1967

Comparative analysis of polynomial root finding techniques

Charles Ray O'Daniel

Department:

COMPARATIVE ANALYSIS OF POLYNOMIAL ROOT FINDING TECHNIQUES

BY

CHARLES RAY O'DANIEL - 1943 -

A

THESIS

submitted to the faculty of

THE UNIVERSITY OF MISSOURI AT ROLLA

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Rolla, Missouri

1967

Approved by

B C Gillett (advisor) Ralph E. Lee

Robert B. Wilhelm W. J. Murphy

ABSTRACT

The purpose of this study was to investigate and recommend various methods instrumental in finding the roots of a polynomial $p(x) = 0$. Many different methods are present today, and each has its advantages and disadvantages.

Through thorough investigation, the author has ascertained the key methods to be the method of Bisection, the Newton-Raphson method, and the Bairstow method. Special support in the form of algebraic theorems on the locations and kind of roots are extremely helpful. This combination of theorems and methods provides assurance, speed, and the ability to obtain complex roots.

The Bisnewbar method developed by this author combines the above methods and the algebraic theorems to provide a method capable of returning all real and complex roots.

ACKNOWLEDGEMENT

The author wishes to extend his deepest appreciation to Dr. Billy E. Gillett for his understanding and guidance throughout the preparation of this thesis. His personality and inspiration have helped to make this study both rewarding and enjoyable.

The author also wishes to thank Mr. Lauren A. Peterson for the Graduate Assistantship which made graduate school financially possible.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

INTRODUCTION

One of the oldest problems plaguing mathematicians and scientists today is the age-old problem of finding the zeroes (roots) of polynomial equations. The fact that there are so many methods from which to choose indicates the fact that no completely satisfactory method exists. Each person has his own favorite method, or methods, but nevertheless cases arise in which one has to resort to other methods when his fails to obtain the desired solution.

Upon encountering the polynomial root-finding problem, mathematicians discovered a need for basic information on the properties and locations of the roots. Since many methods used today require starting values the above information could be quite helpful in obtaining these. Basic algebraic theorems were developed such as Descartes and Sturm which allow the analyst information on the number, kind, and location of real roots. Using these theorems, one may obtain needed support for finding the roots of a polynomial.

There are three basic problems in which established algebraic theorems have helped to a great extent in polynomial solutions. The first concerns the approximation of one root and its solution. Here we may use the common method of graphing to approximate the root. Once an approximation is made, we may use numerical methods to improve our approximation. Secondly, occurs the problem of determining approximations for all the roots. This may effectively be accomplished by repeated

use of algebraic theorems or by use of graphical methods. The first two problems differ in that possibly only one root is desired and further computation would lead to a waste of valuable computer time. The third problem is one of determining the number of roots in a certain vicinity. Again we can rely on special algebraic theorems to aid in this procedure.

Direct and iterative methods have been developed to solve the problem of polynomial root-finding, but it is generally agreed that iterative methods obtain the fastest and best results. Even so, there exists the problem of convergence. Almost every method has certain conditions which, if the polynomial meets these conditions, will cause the process to diverge. Hence, the desired root will never be obtained. Therefore, it is sometimes a painstaking process to check the polynomial for convergence conditions, and often times this check involves much computation.

In summary, we conclude that although special theorems have helped us in the solution of polynomials, many problems are still encountered. Once the initial approximation is obtained, a method must be selected which will converge to the root or roots desired and will give an accurate solution. As mentioned above, these problems are the heart of the solution of polynomial equations and must be overcome before a true solution may be obtained.

It is the goal of this author to investigate many methods of polynomial root-finding. Books could be written solely on all the different methods, therefore, the author

has attempted to choose those methods which are either highly accepted now or show promise for the future. Some of the methods considered will be methods that have been used for a number of years, while others may be relatively new to the field.

Special investigations will be made in the area of convergence. Many methods will be compared as to time of convergence, number of iterations, etc. An internal timer common to the IBM System/360 will provide great assistance in this endeavor.

Attempts will be made to combine certain methods in hopes that a method can be obtained that will converge to all the real roots of a polynomial equation.

The author feels the following two quotes by Hamming [1] represent the basic problems of polynomial root-finding:
(1) ".....the fancier the method and the better it is supposed to be, the worse it can behave when things go wrong for some functions; it may, in fact, be worse, than simpler methods and quite likely is more vulnerable to noise."
(2) ".....it is an art to isolate the various zeroes."

REVIEW OF LITERATURE

The area of polynomial root-finding is not only an old problem, but is probably the oldest problem still existing in the field of mathematics. Much work and research has been devoted to the area, but as of this date no "always-best" method for solving

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$$

has been discovered. Many methods exist, but each has its disadvantages.

A sufficient study into the area of polynomial solutions would not be complete without a nodding acquaintance with the algebraic theorems related to this field. These theorems provide the lifeline to the proper analysis of root locations and types of roots. A thorough knowledge of the following theorems can help us in many ways to obtain valuable information on the roots of $p(x)$.

Theorem 1. If the coefficients in the polynomial equation $p(x) = 0$ are real, and if a and b are real numbers such that $p(a)$ and $p(b)$ have opposite signs, then the equation has at least one real root between a and b. There is, in fact, an odd number of roots in the interval (a,b) if a k-fold root is counted k times.

Theorem 2. Let $p(x) = 0$ be a polynomial equation with real coefficients which is arranged in descending powers of x. The number of positive roots of the equation is either equal to the number of variations of sign presented by the coefficients of $p(x)$ or less than the number of variations by a positive even integer. Here a root of multiplicity m is to be counted as m roots. In particular, there is exactly one positive root if the coefficients present only one variation of sign.

Theorem 3. Let $p(x) = 0$ be an algebraic equation with real coefficients and without multiple roots. If a and b are real numbers, $a<b$, and neither a root of the given equation, then the number of real roots of $p(x) = 0$ between a and b is equal to $V(a) - V(b)$, where $V(a)$ and $V(b)$ denote variations in sign.

Almost every method in existence today requires some kind of an initial approximation to the root(s). Some methods require a quite good initial approximation while in others the matter is not so critical. In some cases, rare but possible, the actual data giving rise to an equation will serve to fix a root as lying between two fairly narrow limits, and this fact may eliminate the necessity of using any special device for obtaining an approximation to the root. Generally speaking, however, the best method for locating an initial approximation to a root is to plot the function $y = p(x)$, where $p(x)$ is a polynomial of the form $a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$, and determine approximately the point at which the plot crosses the x axis. Another method similar to the one mentioned above is to write the polynomial in the form $p_1(x) = p_2(x)$, where $p(x) = p_1(x) - p_2(x)$ and plot as two different functions $y_1 = p_1(x)$ and $y_2 = p_2(x)$. In plotting these functions, it is a good idea to determine as much of the following information as possible: (1) the behavior of the function as $x \to -\infty$ and as $x \to \infty$, (2) the value of $p(x)$ for those values of x which facilitate a quick determination of $p(x)$, (usually $x = 0$ and $x = \pm 1$ are satisfactory for this purpose), (3) the value(s) for which $p(x)$ becomes infinite, (4) the intercept of the functions on the x and y axis.

Usually enough of the above information is available to sketch a rough graph of the function and thereby to determine the approximate location of each of the real roots. By this method we may in effect isolate the roots by separating them into different regions. Accuracy can be increased by evaluating additional points near the roots and may help to isolate two or more adjacent roots.

There are two basic techniques used in the solution of polynomial equations. The first method is considered the direct approach to the solution. An example of a direct method would be the solution of a quadratic equation through use of the quadratic formula. A direct method is said to be one which will produce an exact answer after a finite number of operations. For equations of degree two, three, and four there are special methods to achieve the roots of a polynomial, but for polynomials of degree greater than four; there exists no special (direct) method. Direct methods do possess one major disadvantage in that they may not be the most desirable for computer calculations because of round-off and propagation errors which often compound themselves. Propagation errors are those errors resulting from the subsequent growth of generated errors.

The second and most widely used technique is the iterative approach. This method usually consists of making an initial approximation to a root(s) and using one of several iterative techniques to converge to the desired root(s). Grove [2] feels that a useful iterative technique should

possess the following requirements: (1) a means of making a satisfactory first guess; (in many applications physical or other consideration may provide this guess), (2) a means of systematically improving on the previous approximation, (3) a criterion or choice of several criteria for stopping the iteration when sufficient accuracy has been obtained.

Iterative techniques are extremely useful in programming digital computers for two important reasons: (1) the use of the same set of instructions repetitively saves space in the computer's memory, (2) the round-off errors made are minimized, whereas in direct methods they compound themselves.

We might summarize the use of the iterative technique into two parts, the first being the survey where the general characteristics of the system are investigated and approximate locations of roots are noted, the second part being that of refinement; that is, the root(s) is found to the desired degree of accuracy.

It is of general agreement that iterative techniques are the best tools for solving polynomial equations.

It is possible to obtain one root to a desired degree of accuracy and factor this known root from the polynomial leaving a degree of n - 1. Once the polynomial has been reduced to degree four or less, direct methods of solution may be utilized.

A technical definition of an iterative technique given by Booth [3] is as follows: If x_0 is an approximation to the solution of $p(x) = 0$, the iterative process enables a quantity

x_1 to be calculated by means of some relation: $x_1 = f(x_0)$ in such a manner that x_1 is a closer approximation to the required solution than was x_0. If x_0 differs from the true root by a small quantity of order (ε), say, then the iterative process $x_1 = f(x_0)$ is said to be n^{th} order if the error in x_1 is of order (ε^n). Primarily, the best iterative processes are second order; third and higher order processes exist and can always be constructed from those of lower order, but often they involve greater total computing labor for a given final accuracy than those of second order.

Hartree [4] offers the following ideas on iterative procedures. In many cases our iteration is accomplished in the form $x = f(x)$; where $x_{n+1} = f(x_n)$. If to the degree of numerical accuracy to which the work is carried out, $x_{n+1} = x_n$, then this value of x_{n+1} is a solution of the equation to that degree of accuracy. Let $x = \underline{X}$ be a solution of the equation and let $x_n = \underline{X} + \varepsilon_n$ so that ε_n is the error in x_n. An important feature of an iterative method is the way in which this error varies with the number n of repetitions of the iterative process. This can be examined by expanding the right hand side of $x_{n+1} = f(x_n)$ in a Taylor's series. Then since $x = \underline{X}$ satisfies

$$x = f(x) \Rightarrow \varepsilon_{n+1} = a_1 \varepsilon_n + a_2 \varepsilon_n^2 + \ldots + a_n \varepsilon_n^n$$

where

$a_k = f^{(k)}(x)/k!$ If $a_1 \neq 0$ then the errors ε_n of

results of successive repetitions of the iterative process are ultimately related by $\varepsilon_{n+1} = a_1\varepsilon_n$, $\varepsilon_{n+m} = a_1^m\varepsilon_n$; in order that the process should converge, $|a_1| = |f'(x)|$ must be less than 1, and the magnitude of the error then decreases exponentially with n increasing. This means that the number of additional correct significant figures obtained from each repetition of such a process (or, more often, the number of repetitions required to obtain each new correct significant figure) is the same, however many figures have been obtained. Such a process is called "first-order." But if $a_1 = 0$, $a_2 \neq 0$, in $x_n = \underline{X} + \varepsilon_n$, then the successive errors ε_n are ultimately related by $\varepsilon_{n+1} = a_2\varepsilon_n^2$ or $a_2\varepsilon_{n+m} = (a_2\varepsilon_n)^{2^m}$, where $a_2 = 1/2\ f''(x)$. The number of correct significant figures is approximately doubled for each repetition of the iterative process, so that the better the approximation of x_n to X, the easier it is to improve it further. Such a process is called "second-order," and once a fair approximation to $x = \underline{X}$ has been attained, a second order process is greatly preferred over a first order one; but it must be started from an approximation good enough to insure that $|a_2\varepsilon_0|<1$.

If $a_1 = 0$, $a_2 = 0$, $a_3 \neq 0$ then the successive errors ε_n are ultimately related by $\varepsilon_{n+1} = a_2\varepsilon_n^3$ or $a_3^{\frac{1}{2}}\varepsilon_{n+m} = (a_3^{\frac{1}{2}}\varepsilon_n)^{3^m}$; such a process is called "third-order." The formula for a third order process is usually more complicated than that for a second-order process for the same equation.

The convergence for a second-order process is already so fast once a good approximation has been obtained that the advantage of still quicker convergence obtainable from a third order process may be more than offset by the more complicated formulae which have to be evaluated for each repetition of the iterative process. Third-order processes are not used much in practice. Second-order processes, however, are widely used.

Let us now consider the general problems of convergence and divergence. Convergence or divergence usually depends upon the particular form in which $x = f(x)$ is expressed.

In order to see why this is so, Hildebrand [5] suggests that, since $f(\alpha) = \alpha$, $z_{k+1} = f(z_k)$ implies the relation $\alpha - z_{k+1} = f(\alpha) - f(z_k) = (\alpha - z_k)f(\xi_k)$, where ξ_k lies between z_k and α, under the assumption that $f(x)$ possesses a continuous derivative over that range. If the iteration converges so that $z_k \to \alpha$ as $k \to \infty$, then, for sufficiently large k, we must have $f'(\xi_k) \approx f'(\alpha)$, and hence $\alpha - z_k \approx A[f'(\alpha)]^k$, where A is a constant, and this deviation tends to zero as k increases only if $|f'(\alpha)| < 1$. Thus it appears that, in order for the iteration to converge to $x = \alpha$, it is necessary that $|f'(x)| < 1$ in the neighborhood of $x = \alpha$.

If we here define the convergence factor p_k as the ratio of the error in z_{k+1} to the error in z_k, it follows that if z_k is near α, then $p_k \approx f'(\alpha)$. Unless $|f'(\alpha)| \leq 1$, a small error in z_k is increased in magnitude by the iteration,

and we say that the iteration is then asymptotically unstable at α. The number $f'(\alpha)$ may be called the asymptotic convergence factor.

