COMPUTER METHODS FOR SOLVING POLYNOMIAL EQUATIONS

by

JUDITH ARLENE BENNETT

B. S., Bethany College, 1969

959-5358

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mathematics

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1972

Approved by:

[illegible]

Major Professor

TABLE OF CONTENTS

INTRODUCTION

Most polynomial equations encountered in a high school mathematics class are solved by exact methods such as factoring, completing the square, or applying an algebraic formula. Methods which are other than exact, called numerical methods, can be far more useful but are seldom brought to the attention of the student in standard algebra courses. These numerical methods enable one to calculate the roots of a polynomial equation by a method of successive approximations. For any given equation, a sequence of computational steps is constructed. If these steps are carried out with sufficient accuracy, they lead to as close an approximation to the solution as is desired. Thus, the original problem is transformed into a problem consisting only of simple arithmetic steps.

The problem of finding the zeros of a given polynomial equation arises frequently in mathematics, physics, and many branches of engineering. Rarely do these problems possess simple, analytical solutions. For higher degree polynomials, no general algebraic formulas are available for expressing the roots in terms of radicals. It is therefore of great importance that numerical methods for the solution of such equations be made available to the high school student.

A variety of numerical root-finding techniques are available--some have been known for thousands of years. However, prior to the advent of computers, such solutions were considered relatively inaccessible because of the extensive numerical calculations that were required. The capability for high speed calculations by the computer has changed the picture considerably. The use of procedures involving extensive numerical calculations is now both possible and practical, thus suggesting many different approaches to problems than would otherwise be considered. "Mathematicians have rediscovered processes that are particularly suited to a computer's capabilities. They have also devised new and unusual kinds of mathematics in certain cases."[1]

The computer can be an effective aid in the learning and enrichment of mathematics. Through the use of the computer, the student may become actively engaged in the exploration, problem-solving, and formal organizing aspects of mathematics. A statistical study developed by Kieran reveals that students studying quadratic functions with the aid of a computer scored significantly higher on a unit test than did students studying the same material using a traditional approach.[2]

The computer approach to solving polynomial equations provides the student with much more than a generalizable algorithm for approximating these solutions. As the numerical procedure is illustrated algebraically and geometrically, the teacher has an opportunity to introduce several important concepts such as sequences, convergence, limits, continuity, and the close unity of algebra and geometry. Once the student has mastered the numerical method, he can use the computer to obtain the desired solution of a polynomial equation. Then instead of spending all his study time doing tedious and meaningless hand calculations, he is able to use his result and move ahead to examine other types of equations and their solutions.

---

[1]Ladis D. Kovach, Computer-oriented Mathematics (San Francisco: Holden-Day, Inc., 1964), p. 3.

[2]Thomas E. Kieran, "Quadratic Equations--Computer Style," Mathematics Teacher, LXII (April, 1969), 309.

Although the problem of finding the roots of an equation on a computer is by no means trivial, it is a problem solvable in the majority of cases by straightforward application of simple procedures. The purpose of this paper is to examine and compare the available methods for finding the roots of polynomial equations and decide which methods would be most suitable for use in a computer-oriented high school mathematics class.

ISOLATING THE ROOTS

All methods for solving polynomial equations make use of some unknown quantity x to represent the actual value of the root. Since a computer cannot work with unknown quantities, a guess must be made and a numerical value assigned to x. Then, by using the proper computational scheme, the number can be systematically corrected so that it approaches nearer the true value of the root. This initial guess should be as close to the actual root as possible. Thus, some method of finding a rough approximation to the root must be employed.

This procedure of estimating the answer to a problem should not be restricted to problems which are solved on computers, but should instead be constantly applied to all mathematics. Kovach suggests that the guessed solution is part of good mathematics. "It produces a frame of mind which can more easily cope with the most difficult problems . . . and also provides an excellent check on the reasonableness of the solution when it is finally obtained. Finally, a good first guess will save time and effort and lead one to solutions of problems that are particularly difficult."[3]

Frequently, an approximation to the largest root of a polynomial equation may be obtained by taking the two highest order terms and solving them for the unknown.[4] Consider the equation

$$x^3 - 9x^2 + 5x - 6 = 0.$$

---

[3]Kovach, op. cit., p. 26.

[4]Forman S. Acton, Numerical Methods That Work (New York: Harper & Row, Publishers, 1970), p. 185.

Ignoring the last terms, the equation

$$x^3 - 9x^2 = 0$$

can be solved to obtain an approximation of $x = 9$. The actual value of the root to four significant places is 8.4945. Thus, the estimate is a reasonable one.

The theory behind this method of approximation is that for large x, the two highest-order terms will usually dominate. Therefore, if the largest root is smaller than 1, the theory does not hold true. The best estimates will be obtained when the actual value of the root is very large.

A similar procedure is available for estimating the smallest root of a given polynomial equation.[5] If the actual root is near zero, the high powers of x will be very small. Thus, the solution to the linear equation obtained by ignoring the higher powers will be a good approximation of the root. Note that this estimate can be easily found by taking the negative of the ratio of the constant and first-order coefficient.

Forsyth suggests a slightly more complicated procedure for estimating the smallest root.[6] Consider the equation

$$x^3 + 2x^2 + 10x - 20 = 0.$$

Neglecting the two highest-order terms, the resulting linear equation

$$10x - 20 = 0$$

can be solved, obtaining a value of 2 for x. Then, neglecting only the x^3 term and substituting 2 for one of the x's in the x^2 term

[5] Bruce W. Arden, An Introduction to Digital Computing (Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1963), p. 144.

[6] C. H. Forsyth, "A Simple and Effective Method of Solving a Polynomial," School Science and Mathematics, XV (December, 1915), 802-804

gives the linear equation

$$14x - 20 = 0$$

which has a solution of $x = 10/7$. Finally, substituting $10/7$ for two of the x's in x^3 and one of the x's in x^2 results in the linear equation

$$100/49\ x + 20/7\ x + 10x - 20 = 0.$$

This equation has a solution of $98/73$ which is approximately 1.34. The actual value of the root to the nearest hundredth is 1.37, so the estimate is a good one. This general procedure can be expanded to estimate the smallest root of a polynomial of any degree.

The previously mentioned procedures for estimating a root of a given polynomial apply only for special cases when the desired root is very small or very large. For obtaining a rough approximation to any or all of the roots of a given polynomial equation, the best device proves to be a graph of the given equation. Many clues to the solution of the equation may be obtained from such a graph, and its value cannot be overemphasized. Once the equation is plotted, the student need only estimate the abscissas of the points where the graph crosses the x-axis. The calculus student can also calculate the derivative of the function and investigate all maxima and minima points for possible roots.

Plotting a polynomial can be done very easily on the computer. The machine need only be programmed to evaluate the polynomial at regularly spaced values of x. If the value of the polynomial changes sign between any two points, then by the intermediate value theorem there exists at least one real root between the two points. What is desired is an interval that contains exactly one root. There are procedures due to Sturm which will give the exact number of roots between any two points, thus making it easier to isolate the roots. However, Sturm's process is too complicated to be understood by the average high school student. The most practical method of isolating a root of a polynomial is by running a stepping search on the computer. The flowchart in Figure 1 illustrates the use of such a stepping search, given that there is at least one real root between the two points a and b.

It should be noted that the choice of n is a matter of judgment on the part of the programmer. If n is too large, then h will be very small, and much time will be wasted examining intervals in which there are no zeros. On the other hand, if n is too small, then h will be very large, and some zeros might be missed. When a change of sign in the polynomial value is indicated, the coordinates of these two points are printed and the next interval is examined. These two points could later be used as starting values a and b for a new stepping search.

It can easily be seen that the stepping search requires the frequent evaluation of a polynomial for given values of x. Such an evaluation will be necessary in the use of any numerical method for finding the roots of a polynomial. Therefore, a subroutine should be developed by the student to accomplish this purpose.

One effective method for evaluating a polynomial on the computer is to rewrite the polynomial as a series of products involving x. For example, the equation

$$f(x) = 5x^4 + 4x^3 - 3x^2 + 7x - 2$$

could be rewritten as

$$f(x) = (((5x + 4)x - 3)x + 7)x - 2.$$

The flowchart in Figure 2 shows how a polynomial of degree n with

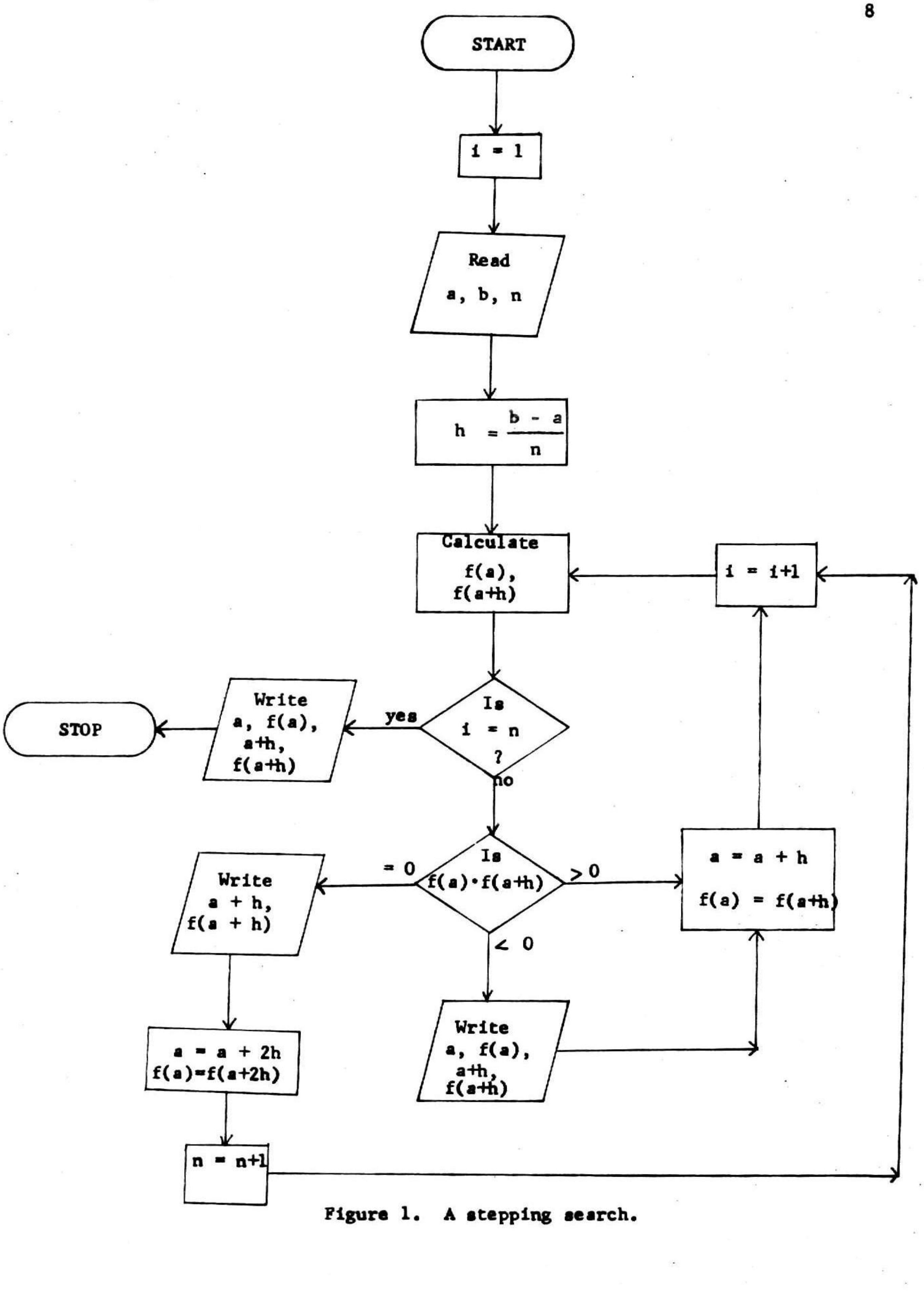


Figure 1. A stepping search.

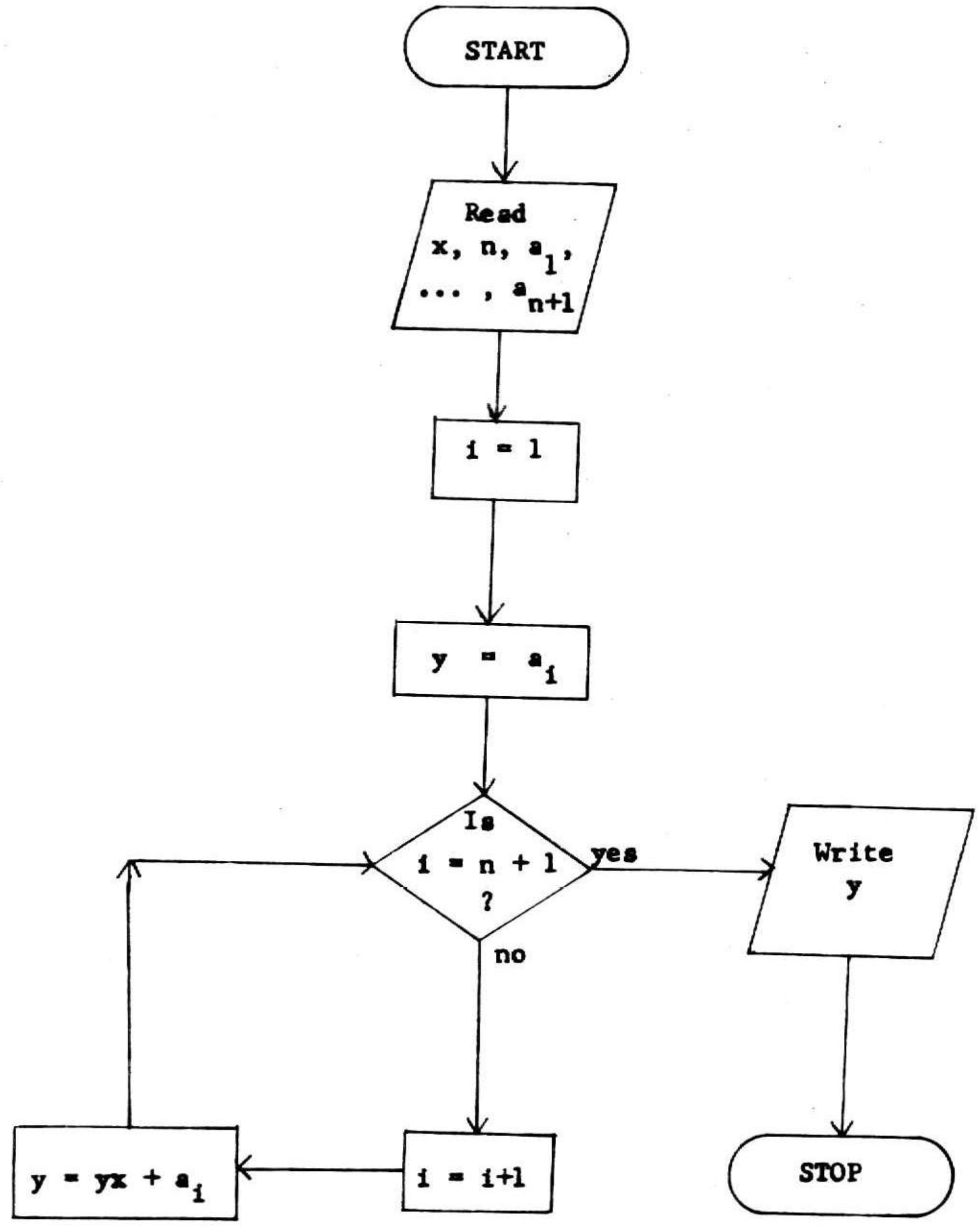


Figure 2. Evaluating a polynomial.

coefficients $a_1, \ldots, a_{n+1}$ can be evaluated for any given value of x.

The above procedure can be used to evaluate a polynomial of any degree, and the method is easily understandable by the student. However, a more efficient method which accomplishes the same purpose does exist. This method uses the process of synthetic division to evaluate the given polynomial, and it can easily be extended to evaluate the derivative of the polynomial also.[7] Let

$$f(x) = 3x^3 - 5x^2 - 2x + 1.$$

Then synthetic division can be used as follows to determine the value of $f(2)$:

$$\begin{array}{r|rrrr} 2 & 3 & -5 & -2 & 1 \\ & & 6 & 2 & 0 \\ \hline & 3 & 1 & 0 & 1 \end{array}$$

Thus, the quotient polynomial is

$$q(x) = 3x^2 + x,$$

and $f(2)$ is equal to the remainder which is 1. To find the derivative of the original polynomial at the point $x = 2$, another synthetic division can be performed using the quotient polynomial:

$$\begin{array}{r|rrr} 2 & 3 & 1 & 0 \\ & & 6 & 14 \\ \hline & 3 & 7 & 14 \end{array}$$

The derivative is equal to the remainder which is 14.

