Solving a Polynomial Equation: Some History and Recent Progress

Victor Y. Pan

SIAM Review, Vol. 39, No. 2 (Jun., 1997), 187-220.

Stable URL:
http://links.jstor.org/sici?sici=0036-1445%28199706%2939%3A2%3C187%3ASAPESH%3E2.0.CO%3B2-M

SIAM Review is currently published by Society for Industrial and Applied Mathematics.

SIAM REV.
Vol. 39, No. 2, pp. 187–220, June 1997

SOLVING A POLYNOMIAL EQUATION: SOME HISTORY AND RECENT PROGRESS*

VICTOR Y. PAN†

Abstract. The classical problem of solving an nth degree polynomial equation has substantially influenced the development of mathematics throughout the centuries and still has several important applications to the theory and practice of present-day computing. We briefly recall the history of the algorithmic approach to this problem and then review some successful solution algorithms. We end by outlining some algorithms of 1995 that solve this problem at a surprisingly low computational cost.

Key words. polynomial equation, fundamental theorem of algebra, complex polynomial zeros, numerical approximation, computer algebra, Weyl's quadtree algorithm, divide-and-conquer algorithms

AMS subject classifications. 65H05, 68Q40, 68Q25, 30C15

PII. S0036144595288554

1. Introduction.

The problem of solving a polynomial equation

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n = 0 \tag{1.1}$$

was known to the Sumerians (third millennium B.C.) and has deeply influenced the development of mathematics throughout the centuries (cf. [Be40], [Bo68], [Ne57], [Ev83]).

In particular, the very ideas of abstract thinking and using mathematical notation are largely due to the study of this problem. Furthermore, this study has historically motivated the introduction of some fundamental concepts of mathematics (such as irrational and complex numbers, algebraic groups, fields, and ideals) and has substantially influenced the earlier development of numerical computing.

Presently, the study of equation (1.1) does not play such a central role in mathematics and computational mathematics. In particular, many computational problems arising in the sciences, engineering, business management, and statistics have been linearized and then solved by using tools from linear algebra, linear programming, and fast Fourier transform (FFT). Such tools may involve the solution of (1.1) but usually for smaller n, where the available subroutines are sufficiently effective in most cases. In fact, as n grows beyond 10 or 20, the present-day practical needs for solving equation (1.1) become more and more sparse, with one major exception: equation (1.1) retains its major role (both as a research problem and a part of practical computational tasks) in the highly important area of computing called *computer algebra*, which is widely applied to algebraic optimization and algebraic geometry computations. In computer algebra applications, one usually needs to solve (1.1) for larger n (typically well above 100 and sometimes of order of several thousands). Furthermore, high (multiple) precision of hundreds (or even thousands) of bits is frequently required for the representation of the coefficients $p_0, p_1, \ldots, p_n$ and/or the solution values x. In these cases, the solution of (1.1) causes problems for the available software, and

*Received by the editors June 30, 1995; accepted for publication (in revised form) May 22, 1996. http://www.siam.org/journals/sirev/39-2/28855.html

†Department of Mathematics and Computer Science, Lehman College, City University of New York, Bronx, NY 10468 (vpan@lcvax.lehman.cuny.edu). The research of this author was supported in part by NSF grant CCR 9020690 and PSC CUNY awards 665301 and 666327.

this motivates further research on the design of effective algorithms for solving (1.1). Such a task and its technical ties with various areas of mathematics keep attracting the substantial effort and interest of researchers so that several new algorithms for solving (1.1) continue to appear every year [MN93]. We will next review the history of the subject (starting with older times and ending with recent important progress) and some samples of further extensions. We had to be selective in these vast subject areas; we have chosen to focus on some recent promising approaches and leave some pointers to the abundant bibliography. The reader may find further material on the latter approaches in [BP,a] and more pointers to the bibliography in [MN93].

2. Some earlier history of solving a polynomial equation. A major step in the history of studying polynomial equations was apparently in stating the problem in the general abstract form (1.1). This step took a millennia of effort and led to the introduction of the modern mathematical formalism. Meanwhile, starting with the Sumerian and Babylonian times, the study focused on smaller degree equations for specific coefficients. The solution of specific quadratic equations by the Babylonians (about 2000 B.C.) and the Egyptians (found in the Rhind or Ahmes papyrus of the second millennium B.C.) corresponds to the use of our high school formula

$$x_{1,2} = (-p_1 \pm \sqrt{p_1^2 - 4p_0p_2})/(2p_2). \tag{2.1}$$

A full understanding of this solution formula, however, required the introduction of negative, irrational, and complex numbers, and the progress of mankind in this direction is a separate interesting subject, closely related indeed to the history of solving polynomial equations of small degrees [Be40], [Bo68], [Ev83]. An important achievement in this area was the formal rigorous proof by the Pythagoreans (about 500 B.C. in ancient Greece) that the equation $x^2 = 2$ has no rational solution, that is, that its solution must use a radical and not only arithmetic operations.

The attempts to find solution formulae which, like (2.1), would involve only arithmetic operations and radicals, succeeded in the 16th century for polynomials of degrees 3 and 4 (Scipione del Ferro, Nicolo Tartaglia, Ludovico Ferrari, Geronimo Cardano), but a very profound influence on mathematics was made by the failure of all attempts to find such formulae for any polynomial of a degree greater than 4. More precisely, such attempts resulted in a theorem, obtained by Ruffini in 1813 and Abel in 1827, on the nonexistence of such a formula for the class of polynomials of degree n for any $n > 4$ and with the Galois fundamental theory of 1832. (In fact, Omar Khayyam, who died in 1122 a famous poet and the leading mathematician of his time, and later Leonardo of Pisa (now more commonly known as Fibonacci), who died in 1250, wrongly conjectured the nonexistence of such solution formulae for $n = 3$.) The Galois theory was motivated by the same problem of solving equation (1.1) and included the proof of the nonexistence of the solution in the form of formulae with radicals, already for simple specific polynomial equations with integer coefficients, such as $x^5 - 4x - 2 = 0$, but this theory also gave a world of major ideas and techniques (to some extent motivated by the preceding works, particularly by Lagrange and Abel) for the development of modern algebra (see [Be40] and [Bo68] for further historical background).

In spite of the absence of solution formulae in radicals, the *fundamental theorem of algebra* states that equation (1.1) always has a complex solution for any input polynomial $p(x)$ of any positive degree n. In clearer and clearer form, this theorem was successively stated by Roth (1608), Girard (1629), and Descartes (1637) and then

repeated by Rahn (1659), Newton (1685), and Maclaurin. Its proof, however, had to wait until the 19th century. Several early proofs, in particular, by D'Alembert, Euler, Lagrange, and Gauss (in his doctoral dissertation defended in 1799) had flaws, although these proofs and even flaws have motivated further important studies. In particular, in the Gauss dissertation of 1799 it was assumed as an obvious fact that every algebraic curve entering a closed complex domain must leave this domain. Proving this assumption actually involves the nontrivial study of complex algebraic curves (which is a subject having substantial impact on pure and applied mathematics). As one of the results of this study in the 19th and 20th centuries, Ostrowski fixed the flaw in the Gauss proof in 1920 (see [Ga73]).

It is easy to extend the fundamental theorem of algebra to prove the existence of the factorization

$$p(x) = p_n \prod_{j=1}^{n} (x - z_j)$$

for any polynomial $p(x)$, where $p_n \neq 0$, so that $z_1, z_2, \ldots, z_n$ are the n zeros of $p(x)$ (not necessarily all distinct) and are the only solutions to (1.1).

The subject of computing or approximating these zeros has been called *algorithmic aspects of the fundamental theorem of algebra* [DH69], [Sm81], [Schö82].

With no hope left for the exact solution formulae, the motivation came for designing iterative algorithms for the approximate solution and, consequently, for introducing several major techniques, in particular, for the study of meromorphic functions, symmetric functions, Padé tables, continued fractions, and structured matrices [Ho70]. Actually, the list of iterative algorithms proposed for approximating the solution $z_1, z_2, \ldots, z_n$ of (1.1) includes hundreds (if not thousands) of items and encompasses about four millennia. In particular, the *regula falsi* (or false position) algorithm appeared in the cited Rhind (or Ahmes) papyrus as a means of solving (1.1) for $n = 2$. (The name regula falsi was given to this algorithm in medieval Europe, where it was brought by Arab mathematicians. After its long trip in space and time, this algorithm has safely landed in the modern undergraduate texts on numerical analysis and, together with its modification called the secant method, is extensively used in computational practice.)

In fact, it is not important whether a computer solution has been obtained via formulae or not because, generally, the solution is irrational and cannot be computed exactly anyway. What matters is how to obtain a solution with a high accuracy at a lower computational cost by using less computer time and memory. From this point of view, the first two algorithms with guaranteed convergence to all the n zeros of $p(x)$ (for any input polynomial $p(x)$ of a degree n), due to Brouwer [BdL24] and Weyl [We24] and both published in 1924, were not fully satisfactory because they were presented without estimating the amount of computational resources, in particular, the computational time required for their performance.

In the next two sections, we will describe how to fill this void of Weyl's algorithm and its modifications. Then we will recall some other successful algorithms which have evolved since 1924. Then again we will restrict our exposition mostly to the subject of computing or approximating all the solutions to equation (1.1), although in many cases one seeks only some partial information about such solutions. For instance, one may try to obtain one of the solutions, all the real solutions, or all the solutions lying in a fixed disc or square on the complex plane. In some cases, one may need to know only if there exists any real solution or any solution in a fixed disc or square on the

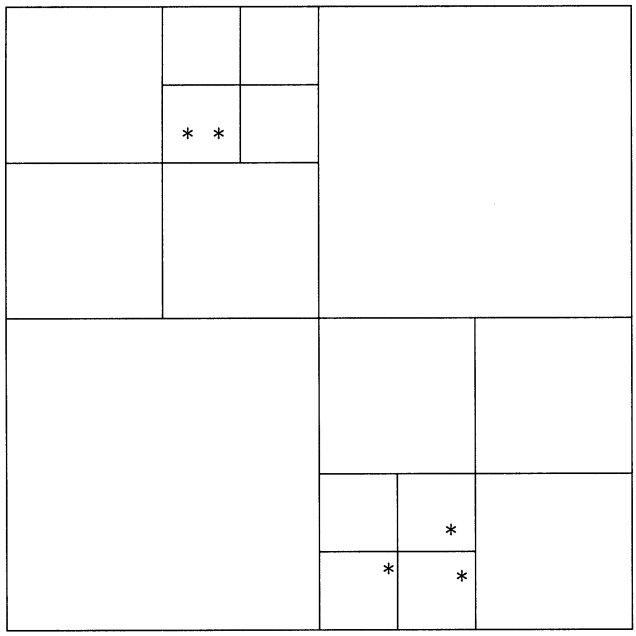

FIG. 1. *Weyl's algorithm partitions each suspect square into four congruent subsquares. The five zeros of $p(x)$ are marked by asterisks.*

complex plane or one may just need to count the number of such solutions. We refer the reader to [BP,a] on the latter subjects.

3. Weyl's geometric construction. The subject of computational complexity had not arisen yet in 1924, but, in fact, Weyl's algorithm can be implemented to perform it at a rather low computational cost; moreover, its subsequent modifications in [HG69], [R87], and [P87] at the time of their appearance implied new record upper bounds on the computational complexity of solving the polynomial equation (1.1) (cf. other effective modifications in [Wi78], [P94], and [P96a]).

Furthermore, Weyl's construction (under the name *quadtree construction*) has been successfully applied to a wide range of other important computational problems in such areas as image processing, n-body particle simulation, template matching, and the unsymmetric eigenvalue problem [Sa84], [Se94], [Gre88], [P95b].

Let us briefly outline this effective construction, which performs search and exclusion on the complex plane and can also be viewed as a *two-dimensional* version of the *bisection* of a line interval (see Figures 1 and 2). On the complex plane, the search starts with an *initial suspect square* $\mathcal{S}$ containing all the zeros of $p(x)$. As soon as we have a suspect square, we partition it into four congruent subsquares. At the center of each of them, we perform a *proximity test*; that is, we estimate the distance to the closest zero of $p(x)$. (The estimates within, say, the relative error of 40% will suffice in the context of this algorithm.) If the test guarantees that this distance exceeds half of the length of the diagonal of the square then the square cannot contain any zero of $p(x)$ and is discarded. The remaining squares are called *suspect*; each of them undergoes the same recursive process of partitioning into four congruent subsquares and of application of proximity tests at their centers. The zeros of $p(x)$ lying in each

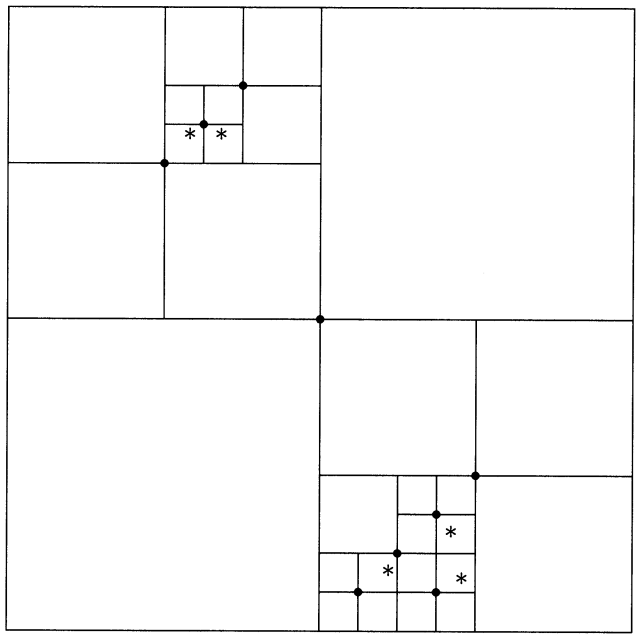

FIG. 2. *Suspect squares computed by Weyl's (quadtree) algorithm. Their centers (marked by dots) approximate the five zeros of* $p(x)$ *marked by asterisks.*

suspect square are approximated by its center with errors bounded by the half-length of its diagonal. Each iteration step decreases the length and the half-length of the diagonals of suspect squares by 50%. Therefore, in h iteration steps, the approximation errors cannot exceed $0.5\,\mathrm{diag}(\mathcal{S})/2^h$, where $\mathrm{diag}(\mathcal{S})$ denotes the length of the diagonal of the initial suspect square $\mathcal{S}$.