If the initial approximation is sufficiently near α, and if the iteration is asymptotically stable at α[so that $|f'(\alpha)|<1$], the sequence of iterates will indeed converge to α in such a way that ultimately the successive approximations tend toward α from one direction if $0<f'(\alpha)<1$, and oscillate about α with decreasing amplitude if $-1<f'(\alpha)<0$.

The next several sections will be detailed discussions of many of the methods being used today. A few of the newer methods will also be discussed.

METHOD OF BISECTION

Pennington [6] presents some of the following ideas on the method of Bisection. This method evolves from the following theorem: Suppose a continuous function $p(x)$ is negative at $x = a$ and positive at $x = b$, then there is at least one root between a and b. Calculation is initialized by the evaluating of $p[(a+b)/2]$, the value of the function halfway between a and b. If this is zero, we have the root. If it is negative, the root is between that point and a. Thus either we have the root or we have it bracketed within an interval half as large as the previous one. This process can be continued, each time bisecting the interval. It can be continued until the root is known to the desired accuracy.

This method while simple in technique does have several virtues. The greatest virtue of the method is the fact that it is virtually assured to converge to a root. It can fail to do so under the unusual circumstance that an accumulation of errors would cause y at some step to be calculated, say, as a small negative value when actually it should have a small positive value. The computer could be halving the wrong interval from then on. If proper care has been taken concerning accuracy, this should not occur. The greatest drawback of the bisection method is the fact that it is slow compared to some of the other methods and that it can be applied only when the function is negative at one value of x and positive at another.

METHOD OF ITERATION

This method well presented by Conte [7] requires the equation $p(x) = 0$ to be expressed in the form $x = f(x)$ such that any solution of the second equation is also a solution of the first. In general, there are many ways in which to express $x = f(x)$ not all of which are satisfactory for this method. Geometrically a root of $x = f(x)$ is a number $x = \xi$ for which the line $y = x$ intersects the curve $y = f(x)$. It may happen that these curves do not intersect, in which case there will be no real solution. We shall assume, however, that these curves do intersect at least once; that we are interested in finding one of these roots, say $x = \xi$; and that

$f(x)$, $f'(x)$ are continuous in an interval about this root.

Theorem 4. Let $x = \xi$ be a root of $p(x) = 0$; let I be an interval containing the point $x = \xi$ (i.e., I is the set of all points x satisfying the inequality $|x-\xi|<\varepsilon$ for a given ε). Let $g(x)$, $g'(x)$ be continuous in I. Then if $|f'(x)|\leq k<1$ for all points in I, and if the initial approximation x_0 is chosen in I, the iteration converges to the root ξ.

From the above theorem it can be shown through differential calculus that $x_{i+1} - \xi = f'(n_i)(x_i-\xi)$. If we define the error e_i as $e_i = x_i - \xi$ we have the equation $x_{i+1} - \xi = f'(n_i)(x_i - \xi)$ revised as

$$e_{i+1} = f'(n_i)e_i \quad . \tag{2.01}$$

When the iteration does converge we see from (2.01) that

$$\lim_{i\to\infty} \frac{e_{i+1}}{e_i} = \lim_{i\to\infty} f'(n_i) = f'(\xi). \tag{2.02}$$

Since n is between x_i and ξ, and since $f'(x)$ is continuous (2.02) states that for large values of i the error at any iteration is proportional to the error at the previous iteration, the proportionality factor being approximately $f'(\xi)$. The iteration of this method is for this reason called a linear iteration. (2.02) also shows that the smaller the value of $f'(\xi)$ the faster the convergence will be.

On the other hand, if the slope $f'(x)>1$ in absolute value for all x in an interval about the root, the method

will diverge.

Scarborough [8] states that the method of iteration is especially useful for finding the real roots of an equation given in the form of an infinite series.

Figure 2.1 below demonstrates the procedure of convergence for the method of iteration given the starting value x_0.

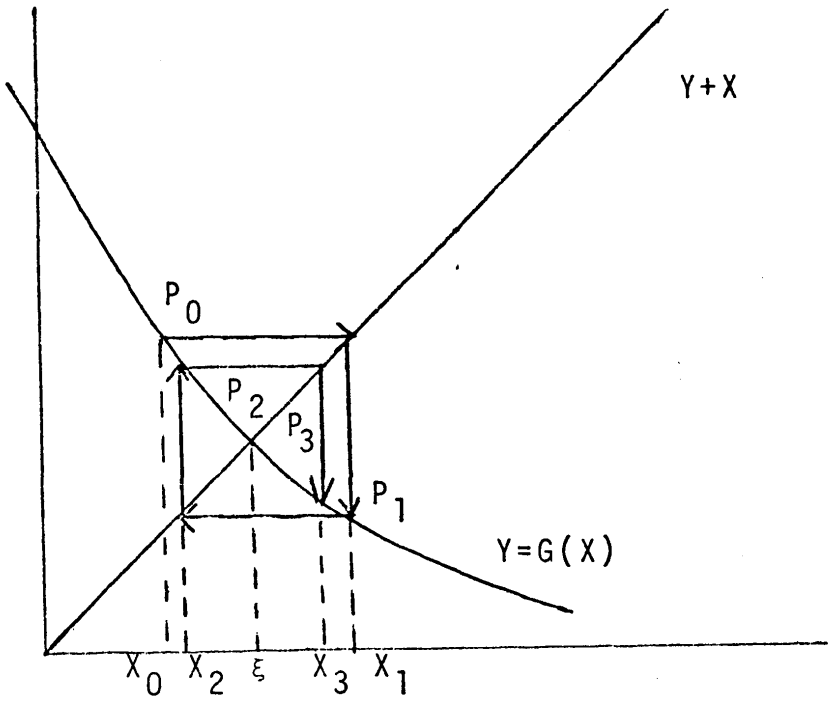


Figure 2.1

AITKEN'S DELTA SQUARED PROCESS

Grove [2] discusses the possibility of using more than one previous x to obtain an improved root of $x = f(x)$. It would seem reasonable that if one were to use more than one of a set of successive approximations in an iteration of the form $x = f(x)$, one should be able to obtain a better next approximation.

Assume that there are three successive approximations x_k, x_{k+1}, x_{k+2} obtained using $x = f(x)$.

$$x_{k+1} = f(x_k)$$

$$x_{k+2} = f(x_{k+1})$$

Let $\bar{x}$ be the required solution. Using the mean value theorem $f(x_k) - f(\bar{x}) = (x_k - \bar{x})\, f'(\xi_1)$ where ξ_1 is between x_k and $\bar{x}$. Also

$$f(x_{k+1}) - f(\bar{x}) = (x_{k+1} - \bar{x}) f'(\xi_2)$$

where ξ_2 is between x_{k+1} and $\bar{x}$. If x_{k+1} and x_k are both near $\bar{x}$, ξ_1 and ξ_2 will close together and $f'(\xi_1)$ and $f'(\xi_2)$ should be near equality. Since $f(x_k) = x_{k+1}$, $f(x_{k+1}) = x_{k+2}$ and $f(\bar{x}) = \bar{x}$ the above equations become

$$x_{k+1} - \bar{x} = (x_k - \bar{x})\, f'(\xi_1) \qquad (2.03)$$

$$x_{k+2} - \bar{x} = (x_{k+1} - \bar{x}) f'(\xi_2) \qquad (2.04)$$

Dividing (2.03) by (2.04) we get

$$\frac{x_{k+1} - \bar{x}}{x_{k+2} - \bar{x}} = \frac{(x_k - \bar{x})f'(\xi_1)}{(x_{k+1} - \bar{x})f'(\xi_2)} = \frac{x_k - \bar{x}}{x_{k+1} - \bar{x}}$$

assuming $f'(\xi_1) = f'(\xi_2)$. Solving for the above for $\bar{x}$, we obtain

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

If as assumed $f'(\xi_1)$ were exactly equal to $f'(\xi_2)$, this would be the required solution $\bar{x}$. Since this is not the case, we have acquired a better approximation of $\bar{x}$. Let us call it x^1_{k+2}. Now we may use this better approximation to continue back through the process until $|x - x_k| < \varepsilon$.

NEWTON-RAPHSON METHOD

Scarborough [8] states that when the derivative of p(x) is a simple expression and easily found, the real roots of p(x) = 0 can be computed rapidly by a process called the Newton-Raphson method. The underlying idea of the method is due to Newton, but the method as now used is due to Raphson.

To derive a formula for computing real roots by this method let "a" denote an approximate value of the desired root, and let h denote the correction which must be applied to "a" to give the exact value of roots, so that x = a + h.

The equation $p(x) = 0$ then becomes $p(a+h) = 0$. Expanding this by Taylor's theorem, we have

$$p(a+h) = p(a) + hp'(a) + h^2/2p''(a+h\theta), \quad 0 \leq \theta \leq 1 .$$

Hence $p(a) + hp'(a) + \frac{h^2}{2} p''(a+\theta h) = 0$. Now if h is relatively small, we may neglect the term containing h^2 and get the simple relation

$$p(a) + hp'(a) = 0$$

from which

$$h_1 = - \frac{p(a)}{p'(a)} . \tag{2.05}$$

The improved value of the root is then

$$a_1 = a + h_1 = a - \frac{p(a)}{p'(a)} .$$

The succeeding approximations are

$$a_2 = a_1 + h_2 = a_1 - \frac{p(a_1)}{p'(a_1)} , \quad a_3 = a_2 - \frac{p(a_2)}{p'(a_2)} ,$$

$$a_n = a_{n-1} - \frac{p(a_{n-1})}{p'(a_{n-1})} . \tag{2.06}$$

Equation (2.06) is the fundamental formula in the Newton-Raphson process. It is evident from the formula that the larger the derivative $p'(x)$ the smaller the correction which must be applied to get the correct value of the root. This means that when the graph is nearly vertical where it crosses

the x-axis the correct value of the root can be found with great rapidity and very little labor. If, on the other hand, the numerical value of the derivative p'(x) should be small in the neighborhood of the root, the values of h given by (2.05) would be large and the computation of the root by this method would be a slow process or might even fail altogether. The Newton-Raphson method should never be used when the graph of p(x) is nearly horizontal where it crosses the x-axis.

The process will evidently fail if p'(x) = 0 in the neighborhood of the root. In such cases the method of false position should be used.

Grove [2] states that Newton-Raphson has the characteristics of second-order convergence for simple roots, but for roots of multiplicity greater than one, the method is first order. He also warns the user of two difficulties which although rare, may occur. The first concerns a change of sign in p"(x) in the neighborhood of a root. If this should happen, the Newton-Raphson method may not converge. The second problem is that the method may converge to a root which is not a desired root.

Hartree [3] gives as a disadvantage the following statement: p(x) and p'(x) must be evaluated at a number of values of x which, though systematic in the sense that each is calculated from the previous one by the same formula such as $x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n)}$ are irregularly spaced and such a set of

numbers is difficult to check adequately. He further states that a great advantage of the method is that a mistake in an intermediate value of x_n does not affect the final result; it is just equivalent to starting a new iteration with the erroneous values of x_n as x_0. This does not eliminate the possibility of a mistake in the last repetition of the iterative process. Tabulation of p(x) at equal intervals of x followed by a process of inverse interpolation is a procedure which provides more, and simpler, checks against occassional mistakes.

Conte [7] formulates the necessary condition of convergence for the Newton-Raphson method as being the following:

$$\left|g'(x)\right| = \frac{\left|p(x)p''(x)\right|}{[p'(x)]^2} < 1 \quad .$$

Also given is a good explanation of the effects of a second-order method on accuracy. In general the number of places of accuracy doubles with each iteration. One cannot, however, expect this in the early iterations due to asymptotic properties. From a purely computational point of view accuracy attained with the Newton-Raphson method depends upon the accuracy to which p(x)/p'(x) can be computed.

METHOD OF FALSE POSITION

Kunz [9] states that the method of false position is the

oldest and most generally applicable method for finding the real roots of polynomials.

Scarborough [8] explains that in this method we find two numbers x_1 and x_2 between which the root lies. These numbers should be as close together as possible. Since the root lies between x_1 and x_2 the graph of $y = p(x)$ must cross the x-axis between $x = x_1$ and $x = x_2$, and y_1 and y_2 must have opposite signs.

Now since any portion of a smooth curve is practically straight for a short distance, it is legitimate to assume that the change in $p(x)$ is proportional to the change in x over a short interval, as in the case of linear interpolation from logarithmic and trigonometric tables. The method of false position is based on the above principle, for it assumes that the graph of $y = p(x)$ is a straight line between the points (x_1, y_1) and (x_2, y_2) where these points are on opposite sides of the x-axis.

To derive a formula for computing the root, consider figure 2.2, which represents a magnified view of that part of the graph between (x_1,y_1) and (x_2,y_2). From the

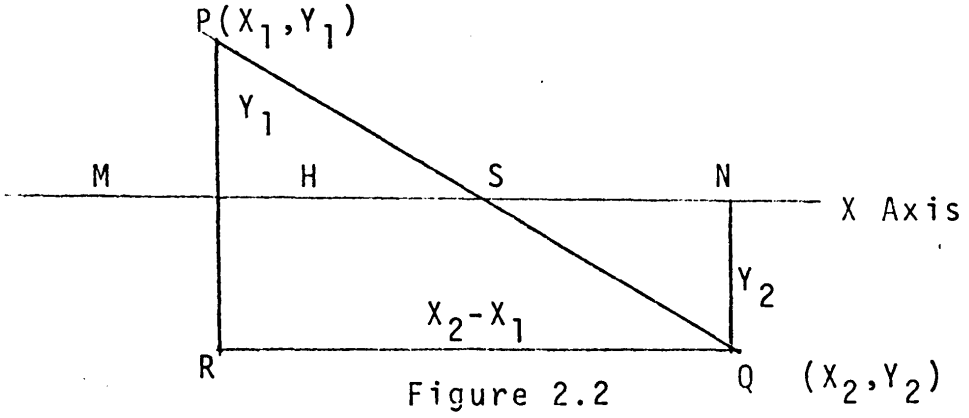


Figure 2.2

similar triangles PMS and PRQ we have

$$\frac{MS}{MP} = \frac{RQ}{RP} \text{ , or } \frac{H}{|y_1|} = \frac{x_2 - x_1}{|y_1| + |y_2|}$$

$$\therefore \quad H = \frac{(x_2 - x_1)|y_1|}{|y_1| + |y_2|} \quad .$$

The value of the desired root, under the assumptions made, is $x = x_1 + MS = x_1 + H$.