This general procedure can be applied to any polynomial

$$P(x) = a_1x^n + \ldots + a_nx + a_{n+1}$$

[7] S. D. Conte, Elementary Numerical Analysis (New York: NcGraw-Hill Book Company, 1965), p. 50.

to evaluate $P(z)$ and $P'(z)$ for any real number z. Dividing $P(x)$ by $(x - z)$ results in the quotient polynomial

$$Q(x) = b_1x^{n-1} + \dots + b_{n-1}x + b_n$$

with a remainder of b_{n+1}. Therefore, the original polynomial can be written as follows:

$$\begin{aligned}
P(x) &= Q(x)\cdot(x - z) + b_{n+1} \\
&= x\cdot Q(x) - z\cdot Q(x) + b_{n+1} \\
&= x(b_1x^{n-1} + \dots + b_{n-1}x + b_n) - z(b_1x^{n-1} + \dots + b_{n-1}x + b_n) + b_{n+1} \\
&= b_1x^n + (b_2 - zb_1)x^{n-1} + \dots + (b_n - zb_{n-1})x + (b_{n+1} - zb_n).
\end{aligned}$$

Equating the above equation to the equation

$$P(x) = a_1x^n + \dots + a_nx + a_{n+1}$$

results in the following recursion formulas:

$$\begin{aligned}
&b_1 = a_1 \\
&b_k = a_k + zb_{k-1}\ ,\ k = 2, \dots , n+1.
\end{aligned}$$

Then $P(z)$ is equal to b_{n+1}.

Applying the same procedure to the quotient polynomial results in these recursion formulas for the derivative:

$$\begin{aligned}
&c_1 = b_1 \\
&c_k = b_k + zb_{k-1}\ ,\ k = 2, \dots , n.
\end{aligned}$$

Then $P'(z) = Q(z) = c_n$. The use of this procedure to evaluate a given polynomial and its derivative for any value of x is explained by the flowchart in Figure 3.

Once a reasonable effort has been made to isolate all the roots, the student is ready to apply an iterative technique to determine the roots to as many decimal places as the limit of the computer will allow.

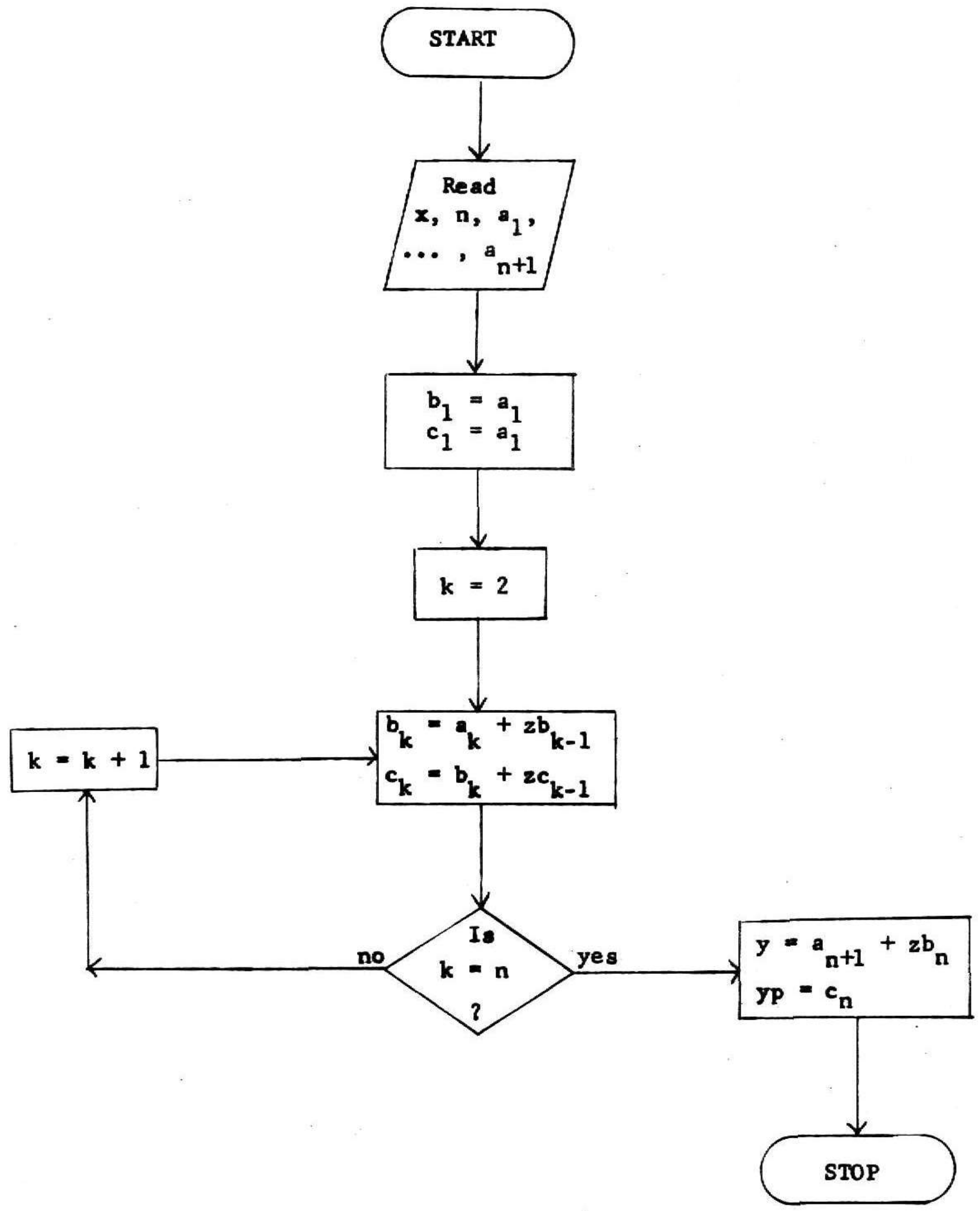


Figure 3. Evaluating a polynomial.

ROOT-FINDING METHODS

One of the simplest numerical methods for approximating a root of a polynomial is called the bisection method.[8] The requirements for use of this process are a polynomial $f(x)$ with real coefficients and two rational numbers a and b such that $f(a)$ is negative and $f(b)$ is positive. The numbers a and b can easily be found by applying a stepping search. Since every polynomial function is continuous, then there exists at least one point c between a and b such that $f(c)$ is a zero of the function.

The bisection procedure consists of halving the interval from a to b, choosing the half whose endpoints again bracket the zero, and repeating the process. First, the midpoint $m = (a+b)/2$ must be calculated. Then a subroutine can be called to evaluate $f(m)$. If the value of $f(m)$ is 0, then m is a root and the process is completed. Otherwise, $f(m)$ is either negative or positive. If the value is negative, then the root is between m and b. If the value is positive, then the root is between a and m. Either way, the root is bracketed within an interval half as large as the original. This process may be continued until the root is known to the desired accuracy.

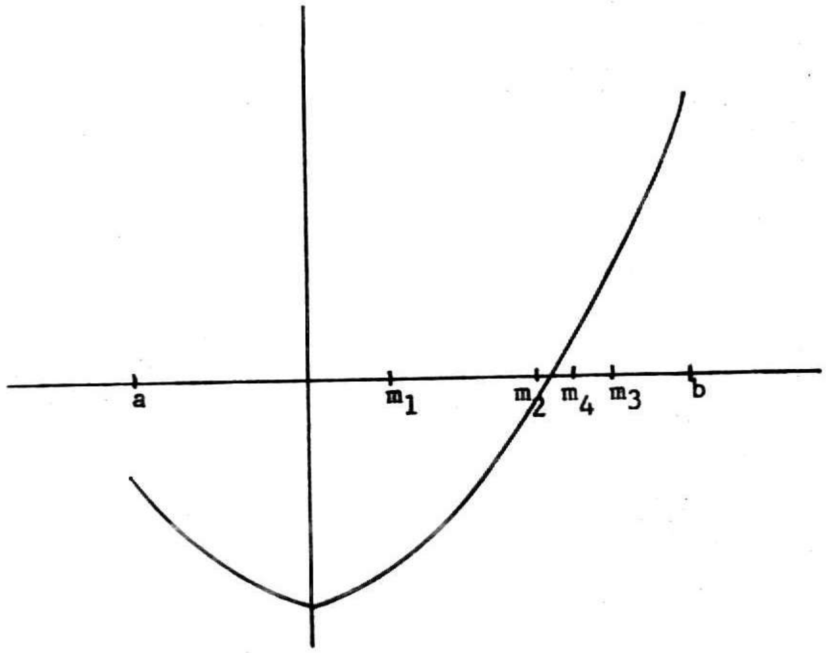
As an example, consider the graph of the equation

$$f(x) = x^2 - 2$$

between the two points $(-1,-1)$ and $(2,2)$. The following data resulted from only four applications of the bisection technique. The fourth bisection provided an estimate of 23/16, or approximately 1.44, for the root. The actual value of the root to three significant places

[8]Richard W. Hamming, Introduction to Applied Numerical Analysis (New York: McGraw-Hill Book Company, Inc., 1971), pp. 35-40.

is 1.414.



$a = -1$	$a = 1/2$	$a = 5/4$	$a = 5/4$
$b = 2$	$b = 2$	$b = 2$	$b = 13/8$
$m_1 = 1/2$	$m_2 = 5/4$	$m_3 = 13/8$	$m_4 = 23/16$
$f(m_1) < 0$	$f(m_2) < 0$	$f(m_3) > 0$	$f(m_4) > 0$

By using the process of bisection on the computer, an approximation can easily be calculated to any desired degree of accuracy, and with a minimum amount of effort. A flowchart of this process is illustrated in Figure 4. Notice that the process is terminated when the functional value of the midpoint differs from zero by some predetermined amount e. Thus, the bisection procedure is continued until the root is approximated to the desired degree of accuracy.

The biggest advantage of the bisection method lies in its simplicity and its assurance of convergence. Operation on the computer makes it seem fairly rapid, since each repetition reduces the maximum error by a factor of two. However, this method is extremely slow compared to other methods. In fact, it would be wise to incorporate a counter into the program that would terminate execution if the

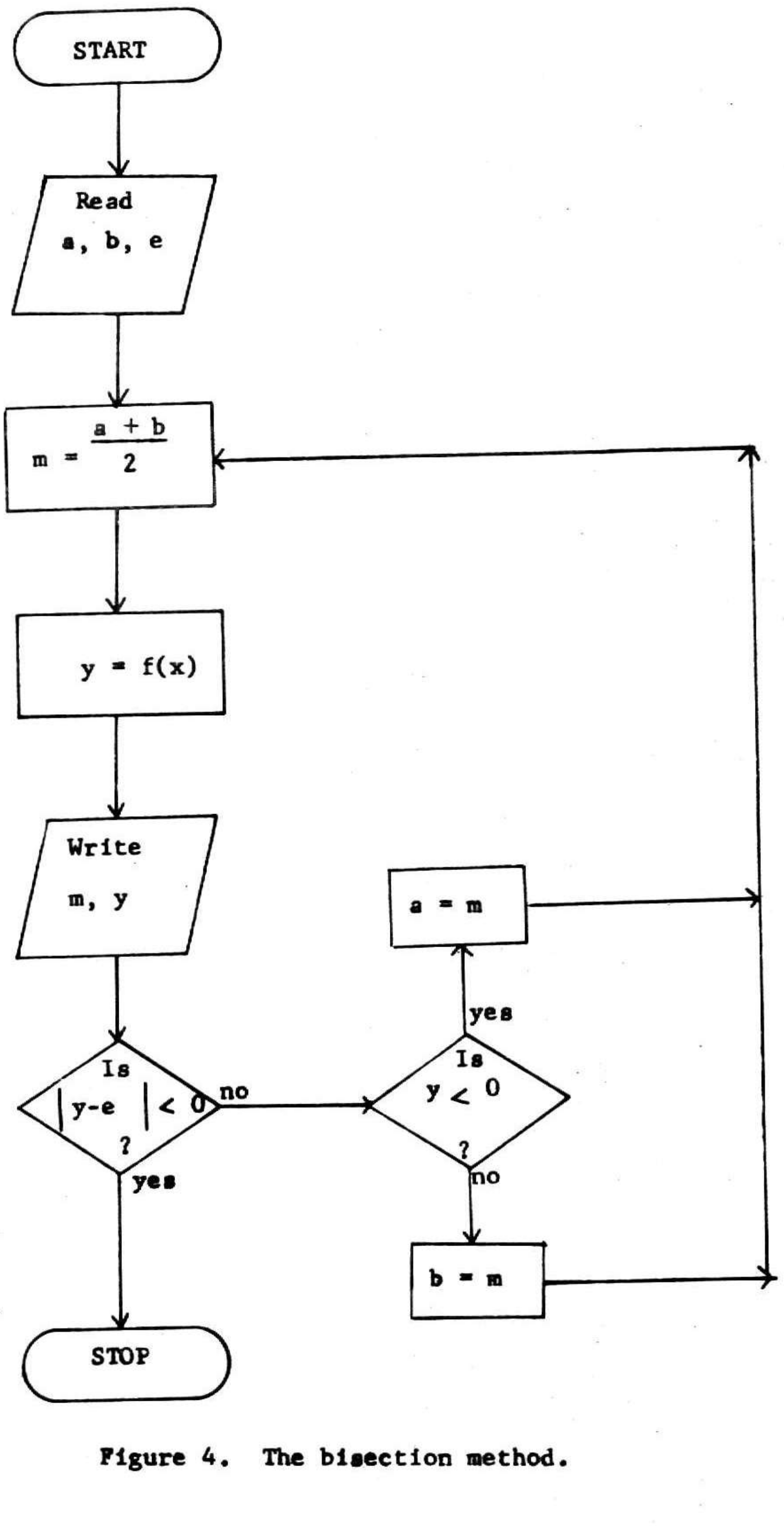


Figure 4. The bisection method.

desired degree of accuracy was not reached after a specified number of iterations.

The following output resulted from a computer program designed to find a zero of the equation

$$f(x) = x^3 - 2x - 5$$

to the fifth significant decimal place, starting with the interval from $a = 1$ to $b = 3$:

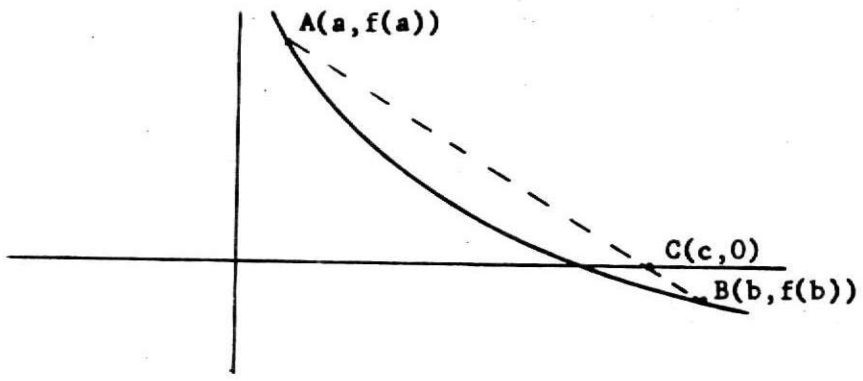
```
               m             f(m)

  1        2.000000       -1.000000
  2        2.500000        5.625000
  3        2.250000        1.890625
  4        2.125000        0.345703
  5        2.062500       -0.351318
  6        2.093750       -0.008941
  7        2.109375        0.166835
  8        2.101562        0.078562
  9        2.097656        0.034714
 10        2.095703        0.012862
 11        2.094727        0.001954
 12        2.094238       -0.003495
 13        2.094482       -0.000770
 14        2.094604        0.000591
 15        2.094543       -0.000089
 16        2.094574        0.000250
 17        2.094559        0.000080
 18        2.094551       -0.000004
```

Possibly the oldest method of finding a root of a polynomial equation is the method of false position. This method was known to the Egyptians between 1850 and 1650 B. C.[9] Again, an interval from a to b is sought in which $f(x)$ changes sign. As a first approximation to the root, consider the point C where the chord AB, formed by the line between points $(a,f(a))$ and $(b,f(b))$, crosses the x-axis. Assuming that the graph of $f(x)$ between A and B is nearly a straight line, this point C represents the false position of the

[9] Kovach, op. cit., p. 38.

root, as shown in the following graph:



Since AB is a straight line, then AC and CB must have the same slope. Hence,

$$\frac{f(a) - 0}{a - c} = \frac{f(a) - f(b)}{a - b}.$$

Solving this equation for c results in the formula

$$c = a + \frac{f(a)(b - a)}{f(a) - f(b)} .$$

Again, c replaces a or b, whichever function value f(a) or f(b) has the same sign as f(c). This process may be continued to obtain closer and closer approximations to the actual root.

