Numerical computation of polynomial zeros by means of Aberth's method*

Dario Andrea Bini

Dipartimento di Matematica, Università di Pisa, via Buonarroti 2, I-56127 Pisa, Italy
E-mail: bini@dm.unipi.it

Received 16 September 1995
Communicated by Å. Björck and M. Redivo-Zaglia

A mio padre

An algorithm for computing polynomial zeros, based on Aberth's method, is presented. The starting approximations are chosen by means of a suitable application of Rouché's theorem. More precisely, an integer $q \geqslant 1$ and a set of annuli $\mathcal{A}_i$, $i = 1, \ldots, q$, in the complex plane, are determined together with the number k_i of zeros of the polynomial contained in each annulus $\mathcal{A}_i$. As starting approximations we choose k_i complex numbers lying on a suitable circle contained in the annulus $\mathcal{A}_i$, for $i = 1, \ldots, q$. The computation of Newton's correction is performed in such a way that overflow situations are removed. A suitable stop condition, based on a rigorous backward rounding error analysis, guarantees that the computed approximations are the exact zeros of a "nearby" polynomial. This implies the backward stability of our algorithm. We provide a Fortran 77 implementation of the algorithm which is robust against overflow and allows us to deal with polynomials of any degree, not necessarily monic, whose zeros and coefficients are representable as floating point numbers. In all the tests performed with more than 1000 polynomials having degrees from 10 up to 25,600 and randomly generated coefficients, the Fortran 77 implementation of our algorithm computed approximations to all the zeros within the relative precision allowed by the classical conditioning theorems with 11.1 average iterations. In the worst case the number of iterations needed has been at most 17. Comparisons with available public domain software and with the algorithm PA16AD of Harwell are performed and show the effectiveness of our approach. A multiprecision implementation in MATHEMATICA^{TM} is presented together with the results of the numerical tests performed.

Keywords: polynomial zeros, Aberth's method, numerical test, starting approximations.

AMS subject classification: 65H05, 65Y20.

* Work performed under the support of the ESPRIT BRA project 6846 POSSO (POlynomial System SOlving).

1. Introduction

Let

$$p(x) = \sum_{i=0}^{n} a_i x^i = a_n \prod_{i=1}^{n} (x - \alpha_i), \quad a_0, a_n \neq 0, \tag{1}$$

be a polynomial of degree n having coefficients in the complex field $\mathbb{C}$ and zeros α_i, $i = 1, \ldots, n$.

The problem of approximating the zeros α_i, $i = 1, \ldots, n$, of $p(x)$ given its coefficients, is a classical problem in pure and applied mathematics which has received a growing interest particularly for the computational and numerical issues strongly related to its solution. In the wide literature on this problem (see the excellent bibliography collected in [31]) we may find contributions on the theoretical analysis of the complexity of approximating polynomial zeros [4, 36, 37, 39, 41], as well as more concrete numerical algorithms for their computation [1, 3, 10–12, 16, 18, 21–24, 26, 27, 29, 30, 36, 38, 39, 43–45]. On the latter subject we refer the reader also to GAMS, the Guide on Available Matematical Software, whose information can be obtained by means of anonymous ftp at http://gams.nist.gov.

Among analytic algorithms a certain interest has been devoted to Durand–Kerner and Aberth methods [1, 11, 12, 27, 44] particularly for their good features in a parallel model of computation. In fact, these methods allow the simultaneous approximation of all the zeros. These methods have been widely analyzed in the literature (see [3, 9, 14, 15, 18, 19, 28]), and some implementations have been proposed in [9, 14] and in the SLATEC library (see GAMS). The methods of Aberth and Durand–Kerner have local superlinear convergence to simple zeros and local linear convergence to multiple zeros.

Let us denote $x_i^{(k)}$, $i = 1, \ldots, n$, the set of approximations to the n zeros of $p(x)$ at the kth step of the algorithm. Then the Durand–Kerner iteration is given by

$$x_i^{(k+1)} = x_i^{(k)} - \frac{p(x_i^{(k)})}{a_n \prod_{j=1,\ j\neq i}^{n} (x_i^{(k)} - x_j^{(k)})}, \quad i = 1, \ldots, n, \tag{2}$$

whereas the Aberth iteration is given by

$$x_i^{(k+1)} = x_i^{(k)} - \frac{p(x_i^{(k)})/p'(x_i^{(k)})}{1 - \left(p(x_i^{(k)})/p'(x_i^{(k)})\right) \sum_{j=1,\ j\neq i}^{n} 1/(x_i^{(k)} - x_j^{(k)})}, \quad i = 1, \ldots, n. \tag{3}$$

In this paper we present an implementation of Aberth's method for the simultaneous approximation of all the zeros in floating point arithmetic with a fixed machine precision (sections 5 and 6), and draw the general lines of a multiprecision algorithm based on both Aberth's and Durand–Kerner's iteration (section 7), which rely on the following tools:

1. A new criterion for selecting initial approximations to the zeros, which is based on a suitable application of Rouché's theorem (section 2).

2. A formula for the computation of the value $p(x)/p'(x)$ which avoids overflow problems (section 3).
3. A criterion for stopping the iterations, based on a rigorous backward rounding error analysis, which guarantees that each computed approximation is the zero of a "nearby" polynomial (section 4).
4. A guaranteed a posteriori error bound on the approximation error, i.e., for each approximation x_i to α_i, a real positive machine number r_i is provided such that $|x_i - \alpha_i| < r_i|x_i|$ (section 4).
5. The property of quadratic convergence of the mean [15] of the Durand–Kerner iteration (used only for the multiprecision algorithm, section 7).

Concerning the criterion for selecting starting approximations, we first consider a method, based on Rouché's theorem, for computing an integer $\hat{q}$, $2 \leqslant \hat{q} \leqslant n$, a set of integers $0 = \hat{k}_1 < \hat{k}_2 < \cdots < \hat{k}_{\hat{q}} = n$, and a set of radii $s_{\hat{k}_i}$, $t_{\hat{k}_i}$, $0 \leqslant s_{\hat{k}_i} < t_{\hat{k}_i}$, $i = 1, \ldots, \hat{q}$, $s_0 = 0$, $t_n = +\infty$, such that the open annulus $\{z \in \mathbb{C}\colon s_{\hat{k}_i} < |z| < t_{\hat{k}_i}\}$ of center 0 and radii $s_{\hat{k}_i}$, $t_{\hat{k}_i}$, contains no zeros of the polynomial $p(x)$ as well as of any polynomial $c(x) \in \mathcal{P}(p) = \{\sum_{j=0}^{n} b_j x^j\colon |b_j| = |a_j|,\ j = 0, \ldots, n\}$, for $i = 1, \ldots, \hat{q}$, while the closed annulus $\mathcal{A}_i = \{z \in \mathbb{C}\colon t_{\hat{k}_i} \leqslant |z| \leqslant s_{\hat{k}_{i+1}}\}$ of radii $t_{\hat{k}_i}$ and $s_{\hat{k}_{i+1}}$ contains exactly $\hat{k}_{i+1} - \hat{k}_i$ zeros of $p(x)$, as well as of any polynomial $c(x) \in \mathcal{P}(p)$, for $i = 1, \ldots, \hat{q}$. Then we propose a cheaper criterion, based on the above technique, for selecting initial approximations distributed along different circles in such a way that each annulus $\mathcal{A}_i$ contains $\hat{k}_{i+1} - \hat{k}_i$ approximations. Our criterion is defined by the following steps:

1. Compute the upper envelope of the convex hull of the set $\{(i, \log|a_i|)\colon i = 0, \ldots, n\}$, and denote $0 = k_1 < k_2 < \cdots < k_q = n$, the abscissas of the vertices of this set.
2. Compute
$$u_{k_i} = \left|\frac{a_{k_{i-1}}}{a_{k_i}}\right|^{1/(k_i - k_{i-1})}, \quad i = 2, \ldots, q.$$
3. For $i = 1, \ldots, q-1$ select $k_{i+1} - k_i$ points of moduli $u_{k_{i+1}}$, that is, put
$$x_{k_i+j}^{(0)} = u_{k_{i+1}} \exp\left(\left(\frac{2\pi}{k_{i+1} - k_i} j + \frac{2\pi i}{n} + \sigma\right)\mathrm{i}\right),$$
$$j = 1, \ldots, k_{i+1} - k_i, \qquad i = 1, \ldots, q-1,$$
where i is the imaginary unit such that $\mathrm{i}^2 = -1$, and σ is any nonzero number.