Hence

$$x = x_1 + \frac{(x_2 - x_1)|y_1|}{|y_1| + |y_2|} \quad .$$

The value of x is not, however, the true value of the root, because the graph of $y = p(x)$ is not a perfectly straight line between the points P and Q. It is merely a closer approximation to the true root.

Booth [3] notes that convergence may be quite rapid if the initial point is well chosen, but one gains only one decimal place at each iteration.

Kunz [9] states that this rule gives best results when used to improve accuracy of roots once approximated by some other method. It can also be used to help locate a root roughly if [a,b] is not too large.

Hartree [4] states that this method has the same disadvantages and advantages as the Newton method except the False Position method does not require evaluation of $p'(x)$.

SECANT METHOD

Grove [2] states that the secant method may be regarded as a modification of the Newton-Raphson method. In the Newton-Raphson method

$$x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n)} ;$$

replace $p'(x_n)$ by the slope of the secant line between two successive approximations, as

$$m = \frac{p(x_n) - p(x_{n-1})}{x_n - x_{n-1}}$$

and now

$$x_{n+1} = x_n - \frac{p(x_n)}{m} .$$

Note here that simplification of the equation yields the same equation as the rule of False Position. An alternative is to use m as a constant. A choice here might be the secant line between two guesses x_1 and x_2 such that $p(x_1) \cdot p(x_2) < 0$ and $p(x)$ is continuous in the vicinity of the root. Thus

pick

$$m = \frac{p(x_1) - p(x_0)}{x_1 - x_0}$$

and use this in each iteration. Convergence is very slow (of order one).

WEGSTEIN METHOD

Grove [2] suggests that this method induces convergence in some otherwise divergent iterations of the form $x = f(x)$. It has been shown that an iteration of the form $x = f(x)$ will converge to a root if $|f'(x)|<1$ and diverges otherwise. Let us consider the case where $x = f(x)$, $-1<f'(x)<0$ as shown in figure 2.3.

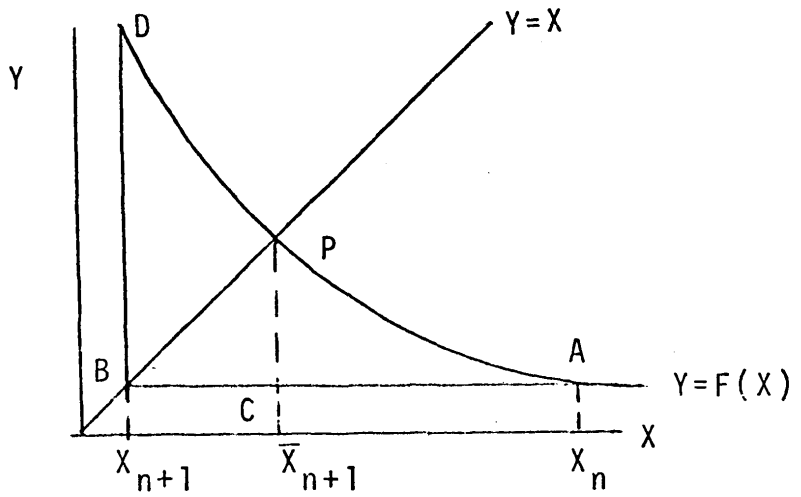


Figure 2.3

In this figure a better approximation for $\bar{x}$ than x_{n+1} would be a value around 1/4 greater. Let $\bar{x}_{n+1}$ be a better approximation than x_{n+1}.

Let q be a fractional part of the distance BA measured to the better approximation from x_{n+1} - that is;

$$\bar{x}_{n+1} = x_{n+1} + q(x_n - x_{n+1})$$

where

$$q = \frac{BC}{BA} ;$$

rearranging, we obtain

$$\bar{x}_{n+1} = qx_n + (1-q)x_{n+1} .$$

Note that BC = CP

Since $q = \frac{BC}{BA}$

then $1 - q = \frac{CA}{BA}$

and $\frac{q}{1-q} = \frac{BC}{CA} = \frac{CP}{CA}$

now CP/CA is the negative of the slope of y = f(x) at some point between A and P.

Let $\frac{q}{1-q} = - a$

where $a = f'(\xi_1)$ and ξ_1 is between A and P. Solving the above for q we obtain

$$q = \frac{a}{a-1}$$

The approximation for "a" that Wegstein uses is

$$a = \frac{f(x_n) - f(\bar{x}_{n-1})}{x_n - x_{n-1}}$$

which is valid for all smooth curves. If one is not particularly interested in information gathered from the value of q, the formula

$$\bar{x}_{n+1} = \frac{x_{n+1}\,\bar{x}_{n-1} - x_n\,\bar{x}_n}{x_n + \bar{x}_{n-1} - x_n - \bar{x}_n}$$

may be used.

The method will converge in more cases than the method of iteration, but may converge to an undesired root.

HALLEY'S METHOD

Grove [2] explains the ideas behind Halley's method. Truncating the Taylor's series expansion of p(x) about the point x_n after 2nd derivatives we obtain

$$p(x) = p(x_n) + p'(x_n)(x-x_n) + \frac{p''(x_n)(x-x_n)^2}{2} \quad .$$

Now we will substitute $x = x_{n+1}$ and assume x_{n+1} is a good approximation to the root so that $p(x_{n+1})$ is nearly zero. In fact, we will call $p(x_{n+1}) = 0$.

$$0 = p(x_n) + p'(x_n)(x_{n+1} - x_n) + \frac{p''(x_n)(x_{n+1} - x_n)^2}{2}$$

Now we will solve for x_{n+1}:

$$(x_{n+1} - x_n)\left[p'(x_n) + \frac{p''(x_n)(x_{n+1} - x_n)}{2}\right] = - p(x_n)$$

$$x_{n+1} - x_n = - \frac{p(x_n)}{p'(x) + \frac{p''(x_n)(x_{n+1} - x_n)}{2}}$$

$$x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n) + \frac{p''(x_n)(x_{n+1} - x_n)}{2}} \qquad (2.7)$$

At this point we recall the Newton-Raphson formulation

$$x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n)}$$

or

$$x_{n+1} - x_n = - \frac{p(x_n)}{p'(x_n)} \qquad (2.8)$$

Replace $x_{n+1} - x_n$ in (2.7) with its equivalent presented in (2.8) so that

$$x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n) - \frac{p''(x_n) \cdot p(x_n)}{2p'(x_n)}}$$

This is Halley's formula.

It appears that for cubic equations this method could be combined with synthetic division to obtain one real root

and the reduced quadratic in a very short time.

MULLER'S METHOD

Conte [7] explains Muller's method as follows. Given an equation $p(x) = 0$ for which we wish to locate a root, we require three starting points. Let us assume the points are (x_i,p_i),(x_{i-1},p_{i-1}), and (x_{i-2},p_{i-2}). Now pass a parabola of the form $y = g(x) = a_2x^2 + a_1x + a_0$ through these points.

The following equations must now be solved to determine the coefficients of the parabola:

$$a_2x_i^2 + a_1x_i + a_0 = p_i$$

$$a_2x_{i-1}^2 + a_1x_{i-1} + a_0 = p_{i-1}$$

$$a_2x_{i-2}^2 + a_1x_{i-2} + a_0 = p_{i-2} \quad .$$

As long as x_i,x_{i-1},x_{i-2} are distinct points, these three equations can easily be solved for the unknown coefficients a_2,a_1,a_0. Once this has been done the quadratic polynomial is completely determined.

The parabola formed intersects the x-axis at two points, say d and d'. Now we must choose one or the other as our next approximation for the root $\bar{x}$. From figure (2.4) the choice of the next approximation is obvious, but of course it is not to the computer. We may evaluate $p(x)$ for d and d' and choose whichever causes $|p(x)|$ to be smaller. Now

we can use the points b, c, and d or d' (whichever was chosen) and iterate the procedure, discarding the "oldest" previous point each time.

Although we have considered this method for real roots, it is usually used to calculate complex roots. An advantage of this method is that no derivatives of the function p(x) need be calculated. For real roots it is not particularly better than interval halving. It does converge faster, but it requires more computations per iteration.

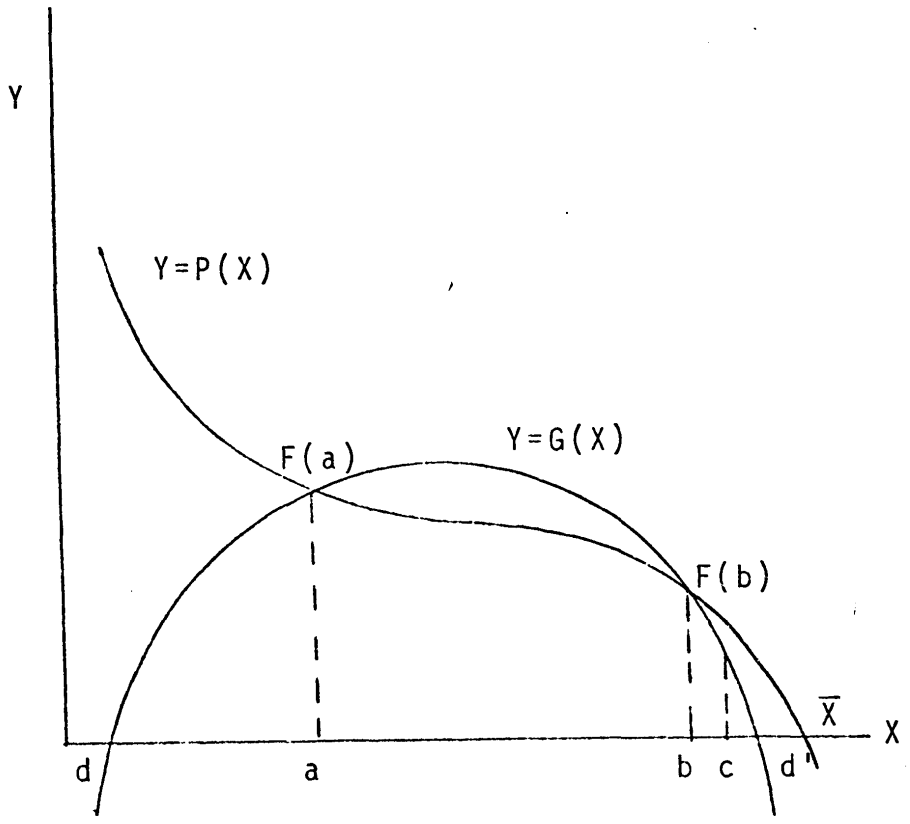


Figure 2.4

LAGRANGE'S METHOD

Todd [10] states that the Lagrange method is an iterative method for determining the zeroes of polynomials. If $f(z) = p_n(z)$ is a polynomial of the nth degree, the Lagrange method is obtained by setting

$$g(z) = z - \frac{nf(z)}{f'(z) \pm \sqrt{(n-1)[(n-1)f'(z)^2 - nf(z)f''(z)]}}$$

In its geometrical interpretation, this method amounts to approximating the polynomial by parabolas between two zeroes. Accordingly, there are two values of g(z), depending on which root is to be approximated. For practical purposes, that sign is used which makes the denominator largest; that is, the root closest to the initial guess is approximated.

The first derivatives at a root x are

$$g'(x)=0,\ g''(x)=0,\ g'''(x) = -\frac{f'''(x)}{f'(x)} - \frac{3}{4}\frac{n-2}{n-1}\left[\frac{f''(x)}{f'(x)}\right]^2$$

so that in general the third derivative is the first different from zero; that is, the method is a third-order iteration procedure.

In comparison with the other methods so far discussed, the Lagrange method has the advantage that it converges faster and that it works also for the complex roots of polynomials with real coefficients, even if one starts out with

a real guess. (The expression under the root in the denominator may become negative). It has, however, the drawback that higher-order derivatives have to be computed. This can be done most conveniently by using synthetic division (Horner scheme); that is if

$$f(z) = \sum_{j=0}^{n} a_{n-j} z^j$$

should be evaluated with its derivatives for $z = x$, then this can be done recursively by computing

$$a_{0,0} = a_0,$$

$$a_{i,0} = a_{i-1,0}\, x + a_i, \quad i=1,2,\cdots n$$

$$a_{0,j} = a_0,$$

$$a_{ij} = a_{i-1,j} x + a_{i,j-1}, \quad i=1,2,\cdots n-j$$

and then

$$f(x) = a_{n,0}, \quad f'(x) = a_{n-1,1}, \quad f''(x) = 2!\, a_{n-2,2},$$

$$f^{(j)}(x) = j!\, a_{n-j,j}, \quad j = 1,2,\cdots n$$

NEWTON AND LAGUERRE

Todd [10] explains that the formulas for the Newton and Laguerre methods can be expressed in terms of the quantities

$s_1 = s_1(z)$, $s_2 = s_2(z)$:

Newton: $$g(z) = z - \frac{1}{s_1(z)}$$

Laguerre: $$g(z) = z - \frac{n}{s_1 \pm \sqrt{(n-1)(ns_2 - s_1^2}}$$

Therefore, for these methods the roots which already have been computed can be eliminated by subtracting the appropriate expressions from $s_1(z)$ and $s_2(z)$, that is, by substituting in the formulas $s_1(z)$ and $s_2(z)$ where

$$s_1(z) = s_1(z) - \sum_{i=1}^{j} \frac{1}{z - x_i}$$

$$s_2(z) = s_2(z) - \sum_{i=1}^{j} \frac{1}{(z - x_i)^2}$$

(j being the number of roots computed already).

This procedure gives in many cases more accurate results than synthetic division. However, it requires appreciably more work for finding all the roots, since one works at all times with the original polynomials.

HORNER'S METHOD

Kunz [9] considers the method of Horner. The Horner

method was devised to reduce pencil work, however, when used with a desk-type calculator it takes a considerable length of time.