The entire algorithm is essentially reduced to defining an initial suspect square $\mathcal{S}$ and performing proximity tests. Furthermore, we need to apply proximity tests only at the origin since we may shift the center $\mathcal{C}$ of a suspect square into the origin by substituting the new variable $y = x - \mathcal{C}$ for the original variable x. Then $p(x)$ is replaced by the polynomial $\sum_{i=0}^{n} q_i y^i = q(y) = p(y+\mathcal{C})$, whose coefficients q_i are easily computed by using from about $9n \log_2 n$ to about $18n \log_2 n$ arithmetic operations [BP94]. Moreover, we may apply a proximity test to the reverse polynomial

$$y^n q(1/y) = q_n + q_{n-1}y + \cdots + q_0 y^n,$$

whose zeros are the reciprocals of the zeros of $q(y)$. Such an application will give us an upper bound $\mathcal{M}$ on the absolute values of all the zeros of $q(y)$ and thus will define an initial suspect square centered at the origin having four vertices defined by the expressions

$$(\pm 1 \pm \sqrt{-1})\mathcal{M}/\sqrt{2}.$$

Also vice versa, the reciprocal of an upper bound on the absolute values of the zeros of the reverse polynomial $y^n q(1/y)$ is exactly what we seek in any proximity test for $q(y)$. Before we approximate the value $\mathcal{M}$, we may try to decrease it by setting

$q(y) = p(y + \mathcal{C})$ for $\mathcal{C}$, the center of gravity of the n zeros of $p(x)$, that is, for

$$\mathcal{C} = -p_{n-1}/(np_n) = \frac{1}{n}\sum_{j=1}^{n} z_j,$$

$$p(x) = \sum_{i=0}^{n} p_i x^i = p_n \prod_{j=1}^{n} (x - z_j), \quad p_n \neq 0. \tag{3.1}$$

In this case, $q_{n-1} = 0$, and from Theorem 5.4 of [VdS70] we have

$$\mathcal{T}\sqrt{2/n} \leq \max_j |z_j - \mathcal{C}| \leq (1 + \sqrt{5})T/2 < 1.62\mathcal{T}, \quad \mathcal{T} = \max_{i \geq 1} |q_{n-i}/q_n|^{1/i}$$

if $\mathcal{C}$ is the center of gravity [VdS70]. In the general case, for any $\mathcal{C}$, we have

$$\mathcal{T}/n \leq \max_j |z_j - \mathcal{C}| < 2\mathcal{T}$$

(compare [He74, pp. 451, 452, 457]). The application of the above bounds to the reverse polynomial $y^n q(1/y)$ defines two proximity tests with output errors bounded by factors $1.62\sqrt{n/2}$ (if $q_{n-1} = 0$) and $2n$, respectively. In Appendix C, we recall Turan's proximity test, which approximates the minimum and maximum distances from any complex C to the zeros of $p(x)$ within the error factor 5 at the cost of performing order of $n \log n$ arithmetic operations. The error factors of the above tests can be decreased to $(1.62\sqrt{n/2})^{1/K}$, $(2n)^{1/K}$, and $5^{1/K}$, respectively, if the tests are applied to the polynomial that we denote $t_k(y)$ whose zeros are the Kth powers of the zeros of $q(y)$ for $K = 2^k$. The transition to such a polynomial is performed by means of k steps of the so-called Graeffe iteration of the form

$$t_0(y) = y^n q(1/y)/q_0,$$

$$t_{i+1}(y) = (-1)^n t_i(\sqrt{y})\, t_i(-\sqrt{y}), \quad i = 1, \ldots, k, \tag{3.2}$$

where, with no loss of generality, we assume that $q_0 \neq 0$, so that the polynomial $t_0(y)$ is monic. (Iteration (3.2) was discovered by Dandelin, soon thereafter was rediscovered by Lobachevsky, and, later on, by Graeffe; compare [O40], [Ho70].) It is easily observed that, for every i, ith iteration step (3.2) squares the zeros of $t_i(y)$ at the cost of performing a polynomial multiplication. To multiply two polynomials fast, we first evaluate these two polynomials at the Nth roots of 1 for a sufficiently large natural N, multiply the N computed values, and, finally, interpolate by applying FFT for the evaluation and interpolation. If $n + 1 = 2^h$ for an integer h, we may choose $N = 2^{h+1}$ and perform this computation by using at most $9n \log_2 n + 4n$ arithmetic operations [BP94]. (The latter cost bound can be further decreased to at most $4.5n \log_2 n + 2n$ based on the representation of $t_i(\sqrt{y})$ and $t_i(-\sqrt{y})$ in the form $t_{i,0}(z^2) + z t_{i,1}(z^2)$, where $t_{i,0}(z^2)$ and $z t_{i,1}(z^2)$ are the sums of the even and the odd powers of z in the expansion in z of the polynomial $t_i(z)$ for $z = \sqrt{y}$ and $z = -\sqrt{y}$, respectively.) Such cost bounds for iteration (3.2) mean that the order of $kn \log n$ arithmetic operations suffice for the desired transition to $t(y) = t_k(y)$. For our purpose of performing a proximity test with a relative error of at most 40% or even at most 10%, it suffices to choose k of order of $\log \log n$ if we apply the above test based on computing $\mathcal{T}$ or to choose $k = 2$ or $k = 5$ if we apply Turan's test. To simplify the formulae, we will ignore the relatively small factors $\log \log n$ in our subsequent estimates, even where we apply the test based on computing $\mathcal{T}$. (Note

that $\log_2 \log_2 n < 5$ for $n = 10^9$, which is a much greater value of n than one would encounter in present-day applications of (1.1).)

To complete estimating the arithmetic complexity of performing Weyl's algorithm, we should only estimate how many proximity tests are required or, equivalently, how many suspect squares are processed in h iteration steps of Weyl's algorithm. A simple observation shows that this is at most $4nh$ since every zero of $p(x)$ makes at most four squares suspect in each recursive step of Weyl's algorithm, provided that the proximity tests output the distances to the closest zeros of $p(x)$ within at most 40% error (or even within any relative error less than $\sqrt{2} - 1 = 0.41\ldots$). The latter bounds on the numbers of suspect squares grow only from 4 to 5, relative to each zero of $p(x)$, and from $4nh$ to $5nh$, relative to h steps of Weyl's construction, if we lift the upper bound on the relative errors of the proximity tests from 40% to 50% (or even to any value less than $5^{1/4} - 1$). (In fact, even fewer than $4nh$ or $5nh$ suspect squares are involved for larger n since we process at most 4^i or 5^i suspect squares at the ith iteration, respectively, and since $4^i < n$ for $i < 0.5 \log_2 n$, whereas $5^i < n$ for $i < \log_5 n$.) Therefore, order of $n^2 h \log n$ arithmetic operations suffice for approximating all the n zeros of $p(x)$ within $diam/2^h$, where $diam$ denotes the diameter of the set of all the zeros of $p(x)$.

In a practical implementation of Weyl's algorithm, the proximity tests should be modified substantially to take into account numerical problems of controlling the impact of roundoff errors (in the case of implementation in floating-point arithmetic with a fixed finite precision) and controlling the precision growth (in the case of implementation in rational arithmetic with no roundoff errors). We refer the readers to the end of Appendix C and to [BP,a] and [BP96] for some recent works on this subject, which still requires the resolution of many open problems.

To give an example of a specific practical modification, we observe that in Weyl's algorithm one does not have to compute the value $\mathcal{T}$, but it suffices to compare the ratio $|q_{n-i}/q_n|$ with the length of the half-diagonals of the tested square for all $i \geq 1$, which means a slightly simpler computation. As another possible practical simplification, one may start with the simpler (one-sided) proximity test that just computes the value $r = n\,|p(C)/p'(C)|$. Then it is known [He74] that the disc $D(C, r) = (x : |x - C| \leq r)$ contains the zero of $p(x)$. Therefore, a candidate square having its center in C and its side length less than $2r$ can be immediately identified as a suspect square without shifting the origin into the point C and applying the other cited (two-sided) proximity tests. We will need to shift to the latter (more involved) tests only if the value $2r$ exceeds the length of the sides of the candidate square. In fact, a variety of alternative proximity tests is available in [He74], and from this variety one may choose a simplified test that works for some considerable class of inputs and/or a more involved test that works for all inputs.

Remark 3.1. The reader may be interested in examining a modification of Weyl's construction, where squares are replaced by hexagons and where at most $3n$ hexagons can be suspect in each recursive step.

4. Acceleration of Weyl's algorithm. Like bisection, Weyl's algorithm converges right from the start with the linear convergence rate and allows its convergence acceleration by means of some analytic techniques. Moreover, since we solve an equation $f(x) = 0$ of a very special form, $f(x)$ being a polynomial, one may define explicit conditions that guarantee quadratic convergence right from a starting point x_0 to a zero of $f(x) = p(x)$. A nontrivial and very general sufficient condition of this kind for Newton's iteration (starting with x_0) $x_{i+1} = x_i - f(x_i)/f'(x_i)$, $i = 0, 1, \ldots$, was

given by Smale in [Sm86] (also compare [Kim88]):

$$\sup_{k>1}\left|\frac{f^{(k)}(x_0)}{k!\,f'(x_0)}\right|^{1/(k-1)} \le \frac{1}{8}\left|\frac{f'(x_0)}{f(x_0)}\right|.$$

This result holds for any analytic function (or map) $f(x)$ defined in a Banach space and has some multidimensional applications [RS92], [SS93], [SS93a], [SS93b], [SS93c], [SS93d].

In the context of Weyl's construction for solving equation (1.1), it is convenient to use assumptions of a more geometric nature stated in terms of an isolation ratio [P87] which quantitatively measures the isolation of the zeros of $p(x)$ or their clusters from each other. Namely, for a pair of concentric discs or squares on the complex plane, both containing exactly the same set of zeros of $p(x)$, let $\rho > 1$ denote the ratio of their diameters. Then we say that the internal disc or square is *ρ-isolated* or, equivalently, has *an isolation ratio of at least ρ.*

Now suppose that a fixed disc or square contains only a single zero z of $p(x)$ or a cluster Z of zeros having a small diameter and that this disc or square is ρ-isolated for $\rho = c_0 + c_1/n^d > 1$ and some nonnegative constants c_0, c_1, and d. Then we may apply some analytic techniques of Newton's iteration [R87], [P94], [P96a] or numerical integration [P87], in both cases with guaranteed quadratic convergence (right from any starting point that lies in the given disc or square) to these zero z or cluster Z of the zeros. The choice of c_0, c_1, and d varies in [R87], [P87], [P94], and [P96a]; so far the mildest restriction on ρ sufficient to ensure the quadratic convergence right from the start, that is, $\rho = 2\sqrt{2} + \sqrt{(12+\epsilon)n}$ for any positive ϵ, has been achieved in [P94] and [P96a].

Relatively straightforward modifications of Weyl's geometric process of search and exclusion, toward achieving isolation rather than approximation of the zeros of $p(x)$, have been proposed in the four papers [R87], [P87], [P94], and [P96a]. All these modifications rely on the following simple observations. Each iteration step of Weyl's process of search and exclusion defines a set of suspect squares whose edge length is by 50% smaller than it was at the previous step. Therefore, each iteration step either generates many more suspect squares than the previous iteration step does and/or partitions the union of the new suspect squares into more components than the previous iteration step does or, otherwise, substantially increases the isolation ratios of the minimal squares superscribing the components. New components appear at most $n-1$ times since the total number of components cannot exceed the number n of the zeros of $p(x)$, and each iteration step generates not more than $4n$ suspect squares, even assuming a proximity test with output errors of 40%, as we have already pointed out. It follows that in a few (actually in at most order of $\log n$) recursive steps of Weyl's process the zeros of $p(x)$ are included into two or several squares on the complex plane that are strongly isolated from each other. At this point, the recursive partition of suspect squares is replaced by a faster analytic iterative process. The latter process stops either where the zeros of $p(x)$ are approximated within a required error tolerance or where a set Z of the zeros, having a diameter Δ, is approximated closely enough, so that it is covered by a square having a diameter comparable with Δ. In the latter case, Weyl's search and exclusion process, starting with such a square as its initial suspect square, rapidly separates and isolates some zeros or their clusters in Z from each other, and then the analytic iterative process is applied again. To ensure that a combination of geometric and analytic techniques gives us a desired

modification of Weyl's construction, we need to decrease the error bound of 40% in the proximity tests; it can be shown that the decrease to 10% will suffice in the cited modifications.

In [P87], [P94], and [P96a] the entire computation was arranged so that only order of $n \log(hn)$ suspect squares had to be treated. This enables us to approximate all the n zeros of $p(x)$ within $diam/2^h$ by involving only order of $(n^2 \log n) \log(hn)$ arithmetic operations, versus order of $n^2 h \log n$ arithmetic operations needed in the previous section. Such an improvement is quite substantial in the most important case, where the zeros are sought with a high precision. (A similar result can be obtained after some refinement of the algorithm of [R87].)

As a by-product of achieving the isolation of the zeros or their clusters, the algorithms of [R87], [P87], [P94], and [P96a] also output the number of the zeros of $p(x)$ approximated within $diam/2^h$ by each output approximation point. When the zeros of $p(x)$ or their clusters are sufficiently well isolated from each other, such a number can be easily computed by using a *winding number algorithm* [R87], [He74]. Alternatively, one may apply the following fact (see [O40], [VdS70], [He74, pp. 458–462], and [Schö82] and also compare [P87], [P94], and [P96a]).