Concerning the computation of $p(x)/p'(x)$ by means of the Ruffini–Horner rule, we observe that overflow problems are very likely whenever $|x| > 1$ and the degree n of the polynomial is very large. For this reason the computation of $p(x)/p'(x)$ is reduced to the computation of $p_R(y)/p_R'(y)$ for $y = x^{-1}$, where $p_R(x) = \sum_{i=0}^{n} a_{n-i} x^i$.

Moreover, we prove the backward numerical stability of the Ruffini–Horner rule even for complex polynomials, deduce from this property the efficient stop condition

$$|\mathrm{fl}(p(x))| < \mu \mathrm{fl}(s(|x|)), \quad s(x) = \sum_{i=0}^{n} |a_i|(1+4i), \tag{4}$$

where $\mathrm{fl}(\cdot)$ denotes the actual computed value in the floating point arithmetic endowed with the machine precision μ ($\mu = 2^{-53}$ for the IEEE arithmetic), and finally obtain the following guaranteed a-posteriori error bound

$$|x_i - \alpha_i| \leqslant r_i, \quad r_i = n \frac{|\mathrm{fl}(p(x_i))| + \mu \mathrm{fl}(s(|x_i|))}{||\mathrm{fl}(p'(x_i))| - \mu \mathrm{fl}(s'(|x_i|))|}.$$

In the light of the result [33], we prove that if the stop condition (4) is satisfied for $x = \xi$ then there exists a polynomial $\widetilde{p}(x) = \sum_{i=0}^{n} \widetilde{a}_i x^i$ such that $\widetilde{p}(\xi) = 0$ and $\widetilde{a}_i = a_i(1+\varepsilon_i)$, $|\varepsilon_i| < 2(1+4i)\mu$, $i = 1, \ldots, n$.

The algorithm that we obtain in this way, implemented in Fortran 77 in floating point arithmetic, resulted to be very robust and numerically stable. Unlike the available software, it allowed to approximate successfully all the zeros of polynomials of very high degree (greater than 20,000) within the precision guaranteed by the conditioning theorems, and to deal with very extreme situations like polynomials with very large and very small coefficients where the normalization by the leading coefficient would generate an overflow condition, and polynomials having some zeros which are too large to be represented as floating point. The situations where our algorithm may fail are extremely unlikely and are detected by the code itself by means of warning messages.

The iteration (3) has been implemented in a Gauss–Seidel style, furthermore, at each step of (3) only the components $x_i^{(k+1)}$ for which the condition (4) is not satisfied for $x = x_i^{(k)}$ are updated. The program stops if either (4) is satisfied for $x = x_i^{(k)}$, $i = 1, \ldots, n$, or if the number k of iterations exceeds the limit of 30.

The algorithm delivers n approximations x_i, $i = 1, \ldots, n$, to the zeros α_i together with n relative error bounds r_i, $i = 1, \ldots, n$, such that $|r_i - \alpha_i| < r_i|x_i|$, and a vector of Boolean components giving information about the reliability of each approximation and of the associated error bound.

The storage required by our program is that needed to store the $n+1$ input coefficients, the n output approximations of the zeros, the n radii which bound the errors and the boolean vector. Moreover, two auxiliary real vectors of $n+1$ components are used.

The cost of each iteration performed on a single component is $24n + O(1)$ real arithmetic operations (ops), where we used the weight 2 for complex additions and the weight 6 for complex multiplications and reciprocations.

In the numerical experiments performed with 1000 polynomials having randomly generated coefficients and degrees 10, 20, 50, 100, 200, the number of iterations for each computed zero has been at most 16 and its average value has been 11.1.

We performed further numerical experiments with specific polynomials, having ill-conditioned zeros, large degree (up to 25,600) or large coefficients. In all the cases the output precision reached the maximum value allowed by the classical conditioning theorems [17, 24].

We compared our program with the programs CZERO of the library NAPACK (Jenkins and Traub algorithm), CPZERO of the library SLATEC (Durand–Kerner's algorithm), available at the URL address http://netlib.bell-labs.com/netlib, and with algorithm PA16AD of the Harwell library. Our program resulted to be faster than CZERO and CPZERO, and comparable with PA16AD in terms of CPU time. On the other hand, PA16AD as well as CPZERO failed to compute the zeros for several test polynomials due to overflow, more specifically in the cases of degree greater than 850 and in the cases of polynomials with large coefficients. CZERO failed in the case where the normalization performed in order to reduce the polynomial in monic form, leads to overflow conditions.

Finally, in section 7 we outline the features concerning the preliminary multi-precision implementation of our algorithm performed in MATHEMATICA^{TM}. In this multiprecision version the program delivers approximations of all the zeros within the precision requested by the user and dynamically adjusts the precision of the floating point arithmetic to the numerical conditioning of each zero. In this way no waste of computational effort is done and the property of quadratic convergence of the mean exploited by Durand–Kerner's method is used to avoid linear convergence in the case of multiple or clustered zeros.

We report a few results of this implementation which show the efficiency of our approach. We refer the reader to the paper [5] for further improvements and for a more efficient implementation of this algorithm. The Fortran 77 code is available on the web at the URL address http://netlib.bell-labs.com/netlib/numeralgo/.

2. Choosing the starting approximations

The choice of the initial points $x_i^{(0)}$, $i = 1, \ldots, n$, for starting the iteration (2) or (3), is rather delicate since the number of steps needed by the iterative method to reach a given approximation strongly depends on it. In [14] several criteria for choosing the initial approximations are analyzed, in [1] the Aberth iteration is started by selecting n equispaced points on a circle of center 0 and radius r, where r is an upper bound to the moduli of the zeros. In [19] a more involved procedure is introduced for selecting a radius r which yields an intermediate value between the maximum and the minimum modulus of the zeros. These criteria of selecting the starting points are inadequate in the case where the polynomial $p(x)$ has zeros with large moduli and zeros with small moduli at the same time. Indeed, in this case it is not possible to choose good approximations to all the zeros by fixing a single circle.

Unlike the criteria for choosing the initial approximations introduced in [1, 14] and [19], our criterion performs this choice by selecting complex numbers along different circles and relies on the result of [35]. We first recall the following tool for locating the zeros of rational functions [40].

Theorem 1 (Rouché). Let $f(x)$ and $g(x)$ be rational functions which are regular in a simply connected bounded domain D and on its boundary Γ, and suppose that

$$|f(x)| > |g(x)|, \quad x \in \Gamma.$$

Then $f(x)$ and $F(x) = f(x) + g(x)$ in D have the same number of zeros, counted according to their multiplicity.

We need also the following localization theorem [24].

Theorem 2. Let $p(x) = \sum_{i=0}^{n} a_k x^k$ be the polynomial (1) having zeros $\alpha_1, \ldots, \alpha_n$, and such that $a_n, a_0 \neq 0$. Then we have

$$|\alpha_i| \leqslant \max\{|a_0/a_n|, 1 + |a_1/a_n|, \ldots, 1 + |a_{n-1}/a_n|\},$$
$$|\alpha_i^{-1}| \leqslant \max\{|a_n/a_0|, 1 + |a_1/a_0|, \ldots, 1 + |a_{n-1}/a_0|\},$$

for $i = 1, \ldots, n$.

Now assume that $a_k \neq 0$ and put $f(x) = a_k x^k$, $g(x) = p(x) - f(x)$. If $r > 0$ is such that

$$|a_k x^k| > |p(x) - a_k x^k| \quad \text{for } |x| = r, \tag{5}$$

then, from theorem 1 it follows that the polynomial $p(x)$ has k zeros in the open disk $C(r) = \{x \in \mathbb{C}: |x| < r\}$ of center zero and radius r.