The method consists of making a first approximation then transforming the equation (synthetic division) by dividing through by the approximation which will transform the equation into an equation with a root between 0 and 1. A second approximation is made on this interval and the amount .x is added to the first approximation, then the equation is transformed into one with roots between 0 and .1. This procedure continues until the desired degree of accuracy is obtained.

LIN'S METHOD

Grove [2] states that in order to find the complex roots of an equation of the form

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots a_0 = 0 \quad (2.9)$$

we may be able to find real roots, divide out the corresponding factors, and solve the resultant equation. However, if the equation is of even degree there may be no real roots. In such cases we would like to be able to find the quadratic factors of (2.9). The following is an iterative procedure for doing this.

Complex Roots occur in conjugate pairs, say (a+bi) and (a-bi). The associated factors are

$$x - (a+bi) \text{ and } x-(a-bi) .$$

From these we construct the quadratic factor

$$[x-(a+bi)][x-(a-bi)] = (x-a-bi)(x-a+bi)$$

or

$$[(x-a)-bi][(x-a)+bi] = x^2 -2ax + a^2 + b^2 .$$

Now set $x^2 + px + q = x^2 - 2ax + a^2 + b^2$.

Equating coefficients $p = - 2a$

and $q = a^2 + b^2$

$a = -p/2$; $b^2 = q - a^2$; $b = \pm \sqrt{q-a^2}$.

Now we can compute p and q; we can find the roots of the factor $x^2 + px + q$. Dividing $a_0x^n + a_1x^{n-1} + a_2x^{n-2} \cdots a_n$ by $x^2 + px + q$ we obtain

$$\frac{a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-2}x^2 + a_{n-1}x + a_n}{x^2 + px + q}$$

$$= b_0x^{n-2} + b_1x^{n-3} + b_2x^{n-4} + \ldots + R$$

where $R = b_{n-1}\, x^{-1} + b_{n-2}x$

and

$$b_0 = a_0$$

$$b_1 = a_1 - b_0 p$$

$$b_2 = a_2 - b_0 q - b_1 p$$

$$b_3 = a_3 - b_1 q - b_q p$$

$$b_4 = a_4 - b_2 q - b_3 p$$

$$\vdots$$

$$b_k = a_k - b_{k-2}\, q - b_{k-1}\, p;\ k = 2,3,4\cdots n$$

Hildebrand [5] states that the method due to Lin consists of applying the method of successive substitutions in the form

$$p = \frac{a_{n-1} - q\, b_{n-3}}{b_{n-2}}, \quad q = \frac{a_n}{b_{n-2}} .$$

In the absence of preliminary information, the iteration may be started with arbitrarily chosen values of p and q, in the hope that convergence to some root pair (real or complex) will ensue.

BAIRSTOW METHOD

Another iterative method for solving polynomial equations, discovered by Bairstow, differs from the Lin method in that the equations $R(p,q) = 0$ and $S(p,q) = 0$ are solved by the Newton-Raphson iteration for two equations, rather than by

the method of successive substitutions used by Lin, so that it is a second-order process.

Today Bairstow's method is probably more widely used than any other method. Since it is a second-order method, convergence is relatively fast. The major advantage is that it has the capabilities of returning both real and complex roots. Hamming [1] states that it is not infallible; occasionally it takes very long to converge on a quadratic factor or even fails; but, on the average, it seems to be better than any other single method.

Let the polynomial $p(x) = a_0 + a_1 x + a_2 x^2 \cdots a_n x^n$ and let us assume that we have a guess at a quadratic factor $x^2 + px + q$. Initially we can choose $p = q = 0$ which will simplify the first step. Using synthetic division, we divide the polynomial by the quadratic factor to get a quotient and a remainder, e.g.,

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = (x^2 + px + q)(b_n x^{n-2} + b_{n-1} x^{n-3} + \cdots + b_2) + b_1 x + b_0 .$$

The reason for the peculiar subscripts on the b's will become apparent as we go on; it makes notation easier. In a skeleton synthetic form, we have

$$\begin{array}{l|cccccc} 1 \quad p \quad q & a_n & a_{n-1} & a_{n-2} & \cdots & a_2 & a_1 & a_0 \\ & - & - & qb_n & \cdots & qb_4 & qb_3 & qb_2 \\ & - & pb_n & qb_{n-1} & \cdots & pb_3 & pb_2 & - \\ \hline & b_n & b_{n-1} & b_{n-2} & \cdots & b_2 & b_1 & b_0 \end{array} .$$

If the remainder is $b_1 x + b_0$, the algebraic relations between the coefficients are

$$\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} - pb_n \\ b_{n-2} &= a_{n-2} - pb_{n-1} - qb_n \\ &\vdots \\ b_{n-k} &= a_{n-k} - pb_{n-k+1} - qb_{n-k+2} \quad (k=2,3\cdots n-1) \\ b_0 &= a_0 - qb_2 \ . \end{aligned} \tag{2.10}$$

We would have the desired quadratic factors if, and only if, the remainder were identically zero; that is, if

$$b_1 = b_0 = 0 \ .$$

Let us consider these coefficients as functions of p and q.

$$\begin{aligned} b_1 &= b_1(p,q) \\ b_0 &= b_0(p,q) \end{aligned}$$

We now use the two dimensional analog of Newton's method and expand b_1, b_0 in a Taylor's series about the present guess (p,q). Writing p* and q* as the desired solution, we have

$$\begin{aligned} b_1(p^*,q^*) = 0 &= b_1(p,q) + \frac{\partial b_1}{\partial p}\Delta p + \frac{\partial b_1}{\partial q}\Delta q + \cdots \\ b_0(p^*,q^*) = 0 &= b_0(p,q) + \frac{\partial b_0}{\partial p}\Delta p + \frac{\partial b_0}{\partial q}\Delta q + \cdots \end{aligned} \tag{2.11}$$

where $\Delta p = p^* - p$

$\Delta q = q^* - q$

are the errors to be corrected for our next guess. Neglecting all but the linear terms in (2.11) we have a pair of linear equations for the changes to be made in p and q.

The problem is how to find the partial derivatives which are the coefficients of the unknowns Δp and Δq. We could make a small change in p and note the change in b_1 and b_0 and do the same using a small change in q. We prefer, instead, to find them in a more analytical fashion. We differentiate (2.10) with respect to p.

$$\frac{\partial b_n}{\partial p} = 0$$

$$\vdots$$

$$\frac{\partial b_{n-1}}{\partial p} = - b_n - \frac{p \partial p_n}{\partial p}$$

$$\vdots$$

$$\frac{\partial b_{n-2}}{\partial p} = - b_{n-1} - p \frac{\partial b_{n-1}}{\partial p} - q \frac{\partial b_n}{\partial p}$$

.....

$$\frac{\partial b_{n-k}}{\partial p} = - b_{n-k+1} - p \frac{\partial b_{n-k+1}}{\partial p} - q \frac{\partial b_{n-k+2}}{\partial p}$$

.....

$$\frac{\partial b_0}{\partial p} = - q \frac{\partial b_2}{\partial p}$$

If we now write $\frac{\partial b_k}{\partial p} = - C^*_k$

then we have

$$C^*_n = 0$$

$$C^*_{n-1} = b_n - pC^*_n$$

$$C^*_{n-2} = b_{n-1} - pC^*_{n-1} - qC^*_n$$

$$C^*_{n-k} = b_{n-k+1} - p\,C^*_{n-k+1} - qC^*_{n-k+2}$$

.....

$$C^*_0 = \qquad\qquad -qC^*_2$$

These equations are of the same form as (2.10) provided that we note that $C^*_n = 0$ means the following: C^*_{n-k} is to be identified with b_{n-k+1}, b_k is to be identified with a_{k-1}. Also, the last equation is not quite correct.

These observations suggest repeating the process of synthetic division, using the same quadratic factor x^2+px+q on the b's (instead of the a's) to obtain coefficient of C_k. This we now do:

$$\begin{array}{l|cccccccc}
1\ p\ q & b_n & b_{n-1} & b_{n-2} & b_{n-3} & \cdots & b_2 & b_1 & b_0 \\
 & & qC_n & qC_{n-1} & & \cdots & qC_4 & qC_3 & qC_2 \\
 & pC_n & pC_{n-1} & & & \cdots & pC_3 & pC_2 & \\
\hline
 & C_n & C_{n-1} & C_{n-2} & & \cdots & C_2 & C_1 & C_0
\end{array}$$

Thus we get

$$
\begin{aligned}
C_n &= b_n \\
C_{n-1} &= b_{n-1} - pC_n \\
C_{n-2} &= b_{n-2} - p\,C_{n-2} - qC_n \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
C_{n-k} &= b_{n-k} - pC_{n-k+1} - q\,C_{n-k+2} \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
C_1 &= b_1 - pC_2 - qC_3 \\
C_0 &= b_0 \qquad - qC_2 \quad .
\end{aligned}
\tag{2.13}
$$

The partial derivatives for which we are looking in (2.11) are

$$\frac{\partial b_1}{\partial p} = - C^*_1; \quad \frac{\partial b_0}{\partial p} = - C^*_0$$

Comparing (2.12) and (2.13) we see that

$$C^*_{k-1} = C_k \quad (k = n, n-1, \cdots 3, 2)$$

$$C^*_0 = C_1 - b_1 + pC_2 \quad .$$

Hence

$$\frac{\partial b_1}{\partial p} = - C_2; \quad \frac{\partial b_0}{\partial p} = - (C_1 - b_1 + pC_2).$$

We now examine the process for the partial derivative with respect to q. Again differentiate

$$\frac{\partial b_{n-1}}{\partial q} = - p \frac{\partial b_n}{\partial q} = 0$$

$$\frac{\partial b_{n-2}}{\partial q} = - b_n - p \frac{\partial b_{n-1}}{\partial q} - q \frac{\partial b_n}{\partial q}$$

.....

$$\frac{\partial b_{n-k}}{\partial q} = - b_{n-k+2} - p \frac{\partial b_{n-k+2}}{\partial q} - q \frac{\partial b_n}{\partial}$$

.....

$$\frac{\partial b_0}{\partial q} = - b_2 \qquad - q \frac{\partial b_2}{\partial q} .$$

We now set $\frac{\partial b_k}{\partial q} = - C_k^{**}$

to get $C^{**}_{n-1} = 0$

$$C^{**}_{n-2} = b_n - p C^{**}_{n-1} - qC^{**}_n$$

.....

$$C^{**}_{n-k} = b_{n-k+2} - pC^{**}_{n-k+1} - q C^{**}_n$$

.....

$$C^{**}_0 = b_2 \qquad - qC^{**}_2$$

Since $C^{**}_n = C^{**}_{n-1} = 0$ we see that we need to identify

$$C^{**}_{k-2} = C_k; \quad (k = n, n-1, \cdots, 3)$$

$$C^{**}_{0} = C_2 - pC_3$$

if we are to compare (2.14) and (2.13). The partial derivatives that we want for (2.11) are

$$\frac{\partial b_1}{\partial q} = - C^{**}_{1} = - C_3; \quad \frac{\partial b_0}{\partial q} = - C^{**}_{0} = - (C_2 - pC_3) .$$

Thus we have

$$b_1(P,q) = C_2\Delta p + C_3\Delta q$$

$$b_0(p,q) = (C_1 - b_1 - pC_2)\Delta p + (C_2 - pC_3)\Delta q.$$

The solution of these two equations produces the amounts to change our guess of the quadratic factor $x^2 + px + q$. The convergence, when it works, is quadratic; that is, the errors when small, are approximately squared each step. Now we proceed to factor out the quadratic factor and use the equation as a new polynomial to be examined by the same process.

Ralston [11] states that when Bairstow's method converges it has the characteristic rapid convergence of the Newton-Raphson method.

Conte [7] notes that the major deficiency lies in the

fact that it is difficult to select the initial approximation (α_0,β_0) properly so as to assure convergence.

LEMUR-SCHUR

Ralston [11] notes that this method may be used to determine whether any zero of a polynomial lies within the unit circle; this idea will be used as the basis of a method to find the roots of $p(x) = 0$.

Define

$$p^*(z) = z^n \bar{p}(\bar{z}^{-1}) = \bar{a}_n + \bar{a}_{n-1} z + \cdots + \bar{a} z^n$$

where bars denote conjugates. Now define

$$T[p(z)] = \bar{a}_0 p(z) - a_n p^*(z)$$

so that, in particular $T[p(0)] = \bar{a}_0 a_0 - a_n \bar{a}_n$

$$= |a_0|^2 - |a_n|^2$$

is real. Note also that $T[p(z)]$ has no term in z^n so that if we define

$$T^j[p(z)] = T\{T^{j-1}[p(z)]\}$$

we get a sequence of polynomials of decreasing degree. Let k be the smallest integer for which $T^k[p(0)] = 0$. The basic idea that we can use is the following: Suppose $p(0) \neq 0$. If for some h such that $0<h<k$, $T^h[p(0)]<0$, then $p(z)$ has at least one zero inside the unit circle. If instead $T^i[p(0)]>0$

for $1 \leq i < k$ and $T^{k-1}[p(z)]$ is a constant, then no zero of $p(z)$ lies inside the unit circle.

The proof of this idea requires various elementary results from complex-variable theory. To use this theorem in determining whether or not $p(z)$ has a zero inside the unit circle, we proceed as follows:

1) Is $p(0) = 0$? If so we have a zero $z = 0$; if not, do step 2.

2) Calculate $T[p(z)]$. Is $T[p(0)] < 0$? If so, there is a root inside the unit circle; if not, go to step 3.

3) Calculate $T^j[p(z)]$, $j = 1,2,\cdots$ until $T^j[p(0)] < 0$, $j<k$ or $T^k[p(0)] = 0$. If the former occurs, there is a root inside the unit circle. If the latter occurs and if $T^{k-1}[p(z)]$ is a constant, then there is no root inside the unit circle. Note that the theorem does not cover one possibility. If $T^k[p(0)] = 0$ but $T^{k-1}[p(z)]$ is not a constant, the theorem tells us nothing. We shall close this loop hole a little later.