The method of false position is based on the observation that in trying to decrease the interval in which there is a change in the sign of the function, if one end value is large and the other is small, the zero is probably closer to the small value than it is to the large. Like the bisection method, the method of false position converges for any polynomial function independent of the starting values. However, it gives the best results when used to improve the accuracy of a root once it is known approximately, since successive approximations get closer and closer together and the graph becomes more nearly a straight line. The required computations are slightly more complicated than for the bisection method, but convergence is usually achieved faster. For example, the bisection method required 18 iterations to determine 2.09455 as an approximation to a root of $f(x) = x^3 - 2x - 5$. Use of the false position method results in this approximation after only 15 iterations.

Convergence by the method of false position is still slower than by most methods. One cause of this slow convergence is that as soon as an interval is reached on which the function is convex or concave, thereafter one of the endpoints of this interval is always retained. Rabinowitz suggests several modifications of the method of false position to remedy this situation and thus speed up convergence.[10] Whenever an interval is reached where the functional values of the last two approximations, c_i and c_{i+1}, have the same sign, then some method must be used to determine a new bracket on the root which does not include the oldest approximation, c_{i-1}. One possible solution is to apply the bisection method to c_{i-1} and c_{i+1} to determine a new endpoint. Figure 5 depicts a flowchart which incorporates this modification into the method of false position.

The following output compares the results of applying the method of false position, both with and without the modification described above, to the function

$$f(x) = x^2 - x - 1.$$

Initial starting values of 1 and 2 are used, and the process is terminated when the value of the function is less than or equal to a value of $e = .00002$.

[10]Philip Rabinowitz, Numerical Methods for Nonlinear Algebraic Equations (New York: Gordon and Breach, Science Publishers, Ltd., 1970), p. 26.

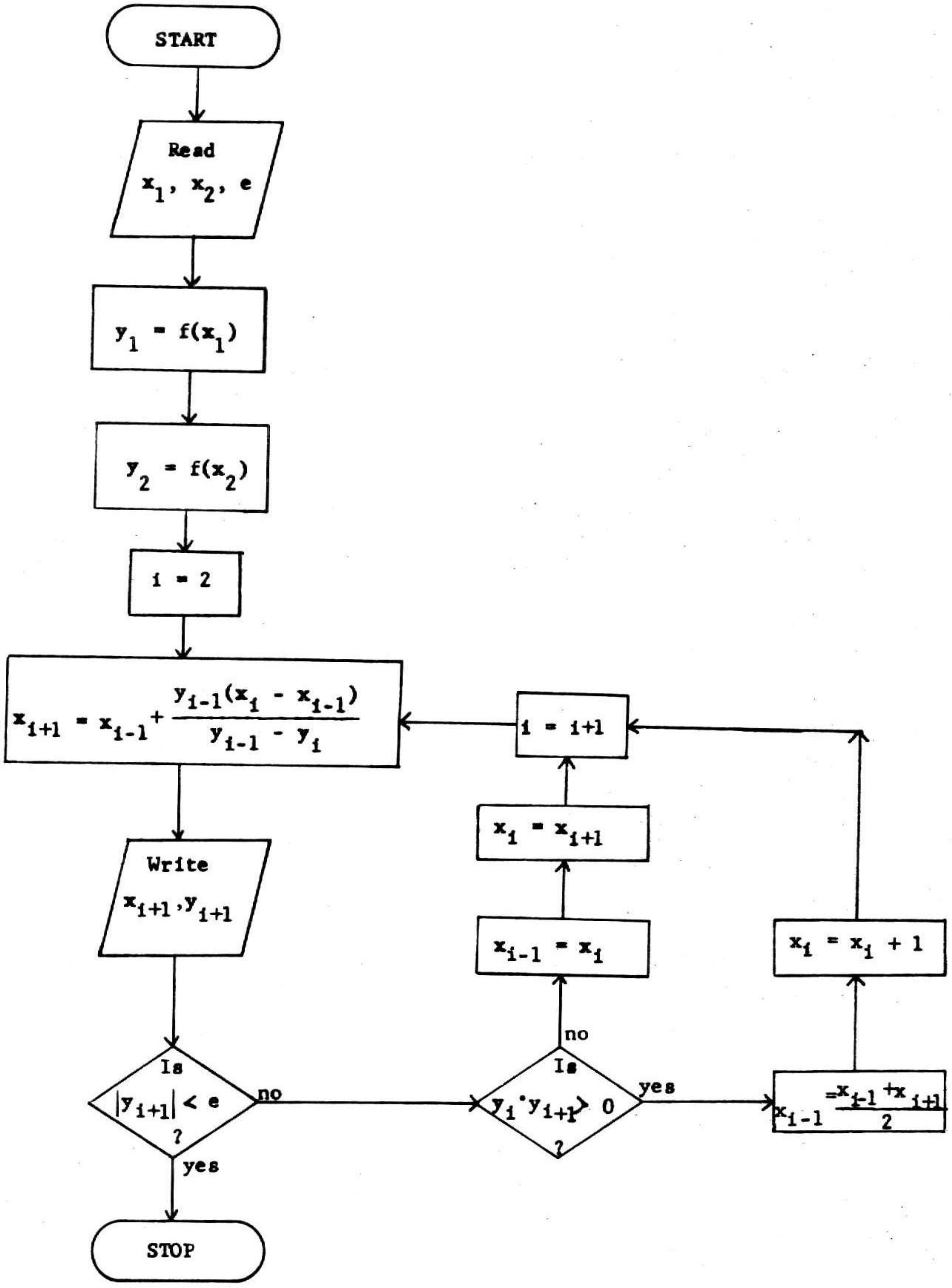


Figure 5. A modification of the method of false position.

<u>Without Modification</u>	<u>x</u>	<u>y</u>
1	1.500000	-0.250000
2	1.600000	-0.040000
3	1.615384	-0.005917
4	1.617647	-0.000865
5	1.617977	-0.000126
6	1.618025	-0.000018
7	1.618032	-0.000002

<u>With Modification</u>	<u>x</u>	<u>y</u>
1	1.500000	-0.250000
2	1.600000	-0.040000
3	1.616666	-0.003055
4	1.618071	0.000083
5	1.618034	-0.000000

Another very simple and efficient method of solving a polynomial equation is by the method of linear iteration. This method is based on the fact that any equation $f(x) = 0$ may be written in the form $g(x) = h(x)$, where a root of the original equation is the intersection of the two equations $y = g(x)$ and $y = h(x)$. Obviously, there are a number of ways to choose $g(x)$ and $h(x)$. Only certain choices will lead to an iterative process that will converge. It can be shown that whenever $|g'(x)| < |h'(x)|$ in the neighborhood of the intersection point, then convergence is assured.[11]

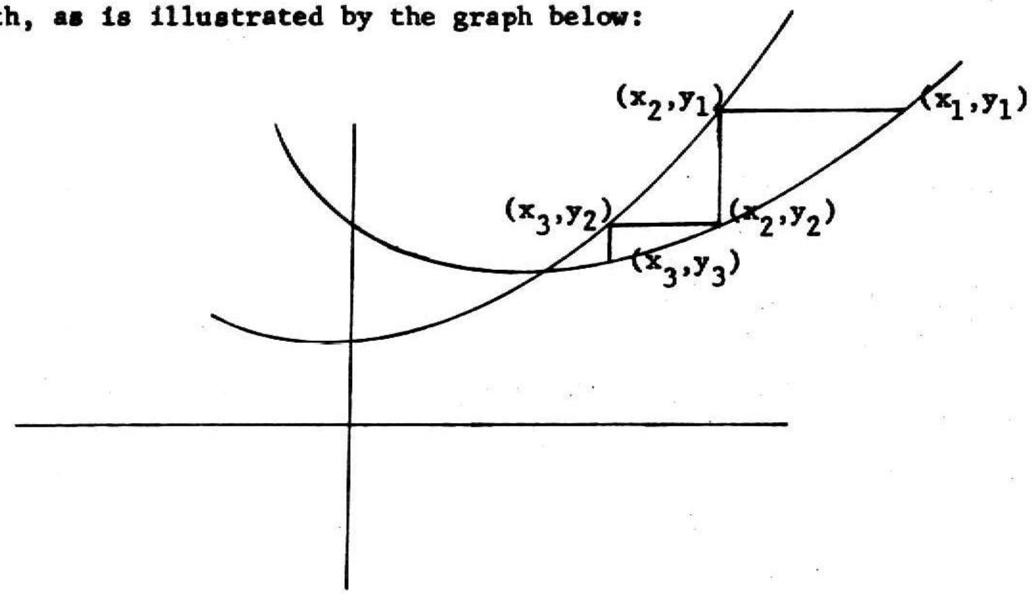
After determining the derivatives of $g(x)$ and $h(x)$, take the inverse of whichever function has the greatest slope in the neighborhood of the point of intersection. For example, if $h(x)$ has the greatest slope near the root, then the inverse function $x = H(y)$ should be used. After obtaining x_1 as a rough approximation to the root, using one of the methods of estimation previously discussed, calculate y_1 by evaluating the function $g(x)$ at the point x_1. Then, find x_2 by

[11] Kaiser S. Kunz, <u>Numerical Analysis</u> (New York: McGraw-Hill Book Company, 1957), p. 7.

evaluating H(y) at y_1. Repeat as follows to get successive approximations until a sufficiently close one is obtained:

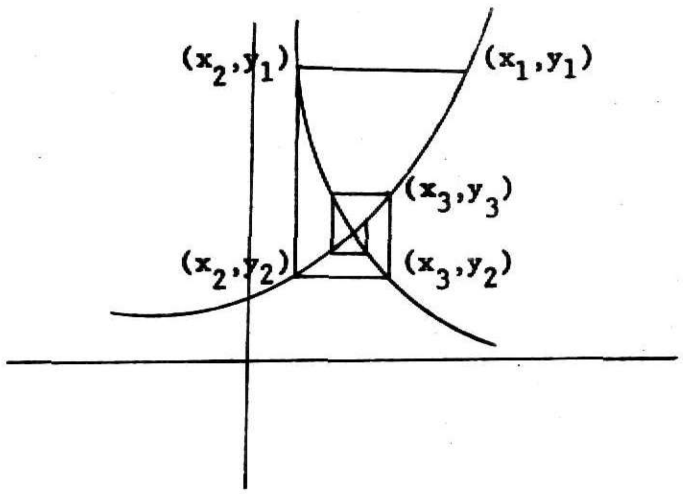
$$\begin{aligned} y_1 &= g(x_1) \\ x_2 &= H(y_1) \\ y_2 &= g(x_2) \\ x_3 &= H(y_2) \\ &\vdots \\ x_n &= H(y_{n-1}) \\ y_n &= g(x_n) \end{aligned}$$

If g(x) and h(x) have slopes of the same sign, the iterates $x_1, \ldots, x_n$ will approach the actual value of the root along a step-like path, as is illustrated by the graph below:



In this case, the approximations to the root are all either larger or smaller than the actual root.

If g(x) and h(x) have slopes of opposite signs, the iterates will approach the actual value of the root along a spiral path, thus furnishing an upper an lower bound for the root, as is illustrated in the following graph:



One major disadvantage of this general iteration procedure is the difficulty of being able to write $f(x) = 0$ in the form $y = g(x)$, $y = h(x)$ and at the same time making certain that the first equation is readily solvable for x. A slight modification in the general procedure will help avoid this difficulty. Since any polynomial equation of the form $f(x) = 0$ can easily be expressed in the form $x = g(x)$, and since the equation $y = x$ has a constant slope of 1, then $y = g(x)$ and $x = y$ can be used as the two required equations. The following iteration formulas result:

$$\begin{aligned} x_2 &= g(x_1) \\ x_3 &= g(x_2) \\ &\vdots \\ x_n &= g(x_{n-1}) \end{aligned}$$

Then, the only necessary requirement for convergence is that $\left|g'(x)\right| < 1$ near the point of intersection of the two curves. If the derivative of the chosen $g(x)$ is greater than 1, merely transform the equation $f(x) = 0$ into a different form.

Geometrically, $g'(x)$ represents the slope of the tangent to the curve $y = g(x)$ at any given point x. However, the average high school student may not be familiar with the concept of the derivative. Therefore, since approximations are being dealt with, it would be possible to substitute the slope of the secant between the two points $(x_1, g(x_1))$ and $(x_2, g(x_2))$ in place of the derivative. Then, if

$$\left| \frac{g(x_2) - g(x_1)}{x_2 - x_1} \right| < 1,$$

the iteration will usually succeed. However, the derivative is still a better test.

A flowchart of the method of linear iteration is illustrated in Figure 6. This procedure was applied to the polynomial

$$f(x) = x^3 + 2x^2 + 10x - 20$$

using the function

$$g(x) = \frac{20}{x^2 + 2x + 10}$$

with an initial approximation of $x = 1$. The following output resulted:

	<u>x</u>	<u>f(x)</u>
1	1.538461	3.759670
2	1.295019	-1.523816
3	1.401825	0.703224
4	1.354209	-0.306678
5	1.375298	0.137168
6	1.365930	-0.060670
7	1.370086	0.026967
8	1.368241	-0.011961
9	1.369060	0.005309
10	1.368696	-0.002361
11	1.368857	0.001043
12	1.368786	-0.000464
13	1.368818	0.000204
14	1.368803	-0.000092
15	1.368810	0.000037

The linear iteration process could have been applied to the same polynomial $f(x)$, but using the function

$$g(x) = \frac{20 - 2x^2 - 10x}{x^2}.$$

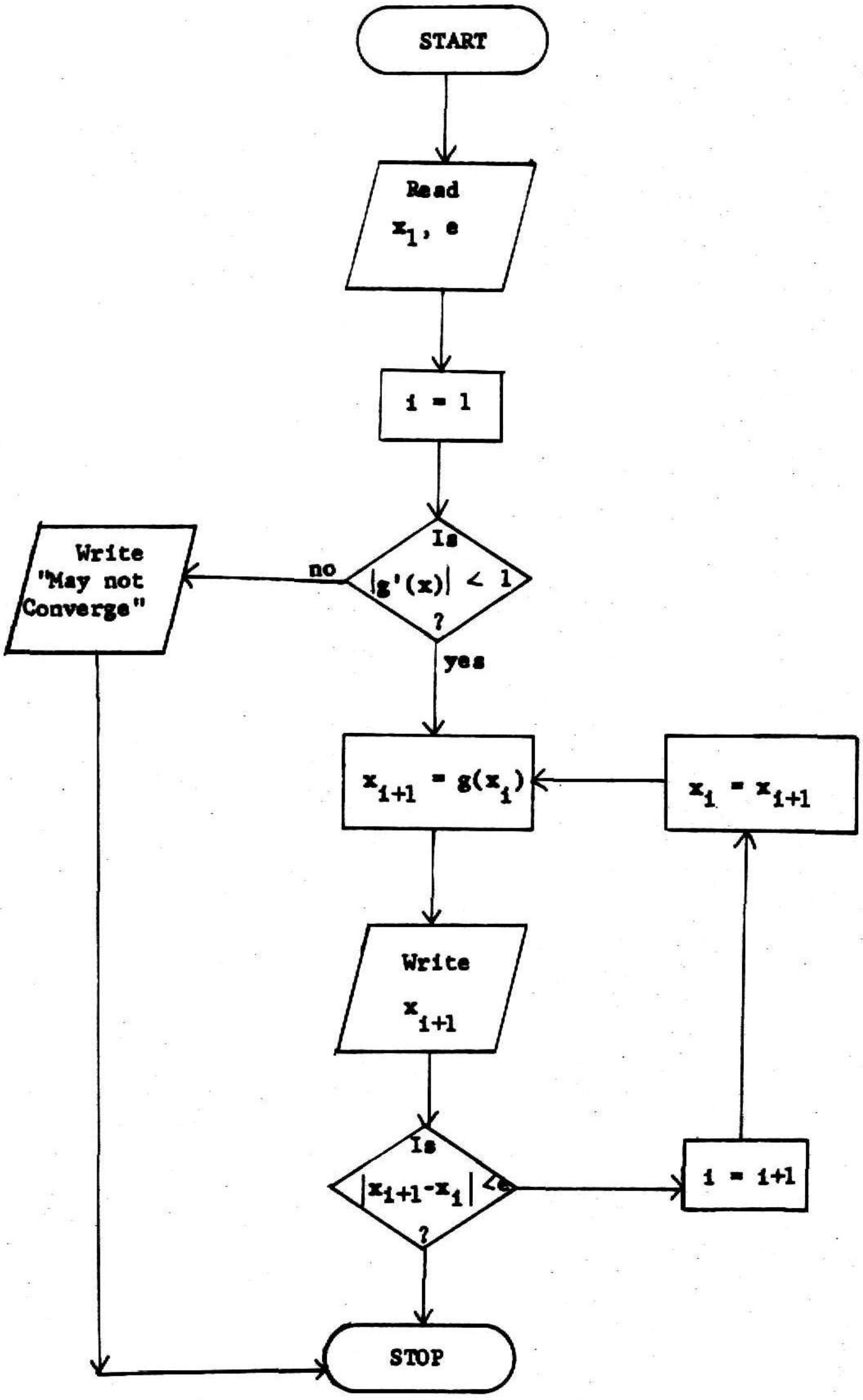


Figure 6. The method of linear iteration.