FACT 4.1. *For any fixed pair of constants c and d, the absolute values of all the n zeros of a polynomial $p(x)$ of degree n can be approximated within relative errors of at most c/n^d by using order of $(\log n)^2 n$ arithmetic operations.*

The algorithm supporting Fact 4.1 can be viewed as a *generalized proximity test* since it simultaneously approximates the absolute values of all zeros of $p(x)$, which gives us n narrow annuli containing the n zeros of $p(x)$.

Besides the above application, this algorithm is substantially used in the divide-and-conquer approach to approximating polynomial zeros (see section 9) and in a recent effective modification of Aberth's method (see [Bi,a], [BP,a]). (The origin of Aberth's method can actually be traced back to [BS63], cf. also [E67].)

5. Comparison of some effective approaches. We have focused on Weyl's approach as a good example for demonstrating some important developments in the field of solving polynomial equation (1.1) and showing some fundamental techniques. Numerous other algorithms have been developed for the same problem since 1924 (see [MN93] and the Guide on Available Mathematical Software, GAMS, accessible via anonymous FTP at http://gams.nist.gov). Most of these algorithms are effective for the "average" polynomial of a small or moderate degree but are heuristic in a global sense; that is, they do not generally converge to the zeros of $p(x)$ unless some convenient initial approximations are available, and no general recipes are provided for finding such approximations for an arbitrary input polynomial $p(x)$. Some of these algorithms have good records of practical performance as subroutines for numerical floating-point computation (with single precision) of the zeros of polynomials of small and moderately large degrees. At least three such approaches should be cited here: Jenkins and Traub's recursive algorithm, based on shifts of the variable and reversions of the polynomial [JT70], [JT72], [IMSL87]; some variations of Newton's iteration [M73], [MR75]; and Laguerre's method [HPR77], [F81], and [NAG88] (rootfinder CO2AGF), all of which first approximate a single zero z of the input polynomial $p(x)$, shift to the next input polynomial $p(x)/(x-z)$, and then recursively repeat these steps to approximate all other zeros of $p(x)$. For most of the input polynomials $p(x)$ of small and moderately large degrees, these algorithms converge in practice to the n zeros of $p(x)$, and their local convergence (near the zeros) is very fast. This does not rule out the possibility of further improvement of these algorithms and the design

of better ones for moderate degrees n. In particular, as a rule, the cited algorithms work much less effectively for polynomials $p(x)$ having multiple zeros and/or clusters of zeros.

Here is the account by Goedecker from [GO94, p. 1062] on the comparative numerical tests of Jenkins and Traub's algorithm, Laguerre's modified algorithm, and the companion (Frobenius) matrix methods (on which we will comment later in this section): "None of the methods gives acceptable results for polynomials of degrees higher than 50." And on p. 1063 Goedecker adds, "If roots of high multiplicity exist, any... method has to be used with caution." The latter conclusion does not actually apply to Weyl's approach and the divide-and-conquer algorithms, which we will describe later. We wish, however, to illustrate Goedecker's observation by the following simple example.

Example 5.1. Compare the multiple zero $z = 10/11$ of the polynomial $p(x) = (x-10/11)^n$ with the n zeros $z_j = (10/11)+2^{-h}\exp((2\pi\sqrt{-1})j/n), j = 0, 1, \dots, n-1$, of the perturbed polynomial $p^*(x) = (x-10/11)^n - 2^{-hn}$. The perturbation by 2^{-hn} causes a jump of the zero of $p(x) = (x-10/11)^n$ at a distance as large as 2^{-h}. Similar jumps (by 2^{-h}) of the multiple zero of the same polynomial $p(x) = (x-10/11)^n$ can be observed if we perturb its coefficient p_{n-i} by 2^{-hi} for $i = 0, 1, \dots, n-1$. Therefore, to be able to approximate (within 2^{-h}) even a single zero of $p(x)$, we need (in the worst case) to deal with at least hi bits in the representation of the coefficient p_{n-i} of $p(x) = (x-10/11)^n$ for $i = 0, 1, \dots, n$, that is, with a total of at least $(n+1)nh/2$ bits. It follows that the approximation (within the error bound 2^{-h}) of even a single zero of a worst-case input polynomial $p(x)$ of degree n, satisfying (3.1), requires the processing of at least $(n+1)nh/2$ bits and, therefore, the use of at least $(n+1)nh/4$ bit operations (also called *Boolean operations*), since each such operation handles at most two bits. Note that this lower bound holds even under the additional normalization assumption that $|z_j| \le 1,\ j = 1, \dots, n$.

Example 5.1 shows that the jump by factor of order n of the bit precision of computing is a more or less inevitable evil in the general-purpose subroutines for polynomial zeros, provided that such subroutines are required to treat polynomials with multiple zeros and/or clusters of zeros. It would not be appropriate to ignore such polynomials since they frequently appear in the practice of scientific computing. (Note that numerical truncation of the coefficients turns multiple zeros into clusters of zeros.) Furthermore, ill-conditioned dependence of the zeros on the coefficients also occurs for many polynomials having no multiple or clustered zeros (compare the well-known example of the polynomials $\prod_{j=1}^{n}(x-j)$, for large n, whose zeros jump dramatically in the result of smaller perturbation of the coefficients).

Poor convergence and the unreliability of the output of the otherwise successful algorithms in the case of ill-conditioned polynomial zeros motivate greater attention to the theoretical study of the problem. Example 5.1 also shows that computations require a higher precision to approximate the ill-conditioned zeros, but, on the other hand, they can be performed with a lower precision to approximate the well-conditioned zeros. This suggests that the precision of computing and the algorithms should vary depending on the condition of the zeros. Unlike many known algorithms, Weyl's construction and Bini's recent modification of Aberth's method [Bi,a] (which we have already cited and will also cite later) enable one to achieve such a variation, thus simplifying substantially the computation of the well-conditioned zeros of $p(x)$.

Weyl's approach is perfectly reliable in its global convergence property, whereas establishing (or disproving) global convergence of the other cited algorithms, except for the divide-and-conquer algorithms (for any input polynomial and with no spe-

cial information about good initial approximations to its zeros), is still an open issue (compare some partial progress regarding the study of the convergence of Newton's method and its modifications reported in [Sm81] and [Sm85]). Theoretical ground is more solid for the companion (Frobenius) matrix approach to approximating polynomial zeros (compare the earlier works [Ku69] and [P87a] and the recent ones [Go94] and [TT94]). In particular, by normalizing $p(x)$ to make it monic, with $p_n = 1$, and by applying the QR algorithm to the associated companion (Frobenius) matrix

$$\begin{bmatrix} 0 & & O & p_0 \\ 1 & \ddots & & p_1 \\ & \ddots & 0 & \vdots \\ O & & 1 & p_{n-1} \end{bmatrix},$$

one may approximate its eigenvalues, which are the zeros of $p(x)$. The recent experiments reported in [Go94] and [TT94] suggest that this approach may successfully compete with modified Laguerre's and Newton's, as well as with Jenkins–Traub's, methods for single precision numerical approximation of the zeros of polynomials of degrees $n < 50$ having no multiple or clustered zeros. The memory space requirement is a major limitation of this method, however. Indeed, the QR algorithm involves about $1.5n^2$ entries of the auxiliary matrices Q and R, which is usually prohibitive or, at least, highly undesirable for the computer algebra applications, where n is large and the output zeros and, consequently, the $1.5n^2$ entries of Q and R are required to be processed with a high (multiple) precision. Some modifications of this approach (for instance, modifications based on using the shifted power method or a modification of the LR algorithm due to [Gem,a], instead of the QR algorithm) should not lead to the latter problem, and also the cited algorithm of Jenkins and Traub, as well as Newton's and Laguerre's modified algorithms, do not have this problem, but they still do not suffice for the computer algebra applications, where n is large, high precision output is required, and clusters of the zeros is a typical phenomenon.

Quite effective and increasingly popular in this application area are the Durand–Kerner-type algorithms, which simultaneously approximate all the zeros of the input polynomial [Ma54], [D60], [Ke66], [A73], [FL77], [AS82], [Wer82], [PT85], [PeS87], [Bi,a]. Each of these algorithms represents an analytic iterative process defined by a certain recursive formula which is not related to the disposition of the zeros of $p(x)$ on the complex plane. In particular, Durand–Kerner's algorithm of [D60] and [Ke66] (also justly called the Weierstrass algorithm, cf. [W903], and sometimes Dochev's algorithm [DB64], [AS82]) amounts to a simplification of Newton's iteration for the Viète system of n equations in $z_1, \dots, z_n$, denoting the n zeros of $p(x) = \prod_{j=1}^{n}(x - z_j)$ (compare (3.1) for $p_n = 1$). Letting $z_j(l)$ denote the approximation to the zero z_j computed in l Durand–Kerner iteration steps, we arrive at the following recurrence formula (defining Durand–Kerner's iteration):

$$z_j(l+1) = z_j(l) - p(z_j(l))/\prod_{i \neq j} (z_j(l) - z_i(l)), \quad j = 1, \dots, n. \tag{5.1}$$

It is customary to choose n equally spaced points on a sufficiently large circle as the initial approximations $z_j(0), j = 1, \dots, n$, by setting, say,

$$z_j(0) = 3t^* \exp(2\pi\, j\sqrt{-1}/n), \ j = 1, \dots, n,$$

where $t^* \geq \max_j |z_j|$; for instance, we may always write $t^* = 2\max_{i<n} |p_i/p_n|^{1/(n-i)}$ (compare similar bounds in section 3).

Hereafter, we will assume that the variable x has been shifted and scaled so that

$$\max_j |z_j| \leq 1, \tag{5.2}$$

and then we may set $t^* = 1$.

Numerous experiments have shown the rapid convergence of Durand–Kerner's algorithm and its several extensions and variations under such a choice of initial approximations, although the theory still gives no adequate explanation of such wonderful behavior. Even more effective behavior has been shown in the experiments for the modification of [A73] proposed in [Bi,a] (cf. also [BP,a]), where initial approximations are chosen on several concentric circles, defined based on Fact 4.1, but again, no theoretical global convergence proof confirms these experimental results. (Some theoretical insight into such behavior can perhaps be drawn from the study of the path lifting method of [Sm85], developed in [RS92] for the linear programming problem, in [SS93], [SS93a], [SS93b], [SS93c], and [SS93d] for solving a system of polynomial equations, and in [KS94] for the univariate polynomial equation (1.1).)

It is fair to say that Weyl's accelerated algorithms of section 4 are potentially competitive with the Durand–Kerner-type approach. Weyl's algorithm itself (due to the factor h in its time estimate) is out of play in computer algebra applications where h is large, whereas the more recent accelerated versions (involving both analytic and complex geometry techniques) have not been implemented yet. The comparison, therefore, can be only preliminary and theoretical. A clear advantage of Weyl's (accelerated) approach is its robustness: it works well for any input polynomial $p(x)$ and does not lead to any numerical stability problems, unless such problems are created by incorporating a poor proximity test. Furthermore, even assuming very fast convergence of the Durand–Kerner-type algorithms, we cannot conclude that they are substantially superior to Weyl's accelerated algorithms in terms of the numbers of arithmetic operations involved. Indeed, the former algorithms use order of n^2 arithmetic operations in each iteration step (compare the recurrence formula (5.1) for Durand–Kerner's iteration), which is roughly the level of the proven worst-case upper bounds on the arithmetic cost of the latter algorithms. Unlike Durand–Kerner-type algorithms, Weyl's construction *enables us to decrease the computational cost* by roughly factor n/k in the cases where one seeks only the k zeros of $p(x)$ lying in a fixed isolated square, rather than all the n zeros of $p(x)$. Some further simplification is possible in the important case where only the real zeros of $p(x)$ are sought (cf. [PKSHZ96]).

The cited attractive features of Weyl's accelerated algorithms are shared by the algorithm based on the distinct *divide-and-conquer* techniques, recently proposed in [NR94] and then improved in [P95] and [P96]. Furthermore, the algorithms of [P95] and [P96] enable us to reach the record arithmetic cost bound of order of $(\log n)^2 n \log(hn)$ for approximating all the n zeros of $p(x)$ within the absolute error bound 2^{-h} (under the normalization assumption (5.2)).

To realize that this upper bound is quite low, recall that n arithmetic operations are already necessary to output the n zeros of $p(x)$. Furthermore, since every arithmetic operation has two operands, we need at least $(n+1)/2$ arithmetic operations to process $n+1$ input coefficients of $p(x)$, and this is already required to approximate a single zero of $p(x)$. Thus, the algorithms of [P95] and [P96] are *optimal* (up to a polylogarithmic factor) in terms of the number of arithmetic operations they involve.

Since the latter algorithms have not been implemented yet, their evaluation is only theoretical and preliminary, but it does suggest their promise from a numerical point of view also. In particular, the analysis shows no excessive increase of the precision of the computation in these algorithms. To make this more precise, recall Example 5.1, which shows that order of hn^2 bit operations must be used by any algorithm for approximating (within 2^{-h}) even a single zero of an arbitrary polynomial $p(x)$ of (1.1) under (5.2). If such an algorithm uses n arithmetic operations, then these operations must be performed with the precision of at least order of nh bits.

One of the most attractive features of the algorithms of [P95] and [P96] is that they indeed use optimal orders of n^2h bit operations and nh bit precision of the computations (up to polylog factors). Let us give more comments on this and the correlation among the bit cost, arithmetic cost, and the precision of computing.

Since bit operation cost estimates (also called *Boolean* cost estimates) cover both arithmetic cost and computational precision estimates, we will focus our analysis on the Boolean complexity of these algorithms.

