The Mathematical Basis for a New Polynomial Rootfinder with Quadratic Convergence

T. E. HULL and R. MATHON
University of Toronto

Formulas developed originally by Weierstrass have been used since the 1960's by many others for the simultaneous determination of all the roots of a polynomial. Convergence to simple roots is quadratic, but individual approximations to a multiple root converge only linearly. However, it is shown here that the mean of such individual approximations converges quadratically to that root. This result, along with some detail about the behavior of such approximations in the neighborhood of the multiple root, suggests a new approach to the design of a polynomial rootfinder.

This paper first provides the relevant mathematical results: a short derivation of the formulas, convergence proofs, an indication of the behavior near a multiple root, and finally some error bounds. It then provides the outline of an algorithm based on these results, along with some preliminary experimental results to illustrate the major points, and to demonstrate some of the potential for a future software package. It should also be noted that the method is well-suited to take advantage of a parallel environment.

Categories and Subject Descriptors: G.1 [**Numerical Analysis**]: General—*error analysis, numerical algorithms;* Roots of Nonlinear Equations —*convergence, error analysis, iterative methods, methods for polynomials;* G.4 [**Mathematical Software**]: *Algorithm analysis*

General Terms: Algorithms, design

Additional Key Words and Phrases: Polynomial rootfinder, polynomial zerofinder, quadratic convergence

1. INTRODUCTION

Suppose that

$$P(z) = c_n z^n + c_{n-1} z^{n-1} + \cdots + c_0,$$

where the c's are in general complex coefficients, and $c_n \neq 0$.

Suppose also that $z_1, z_2, \cdots, z_n$, where the z_i are distinct, are approximations to the zeros of $P(z)$. Then consider the following iteration formulas:

$$\overline{z}_j = z_j - P(z_j)/Q'(z_j), \quad 1 \leq j \leq n,$$

where

$$Q(z) = c_n(z - z_1)(z - z_2) \cdots (z - z_n).$$

This work was supported by the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Centre of Ontario.
Author's addresses: Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4 (e-mail: {tehull,combin}@cs.utoronto.ca).

Since the 1960's these formulas have been used by many persons as the basis of methods for calculating approximations to all the roots of a polynomial simultaneously. The formulas were introduced much earlier, by Weierstrass in 1891 [20], but in another context related to the fundamental theorem of algebra. In 1962 Dochev [5] developed these formulas for the solution of polynomial equations. Using a different approach, they were rediscovered by Kerner [12] in 1966. (In 1960 Durand [6] had taken an approach like Kerner's but did not carry it to the point of developing formulas in the closed form shown here.) We will refer to the formulas as the WDK formulas.

We have experimented with these formulas for a number of years, using starting values distributed around a circle which is known to contain all the roots of the polynomial, and using stopping criteria similar what was proposed by Adams [4] and Peters and Wilkinson [16]. In earlier work with Trevizan [19] and Solowiej [18], as well as in more recent comparisons we have made with rootfinding programs in the Harwell [9], IMSL [10] and NAG [14] libraries, we have found programs based on these formulas to be both accurate and reliable – but relatively quite slow.

The main purpose of this paper is to develop some mathematical properties of the WDK formulas which will suggest ways in which rootfinding programs based on these formulas can be speeded up. The main new result is that the *average* of the m approximations to a root of multiplicity m converges quadratically to that root, whereas the individual approximations converge only linearly. We indicate how one might take advantage of this result, and we give a specific example of how this idea can be used to provide considerable speed-ups in particular cases.

We also mention other ways in which programs based on the WDK formulas might be improved. And of course the formulas are well-suited to take advantage of a parallel environment. We hope eventually to provide a generally useful piece of software based on the results described in this paper.

In Section 2 we give for completeness a simple derivation of the WDK formulas. Convergence to simple roots is established in Section 3, and to multiple roots in Section 4. Some detail about behavior near a multiple root is discussed in Section 5, and error bounds are developed in Section 6.

An algorithm based on the results in Sections 2-6 is outlined in Section 7. The main idea is to monitor progress in WDK iterations by, at intermediate stages in the calculation, using the individual error bounds to identify clusters of approximations that are close together and determining if the average of any such cluster can be taken to be a root of corresponding multiplicity. In Section 8 a specific program based on the ideas outlined in Section 7, along with some experimental results, are presented to illustrate the usefulness of this approach.

Other considerations to be taken into account in further development of practical rootfinding software, are mentioned in Section 9.

2. THE WDK FORMULAS

The polynomial in Section 1 can be rewritten in terms of its roots:

$$P(z) = c_n(z - r_1)(z - r_2)\cdots(z - r_n),$$

where the roots r_j are not necessarily distinct.