Since

$$|p(x) - a_k x^k| \leqslant \sum_{i=0,\ i\neq k}^{n} |a_i||x|^i,$$

then any $\theta_k > 0$ which solves the inequality

$$|a_k|\theta^k > \sum_{i=0,\ i\neq k}^{n} |a_i|\theta^i, \tag{6}$$

provided it exists, is such that $p(x)$ has k zeros in the disk $C(\theta_k)$. Therefore we restrict our attention to the inequality (6).

Indeed, (6) can be more easily manipulated than (5), on the other hand, by restricting our interest to (6), we may lose solutions of (5). In other words, the information that we recover from (6) is weaker than that obtained by (5), in fact it concerns localization properties shared by any polynomials $c(x) \in \mathcal{P}(p) = \{\sum_{i=0}^{n} b_i x^i: |b_i| = |a_i|,\ i = 0, \ldots, n\}$. This information is strict if it is referred to all the polynomials in the class $\mathcal{P}(p)$, as is shown in the following

Theorem 3. Let θ_k be a positive solution of (6), then any polynomial $c(x) \in \mathcal{P}(p)$ has k zeros in the disk $C(\theta_k)$. Moreover, if θ_k solves the equation $|a_k|\theta^k = \sum_{i=0,\ i\neq k}^{n} |a_i|\theta^i$ then there exists a polynomial $c(x) \in \mathcal{P}(p)$ having a zero on the boundary of $C(\theta_k)$ and $C(\theta_k)$ contains either k or $k-1$ zeros of $c(x)$.

Proof. The proof of the first part of the theorem is trivial. The proof of the second part follows from remark 1. □

Observe that any solution θ_k of (6) (if it exists) satisfies the condition $\theta_k^k|a_k| > \theta_k^i|a_i|$ for any $i \neq k$, that is,

$$\left|\frac{a_i}{a_k}\right|^{1/(k-i)} < \theta_k < \left|\frac{a_j}{a_k}\right|^{1/(k-j)}, \quad i < k < j. \tag{7}$$

Whence we obtain the inequalities

$$u_k < \theta_k < v_k, \quad k = 0, \ldots, n, \tag{8}$$

where

$$\begin{aligned} u_k &= \max_{i<k} \left|\frac{a_i}{a_k}\right|^{1/(k-i)}, \quad k = 1, \ldots, n, \\ v_k &= \min_{i>k,\ a_i \neq 0} \left|\frac{a_i}{a_k}\right|^{1/(k-i)}, \quad k = 0, \ldots, n-1, \\ u_0 &= 1 \Big/ \left(1 + \max_{i>0} \left|\frac{a_i}{a_0}\right|\right), \\ v_n &= 1 + \max_{i<n} \left|\frac{a_i}{a_n}\right|, \end{aligned} \tag{9}$$

and the expressions for u_0 and v_n hold in the light of theorem 2.

From (8) it follows that if $u_k \geqslant v_k$ then there exist no positive solutions to equation (6). Moreover, for $k = 0$ and $k = n$ there exists only one positive solution to the equation

$$|a_k|\theta^k - \sum_{i=0,\ i\neq k}^{n} |a_i|\theta^i = 0, \tag{10}$$

which gives, for $k = n$, the radius of the circle containing all the zeros of $p(x)$ (compare [24]).

As pointed out in [35], if $0 < k < n$ then (10) can have either no positive solutions or two positive solutions. Below we give an inductive proof of this fact.

Theorem 4. Assume that $a_0, a_n \neq 0$. If (10) has a solution for $k \neq 0, n$, then it must have two solutions, counted with their multiplicity, and they belong to the interval $[u_k, v_k]$.

Proof. First observe that the number of positive solutions is even. In fact, the continuous function in (10) is negative at zero and at $+\infty$. Moreover, if $a_k = 0$ there are no positive solutions of (10). Now, let us proceed by induction on k. For $k = 1$ there are

two solutions, in fact, they are given by the intersection of the straight line of equation $y = |a_1|x$ and the convex function of equation $y = \sum_{i=0,\ i\neq k}^{n} |a_i|x^i$ (observe that the second derivative of the latter function is positive for $x > 0$). For the inductive step assume that for $k = h$, $1 < h < n-1$, the number of positive solutions is either 2 or zero and prove it for $k = h+1$. Consider the case $a_k \neq 0$, since otherwise the number of solutions of (10) would be zero. If for $k = h+1$ there were more than 2 solutions, these would be at least 4. Therefore the first derivative of the function in (10) would have at least three positive zeros. This contradicts the inductive assumption since the derivative of (10) for $k = h+1$ has the same representation of (10) where k is replaced by h and $a_i\theta^i$ by $ia_i\theta^{i-1}$. □

Consider the case where (10) has two positive solutions t_k, s_k such that $u_k \leqslant t_k < s_k \leqslant v_k$, then (6) is satisfied for $t_k < \theta < s_k$, consequently (5) is satisfied for $r = \theta$. Therefore from Rouché's theorem we may deduce that the polynomial $p(x)$, as well as any polynomial $c(x) \in \mathcal{P}(p)$, has k zeros in the closed disk of radius t_k and no zeros in the open annulus determined by the disks of radii t_k and s_k.

Remark 1. From theorem 4 it follows that if (10) has two positive solutions $t_k < s_k$, then the polynomial $c(x) = |a_k|x^k - \sum_{i=0,\ i\neq k}^{n} |a_i|x^i \in \mathcal{P}(p)$, has k zeros in the disk $C(r)$ for $t_k < r < s_k$. Moreover the positive zeros t_k and s_k lie on the boundary of $C(t_k)$, $C(s_k)$, respectively. This proves the second part of theorem 3.

Theorem 4 allows us to recover important information on the localization of the zeros of $p(x)$. In fact, we have the following result [35].

Theorem 5. Let $0 = \hat{k}_1 < \hat{k}_2 < \cdots < \hat{k}_{\hat{q}} = n$ be all the values of k for which (10) has a positive solution, and let $s_0 = s_{\hat{k}_1} < t_{\hat{k}_2} < s_{\hat{k}_2} < \cdots < t_{\hat{k}_{\hat{q}-1}} < s_{\hat{k}_{\hat{q}-1}} < t_{\hat{k}_{\hat{q}}} = t_n$ be these solutions. Moreover define $t_0 = 0$, $s_n = +\infty$. Then, any polynomial $c(x) \in \mathcal{P}(p)$ has $\hat{k}_{i+1} - \hat{k}_i$ zeros in the closed annulus $\mathcal{A}_i = \{z \in \mathbb{C}\colon s_{\hat{k}_i} \leqslant |z| \leqslant t_{\hat{k}_{i+1}}\}$ of radii $s_{\hat{k}_i}$, $t_{\hat{k}_{i+1}}$, for $i = 1, \ldots, \hat{q}-1$, and no zeros in the open annulus of radii $t_{\hat{k}_i}$, $s_{\hat{k}_i}$, for $i = 1, \ldots, \hat{q}$.

Observe that the inclusion results expressed in theorem 5 are strict if referred to the class $\mathcal{P}(p)$. In fact, as can be easily seen, there exist polynomials $c(x) \in \mathcal{P}(p)$ having zeros on the boundary of the annuli of radii $s_{\hat{k}_i}$, $t_{\hat{k}_{i+1}}$. Moreover, it is proved in [35] that the set made up by the zeros of all the polynomials $c(x) \in \mathcal{P}(p)$ coincides with $\bigcup_{i=1}^{\hat{q}-1} \mathcal{A}_i$.

These inclusion results can be used for determining a set of starting points for Aberth's iterations. Different strategies can be applied. We may choose $\hat{k}_{i+1} - \hat{k}_i$ points randomly distributed in the annulus of radii $t_{\hat{k}_i}$, $s_{\hat{k}_{i+1}}$, or, alternatively, we may choose $\hat{k}_{i+1} - \hat{k}_i$ equispaced zeros on the circle of radius $t_{\hat{k}_i}$, or we may distribute the zeros on the circle of radius $(t_{\hat{k}_i} + s_{\hat{k}_{i+1}})/2$.

