^4 - 8x^2 + 1$	-1.0, -0.5	10	-0.9238795	-0.923879532
$x^4 + 4x^3 + 6x^2 + 4x + 1$	-1.5, -0.4	27	-1	-1.00415797
$16x^5 - 20x^3 + 5x$	-1.0, -0.8	7	-0.9510565	-0.951056516
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	-1.5, -0.4	9	-2	-2.00000001
$x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$	-1.5, -0.4	18	-1	-1.01567717

Again, more iterations are required when multiple roots are present. The computed root in these cases is not as accurate as when no multiple roots are present. The secant method has an advantage over the bisection method in that it can compute the root when the function does not cross the x-axis.

The secant method does have a disadvantage in that the two initial approximations submitted cannot also be roots of the equation.

Nor can they be values such that $P(r_1) = P(r_2)$. If this occurs, the slope will be zero, and the secant method will no longer work.

The last alternative method to be presented is a variation of Newton's method. This method is provided by Haggerty in [6, p. 101], and is not dependent upon the multiplicity of the root.

If $P(x) = (x-r)^m q(x) = 0 \qquad (q(r) \neq 0)$

then $P'(x) = (x-r)^m q'(x) + m(x-r)^{m-1} q(x)$

and
$$\frac{P(x)}{P'(x)} = \frac{(x-r)^m q(x)}{(x-r)^m q'(x) + m(x-r)^{m-1} q(x)}$$

$$= \frac{(x-r)^{m-1}[(x-r)q(x)]}{(x-r)^{m-1}[(x-r)q'(x) + mq(x)]}$$

$$= \frac{(x-r)q(x)}{(x-r)q'(x) + mq(x)}$$

Setting this equation equal to zero yields $x = r$ as a root. (For examples of the graph of $f(x) = P(x)/P'(x)$, the reader is referred again to Graphs 4.1, 4.2, and 4.3.)

Let $F(x) = P(x)/P'(x)$ then:

$$F'(x) = \frac{P'(x)P'(x) - P(x)P''(x)}{[P'(x)]^2}$$

$$= 1 - \frac{P(x)P''(x)}{[P'(x)]^2}$$

Since $F(x)$ has only one root at $x = r$, $F(x)$ may be substituted for $P(x)$ in Newton's method. Thus $r_{k+1} = r_k - F(r_k)/F'(r_k)$ where $F(r_k) = P(r_k)/P'(r_k)$ and $F'(r_k) = 1 - P(r_k)P''(r_k)/[P'(r_k)]^2$.

Program 5.3 will compute the root of a polynomial equation using Haggerty's version of Newton's method. Table 5.5 summarizes some of the results obtained from Program 5.3.

PROGRAM 5.3

```
10  REM  *** NEWTON (HAGGERTY) ***
20  REM  *** N = DEGREE OF THE POLYNOMIAL
30  REM  *** A(X) = COEFFICIENTS OF POLYNOMIAL
40  REM  *** B(X) = FIRST DERIVATIVE
50  REM  *** C(X) = SECOND DERIVATIVE
60  REM  *** R = INITIAL APPROXIMATION
70  REM  *** R1 = F(R)
80  REM  *** R2 = F'(R)
90  REM  *** R3 = F''(R)
100  REM  *** R4 = SUCCESSIVE APPROXIMATION
110  REM  *** F = F(X)
120  REM  *** FP = F'(X)
130  REM  *** I = ITERATION NUMBER
140  HOME
150  PRINT "THIS PROGRAM WILL FIND THE ROOT OF A"
160  PRINT "POLYNOMIAL USING HAGGERTY'S VERSION OF"
170  PRINT "NEWTON'S METHOD"
180  PRINT
190  PRINT "ENTER THE DEGREE OF THE POLYNOMIAL"
200  INPUT "(MAXIMUM DEGREE IS 10) ";N
210  PRINT
220  REM  *** ENTER COEFFICIENTS
230  PRINT "THE COEFFICIENT OF THE X^NTH TERM"
240  PRINT "IS A(N)"
250  PRINT
260  FOR X = N TO 0 STEP  - 1
270  PRINT "ENTER A("X")";
280  INPUT " ";A(X)
290  NEXT X
300  PRINT
310  INPUT "ENTER THE INITIAL GUESS ";R
320  PRINT
330 I = 1
340  REM  *** SEND OUTPUT TO PRINTER
350  PR# 1
360  REM  *** PRINT EQUATION AND HEADINGS
370  PRINT  TAB( 10)" ";
380  FOR X = N TO 2 STEP  - 1
390  PRINT A(X)"X^"X" + ";
400  NEXT X
410  PRINT A(1)"X + "A(0)
420  PRINT
430  PRINT "IT #","    ROOT"
440  PRINT
450  REM  *** COMPUTE FIRST DERIVATIVE
460  GOSUB 3000
470  REM  *** COMPUTE SECOND DERIVATIVE
480  GOSUB 4000
490  REM  *** EVALUATE F(R)
500  GOSUB 2000
510 R1 = Y
520  REM  *** EVALUATE F'(R)
530  GOSUB 3500
```