However, since $\left|g'(1)\right| = 30$, the method would fail to converge in this case.

The method of linear iteration is very easily adapted to use on the computer, since the program merely involves the repeated calculation of the same function. Its biggest disadvantage is that convergence occurs only if certain requirements of the first derivative are met. Since each repetition uses different numbers and produces a more accurate result, convergence is fairly rapid when it does occur. However, it still may be too slow, especially if the slope of $g(x)$ is near 1 in the neighborhood of the root.

An acceleration device does exist for speeding up the convergence of the linear iteration process.[12] This device, called Aitken's Δ^2 process, uses three consecutive values of an iterate, x_k, x_{k+1}, and x_{k+2}. These values will differ from some better approximation x by $x - x_k$, $x - x_{k+1}$, and $x - x_{k+2}$. Since the error is reduced by a constant factor at each iteration, the following ratios are found to be equal:

$$\frac{x - x_k}{x - x_{k+1}} = \frac{x - x_{k+1}}{x - x_{k+2}}$$

Solving for x gives

$$x = x_k - \frac{(x_k - x_{k+1})^2}{x_k - 2x_{k+1} + x_{k+2}} .$$

If $\{x_k\}$ is a sequence of iterates generated from some convergent linear iteration, then a new sequence $\{x'_k\}$ can be generated using the above formula. This new sequence converges to the desired root

[12] Conte, op. cit., p. 28.

faster providing that the derivative is not equal to zero on the interval.

The flowchart in Figure 7 illustrates the use of Aitken's Δ^2 process. Applying the method of linear iteration with acceleration to the polynomial

$$f(x) = x^3 + 2x^2 + 10x - 20 = 0$$

produces the following results:

	x	f(x)
1	1.538461	3.759670
2	1.295019	-1.523816
3	1.370813	0.042332
4	1.367918	-0.018767
5	1.369203	0.008333
6	1.368808	-0.000002
7	1.368808	0.000002

Note that the use of Aitken's acceleration device reduced the required number of iterations from 15 to 7 to approximate the root to the fourth significant decimal place.

Probably the best known of all iterative methods is one developed by Isaac Newton.[13] Newton suggested that since the derivative of a function tells how fast the function is changing with respect to x, then it can be used to adjust x. Rather than assuming a straight line between two points on the curve, Newton's method makes use of one point and estimates where the approximating tangent to the curve at that point crosses the x-axis.

Let x_1 be the first approximation to the solution of the equation $f(x) = 0$. The tangent to the graph at the point $(x_1, f(x_1))$ crosses the x-axis at some point $(x_2, 0)$. This point x_2 becomes

[13]Ralph H. Pennington, Introductory Computer Methods and Numerical Analysis (London: The Macmillan Company, 1970), pp. 286-91.

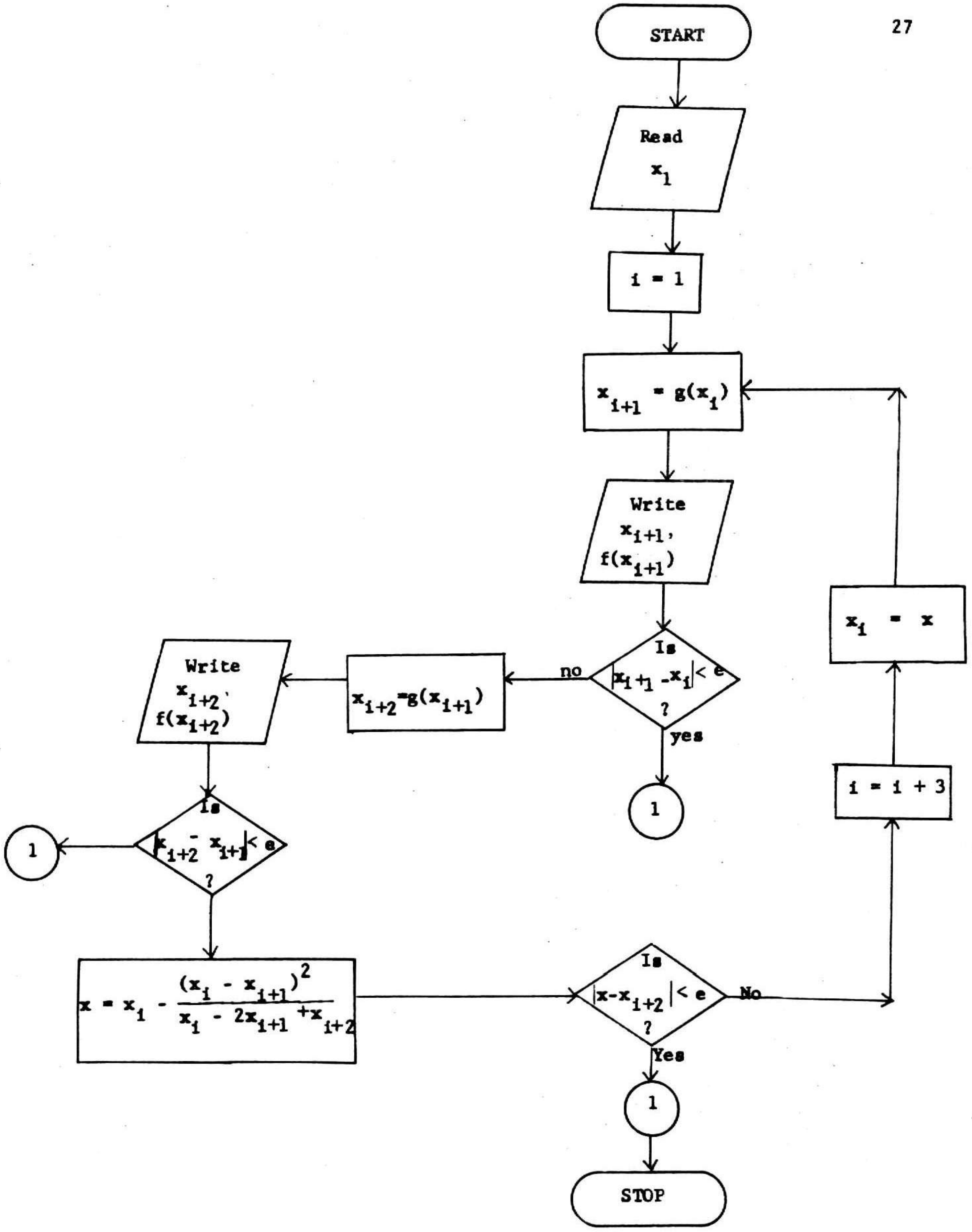
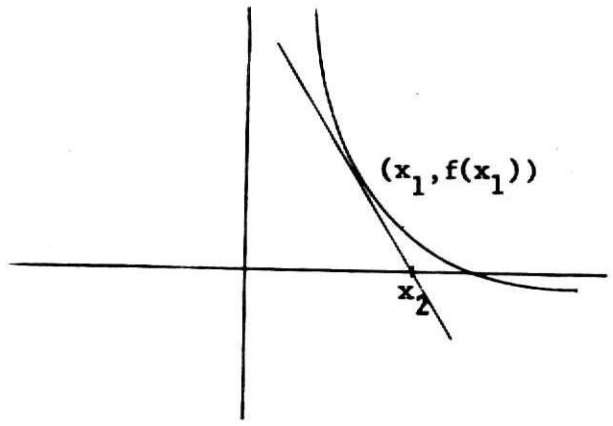

START
Read x_1
$i = 1$
$x_{i+1} = g(x_i)$
Write x_{i+1}, $f(x_{i+1})$
Is $|x_{i+1} - x_i| < e$?
no
yes
1
$x_{i+2} = g(x_{i+1})$
Write x_{i+2}, $f(x_{i+2})$
Is $|x_{i+2} - x_{i+1}| < e$?
1
$x = x_i - \frac{(x_i - x_{i+1})^2}{x_i - 2x_{i+1} + x_{i+2}}$
Is $|x - x_{i+2}| < e$?
No
Yes
1
$i = i + 3$
$x_i = x$
STOP

Figure 7. The method of linear iteration using Aitken's Δ^2 process.

the second approximation to the root.



Since $f'(x_1)$ represents the tangent to the graph at the point x_1, then by the definition of slope

$$f'(x_1) = \frac{f(x_1) - 0}{x_1 - x_2},$$

so

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

Using the iteration formula

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

successive approximations can be generated to the desired degree of accuracy.

If the derivative does not change too rapidly, and if the curve does not become nearly horizontal near the x-intercept, convergence by Newton's method will be good. However, if one of the approximations is near a maxima or minima point, divergence could occur. In such a case, the derivative could become so small that division by it might result in a number too large for the computer to represent. A check for the value of the derivative should be written into any program which uses

Newton's method. A flowchart for Newton's method is depicted in Figure 8.

Macon suggests that if, while applying Newton's method, it is discovered that $f'(x)$ becomes very small, it is practical to depart from the standard sequence and attempt to obtain two new starting values.[14] To obtain these values, apply Newton's method to the equation $f'(x) = 0$. That is, perform the iteration

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

using the last available iterate as a starting value. Then compute

$$x = x_{i+1} \pm \sqrt{\frac{-2f(x_{i+1})}{f''(x_{i+1})}} .$$

These two numbers will separate the two close roots which caused $f'(x)$ to vanish, and either of them may be used as a starting value for Newton's iteration with some hope of converging.

Newton's method obviously requires more information and generally more work per step than any other method. Convergence is not assured, and a well-chosen initial approximation is a necessity. However, Newton's method can converge very quickly. Only 8 iterations are required to obtain the approximation $x = 2.09455$ for a root of $f(x) = x^3 - 2x - 5$, as compared with 18 iterations for the bisection method and 15 iterations for the method of false position. The convergence of Newton's method is quadratic, or of the second order, as opposed to the linear, or first order, convergence of the methods previously discussed. This means that the absolute error at one step

---

[14]Nathaniel Macon, Numerical Analysis (New York: John Wiley & Sons, Inc., 1963), p. 34.

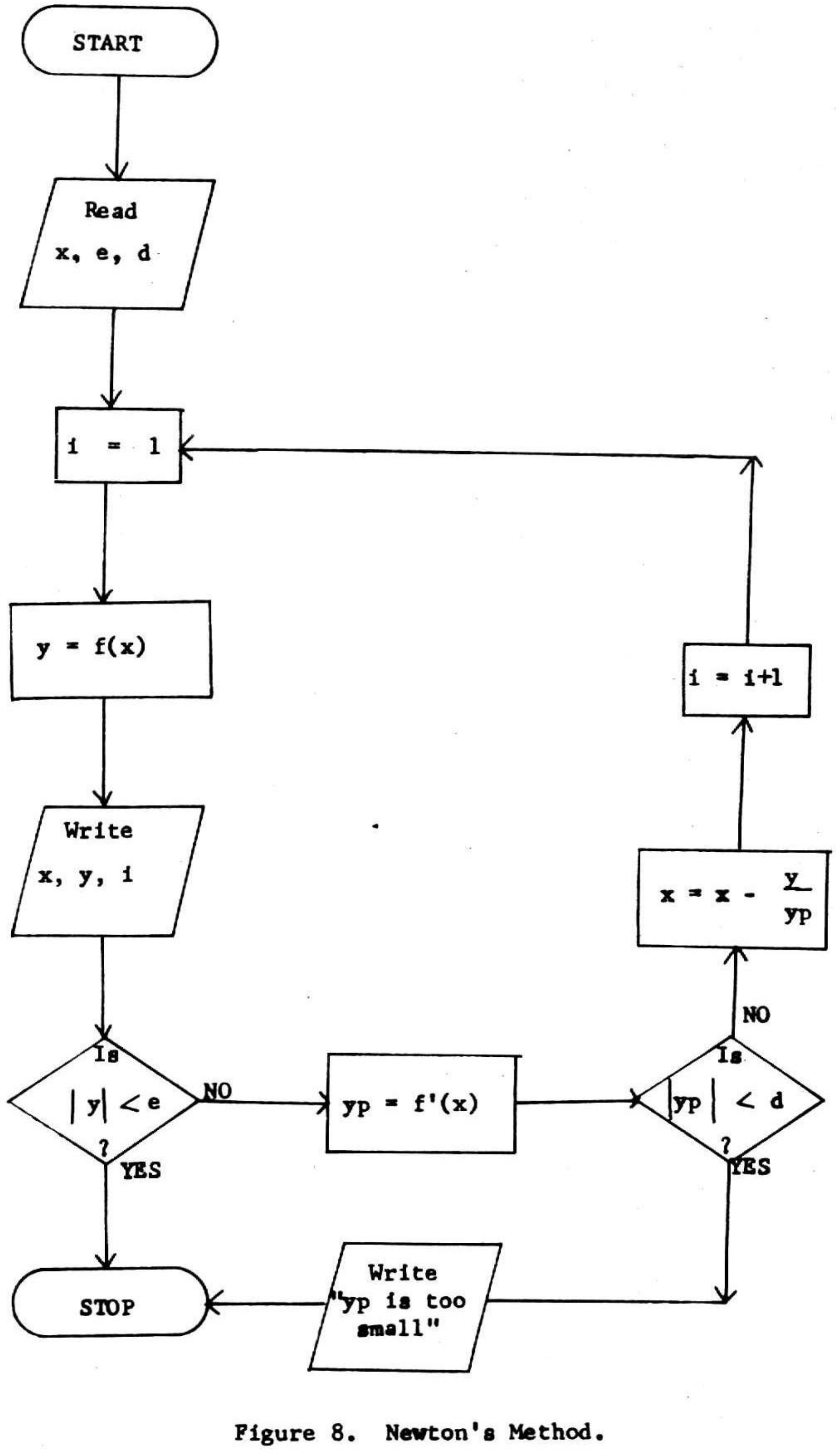


Figure 8. Newton's Method.

is proportional to the square of the absolute error at the previous step; that is, an answer correct to 1 decimal place at one step should be correct to 2 decimal places at the next step, 4 at the next, and so on.

One argument against using Newton's method for approximating the roots of a polynomial at the high school level is that it makes use of the derivative. The average high school student has not studied derivatives, so the formula for finding the tangent to the curve at any given point is meaningless to him. The secant method provides an alternative to Newton's method by replacing the slope of the tangent at the point $(x_i, f(x_i))$ by the slope of the secant line between the two points $(x_i, f(x_i))$ and $(x_{i-1}, f(x_{i-1}))$.[15] Thus, the expression

$$\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

is substituted for $f'(x_i)$, and the subsequent iteration formula becomes

$$x_{i+1} = x_i - \frac{(x_i - x_{i-1})\, f(x_i)}{f(x_i) - f(x_{i-1})} .$$

Algebraic manipulation will show that this is the same iteration formula that is used in the method of false position. However, the method of false position requires that $f(x)$ have opposite signs at the two points used to generate the next point. The secant method keeps the two most recently computed points at each iteration, regardless of whether or not they bracket the root. A flowchart for the secant method is shown in Figure 9.

[15] Conte, _op. cit._, p. 40.