Recall that the approximations, $z_1, z_2, \cdots, z_n$, to these roots, where the z_j *are* distinct, were used to define

$$Q(z) = c_n(z - z_1,)(z - z_2)\cdots(z - z_n).$$

Now, suppose new approximations, $\overline{z}_1, \overline{z}_2, \cdots, \overline{z}_n$, are introduced where $\overline{z}_j = z_j + \Delta z_j$, $1 \le j \le n$, and $R(z)$ is defined by

$$R(z) = c_n(z - \overline{z}_1)(z - \overline{z}_2)\cdots(z - \overline{z}_n).$$

Then we can expand $R(z)$ in powers of the Δz's to obtain

$$R(z) = Q(z) - c_n \sum_{k=1}^{n} \Delta z_k \prod_{\substack{j=1 \\ j\neq k}}^{n} (z - z_j) + p(z),$$

where $p(z)$ is a perturbation consisting of higher order terms in the Δz's.

If we neglect $p(z)$ and insist that what is left be exactly $P(z_j)$ when $z = z_j$, we can solve for the Δz's to obtain

$$\Delta z_j = -P(z_j)/Q'(z_j),$$

so that

$$\overline{z}_j = z_j - P(z_j)/Q'(z_j),$$

which are the WDK formulas.

This simple derivation of the formulas is essentially the one given by Aberth [1].

An earlier derivation was based on the fact that the roots of $P(z)$ satisfy the following equations:

$$\begin{aligned}
\sum_{j=1}^{n} r_j &= -c_{n-1}/c_n, \\
\sum_{j=1}^{n} \sum_{k>j}^{n} r_j r_k &= c_{n-2}/c_n, \\
&\vdots \\
\prod_{j=1}^{n} r_j &= (-1)^n c_0/c_n,
\end{aligned}$$

where the left sides are the elementary symmetric functions of the roots. Durand [6] took this approach, and proposed using Newton's method for solving this system of n non-linear equations for the r's. This leads to a system of n linear equations for our Δz's. Kerner [12] followed this line of reasoning as well, but he also found analytic solutions for the system of linear equations, which turn out to be exactly the WDK formulas. (Dochev's approach [5] was altogether different, but led to the same formulas.)

For later purposes (Section 4) it is worth noting that the first of the n linear equations for the Δz's is

$$\sum_{j=1}^{n} \Delta z_j = -\sum_{j=1}^{n} z_j - c_{n-1}/c_n,$$

so that the sum of the new approximations in each iteration, and consequently their average, will remain constant, and equal to $-c_{n-1}/c_n$, throughout the iteration process.

Two features of the WDK formulas should be noted. One is that, since the approximations are determined simultaneously, there is no need to be concerned about difficulties introduced by deflation as there is with more traditional methods.

The other feature is that the formulas are obviously well-suited to take advantage of a parallel environment. This aspect has been investigated by Freeman [7] and Freeman and Bane [8].

In a serial environment it is natural to consider a Gauss-Seidel modification, and there is some experimental evidence that this is helpful (e.g., see Milovanović and Petković [13]), but our further analysis will assume the Jacobi approach directly implied by the formulas, and some of our results would not be valid if the Gauss-Seidel modification is used.

3. CONVERGENCE TO SIMPLE ROOTS

From the WDK formulas we have immediately that, for $1 \leq j \leq n$,

$$\begin{aligned} \overline{z}_j - r_j &= z_j + \Delta z_j - r_j \\ &= z_j - r_j - (z_j - r_j) \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{z_j - r_k}{z_j - z_k}. \end{aligned}$$

Now suppose that $\epsilon = \max_{1 \leq j \leq n} \mid z_j - r_j \mid$ and note that

$$\frac{z_j - r_k}{z_j - z_k} = 1 + \frac{z_k - r_k}{z_j - z_k}.$$

Then, if r_j is a simple root (so that, for sufficiently small ϵ, $\mid z_j - z_k \mid$ is bounded away from 0 for all $k \neq j$), we have

$$\frac{z_j - r_k}{z_j - z_k} = 1 + O(\epsilon),$$

so that

$$\begin{aligned} \overline{z}_j - r_j &= (z_j - r_j)(1 - (1 + O(\epsilon))) \\ &= O(\epsilon^2). \end{aligned}$$

Thus we have quadratic convergence to simple roots, as one would expect of a Newton method.

4. CONVERGENCE TO MULTIPLE ROOTS

Let us now consider convergence to a multiple root, r say, of multiplicity m. We can assume that $r_1 = r_2 = \cdots = r_m = r$, and that $r_{m+1}, r_{m+2}, \cdots, r_n$ are different from r, but not necessarily from each other. It is sufficient to consider this one case.