If we approximate the polynomial zeros by using (on average) $\mu(d)$ bit operations per an arithmetic operation performed with a pair of d bit numbers, then we need at least $n\mu(hn)$ bit operations even to approximate a single zero of $p(x)$ within 2^{-h} (for the worst-case input polynomial $p(x)$ satisfying (3.1) and (5.2)). More formally, we shall let $\mu(d)$ denote the number of bit operations required to perform an arithmetic operation with two integers modulo 2^d+1. Then an upper bound on $\mu(d)$ of order d^2 is supported by the straightforward algorithms for performing arithmetic operations. Based on faster algorithms for performing the latter operations (see [KO63], [To63], [SchöSt71], [AHU74], [Kn81], and [BP94]), one may decrease the bound on $\mu(d)$ to reach the orders $d^{\log_2 3}, \log_2 3 = 1.5849\ldots$, or even $O((d\log d)\log\log d)$. We will state our Boolean complexity estimates based on the latter (record) upper bound $O((d\log d)\log\log d)$, but they can be easily restated based on any other upper bound on $\mu(d)$ supported by the known algorithms.

Now we may specify that the lower bound $(n+1)nh/4$ on the overall bit operation (Boolean) complexity of approximation of polynomial zeros has been met by the upper bounds supported by the algorithms of [P95] and [P96]. (Here and hereafter, we assume the output precision of order of n bits or higher, which is required in various computer algebra applications, for instance, to solve polynomial systems of equations, and we simplify the bit operation (Boolean) complexity estimates by stating them up to polylogarithmic factors; more precise estimates can be found in [P95] and [P96].) In other words, the algorithms of [P95] and [P96] are optimal (up to polylog factors) *under both Boolean and arithmetic models of computing.*

An additional advantage of the algorithms of [P95] and [P96] (versus, for instance, Weyl's algorithm and its modifications or versus the algorithm of [NR94]) is the possibility of their fully efficient parallelization. Formally, the algorithms allow their implementation in polylog parallel time by using n arithmetic processors or $(n+h)n^2$ Boolean (bit) processors under the PRAM customary model of parallel computing (even assuming its least powerful version of EREW PRAM) [KR90], [Q94]. Furthermore, the effective parallelization of these algorithms is model independent since they are reduced essentially to performing a polylogarithmic number of basic operations, the hardest of which is FFT at n or order of n points, to be performed with the precision of order of $(n+h)n$ bits.

It seems appropriate to point out some obvious limitations of the algorithms of [P95] and [P96]. The algorithms involve some recursive geometric construction on

the complex plane (for the search of a basic annulus for splitting a polynomial into two factors), which does not seem to be easy to code. The promise of the substantial advantages of these algorithms for computer algebra computations should probably motivate sufficient efforts to overcome such a difficulty, but this problem also presents a challenge to simplify the geometric construction of the algorithms. Due to the cited complication, the algorithms in their present state do not seem to be very promising in the cases of a smaller degree input and a low precision output, although some of the techniques used can be relevant as auxiliary tools in these cases also (see, in particular, our remark at the very end of section 11).

In the next sections, we will review the divide-and-conquer approach to approximating polynomial zeros and, particularly, its specific recent versions in [NR94], [P95], and [P96]. The omitted details can be found in the bibliography and, in particular, in [P95], [P96], and [BP,a]. [BP,a] also includes some details on several alternative approaches.

6. The divide-and-conquer approach to approximating polynomial zeros. Divide-and-conquer algorithms for approximating the zeros of a polynomial $p(x)$ proceed by splitting $p(x)$ into the product of two nonconstant factors and then, recursively, by splitting each nonlinear factor into the product of two nonconstant factors. Finally, all the zeros of $p(x)$ are recovered from its linear factors. Of course, splitting should be done numerically, with control of the approximation errors.

Some auxiliary results (compare [Schö82]) facilitate such a control. In stating them, and throughout this paper, we will use the norm $\|\sum_i u_i x^i\| = \sum_i |u_i|$.

LEMMA 6.1. *Let*

$$\|p(x) - f_1(x)\dots f_\nu(x)\| \le \nu\epsilon\|p(x)\|/n,$$

$$\|f_1(x) - f(x)g(x)\| \le \epsilon_1\|f_1(x)\|$$

for some polynomials $f_1(x),\dots,f_\nu(x)$, $f(x)$ *and* $g(x)$, *and for*

$$\epsilon_1 \le \epsilon\|p(x)\|/\left(n\prod_{i=1}^{\nu}\|f_i(x)\|\right).$$

Then

$$\|p(x) - f(x)g(x)f_2(x)\dots f_\nu(x)\| \le (\nu+1)\epsilon\|p(x)\|/n.$$

To control the errors of the recursive splitting process, we apply Lemma 6.1 recursively; in each recursive step, we at first write $f_1(x) = f(x), f_{\nu+1}(x) = g(x)$ and then replace ν by $\nu+1$. To fulfill the assumptions of Lemma 6.1 in all recursive splitting steps it suffices to choose $\epsilon_1 \le \epsilon/(n2^n)$ in all steps due to the following simple but useful estimate.

LEMMA 6.2 (compare [Schö82, section 4]). *If* $n > 0$, $p(x) = \prod_{i=1}^{\nu} f_i(x)$ *is a polynomial of degree* n, *and all* $f_i(x)$ *are polynomials, then*

$$\|p(x)\| \le \prod_{i=1}^{\nu}\|f_i(x)\| \le 2^{n-1}\|p(x)\|.$$

We stop our recursive process when we approximate $p(x)$ by a product of linear factors, $p_n\prod_{i=1}^{n}(z-z_i^*)$. Then, by virtue of Lemma 6.1, the error norm of approximating $p(x)$ by this product is bounded by $\epsilon\|p(x)\|$. By applying Ostrowski's well-known

perturbation theorem [O40], [Ho70] or its extension given in [Schö82], we find that the output approximations of all the zeros of $p(x)$ by the values z_i^* are within the absolute error bound $2^{-\nu}$ if $\log(1/\epsilon) = O(\nu n)$.

The attempts at developing the divide-and-conquer approach to approximate polynomial zeros can be traced back a few decades [SeS41], [Schr57], [DL67], [DH69], [Grau71], [Ho70], [Ho71]. Several effective splitting techniques were summarized and further advanced in [Schö82], which is a comprehensive and quite extensive work on splitting a polynomial over a fixed (sufficiently wide and zero-free) annulus A (compare Appendices A and B of [P95a] and [Ki94]). In this case, one of the computed factors of $p(x)$, to be denoted $F(x)$, has all its zeros lying inside the internal disc D_{in} of the annulus A, whereas the other factor, to be denoted $G(x)$, has no zeros in the disc D_{in}, and, by assumption, no zeros of $p(x)$ (and, therefore, of its factors) lie in A. According to the definition of section 4, the disc D_{in} is ρ-isolated for $\rho = R/r > 1$, where R and r denote the two radii of the two boundary circles of the annulus A.

We will seek splitting factors numerically. Under the normalization assumption (5.2) for the polynomial $p(x)$ of (3.1), we seek approximations $F^*(x)$ and $G^*(x)$ to the factors $F(x)$ and $G(x)$ of $p(x)$ satisfying the bound

$$\|p(x) - F^*(x)G^*(x)\| \le 2^{-\hat{h}}\|p(x)\|; \tag{6.1}$$

$\hat{h}$ has the order nh. In light of Example 5.1, such a choice of $\hat{h}$ is necessary to ensure approximation of the zeros of $p(x)$ within 2^{-h}; on the other hand, such a choice of $\hat{h}$ is sufficient due to the cited perturbation theorem of Ostrowski or its extension in [Schö82].

In our divide-and-conquer construction, we will rely on the following basic result.

PROPOSITION 6.3. *Let a polynomial $p(x)$ of degree n satisfy relations* (3.1) *and* (5.2). *Let a sufficiently wide and zero-free annulus A be bounded by two circles of radii R and r such that*

$$\rho - 1 = (R - r)/r \ge \bar{c}/n^{\bar{d}} \tag{6.2}$$

for some fixed constants $\bar{c} > 0$ and $\bar{d} \ge 0$. Then, for the given annulus A, a splitting of $p(x)$ over A into two factors $F^*(x)$ and $G^*(x)$ satisfying (6.1) *can be computed at the cost of performing*

(a) *order of n arithmetic operations or*

(b) *if $\bar{d} \le 1$, order of $(h + n)n^2$ Boolean operations (in both cases, up to some polylogarithmic factors).*

The algorithms supporting the latter proposition are the older part of the entire construction. This part has been extensively studied and successfully tested. We will recall some details of these algorithms in sections 11 and 12.

Now, with the results of Proposition 6.3 in mind, we will turn to estimating the entire cost of approximating the n zeros of $p(x)$ based on the divide-and-conquer approach. Let $A_Z(n)$, $A_S(n)$, and $A_A(n)$ denote the arithmetic cost of approximating the zeros of a polynomial $p(x)$ of degree n, of splitting it over a fixed and sufficiently wide zero-free annulus, and of computing such an annulus, respectively. Then we have

$$A_Z(n) \le A_Z(degF(x)) + A_Z(degG(x)) + A_S(n) + A_A(n).$$

The recursive extension of this inequality to A_Z $(deg\ F(x))$ and $A_Z(degG(x))$ immediately yields an upper bound on the arithmetic complexity of application of a

divide-and-conquer algorithm to approximating the n zeros of $p(x)$. (A similar recurrence relation enables us to bound the Boolean (bit operation) complexity of the same computational problem.)

Proposition 6.3 gives us a linear in n (up to a polylogarithmic factor) upper bound on $A_S(n)$. Due to the above recurrence relation, we may extend this bound and arrive at a similar bound on $A_Z(n)$ as soon as we ensure that the degrees of the polynomials $F(x)$ and $G(x)$ are balanced with respect to a fixed constant $a, 0 < a < 1/2$ (we will say *a-balanced*), that is, if we ensure that

$$a < degF(x)/n < 1 - a, degG(x) = n - degF(x). \tag{6.3}$$

7. The problem of balancing the degrees. When we compute a basic annulus for splitting $p(x)$, we will additionally require that the splitting over this annulus be a-balanced (according to relation (6.3)). (Without balancing, up to $n-1$ splittings could be required, which would imply the extra factor n in the estimates for $A_Z(n)$; consider, for instance, the case where one of the two factors in each splitting is linear.) On the other hand, the task of devising a balanced splitting is not straightforward since we require balancing for any disposition of the zeros of a polynomial $p(x)$ on the complex plane. In particular, we cannot ignore the cases of various clusters of the zeros, which typically arise in the numerical treatment of polynomials with multiple zeros. For demonstration, we recall (from [Schö82]) the example of polynomials such as $p(x) = \prod_{i=1}^{n}(x - 5/7 - 4^{-i^2})$, for a large n, whose balanced splitting is hard to compute. Indeed, for this and similar polynomials balancing can be achieved only by means of computing very high precision approximations to a large fraction of all the zeros of $p(x)$ clustered about the point $5/7$. Note that shifting the origin to the point $5/7$ still would not help us solve the balancing problem in this case. (Because of such difficulties, the otherwise advanced work of [Schö82] gave no solution to the balancing problem and, exactly for this reason, ended with an algorithm for approximating the n zeros of $p(x)$ that required the extra factor n in its operation count.)

Various nontrivial techniques for balanced splitting were proposed in [BFKT89], [P89], [BT90], [BP91], [N94], [P94a], and [P95a], but none of them worked sufficiently well in the general case. Specifically, the algorithms of [P94a] and [P95a] exploited the correlation of the balancing problem to some properties of the discriminant of $p(x)$ and achieved effective balancing, but only at the first recursive splitting steps, where higher-degree polynomials had to be split into factors. The algorithms of [BFKT89], [P89], [BT90], and [N94] exploited the properties of Sturm sequences and pseudoremainder sequences to achieve balancing and yielded fast parallel algorithms (running in polylogarithmic time), but in the cases of [BFKT89], [P89], and [N94] at the expense of using very many processors and substantially increasing the overall number of arithmetic operations involved, and in the cases of [BFKT89], [P89], and [BT90] at the expense of limiting the solution to a rather special case of polynomial $p(x)$ having only real zeros. The divide-and-conquer algorithm of [BP91] and [BP,b] achieved a balanced splitting of matrix eigenvalues to obtain a fully parallelizable solution of the tridiagonal symmetric eigenvalue problem; the algorithm only required order n (up to a polylogarithmic factor) arithmetic operations (which improved, by roughly the factor n, the previous record estimates for the arithmetic computational complexity of this problem; compare [BP92]). Moreover, this result had an immediate extension to approximating the zeros of a polynomial $p(x)$ but, again, only under the same restrictive assumption that all the zeros are real. The first major advance toward achieving balancing (at a lower computational cost), in the case of any general

polynomial $p(x)$, was due to [NR94] and was based, in particular, on using Fact 4.1, some geometric constructions on the complex plane, and an extension of Rolle's well-known theorem to the complex case obtained in [CN94] (see the next section). In the next sections, we will describe the approach of [NR94] and its further improvements due to [P95] and [P96].

8. An auxiliary result based on an extension of Rolle's theorem. Hereafter, $D(X,r)$ will denote the disc on the complex plane having a center C and a radius r,

$$D(X,r) = \{x : |x - X| \le r\}. \tag{8.1}$$

In this section, we will recall an auxiliary result (see Corollary 8.1) based on the following extension of Rolle's well-known theorem to the complex case (this extension can also be viewed as an extension of the fact that a zero of multiplicity l for a function is also a zero of its $(l-1)$st order derivative).

FACT 8.1 (see [CN94]). *Let $l \le n$ be a positive integer, let ϕ be a fixed constant satisfying $0 < \phi < l/n \le 1$, and let a disc $D(X,r)$ contain at least l zeros of $p(x)$ (counting them with their multiplicities). Then the concentric disc $D(X,sr)$ (obtained by means of dilation of $D(X,r)$) contains a zero of $p^{(l-1)}(x)$, the $(l-1)$st derivative of $p(x)$, provided that s satisfies either of the following two bounds:*

(a) $s \ge 1/\sin(\pi/(n-l+1)) = O(n)$ *for* $l \le n-1$, $s = 1$ *for* $l = n$,

(b) $s \ge c\max\{(n-l+1)^{1/2}/l^{1/4}, (n-l+1)/l^{2/3}\}$ *for some fixed constant c and* $l \ne 2$.