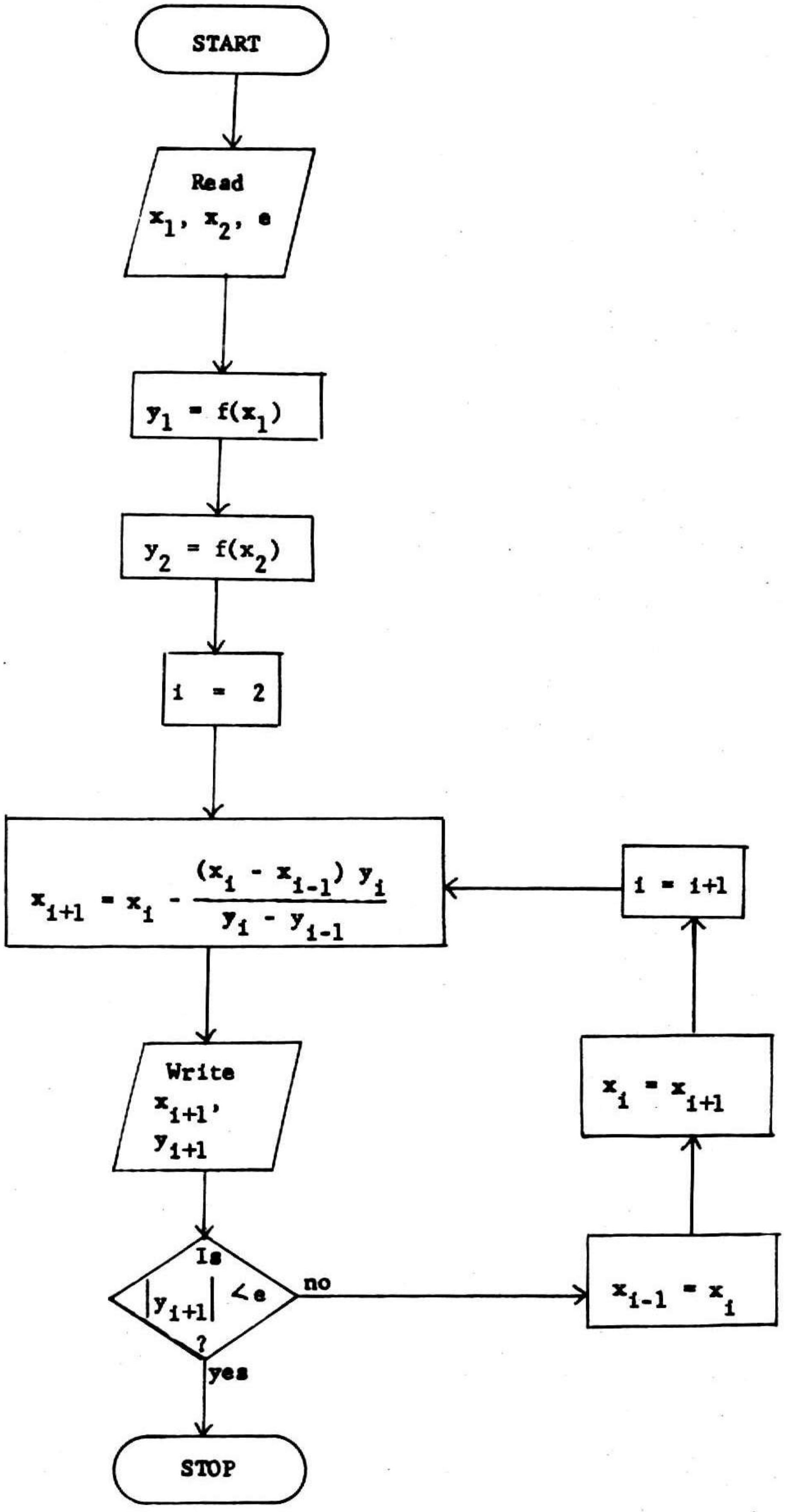


Figure 9. The secant method.

The convergence of the secant method is noticeably faster than for the bisection method or the method of false position because it uses more up-to-date information. Also, the secant method is much more efficient. Its advantage over Newton's method lies in the fact that it does not require the evaluation of a derivative. Also, it is more stable than Newton's method, but it is usually somewhat slower to converge. Convergence of the secant method is not always guaranteed. When it does converge, its rate is not quadratic as in Newton's method, but it is considerably better than linear.

The following output resulted from applying the secant method to find a root of the polynomial equation

$$x^3 - 9x^2 + 5x - 6 = 0 ,$$

using starting values of 8 and 10:

	x	f(x)
1	8.344827	-9.899662
2	8.451295	-2.934514
3	8.496150	0.110492
4	8.494522	-0.001242
5	8.494539	-0.000065

The bisection method required 18 iterations to arrive at the same approximation, the method of false position required 11 iterations, and Newton's method required only 4 iterations.

Whittaker's method is another procedure that avoids the computation of the derivative in Newton's method.[16] This process consists of replacing the derivative by some constant value M. The resulting formula

$$x_{n+1} = x_n - \frac{f(x_n)}{M}$$

[16]Peter Henrici, Elements of Numerical Analysis (New York: John Wiley & Sons, Inc., 1964), p. 87.

then defines, for a certain range of values of M, a linearly converging sequence. If the estimate of M is good, convergence may be quite rapid. This method would save time when doing calculations by hand, but the secant method would be more accurate and just slightly more complicated for use on the computer.

Horner's method is a method which was popular when numerical calculations were done entirely by hand.[17] Consider the polynomial equation

$$P(x) = x^3 + 18x - 30 = 0.$$

To find the roots of this equation between $x = 1$ and $x = 2$, set x equal to $(1 + p)$. An attempt will then be made to approximate p. Substituting the expression $(1 + p)$ for x in the original equation results in the equation

$$p^3 + 3p^2 + 21p - 11 = 0.$$

Since $p = x - 1$, this equation can be rewritten as

$$(x - 1)^3 + 3(x - 1)^2 + 21(x - 1) - 11 = 0.$$

Hence, it can be seen that -11 is the remainder when $P(x)$ is divided by $(x - 1)$, resulting in the quotient polynomial

$$Q(x) = (x - 1)^2 + 3(x - 1) + 21.$$

Repeated divisions by $(x - 1)$ give the remainders 21, 3, and 1. This suggests an easier way than direct substitution for finding the coefficients 1, 3, 21, and -11 of the terms in the transformed equation--that is, through a process of repeated synthetic division.

Divide $P(x)$ by $(x - 1)$ and continue dividing the resulting quotient polynomials by $(x - 1)$ as follows:

[17] Leonard Eugene Dickson, New First Course in the Theory of Equations (New York: John Wiley & Sons, Inc., 1939), pp. 90-93.

1	1	0	18	-30
		1	1	19
1	1	1	19	-11
		1	2	
1	1	2	21	
		1		
1	1	3		
	1			

Taking the remainders in reverse order gives 1, 3, 21, and -11, the coefficients of the desired equation. Let p be a root of

$$p^3 + 3p^2 + 21p - 11 = 0.$$

Now if x is a root of the original equation, then $x = p + 1$. To obtain an approximation to the decimal p, ignore the p^3 and p^2 terms of the equation and solve the resulting linear equation

$$21p - 11 = 0,$$

obtaining .5 as an approximate value for p. Since the ignored terms were positive, this approximation will be too large. Therefore, let $p = .4 + h$, and repeat the synthetic division process, dividing the polynomial

$$p^3 + 3p^2 + 21p - 11$$

by $(p - .4)$ to approximate h.

.4	1	3	21	-11
		.4	1.36	8.944
.4	1	3.4	22.36	-2.056
		.4	1.52	
.4	1	3.8	23.88	
		.4		
.4	1	4.2		
	1			

Use the resulting equation

$$h^3 + 4.2h^2 + 23.88h - 2.056 = 0$$

to approximate h, obtaining h = .08 + t. Repeat the synthetic division process, dividing by (h - .08).

.08	1	4.2	23.88	-2.056
		.08	.3424	1.937792
.08	1	4.28	24.2224	- .118208
		.08	.3488	
.08	1	4.36	24.5712	
		.08		
.08	1	4.44		
	1			

Use the resulting equation to approximate t = .004 + s, and divide again.

.004	1	4.44	24.5712	-.118208
		.004	.017776	.098355904
.004	1	4.444	24.588976	-.019852096
		.004	.017792	
.004	1	4.448	24.606768	
		.004		
.004	1	4.452		
	1			

This equation can then be used to obtain a final approximation of .0008 for s. It follows that x = 1 + .08 + .004 + .0008 = 1.4848 is a good approximation to the root. This process could have been continued to any desired degree of accuracy. It is a very practical method for finding the root when doing the calculations by hand, since accurate division is not essential. However, because of the work involved and the slow rate of convergence, Horner's method is not recommended for use on the computer.

Another interesting root-finding method was developed by Daniel Bernoulli in the eighteenth century. Bernoulli's method is very useful when the root of largest magnitude is the only one desired.[18] The theory behind this method is based on the solution of linear difference equations and will not be proved in this paper. The basic procedure consists of arbitrarily choosing a sequence of numbers $p_1, \dots, p_n$ and replacing each x^k in a given polynomial by the corresponding p_{k+1}. For example, consider the polynomial equation

$$P(x) = x^3 - 3x^2 - x + 3.$$

Arbitrarily choose $p_1 = 0$, $p_2 = 1$, and $p_3 = 1$. Replace each x^k by the corresponding p_{k+1}, forming the new equation

$$p_4 - 3p_3 - p_2 + 3p_1 = 0.$$

Substituting the chosen values for p_1, p_2, and p_3, solve the resulting linear equation obtaining $p_4 = 4$. Ignore p_1 and repeat the process using p_5, p_4, p_3, and p_2, solving for p_5. This procedure may be repeated as many times as desired, resulting in a sequence of numbers $p_4, p_5, \dots, p_i$. Another sequence of numbers can then be formed by taking the ratio of each number in the original sequence to the one before it. If these ratios approach a limit, that limit is the largest root. If the ratios oscillate periodically, then either a pair of complex roots or two roots equal in absolute value are present, so a different method must be applied.

The flowchart in Figure 10 illustrates the use of Bernoulli's method. Applying this process to the original polynomial equation

$$P(x) = x^3 - 3x^2 - x + 3 = 0,$$

using a value of 15 for M, results in the following output:

---

[18]George R. Stibitz and Jules A. Larrivee, Mathematics and Computers (New York: McGraw-Hill Book Company, Inc., 1957), pp. 77-78.

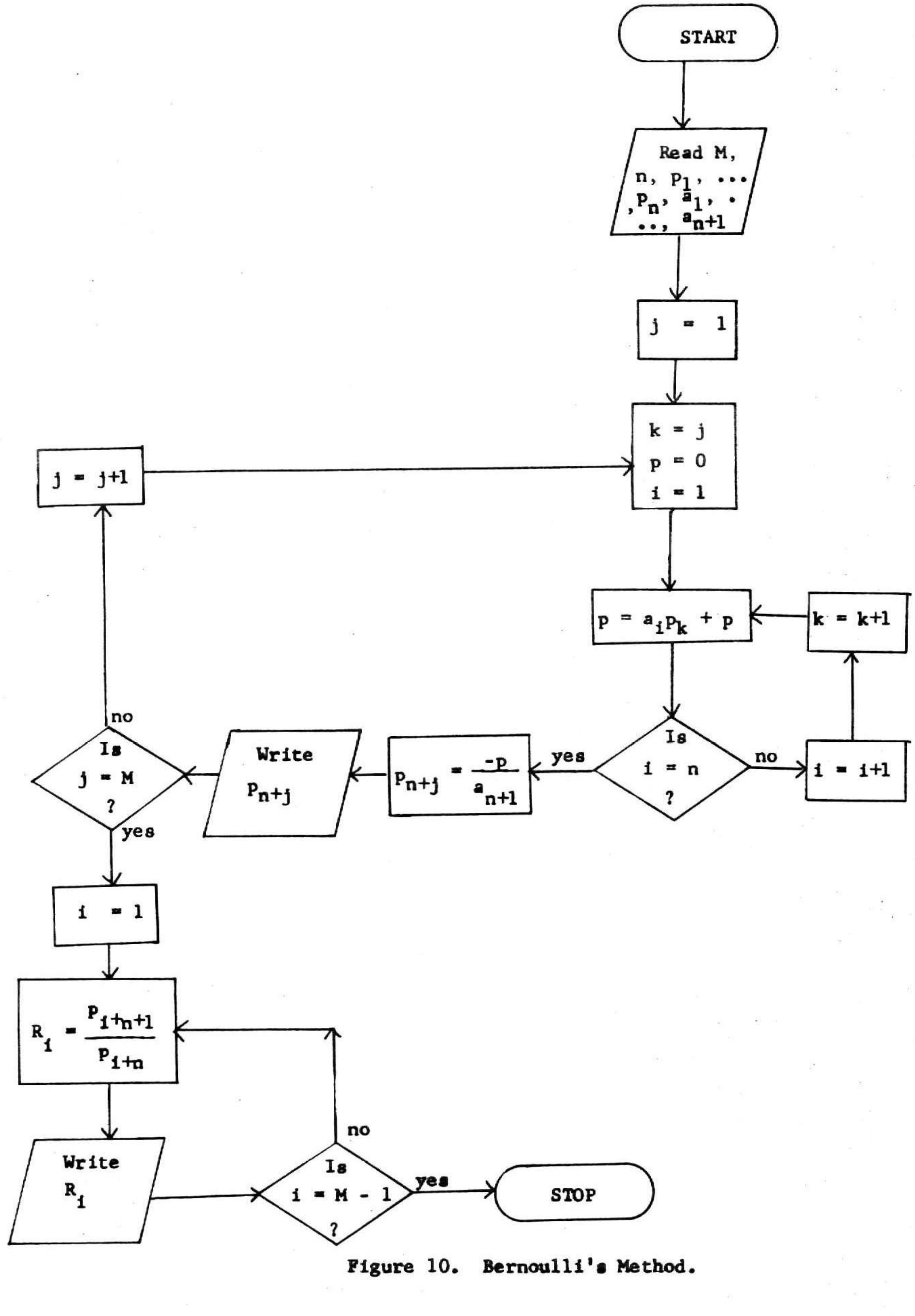


Figure 10. Bernoulli's Method.

	P_k	P_{k+1}/P_k
1	4.0	2.500000
2	10.0	3.100000
3	31.0	2.935483
4	91.0	3.010989
5	274.0	2.992701
6	820.0	3.001219
7	2461.0	2.999187
8	7381.0	3.000135
9	22144.0	2.999909
10	66430.0	3.000015
11	199291.0	2.999989
12	597871.0	3.000001
13	1793614.0	2.999999
14	5380841.0	3.000000
15	16142523.0	

Therefore, 3 is determined to be the magnitude of the largest root of $P(x)$. The actual roots of the polynomial are -1, 1, and 3.

Bernoulli's method works only when the roots are all real and distinct. Multiple roots and complex roots cannot be determined. Convergence is slow, and only the root with the largest absolute value can be found. For these reasons, the method is far from ideal as a general purpose program for root-finding on the computer.

All of the methods discussed up to this point enable one to obtain a more accurate value for a real root whose approximate location is already known by other means. Only one real root at a time can be approximated, and it is difficult to know when all the real roots have been found. Rarely is it necessary in practical applications to find all real and complex solutions to a polynomial equation. However, the student should realize that numerical methods to do this do exist, and he should be introduced to some of them.

One solution to the problem of finding all the roots of a given polynomial is to search out the real roots one at a time. Once a real root has been determined, a new equation with degree reduced by 1 can be obtained by dividing the original equation by the known factor. Any roots of this new equation will be roots of the original equation as well. The procedure for performing such a division synthetically on the computer has already been discussed. However, if successive roots are found by continuously reducing the degree of the polynomial, a loss of accuracy in the later roots must be expected. This inaccuracy is caused by errors in the coefficients of the reduced polynomials due to round-off and incomplete convergence. The accuracy of the root can be improved by reiterating with the original polynomial. However, some polynomials, especially those of high degree, are very unstable in the sense that small changes in the coefficients will lead to large changes in the roots. Conte uses the polynomial equation

$$x^7 - 28x^6 + 322x^5 - 1960x^4 + 6769x^3 - 13{,}132x^2 + 13{,}068x - 5040 = 0$$

to illustrate this danger.[19] The exact roots of this polynomial are 1, 2, 3, 4, 5, 6, and 7. Applying Newton's method to approximate each of these roots results in the approximations in column 1 below. The numbers in column 2 were obtained using the same method and same initial conditions, but the coefficient of the x^2 term was replaced by -13,133.

	1	2
1	.99999999	1.00139750
2	1.99999780	1.96891970
3	3.00001310	3.31839620
4	3.99998170	3.50497150
5	4.99997960	7.05993460
6	6.00006230	5.30904000
7	6.99996480	5.83734000

Note that a change of 1/100 of 1 percent in one of the coefficients

[19] Conte, op. cit., p. 57.

has led to a 10 percent change in some of the roots. Therefore, caution must be exercised when obtaining roots from a reduced polynomial, especially one of high degree.

An alternative method of finding all the real roots of a polynomial makes use of the reciprocal polynomial.[20] Consider the polynomial equation

$$x^3 - 4x^2 + x + 6 = 0,$$

which has the exact roots -1, 2, and 3. Replacing x by 1/y results in the polynomial equation

$$1 - 4y + y^2 + 6y^3 = 0.$$

The exact roots of this new polynomial are -1, 1/2, and 1/3--the reciprocals of the roots of the original polynomial. Therefore, all the roots of the original polynomial that lie outside the range ± 1 now lie inside it for the reciprocal polynomial. Thus, one can search for all the real roots in the interval from -1 to +1 for the original polynomial, and then form the reciprocal polynomial and search for its roots in the same region. Since the reciprocal polynomial will always have the same coefficients as the original polynomial, but in reverse order, this method is easily incorporated into a computer program.

Numerical methods do exist which compute all the real roots of a polynomial simultaneously. One of the oldest and simplest procedures for doing this is due to Graeffe.[21] His process is based on replacing the given polynomial by one whose roots are the squares

---

[20]Acton, loc. cit.