From the WDK formulas we now have, for $1 \leq j \leq m$,

$$\overline{z}_j - r = z_j - r - A \prod_{k=m+1}^{n} (\frac{z_j - r_k}{z_j - z_k}),$$

where

$$A = \frac{(z_j - r)^m}{\prod\limits_{\substack{k=1\\k\neq j}}^{m} (z_j - z_k)}.$$

As before we suppose that $\epsilon = max_{1\leq j\leq n} \mid z_j - r_j \mid$. Then the terms of the product in the expression for $\overline{z}_j - r$ can be shown as before to be $1 + O(\epsilon)$, so that

$$\overline{z}_j - r = z_j - r - A(1 + O(\epsilon)).$$

If we now further assume that the method does converge at least linearly, so that $\overline{z}_j - r = O(\epsilon)$, we can conclude that $A = O(\epsilon)$, and

$$\overline{z}_j - r = z_j - r - A + O(\epsilon^2).$$

To show that the convergence of the mean is quadratic, we sum the above over all j for $1 \leq j \leq m$. But the sum of the terms on the right, apart from the $O(\epsilon^2)$ terms, is exactly the sum of the approximations one would get by applying the WDK formulas to the equation $w^m = 0$, where $w = z - r$. And we know from the Durand-Kerner approach mentioned in Section 2, that the WDK formulas will always produce approximations whose sum is correct, namely 0 in this case. We can therefore conclude that

$$\sum_{j=1}^{m} \overline{z}_j / m - r = O(\epsilon^2),$$

as required.

To prove this quadratic convergence of the mean to multiple roots, we had to assume more than just that $\max_{1\leq j\leq n} \mid z_j - r_j \mid = \epsilon$, for otherwise a term in the denominator of A could, for example, be $O(\epsilon^2)$, in which case A could increase indefinitely as $\epsilon \to 0$.

We assumed that the $\overline{z}$'s converged at least linearly. If we had not made this assumption we would have had to assume something to make sure the z's approximating r were kept reasonably separated, so that A could not increase indefinitely. For example, we could have assumed that $\mid z_j - z_k \mid \geq c\epsilon$ for some $c > 0$ and all $j \neq k$ with $1 \leq j, k \leq m$. A more restrictive, but quite plausible assumption that satisfies our needs – and also is a good indication of what happens in practice – will be discussed in the next section.

5. BEHAVIOR NEAR A MULTIPLE ROOT

We now examine the behavior of the calculated approximations to a multiple root in the neighborhood of that root.

Suppose that r is a multiple root of multiplicity m, specifically that

$$P(z) = (z - r)^m F(z),$$

where $F(r) \neq 0$.

We can interpret what we did in Section 2 to determine the Δz's in the WDK formulas was to require that

$$R(z) = P(z) + p(z),$$

for $z = \overline{z}_j$, $1 \leq j \leq m$. But $R(\overline{z}_j) = 0$, so that we have

$$(\overline{z}_j - r)^m F(\overline{z}_j) + p(\overline{z}_j) = 0,$$

and this means that the m $\overline{z}$'s which approximate r can be represented by

$$\overline{z}_j = r + \left(\frac{-p(\overline{z}_j)}{F(\overline{z}_j)} \right)^{1/m}.$$

This suggests that the approximations to a multiple root tend to approach that root from uniformly spaced positions around a circle with center at the multiple root – and this in turn suggests that the mean of those approximations could be a particularly good approximation to the root.

This kind of behavior, or something like it, has been observed experimentally by a number of authors, such as Aberth [1], Pasquini and Trigiante [15], and Smith [17], and will be illustrated later (Section 8). Related results about the behavior of approximations near a multiple root have been given by Wilkinson [21, p.39].

Suppose we assume the approximations to a multiple root are distributed in such a manner. Specifically, let us assume that, for $1 \leq j \leq m$,

$$z_j = r + \delta e^{(2\pi j/m + \alpha)i}(1 + O(\epsilon)),$$

for some fixed δ and α.

Then, proceeding as in Section 4, we obtain again

$$\overline{z}_j - r = z_j - r - A(1 + O(\epsilon)),$$

where now

$$A = (z_j - r) \prod_{\substack{k=1 \\ k \neq j}}^{m} \frac{z_j - r}{z_j - z_k}$$

$$
\begin{aligned}
&= (z_j - r) \prod_{\substack{k=1\\k\neq j}}^{m} \frac{\delta e^{(\frac{2\pi j}{m}+\alpha)i}}{\delta e^{(\frac{2\pi j}{m}+\alpha)i} - \delta e^{(\frac{2\pi k}{m}+\alpha)i}}(1+O(\epsilon)) \\
&= (z_j - r) \prod_{\substack{k=1\\k\neq j}}^{m} \frac{1}{1 - e^{\frac{2\pi(k-j)}{m}i}}(1+O(\epsilon)) \\
&= (z_j - r) \prod_{k=1}^{m-1} \frac{1}{1 - e^{\frac{2\pi k}{m}i}}(1+O(\epsilon)) \\
&= (z_j - r)\frac{1}{m}(1+O(\epsilon))
\end{aligned}
$$

(this last equality follows from the fact that

$$
\prod_{k=1}^{m-1}(1 - e^{\frac{2\pi k}{m}i}) = \lim_{z\to 1}\left(\frac{z^m - 1}{z-1}\right) = m).
$$

Thus, we have, for $1 \leq j \leq m$,

$$
\begin{aligned}
\overline{z}_j - r &= (z_j - r)(1 - \frac{1}{m}(1+O(\epsilon)))(1+O(\epsilon)) \\
&= \frac{m-1}{m}(z_j - r) + O(\epsilon^2).
\end{aligned}
$$

This shows that our new assumption about the individual approximations to a multiple root is preserved – with δ replaced by $(m-1)\delta/m$, which is not surprising for a Newton method near a root of multiplicity m. It also establishes the *linear* convergence of the individual approximations. (For the special case $m = 1$, we of course have quadratic convergence.)