Remark 8.1. Instead of Fact 8.1, one could have used a distinct extension of Rolle's theorem to the complex case due to Gel'fond [Ge58] (also cf. [Go94]). The resulting algorithms would support the same asymptotic computational cost estimates but with slightly larger overhead constants.

Part (b) of Fact 8.1 enables us to choose s of order $n^{1/3}$; its proof in [CN94] relies on some nontrivial properties of symmetric polynomials. The proof of part (a) is relatively simple (see Appendix A), and using part (a) is sufficient for the design and analysis of the (nearly optimal) algorithms of [P95] and [P96]. It is an open problem whether Fact 8.1 can be extended to allow s to be a fixed constant.

Now let $D(X,r)$ be a disc of (8.1) that contains at least l zeros of $p(x)$ and has the minimum radius and let s satisfy the assumptions of parts (a) and/or (b) of Fact 8.1. Then, by virtue of Fact 8.1, $p^{(l-1)}(x)$ has a zero z in the disc $D(X,sr)$. Furthermore, let $l > n/2$. Then any disc $D(Y,R)$ containing at least l zeros of $p(x)$ intersects $D(X,r)$ and has a radius of at least r. Therefore, the disc $D(Y,(s+2)R)$ contains the disc $D(X,sr)$ and, consequently, contains z. We have arrived at the following result.

COROLLARY 8.1 (see [NR94]). *If, under the assumptions of Fact* 8.1, *we have $l > n/2$, then there exists a zero z of $p^{(l-1)}(x)$ that lies in the dilation $D(Y,(s+2)R)$ of any disc $D(Y,R)$ containing at least l zeros of $p(x)$.*

Hereafter, such a zero z of $p^{(l-1)}(x)$ will be called *critical* for $p(x)$ and l.

9. Reduction to approximating the zeros of a higher-order derivative and of two factors of a given polynomial. In this section, we will review the powerful balancing techniques and some complexity results of [NR94]. Our next goal is the computation of a basic annulus A for an a-balanced splitting of $p(x)$, whose two boundary circles have radii R and r such that $R/r = \rho > 1$ for some fixed a and ρ (compare (6.2) and (6.3)). In such a case, the internal disc D_{in} of the annulus A is *ρ-isolated*, and we will say that the annulus A has a *relative width* of at least ρ and

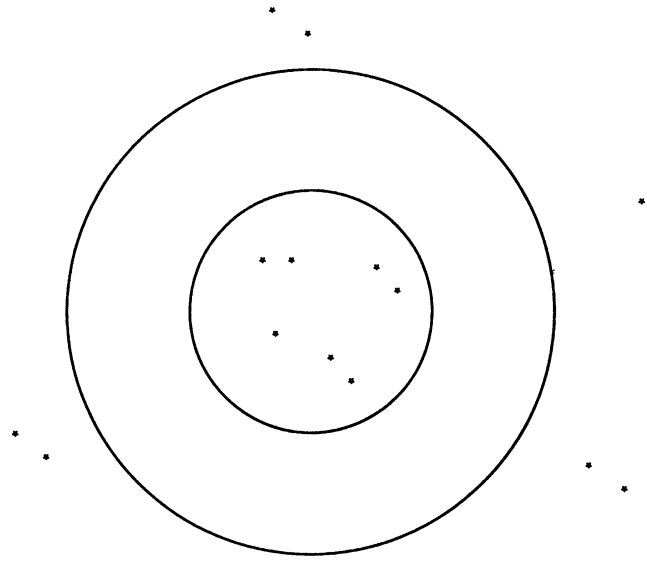

FIG. 3. *The twelve zeros of $p(x)$ are marked by asterisks. The annulus A_6 is wide enough to support a-balanced splitting for any positive $a < 1/2$.*

supports ρ-isolation (of its internal disc). At first, we will seek such an annulus by using only Fact 4.1. Set

$$\beta = \lceil an \rceil, \quad \gamma = n - \beta. \tag{9.1}$$

Apply the algorithm supporting Fact 4.1 and let r_i^- and r_i^+ denote the computed lower and upper bounds on the magnitude of the ith absolutely smallest zero of $p(x)$, where

$$1 \le r_i^+ / r_i^- \le 1 + cn^d \quad \text{for } i = 1, \dots, n \text{ and some fixed constants } c \text{ and } d. \tag{9.2}$$

Consider the open annuli

$$A_i = \{x : r_i^+ < |x| < r_{i+1}^-\}, \ i = \beta, \beta + 1, \dots, \gamma,$$

which are empty where $r_i^+ \ge r_{i+1}^-$ and which, by their definition, are always free of the zeros of $p(x)$. Clearly, the annulus A_i supports (r_{i+1}^-/r_i^+)-isolation of its internal disc and, therefore, can serve as a desired basic annulus for splitting if $r_{i+1}^-/r_i^+ \ge \rho$. (Property (6.3) of a-balancing will hold due to (9.1).) This solves the balancing problem in the case where at least one of the annuli A_i is wide enough (see Figure 3).

It remains to examine the "all narrow annuli" case, where

$$r_{i+1}^-/r_i^+ < \rho, \quad i = \beta, \beta + 1, \dots, \gamma. \tag{9.3}$$

By combining bounds (9.2) and (9.3) and taking into account that $\gamma - \beta = n - 2\beta$ (see (9.1)), we find that

$$r_{\gamma+1}^+/r_\beta^- \le (1 + c/n^d)^{n-2\beta+2} \rho^{n-2\beta}. \tag{9.4}$$

Thus, Fact 4.1 is already powerful enough to confine our splitting problem to the case where (9.4) holds. Now consider the annulus

$$A_{\beta,\gamma+1} = \{x : r_\beta^- \le |x| \le r_{\gamma+1}^+\} \tag{9.5}$$

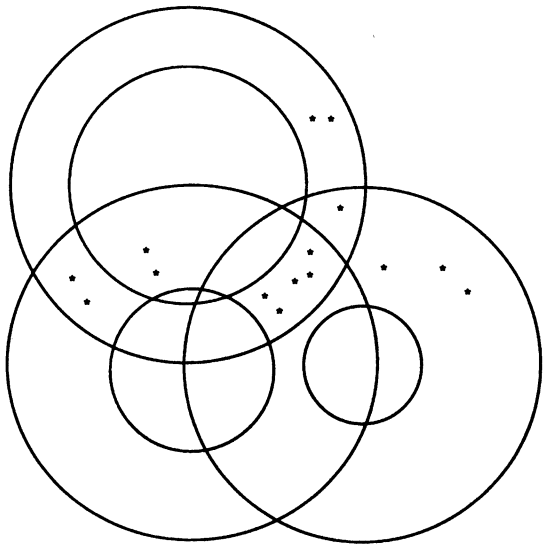

FIG. 4. *The zeros of $p(x)$ are marked by asterisks.*

containing at least $\gamma - \beta + 1 = n - 2\beta + 1 = n - 2\lceil an \rceil + 1$ zeros of $p(x)$. Due to (9.4), we can make this annulus quite narrow by choosing ρ, c, and d such that $1/c$ and d are sufficiently large and ρ is close to 1. Furthermore, we twice apply the same construction for the origin shifted into the points $2r_{\gamma+1}^{+}$ and $2r_{\gamma+1}^{+}\sqrt{-1}$, respectively, and again, we only need to consider the case where these applications define no basic annulus for a-balanced splitting. In this case, we have three narrow annuli, each containing at least $n - \beta + 1 = n - 2\lceil an \rceil + 1$ zeros of $p(x)$ (see Figure 4). The intersection I of these three annuli can be covered by a readily available disc $D(X, r) = \{x : |x - X| \leq r\}$, whose relative radius $r/|X|$ can be made small since the three annuli can be made narrow. That is, we will ensure that

$$(s+2)r < |X| \tag{9.6}$$

or, equivalently, that the disc $D(X, (s+2)r)$ does not contain the origin. (Here and hereafter we assume that s satisfies assumptions (a) and/or (b) of Fact 8.1.)

We will seek a contradiction of the latter property of the disc $D(X, (s+2)r)$ to the assumption that each of the three annuli contains at least $n - 2\lceil an \rceil + 1$ zeros of $p(x)$. The contradiction will imply that the described approach is bound to output a desired basic annulus for an a-balanced splitting of $p(x)$. As the first step toward obtaining the contradiction, we will impose the requirement that $a \leq 1/12$, so that $t = (\gamma - \beta + 1)/n > 5/6, 3t - 2 > 1/2$, and then, rather easily, we will deduce that the intersection I of the three annuli and, therefore, also the disc $D(X, r)$, contains more than $n/2$ zeros of $p(x)$ (see Appendix B). Then Corollary 8.1 will imply that the disc $D(X, (s+2)r)$ must contain a critical zero z of $p^{(l-1)}(x)$ for $p(x)$ and any $l > n/2$. Now, we recall that the disc $D(X, (s+2)r)$ does not contain the origin, and we may enforce the desired contradiction by applying the above construction for the origin shifted into z. This gives us a simple algorithm for computing a desired disc for splitting $p(x)$ provided that we know z.

Under the milder assumption that we know all the $n-l+1$ zeros of $p^{(l-1)}(x)$ but do not know which of them is (or are) critical for $p(x)$ and l, we may still compute the desired basic annulus if we apply the described algorithm $n-l+1$ times for the origin shifted into each of these $n-l+1$ zeros. In fact, $\lceil \log_2(n-l+1) \rceil$ such applications suffice since we may implicitly perform the binary search of a critical zero z. (At least one-half of the candidates for a critical zero can be discarded if the described algorithm is performed for the origin shifted into the quasi median μ of the set of the candidates, where $\mu = \mu_0 + \mu_1\sqrt{-1}$ for μ_0 and μ_1 being the medians of the two sets of the real parts and the imaginary parts of all the candidate points, respectively, [NR94].) These observations enable us to reduce the original problem of approximating the zeros of $p(x)$ to three similar problems, where the input polynomials replacing $p(x)$ are $p^{(l-1)}(x), F(x)$, and $G(x)$, respectively (the latter two polynomials denoting the two computed factors of $p(x)$).

The computational cost of this reduction is dominated by the cost of splitting $p(x)$ into two factors. The resulting recurrence relation for the arithmetic computational cost of approximating polynomial zeros, together with Proposition 6.3, leads, after some work, to an upper bound $O((\log h)n^{1+\epsilon})$ (for any fixed positive ϵ) on the arithmetic cost of approximating the n zeros of $p(x)$ within 2^{-h} (compare [NR94]). This does not give us a desired linear (up to a polylog factor) estimate yet because of the extraneous factor n^{ϵ}. The factor is also disturbing because the overhead constant factor hidden in the estimate $O((\log h)n^{1+\epsilon})$ grows to ∞ as $\epsilon \to 0$. Besides, the possibility of parallel acceleration of the resulting algorithm is limited since the algorithm requires the approximation of the zeros of the higher-order derivative $p^{(l-1)}(x)$ before obtaining the first splitting of $p(x)$. Next, we will remove these deficiencies by avoiding the approximation of the zeros of $p^{(l-1)}(x)$.

Remark 9.1. The paper [NR94] contains some interesting new techniques but unfortunately also claims much stronger estimates than its algorithms support. Moreover, the algorithm of [NR94] fails or performs very poorly for some input polynomials, in particular, for ones having large clusters of zeros (compare our specific comments in [P95] and [P96]).

10. Avoiding approximation of the zeros of a higher-order derivative. The algorithm of the previous section involves the approximation of all the zeros of $p^{(l-1)}(x)$, a higher-order derivative of $p(x)$. This costly step was our means but not, however, our final objective. In this section, we will avoid such a step by replacing it by another means, which we call *recursive screening of the zeros of a higher-order derivative*. Suppose that we have a basic annulus $A(l)$ for splitting a higher-order derivative $p^{(l-1)}(x)$ into the product of two factors $f_l(x)$ and $g_l(x)$. Let w denote the absolute width of the annulus $A(l)$, that is, the difference between the radii of the two circles bounding $A(l)$. Let $D(X,r)$ denote the disc computed by the algorithm of the previous section, so that its dilation $D(X,(s+2)r)$ contains a zero z of $p^{(l-1)}(x)$, which is critical for $p(x)$ and l. We recall that we may bound from above the values r and, consequently, $(s+2)r$, and we will use this power to ensure that $(s+2)r < w$. Due to the latter bound, the disc $D(X,(s+2)r)$ cannot simultaneously intersect both the internal disc of the annulus $A(l)$ and the exterior of $A(l)$ (see Figure 5). This enables us to determine whether the point z lies in the internal disc or in the exterior of $A(l)$, that is, whether $f_l(z) = 0$ or $g_l(z) = 0$. Then, in our search for z, we need to work with only one of the two factors of $p^{(l-1)}(x)$, and we discard the other factor. Since we also enforce a-balancing of the splitting of $p^{(l-1)}(x)$, the degree of the remaining factor is substantially lower than the degree of $p^{(l-1)}(x)$, and we need at most order

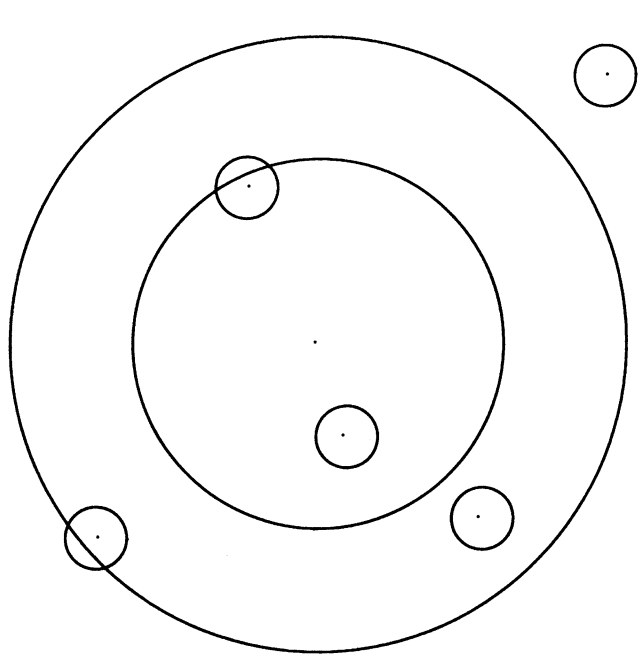

FIG. 5. *Five possible positions of the disc* $D(X,(s+2)r)$ *relative to the annulus* $A(l)$.

of $\log_2(n-l+1)$ recursive repetitions of this process before arriving at a desired basic annulus A for an a-balanced splitting of $p(x)$. This is a substantial advantage over the approach of [NR94] described in the previous section. Indeed, the latter approach required the performance of at first $n-l$ splittings of the polynomial $p^{(l-1)}(x)$ and its factors into smaller degree factors to approximate all the $n-l+1$ zeros of $p^{(l-1)}(x)$, and only then activated the binary search for z.