PROGRAM 5.3 (continued)

```
540 R2 = Y
550  REM  *** EVALUATE F''(R)
560  GOSUB 4500
570 R3 = Y
580  REM  *** CALCULATE F AND FP
590 F = R1 / R2
600 FP = 1 - R1 * R3 / (R2 * R2)
610  REM  *** CALCULATE NEW APPROXIMATION
620 R4 = R - F / FP
630  REM  *** CHECK FOR CLOSENESS
640  IF  ABS (R4 - R) < 1E - 6 THEN 710
650 R = R4
660  PRINT "  "I,R4
670 I = I + 1
680  GOSUB 2000
690  IF  ABS (Y) < 1E - 6 THEN 710
700  GOTO 510
710  PRINT
720  PRINT "THE ROOT IS ";R4
730  PR# 0
1999  END
2000  REM  *** EVALUATE F(R)
2010 Y = A(N) * R
2020  FOR X1 = N - 1 TO 1 STEP  - 1
2030 Y = (Y + A(X1)) * R
2040  NEXT X1
2050 Y = Y + A(0)
2060  RETURN
3000  REM  *** COMPUTE FIRST DERIVATIVE
3010  FOR X = N TO 1 STEP  - 1
3020 B(X - 1) = A(X) * X
3030  NEXT X
3040  RETURN
3500  REM  *** EVALUATE F'(R)
3510  IF N = 1 THEN 3570
3520 Y = B(N - 1) * R
3530  IF N = 2 THEN 3570
3540  FOR X1 = N - 2 TO 1 STEP  - 1
3550 Y = (Y + B(X1)) * R
3560  NEXT X1
3570 Y = Y + B(0)
3580  RETURN
4000  REM  *** COMPUTE SECOND DERIVATIVE
4010  FOR X = N TO 2 STEP  - 1
4020 C(X - 2) = B(X - 1) * (X - 1)
4030  NEXT X
4040  RETURN
4500  REM  *** EVALUATE F''(R)
4510  IF N = 1 THEN Y = 0: RETURN
4520  IF N = 2 THEN 4580
4530 Y = C(N - 2) * R
4540  IF N = 3 THEN 4580
4550  FOR X1 = N - 3 TO 1 STEP  - 1
```

PROGRAM 5.3 (continued)

```
4560 Y = (Y + C(X1)) * R
4570  NEXT X1
4580 Y = Y + C(0)
4590  RETURN
```

TABLE 5.6

Equation	Init. Approx.	No. of Iter.	Actual Root	Computed Root
$x^3 - 2x^2 - x + 2$	0	5	1	1.0000001
$x^3 + 2x^2 - 5x - 6$	0	4	-1	-1
$4x^3 - 3x$	-1	4	-0.8660254	-0.866025404
$x^3 + 6x^2 + 11x + 6$	0	7	-1	-1
$x^3 + 4x^2 + 5x + 2$	0	3	-1	-1.00003033
$x^3 + 3x^2 + 3x + 1$	0	1	-1	-1
$x^4 + 16x^3 + 35x^2 + 50x + 24$	0	6	-2	-2.00000036
$8x^4 - 8x^2 + 1$	-1	4	-0.9238795	-0.923879533
$x^4 + 4x^3 + 6x^2 + 4x + 1$	0	1	-1	-1
$16x^5 - 20x^3 + 5x$	-1	4	-0.9510365	-0.951056516
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	0	5	-2	-2.00000004
$x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$	0	1	-1	-1

The number of iterations required when multiple roots are present decreased significantly in Haggerty's version of Newton's method. In the other cases, Haggerty's version required the same, or perhaps one additional, number of iterations to find the root. The roots obtained by using Haggerty's method appeared to have the same degree of accuracy as those obtained by using Newton's method.