Consider now the mean of the approximations to r. We have

$$
\overline{z}_j - r = \frac{m-1}{m}(\delta e^{(2\pi j/m+\alpha)i}) + O(\epsilon^2),
$$

so that the mean

$$
\begin{aligned}
\left(\sum_{j=1}^{m}\overline{z}_j - r\right)/m &= \frac{m-1}{m^2}\delta e^{\alpha i}\sum_{j=1}^{m} e^{\frac{2\pi j}{m}i} + O(\epsilon^2) \\
&= O(\epsilon^2),
\end{aligned}
$$

since the sum on the right is the sum of the mth roots of unity, which is 0. This establishes the quadratic convergence of the mean, under the new assumption about the distribution of the z's.

6. ERROR BOUNDS

We want now to determine error bounds, first for an individual approximation to a root, and then for the mean of a cluster where the mean is being considered as the approximation to a multiple root.

From the theory of partial fractions we can write

$$P(z)/Q(z) = 1 - \sum_{k=1}^{n} \Delta z_k/(z - z_k),$$

provided of course that $P(z_k) \neq 0$. If we put $z = r_j$, we have

$$1 = \sum_{k=1}^{n} \Delta z_k/(r_j - z_k).$$

Taking absolute values, we obtain

$$1 \leq \sum_{k=1}^{n} \mid \Delta z_k/(r_j - z_k) \mid .$$

The maximum term in the sum will occur at a particular value of k, and for that value of k,

$$1 \leq n \mid \Delta z_k/(r_j - z_k) \mid,$$

so that

$$\mid z_k - r_j \mid \leq n \mid \Delta z_k \mid .$$

The right side is the first bound we wished to obtain. We thus have a bound associated with each approximation. (This does not mean that there is a root within each bound of the corresponding approximation – only that there is an approximation within its bound of each root. However, the union of the discs enclosed by approximations and their associated bounds does contain all the roots – so the bounds are valid in the Gerschgorin sense. It can also be shown by the usual continuation argument that, if the discs are disjoint, each disc contains exactly one root (e.g., see Aberth [2, p.79]).)

Another bound can be obtained by first going back to

$$1 \leq \sum_{k=1}^{n} \mid \Delta z_k/(r_j - z_k) \mid .$$

and noticing that if k is such that $\mid r_j - z_k \mid$ is a minimum for $1 \leq k \leq n$, then, for that value of k,

$$\mid z_k - r_j \mid \leq \sum_{k=1}^{n} \mid \Delta z_k \mid .$$

We have a choice between the two bounds we have just found, or we could use whichever is the smaller of the two.

We turn now to finding an error bound for the mean of a cluster. In this case we assume that the polynomial $Q(z)$ approximating $P(z)$ has roots z_k of multiplicity m_k and that $P(z_k) \neq 0$, $1 \leq k \leq s$. We can now write

$$P(z)/Q(z) = 1 - \sum_{k=1}^{s} \sum_{\ell=1}^{m_k} d_{k\ell}/(z - z_k)^{\ell},$$

where

$$d_{k\ell} = \frac{1}{(m_k - \ell)!} \left[P(z) / \prod_{\substack{i=1 \\ i \neq k}}^{s} (z - z_i)^{m_i} \right]_{z=z_k}^{(m_k - \ell)} .$$

If we put $z = r_j$, we can conclude that

$$1 \leq \sum_{k=1}^{s} \sum_{\ell=1}^{m_k} \mid d_{k\ell} / (r_j - z_k)^{\ell} \mid .$$

The maximum term in the sum will occur at particular values of k and ℓ, and, for those values,

$$1 \leq n \mid d_{k\ell} / (r_j - z_k)^{\ell} \mid .$$

This yields the bound

$$\mid z_k - r_j \mid \leq \max_{1 \leq \ell \leq m_k} (n \mid d_{k\ell} \mid)^{1/\ell}.$$

We now have an error bound for the mean of a cluster. (If the j-th disc is disjoint from the rest, then it contains exactly m_j roots of $P(z)$ (see Aberth [2, p.83]).)

It was indicated earlier in Section 1 that starting values could be obtained by distributing them around a circle which is known to contain all the roots of the polynomial. One way of choosing such a circle is to consider all the roots to be just one cluster, and take its center to be the mean of all the roots, i.e. $-c_{n-1}/(nc_n)$, and to take the radius to be the bound just found in the special case when it is assumed that $Q(z)$ has only one root, say z_0, of multiplicity n, at the mean. This leads to

$$\max_{1 \leq \ell \leq n} \left(\frac{n}{(n-\ell)!} \left| \frac{P^{(n-\ell)}(z_0)}{c_n} \right| \right)^{1/\ell}$$

as the radius.