A very cheap strategy, that we have adopted in our implementation, allows us to select $\hat{k}_{i+1} - \hat{k}_i$ starting approximations in the annulus $\mathcal{A}_i$ without computing $t_{\hat{k}_i}$ and $s_{\hat{k}_i}$. The idea consists in selecting all the integers k_i, $i = 1, \ldots, q$, such that $u_{k_i} \leqslant v_{k_i}$ no matter whether (10) has positive solutions or not, and to choose $k_{i+1} - k_i$ equispaced points on the circle of radius u_{k_i}.

More precisely, the starting approximations are selected in the following way:

1. Compute an integer q and $k_1 < k_2 < \cdots < k_q$ such that $u_{k_i} \leqslant v_{k_i}$, $i = 1, \ldots, q$, $u_j > v_j$ for $j \neq k_i$, where u_k and v_k are defined by (9).
2. For $i = 1, \ldots, q-1$ select $k_{i+1} - k_i$ points of moduli $u_{k_{i+1}}$, that is, put

$$x_{k_i+j}^{(0)} = u_{k_{i+1}} \exp\left(\left(\frac{2\pi}{k_{i+1} - k_i} j + \frac{2\pi i}{n} + \sigma\right) \mathrm{i}\right),$$
$$j = 1, \ldots, k_{i+1} - k_i, \qquad i = 1, \ldots, q-1, \tag{11}$$

where i is the imaginary unit such that $\mathrm{i}^2 = -1$, and σ is any nonzero number.

Since $\{\hat{k}_1, \ldots, \hat{k}_q\} \subset \{k_1, \ldots, k_q\}$ and $u_{\hat{k}_i} \leqslant t_{\hat{k}_i} < s_{\hat{k}_i} \leqslant v_{\hat{k}_i}$, with the criterion (11) we select $\hat{k}_{i+1} - \hat{k}_i$ initial approximations in the annulus $\mathcal{A}_i$ of radii $t_{\hat{k}_i}$, $s_{\hat{k}_{i+1}}$. Moreover this selection is cheap since it avoids the approximation of the solutions $t_{\hat{k}_i}$ and $s_{\hat{k}_i}$ of (10). The computation of the integers k_i, $i = 1, \ldots, q$, at stage 1 and of the radii $u_{k_{i+1}}$ at stage 2, can be efficiently performed by using the same technique of computational geometry exploited in [3, 39].

Consider the set $\mathcal{C} = \{(i, \log|a_i|),\ i = 0, \ldots, n\}$, and define the upper envelope of the convex hull of $\mathcal{C}$ as the set $\mathrm{convex}(\mathcal{C}) = \{(k_i, \log|a_{k_i}|),\ 0 = k_1 < k_2 < \cdots < k_q = n\}$ such that the piece-wise linear function obtained by joining the point $(k_i, \log|a_{k_i}|)$ with $(k_{i+1}, \log|a_{k_{i+1}}|)$, for $i = 1, \ldots, q-1$, is convex and lies above the points $(i, \log|a_i|)$, $i = 0, \ldots, n$.

We may prove the following result.

Theorem 6. The set $\{k_1 < k_2 < \cdots < k_q\}$ made up by the vertices of $\mathrm{convex}(\mathcal{C})$ is such that $u_{k_i} \leqslant v_{k_i}$, $i = 1, \ldots, q$ (compare (9)) and $u_i > v_i$ for $i \neq k_1, \ldots, k_q$. Moreover, we have

$$v_{k_i} = u_{k_{i+1}} = \left|\frac{a_{k_i}}{a_{k_{i-1}}}\right|^{1/(k_{i+1}-k_i)} \qquad \text{for } i = 1, \ldots, q-1.$$

Proof. The convexity conditions in k_i, for $1 < i < q$ are given by

$$\left|\frac{a_{k_i}}{a_{k_{i-1}}}\right|^{1/(k_i-k_{i-1})} < \left|\frac{a_j}{a_{k_i}}\right|^{1/(j-k_i)}, \quad j < k_i,$$

$$\left|\frac{a_{k_i}}{a_{k_{i+1}}}\right|^{1/(k_i-k_{i+1})} > \left|\frac{a_{k_i}}{a_j}\right|^{1/(j-k_i)}, \quad j > k_i,$$

$$\left|\frac{a_{k_i}}{a_{k_{i+1}}}\right|^{1/(k_i-k_{i+1})} \leqslant \left|\frac{a_{k_i}}{a_{k_{i-1}}}\right|^{1/(k_i-k_{i-1})}.$$

Whence,

$$u_{k_i} = \max_{j<k_i} \left|\frac{a_j}{a_{k_i}}\right|^{1/(k_i-j)} = \left|\frac{a_{k_{i-1}}}{a_{k_i}}\right|^{1/(k_i-k_{i-1})} \leqslant \min_{j>k_i} \left|\frac{a_j}{a_{k_i}}\right|^{1/(k_i-j)}$$
$$= \left|\frac{a_{k_{i+1}}}{a_{k_i}}\right|^{1/(k_i-k_{i+1})} = v_{k_i}.$$

Moreover

$$v_{k_i} = \left|\frac{a_{k_{i+1}}}{a_{k_i}}\right|^{1/(k_i-k_{i+1})} = u_{k_{i+1}}.$$

The inequalities $u_i > v_i$, $i \neq k_1, \ldots, k_q$, can be similarly proved. □

Since the computation of the upper envelope of the convex hull of $n+1$ points in the plane can be carried out in at most $2n\lceil \log_2 n \rceil$ convexity tests, the computation of the starting approximations according to (11) costs $O(n \log n)$ arithmetic operations in the light of theorem 6.

3. Implementation of Newton's correction

The evaluation of $p(x)/p'(x)$ at $x = \xi$, can be performed by computing $p(\xi)$ and $p'(\xi)$ by means of the well-known Ruffini–Horner rule. That is, $p(\xi)$ is computed as remainder of the division of $p(x)$ by $x-\xi$, while $p'(\xi)$ is computed as remainder of the division of $q(x)$ by $x-\xi$, where $q(x)$ is the quotient of $p(x)$ and $x-\xi$. Alternatively, the value of $p'(x)$ can be computed by means of the Ruffini–Horner rule applied to the polynomial $\sum_{i=0}^{n-1}(i+1)a_{i+1}x^i$.

This computation is numerically stable, since the values of $p(\xi)$ and $p'(\xi)$ actually computed with a floating point arithmetic can be viewed as the exact values obtained from the coefficients of a slightly perturbed polynomial. In order to see this we state the following straightforward result which extends to the complex case some properties of the floating point arithmetic.

Theorem 7. Let $x = x_1 + ix_2$, $y = y_1 + iy_2$, $z = z_1 + iz_2$, $w = w_1 + iw_2$, be complex numbers such that $z = x + y$, $w = xy$. Then, for the values $\tilde{z}$, $\tilde{w}$ actually computed in floating point arithmetic, with machine precision μ, by means of the relations $z_1 = x_1 + y_1$, $z_2 = x_2 + y_2$, $w_1 = x_1y_1 - x_2y_2$, $z_2 = x_1y_2 + x_2y_1$, we have

$$\tilde{z} = z(1+\nu), \qquad \tilde{w} = w(1+\eta), \quad |\nu| < \mu, \ |\eta| < 2\sqrt{2}\mu + O(\mu^2).$$

From the above theorem we easily obtain the following result:

Theorem 8. Let $\mathrm{fl}(p(\xi))$ be the value computed by means of a floating point arithmetic having machine pecision μ by means of the Ruffini–Horner rule

$$p(\xi) = \big(\cdots\big((a_n\xi + a_{n-1})\xi + \cdots + a_1\big)\xi\big) + a_0$$

then $\mathrm{fl}(p(\xi)) = \tilde{p}(\xi)$, where

$$\tilde{p}(x) = \sum_{i=0}^{n} a_i(1+\varepsilon_i)x^i,$$
$$|\varepsilon_i| < \big((2\sqrt{2}+1)i+1\big)\mu + \mathrm{O}(\mu^2). \tag{12}$$

A similar relation holds for the first derivative $p'(x)$.