[21]Pennington, op. cit., pp. 320-29.

of those of the original polynomial. Consider the polynomial equation

$$P(x) = 2x^3 - 7x^2 + 7x - 2 = 0.$$

Suppose the actual roots of this equation are a, b, and c. Then by the factor theorem, the equation can be rewritten as

$$P(x) = 2(x - a)(x - b)(x - c) = 0.$$

Replacing x by $-x$ in the original equation results in the new equation

$$P(-x) = -2x^3 - 7x^2 - 7x - 2 = 0.$$

The roots of this equation are $-a$, $-b$, and $-c$; therefore, the equation can also be written as

$$P(-x) = -2(x + a)(x + b)(x + c) = 0.$$

Multiplying $P(x)$, $P(-x)$, and (-1) results in the equation

$$P(x^2) = (-1)P(x)P(-x) = 4(x^2 - a^2)(x^2 - b^2)(x^2 - c^2).$$

The roots of this new polynomial are the squares of the roots of the original polynomial $P(x)$. Now

$$\begin{aligned} P(x^2) &= (-1)P(x)P(-x) \\ &= (-1)(2x^3 - 7x^2 + 7x - 2)(-2x^3 - 7x^2 - 7x - 2) \\ &= (-1)(-4x^6 + 21x^4 - 21x^2 + 4) \\ &= 4x^6 - 21x^4 + 21x^2 - 4. \end{aligned}$$

The actual roots of this equation are $1/4$, 1, and 4. If these can somehow be determined, then the roots of the original polynomial must be among $\pm 1/2$, ± 1, and ± 2.

Given the coefficients of the original polynomial $P(x)$, what is needed is a general formula for generating the coefficients of the polynomial $P(x^2)$. Let

$$P(x) = a_1 x^n + \ldots + a_n x + a_{n+1}$$

be a polynomial with actual roots $x_1, \ldots, x_n$. Then

$$P(x) = a_1(x - x_1)(x - x_2) \dots (x - x_n).$$

From the original polynomial, determine the new polynomial $P(-x)$ such that

$$P(-x) = a_1(-1)^n x^n + a_2(-1)^{n-1}x^{n-1} + \dots + -a_n x + a_{n+1}$$
$$= (-1)^n a_1(x + x_1)(x + x_2) \dots (x + x_n).$$

Let

$$P(x^2) = b_1 x^{2n} + b_2 x^{2n-2} + \dots + b_n x^2 + b_{n+1}$$

be the polynomial whose roots are the squares of the roots of $P(x)$. Then

$$\begin{aligned} P(x^2) &= b_1(x^2 - x_1^2)(x^2 - x_2^2) \dots (x^2 - x_n^2) \\ &= b_1(x - x_1)(x + x_1)(x - x_2)(x + x_2) \dots (x - x_n)(x + x_n) \\ &= b_1 \left[(x - x_1) \dots (x - x_n)\right] \left[(x + x_1) \dots (x + x_n)\right] \\ &= b_1/a_1^2 \, (-1)^n \, P(x) \;\; P(-x) \\ &= a_1^2 x^{2n} + (2a_1a_3 - a_2^2)x^{2n-2} + (2a_1a_5 - 2a_2a_4 + a_3^2)x^{2n-4} + \dots. \end{aligned}$$

Comparing coefficients of like powers of x results in the following recursion formulas:

$$\begin{aligned} b_1 &= a_1^2 \\ b_2 &= -a_2^2 + 2a_1a_3 \\ b_3 &= a_3^2 - 2a_2a_4 + 2a_1a_5 \\ &\vdots \\ b_i &= (-1)^{i-1}(a_i - 2a_{i-1}a_{i+1} + 2a_{i-2}a_{i+2} + \dots \\ &\vdots \\ b_n &= (-1)^{n-1}(a_n^2 - 2a_{n-1}a_{n+1}) \\ b_{n+1} &= (-1)^n(a_{n+1})^2 \end{aligned}$$

If this root-squaring procedure is applied to a polynomial m times, the result is a polynomial with roots $x_1^{2^m}, x_2^{2^m}, \ldots, x_n^{2^m}$. If the roots of the original polynomial are real and unequal, then for large m, the following ratios can be shown to be approximately true, where $k = 2^m$:

$$\begin{aligned} b_2/b_1 &= -x_1^k \\ b_3/b_2 &= -x_2^k \\ &\vdots \\ b_{n+1}/b_n &= -x_n^k \,. \end{aligned}$$

The absolute values of the desired roots may then be found by taking the ratios of the coefficients as above, and then taking kth roots. When root-squaring has been applied enough times to make the approximations good ones, the coefficients themselves will be approximately squared at each step.[22]

Figures 11 and 12 illustrate how Graeffe's root-squaring method can be arranged for use on the computer. Figure 11 is a flowchart of the actual root-squaring process, while the flowchart in Figure 12 computes the ratios, decides whether to root-square again, and calculates and prints the absolute values of the roots. Applying Graeffe's root-squaring method to the polynomial equation

$$x^3 - 6x^2 + 11x - 6 = 0$$

results in the following output:

```
M     B(1)             B(2)             B(3)              B(4)

2     1.0             -14.0             49.0             -36.0
4     1.0             -98.0           1393.0           -1296.0
6     1.0           -6818.0        1686433.0        -1679616.0
8     1.0       -43112260.0     2.821152E12      -2.821110E12
10    1.0      -1.853024E15     7.958661E24      -7.958661E24
12    1.0      -3.433684E30     6.334029E49      -6.334029E49

ROOTS:  3.000000,  2.000000,  1.000000
```

[22]*Ibid.*

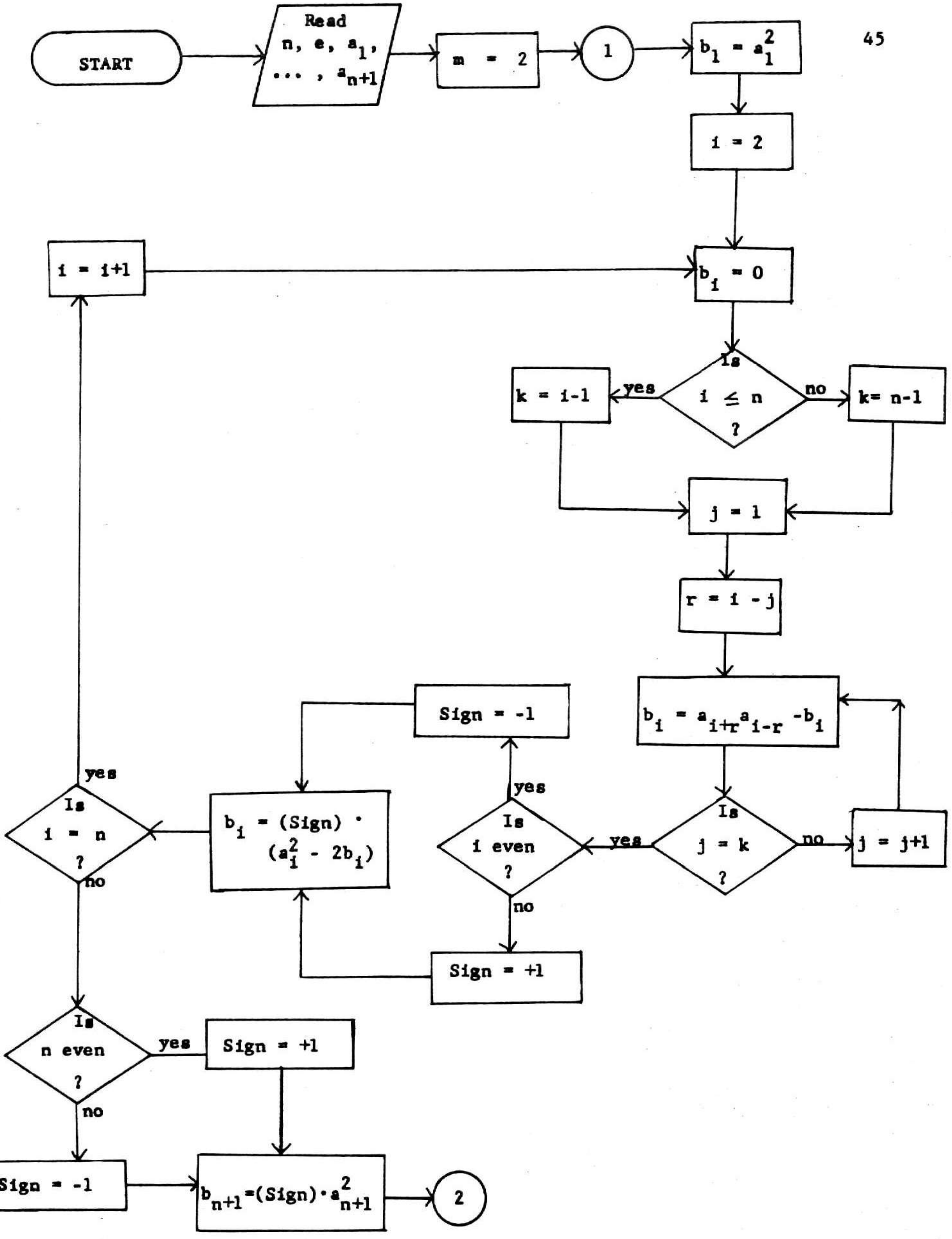


Figure 11. The root-squaring process.

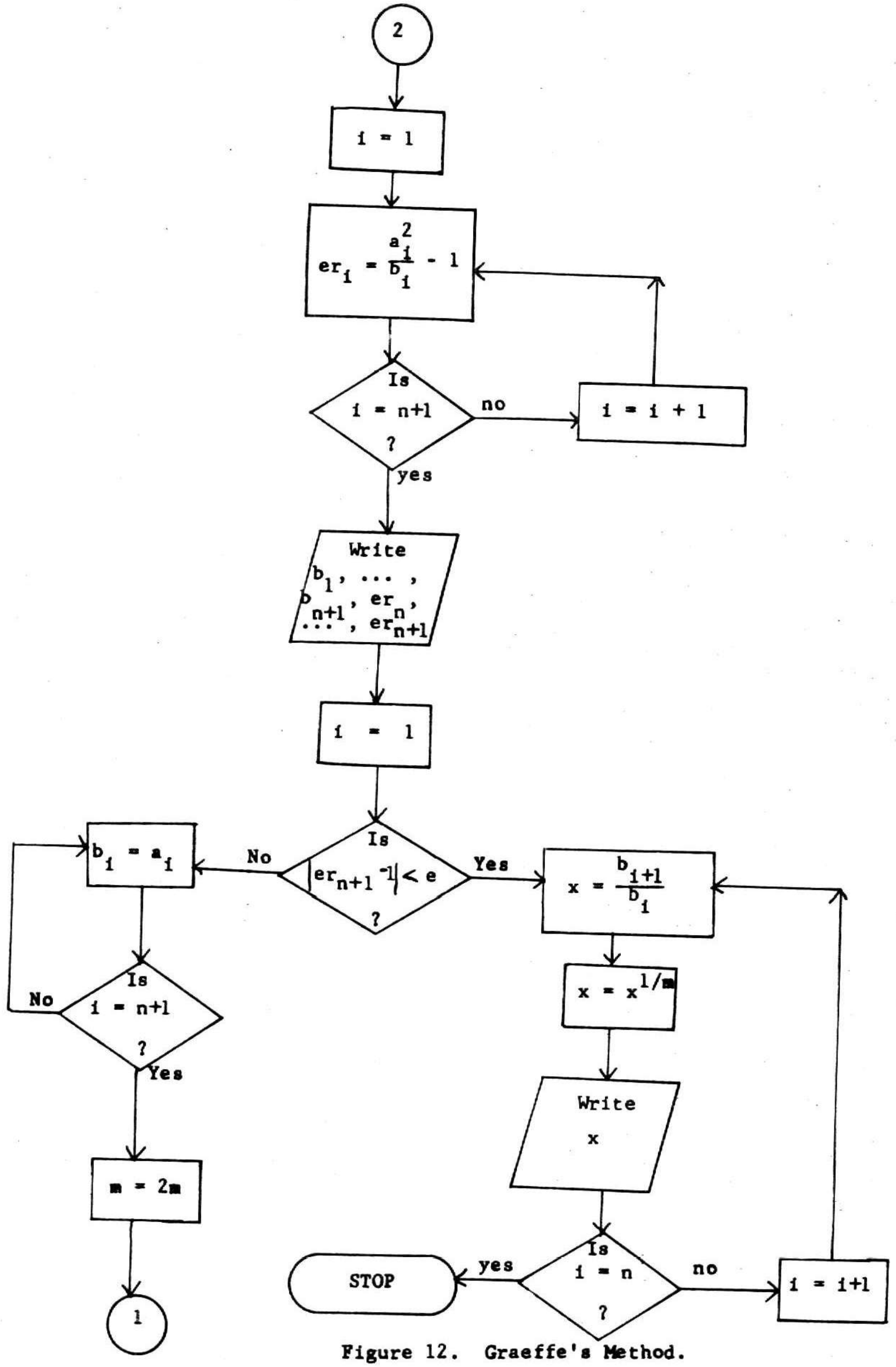


Figure 12. Graeffe's Method.

In cases where the roots of a given polynomial are all real and distinct, Graeffe's root-squaring method can be used to find all the roots directly without any prior information. However, the coefficients become squares of their previous values only when the roots are real and simple. When multiple or complex roots are present, the coefficients will not all display this behavior. Only certain coefficients, termed pivotal coefficients, will become nearly squared at each application. It can be shown that if b_i and b_{i+j} are two adjacent pivotal coefficients, then there are exactly j roots having absolute value p given by

$$p^{jk} = (-1)^j \, b_{i+j}/b_i,$$

where $k = 2^m$, and m is the number of times the root-squaring has been performed.[23]

One can see that the coefficients become quite large when applying Graeffe's root-squaring method. If root-squaring is done very many times, an exponent overflow can occur. For this reason, the use of this method on the computer is limited. This problem can sometimes be alleviated by dividing all the roots by some number such as 2^m at each stage. However, this procedure could go too far the other way and cause exponent underflow. Therefore, the most practical use of Graeffe's method, especially at the high school level, is as a method of separating close roots. If the roots of a given polynomial are very close together, there is considerable trouble in finding them to a high degree of accuracy. However, after applying the root-squaring process several times, the largest root will grow fastest and will soon be easily isolated.

[23]Pennington, _op_. _cit_., p. 331.

Another disadvantage of Graeffe's method is that only the absolute values of the roots are found. The signs of the roots must somehow be determined. One method of determining the sign of a root is to substitute both the number and its negative in to the original equation and see which satisfies it best. If it is determined that k roots all have the same absolute value, it is difficult to find how many of these are real and how many are complex. After substituting in all the real possibilities, it should be assumed that the remaining roots are complex.

Once a reasonable effort has been made to find all the real roots of a given polynomial and its degree has been reduced as far as possible, then some method for determining the complex roots of the polynomial must be applied. Complex roots can be evaluated by many of the iterative methods already discussed. Consider, for example, applying Newton's method to the complex polynomial

$$z^3 - 2 = 0,$$

using $z_k = i$ as an initial approximation. Then

$$z_1 = i$$

$$z_2 = i - \frac{(i^3 - 2)}{3i^2} = -2/3 + 2/3i$$

$$z_3 = (-2/3 + 2/3i) - \frac{(-2/3 + 2/3i)^3 - 2}{3(-2/3 + 2/3i)^2} = -4/9 + 25/9i.$$

Obviously, the arithmetic involved in handling the complex quantities can soon become rather complicated.

An easier numerical method does exist for finding the complex roots of a polynomial equation of any degree. This method, suggested by Bairstow in 1914, searches for quadratic factors of the given polynomial. Since the complex roots occur in conjugate pairs, $a + bi$ and

$a - bi$, then any equation with complex roots has at least one quadratic factor.[24] This quadratic factor is of the form

$$(x - a - bi)(x - a + bi) = x^2 + px + q.$$

If p and q can be determined, then the complex roots can easily be found by applying the quadratic formula.

Consider the polynomial equation

$$P(x) = x^4 - x^3 + 13x^2 + 9x + 25 = 0.$$

Let $p_1 = 1$ and $q_1 = 2$ be initial approximations to p and q. Divide the original polynomial by the trial quadratic factor $(x^2 + x + 2)$, obtaining the quotient polynomial

$$Q(x) = x^2 - 2x + 13$$

with a remainder of -1. If the trial quadratic term were a factor of the original polynomial, then the remainder would be zero. Thus, what is now needed is a method of adjusting the approximations to p and q so that the remainder becomes zero.

In general, if p_1 and q_1 are initial approximations to p and q, then dividing the original polynomial

$$P(x) = a_1x^n + a_2x^{n-2} + \dots + a_nx + a_{n+1}$$

by the trial quadratic factor $(x^2 + p_1x = q_1)$ results in the quotient polynomial

$$Q(x) = b_1x^{n-2} + b_2x^{n-3} + \dots + b_{n-2}x + b_{n-1}$$

with a remainder of $b_n(x + p_1) + b_{n+1}$. Thus

$$\begin{aligned} P(x) &= a_1x^n + a_2x^{n-2} + \dots + a_nx + a_{n+1} \\ &= (x^2 + p_1x + q_1)(b_1x^{n-2} + \dots + b_{n-2}x + b_{n-1}) + b_n(x + p_1) + b_{n+1} \\ &= b_1x^n + (b_2 + p_1b_1)x^{n-1} + \dots + (b_{n+1} + p_1b_n + q_1b_{n-1}). \end{aligned}$$

---

[24] A. Balfour and A. J. McTernan, The Numerical Solution of Equations (London: Heinemann Educational Books Ltd., 1967), pp. 52-59.