7. AN ALGORITHM TO ILLUSTRATE

We present the outline of an algorithm in Figure 1. It shows how the formulas and ideas in the preceding sections can be organized in a relatively straightforward way to determine approximations to the roots of a polynomial. In the next section we will show how a program based on this outline demonstrates the main ideas in the preceding sections, and we indicate that such a program can be both reliable and accurate. Making such a program more efficient and more generally useful will be discussed briefly in Section 9.

We first discuss the outline in more detail. The algorithm expects the input of n (the degree of the polynomial) and *coeffs* (an array of the $n+1$ complex coefficients of the polynomial). It returns *roots* (an array of n approximations to the complex roots of the polynomial), *errors* (an array of n real error bounds associated with the roots), and *ind* (an integer value to indicate why the return occurred: because convergence was achieved, or the maximum number of allowed iterations was reached, or some error condition occurred).

```
ROOTFINDER(n : int, coeffs : complex 0..n, ind: int,
      roots: complex 1..n, errors : real 1..n)
   look after special cases (e.g., n = 1, 2)
   determine starting values (the z's)
   other initializations (numits, maxits, etc.)
   loop
      use WDK formulas to compute Δz's
      compute individual bounds and use them to
         identify clusters and determine centers of clusters
      if convergence or numits = maxits then
         set ind accordingly
   ......... exit loop
      end if
      update z := z + Δz
      numits := numits + 1
   end loop
   compute cluster bounds
   assign to roots and errors
   return
end ROOTFINDER
```

Fig. 1. Outline of an algorithm which shows how the formulas and ideas of Sections 2-6 can be organized into a procedure for finding simultaneous approximations to all the roots of a polynomial.

The procedure first looks after special cases, such as when $n = 1$ or 2, and possibly 3 or 4, or when there are leading or trailing zero coefficients. It then determines starting values, for example by distributing the initial approximations around a circle which is known to contain all the roots (e.g., using known bounds, or the bound at the end of the preceding section). Then other initializations are needed, such as setting the iteration counter *numits* to zero, and assigning a maximum number of iterations to *maxits*.

The first statement and the last two statements inside the loop carry out the basic WDK iteration, and also keep track of how often the iteration has been completed.

The other statements inside the loop start with a situation in which the z's and Δz's are known. This situation is analyzed, first by using the Δz's to determine individual error bounds according to one or both of the first two bounds given in Section 6. These bounds, along with their associated z's define n discs in the complex plane. Isolated discs correspond to simple roots. The others form clusters, which are identified, and the mean of each cluster is determined.

To determine if convergence has occurred, the program can use a running error analysis (e.g., as in Adams [4] or Peters and Wilkinson [16]) to determine if, for each cluster of multiplicity m, the error bounds in evaluating the polynomial and its first $m-1$ derivatives are greater than the corresponding calculated values of the polynomial and its derivatives. If they are, in each case, convergence is assumed. (By including the situations in which $m = 1$ in the above test, we have also included the simple roots.)

If convergence has occurred, or if the number of iterations has reached the maximum allowed, the program sets the indicator *ind* to an appropriate value, and then assigns appropriate values to the *roots* and *errors* arrays. It already has all these values available, except that it must compute the bounds for the clusters with $m > 1$, and to do that it uses the remaining formulas for the bounds given in Section 6.

Error returns might also occur. For example, overflow or underflow might occur, but most programming languages make it difficult or even impossible to cope with such situations in a convenient way. Sometimes such situations can be anticipated, in which case an error return can be made, along with an appropriate value of *ind*.

8. A SPECIFIC PROGRAM AND SOME RESULTS

For a preliminary test of the ideas outlined in Section 7 we have developed a double precision Fortran program for finding roots of polynomials, along with a test program which determines the speed and accuracy of polynomial rootfinders over a large number of test problems. The programs run on a Sun 4.

Our rootfinder does not make any provision for special cases. For starting values, it first calculates a center and radius as described at the end of Section 6. It distributes $n-1$ starting values around the circle so that the radii through these values make the angles $2\pi k/n$, $1 \leq k \leq n-1$, with a horizontal line through the center, and then chooses a final starting value so that its angle is π/n The maximum number of iterations is arbitrarily set to 200.

The bounds for determining clusters are obtained from the minimum of the first two bounds given in Section 6. In this way a disc is associated with each approximation. A collection of approximate roots is identified as a cluster if the union of their discs form a continuous region which is disjoint from all the other discs.

The test for convergence is based on the running error analysis described by Peters and Wilkinson [16], but using Kahan's suggestion for reducing the number of operations [16, p.20].

Our test program has two special features. If the roots of a test polynomial are not supplied, it can find those roots by calculating them first in very high precision and then rounding them to double precision. (The results are saved for future testing.) It can also determine a measure of accuracy of the approximations produced by any particular zerofinder. To do this it uses an algorithm for matching the true roots pairwise with the approximations in such a way that the maximum deviation of the true values, from the approximations with which they are paired, is a minimum.