Proof. It follows by applying theorem 7 to each multiplication and addition in the Ruffini–Horner rule. □

Despite its numerical stability the Ruffini–Horner rule may suffer from overflow problems. This situation occurs, for instance, in the case where a polynomial having positive coefficients and large degree is computed at a point ξ where $|\xi| > 1$. For this reason in our implementation we apply the Ruffini–Horner rule, as described above, only in the case where $|\xi| \le 1$. If $|\xi| > 1$ we compute $p_R(\gamma)/p'_R(\gamma)$, with the customary Ruffini–Horner rule, where $\gamma = 1/\xi$ and $p_R(x) = \sum_{i=0}^{n} a_i x^{n-i}$ is the reverse polynomial of p. Then we apply the following formula, that can be easily proved by direct inspection,

$$p(\xi)/p'(\xi) = \frac{1}{n\gamma - \gamma^2 p'_R(\gamma)/p_R(\gamma)} \tag{13}$$

in order to recover the sought value of the Newton correction.

4. Stop condition and a-posteriori error bound

Stop conditions for iterative methods have been proposed by several authors [2, 25, 32, 45]. Here we observe that theorem 8 gives us as a by-product a cheap and efficient stop criterion which is somehow similar to the one described in [45] for real polynomials. In fact, from theorem 8 we readily deduce that

$$\tilde{p}(\xi) - p(\xi) = s(\xi), \quad s(x) = \sum_{i=0}^{n} a_i\varepsilon_i x^i, \tag{14}$$

whence we obtain the following upper bound to the relative error

$$\left|\frac{\tilde{p}(\xi) - p(\xi)}{\tilde{p}(\xi)}\right| < \mu\frac{\tilde{s}(|\xi|)}{|\tilde{p}(\xi)|}, \quad \tilde{s}(x) = \sum_{i=0}^{n} |a_i|(4i+1)x^i.$$

The above relation implies that if $|\tilde{p}(\xi)| > \mu\tilde{s}(|\xi|)$ then the computed value is affected by a relative error less than 1 and therefore it is still reliable (actually the

condition $|\widetilde{p}(\xi)| > 2\mu\widetilde{s}(|\xi|)$ would imply at least one bit of guaranteed information). Thus we obtain the following necessary stop (unreliability) condition

$$|\widetilde{p}(\xi)| \leqslant \mu\widetilde{s}(|\xi|), \tag{15}$$

where both the left and right hand sides are computable. Moreover the left hand side is the value actually computed in floating point arithmetic, while the right hand side coincides with the actually computed value up to within $O(\mu^2)$ terms.

It is a simple matter to deduce from the results of [33] the following

Theorem 9. Let $\xi \in \mathbb{C}$ be such that the stop condition (15) holds, then there exists a "nearby" polynomial $\widehat{p}(x) = \sum_{i=0}^{n} \widehat{a}_i x^i$ such that $\widehat{p}(\xi) = 0$ and $\widehat{a}_i = a_i(1 + \delta_i)$, $|\delta_i| < 2(1 + 4i)\mu$.

Equation (14) can be used for deriving an efficient implementation of an a-posteriori upper bound to the approximation error (for similar bounds we refer the reader to [6–8, 13, 20, 42]). We recall that the disk of center ξ and radius $r(\xi) = |np(\xi)/p'(\xi)|$ contains a zero of $p(x)$ (see [22]). Therefore if ξ is an approximation to a zero then $r(\xi)/(|\xi| - r(\xi))$ constitutes a bound to the relative approximation error, provided that $|\xi| > r(\xi)$, i.e., the disk which contains the zero does not intersect the origin of the complex plane. Indeed this result could be effective if the values of $p(\xi)$ and $p'(\xi)$ were computed with no rounding errors. In actual computations the value of $p(\xi)$ obtained in floating point arithmetic may be affected by large cancellation errors (since ξ is close to a zero), thus making meaningless the criterion itself.

However, in the light of (14) we may compute a guaranteed upper bound to $|p(\xi)|$, i.e., $|p(\xi)| \leqslant |\mathrm{fl}(p(\xi))| + \mu\widetilde{s}(|\xi|)$ and a guaranteed lower bound to $|p'(\xi)|$, i.e., $|p'(\xi)| \geqslant ||\mathrm{fl}(p'(\xi))| - \mu\widetilde{s}'(|\xi|)|$ which lead to the following a-posteriori bound $\widetilde{r}(\xi)$ to the absolute error $|\xi - z|$ of the approximation of ξ to the zero α

$$|\xi - \alpha| \leqslant \widetilde{r}(x) = \frac{n(|\mathrm{fl}(p(\xi))| + \mu\widetilde{s}(\xi))}{||\mathrm{fl}(p'(\xi))| - \mu\widetilde{s}'(\xi)|}. \tag{16}$$

Observe that, if $|\xi| > \widetilde{r}(\xi)$ then $\widetilde{r}(\xi)/(|\xi| - \widetilde{r}(\xi))$ is an upper bound on the relative error $|\xi - z|/|z|$.

It is worth pointing out that both the stop condition (15) and the guaranteed error bound (16) are obtained by assuming the maximum rounding error in each arithmetic operation. Therefore it may happen that the computed value of $p(x)$ is still reliable even if (15) is satisfied, and that the actual approximation error is much less than its bound (16). In our implementation of the algorithm we adopted the simple bound $\widetilde{r}(\xi) = n(|\mathrm{fl}(p(\xi))| + \widetilde{s}(\xi))/|\mathrm{fl}(p'(\xi))|$.

5. A Fortran implementation, results of numerical tests

The algorithm has been implemented in Fortran 77 with the following features.

The starting points are selected according to (11) where to the parameter σ has been given the value 0.7 and where the upper envelope of the convex hull computation is performed with a divide-and-conquer algorithm having an $O(n \log n)$ cost. The Aberth iteration has been implemented in a Gauss–Seidel style, i.e., the updated components are immediately used inside the current iteration, according to the formula

$$x_i^{(k+1)} = x_i^{(k)} - \frac{N(x_i^{(k)})}{1 - N(x_i^{(k)})A(x_i^{(k)})},$$

$$N(x_i^{(k)}) = \frac{p(x_i^{(k)})}{p'(x_i^{(k)})},$$

$$A(x_i^{(k)}) = \sum_{j=1}^{i-1} \frac{1}{x_i^{(k)} - x_j^{(k+1)}} + \sum_{j=i+1}^{n} \frac{1}{x_i^{(k)} - x_j^{(k)}}, \quad i = 1, \ldots, n.$$

The updating of the ith components is not performed if condition (15) is satisfied for $x = x_i^{(k)}$.

The algorithm stops if either condition (15) is satisfied for $x = x_i^{(k)}$, $i = 1, \ldots, n$, or if the number of iterations exceeds a fixed value NITMAX (in the driver program we set NITMAX = 30).

The arithmetic cost of each iteration performed on all the n components is $24n^2 + O(1)$ real arithmetic operations (ops). In fact, $16n + O(1)$ ops are needed for the Newton correction, and $8n + O(1)$ ops for the Aberth correction on a single component.

The space required by this program is $2n + 1$ complex*16 numbers needed for storing the input coefficients of the polynomials and the approximations to the zeros; n real*8 numbers needed to store the relative errors; $2n + 2$ real*8 numbers and n bytes for storing the components of three auxiliary vectors.

Our program has been tested with several classes of polynomials and compared with the public domain software CZERO of the library NAPACK and CPZERO of the library SLATEC, and with the routine PA16AD of Harwell based on Madsen–Reid's algorithm [30], on a Sparc workstation with standard IEEE arithmetic.

We considered polynomials having random coefficients, and classes of specific polynomials.