Comparing coefficients of like terms results in

$$\begin{aligned}
a_1 &= b_1 \\
a_2 &= b_2 + p_1 b_1 \\
a_3 &= b_3 + p_1 b_2 + q_1 b_1 \\
&\cdot \\
&\cdot \\
&\cdot \\
a_n &= b_n + p_1 b_{n-1} + q_1 b_{n-2} \\
a_{n+1} &= b_{n+1} + p_1 b_n + q_1 b_{n-1}.
\end{aligned}$$

Hence the recursion formula

$$b_i = a_i - pb_{i-1} - qb_{i-2}\ , \quad i = 1, \ldots , n+1,$$

with $b_{-1} = b_0 = 0$, can be used to determine the coefficients of the quotient polynomial $Q(x)$ given the coefficients of the original polynomial $P(x)$.

The objective is to obtain approximations of p and q so that $(x^2 + px + q)$ is an exact divisor of $P(x)$. In order for this to be true, the remainder must be equal to zero. That is,

$$b_n = a_n - b_{n-1}p_1 - b_{n-2}q_1 = 0$$

and

$$b_{n+1} = a_{n+1} - b_n p_1 - b_{n-1}q_1 = 0.$$

The method of solving this system of two nonlinear equations in two unknowns is called the method of successive approximations.[25] The theory behind the process involves the use of partial derivatives and becomes rather complicated. However, the basic computational scheme is simple enough. Merely repeat the division process using the trial quadratic factor $(x^2 + p_1x + q_1)$ as the divisor, but this time using $Q(x)$ instead of $P(x)$ as the dividend. The result is a new quotient

[25] Kaj L. Nielsen, Methods in Numerical Analysis (New York: The Macmillan Company, 1956), p. 213.

polynomial

$$Q_2(x) = c_1x^{n-4} + c_2x^{n-5} + \dots + c_{n-2}x + c_{n-3},$$

with a remainder of $c_{n-2}(x + p_1) + c_{n-1}$. The c_k are obtained from the b_k just as the b_k were obtained from the a_k.

Using the values c_{n-2}, c_{n-1}, b_{n+1}, and b_n compute N_1, N_2, and D such that

$$N_1 = b_nc_{n-1} - b_{n+1}c_{n-2}$$

$$N_2 = b_{n+1}c_{n-1} - b_nc_n + b_n^2$$

$$D = c_{n-1}^2 - c_nc_{n-2} + b_nc_{n-2}.$$

Then, better approximations for p_1 and q_1 can be found by taking

$$p_2 = p_1 + N_1/D$$

$$q_2 = q_1 + N_2/D$$

When p and q are determined with sufficient accuracy, the two complex roots of the quadratic factor can easily be found by the quadratic formula.

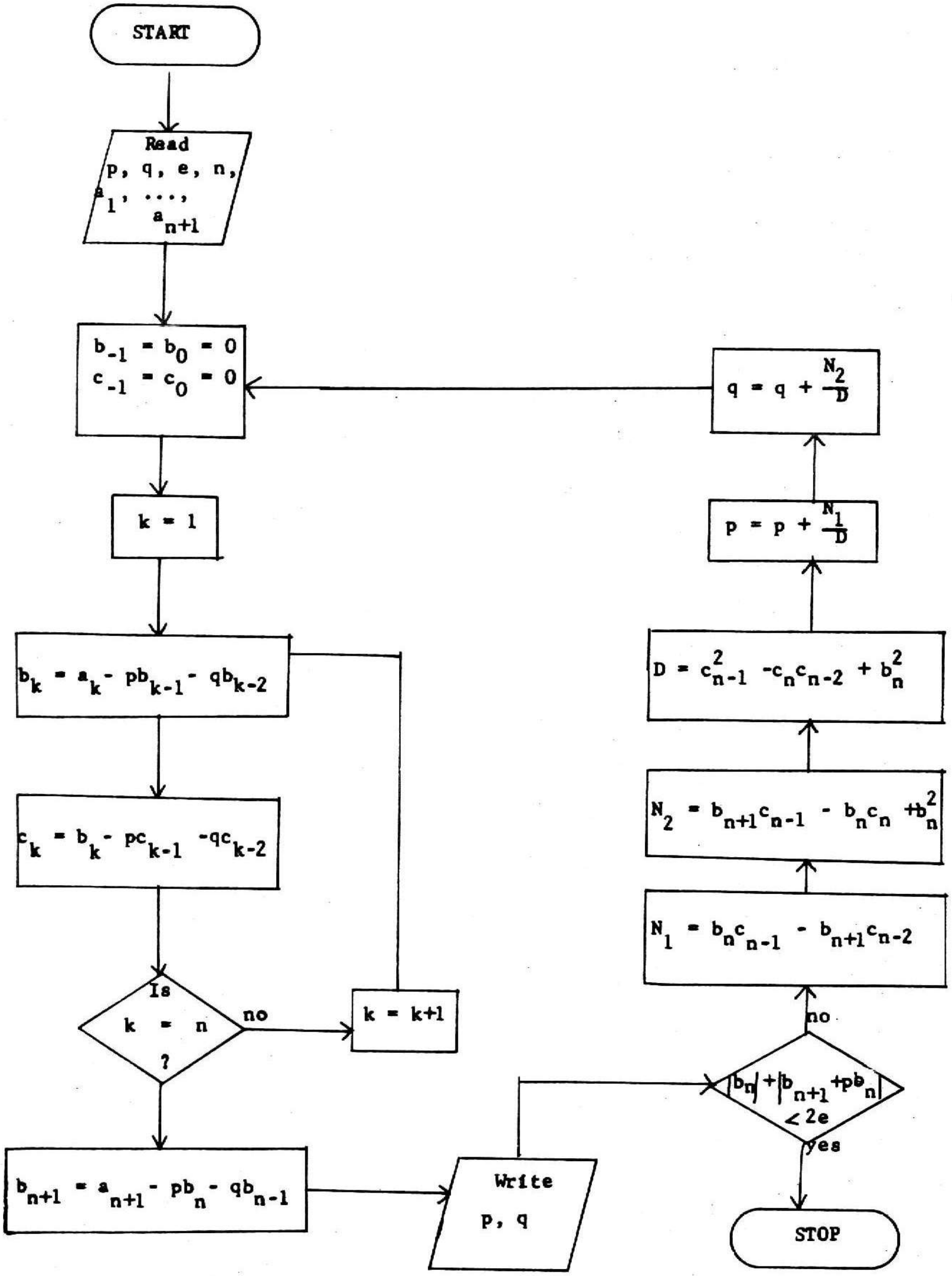
The flowchart in Figure 13 shows how easily Bairstow's method can be arranged for use on the computer. Applying Bairstow's process to the polynomial equation

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

results in the following approximations for p and q, using the initial approximations $p_1 = -1$ and $q_1 = 1$:

	p	q
1	-1.000000	1.000000
2	-2.000000	3.000000
3	-1.750000	1.750000
4	-2.058516	2.042340
5	-2.002377	2.000615
6	-2.000004	1.999997
7	-2.000000	2.000000

Therefore, the actual values of p and q are -2 and 2. Solving



START
Read $p, q, e, n, a_1, \ldots, a_{n+1}$
$b_{-1} = b_0 = 0$
$c_{-1} = c_0 = 0$
$k = 1$
$b_k = a_k - pb_{k-1} - qb_{k-2}$
$c_k = b_k - pc_{k-1} - qc_{k-2}$
Is $k = n$?
no
$k = k+1$
$b_{n+1} = a_{n+1} - pb_n - qb_{n-1}$
Write p, q
$|b_n| + |b_{n+1} + pb_n| < 2e$
yes
no
STOP
$N_1 = b_n c_{n-1} - b_{n+1} c_{n-2}$
$N_2 = b_{n+1} c_{n-1} - b_n c_n + b_n^2$
$D = c_{n-1}^2 - c_n c_{n-2} + b_n^2$
$p = p + \frac{N_1}{D}$
$q = q + \frac{N_2}{D}$

Figure 13. Bairstow's Method.

the quadratic equation

$$x^2 - 2x + 2 = 0$$

gives $x = (1 + i)$ and $x = (1 - i)$ as the two complex roots of the equation. Dividing $P(x)$ by $(x^2 - 2x + 2)$ gives the second quadratic equation of $(x^2 + 2)$. Solving this equation by the quadratic formula results in the two complex roots, $x = \sqrt{2}\,i$ and $x = -\sqrt{2}\,i$. These are also roots of the original equation.

Bairstow's method is a powerful and efficient method for finding complex zeros of polynomials. It provides a simple algorithm for computing a pair of complex roots of a polynomial having real coefficients by operating only with the real numbers, and it converges quadratically. However a good initial approximation to p and q is essential, and it is often difficult to select these initial values properly to assure convergence. In the absence of any other information, one can use as a trial quadratic factor the one suggested by the three terms of lowest degree in the given polynomial. That is, take

$$a_{n-1}x^2 + a_n x + a_{n+1} = 0$$

obtaining $p_1 = a_n/a_{n-1}$ and $q_1 = a_{n+1}/a_{n-1}$ as initial approximations for p and q.

One final root-finding method that should be mentioned is the Lehmer-Schur method.[26] This procedure is a relatively new one developed in 1961. It provides a criterion which may be used to determine whether any zero of a polynomial lies within the unit circle and uses this as the basis for a method to determine all the roots. Speed of convergence is in no way affected by the multiplicity or closeness of

[26] Anthony Ralston, A First Course in Numerical Analysis (New York: McGraw-Hill Book Company, 1965), pp. 355-59.

the roots. Although the application of the Lehmer-Schur procedure is too complex to be used in a general high school situation, an advanced student may wish to study this process further.

CONCLUSION

For any given equation it is obvious that several different iterative formulas are available for determining its roots. Some of these methods, such as the bisection method or the method of false position, converge very slowly but their convergence is assured. Others are more complicated, such as Newton's method, but their convergence, when it occurs, is very rapid. The type of root-finding method to be tried in a given situation depends first of all on the way the resulting computer program is to be used. For programs which will receive relatively little usage, a slower method with assured convergence would be better than a faster method which might diverge. However, for programs which are to be used repeatedly, a process which converges more rapidly would be desired.

Most root-finding formulas depend on the root being simple. If multiple zeros do exist, the bisection method and the method of false position will locate a zero of odd-order in a given interval, although convergence will be slowed down considerably. To find even-order zeros, one must examine the zeros of the first derivative. Multiple roots may also be found with Newton's method if the multiplicity of the root is known beforehand.[27] The following modification of Newton's iterative formula will find a root of multiplicity r, and will converge quadratically:

$$x_{i+1} = x_i - r \cdot \frac{f(x_i)}{f'(x_i)}$$

[27] *Ibid.*, p. 343.

However, the multiplicity of a desired root is usually unknown.

Another factor which must be considered when choosing a root-finding method is how much is known about the given polynomial. If a priori information about the location of the roots is poor, it is advisable to use a method which is not dependent upon the initial approximations. Such methods as the bisection method, method of false position, Bernoulli's method, and Graeffe's method have guaranteed convergence regardless of the starting values. They can be used to approximate the root locations so that one of the more rapidly-converging methods may be applied to determine the root to the desired degree of accuracy.

In general, the root-finding methods which are best suited for use on the computer in a high school situation are the secant method and the method of linear iteration using Aitken's acceleration device. The convergence of these methods is somewhat faster than linear. Also, the theory behind them is simple enough for the average student to understand. Of course, for students who have studied derivatives, Newton's method is even better than either of these. The bisection method and method of false position, because of their slow convergence, should be used only when one of the other methods fails. Bernoulli's method and Graeffe's can be used in special cases such as finding the root with the largest magnitude or separating close roots. If complex roots are desired, Bairstow's method would be the easiest one to apply.

Figure 14 contains a flowchart which illustrates a general procedure that can be followed for finding all the real and complex roots of a given polynomial. If no initial information is available about the location of the roots, one must first apply a stepping search

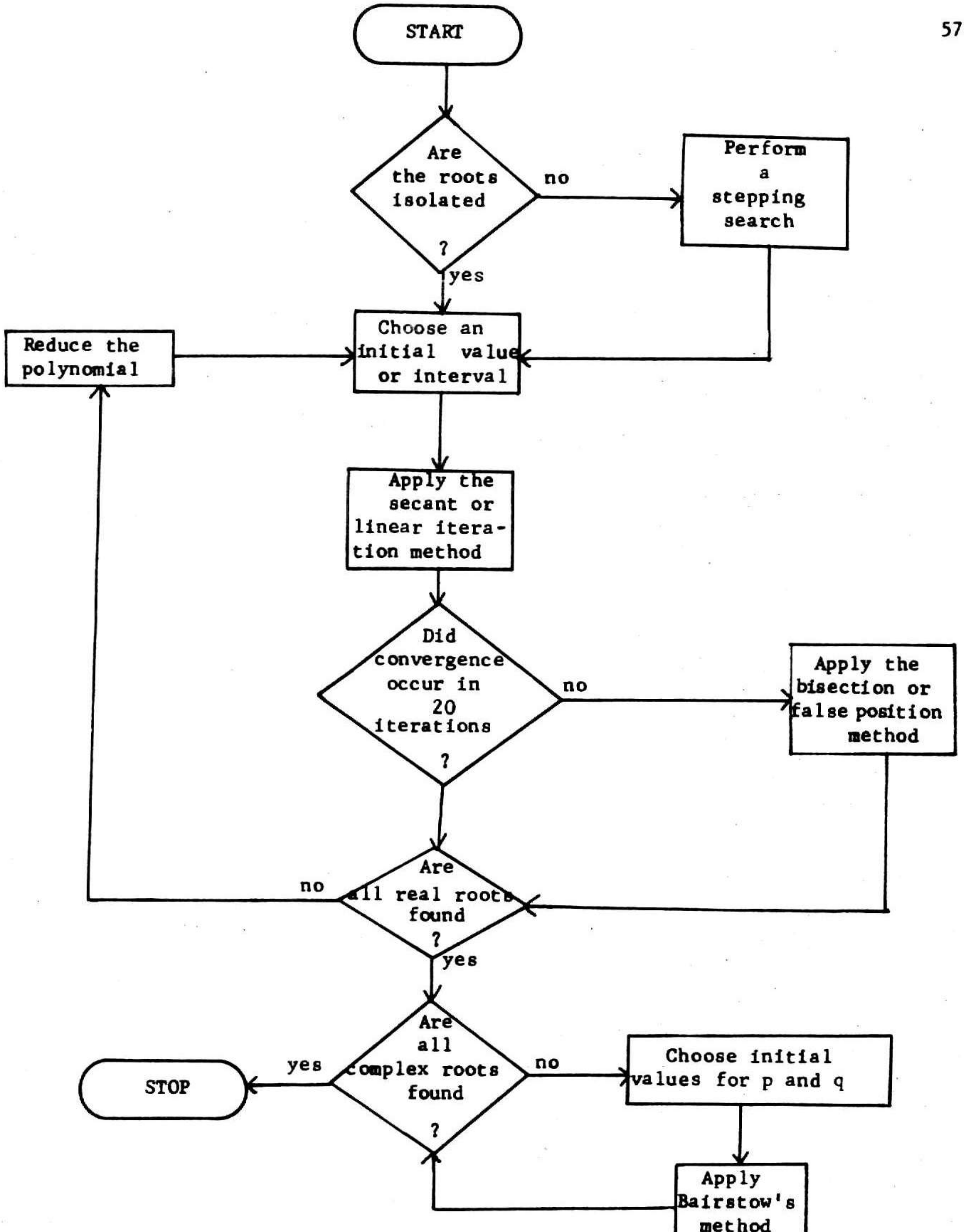

START
Are the roots isolated ?
no
Perform a stepping search
yes
Choose an initial value or interval
Reduce the polynomial
Apply the secant or linear itera-tion method
Did convergence occur in 20 iterations ?
no
Apply the bisection or false position method
Are all real roots found ?
no
yes
Are all complex roots found ?
yes
STOP
no
Choose initial values for p and q
Apply Bairstow's method

Figure 14. A general method for finding all the real and complex roots of a polynomial.

in an attempt to isolate all the real roots. One of these initial values or intervals can then be chosen as a starting point in applying some rapidly-converging method such as the secant method or the linear iteration method with acceleration. A counter should be incorporated into the program so that if convergence does not occur after a certain number of iterations, the program will switch to an always-convergent method such as the bisection method or the method of false position.

Once an approximation to a root has been determined, the degree of the polynomial can be reduced by dividing out the factor involving the obtained root. Then, a new initial value can be chosen and the process can be repeated. After a reasonable effort has been made to approximate all the real roots, Bairstow's procedure can be applied repeatedly to determine all the complex roots.

ACKNOWLEDGEMENT

The author wishes to express her sincere appreciation to Dr. Lyle J. Dixon for his patient guidance and assistance in the preparation of this report.