The first test problem is the following:

Problem 1 – *monic polynomial with*

a simple root at $(1, 2)$,

a triple root at $(3, -1)$,

a quintuple root at $(5, 3)$.

Graphical output from the test program shows the progression of approximations produced by our rootfinder. Figure 2(b) gives a closer view of what happens near the quintuple root, and Figure 2(c) shows a close-up of what happens near the quintuple root. The main point that is illustrated here is that the program has analyzed clusters near the multiple roots and, at one stage, long before the individual approximations

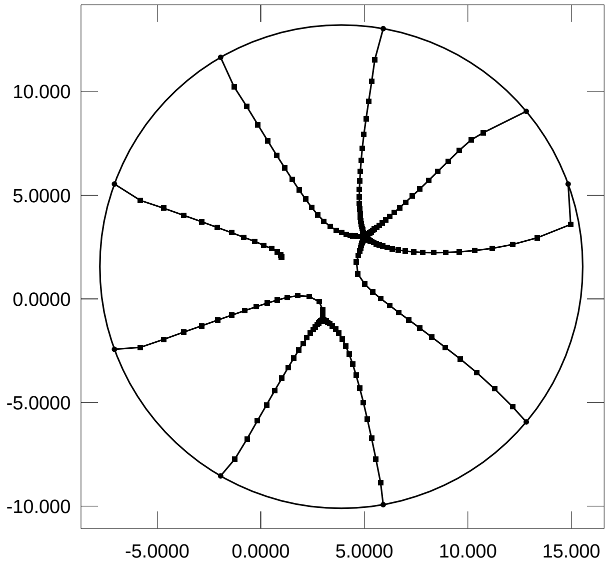


Fig.2(a) Overall view of progress of rootfinder on problem 1 in Section 8.

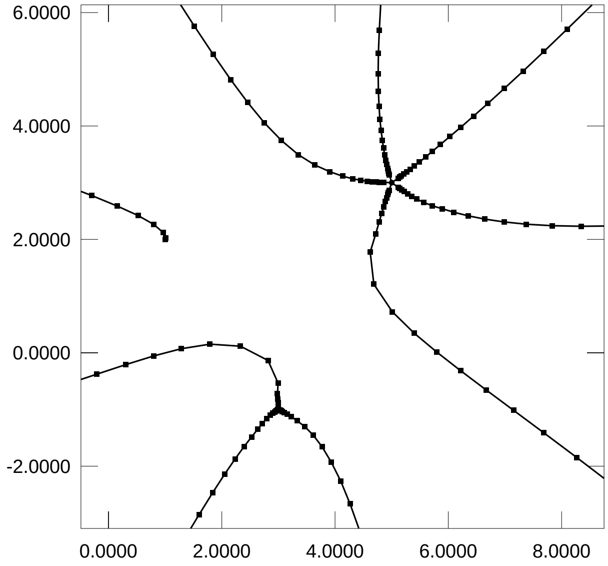


Fig.2(b) Closer view of progress towards quintuple root at (5,3).

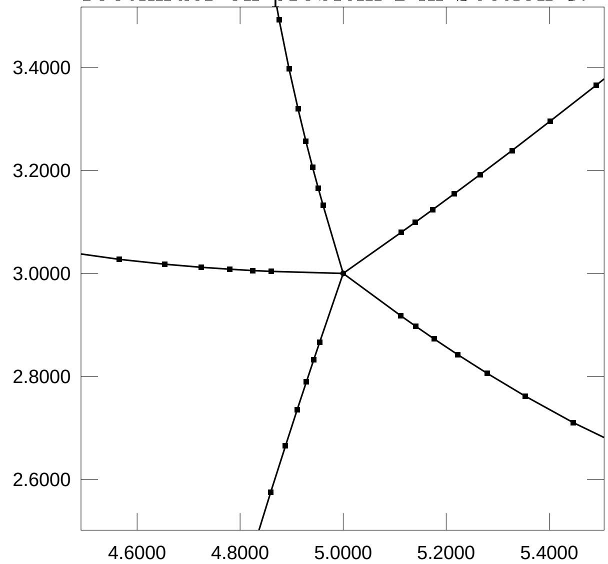


Fig.2(c) Close up of progress near the quintuple root at (5,3).