Concerning random polynomials, we generated 100 polynomials having complex coefficients with real and imaginary parts between -1 and 1, for each of the following values of the degree n: 10, 20, 50, 100, 200. Moreover we considered a single random polynomial for each of the following values of the degree n: 400, 800, 1,600, 3,200, 6,400, 12,800, 25,600. In all the cases the computation has been carried out successfully by our program, whereas the other routines have failed in some cases (this is denoted by a "F" in the tables below). The number of iterations has been at most 17 and its average value has been 11.1. Table 1 reports, for each value of n, the maximum and the average time (t_m, t_a), the maximum and the average number of iterations (it_m, it_a), the ratios R_H, R_N, R_S between the average time needed by

Table 1
Random polynomials.

n	t_m	t_a	it_m	it_a	R_H	R_N	R_S
10	0.02	0.01	12	7.04	1.49	2.08	3.33
20	0.06	0.05	10	7.84	1.23	1.72	5.48
50	0.29	0.24	13	9.46	1.01	1.33	F
100	1.03	0.86	15	10.1	0.91	1.27	F
200	3.68	3.37	16	11.0	0.88	1.18	F

Table 2
Random polynomials of large degree.

n	it	t	R_H	R_N
400	11	14.7	0.83	1.04
800	11	56.1	0.85	1.10
1,600	13	234	F	1.14
3,200	16	918	F	1.18
6,400	15	3,616	F	1.25
12,800	16	14,883	F	1.37
25,600	17	59,934	F	1.33

the programs of Harwell, Napack, Slatec, respectively, and the average time needed by our program. An "F" denotes the failure of the algorithm.

Table 2 reports the value of n, the number it of iterations and the time t needed by our program, together with the ratios R_H, R_N between the time needed by the programs of Harwell and Napack, respectively, and the time needed by our program for a single random polynomial of degree n.

For small degrees our program is faster than the ones of Harwell, Napack and Slatec, for higher degrees the Harwell routine is slightly faster but it fails for polynomials of degree greater than 850 due to overflow problems. The program of Slatec fails to compute the zeros for some random polynomials of degree greater than or equal to 50.

Concerning specific polynomials, we considered several classes, the first, class A, is made up by monic polynomials having zeros with very large and very small moduli:

$(A1)$ $x^{20} + 10^{200}x^{14} + x^5 + 1,$

$(A2)$ $x^{20} + 10^{250}x^{14} + x^5 + 1,$

$(A3)$ $x^{20} + 10^{300}x^{14} + x^5 + 1,$

$(A4)$ $x^{20} + x^{11} + 10^{10}x + 10^{-10}.$

Class B is made up by monic polynomials having one zero whose floating point representation generates an underflow condition:

$(B1)$ $x^{20} + x^{11} + 10^{200}x + 10^{-200},$

$(B2)$ $x^{20} + x^{11} + 10^{250}x + 10^{-250},$

$(B3)$ $x^{20} + x^{11} + 10^{300}x + 10^{-300}.$

Table 3
Polynomials having small and large zeros.

	$A1$	$A2$	$A3$	$A4$	$B1$	$B2$	$B3$	$C1$	$C2$	$C3$
it	7	7	7	5	5	5	5	6	6	5
R_H	1.5	F	F	F	F	F	F	F	F	F
R_N	F	F	F	F	F	F	F	F	F	F

Class C is made up by polynomials whose normalization generates an overflow condition, moreover, the polynomial $C1$ has a zero whose real part is too large to be represented as a floating point number:

$(C1)$ $10^{-200}x^{20} + 10^{200}x^{19} + 10^{200}$,

$(C2)$ $10^{-200}x^{20} + 10^{100}x^{19} + 10^{200}$,

$(C3)$ $10^{-300}x^{20} + 10^{300}x + 1$.

Table 3 reports, for each polynomial of the classes A, B and C, the number of iterations needed by our program and the ratios R_H and R_N of the time needed by the Harwell and the Napack program, respectively, and the time needed by our routine. An "F" denotes the failure of the program. Our program has never failed, even for the polynomial $C1$ our program computes all the zeros which are representable as floating point numbers and outputs a warning message informing the user about the existence of a too large zero.

We considered also polynomials of the following classes:

(i) $(x+1)^n - (1-\mathrm{i})x^n$ $(\mathrm{i}^2 = -1)$, $n = 10, 15, 20, 25, 30, 35, 40$.

(ii) $(x+1)^n - x^n + 0.125\mathrm{i}$, $n = 10, 15, 20, 25, 30, 35, 40$.

(iii) Modified Legendre polynomials of degree n where the leading coefficient is multiplied by i, $n = 10, 20, 30$.

(iv) $\mathrm{i} + 2x + 3x^2 + \cdots + (n+1)x^n$, $n = 10, 20, 30, \ldots, 80$.

(v) $\mathrm{i}x^n - (ax-1)^m$, $\begin{cases} n = 10, 20, \ m = 2, 4, 6, \ a = 100, \\ n = 40, \ m = 10, \ a = 100. \end{cases}$

(vi) $(x+1)^2(x+\mathrm{i})^3(x^{15} + x^7 + 1)$.

(vii) $x^n - \mathrm{i}$, $n = 10, 20, 40, 80$.

The polynomial (i) belongs to the class $\mathcal{P}((x+1)^n)$ introduced in section 2; its zeros can be explicitly expressed in terms of the nth roots of $-\mathrm{i}$. The polynomial (ii) represents a perturbation of a polynomial in the class $\mathcal{P}((x+1)^n)$. Legendre polynomials have coefficients growing exponentially with the degree. The polynomials (v), which are a slight modification of Mignotte's polynomials, have a cluster of m zeros around a^{-1} of radius roughly $a^{-n/m-1}$. For $n = 20$, $m = 2$, $a = 100$, the two zeros of the cluster cannot be separated in the floating point arithmetic.

Tables 4–8 report, together with the values of n, the number it of iterations needed by our program and the ratios R_H, R_N, R_S of the time t required by the

Table 4
Polynomials in the classes (i) and (ii).

n	it	R_H	R_N	R_S	it	R_H	R_N	R_S
10	8	1.14	1.92	5.57	9	1.16	1.81	4.93
15	10	0.96	1.96	7.78	8	1.19	1.89	8.93
20	11	0.85	1.30	12.8	12	0.96	1.37	11.3
25	13	0.77	1.20	F	11	1.02	1.41	2.40
30	16	0.83	0.91	F	11	0.93	1.23	F

Table 5
Modified Legendre polynomials.

n	it	R_H	R_N	R_S
10	7	1.27	2.08	4.80
20	7	1.14	1.79	10.3
30	9	0.8	1.37	10.6

Table 6
Polynomials in the class (iv).

n	it	R_H	R_N	R_S
10	8	1.41	1.92	5.15
20	7	1.49	2.08	6.80
30	9	1.15	1.56	6.40
40	8	1.15	1.59	18.1
50	9	1.09	1.49	F
60	10	1.01	1.47	F
70	8	1.06	1.52	F
80	8	1.1	1.45	F

Table 7
Polynomials in the class (v).

n	m	it	R_H	R_N	R_S
10	2	14	1.19	1.92	3.00
10	4	15	0.83	1.23	3.05
10	6	13	0.71	1.23	6.60
20	2	15	1.45	2.08	5.74
20	4	16	1.19	1.89	5.26
20	6	15	0.71	1.20	3.87
40	10	16	1.10	1.20	6.79

Table 8
Polynomials in the class (vii).

n	it	R_H	R_N	R_S
10	4	2.00	3.00	2.50
20	4	1.67	2.30	5.00
40	5	1.67	2.12	7.13
80	5	6.24	1.92	10.2
160	5	F	2.5	20.67

programs of Harwell, Napack and Slatec, respectively, and the time required by our program.

For the polynomial (vi) with multiple zeros we have it = 17, $R_H = 1.04$, $R_N = 1.35$, $R_S = 3.33$.