BIBLIOGRAPHY

BIBLIOGRAPHY

Acton, Forman S. Numerical Methods That Work. New York: Harper & Row, Publishers, 1970.

Arden, Bruce W. An Introduction to Digital Computing. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1963.

Balfour, A., and A. J. McTernan. The Numerical Solution of Equations. London: Heinemann Educational Books Ltd., 1967.

Carnahan, Walter H. "Iteration," School Science and Mathematics, LXVI (June, 1966), 551-55.

Conte, S. D. Elementary Numerical Analysis. New York: McGraw-Hill Book Company, 1965.

Dickson, Leonard Eugene. New First Course in the Theory of Equations. New York: John Wiley & Sons, Inc., 1939.

Forsyth, C. H. "A Simple and Effective Method of Solving a Polynomial," School Science and Mathematics, XV (December, 1915), 802-04.

Hamming, Richard W. Introduction to Applied Numerical Analysis. New York: McGraw-Hill Book Company, Inc., 1971.

Henrici, Peter. Elements of Numerical Analysis. New York: John Wiley & Sons, Inc., 1964.

Hesse, Allen R. "Iterative Methods in High School Algebra," Mathematics Teacher, LVII (January, 1964), 16-19.

Jacquez, John A. A First Course in Computing and Numerical Methods. Reading, Massachusetts: Addison-Wesley Publishing Company, 1970.

Kieran, Thomas E. "Quadratic Equations--Computer Style," Mathematics Teacher, LXII (April, 1969), 305-09.

Kovach, Ladis D. Computer-oriented Mathematics. San Francisco: Holden-Day, Inc., 1964.

Kunz, Kaiser S. Numerical Analysis. New York: McGraw-Hill Book Company, 1957.

Kuo, Shan S. Numerical Methods and Computers. Reading, Massachusetts: Addison-Wesley Publishing Company, 1965.

Luke, Yudell L. "Numerical Analysis and High School Mathematics," Mathematics Teacher, L (November, 1957), 507-512.

Macon, Nathaniel. Numerical Analysis. New York: John Wiley & Sons, Inc., 1963.

Nielsen, Kaj L. Methods in Numerical Analysis. New York: The Macmillan Company, 1956.

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

Rabinowitz, Philip. Numerical Methods for Nonlinear Algebraic Equations. New York: Gordon and Breach, Science Publishers, Ltd., 1970.

Ralston, Anthony. A First Course in Numerical Analysis. New York: McGraw-Hill Book Company, 1965.

Stibitz, George R., and Jules A. Larrivee. Mathematics and Computers. New York: McGraw-Hill Book Company, Inc., 1957.

APPENDIX

```
C SUBROUTINE FOR EVALUATING A FUNCTION AND ITS DERIVATIVE.

      SUBROUTINE  F(N, A, Y, YP, X)
      DIMENSION  A(10)
      M = N - 1
      B = A(1)
      C = A(1)
      DO  2  K = 1, M
      B = A(K + 1) + Z * B
    2 C = B + Z * C
      Y = A(N + 1) + Z * B
      YP = C
      RETURN
      END
```

```
C  BISECTION METHOD
      DIMENSION A(10)
      READ  (2, 10)  N,  (A(I), I = 1, N), X1, X2
   10 FORMAT (I2, 12F5.1)
      DO  2  I = 1, 20
    2 X = (X1 + X2) / 2.
      CALL F(N, A, Y, YP, X)
      WRITE  (5, 20)  X, Y
   20 FORMAT  (2 (F9.6, 3X) )
      IF  (ABS (X2 - X) - .00001 )  3, 3, 4
    4 IF  (Y)  5, 3, 6
    5 X1 = X
      GO TO 2
    6 X2 - X
    2 CONTINUE
      WRITE  (5, 30)
   30 FORMAT ('FAILED TO CONVERGE IN 20 ITERATIONS')
      CALL EXIT
    3 END
```

```
C  METHOD OF FALSE POSITION

      DIMENSION  A(10)
      READ  (2, 10)  N, (A(I), I = 1, N), X1, X2
   10 FORMAT (I2, 12F5.1)
      DO 6  I = 1, 20
      CALL F(N, A, Y1, YP1, X1)
      CALL F(N, A, Y2, YP2, X2)
      X3 = X1 + (Y1 * (X2 - X1) ) / (Y1 - Y2)
      CALL F(N, A, Y3, YP3, X3)
   20 FORMAT  (' ', I2, 2X, 6(F9.6, 2X) )
      IF  (ABS (Y3) - .00001 ) 2, 2, 3
    3 IF  (Y3)  4, 5, 5
    5 X2 = X3
      GO TO 6
    4 X1 - X3
      6 CONTINUE
      WRITE  (5, 30)
   30 FORMAT ('FAILED TO CONVERGE IN 20 ITERATIONS')
    2 CALL EXIT
      END
```

```
C  A MODIFICATION OF THE METHOD OF FALSE POSITION

      DIMENSION  A(10), X(20), Y(20)
      READ  (2, 10), N, ( A(I), I = 1, N), X1, X2
   10 FORMAT (10F5.1)
      X(1) = X1
      X(2) = X2
      DO  6  I = 1, 20
      CALL  F(N, A, Y1, YP1, X1)
      CALL  F(N, A, Y2, YP2, X2)
      Y(1) = Y1
      Y(2) = Y2
      X3 = X1 + (Y1 * (X2 - X1) ) / (Y1 - Y2)
      CALL  F(N, A, Y3, YP3, X3)
      X(I + 2) = X3
      Y(I + 2) = Y3
      WRITE  (5, 20)  I, X1, Y1, X2, Y2, X3, Y3
   20 FORMAT ( ' ', I2, 2X, 6(F9.6, 2X) )
      IF  (ABS (Y3) - .00001)  2, 2, 3
    3 IF  ( Y(I + 2) * Y(I + 1) ) 4, 2, 5
    4 X1 = X(I + 2)
      GO TO 6
    5 X1 = ( X(I) + X(I + 2) ) / 2.
      X2 = X(I + 2)
    6 CONTINUE
      WRITE  (5, 30)
   30 FORMAT ( 'FAILED TO CONVERGE IN 20 ITERATIONS' )
    2 CALL EXIT
      END
```

```
C  THE METHOD OF LINEAR ITERATION

      F(X) = X ** 3 + 2. * X ** 2 + 10. * X - 20.

      G(X) =  20. / (X ** 2 + 2. * X + 10.)

      GP(X) = (-40.) * (X + 1.) / (X ** 2 + 2. * X + 10.) ** 2

      READ (2, 10)  XO

   10 FORMAT  (F5.1)

      X = XO

      IF (ABS (GP(X) - 1.) 1, 4, 4

    1 WRITE (5, 15) X, Y

   15 FORMAT ( ' ', 2(F9.6), 2X) )

      DO  3  I = 1, 20

      X1 = G(X)

      Y1 = F(X1)

      WRITE (5, 20)  I, X1, Y1

   20 FORMAT  ( ' ', I2, 2X, 2(F9.6, 2X) )

      IF  (ABS (X1 - X) - .00001)  5, 3, 3

    3 X = X1

      WRITE  (5, 30)

   30 FORMAT ('FAILED TO CONVERGE IN 20 ITERATIONS')

      GO TO 5

    4 WRITE (5, 40)

   40 FORMAT  ('CONVERGENCE NOT ASSURED')

      5 CALL EXIT

      END
```

```
C  THE METHOD OF LINEAR ITERATION USING AITKEN'S DELTA**2 PROCESS.

      F(X) = X ** 3 + 2. * X ** 2 + 10. * X - 20.

      G(X) = 20. / (X ** 2 + 2. * X + 10.)

      GP(X) = (-40.) * (X + 1.) / (X ** 2 + 2. * X + 10.) ** 2

      READ  (2, 10)  XO

   10 FORMAT (F5.1)

      X = XO

      IF  (ABS (GP(X) - 1.) 1, 4, 4

    1 WRITE  (5, 15)  X, Y

   15 FORMAT ( ' ', 2(F9.6, 2X) )

      DO  11  I = 1, 20, 3

      X1 = G(X)

      Y1 - F(X1)

      WRITE  (5, 20)  I, X1, Y1

   20 FORMAT  ( ' ', I2, 2X, 2(F9.6, 2X) )

      IF  (ABS (X1 - X) - .00001)  12, 13, 13

   13 X2 - G(X1)

      Y2 = F(X2)

      I1 - I + 1

      WRITE  (5, 20)  I1, X2, Y2

      IF  (ABS (X2 - X1) - .00001)  12, 14, 14

   14 X = X - ( (X1 - X) ** 2 ) / (X2 - 2. * XL + X)

      Y = F(X)

      I2 - I1 + 1

      WRITE  (5, 20)  I2, X, Y

      IF  (ABS (X - X2) - .00001) 12, 11, 11

   11 CONTINUE
   WRITE  (5, 70)
70 FORMAT ('FAILED TO CONVERGE IN 20 ITERATIONS')
   GO TO 12
 4 WRITE  (5, 40)
40 FORMAT ('CONVERGENCE NOT ASSURED')
12 CALL EXIT
   END
```

```
C  NEWTON'S METHOD

      DIMENSION  A(10)
      READ  (2, 10)  N,  ( A(I), I = 1, N ), X
   10 FORMAT  (I2, 12F5.1)
   11 CALL  F(N, A, Y, YP, X)
      WRITE  (5, 20)  X, Y
   20 FORMAT  ( ' ', 2(F9.6) )
      DO  6  I = 1, 20
      X1 = X - Y / YP
      CALL  F(N, A, Y, YP, X1)
      WRITE  (5, 30)  I, X1, Y
   30 FORMAT  ( ' ', I2, 2X, 2(F9.6, 2X) )
      IF  (ABS (X1 - X) - .00001) 3, 3, 4
    4 IF  (ABS (YP) - .01)  5, 5, 6
    6 X = X1
      WRITE  (5, 40)
   40 FORMAT  ('FAILED TO CONVERGE IN 20 ITERATIONS')
      GO TO 3
    5 WRITE  (5, 50)
   50 FORMAT  ('DERIVATIVE IS TOO SMALL')
    3 CALL EXIT
      END
```

```
C  BERNOULLI'S METHOD

      DIMENSION  P(30), A(30), R(30)
      READ  (2, 5)  N
    5 FORMAT  (I2)
      M = N + 1
      READ  (2, 20)  ( P(I), I - 1, N), ( A(I), I - 1, M)
   10 FORMAT  (16F5.2)
   11 DO  2  J = 1, 15
      K = J
      P1 = 0
      DO  3  L = 1, N
      P1 = A(L) * P(K) + P1
    3 K = K + 1
      I = J + N
      P(I) = (-PL) / (A(M) )
      WRITE  (5, 20)  P(I)
   20 FORMAT  (F16.8)
    2 CONTINUE
      DO  4  J = 1, 20
      K = J + N
      R(J) = P(K + 1) / P(K)
    4 WRITE  (5, 20)  R(J)
      CALL EXIT
      END
```

```
C  THE SECANT METHOD

      DIMENSION  A(10)
      READ  (2, 10) N, ( A(I), I = 1, N), X1, X2
   10 FORMAT  (10F5.1)
   11 DO  6  I = 1, 20
      CALL  F(N, A, Y1, YP1, X1)
      CALL  F(N, A, Y2, YP2, X2)
      X3 = X1 + (Y1 * (X2 - X1) ) / (Y1 - Y2)
      CALL  F(N, A, Y3, YP3, X3)
      WRITE  (5, 20)  I, X1, Y1, X2, Y2, X3, Y3
   20 FORMAT  ( ' ', I2, 2X, 6(F9.6, 2X) )
      IF  (ABS (Y3) - .00001) 2, 2, 5
    5 X1 = X2
      X2 = X3
    6 CONTINUE
      WRITE  (5, 30)
   30 FORMAT ('FAILED TO CONVERGE IN 20 ITERATIONS')
    2 CALL EXIT
      END
```

```
C  A ROOT-SQUARING SUBROUTINE

      SUBROUTINE  ROOTS (A, B, N)
      DIMENSION  A(20), B(20)
      B(1) = A(1) ** 2
      DO 6 I = 2, N
      B(I) = 0
      K - I - 1
      KK = N - 1
      IF  (K - KK) 4, 4, 3
    3 K = KK
    4 DO 5 J = 1, K
      L = K + 1 - J
      IL = I + L
      LI = I - L
    5 B(I) = A(IL) * A(LI) - B(I)
      SIGN = 2 * (I - 2 * (I/2) ) - 1
    6 B(I) = SIGN * (A(I) ** 2 - 2. * B(I) )
      SIGN = 1 + 2 * (2 * (N/2) - N)
      B(N + 1) = SIGN * A(N + 1) ** 2
      RETURN
      END
```

```
C  GRAEFFE'S ROOT-SQUARING METHOD

      DIMENSION  A(20), B(20), C(20)
      READ  (2, 10) N
   10 FORMAT  (I2)
      M = N + 1
      READ  (2, 20) (A(I), I - 1, M)
   20 FORMAT  (10F5.1)
      K = 2
      DO  4  J = 1, 6
    2 CALL ROOTS  (A, B, N)
      DO  3  I - 1, M
    3 E(I) = A(I) ** 2 / (B(I) -1)
      WRITE  (5, 30) K
   30 FORMAT  (I4)
      WRITE  (5, 35)  (B(I), I = 1, M), (ER(I), I = 1, M)
   35 FORMAT  (E16.7)
    5 DO  6  I = 1, M
    6 A(I) = B(I)
    4 K = 2 * K
    7 PK = K
      PWR = 1. / PK
      DO  8  I = 1, N
      X - B(I + 1) / B(I)
      WRITE (5, 40) X
   40 FORMAT  (F18.6)
      CALL EXIT
      END
```

```
C  BAIRSTOW'S METHOD

      REAL  N1, N2
      DIMENSION  A(20), B(20), C(20)
      READ (2, 5) N
    5 FORMAT  (I2)
      M = N + 3
   11 READ (2, 10) P, Q, ( A(I), I = 3, M)
   10 FORMAT  (16F5.2)
      DO  7  I = 1, 10
      WRITE  (5, 20)  P, Q
      B(1) = 0
      B(2) = 0
      C(1) = 0
      C(2) = 0
      DO  2  K = 3, M
      B(K) = A(K) - P * P(K-1) - Q * B(K-2)
    2 C(K) = B(K) - P * C(K-1) - Q * C(K - 2)
      IF  (ABS (B(M - 1)) + ABS (B(M) + P * B(M - 1)) - .00001) 4, 4, 3
    3 N1 = B(M - 1) * C(M - 2) - B(M) * C(M - 3)
      N2 = B(M) * C(M - 2) - B(M - 1) * C(M - 1) + B(M - 1) ** 2
      D = C(M - 2) ** 2 - C(M - 1) * C(M - 3) + B(M - 1) * C(M - 3)
      P = P + N1/D
    7 Q = Q + N2/D
    4 WRITE  (5, 20)  P, Q
   20 FORMAT ( ' ', 2(F9.6, 2X) )
   21 CALL EXIT
      END
```

COMPUTER METHODS FOR SOLVING POLYNOMIAL EQUATIONS

by

JUDITH ARLENE BENNETT

B. S., Bethany College, 1969

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mathematics

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1972

The problem of finding the zeros of a given polynomial equation arises frequently in mathematics, physics, and many branches of engineering. Rarely do these problems possess simple, analytical solutions. It is therefore of great importance that numerical methods for the solution of such equations be made available to the high school student.

A variety of numerical root-finding techniques have long been available, but their use was limited because of the extensive numerical calculations that they required. However, with the advent of the computer, the use of these procedures is now both possible and practical. Although the problem of finding the roots of a polynomial on the computer is by no means trivial, it is a problem solvable in the majority of cases by straightforward application of simple procedures. The purpose of this paper was to examine and compare the available methods for finding the roots of polynomial equations and decide which methods would be most suitable for use in a computer-oriented high school mathematics class.

This study explored various methods of solving polynomial equations and attempted to present them at a level that the average high school student could understand. The application of each procedure was demonstrated with a simple example using a polynomial of low degree. Flowcharts were developed for those processes that are most easily adapted to computer use, and the actual output from a computer run of a given problem was presented. Advantages and disadvantages of each method were also discussed.

Many factors enter into a decision concerning which root-finding method to use in a given situation. One must consider how the program will be used, what information is available about the location of the roots, the possible existence of multiple and complex roots, and many other factors before choosing a particular method. However, it was concluded that the methods which are generally best suited for use in a high school situation are the secant method and the method of linear iteration using Aitken's acceleration device. Either of these root-finding methods can be used in most cases to determine the real roots of a given polynomial. If either of them fails to converge, then one of the always-convergent methods such as the bisection method or the method of false position can be used. When complex roots are desired, the real roots can first be found and the degree of the polynomial can be reduced. Then Bairstow's mathod can be repeatedly applied until all the complex roots have been determined.