The analysis of the computational cost of the improved algorithm gives us the desired arithmetic complexity estimates of order n (up to polylog factors); furthermore, the algorithm runs in polylogarithmic parallel time by using n arithmetic processors, as we claimed earlier.

There are, however, some delicate points that require further elaboration. One point is that we must specify the splitting that supports Proposition 6.3. We will do this in the next section, which will bring us to some additional study of the relative widths of the computed basic annuli for splitting. Another (related) point is that the three annuli defined in the construction of the previous section must be made narrow to ensure bounds (9.6), and this implies that generally we arrive at the problem of splitting $p(x)$ over a narrow annulus A. Moreover, the recursive application of the same construction to splitting $p^{(l-1)}(x)$ requires obtaining more and more narrow annuli in each recursive step. At some stage of the recursive process, this restriction does not allow us to meet the required bound (6.2) of Proposition 6.3 on the isolation ratio ρ of the internal disc D_{in} of the annulus A (that is, to meet the requirement that $\rho \geq 1+\bar{c}/n^{\bar{d}}$ for two fixed constants $\bar{c}$ and $\bar{d}$). In fact, generally we need to deal with narrow annuli for which $\bar{d} \to \infty$ as $n \to \infty$, whereas, on the other hand, we recall that we must have $\bar{d} \leq 1$ if we wish to apply Proposition 6.3 to also bound the precision of computing and the Boolean complexity of the splitting.

Fortunately, there is a way to salvage the algorithm, which we call *recursive contraction of the area for splitting annuli.* The idea is to proceed recursively to

decrease the diameter of the small disc $D(X,r)$ covering the intersection of the three narrow annuli. Indeed, we may apply the same construction for the origin shifted into the center X of the latter disc and recursively repeat this process until we compute either a desired basic annulus for splitting $p(x)$ or a small disc covering more than $n/2$ zeros of $p(x)$ having diameter less than the width w of the annulus $A(l)$. We may ensure the decrease of the diameter of the small discs by at least a fixed constant factor exceeding 1 in each recursive step, even when we apply the construction of the previous section for ρ of the order $1+\bar{c}/n$ and for a certain fixed positive constant $\bar{c}$ (for instance, one may choose $\bar{c}=0.01, \rho=1+1/(100n)$; see [P95] and [P96]). Then in at most order $\log n$ such recursive steps we will achieve our goal of computing either a desired basic annulus for splitting $p(x)$ or a sufficiently small disc containing more than $n/2$ zeros of $p(x)$, which will enable us to discard one of the factors of $p^{(l-1)}(x)$. Because of the choice of ρ of the order $1+\bar{c}/n$, we will satisfy the requirement of Proposition 6.3 to the relative width of the computed annulus A for splitting, and then we will deduce the claimed upper estimates for both arithmetic and Boolean complexity of approximating polynomial zeros.

Yet another delicate point is that the zeros of $p(x)$ (and, similarly, the zeros of $p^{(l-1)}(x)$ and/or other polynomials involved) may form a *massive cluster*, including, say, $n-\lceil \log_2 n \rceil$ zeros. Then for the computation of a balanced splitting it would be required to separate some zeros of the cluster from its other zeros; if the cluster has a very small diameter, this problem cannot be solved within the required bounds on the overall computational complexity. To avoid solving such a difficult problem, we simply compute a single point of the approximation to all the zeros of a cluster without obtaining a balanced splitting of $p(x)$. (For instance, for the polynomial $\prod_{i=1}^{n}(x-5/7-4^{-2^i})$ and h of order n we shall choose $Y=5/7$ as such an approximation point.) To recognize the existence of massive clusters of the zeros and to handle their case we must modify the presented algorithm, however. We refer the reader to [P95] and [P96] on this and other omitted details and to the next sections on the algorithms that support Proposition 6.3 for splitting a polynomial over an annulus.

11. Splitting a polynomial over a fixed annulus. There are a few algorithms that support part (a) of Proposition 6.3, which states the upper estimate n (up to a polylog factor) for the arithmetic complexity of splitting a polynomial over a fixed annulus; in particular, this part of Proposition 6.3 is supported by the modern modification [BG92] of the old algorithm [SeS41], [Ho71] or, alternatively, by any of the three algorithms of [Bi89], [P96c], and Appendix C of [P96]. The listed algorithms exploit various nontrivial techniques involving computations with structured matrices, but all of these algorithms have a major deficiency: they assume no reasonable control over the precision of computing. Actually, the precision is prone to blowing up since these algorithms recursively apply polynomial division or similar operations, which substantially magnify the input errors in each recursive step. Therefore, to prove part (b) of Proposition 6.3 (on the Boolean (bit operation) complexity of splitting) and to produce an effective realistic algorithm for splitting, one must change or modify these approaches. Next, we will briefly sketch a distinct approach, presented in [Schö82], [P95a], and [Ki94] and earlier developed in [Grau71], [DH69], [DL67], and [Schr57]. (We have already cited the latter bibliography.) Presently, this is the most successful approach to splitting a polynomial over a fixed annulus in terms of the worst-case bit complexity estimates.

We will list only its main stages and describe a recent improvement of one of them, referring the reader for further details to the cited papers. Our omissions include

several technicalities and the somewhat tedious treatment of many numerical aspects of (balanced) splittings. Furthermore, as indicated at the end of the previous section, in the case of a massive cluster of the zeros we even need to abandon the requirement of balancing the degrees. In this section, we will avoid the case of massive cluster; we will assume that we are given a basic annulus A, over which the splitting of $p(x)$ is a-balanced, and that we must approximate this splitting.

At first, an initial approximation $F^*(x)$ to the factor $F(x)$ of $p(x)$ (having all its k zeros in D_{in}) is computed, so that the coefficients of the polynomial $F^*(x)-F(x)$ have absolute values less than 2^{-cn} for a fixed constant c. Then an approximation $G^*(x)$ to the other factor $G(x)$ is obtained by the high precision numerical division of $p(x)$ by $F^*(x)$, so that all the coefficients of the polynomial $\Delta(x) = F^*(x)G^*(x) - p(x)$ have sufficiently small absolute values (compare (6.1)). A sophisticated and even tedious but computationally simple and effective Newton-type iterative process has been developed, which, under the latter assumption on the coefficients of $\Delta(x)$, rapidly refines desired approximations $F^*(x)$ to $F(x)$ and $G^*(x)$ to $G(x)$ (compare [Schö82], [Ki94], or Appendices A and B of [P95a]).

The entire algorithm (including both the computation of a relatively rough initial approximation and its subsequent rapid refinement) is performed at a sufficiently low computational cost within the arithmetic and Boolean complexity bounds n and $(n+h)n^2$, respectively (in both cases up to polylog factors), except that some additional care is required at the first stage of computing the coefficients of the polynomial $F^*(x)$. That is, $F^*(x)$ is recovered from the approximations s_m^* to the power sums $s_m = s_{m,k}$ of the k zeros of $p(x)$ lying in the disc D_{in},

$$s_m = \sum_{i=1}^{k} z_i^m, \quad i = 1, \dots, 2k-1.$$

(We assume that the zeros $z_1, \dots, z_n$ of $p(x)$ have been enumerated so that z_i lies in D_{in} if and only if $i \le k$, provided that we count μ times every zero of a multiplicity μ, and we abuse the notation slightly when we write s_m rather than $s_{m,k}$, assuming that k is fixed.) In principle, a desired approximation $F^*(x)$ to the factor $F(x)$ can be recovered from $s_1^*, \dots, s_k^*$ by using Newton's identities at the cost of performing order of $(\log k)^2 k$ arithmetic operations [BP94], but we achieve much better error control and a lower cost of order of $k \log k$ arithmetic operations for $k < n$ by using a special *algebraic version of Newton's iteration*; see [Schö82], [BP94, pp. 34–35], or [P,a]. (This algorithm involves $s_1^*, \dots, s_M^*$, where $2k > M = 2^g \ge k$.) Due to the efficacy of this algorithm, some additional care is required only at the stage of computing the approximations s_m^* to the power sums s_m. This is done based on the known equations [He74]

$$s_m = \frac{1}{2\pi\sqrt{-1}} \int_\Gamma (x_m p'(x)/p(x))\, dx,$$

where Γ denotes any circle lying strictly inside the annulus A and concentric with it. The approximations s_m^* to s_m for $m = 1, \dots, 2k-1$ are computed by means of numerical integration with Q nodes of integration equally spaced on Γ. Due to this choice of the nodes, the integration is reduced to performing three FFTs on a set of Q points. (Due to the application of FFT, this means about $4.5\, Q \log Q$ arithmetic operations if Q is a power of 2.) Estimates presented in [Schö82], [PD93], [P90], and [P95a, Appendix A] show that to achieve the desired accuracy of the splitting one

should choose Q of order $nr/(R-r)$. This analysis suggests that ensuring the bound $(R-r)/r \geq \theta$, for a positive constant θ, is needed to split $p(x)$ over A at the arithmetic cost bounded at the level $(\log hn)(\log n)n$ (provided that we apply the latter approach to splitting).

The algorithm of section 9 (even after its amelioration in section 10 by means of recursive contraction of the search area for splitting annuli) computes a basic annulus A for splitting that may be too narrow to satisfy such a lower bound on $(R-r)/r$. Indeed, according to the construction of section 9, we try to find a basic annulus for splitting $p(x)$ among the $\gamma - \beta + 1 = n - 2\lceil an \rceil + 1$ annuli

$$A_i = \{x: \ r_i^+ < |x| < r_{i+1}^-\}, \quad i = \beta, \beta+1, \ldots, \gamma \tag{11.1}$$

(compare (9.1)–(9.5)), and if, say, the ratios $\rho_i = r_{i+1}^-/r_i^+$ are invariant in i, then $r_{\gamma+1}^-/r_\beta^+ = (n - 2\lceil an \rceil + 1)\rho_i$. In this case, even if the annulus $A_{\beta,\gamma+1}$ of (9.5) has relative width 4 or 10, say, then all the annuli A_i still have relative widths of at most $1 + c/n$ for a constant c, and this only supports an isolation ratio of at most $1 + c/n$ for their internal discs. To approximate (by means of numerical integration) the power sums s_m of the zeros of $p(x)$ lying in such an internal disc, we need order of n^2 nodes of integration and, consequently, order of $n^2 \log n$ arithmetic operations if we wish to ensure the desired bound on the output errors. (Moreover, the isolation ratios of the internal discs would have decreased to $1 + c/(ns)$ for s of Fact 8.1, that is, for s of order $n^{1/3}$, if (as, for example, in the case of the algorithm of [NR94]) one proceeded without using the techniques of section 10 for recursive contraction of the search area for the splitting annuli. With such smaller isolation ratios, the desired approximation of the power sums s_m would have required order of $n^2 s$ nodes, more than $n^2 s$ arithmetic operations, and more than $(h+n)n^2 s$ Boolean operations.) On the other hand, at the stage of approximating the power sums s_m we need to satisfy only a relatively rough bound on the output errors to approximate the coefficients of $F(x)$ within 2^{-cn}, as required. Consequently, a precision of computing may stay at a relatively low level of order of n bits, versus order of hn bits, required at some other stages of the computation of a splitting of $p(x)$. Therefore, at this stage, order of $n^2 \log n$ arithmetic operations can be performed by using $(n+h)n^2$ Boolean operations (up to a polylog factor), as desired.

In the next section, we will modify the computation to reach this Boolean complexity bound by using a total of n arithmetic operations (up to a polylog factor).

The improvement of only the arithmetic cost bound, not followed by any improvement of the Boolean cost bound (which generally is a more realistic bound and in our case is already nearly optimal), is not practically important. However, obtaining an algorithm that simultaneously supports both Boolean and arithmetic complexity bounds at nearly optimal levels is a theoretically important task. Besides, we obtain such an algorithm by means of some techniques that enable us to yield a dramatic increase of the relative width of a basic annulus for splitting (say, from $1 + c/n$ to 4). Such an increase may turn out to be useful in combination with other algorithms (such as the one that supports Fact 4.1) that may generally compute a rather narrow basic annulus for splitting a polynomial, but very frequently the application of the latter techniques turns it into a sufficiently wide annulus. Moreover, some techniques to be used in the analysis of this approach (in particular, in the study of the numerical condition properties of Padé approximation) may be of independent interest.

12. Enforcing a stronger isolation of a basic annulus for splitting. Suppose that we have a basic annulus A for splitting $p(x)$ and that such an annulus has

a relative width $1 + c/n$, that is, supports $(1 + c/n)$-isolation of its internal disc for a fixed positive constant c, and suppose that we wish to strengthen this isolation by raising its ratio to the level 4, say. Since we may shift and scale the variable x, we may assume, with no loss of generality, that the annulus A is bounded by the two circles $\{x : |x| = 1\}$ and $\{x : |x| = 1 + c/n\}$, and we will assume also that

$$p(x) = p_n\ F(x)\ G(x),\ F(x) = \prod_{j=1}^{k}(x - z_j),\ G(x) = \prod_{j=k+1}^{n}(x - z_j),$$

$|z_j| \le 1$ for $j = 1, \dots, k$, $|z_j| \ge 1 + c/n$ for $j = k + 1, \dots, n$.