To apply this theorem to find the roots of

$$p(z) = a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0 = 0$$

we note first that if $p(z)$ has a zero inside the circle $|z| = q$, then $g(z) = p(qz)$ has a zero inside the unit circle. More generally, if $p(z)$ has a zero inside the circle $|z-c| = q$ $g(z) = p(qz+c)$ has a zero inside the unit circle. Thus we

proceed as follows, using the theorem at each step:

1) Does $p(z)$ have a zero inside the unit circle? If not consider $g(z) = p(2z)$ and ask whether $g(z)$ has a zero inside the unit circle. If not, consider $p(2^2 z)$. Continuing in this way, sooner or later we find an annulus

$$R = 2^j \leq |z| < 2^{j+1} = 2R$$

such that $p(z)$ contains a zero in this annulus and none inside the circle $|z| = R$. [If $p(z)$ does have a zero inside the unit circle, we halve the radius until we find a circle inside which there is no zero. Again we get an inequality for the annulus that contains the zero].

2) This annulus can be completely covered by eight overlapping circles each of radius $4R/5$ with centers at

$$\frac{3R}{2\cos(\pi/8)} e^{2\pi i k/8} \quad k = 0,1,\cdots 7, \quad i = \sqrt{-1}.$$

Testing each of these circles in turn using the theorem, we shall find at least one containing a root. If the coefficients of the polynomial are real, there must be a root in a circle for $k = 0, 1, 2, 3, 4$.

3) Calling the center of this circle C, and starting with the radius $4/5\ R$, we proceed as in step 1 except that now we have the radius at each stage. Finally we find an annulus

$$R_1 = (4R/5) \times 2^{-j_1} \leq |z-c_1| < (4R/5) \times 2^{-(j_1-1)} = 2R_1$$

for some positive integer j_1 which contains a zero of $p(z)$. As in step 2 we cover this annulus with eight circles and repeat steps 2 and 3 as long as desired.

The loophole in the theorem will occur with probability zero for a random choice of coefficients but nevertheless, it occurs when $a_0 = a_n$ and for certain simple cases of integral coefficients. If for example in performing step 1 for a radius R, $T^k[g(0)] = 0$, but $T^{k-1}[g(z)]$ is not a constant, the simplest thing to do is to choose a new radius βR where $1/2<\beta<1$, say $\beta = 3/4$, and continue with this value of the radius, choosing as the next radius $2\beta R$. If this case occurs in step 3, then we use a value $1<\beta<2$ and continue in an obvious fashion. This procedure converges inevitably to a root of $p(x)$.

One can make the following points concerning the Lemur-Schur method:

1) The speed of the convergence is in no way affected by the multiplicity of the roots or by whether the roots are clustered in any way.

2) At any stage we may switch over to a more rapidly convergent method. Whether this more rapidly convergent method does indeed converge will depend on how close we are to the root. If it does not converge, we may switch back to the Lemur-Schur method.

3) Having found one root, we may remove it from $p(x)$ by synthetic division and proceed to find the others, remembering to make 2R the starting radius for the next root.

BERNOULLI'S METHOD

Ralston [11] suggests that one consider the difference equation $a_n u_k + a_{n-1} u_{k-1} + \cdots + a_0 u_{k-n} = 0$, (2.15) where the coefficients a_i, $i = 0, \cdots, n$ are those of $p(x)$. If the roots α_i of $p(x)$ are distinct then the solution of this equation is given by

$$u_k = \sum_{i=1}^{n} c_i \alpha_i^k \tag{2.16}$$

where the c_i's depend on the initial conditions used to solve (2.15). If the roots are ordered in magnitude as in $p_1 > p_2 > \cdots > p_n$, then by rewriting (2.16)

$$u_k = c_1 \alpha_1^k [1 + \sum_{i=2}^{n} \frac{c_i}{c_1} (\frac{\alpha_i}{\alpha_1})^k] \text{ we have, if } c_1 \neq 0$$

$$\lim_{k \to \infty} \frac{u_k}{u_{k-1}} = \alpha_1 \tag{2.17}$$

The essence of Bernoulli's method is to use (2.15) to compute successive values of u_k and then to compute the ratio of successive values of u_k until these ratios converge to α_1.

For this method to work at all, it is necessary that $c_1 \neq 0$. The c_i's depend, as we said, on the n initial

conditions required by (2.15). If we generate these initial values using the equation

$$a_n u_m + a_{n-1} u_{m-1} + \cdots + a_{n-m+1} u_1 + m a_{n-m} = 0 \quad m = 1, \cdots, n$$

then it can be shown that all c_i's are unity and thus

$$u_k = \sum_{i=1}^{n} \alpha_i^k . \tag{2.18}$$

Therefore (2.17) always holds for this choice of initial conditions.

The above was predicted on the assumption that α_1, the root of largest magnitude, is real and distinct. Nevertheless, the method also holds if α_1 is multiple but real. When the root of largest magnitude is complex or when there is some combination of real and complex roots or largest magnitude, (2.17) no longer holds. The number of possible special cases is, therefore, very large. Each such special case can be taken care of by a suitable modification of (2.17). For example, if there is a single pair of complex conjugate roots of largest magnitude, then writing

$$\alpha_1 = \beta_1 e^{i\phi_1}; \quad \alpha_2 = \beta_1 e^{-i\phi_1}$$

we may write (2.18) as

$$u_k = 2\beta_1^k \cos k\phi_1 \left(1 + \sum_{i=3}^{n} \frac{\alpha_i^k}{\alpha_1^k + \alpha_2^k}\right) . \qquad (2.19)$$

Using (2.19)

$$\beta_1^2 = \lim_{k\to\infty} \frac{u_k^2 - u_{k+1}\, u_{k-1}}{u_{k-1}^2 - u_k u_{k-2}}$$

and

$$2\beta_1 \cos \phi_1 = \lim_{k\to\infty} \frac{u_k u_{k-1} - u_{k+1}\, u_{k-2}}{u_{k-1}^2 - u_k\, u_{k-2}} .$$

Other special cases such as repeated complex roots and combinations of real and complex roots of equal magnitude can also be handled separately but especially for automatic computation; it is extremely tedious to have to provide for all these cases.

Moreover, if α_2 has nearly the same magnitude as α_1, the convergence of the process is extremely slow. Thus as a general purpose method, Bernoulli's method is inferior to the Lehmer-Schur and root-squaring methods.

When the root of largest or smallest magnitude is the only one that is desired and is distinct, Bernoulli's method can be very useful. When a root has been found and removed from p(x) by synthetic division, Bernoulli's method can then be used to find an approximation to the root of next greatest magnitude.

With slowly convergent methods like the Lehmer-Schur or Bernoulli methods, it is desirable to use them only to get a good approximation to a root.

Hildebrand [5] states that the calculation is remarkably simple (and readily mechanized) when the dominant root is real and unequaled in absolute value and is not unduly complicated otherwise.

GRAEFFE'S ROOT-SQUARING METHOD

Ralston [11] suggests that the essence of Graeffe's method is to replace p(x) by an equation, still of degree n, whose roots are the squares of the roots of p(x). By iterating this procedure, roots of p(x) which are unequal in magnitude become more widely separated in magnitude. By separating the roots sufficiently we can, as we shall see, calculate the roots directly from the coefficients. When there are roots of equal magnitude, this process runs into difficulties, but these can be overcome.

Let the roots of p(x) be α_i, i = 1,$\cdots$,n. We assume in the remainder of the section that $a_n = 1$. Then, writing $p_0(z)$ for p(z), we have

$$p_0(z) = (z-\alpha_1)(z-\alpha_2)\cdots(z-\alpha_n).$$

Using this we may write

$$p_1(w) = (-1)^n\, p_0(z)p_0(-z) = (w-\alpha_1^2)(w-\alpha_2^2)(w-\alpha_n^2),\ w=z^2$$

so that the zeroes of $p_1(w)$ are the squares of those of $p_0(z)$. Therefore, the sequence

$$p_{r+1}(w) = (-1)^n\, p_r(z)p_r(-z),\ r = 0,1,\cdots$$

is such that the zeroes of each polynomial are the squares of the zeroes of the previous polynomial. If we denote the coefficient of $p_r(z)$ by $a_j^{(r)}$, $j = 0,\cdots\cdots,\ n$, then we have

$$a_j^{(r+1)} = (-1)^{n-j}\ \{[a_j^{(r)}]^2 + 2 \sum_{k=1}^{\min(n-j,j)} (-1)^k a_{j-k}^{(r)}\ a_{j+k}^{(r)}\ .$$

To use the sequence of polynomials $\{p_r(z)\}$, we need the well known relationship between the coefficients of a polynomial and its zeroes. This relationship is expressed by the equation

$$a_j^{(r)} = (-1)^{n-j}\ S_{n-j}(\alpha_1^2, \alpha^{2^r}, \cdots, \alpha_n^{2^r})\ \ j = 0,1,\cdots\cdots,\ n-1$$

where $S_k(x_1,\cdots,x_n)$ is the kth symmetric function of $x_1,\cdots,x_n$. This function is defined by the equation

$$S_k(x_1,\cdots,x_n) = \sum_{1^c}^{n} x_{r_1}\, x_{r_2} \cdots x_{r_k}.$$

where the notation $\sum_{1^c}^{n}$ denotes that the sum is over all

combinations of the digits 1 to n in the subscripts. Thus,

for example,

$$a_{n-1}^{(r)} = - S_1(\alpha_1^{2^r}, \cdots, \alpha_n^{2^r}) = - \sum_{k=1}^{n} \alpha_k^{2^r} \quad . \qquad (2.20)$$

Let

$$\alpha_k = q_k e^{i\phi_k} \quad k = 1, \cdots, n.$$

Suppose first that all the roots are distinct in magnitude and ordered so that

$$q_1 > q_2 > \cdots > q_n \quad . \qquad (2.21)$$

We write (2.20) as $a_{n-1}^{(r)} = - \alpha_1^{2^r} \left[1 + \sum_{k=2}^{n} (\frac{\alpha_k}{\alpha_1})^2\right]$.

Then using (2.21) $\lim_{r\to\infty} |-a_{n-1}^{(r)}|^{1/2^r} = |\alpha_1|$.

Therefore, for sufficiently large r

$$q_1 \approx |-a_{n-1}^{(r)}|^{1/2^r} \quad . \qquad (2.22)$$

Similarly, we have

$$a_{n-2}^{(r)} = \sum_{1^c}^{n} \alpha_{r_1}^{2^r} \alpha_{r_2}^{2^r} = \alpha_1^{2^r} \alpha_2^{2^r} \left[1 + \sum_{1^c}^{n} \left(\frac{\alpha_{r_1}\alpha_{r_2}}{\alpha_1\alpha_2}\right)^{2^r}\right]; \quad (r_1, r_2) \neq (1,2)$$

and, therefore, for sufficiently large r

$$q_2 \simeq \frac{1}{q_1} \left|a_{n-2}^{(r)}\right|^{1/2^r} \simeq \left|\frac{a_{n-2}^{(r)}}{a_{n-1}^{(r)}}\right|^{1/2^4} \tag{2.23}$$

Continuing in this way we have in general

$$q_k \simeq \left|\frac{a_{n-k}^{(r)}}{a_{n-k+1}^{(r)}}\right|^{1/2^r} \qquad k = 3, \cdots, n \tag{2.24}$$

In practice "sufficiently large r" means only that we must continue the root-squaring process until the approximations to the magnitudes have stabilized to the number of decimal places that we desire.

Once the roots are separated, and their magnitudes obtained, determining the sign is easily accomplished by inserting the magnitude into p(x).

The difficulties in using this procedure arise when some of the roots have equal magnitudes. These difficulties are of two kinds: (1) The relations (2.22), (2.23), (2.24) are no longer correct in general. Therefore, determining the magnitude of the roots is more difficult. (2) Since some roots

may be complex, it is no longer simple to determine the root given the magnitude.

Further transformations may be made to eliminate these problems, but as a result the procedure is greatly complicated.

Large coefficients may cause trouble, particularly if a large number of root squarings are required. Also how do we know when we have performed a sufficient number of root squarings to separate adequately the roots of different magnitudes? Although a more complicated program, the root-squaring method appears to be more efficient than the Lemur-Schur method. A comparison of the speeds is quite difficult, because the Lemur-Schur method would be coupled with a more rapidly convergent method which would be used when a good approximation to a root has been found.

Hildebrand [5] explains that the Graeffe method possesses the theoretical advantage that the iteration leads to all zeroes of $p(x)$ at the same time, and there is no question of ultimate convergence if appropriate attention is paid to round-off error. However, it is often rather laborious, and the extraction of algebraic roots of high order, which is involved in the process, is conveniently effected in machine calculation only by an iterative process.

A serious disadvantage follows from the fact that a gross error committed at any stage of the calculation invalidates all subsequent calculations, whereas the other iterative methods considered would suffer only a reduction

in the rate of convergence.

Rather than use this method for the complete determination of the roots, it is often convenient merely to iterate sufficiently to obtain crude approximations, when such approximations are not easily obtained by other methods, and then to improve these approximations by simpler or more rapidly convergent methods.

This in effect concludes the detailed look at the review of literature on polynomial root-finding. As we mentioned before, hundreds of books have been written on the matter, and almost every numerical analysis book in existence today has at least one chapter on the subject. For further information, one may refer to the bibliography of this paper.

DISCUSSION

The importance of the field under consideration cannot be overemphasized. On many occasions, scientists and mathematicians encounter polynomials of which they desire the roots. Many methods have been derived to aid them in their endeavors, but due to the number of problems existing in the field, no one method has proved adequate for all occasions.

It is the purpose of this study to delve into some of these methods in order to compare and analyse them. The works of many renowned authors as presented in the previous section have provided factual information which has aided in the study of these methods.