6. Notes on the Fortran 77 code

The main subroutine for the computation of polynomial roots is called in the following way: CALL POLZEROS (N, POLY, EPS, BIG, SMALL, NITMAX, ROOT, RADIUS, ERR, ITER, APOLY, APOLYR), where N (integer) is the degree of the polynomial; POLY (complex*16, dimension N + 1, input) is the coefficient vector, i.e., $p(x) = \sum_{i=0}^{N} \mathrm{POLY}(i+1)x^i$. The parameters EPS, SMALL and BIG are related to the floating point arithmetic used by the computer, more precisely, EPS is the machine precision, SMALL is the smallest positive real*8, BIG is the largest real*8. For instance, for machines using the IEEE arithmetic like PCs, we have EPS = 2.d0**−53, SMALL = 2.d0**−1074, BIG = 2.d0**1023. The parameter NITMAX is the maximum number of iterations allowed in Aberth's method, in the driver program it is set equal to 30. ROOT (complex*16, dimension N, output) is the vector of the approximations of the roots; RADIUS (real*8, dimension N, output) is the vector of the absolute error bounds such that the disk of center ROOT(i) and radius RADIUS(i) contains a root of $p(x)$, $i = 1, \ldots, N$; ERR (logical, dimension N, output) gives information on the unreliability of the roots, more precisely, ERR(j) = .TRUE. if after NITMAX iterations the stop condition (15) is not satisfied for ξ = ROOT(j), ERR(j) = .FALSE., otherwise; ITER (integer, output) is the number of iterations performed by the algorithm; APOLY and APOLYR (real*8, dimension N + 1, auxiliary) are two auxiliary vectors used to store the coefficients of the polynomial $s(x)$ of (4) and of the analogous polynomial obtained by performing a backward error analysis in the computation of $p_R(x)$ of (13).

In the subroutine POLZEROS there are calls to the following subroutines:

ABERTH, computation of the Aberth correction;

NEWTON, computation of $p(x)/p'(x)$, of the inclusion radius and test of the stop condition;

START, computation of the starting approximations according to the method of section 2;

CNVEX, computation of the upper envelope of the convex hull of $\{(i, \log|a_i|), i = 0, \ldots, n\}$ by means of a divide-and-conquer technique;

CMERGE, LEFT, RIGHT, CTEST, auxiliary subroutines needed only by CNVEX.

A driver program is also provided. This program sets the parameters EPS, SMALL, BIG, NITMAX, MAXDEG, where the latter is the maximum degree, and reads from the standard input the file name of the input data. This file must contain on each line the degree N, the real part and the imaginary part of the coefficients

POLY(1), ..., POLY($N+1$) of $p(x)$. After the call to POLZEROS the driver program writes on the standard output and on the file <input file name>.out the values of

$$I, \quad \text{ERR}(I), \quad \text{ROOT}(I), \quad \text{RADIUS}(I), \quad \text{RADIUS}(I)/\text{ABS}(\text{ROOT}(I)),$$

for $I = 1, \ldots, N$.

7. Towards a multiprecision implementation

We have implemented our algorithm by using multiprecision floating point arithmetic in MATHEMATICA^{TM} according to the following lines. Here we just give an outline of our implementation and refer the reader for more details to [5].

The algorithm takes as input an error bound $\varepsilon > 0$, the degree n and the complex coefficients of the polynomial $p(x)$, whose real and imaginary parts can be integer, rational or floating point numbers. The algorithm delivers n disks of centers c_i and radii r_i, $i = 1, \ldots, n$, such that each disk contains a zero of $p(x)$, the union of the disks contains all the zeros, and one of the two following conditions holds:

(a) (cheap option) $r_i \leqslant \varepsilon|c_i|$, for all the values of i corresponding to *nonisolated* disks.

(b) (strong option) $r_i \leqslant \varepsilon|c_i|$, for $i = 1, \ldots, n$.

The algorithm dynamically adjusts the precision of the floating point arithmetic (precision of computation), according to the numerical conditioning of each zero. In this way the high precision is used only for ill-conditioned zeros.

In order to avoid linear convergence to multiple zeros or to clusters, the Rouché based criterion of section 2 is applied in a suitable way to the polynomial obtained by shifting $p(x)$ to the center of gravity of the cluster. The center of gravity of each cluster is approximated by using the superlinear convergence of the means of the components generated by the Durand–Kerner method and converging to the zeros of the cluster [15].

The algorithm proceeds at different levels of working precision. Once a working precision has been fixed, the computation is carried out until the computed approximations are the zeros of a "nearby" polynomial. This situation occurs when the computed value of $p(x)$ at the approximation ξ is not reliable with respect to the working precision, i.e., (15) holds.

The computation is performed with a higher working precision (double number of digits) only for those approximations which do not satisfy conditions (a) or (b). If a cluster has been detected, i.e., a set of $m > 1$ overlapping disks of centers c_i and radii r_i, then the gravity center g of the cluster is approximated with few iterations of Durand–Kerner's method. Then the Rouché based criterion is applied to the polynomial $q(x) = p(x+g)$ in order to find m new starting approximations to the zeros of the cluster.

The resulting algorithm is outlined in the following steps:

1. The number of digits of the working precision is set to the standard machine value; a set of starting approximations is computed by the Rouché based criterion applied to the polynomial obtained by rounding the coefficients of $p(x)$ to d-digits.
2. Aberth's method is applied selectively only to the components which are still "far" from the sought zeros. The computation in the ith component is stopped if the computed value of $p(x)$ is not reliable in the sense of (15). A set of n disks is output. Each disk contains a zero, the union of all the disks contains all the zeros.
3. The isolated disks are added to the output list (cheap option); the connected components made up by more than one disk are considered and the gravity center g_j of the zeros belonging to the jth component is approximated by means of Durand–Kerner's algorithm.
4. For each j the coefficients of $q_j(x) = p(x + g_j)$ are computed and the Rouché based criterion is applied to $q_j(x)$ to find better starting approximations. The shift is made with a higher working precision depending on the multiplicity m_j of the cluster, whereas the criterion for selecting new starting approximations is applied with the standard machine precision.
5. If for the jth cluster all the new m_j approximations have a distance from their gravity center g_j lower than $|g_j|\varepsilon$, then m_j copies of the disk of center g_j and radius $\varepsilon|g_j|$ are added to the output list.
6. If the output list is not complete then the number d of the digits of the working precision is doubled and the computation is recursively applied from stage 2.

In the numerical tests that we have performed our algorithm has shown a superlinear convergence independently of the multiplicity of the clusters and of the zeros. Here we just report a few results of the many tests that we have performed with a set of input polynomials suggested by John Abbot.

The polynomial $(x - 3c^2)^2 + icx^7$, for $c = 10^{-6}$ has a cluster of two zeros having imaginary parts with very small moduli (about 3.3×10^{-44}) and real parts with small moduli (about 10^{-12}) which match in the first 31 digits. The condition number of this pair of zeros is roughly 10^{55} so that the working precision needed for separating the two zeros must have at least 86 digits. Our method performed 13 iterations in the standard machine precision (roughly 15 digits), and delivered 5 out of 7 isolated disks. With the working precision of 30 digits no iteration has been performed since the computed value of $p(x)$ was unreliable right from the start. The same situation occurred with the working precision of 60 digits. Only 5 iterations (on two components) were sufficient to compute the two zeros in the cluster, with the precision of about 70 digits, by using a working precision of 120 digits.

The polynomial $x^{14} + 2 \times 10^{24}x^{11} + 10^{48}x^8 + 4x^7 - 4 \times 10^{24}x^4 + 4$, has a cluster of 4 complex zeros with real and imaginary parts matching in the first 27 digits, having big moduli (about 10^8); a cluster of two real zeros matching in the first 29 digits, with

big moduli (about 10^8); a cluster of two zeros having small moduli (about 10^{-6}) and very small imaginary parts (about 10^{-28}); a cluster of two real zeros with small moduli (about 10^{-6}) matching in the first 21 digits; finally a cluster of 4 zeros with small moduli (about 10^{-6}), with very small real parts (about 10^{-28}) and imaginary parts matching in the first 22 digits. After 15 iterations in the standard machine precision no isolated disks were found by our algorithm. No iterations were performed with the working precision of 30 digits, while after 5 iterations with the working precision of 60 digits 8 zeros out of 14 were found. With the working precision of 120 digits the remaining 6 zeros were computed in 5 iterations (on 6 components).