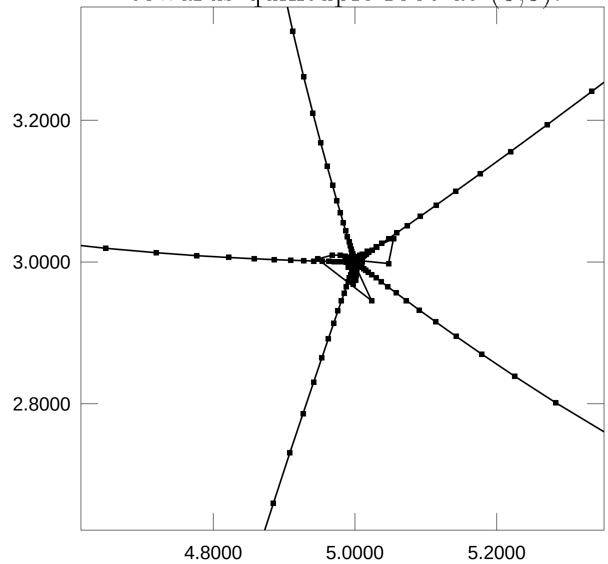
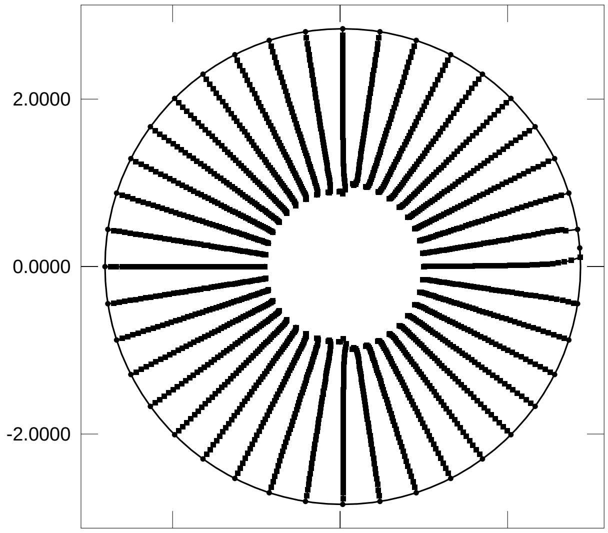


Fig.2(d) Close-up if test for multiplicity is removed from rootfinder.



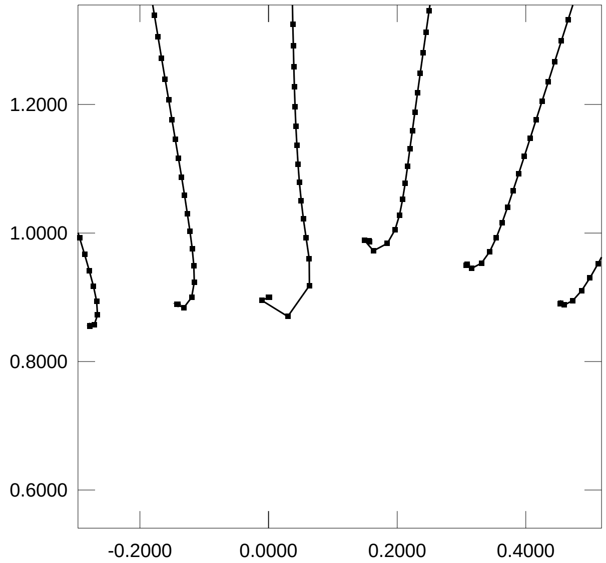


Fig.3(a) Overall view of progress of rootfinder on problem 2 in Section 8.

Fig.3(b) Closer view of progress towards a few of the roots of problem 2 near (0,1).

get very near those roots, it finds that the means of the individual approximations pass the convergence test for multiple roots. It then accepts these means as multiple roots, calculates their error bounds, and terminates execution. By contrast, Figure 2(d) shows what happens near the quintuple root if the test for multiplicity is removed from the rootfinder.

It should also be noted that the individual approximations to the quintuple root converge on that root in the "equiangular" manner suggested in Section 5. We have found this to be the case in all our test results.

The test program determined that the true error for each of the three roots was bounded by the error bound produced by the rootfinder. These errors were: true errors of 2.7×10^{-15}, 1.8×10^{-11} and 0.46×10^{-9}, and computed bounds of 7.8×10^{-15}, 5.3×10^{-11}, and 2.32×10^{-9}, for the roots (1,2), (3,-1), and (5,3), respectively.

When the test for multiplicity was removed the errors in the approximations near (5,3) were approximately 10^{-2}, despite having used twice as many iterations (48 vs. 24).

The results for this test problem are typical of what happened with a large number of test problems, including all of the test problems suggested by Jenkins and Traub [11]. Figure 3(a) gives an overall view of what happens when our rootfinder is presented with one like the last (and very challenging) problems suggested by Jenkins and Traub. This is the following problem:

Problem 2 – *the monic polynomial with*

roots at $e^{k\pi i/20}$, $k = -9, -8, \cdots, 9$

and at $.9e^{k\pi i/20}$, $k = 10, 11, \cdots, 30$

Figure 3(b) gives a closer view at what happens for a few of the roots of this polynomial near $(0, 1)$.

The true error for each of the 40 approximations was less than the corresponding computed error bound. The maximum of these true errors was 1.57×10^{-16}, while the maximum of the computed error bounds was 1.85×10^{-15}.

There are no multiplicities to contend with here. The point is only to illustrate that the rootfinder can handle some quite challenging polynomials.