In order to effectively discuss the polynomial root-finding problem, it is necessary to know something about polynomials in general. Several basic algebraic theorems allow us invaluable information on the number, kind, and location of real and complex roots. The following theorems provide us with the above mentioned information: (1) if $p(a)\cdot p(b)<0$, there exists at least one real root, and in fact there are an odd number of real roots in the interval $[a,b]$; (2) if in the synthetic division of $p(x)$ by $x-h$ all the coefficients of the reduced polynomial are positive, h is an upper bound to the real roots, also if in the synthetic division of $p(-x)$ by $x-h$ all the new coefficients are positive then $-h$ is a lower limit to the real roots, and (3) the number of positive real roots is equal to the number of

changes in sign or less than that number by a positive even integer. Likewise, the number of changes in sign of p(-x) indicates the number of possible negative real roots. In particular if p(x) or p(-x) possesses no change in the signs of its coefficients, there are no real roots in that form of p(x). If exactly one change is incurred only one real root exists in that form of p(x). These theorems have provided information to previous researchers and are useful tools in today's investigations.

In researching this particular field several methods were chosen to be compared. The primary concern of the first part of this study was related to methods instrumental in obtaining a single real root. Of the methods encountered upon reviewing the literature, the following were chosen for the comparison: (1) the Method of False Position, (2) the Secant Method, (3) the Bisection Method, (4) the Wegstein Method, (5) the Method of Iteration, (6) the Aitken-Delta Squared Method, and the (7) Newton-Raphson Method. The reason for choosing only single root methods is that once one real root is acquired, the original polynomial may be reduced by synthetic division leaving a polynomial of degree n-1 to which one can again apply a single root method to reduce the polynomial further. Cautions, restrictions, and suggestions on doing this type of procedure will be discussed later.

The above methods were divided into two separate groups. As one might notice, the methods of Iteration, Aitken Delta Squared, Newton-Raphson and Wegstein all require one starting

value while the remaining methods require two starting values. Thus the groups were selected based on the number of starting values.

Let us consider the group consisting of the methods of Bisection, False Position, and Secant. A program was written combining all of these methods so they might be executed in one run through the computer rather than several single runs. Since these methods do require two starting values, a systematic method for providing these starting values had to be implemented. An upper and lower bound to the real roots of the equation under consideration were ascertained. One starting value was fixed at the upper bound and the other starting value was initialized at the lower bound. Once these two values were used in the computation involved in all methods of that group, the second starting value was incremented. This procedure provided new starting values at each step of the comparison and allowed for a wide range of situations. The comparing of the methods continued until the second starting value approached the first value (upper bound). The type of information obtained from this comparison included the following: (1) the number of iterations required to converge to a root, providing convergence occurred, (2) the computed root versus the true root, (3) the actual execution time involved in converging to a root. The last piece of information is interesting in that it was obtained through the use of the special subroutine which enables the user to utilize the IBM System/360 internal timer capable of expressing

increments of time in one-hundredths of a second. Thus, the actual execution time of each method was obtained by calling this subroutine at the beginning and end of each method.

The second group consisting of the methods of Iteration, Newton-Raphson, Wegstein, and Aitken Delta Squared was compared in much the same manner as was the previous group. There were, however, two main differences between the two groups. First, as mentioned before, they had different requirements on the number of starting values; secondly, three of the four members of the second group required iteration in the form $x = f(x)$. To facilitate this requirement, several forms of $x = f(x)$ were used. After trying several forms of $x = f(x)$, the method of iteration was excluded from the comparison due to the fact that it was very critical of the form of $x = f(x)$ and often times converged in only a few instances. To compare the remaining methods, the lower bound to the real roots was used as the first starting value and after each group comparison, was incremented, thus allowing many different starting values. Exactly the same information was gathered on these methods as was described above for the first group.

A special subroutine was written to form a polynomial from a number of real roots. Since some large degree equations were desired, it was decided to use double precision and integer roots. These two endeavors aided in controlling the round off error which might be present in the coefficients of high degree polynomials. Eighteen different polynomials varying in degree from four to fifteen were used in the above

comparisons.

Contained in the following pages are examples of the results obtained in comparing the single root methods. One should note the different activities that occur pertaining to the starting value(s) involved.

The letters NC mean that the method failed to converge to a root. Also, the increments of time as shown on the examples are measured in seconds, thus .01 would be one-hundredth of a second. The expression .10E-05 means .000001. The signed constant following the E indicates the number of places and direction to move the decimal point to obtain the actual figure.

The above methods primarily applied to the removal of real roots from p(x). Should a need for finding the complex roots exist, a special method should be used. Although some of the methods above coupled with complex arithmetic could do the job, not all computers have complex arithmetic available and some languages have no provisions for this. Of prime concern in finding complex roots are the methods of Lin and Bairstow. There is much similarity in these methods in that they both factor from the polynomial a second degree polynomial of the form $x^2 + px + q$ and they both require starting values for p and q. Several polynomials with complex roots were formed and the methods of Lin and Bairstow compared. The results of this comparison will be presented in the next section.

$$p(x) = x^{10} - 13x^9 - 98x^8 + 1734x^7 + 825x^6$$
$$-71565x^5 + 118808x^4 + 927316x^3 - 2175856x^2$$
$$-2671872x + 6773760$$

ROOTS ARE: 8, 9, 7, 5, 3, 2, -2, -4, -7, -8

Starting Values	Method	Number of Iterations	Computed Root	Error	Time To Converge
-7.95,10.0	Bisection	27	-2.000000	0.0E-05	.16
-7.95,10.0	False Position	NC	NC	NC	NC
-7.95,10.0	Secant	16	-7.000000	NC	.33
-7.35,10.0	Bisection	27	-2.000000	0.0E-05	.15
-7.35,10.0	False Position	18	-7.000004	0.6E-05	.53
-7.35,10.0	Secant	28	-3.999999	0.1E-05	.55
-2.25,10.0	Bisection	26	-2.000000	0.0E-05	.15
-2.25,10.0	False Position	8	-2.000000	0.0E-05	.21
-2.25,10.0	Secant	NC	NC	NC	NC

$$p(x) = x^6 - 15x^5 + 49x^4 + 195x^3 - 1166x^2 + 720x + 2016$$

ROOTS ARE: 7, 6, 4, 3, -1, -4

Starting Values	Method	Number Of Iterations	Computed Root	Error	Time To Converge
3.84,8.0	Bisection	25	3.999999	0.1E-05	.11
3.84,8.0	False Position	44	3.999999	0.1E-05	.80
3.84,8.0	Secant	6	4.000000	0.0E-05	.08
-1.25,8.0	Bisection	26	3.999999	0.1E-05	.10
-1.25,8.0	False Position	10	-1.000000	0.0E-05	.18
-1.25,8.0	Secant	8	-1.000000	0.0E-05	.11
-2.15,8.0	Bisection	26	- .999999	0.1E-05	.11
-2.15,8.0	False Position	54	3.999999	0.1E-05	.98
-2.15,8.0	Secant	12	3.000000	0.0E-05	.16

$$p(x) = x^{10} - 23x^9 + 152x^8 + 130x^7 - 4627x^6$$
$$+ 9961x^5 + 32626x^4 - 117780x^3 - 12024x^2$$
$$+ 309312x - 217728$$

ROOTS ARE: 9, 7, 6, 4, 3, 2, 1, -2, -3, -4

Starting Values	Method	Number Of Iterations	Computed Root	Error	Time To Converge
.999,10.0	Bisection	26	8.999999	0.1E-05	.16
.999,10.0	False Position	1	1.000000	0.0E-05	.01
.999,10.0	Secant	1	1.000000	0.0E-05	.01
8.34,10.0	Bisection	23	9.000000	0.0E-05	.13
8.34,10.0	False Position	54	9.000000	0.0E-05	1.51
8.34,10.0	Secant	21	5.999999	0.1E-05	.41
8.94,10.0	Bisection	23	9.000000	0.0E-05	.13
8.94,10.0	False Position	52	9.000002	0.8E-05	1.48
8.94,10.0	Secant	14	9.000000	0.0E-00	28

$$p(x) = x^5 - 12x^4 - 3x^3 + 358x^2 - 264x - 2880$$

ROOTS ARE: 8, 6, 5, -3, -4

Starting Value	Method	Number Of Iterations	Computed Root	Error	Time To Converge
3.50	Newton	6	4.999999	0.1E-05	.03
3.50	Aitken	14	5.000000	0.0E-00	.10
3.50	Wegstein	18	5.000001	0.1E-05	.10
2.00	Newton	6	4.999999	0.1E-05	.03
2.00	Aitken	19	8.000000	0.0E-05	.15
2.00	Wegstein	14	5.000003	0.3E-05	.08
1.25	Newton	6	6.000000	0.0E-00	.01
1.25	Aitken	NC	NC	NC	NC
1.25	Wegstein	NC	NC	NC	NC

$$p(x) = x^8 + 17x^7 + 44x^6 - 462x^5 - 1631x^4 + 3493x^3$$
$$+ 10226x^2 - 3048x - 8640 = 0$$

ROOTS ARE: 4, 3, 1, -1, -2, -5, -8, -9 → 784 records

Starting Value	Method	Number of Iterations	Computed Root	Error	Time To Converge
-3.95	Newton	NC	NC	NC	NC
-3.95	Aitken	6	-5.000000	0.0E-05	.05
-3.95	Wegstein	NC	NC	NC	NC
-6.80	Newton	NC	NC	NC	NC
-6.80	Aitken	NC	NC	NC	NC
-6.80	Wegstein	9	4.000000	0.0E-05	.06
-7.70	Newton	4	-8.000000	0.0E-05	.03
-7.70	Aitken	5	-5.000000	0.0E-05	.05
-7.70	Wegstein	5	-5.000000	0.0E-05	.03

$$p(x) = x^6 - 15x^5 + 49x^4 + 195x^3 - 1166x^2 + 720x + 2016$$

ROOTS ARE: 7, 6, 4, 3, -1, -4

Starting Value	Method	Number Of Iterations	Computed Root	Error	Time To Converge
.05	Newton	8	-4.000000	0.0E-05	.05
.05	Aitken	22	2.999996	0.4E-05	.20
.05	Wegstein	8	5.999992	0.2E-05	.05
.80	Newton	6	3.999999	0.1E-05	.03
.80	Aitken	18	2.999991	0.9E-05	.16
.80	Wegstein	NC	NC	NC	NC
1.55	Newton	6	3.000000	0.0E-05	.03
1.55	Aitken	NC	NC	NC	NC
1.55	Wegstein	7	2.999973	0.27E-05	.03

Upon examining the results of the above mentioned comparisons it was decided to take the "best" method from each group and form a combined method which we shall call the Bisnewbar method. This combined method consisted basically of the following four parts: (1) information gathered from certain algebraic theorems on the number, kind, and location of the real roots, (2) the method of Bisection, (3) the method of Newton-Raphson, and the (4) method of Bairstow. This combined method is capable of returning all the roots both real and complex of a polynomial. One will note that the Bairstow method will do this, but for equations of sufficiently high degree some round-off error may be accumulated and convergence may be slow.

The above method (presented in flow chart form in a later section) will now be described in detail. First of all, it was necessary to determine if the polynomial $p(x)$ possibly had real roots. This was done through the use of some of the algebraic theorems discussed previously. Secondly, an upper and lower limit to the real roots were ascertained also by using one of these basic theorems. Third, a systematic procedure was used to obtained two values such that $p(a)\cdot p(b)<0$, the requirement for using the Bisection method. Fourth, those values were used as starting points for the method of Bisection. The method of Bisection was used until two approximations met the following criteria:

$$|x_n - x_{n+1}|<.01.$$

where x_n represents the nth approximation to the root. Fifth,

once this criterion was met, x_{n+1} was used as the starting value for the Newton-Raphson method. The Newton-Raphson method was allowed to continue until

$$|x_n - x_{n+1}| \leq .1x10\text{-}9.$$

The Newton-Raphson method will converge in all cases except possibly when

$$\frac{|p(x)p''(x)|}{[p'(x)]^2}$$

is greater than one, or $p'(x)$ is equal to zero or very close to zero. Sixth, once this final result was obtained, the root was factored from the polynomial leaving a polynomial of degree n-1. Seventh, the procedure was repeated (third through sixth steps) until all accessible real roots were removed from the polynomial. Note that roots very close together and multiple roots are considered inaccessible at this point. Eighth, once all accessible real roots were obtained, the procedure referred to the Bairstow method to remove those real and complex roots not already obtained.

Round-off error is the major concern in a method such as the one described above. Since this combined method was written to decrease the amount of round off error in the roots, a special step was implemented in the above procedure to accomplish this task. Once the starting values are obtained

for the next root, the method of Bisection is used utilizing the reduced equation, but once its criterion is satisfied the Newton-Raphson method is used utilizing the original polynomial. Through utilization of the original polynomial to obtain the final root to be removed from the equation, a more accurate final root is obtained and the coefficients of the reduced polynomial will contain a minimum amount of round-off error. It might also be mentioned here that double precision was used in all endeavors of this study thus allowing another assurance of maximum accuracy.

Since the Bairstow method is probably the most popular method today, the decision was made to form a comparison between the Bairstow and Bisnewbar methods. A total of forty-eight polynomials were used varying in degree from two to twenty. Multiple roots, wide-spread roots, and roots very close together were used to form polynomials thus allowing many different situations. The time of convergence, accuracy and dependability were all noted. Contained in the following pages are examples of results from this comparison. One will note how extremely fast both methods are. Double precision has been used on both methods, thus keeping accuracy at a maximum. In fact, for the epsilon involved, accuracy for both methods was generally the same.

Again, the letters NC mean the method did not converge. The increments of time are, as before, measured in seconds. The expression .329D03 is equivalent to 329.0.