The polynomial $(x-1)^2(x-2)^5(x^{10}-(1000x+1)^2)$ has a zero of multiplicity 2, a zero of multiplicity 5, a cluster of two zeros having 12 common digits, and 8 well conditioned zeros. We applied our method in the cheap option (a) with $\varepsilon = 10^{-100}$, i.e., about 100 digits of the nonisolated zeros must be computed. The well conditioned zeros were computed after 15 iterations in the standard machine precision. No iterations were performed with the working precision of 30 digits. Only 3 iterations were sufficient to isolate the two zeros in the cluster with a working precision of 60 digits. The double root has been computed, within the relative precision ε, as center of gravity of the cluster with *no Aberth's iterations* and with 120 digits of working precision. The root of multiplicity 5 is computed at the next stage, i.e., with the working precision of 240 digits, as the gravity center of the cluster again with no Aberth's iterations.

References

[1] O. Aberth, Iteration methods for finding all zeros of a polynomial simultaneously, Math. Comp. 27(122) (1973) 339–344.

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

[3] G. Alefeld and J. Herzberger, On the convergence speed of some algorithms for the simultaneous approximation of polynomial roots, SIAM J. Numer. Anal. 11(2) (1974) 237–243.

[4] M. Ben-Or and P. Tiwari, Simple algorithms for approximating all roots of a polynomial with real roots, J. Complexity 6 (1990) 417–442.

[5] D. Bini and G. Fiorentino, A multiprecision implementation of a poly-algorithm for univariate polynomial zeros, in: *Proc. of the POSSO Workshop on Software*, eds. J. C. Faugère, J. Marchand and R. Rioboo (Paris, 1995).

[6] W. Börsch-Supan, A-posteriori error bounds for the zeros of polynomials, Numer. Math. 5 (1963) 380–398.

[7] D. Braess and K. P. Hadeler, Simultaneous inclusion of the zeros of a polynomial, Numer. Math. 21 (1973) 161–165.

[8] C. Carstensen, Inclusion of the roots of a polynomial based on Gerschgorin's theorem, Numer. Math. 59 (1991) 349–360.

[9] M. Cosnard and P. Fraigniaud, Asynchronous Durand–Kerner and Aberth polynomial root finding methods on a distributed memory multicomputer, Parallel Computing 89 (1990) 79–84.

[10] D. K. Dunaway, Calculation of zeros of a real polynomial through factorization using Euclid's algorithm, SIAM J. Numer. Anal. 11(6) (1974) 1087–1104.

[11] E. Durand, *Solutions Numériques des Équations Algébriques, Tome I: Equations du Type* $F(X)=0$; *Racines d'un Polynôme* (Masson, Paris 1960).

[12] L. W. Ehrlich, A modified Newton method for polynomials, Comm. ACM 10(2) (1967) 107–108.

[13] L. Elsener, A remark on simultaneous inclusions of the zeros of a polynomial by Gerschgorin's theorem, Numer. Math. 21 (1973) 425–427.
[14] P. Fraigniaud, Analytic and asynchronous root finding methods on a distributed memory multicomputer, Research Report LIP-IMAG (1989).
[15] P. Fraigniaud, The Durand–Kerner's polynomials root-finding method in case of multiple roots, BIT 31 (1991) 112–123.
[16] I. Gargantini and P. Henrici, Circular arithmetic and the determination of polynomial zeros, Numer. Math. 18 (1972) 305–320.
[17] W. Gautschi, Questions of numerical condition related to polynomials, in: *Recent Advances in Numerical Analysis*, eds. C. de Boor and G. H. Golub (Academic Press, New York, 1978) pp. 45–72.
[18] M. W. Green, A. J. Korsak and M. C. Pease, Simultaneous iteration towards all roots of a complex polynomial, SIAM Rev. 18 (1976) 501–502.
[19] H. Guggenheimer, Initial approximations in Durand–Kerner's root finding method, BIT 26 (1986) 537–539.
[20] M. Gutknecht, A-posteriori error bounds for the zeros of a polynomial, Numer. Math. 20 (1972) 139–148.
[21] E. Hansen, M. Patrick and J. Rusnak, Some modifications of Laguerre's method, BIT 17 (1977) 409–417.
[22] P. Henrici, *Applied and Computational Complex Analysis*, Vol. 1 (Wiley, 1974).
[23] A. S. Householder, Generalization of an algorithm of Sebastião e Silva, Numer. Math. 16 (1971) 375–382.
[24] A. S. Householder, *The Numerical Treatment of a Single Nonlinear Equation* (McGraw-Hill, Boston 1970).
[25] M. Igarashi, A termination criterion for iterative methods used to find the zeros of polynomials, Math. Comp. 42 (1984) 165–171.
[26] M. A. Jenkins and J. F. Traub, A three stage variable shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration, Numer. Math. 14 (1970) 252–263.
[27] I. O. Kerner, Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen, Numer. Math. 8 (1966) 290–294.
[28] N. Kjurkchev and K. Mahdi, Some remarks on Dvorcuk's root-finding method, BIT 34 (1994) 318–322.
[29] D. H. Lehmer, A machine method for solving polynomial equations, J. ACM 8 (1961) 151–162.
[30] K. Madsen and K. Reid, Fortran subroutines for finding polynomial zeros, Report HL 75/1172(C.13), Computer Science and Systems Divisions, A.E.R.E. Harwell, Oxford (1975).
[31] J. M. McNamee, A bibliography on roots of polynomials, J. Comput. Appl. Math. 47 (1993) 391–394.
[32] J. M. McNamee, A comparison of methods for terminating polynomial iterations, J. Comput. Appl. Math. 21 (1988) 239–244.
[33] R. G. Moiser, Root neighborhoods of a polynomial, Math. Comp. 47 (1986) 265–273.
[34] C. A. Neff, Specified precision polynomial root isolation is in NC, in: *Proc. 31st Annual IEEE Symp. on Foundation of Computer Science* (IEEE Computer Science Press, 1990) pp. 152–162.
[35] A. Ostrowski, On a theorem by J. L. Walsh concerning the moduli of roots of algebraic equations, Bull. Amer. Math. Soc. 47 (1941) 742–746.
[36] V. Pan, On approximating complex polynomial zeros: modified quadtree (Weyl's) construction and improved Newton's iteration, in: *5th Annual ACM–SIAM Symposium on Discrete Algorithms*, Arlington, VA (1994).
[37] V. Pan, Sequential and parallel complexity of approximate evaluation of polynomial zeros, Comput. Math. Appl. 14(8) (1987) 591–622.
[38] L. Pasquini and D. Trigiante, A globally convergent method for simultaneously finding polynomial roots, Math. Comp. 44(169) (1985) 135–149.

[39] A. Schönhage, The fundamental theorem of algebra in terms of computational complexity, Technical Report, Mathematisches Institut der Universität Tübingen (1982).
[40] Yu. V. Sidorov, M. V. Fedoryuk and M. I. Shabunin, *Lectures on the Theory of Functions of a Complex Variable* (Mir, Moscow, 1985).
[41] S. Smale, The fundamental theorem of algebra and complexity theory, Bull. Amer. Math. Soc. 4(1) (1981) 1–36.
[42] B. T. Smith, Error bounds for the zeros of a polynomial based upon Gerschgorin's theorem, J. ACM 17 (1970) 661–674.
[43] G. W. Stewart, On the convergence of Sebastião e Silva's method for finding a zero of a polynomial, Math. Comp. 12 (1970) 458–460.
[44] W. Werner, On the simultaneous determination of polynomial roots, in: *Lecture Notes in Mathematics 953* (Springer, Berlin, 1982) pp. 188–202.
[45] J. H. Wilkinson, Practical problems arising in the solution of polynomial equations, J. Inst. Math. Appl. 8 (1971) 16–35.