For purposes of comparing the results of the four principal methods discussed, Table 5.7 summarizes the results obtained by using Newton's, Haggerty's, the secant, and the bisection methods.

TABLE 5.7

Equation	Init. Approx.	No. of Iter.	Newton	No. of Iter.	Haggerty
$x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$	0	32	-1.00110733	1	-1
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	0	7	-1	5	-2.00000004
$x^5 + 5x^4 - 25x^3 - 125x^2 + 144x + 720$	0	1	-5	7	-3
$16x^5 - 20x^3 + 5x$	-1	4	-0.951056516	4	-0.95105616

Equation	Init. Approx.	No. of Iter.	Bisection	No. of Iter.	Secant
$x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$	-1.5, -0.4	5	-0.984375	17	-1.01567711
$x^5 + 15x^4 + 85x^3 + 225x^2 + 274x + 120$	-1.5, -0.4	29	-1	9	-2.00000008
$x^5 + 5x^4 - 25x^3 - 125x^2 + 144x + 720$	-3.5, -2.4	28	-3	10	-3.00000001
$16x^5 - 20x^3 + 5x$	-1.0, -0.8	30	-0.951056516	7	-0.951056516

Although the results will not be presented in this thesis, the four programs used to generate Table 5.7 were translated into FORTRAN. The results obtained on the microcomputer were very close (to three significant figures) to those obtained using FORTRAN in double-precision.

CHAPTER VI

SUMMARY

The purpose of this thesis has not been to show how the "tried-and true" methods of solving polynomial equations may be adapted for use on the microcomputer. Nor has it been to develop new methods. Rather, the purpose has been to show that while the methods already in existence may be adapted, they should not be blindly adapted.

All methods do not work equally well for all types of equations. The user should, therefore, be aware of some of the types of equations, and some of the areas in solving equations in general, in which problems may occur. With this information, steps can be taken to avoid, or to compensate for, these problems.

Specifically, programs that will solve polynomial equations of degree four or less are given. While no formula exists for solving polynomial equations of degree five or more, there are many iterative methods available for approximating the roots of these equations. Programs for solving equations using Newton's, the secant, and the bisection methods are given.

When using any iterative method, the task of evaluating $P(x)$ is highly critical. As $x \longrightarrow r$, the amount of round-off error increases significantly. Thus, no method is more accurate than its evaluation of $P(x)$. Polynomial equations which contain multiple roots are especially difficult to evaluate.

Polynomials are ill-conditioned. That is, a small change in one of the coefficients may produce a large change in the roots. The equations illustrated in this thesis are of degree five or less. A significant amount of change can be detected with these. Higher degree equations would be affected even more. This is important not only when working with depressed equations (as illustrated in Table 4.6), but also when entering irrational numbers or repeating decimals as coefficients.

Other problems may be encountered when solving equations with six or more complex roots, or when working with equations with complex coefficients. Although not presented in this thesis, some methods, such as Muller's, will develop intermediate complex iteratives. Special care will have to be used to work with these on a microcomputer.

Since programs for iterative methods are normally verified by using equations with known roots, the user needs to exercise extreme caution in attempting to find the roots of an equation whose roots are unknown. Thus, while the user may think the correct roots have been obtained, the error incurred by the machine during the procedure may have circumvented ever finding the correct roots. Therefore, it is extremely important for the user to be aware that problems may occur, and to be conscientious enough to look for them.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Atkinson, Kendall E. An Introduction to Numerical Analysis. New York: John Wiley & Sons, 1978.

2. Borofsky, Samuel. Elementary Theory of Equations. New York: The Macmillan Company, 1954.

3. Boyer, Carl B. A History of Mathematics. New York: John Wiley & Sons, Inc., 1968.

4. Conte, S. D., and de Boor, Carl. Elementary Numerical Analysis: An Algorithmic Approach. New York: McGraw-Hill Book Company, 1972.

5. Forsythe, George E., Malcolm, Michael A., and Moler, Cleve B. Computer Methods for Mathematical Computations. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1977.

6. Haggerty, Gerald B. Elementary Numerical Analysis with Programming. Boston: Allyn and Bacon, Inc., 1972.

7. Pennington, Ralph H. Introductory Computer Methods and Numerical Analysis. London: The Macmillan Company, 1970.

8. Toralballa, L. V. Theory of Functions. Columbus, Ohio: Charles E. Merrill Books, Inc., 1963.