COMPARISON EXAMPLES

I. Roots Are: -2.236, 2.236, -2.236, 2.236,
-3.0, -6.0, -1+2I, -1-2I, -1+2I, -1-2I

Bairstow Time: NC

Bisnewbar Time: 1.58

II. Roots Are: -2.236, 2.236, -2.236, 2.236,
9.0, -5.1, -3.2, 4.2

Bairstow Time: NC

Bisnewbar Time: 1.91

III. Roots Are: -9.2, -8.6, 3.0, 6.0, -1.2-4.6I, -1.2+4.6I,
8-6.5I, 8+6.5I, -8.3+4.6I, -8.3-4.6I

Bairstow Time: .88

Bisnewbar Time: 1.15

IV. Roots Are: -7.6, -3.0, -4.0, 7.1, -8.6, 4.1, -5.2,
-3.2, 6.1, -8.6, -8+4I, -8-4I

Bairstow Time: 3.15

Bisnewbar Time: 1.61

V. Roots Are: -2.0, -4.0, -3.7, -8.1, -4.1, -6.2, -3.9+6.2I,
-3.9-6.2I, -3.1+8.7I, -3.1-8.7I

Bairstow Time: .88

Bisnewbar Time: 1.20

COMPARISON EXAMPLES (con't)

VI. Roots Are: -1.9, -1.9, 4.7, -5.3, -6.0, 7.3, -6.1, 4.1

Bairstow Time: NC

Bisnewbar Time: .78

VII. Roots Are: -1.8, 3.2, 4.1, 3.0, -9.0, -5.0, -3.9+6.2I, -3.9-6.2I, 2.0-4.7I, 2.0+4.7I

Bairstow Time: .95

Bisnewbar Time: 1.15

VIII. Roots Are: -3.21, -9.11, -6.25, -8.11, -6.12, 1.42, 7.33, -2.12, -8.12, -4.22

Bairstow Time: 1.21

Bisnewbar Time: 1.18

IX. Roots Are: -3.12, -4.5, 7.45, 3.21, 9.12, 7.86, 2.89, -4.85, -7.56, -4.38, -5.46, 8.16, -2.99, -5.44

Bairstow Time: 2.70

Bisnewbar Time: 2.11

X. Roots Are: -1.9, -1.9, 4.7, -5.3, -6.0, 7.3, -6.1, 4.1, -6.3-4.2I, -6.3+4.2I, -1.9+6.2I, -1.9-6.2I

Bairstow Time: 1.43

Bisnewbar Time: 1.38

Variations and refinements could be made on the Bisnewbar method to assure better convergence under certain situations, but one must remember the point of this method was to produce a relatively fast converging method capable of keeping round-off errors to a minimum. Any drastic changes in the method could defeat the prime purpose of the method.

This ends the discussion of some of the problems and solutions of the polynomial root-finding field. The conclusions reached in the study of this field will be presented in the next section.

CONCLUSIONS

Choosing a method instrumental in finding the roots (zeroes) of $p(x) = 0$ is often a very difficult task. Many methods exist in this field, but each has its disadvantages. Speed, convergence, starting values, desired root, and accuracy are all key words relating to problems involved in selecting a method.

This author feels the basis for finding the roots of $p(x)$ lies in the ability to locate the roots roughly and supply proper starting values. Should we be able to supply the proper starting values, fast converging methods may be utilized which will keep computation time at a minimum. The question arises on how do we obtain this information on the roots. The best answer available consists of (1) obtaining information from rough graphing of the polynomial and (2) from the algebraic theorems mentioned throughout this study. The utilization of one or both of the above can allow us much information on the locations of the roots thus enhancing the possibility of supplying a good starting value.

Let us first consider the case of finding a single root of $p(x) = 0$. Assume we have obtained a starting value(s) and are now proceeding to choose a method for finding the desired root. Examining the following chart which the author has compiled in regard to his investigation of single root methods, we hope to obtain an idea on which method or methods to use in finding the root under consideration. Each of the methods

was rated as to the per cent of time it converged, and on how its speed compared with the other methods.

METHOD	DEPENDABILITY %	SPEED*	AVERAGE SPEED (SECONDS)
Bisection	100	4	.11
Newton-Raphson	95	1	.03
Secant	75	5	.18
Wegstein	40	2	.05
Aitken	38	3	.10
False Position	34	6	.50

* the methods are ranked from high to low (1 thru 6) depending on their average speed.

While many of these methods could apply in the case under consideration, some of them will under certain circumstances fail to return the desired root. If one is successful in forming an iteration of the form $x = f(x)$ such that $|f'(x)|<1$ for the starting value involved, convergence to the desired root is relatively assured if one uses the method of iteration. The main disadvantages here are that it is often hard to find the proper form of $x = f(x)$, and convergence is relatively slow.

Based upon the comparison of the eighteen polynomials, the author feels the "best" method for finding a single root lies in the combining of the Bisection method and the Newton-Raphson method. The Bisection method allows us dependability,

while the Newton-Raphson method allows us speed. If we can determine an interval [a,b] (as small as possible) such that $p(a)\cdot p(b)<0$, we can use the Bisection method to obtain $|x_n - x_{n+1}| \leq .01$, and refer to the Newton-Raphson to converge to the root in question. Since the Newton-Raphson method does require a good initial approximation, the Bisection method is used to provide this. It should be noted, however, that for roots of even multiplicity, the method of Bisection does not apply. One can, however, use the Newton-Raphson method since the complications of repeated roots only affect the method by a decrease in convergence speed.

Should we wish to find several or all of the real roots of p(x), a repetition of the above method may be used. Once we have obtained an interval such that $p(a)\cdot p(b)<0$, we may use the methods of Bisection and Newton-Raphson to find a root. Once the root is found, it may be removed from the polynomial by synthetic division leaving a polynomial of degree n-1. This procedure may be continued until all singular real roots have been removed, at which time we can use the Newton-Raphson method to remove multiple roots should they exist.

The two methods most popular for finding the complex roots of p(x) are the methods of Lin and Bairstow. The comparisons of this author have supported previous authors in that the Bairstow method appears to be the better of the two methods. The Bairstow method is of second order while the Lin method is of order - one, thus affording the Bairstow

method superior convergence. The Bairstow method is not as critical of the starting values of p and q as is the method of Lin. Should the complex roots of p(x) be desired, the method due to Bairstow will provide these with good accuracy and speed. The initial starting values of p = 0 and q = 0 are usually sufficient for the Bairstow method to converge.

Now we come to what the author feels is the most important aspect of polynomial root finding. This deals with the ability to find all the roots (real and complex) of p(x) = 0.

This problem can at times be extremely difficult. One must note, however, that there are special always-convergent methods available such as the Lehmer-Schur, Graeffe, and Bernoulli methods that will eventually obtain either all the real roots or both real and complex roots. These methods are, however, extremely slow and special cases arise in which the procedures may have to be altered in such a manner that programming becomes extremely complicated. The author suggests that if one of the above methods is used, it should only be used to provide approximations to the roots and then the use of a method such as the Newton-Raphson method should be implemented to converge to the true roots. In reference to the previous statement, the Bairstow method may also be used to supply rough approximations to the real roots of p(x), and again we may use the Newton-Raphson method to converge to the true roots.

The Bisnewbar method, developed by this author, and the Bairstow method were compared as to accuracy, speed and

dependability. Summarized in the following table are the results achieved in the comparing of the methods for forty-eight different polynomials.

METHOD	DEPENDABILITY %	SPEED*
Bairstow	62	42
Bisnewbar	100	58

* Speed is indicated by the per cent of the time one method was faster than its counterpart.

One will notice that the Bisnewbar method converged for 100% of the polynomials examined, and 58% of the time was faster than the Bairstow method. Since double precision and an epsilon (error) of .1D-09 were used on both the Bairstow and Bisnewbar methods, the overall accuracy appears to be the same for both methods.

The Bisnewbar method has proven to be better than the Bairstow method in that it converges more often, and on the average converges faster. The reason for this appears to be in the fact that the Bisnewbar method attempts to remove all singular real roots from the polynomial before any attempt is made to find the complex roots or roots of multiplicity. The Bisnewbar method appears to remove the influence of these roots, thus allowing the Bairstow part of the method to remove the remaining roots with some degree of complication

removed.

Thus, this author feels that the Bisnewbar method is a good candidate for finding the roots of p(x) because it is fast, accurate, and dependable.

APPENDIX I

This section contains a flow chart and program for the Bisnewbar method. Should the reader desire to use the program, certain information is essential. The user must supply the number of coefficients of the polynomial as an integer right justified in column four of the first data card. On the same card the actual coefficient should be entered in descending order, right justified in fields of fifteen. These coefficients should be in double precision and any coefficient non-existent in the polynomial should be represented as zero. Consider the following example:

$$p(x) = x^7 + 13x^6 - 7x^5 + 5x^4 + 3x^2 + 2x - 1 = 0.$$

The data would appear as:

Data card #1

COLUMN	4	19	34	48	64
	↓	↓	↓	↓	↓
DIGIT	8	1.D00	13.D00	-7.D00	5.D00

Data Card #2

COLUMN	15	30	45	60
	↓	↓	↓	↓
DIGIT	0.D00	3.D00	2.D00	-1.D00

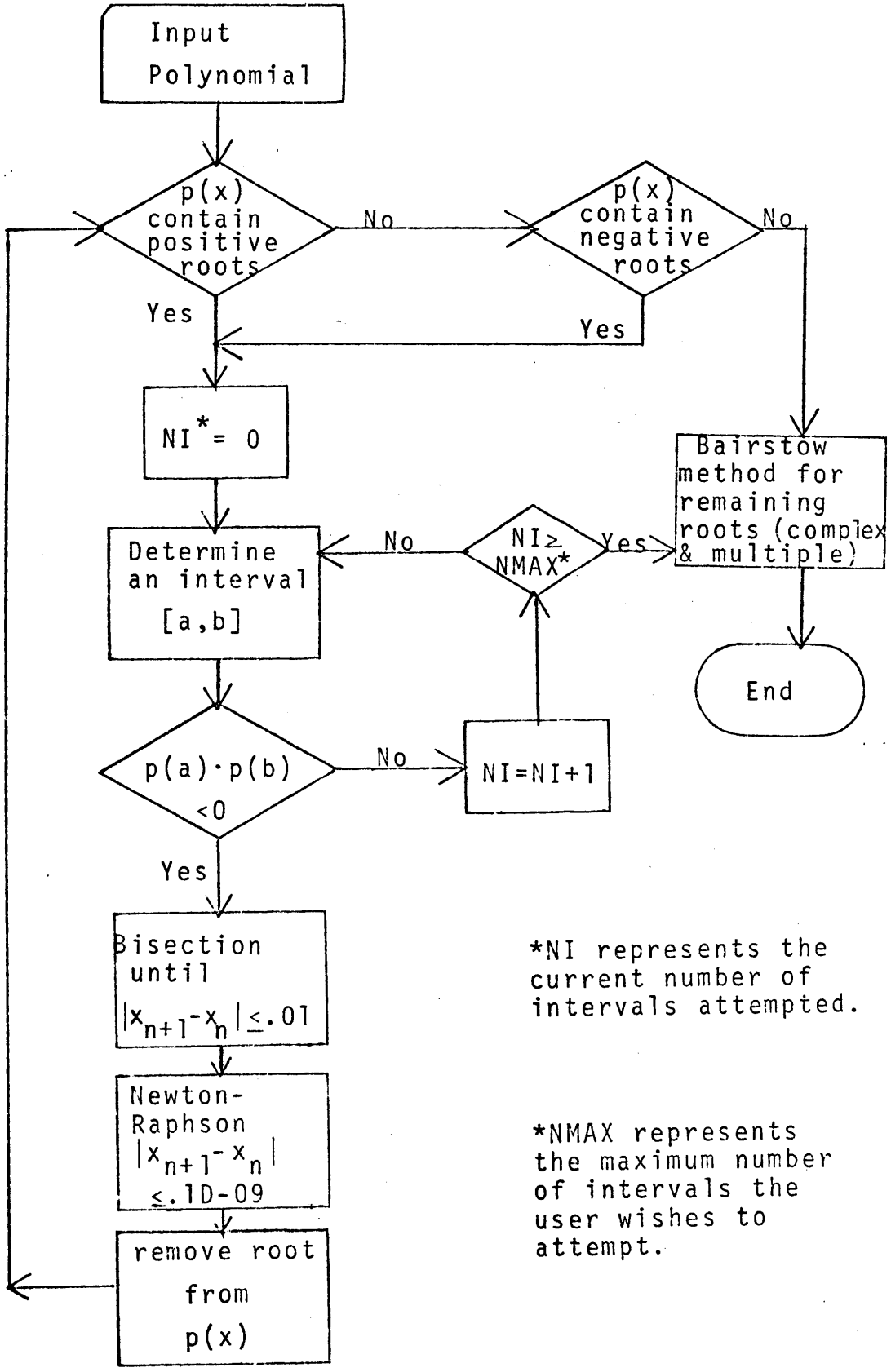
Note if more than four coefficients are entailed, the remaining coefficient should be put on successive data cards

four to a card right justified in fields of fifteen. If the previous instructions are followed, the Bisnewbar method will return all real and complex roots.

One should also note that there are several subprograms necessary to the Bisnewbar method. The routines are as follows: (1) a routine to determine an upper and lower bound to the real roots, (2) a routine to evaluate a polynomial (nested multiplication), (3) a routine to form the first derivative of p(x) and (4) a Bairstow method with double precision accuracy.

All necessary parts of the Bisnewbar method are contained in this appendix.

BISNEWBAR FLOWCHART



*NI represents the current number of intervals attempted.