We also did some comparisons with rootfinding programs in the Harwell [9], IMSL [10] and NAG [14] libraries. As mentioned earlier, our experience has been that our rootfinder has almost always been at least as accurate and reliable, but is frequently slower, and sometimes much slower.

9. CONCLUDING REMARKS

We plan to improve the rootfinder in a number of ways in order to provide a version which will be a more generally useful piece of software. We will of course provide for the special cases of low degree polynomials, but our primary interest is in trying to speed up the program. For example, we are experimenting with ways of getting off to a faster start. We have also carried out some preliminary experiments in which the program uses the multiplicity test not just to test for convergence, but also to decide earlier if convergence is close enough that the approximations can be moved towards the mean before carrying out the next iteration. It might also help to avoid doing the convergence test at every step in the iteration process.

We also plan to prepare a detailed report on the testing program, including a comprehensive list of test problems.

ACKNOWLEDGEMENTS

Besides Catalina Trevizan and Margaret Solowiej, whom we have already mentioned, we would like to thank Edgar Smart and Tom Fairgrieve for their help in much of the earlier programming and testing. Tom Fairgrieve also developed most of our current test program, while Hisao Tamaki designed and implemented the algorithm for matching the true roots with the computed approximations to those roots. We also wish to thank Pierre Milman for stimulating dicussions during the preparation of this paper.

We also very much appreciate the generosity of Harwell, IMSL and NAG for giving us Sun 4 versions of their rootfinding programs. And we found the high precision package developed by Aberth and Schaeffer [3] very helpful in finding true solutions for some of our test polynomials.

REFERENCES

1. Aberth, O. Iteration methods for finding all zeros of a polynomial simultaneously. *Math. Comp. 27*, 122 (1973), 339–344. (See also Statement of Priority. *Math. Comp. 30*, 135 (1976), 680.).

2. Aberth, O. *Precise Numerical Analysis.* Wm. C. Brown Publishers, Dubuque, Iowa, 1988.

3. Aberth, O., and Schaeffer, K. J. Precise computation using range arithmetic, via C++. *ACM Trans. Math. Softw. 18*, 4 (Dec. 1992), 481-491.

4. Adams, D. A. A stopping criterion for polynomial root finding. *ACM Comm. 10*, 10 (1967), 655–658.

5. Dochev, K. A modified Newton method for the simultaneous approximate calculation of all the roots of a given algebraic equation. *Phys.-Math. Journal, Bulgarian Academy of Science*, 5 (1962), 136–139 (in Bulgarian).

6. Durand, E. *Solutions Numériques des Équations Algébriques, Tome I (Équations du Type F(x) = 0, Racines d'un polynôme).* Masson, Paris, 1960.

7. Freeman, T. L. Calculating polynomial zeros on a local memory parallel computer. *Parallel Comput. 12* (1989), 351–358.

8. Freeman, T. L., and Bane, M. K. Asynchronous polynomial zero-finding algorithms. *Parallel Comput. 17* (1991), 673–681.

9. Harwell Subroutine Library. A catalogue of subroutines, Harwell, Sept. 1990.

10. IMSL User's Manual. International Mathemetical and Statistical Libraries, Houston, 1985.

11. Jenkins, M. A., and Traub, J. F. Principles for testing polynomial zerofinding programs. *ACM Trans. Math. Softw. 1*, 1 (1975), 26–34.

12. Kerner, I. O. Ein Gesamtschrittfahren zur Berechnung der Nullstellen von Polynomen. *Numer. Math. 8* (1966), 290–294.

13. Milovanović, G. V., and Petković. On computational efficiency of the iterative methods for the simultaneous appriximation of polynomial zeros. *ACM Trans. Math. Softw. 12*, 4 (1986), 295–306.

14. NAG Fortran Library Manual, Mark 15A. Numerical Algorithms Group Ltd., Oxford, 1991.

15. Pasquini, L., and Trigiante, D. A globally convergent method for simultaneously finding polynomial roots. *Math. Comp. 44*, 169 (1985), 135–149.

16. Peters, G., and Wilkinson, J. H. Practical problems arising in the solution of polynomial equations. *J. Inst. Math. Appl. 8* (1971), 16–35.

17. Smith, B. T. Error bounds for zeros of a polynomial based upon Gerschgorin's theorems. *JACM 17*, 4 (1970), 661–674.

18. Solowiej, M. *N-Dimensional Newton Method with Precision Control for Determining Roots of Complex Polynomials.* M.Sc. thesis, Dept. of Comp. Sc., Univ. of Toronto, 1983.

19. Trevizan, C. A. *A Zero Finding Algorithm for Complex Polynomials.* M.Sc. thesis, Dept. of Comp. Sc., Univ. of Toronto, 1977.

20. Weierstrasse, K. Neuer Beweis des Satzes, dass jede ganze rationale Function einer Veränderlichen dargestellt werden kann als ein Product aus linearen Functionen derselben Veränderlichen (1891). *Ges. Werke 3* (1903), 251–269.

21. Wilkinson, J. H. *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, N.J., 1963.