Now we write $y = x$, $t_0(y) = p(y)/p_n$ to arrive at a monic polynomial $t_0(y)$ and apply Graeffe's iteration (3.2); that is, we write

$$t_{i+1}(y) = (-1)^n t_i(\sqrt{y}) t_i(-\sqrt{y}),\quad i = 0, 1, \dots.$$

For every i, the ith step of this iteration squares the zeros of $t_i(y)$, so that

$$t_i(y) = \prod_{j=1}^{n}(y - z_j^{2^i}),\ i = 0, 1, \dots.$$

It immediately follows that the unit disc $D(0,1) = \{y : |y| \le 1\}$ is ρ_i-isolated (for $\rho_i = (1 + c/n)^{2^i}$) with respect to $t_i(y), i = 0, 1, 2, \dots$. In particular, assuming that $u \ge 1 + \log_2(n/c)$, so that $(1 + c/n)^{2^u} > 4$, we obtain that the disc $D(0,1)$ is 4-isolated with respect to $t_u(y)$. Therefore, the algorithm of the previous section enables us to split $t_u(y)$ over the annulus $A(u)$ bounded by the circles $\{y : |y| = 1\}$ and $\{y : |y| = (1 + c/n)^{2^u}\}$; moreover, since $(1 + c/n)^{2^u} > 4$, we obtain this splitting at a low arithmetic and Boolean computational cost. It remains to descend from spitting $t_u(y)$ over the annulus $A(u)$ (into two factors $F_u(y)$ and $G_u(y)$) to splitting $t_0(y)$ over the annulus $A = A(0) = \{y : 1 \le |y| < 1 + c/n\}$. We will proceed recursively in u steps of descending. The ith descending step will output two factors $F_{u-i}(y)$ and $G_{u-i}(y)$ of $t_{u-i}(y)$, $i = 1, 2, \dots, u$, provided that the polynomials $t_{u-i}(y), F_{u-i+1}(y)$, and $G_{u-i+1}(y)$ are given as the input of this step and

$$t_v(y) = F_v(y)\ G_v(y),\quad F_v(y) = \prod_{j=1}^{k}(y - z_j^{2^v}),\quad G_v(y) = \prod_{j=k+1}(y - z_j^{2^v})$$

for $v = 0, 1, \dots, u$.

In addition to these equations, in particular, to the equation $t_v(y) = F_v(y)\ G_v(y)$, we recall that

$$F_{v+1}(y^2) = (-1)^k\ F_v(y) F_v(-y),$$
$$G_{v+1}(y^2) = (-1)^k\ G_v(y) G_v(-y),$$

and for every v the two polynomials $F_v(y)$ and $G_v(-y)$ have no common zeros and, therefore, have only constant common divisors. It follows that

$$M_i(y) = \frac{t_i(y)}{G_{i+1}(y^2)} = \frac{F_i(y)}{G_i(-y)}$$

is a meromorphic function whose Padé approximation table has its $(k, n-k)$-entry filled with the pair of polynomials $F_i(y)$ and $G_i(-y)$. (The $(k, n-k)$-entry of the Padé

approximation table for a meromorphic function $M(y)$ is defined as the pair of polynomials $F(y)$ and $G(y)$ satisfying the relations $F(y)M(y) = G(y) \bmod y^{n+1}, degF(y) \le k, degG(y) \le n-k$. For fixed $M(y), n$ and k, this pair defines a unique rational function $F(y)/G(y)$ according to the Frobenius theorem (see [Gra72, Theorem 3.1]). Gragg's paper [Gra72] contains ample material on Padé approximation; see also [BP94] on some related algorithms.) Therefore, the desired transition from the polynomials $t_i(y)$ and $G_{i+1}(y^2)$ to the polynomials $F_i(y)$ and $G_i(-y)$ (and therefore, also to $G_i(y)$) can be achieved simply by computing the $(k, n-k)$-entry of the Padé approximation table for the meromorphic function

$$M_i(y) = t_i(y)/G_{i+1}(y^2).$$

This computation can be reduced to solving a Toeplitz linear system of $n-k$ equations (compare, for instance, equations (2.5.5) and (2.5.6) of [BP94]). In our case, the system is nonsingular because the absence of common nonconstant divisors of $F_i(x)$ and $G_i(-y)$ excludes the degeneration of the Padé approximation. Moreover, it was proven in [P96] that the perturbation of $M_i(y)$ by a polynomial $m_i(y)$ implies the perturbation of $F_i(y)$ and $G_i(-y)$ by polynomials $f_i(y)$ and $g_i(y)$ satisfying the bound

$$\|f_i(y)\| + \|g_i(y)\| \le \|m_i(y)\|(2 + 1/(\rho_i - 1))^{Cn} \tag{12.1}$$

for some fixed constant C, where ρ_i is the isolation ratio associated with $F_i(y)$ and $G_i(y)$, $\rho_i = (1 + c/n)^{2^i}$, $i \le u$ in our case.

The known algorithms support the solution of a nonsingular Toeplitz linear system of $n-k$ equations by using order of $(\log(n-k))^2(n-k) < (\log n)^2 n$ arithmetic operations (compare [BP94] for these fast solution algorithms for a Toeplitz system and their efficient parallel counterparts). We increase the latter bound to the order $(\log n)^3 n$ when we solve $u = O(\log n)$ such systems in the process of descending from splitting $t_u(y)$ to splitting $t_0(y)$.

To control the Boolean computational cost of the described descending process, we compute the splitting of $t_i(x)$, for each i, in two stages. At first, relatively rough initial approximations to the two factors of $t_i(x)$ are computed (compare (12.1)). Then these approximations are rapidly refined (at a low computational cost) by the available techniques of Newton's iteration (we already cited these techniques while describing the splitting process for $p(x)$). To keep the Boolean complexity lower at the first stage, we perform the computation at this stage with a lower precision. Bound (12.1) enables us to control the approximation and rounding errors in this process. The resulting Boolean computational cost is shown to be sufficiently small [P96], [BP,a], and we arrive at the desired algorithm that supports the desired optimal (up to polylogarithmic factors) upper bounds on the sequential and parallel computational cost of approximating polynomial zeros simultaneously under both arithmetic and Boolean models of computing.

13. Polynomial zeros and matrix eigenvalues. In addition to the companion matrix approach discussed in section 5, let us consider approximating the zeros of $p(x)$ as the eigenvalues of an associated tridiagonal matrix T. If all the (coefficients and) zeros of $p(x)$ are real, then such a tridiagonal matrix T is immediately computed via an application of the extended Euclidean algorithm to $p(x)$ and $p'(x)$ (see [BP94, Chapter 2, section 3; Chapter 4, section 5] or [BP,b]). Furthermore, the matrix T is real and symmetric in this case, and its eigenvalues can be effectively approximated by the known methods (bisection, divide-and-conquer, or QR). In particular, [BP91] and [BP,b] show how this can be done in linear arithmetic time (up

to a polylog factor) based on the divide-and-conquer approach (see [BNS78], [C81], and [DS87] on some preceding works), and [BP92] shows a practical modification of the algorithm of [BP91]. Would a similar approach work for an arbitrary polynomial $p(x)$ or at least for a large class of polynomials $p(x)$ with complex zeros? The idea is to start by computing a complex tridiagonal or just companion matrix A that has characteristic polynomial $p(x)$ (and is in Hessenberg form) and then to approximate the eigenvalues of A by using a divide-and-conquer algorithm. (In the experiments reported in [DSi93], such a heuristic divide-and-conquer algorithm worked well for quite a large class of unsymmetric matrices.) One may also try to reverse the direction to approximate the eigenvalues of an unsymmetric matrix as the zeros of its characteristic polynomial $c_A(x)$. In practice, computing the coefficients of $c_A(x)$ is avoided since this generally blows up the precision of computing, thus creating numerical stability problems. Such an observation rules out the numerical computation of the eigenvalues by any method for polynomial zeros that involves these coefficients. The techniques of such methods, however, can be potentially useful for the eigenvalue approximation. In particular, one may try to extend Weyl's algorithm in this direction. The initial square, containing all the eigenvalues, can be easily obtained by applying Gershgorin's theorem, and it remains to find a recipe for proximity tests not using the coefficients of the characteristic polynomial. This seems to be an interesting challenge. A heuristic recipe, based on the results of Eckart–Young [EY39] and Gastinel [Ka66] regarding the distance to the nearest singular matrix, was proposed in [P95b].

14. Some further applications. We have already mentioned the important impact of the study of the solution of a polynomial equation on pure and computational mathematics. The list of examples of such an impact can be continued extensively, but we prefer to conclude by recalling just three major extensions of the problem of solving a polynomial equation to computing

(a) a solution or an approximate solution of a system of complex polynomial equations,
(b) a factorization of a polynomial over the field of rational numbers, and
(c) the greatest common divisor (gcd) of two univariate polynomials $u(x)$ and $v(x)$.

All of these subjects (particularly the first one, which also has a further important variation in which one must solve a real system of polynomial equations and inequalities; compare [R92]) are highly important for both mathematics and the theory and practice of computing. All three subjects have been intensively investigated by researchers for many years. We will now cite some known techniques for the effective reduction of the first two of these computational problems to solving a polynomial equation. The elimination theory [VdW53], [R89], [CKL89] reduces a polynomial system to a single equation (1.1). Both diophantine approximation [Schö84] and algebraic relation finding [Mi92] reduce factorization over the rationals to the approximation of polynomial zeros. The cited reductions, presented in [R89], [CKL89], and [Mi92], lead to the current record computational complexity estimates for the worst-case solution of these two major computational problems. Two other major approaches to solving a system of polynomial equations support inferior asymptotic estimates for the computational complexity of the solution in the worst case (of a dense input), but are preferred by the users for practical implementation. These two approaches rely on computing Groebner bases [KL92] and Newton's polytops (for sparse polynomial systems) [E96], respectively. Then again, in both cases the solution reduces to solv-

ing some polynomial equations in a single variable. Furthermore, the major stage of bounding the step size in the recent successful path-following algorithms of [SS93], [SS93a], [SS93b], and [SS93d] (proposed for solving a polynomial system of equations and based on multidimensional Newton's iteration) is also reduced to solving a single univariate polynomial equation. Solving a polynomial system of equations with several variables is usually much harder than solving the single polynomial equation (1.1), in particular, because the total number of solutions to the system is generally very large (exponential in the number of variables). Thus, one frequently seeks only partial information about the solutions to the system, such as computing or just counting only the real solutions or the solutions in a fixed domain. Toward this goal, some techniques known for approximating the zeros of a univariate polynomial (such as ones for the proximity test) can be effectively extended to the multivariate case [DY92].

The third problem, of computing the gcd of $u(x)$ and $v(x)$, immediately reduces to computing the zeros of $u(x)$ and $v(x)$. We recently proposed using this reduction for the numerical approximation of the gcd [P96b]. This was partly motivated by the recent progress in approximating polynomial zeros and partly by the very poor numerical behavior of the available algorithms for polynomial gcds. The numerical reduction of the gcd to the zeros involves some bipartite graph algorithms (matching, connected components) and has further correlations to the computation of numerical ranks of Toeplitz matrices [P96b]. The latter problem is of practical importance because of its application to sparse multivariate polynomial interpolation and Padé approximation.

Appendix A. Extension of Rolle's theorem to the complex case. We will follow the technique of [CN94] to prove part (a) of Fact 8.1. We will start by recalling a little known but simple lemma.

LEMMA A.1 (see [CN94]). *Let $v_1, \dots, v_l$ denote the vertices of a simplex σ in the $(l-1)$-dimensional real space R^{l-1}. Let $c_1, \dots, c_l$ be l complex points in C and let $\alpha : R^{l-1} \to C$ be the real affine map taking v_i to c_i. Let f be an analytic function on the image of α. Let $[c_1, c_2, \dots, c_l]\ f$ denote the divided difference operator applied to f and let $v(\vec{t})$ be the standard volume form on R^{l-1}. Then*

$$[c_1, c_2, \dots, c_l]\ f = \int_\sigma f^{(l-1)}(\alpha(\vec{t}))dv(\vec{t}). \tag{A.1}$$

Proof of part (a) *of Fact* 8.1. Apply Lemma A.1, where $f(x) = p(x)$ and $c_1, \dots, c_l$ are the zeros of $p(x)$. Then the left-hand side of (A.1) vanishes. Therefore, so does the right-hand side. This means that its integrand varies by at least π, and this implies the condition on the zeros of $p^{(l-1)}(x)$ of part (a) of Fact 8.1. □

Appendix B. Correlation between the cardinalities of intersection and union.

PROPOSITION B.1. *Let $S_1, S_2, \dots, S_R$ denote R finite sets, let U denote their union, and let I denote their intersection. Let $|S|$ denote the cardinality of a set S. Then*

$$|I| \geq \sum_{i=1}^{h} |S_i| - (h-1)|U|.$$

Proof. We only need this result for $h = 3$ and will prove it for this h by following the technique of [NR94]. Let s_i and s_{ij} denote the set cardinalities $s_i = |S_i - (S_j \bigcup S_k)|$

and $s_{ij} = |(S_i \bigcap S_j) - I|$, where i, j, and k are distinct integers chosen from among 1, 2, and 3 (in any order). Then clearly

$$|S_1| = s_1 + s_{12} + s_{13} + |I|,$$
$$|S_2| = s_2 + s_{12} + s_{23} + |I|,$$
$$|S_3| = s_3 + s_{13} + s_{23} + |I|,$$
$$s_1 + s_2 + s_3 + s_{12} + s_{13} + s_{23} + |I| = |U|.$$

By twice subtracting the last equation from the sum of the preceding three equations, we find that

$$|I| - s_1 - s_2 - s_3 = \sum_{i=1}^{3} |S_i| - 2|U|,$$

which implies Proposition B.1 in the case where $h = 3$. □

We used Proposition B.1 in section 9 in the case where S_1, S_2, and S_3 denoted the three sets of the zeros of $p(x)$ lying in three fixed annuli.