*NMAX represents the maximum number of intervals the user wishes to attempt.

```
C     BISNEWRAP COMBINATION METHOD FOR SOLVING NTH DEGREE POLYNOMIALS
      DOUBLE PRECISION B(21),A(3,21),X,Y,AA,AC,SUM,C(21),D(21),EPS,UPP
      DOUBLE PRECISION TUPP                    ,TIME,H(21)
 1039 READ (1,163) NUM,(B(I),I=1,NUM)
  163 FORMAT(I4,(4D15.0))
      AC=B(1)
      DO 747 I=1,NUM
      B(I)=B(I)/AC
  747 H(I)=B(I)
      LND=NUM-1
      WRITE (3,892) LND,(B(I),I=1,NUM)
  892 FORMAT(' DEGREE OF POLYNOMIAL ',5X,I4,/,' COEFFICIENTS ARE ',5X,(5D1
     15.5))
      CALL CSOD2(B,C,LND)
      EPS=.1D-09
      AINC = .01
      KPT=1
      T = 1.
      NE=3
      TT=5.0
      CALL CSOD5(B,NE,UPP,TUPP,TT ,NUM)
      GO TO 103
  189 NDEG = NUM - 1
      IF(T) 947,943,943
  943 SUM=UPP
      GO TO 942
  947 SUM=-TUPP
  942 DO 439 I=1,NUM
  439 A(1,I)=B(I)
      NN = 0
C     ADAPTIONS TO BISECTION METHOD
      DELTA = .1
      NN = 1
      O = 1
      LKJ=1
```

```
      TQ =1
      IF(T) 431,434,434
  431 AB=TUPP
      GO TO 433
  434 AB=UPP
  433 BT= AB
      AC=0.0000
      IF(CSOD(B,NDEG,AC)) 118,119,117
  117 IF(CSOD(B,NDEG,AB)) 120,119,121
  119 X=AB
      GO TO 8
  132 X=BT
      GO TO 8
  118 AB = AC
  135 IF(CSOD(B,NDEG,BT)) 282,132,283
  283 AC = BT
      GO TO 120
C     ADAPTIONS FOR BISECTION METHOD
  121 GO TO (918,919,931),NN
  918 PR =  .5*(BT-AC)
      AB = PP
      R =  BT-PR
      S =  PP-AC
      NN = 2
      GO TO 117
  919 AB = PP + DELTA*(R)*Q
      NN = 3
      IF(AB*T-SUM) 117,136,136
  931 AB = PP-DELTA*(S)*Q
      NN = 2
      Q = Q + 1.
      IF(AB*T) 136,136,117
  282 GO TO (961,962,963),LKJ
  961 PR = .5*(BT-AB)
      BT = BT-PR
      BT = PP
```

```
      ST = PP-AB
      LKJ = 2
      GO TO 135
  962 PT = PP + DELTA * PT * TO
      LKJ = 3
      IF(PT*T-SUM) 135,136,136
  963 PT = PP - DELTA * ST * TO
      LKJ= 2
      TO = TO + 1.
      IF(PT*T) 136,136,135
  136 T = T *(-1)
 1432 IF(KPT-2)972,999,972
  972 KPT=2
      GO TO 139
  120 NITER = 0
    2 X = (AB+AC)/2.
      Y = CSPD(B,NDEG,X)
      IF(DABS(AC-X)-AINC)3,3,4
    4 NN= NN + 1
      IF(NN-20) 7,7,33
    7 IF(Y) 5,3,6
    5 AB = X
      GO TO 2
    6 AC = X
      GO TO 2
   33 WRITE (3,18)
   18 FORMAT(' BISECTION METHOD FAILED TO CONVERGE')
      GO TO 999
    3 NOT = 0
   30 Y = X-CSPD(H,LND ,X)/CSPD(C,LND-1 ,X)
      IF(DABS(Y-X)-EPS     )8,8,1
    1 X=Y
      NOT=NOT + 1
      IF(NOT-30) 30,43,43
   43 WRITE (3,91)
   19 FORMAT(3X5H X = ,D20.10)
```

```
 81 FORMAT(' NEWTON FAILED TO CONVERGE')
    GO TO 999
  3 WRITE (3,19) X
    A(3,1) = A(1,1)
    DO 35 J=1,NDEG
    A(2,J+1) = X * A(3,J)
    A(3,J+1) = A(2,J+1) + A(1,J+1)
    IF(DABS(A(3,J+1))-.1D-09) 636,636,35
636 A(3,J+1)=0.0D00
 35 CONTINUE
    NUM = NUM-1
    DO 888 I=1,NUM
888 B(I)= A(3,I)
    IF(NDEG-2) 82,82,103
 82 X = -A(3,2)
    WRITE (3,19) X
    GO TO 1038
103 L=1
    IF(T) 161,2103,2103
2103 DO 107 J=1,NUM
107 D(J)=B(J)
199 M=0
    N=0
    DO 133 J=1,NUM
    IF(D(J)) 158,133,220
158 IF(N-1) 36,99,36
 36 N=0
    M=1
    GO TO 133
220 IF(M-1) 138,99,138
138 M=0
    N=1
133 CONTINUE
    GO TO (161,999),L
 99 GO TO (190,191),L
190 T = 1.
```

```
      GO TO 189
  191  IF(T) 189,136,136
  161 LT=NDEG-1
      DO 101 I=1,NUM
  101 D(I)=B(I)
      DO 170 KL=2,20,2
      IF(LT  -KL) 388,77,170
   77 DO 173 J=2,LT   ,2
  173 D(J) = -B(J)
      GO TO 83
  388 DO 174 J=1,LT   ,2
  174 D(J) = -B(J)
      GO TO 83
  170 CONTINUE
   83 L=2
      GO TO 199
  999 LN=2
      CALL BAIR(A,B,NUM,LN)
 1038 CONTINUE
      GO TO 1039
      END

C     SUBROUTINE FOR DETERMINING DERIVATIVES
      SUBROUTINE CSDD2(A,B,N)
      DOUBLE PRECISION  A(21),B(21)
      DO 1  I=1,N
      FN = N+1-I
    1 B(I)= FN * A(I)
      RETURN
      END
```

```
      SUBROUTINE CSOD5(A,K,AL ,TUPP,AINC,NUM)
      DOUBLE PRECISION A(21),B(3,21),TIME,AL,UPP,TUPP
C     K=1 GIVES UPPER BOUND; K=2 GIVES LOWER BOUND ; K = 3 GIVES
C     BOTH
      L=1
   72 DO 2 J=1,NUM
    2 B(1,J) = A(J)
      B(3,1)= A(1)
      NDEG=NUM-1
      GO TO (37,42,37),K
   37 UPP =.001
  162 DO 3  J = 1,NDEG
      B(2,J+1)=UPP * B(3,J)
      B(3,J+1)=B(2,J+1)+ B(1,J+1)
      IF(B(3,J+1)) 18,3,3
    3 CONTINUE
      GO TO(19,20),L
      WRITE (3,142) UPP
  142 FORMAT(' UPP',F12.6)
   18 UPP = UPP + AINC
      GO TO 162
   19 WRITE (3,31) UPP
      AL=UPP
   31 FORMAT(' UPPER LIMIT',D18.6)
      GO TO (99,42,42),K
   42 DO 60 KL = 2,20,2
      IF(NDEG-KL) 8,7,60
    7 DO 73 J=2,NDEG,2
   73 B(1,J) = -B(1,J)
      GO TO 83
    8 DO 74 J=3,NDEG,2
   74 B(1,J) = -B(1,J)
      DO 87 NK=2,NUM
   87 B(1,NK) = -B(1,NK)
      GO TO 83
```

```
   60 CONTINUE
   83 L = 2
      B(3,1) = B(1,1)
      GO TO 27
   20 TUPP = -UPP

      WRITE (3,45) TUPP
   45 FORMAT(' LOWER LIMIT',D19.7)
   99 RETURN
      END
      SUBROUTINE BAIR(F,B,N2,NL)
C     BAIRSTOW METHOD
      DOUBLE PRECISION A(21),B(21),C(21),P,Q,E1,E2,AA,BB,AAA,BBB,AP,X1
      DOUBLE PRECISION T1,T2,D(21),TOL,DELP,DELQ,TIME
      NNL=0
      KPT=1
      LK=N2
      DO 492 I=1,N2
      D(I)=B(LK)
  492 LK=LK-1
      IF(NL) 83,82,83
   82 CONTINUE
   83 TOL=.1D-09
      P=0.0D00
      NN=N+1
      Q=0.0D00
      N=N2-1
      KK=N+1
      DO 301 IJK = 1,KK
  301 A(IJK)=D(IJK)
   11 B(KK)=A(KK)
      C(KK)=A(KK)
   13 B(KK-1)=A(KK-1)-P*B(KK)
      J=KK-2
   20 B(J)=A(J)-P*B(J+1)-Q*B(J+2)
      J=J-1
```

```
      IF(J-2)21,20,20
   21 B(1)=A(1)-Q*B(3)
      C(KK-1)=B(KK-1)-P*C(KK)
      J=KK-2
   30 C(J)=B(J)-P*C(J+1)-Q*C(J+2)
      J=J-1
      IF(J-2)31,30,30
   31 C(1)=B(1)-Q*C(3)
      T1=(C(2)-B(2)+P*C(3))
      T2=C(3)+P*C(4)
      DELQ=(B(1)*C(3)-B(2)*T1)/(T2*C(3)-C(4)*T1)
      DELP=(B(2)/C(3))-C(4)*DELQ/C(3)
      IF(DABS(DELP)-TOL)33,33,41
   33 IF(DABS(DELQ)-TOL)23,23,41

   41 CONTINUE
      P=P+DELP
      Q=Q+DELQ
      NNL=NNL+1
      IF(NNL-100) 13,13,134
  134 IF(KPT) 108,108,493
  493 KPT=0
  209 P=A(2)/A(3)
      NNL=0
      Q=A(1)/A(3)
      GO TO 11
   23 F1=P**2-4.0*Q
      IF(DABS(F1)-.1D-07)  121,121,120
  121 F1=0.0000
  120 F2=DSQRT(DABS(F1))
      IF(F1)51,52,52
   51 AA=(-1.0*P)/2.0
      BB=F2/2.0
   90 WRITE(3,55) AA,BB
      WRITE (3,56) AA,BB
      GO TO 100
   52 AAA=((-1.0*P)/2.0)+F2/2.0
```

```
      BBB=((-1.0*P)/2.0)-SQ/2.0
      AB=0.0
      WRITE (3,55) AAA,AB
      WRITE(3,56) BBB,AB
  100 N=N-2
      IF(N-2)104,101,102
  101 P=B(KK-1)/B(KK)
      Q=B(KK-2)/B(KK)
      GO TO 23
  102 NN=N+1
      DO 105 I=1,NN
  105 A(I)=B(I+2)
      KK=N+1
      NNL=0
      IF(KPT) 209,209,214
  214 P=0.0
      Q=0.0
      GO TO 11
  104 IF(N-1)108,106,106
  106 X1=-B(3)/B(4)

      AB=0.0
      WRITE (3,55) X1,AB
  108 IF(NL) 94,89,94
   89 CONTINUE
   94 RETURN
   55 FORMAT(3X5H X = ,D20.10,1X1H+,D20.10,1X1HI)
   56 FORMAT(3X5H X = ,D20.10,1X1H-,D20.10,1X1HI)
      END
```

```
C       FUNCTION FOR POLYNOMIAL EVALUATION
        DOUBLE PRECISION FUNCTION CSOD(A,N,X)
        DOUBLE PRECISION A(21),X,Y
        Y = A(1)
        DO 1 I =1,N
      1 Y = Y * X + A(I+1)
        CSOD = Y
        RETURN
        END
```

BIBLIOGRAPHY

1. Hamming, R. W. (1962) Numerical Methods for Scientists and Engineers. New York, McGraw-Hill, p. 351-356.

2. Grove, Wendell E. (1966) Brief Numerical Methods. Englewood Cliffs, New Jersey, Prentice-Hall, p. 1-30.

3. Booth, Andrew D. (1957) Numerical Methods. New York, Academic Press Inc., p. 142-153.

4. Hartree, D. R. (1952) Numerical Analysis. Oxford, Clarendon Press, p. 190-212.

5. Hildebrand, F. B. (1956) Introduction to Numerical Analysis. New York, McGraw-Hill, p. 458-476.

6. Pennington, Ralph H. (1965) Introductory Computer Methods and Numerical Analysis. New York, MacMillan Company, p. 214-279.

7. Conte, S. D. (1965) Elementary Numerical Analysis. New York, McGraw-Hill, p. 19-69.

8. Scarborough, James B. (1930) Numerical Mathematical Analysis. Baltimore, John Hopkins Press, p. 171-217.

9. Kunz, Kaiser S. (1957) Numerical Analysis. New York, McGraw-Hill, p. 1-37.

10. Todd, John (1962) Survey of Numerical Analysis. New York, McGraw-Hill, p. 255-277.

11. Ralston, Anthony (1965) A First Course in Numerical Analysis. New York, McGraw-Hill, p. 318-383.

12. Conkwright, Nelson B. (1957) Introduction to the Theory of Equations. Boston, Ginn and Company, 212 p.

13. Henrici, Peter (1964) Elements of Numerical Analysis. New York, John Wiley and Sons, p. 61-162.

14. Macon, Nathaniel (1963) Numerical Analysis. New York, John Wiley and Sons, p. 29-39.

15. Jennings, Walter (1964) First Course in Numerical Methods. New York, Macmillan Company, p. 23-40.

16. Nielson, Kaj. L. (1965) Methods in Numerical Analysis. 2nd Edition. New York, Macmillan Company, p. 172-223.

BIBLIOGRAPHY (con't)

17. Weeg, Gerard D. (1966) Introduction to Numerical Analysi Massachusetts, Blaisdell Publishing Company, p. 13-40.

18. Householder, Alston S. (1953) Principles of Numerical Analysis. New York, McGraw-Hill, p. 150-184.

19. Burnside, William S. (1960) Theory of Equations. Vol. 1 and 2. New York, Dover Publications, 286 p.

20. Berezin, I. S. (1965) Computing Methods. Oxford, Pergamon Press, p. 71-176.

21. Scarborough, James B. (1955) Numerical Mathematical Analysis. 3rd Edition. Baltimore, John Hopkins Press, p. 185-211.

22. Scarborough, James B. (1958) Numerical Mathematical Analysis. 4th Edition. Baltimore, John Hopkins Press, p. 185 -247.

VITA

The author was born May 31, 1943 in Amarillo, Texas. He received his primary and secondary education in the Amarillo school system. He received his college education from Amarillo Junior College, in Amarillo, Texas; West Texas State University in Canyon, Texas and the University of Missouri at Rolla, Rolla, Missouri. He received a Bachelor of Arts degree in Mathematics from West Texas State University in May, 1965, and at the time of this publication is a candidate for the Master of Science degree in Computer Science.

He has been enrolled in the Graduate School of the University of Missouri at Rolla since September, 1965, and has held the title of Graduate Assistant to the Registrar for the period September, 1965 to March, 1967.