Appendix C. Turan's proximity test. Turan (in [Tu68] and [Tu75]) proposed a nontrivial proximity test for polynomial zeros using $O(n \log n)$ arithmetic operations to ensure an error bound of at most 10%, 40%, or 50%, say. This test relies on the following result, which he obtained by using some advanced number-theoretic techniques:

$$1 \le \max_j |z_j|/\rho_N \le 5^{1/N}.$$

Here N is a positive integer, $z_1, \ldots, z_n$ are the zeros of $p(x)$,

$$\rho_N = \max_{g=1,\ldots,n} \left| \frac{s_{gN}}{n} \right|^{1/(gN)},$$

and $s_i = \sum_{j=1}^{n} z_j^i$, $i = 1, 2, \ldots$.

Since $5^{1/32} < 1.052$, $5^{1/3} < 1.4$, and $5^{1/4} < 1.5$, it is sufficient to set $N = 32, N \ge 3$, or $N = 2$, respectively, for the purpose of having the relative output error within 10%, 40%, or 50%, respectively. By applying h steps of Graeffe's iteration (3.2), we make a transition from $p(x) = p_n \prod_{j=1}^{n} (x - z_j)$ to $p^*(x) = p_n \prod_{j=1}^{n} (x - z_j^N) = \sum_{i=0}^{n} p_i^* x^i$, for $N = 2^h$, so that we may choose $h = 5$, $h = 2$, or $h = 1$ in the cases listed above. Then we obtain the power sums s_{gN} for $g = 1, \ldots, n$ and $N = 2^h$ by solving the well-known system of Newton's identities (see, e.g., [Ho70] or [BP94])

$$p_n^* s_N + p_{n-1}^* = 0,$$
$$p_n^* s_{2N} + p_{n-1}^* s_N + 2p_{n-2}^* = 0,$$
$$p_n^* s_{3N} + p_{n-1}^* s_{2N} + p_{n-2}^* s_N + 3p_{n-3}^* = 0,$$
$$\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots$$

This is a nonsingular triangular Toeplitz linear system of n equations in s_N, $s_{2N}, s_{3N}, \ldots, s_{nN}$, whose solution can be reduced to polynomial division [BP94] and performed (together with all other stages of computing the desired value ρ_N) by using an order of $n \log n$ arithmetic operations. The overall number of arithmetic operations

involved in Turan's test (for $h = 5$, $h = 2$, or $h = 1$) is still of order $n \log n$, which is smaller by factor $\log \log n$ than in the proximity test of section 3.

Let us briefly comment on the numerical behavior of Turan's test. It is not efficient to perform this test by using rational arithmetic with no errors because the number of digits that must be carried out grows too rapidly during this computation. Indeed, performing the stage of solving a system of Newton's identities at the arithmetic cost of order $n \log n$ by means of polynomial division algorithms generally requires one to increase roughly by factor n the number of digits in the process of rational computation. Another increase by factor n is due to computing s_{gN} for g of order n. The overall growth of the precision is by a factor of order n^2, whereas at most factor n can be due to the ill conditioning of polynomial zeros. The latter considerations suggest that Turan's test should perform poorly when applied in rational arithmetic, and this has indeed been confirmed by numerical experiments. On the other hand, numerical experiments of Turan's test performed by using finite precision floating-point arithmetic have showed much better numerical behavior. In fact, Turan's test works better numerically when the distances from some zeros of the polynomial $p(x)$ to the origin are small, relative to the maximum magnitude of the coefficients of the polynomial, whereas the tests of section 3 work better in the opposite case, where all the zeros have larger magnitudes.

As we have already mentioned, there still remain many open problems on the numerical implementation of Weyl's algorithm and its modifications. (The reader is referred to [BP96] and [BP,a] on some recent work on this subject and to [Tu84] on many important applications of the techniques of [Tu68] and [Tu75] to various areas of mathematics.)

REFERENCES

[A73] O. ABERTH, *Iteration methods for finding all zeros of a polynomial simultaneously*, Math. Comp., 27 (1973), pp. 339–344.

[AS82] E. D. ANGELOVA AND K. I. SEMERDZIEV, *Methods for the simultaneous approximate derivation of the roots of algebraic, trigonometric and exponential equations*, USSR Comput. Math. and Math. Phys., 22 (1982), pp. 226–232.

[BdL24] L. E. J. BROUWER AND B. DE LOER, *Intuitionisher Beweis des Fundamentalsatzes der Algebra*, Amsterdam Konigl. Acad. Van Wetenschapen, Proc., 27 (1924), pp. 186–188 (also in L. E. J. Brouwer, *Coll. Works*, North–Holland, Amsterdam, 1975).

[Be40] E. T. BELL, *The Development of Mathematics*, McGraw–Hill, New York, 1940.

[BFKT89] M. BEN-OR, E. FEIG, D. KOZEN, AND P. TIWARI, *A fast parallel algorithm for determining all roots of a polynomial with real roots*, SIAM J. Comput., 17 (1988), pp. 1081–1092.

[BG92] D. BINI AND L. GEMIGNANI, *On the complexity of polynomial zeros*, SIAM J. Comput., 21 (1992) pp. 781–799.

[Bi89] D. BINI, *Complexity of parallel polynomial computations*, in Proc. Parallel Computing: Methods, Algorithms, Applications, J. Evans and C. Nodari, eds., Adam Hilger, Bristol, 1989, pp. 115–126.

[Bi,a] D. BINI, *Numerical computation of polynomial zeros by Aberth's method*, Numer. Algorithms, to appear.

[Bo68] C. A. BOYER, *A History of Mathematics*, Wiley, New York, 1968.

[BNS78] J. R. BUNCH, C. P. NIELSEN, D. C. SORENSEN, *Rank-one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

[BP91] D. BINI AND V. Y. PAN, *Parallel complexity of tridiagonal symmetric eigenvalue problems*, in Proc. 2nd Ann. ACM-SIAM Symp. on Discrete Algorithms, ACM Press, New York and SIAM, Philadelphia, PA, 1991, pp. 384–393.

[BP92] D. BINI AND V. Y. PAN, *Practical improvement of the divide-and-conquer eigenvalue algorithms*, Computing, 48 (1992), pp. 109–123.

[BP94] D. BINI AND V. Y. PAN, *Polynomial and Matrix Computations, v. 1: Fundamental Algorithms*, Birkhäuser Boston, Cambridge, MA, 1994.

[BP96] D. BINI AND V. Y. PAN, *Graeffe's, Chebyshev-like and Cardinal's processes for splitting a polynomial into factors*, J. Complexity, 12 (1996), pp. 492–511.

[BP,a] D. BINI AND V. Y. PAN, *Polynomial and Matrix Computations, v. 2: Selected Topics*, Birkhäuser Boston, Cambridge, MA, to appear.

[BP,b] D. BINI AND V. Y. PAN, *Computing matrix eigenvalues and polynomial zeros where the output is real*, SIAM J. Comput., to appear.

[BS63] W. BOERSCH–SUPAN, *A priori error bounds for the zeros of polynomials*, Numer. Math., (1963), pp. 380–398.

[BT90] M. BEN-OR AND P. TIWARI, *Simple algorithm for approximating all roots of a polynomial with real roots*, J. Complexity, 6 (1990) pp. 417–442.

[C81] J. J. M. CUPPEN, *A divide-and-conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.

[CKL89] J. F. CANNY, E. KALTOFEN, AND Y. LAKSHMAN, *Solving systems of non-linear polynomial equations faster*, in Proc. Ann. ACM-SIGSAM Intern. Symp. on Symbolic and Algebraic Comput. (ISSAC '89), ACM Press, New York, 1989, pp. 121–128.

[CN94] D. COPPERSMITH AND C.A. NEFF, *Roots of a polynomial and its derivatives*, in Proc. 5th Ann. ACM-SIAM Symp. on Discrete Algorithms, ACM Press, New York and SIAM, Philadelphia, PA, 1994, pp. 271–279.

[D60] E. DURAND, *Solutions Numeriques des Equations Algebriques, Tome* I: *Equations du Type* $F(x) = 0$: *Racines d'un Polynome*, Masson, Paris, 1960.

[DB64] K. DOCHEV AND P. BYRNEV, *Certain modifications of Newton's method for the approximate solution of algebraic equations*, Zhournal Vychisl. Mat. i Mat. Phiziki, 4 (1964), pp. 915–920 (in Russian).

[DH69] B. DEJON AND P. HENRICI, EDS., *Constructive Aspects of the Fundamental Theorem of Algebra*, John Wiley, London, 1969.

[DL67] L. M. DELVES AND J. N. LYNESS, *A numerical method for locating zeros of an analytic function*, Math. Comp., 21 (1967), pp. 543–560.

[DS87] J. J. DONGARRA AND D. C. SORENSEN, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 139–154.

[DSi93] J. J. DONGARRA AND M. SIDANI, *A parallel algorithm for the nonsymmetric eigenvalue problem*, SIAM J. Sci. Comput., 14 (1993), pp. 542–269.

[DY92] J.-P. DEDIEU AND J.-C. YAKOUBSOHN, *Localization of an algebraic hypersurface and the exclusion algorithm*, Appli. Algebra Engrg. Comm. Comput., 2 (1992), pp. 239–256.

[E67] L. W. ERLICH, *A modified Newton method for polynomials*, Communications ACM, 12 (1967), pp. 107–108.

[E96] I. Z. EMIRIS, *On the complexity of sparse elimination*, J. Complexity, 12 (1996), pp. 134–166.

[Ev83] H. EVES, *An Introduction to the History of Mathematics*, Saunders College Publishing, Philadelphia, PA, 1983.

[EY39] C. ECKART AND G. YOUNG, *A principal axis transformation for non-Hermitian matrices*, Bull. Amer. Math. Soc. (N. S.), 45 (1939), pp. 118–121.

[F81] L. V. FOSTER, *Generalizations of Laguerre's method: Higher order methods*, SIAM J. Numer. Anal., 18 (1981), pp. 1004–1018.

[FL77] M. R. FARMER AND G. LOIZOU, *An algorithm for the total, or partial, factorization of a polynomial*, Math. Proc. Cambridge Philos. Soc., 82 (1977), pp. 427–437.

[Ga73] C. F. GAUSS, *Werke*, Band X, Georg Olms Verlag, New York, 1973.

[Ge58] A. GEL'FOND, *Differenzenrechnung, Deutscher Verlag Der Wissenschaften*, Berlin, 1958 (Russian edition: Moscow, 1952).

[Gem,a] L. GEMIGNANI, *Polynomial root computation by means of the LR algorithm*, BIT, submitted.

[Go94] A. GOLDENSHLUGER, *Nonparametric Estimation in Linear Dynamic Uncertain Systems*, Ph.D. thesis, Technion Faculty of Industrial and Applied Engineering and Management, 1994.

[GO94] S. GOEDECKER, *Remark on algorithms to find roots of polynomials*, SIAM J. Sci. Comput., 15 (1994), pp. 1059–1063.

[Gra72] W. B. GRAGG, *The Padé table and its relation to certain algorithms of numerical analysis*, SIAM Rev., 14 (1972), pp. 1–62.

[Grau71] A. A. GRAU, *The simultaneous improvement of a complete set of approximate factors of a polynomial*, J. Numer. Anal., 8 (1971), pp. 425–438.

[Gre88] L. GREENGARD, *Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.

[He74] P. HENRICI, *Applied and Computational Complex Analysis*, 1, John Wiley, New York, 1974.

[HG69] P. HENRICI AND I. GARGANTINI, *Uniformly convergent algorithms for the simultaneous approximation of all zeros of a polynomial*, (in [DH69]).

[Ho70] A. S. HOUSEHOLDER, *The Numerical Treatment of a Single Nonlinear Equation*, McGraw–Hill, New York, 1970.

[Ho71] A. S. HOUSEHOLDER, *Generalization of an algorithm of Sebastiao e Silva*, Numer. Math., 16 (1971), pp. 375–382.

[HPR77] E. HANSEN, M. PATRICK, AND J. RUSNACK, *Some modifications of Laguerre's method*, BIT, 17 (1977), pp. 409–417.

[IMSL87] *IMSL User's Manual*, version 1.0, chapter 7, 1987.

[JT70] 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), pp. 252–263.

[JT72] M. A. JENKINS AND J. F. TRAUB, *Algorithm* 419: *Zeros of a complex polynomial*, Comm. ACM, 15 (1972), pp. 97–99.

[Ka66] W. KAHAN, *Numerical linear algebra*, Canad. Math. Bull., 9 (1966), pp. 757–801.

[Ke66] I. O. KERNER, *Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen*, Numer. Math., 8 (1966), pp. 290–294.

[Ki94] P. KIRRINNIS, *Polynomial factorization and partial fraction decomposition by simultaneous Newton's iteration*, extended abstract, 1994, J. Complexity, submitted.

[Kim88] M. H. KIM, *On approximate zeros and rootfinding algorithms for a complex polynomial*, Math. Comp., 51 (1988), pp. 707–719.

[KL92] D. KAPUR AND Y. N. LAKSHMAN, *Elimination methods: An introduction*, in Symbolic and Numerical Computation for Artificial Intelligence, B. Donald, D. Kapur, and J. Mundy, eds., Academic Press, New York, 1992, pp. 45–89.

[Kn81] D. E. KNUTH, *The Art of Compupter Programming* 2: *Seminumerical Algorithms*, Addison-Wesley, Reading, MA, 1981 (new edition in 1997).

[KO63] A. KARATSUBA AND YU. OFMAN, *Multiplication of multidigit numbers on automata*, Soviet Phys., Dokl., 7 (1963), pp. 595–596.

[KR90] R. KARP AND V. RAMACHANDRAN, *A survey of parallel algorithms for shared memory machines*, Handbook for Theoretical Computer Science, J. van Leeuwen, ed., North–Holland, Amsterdam, 1990, pp. 869–941.

[KS94] M.-H. KIM AND S. SUTHERLAND, *Polynomial root-finding algorithms and branched covers*, SIAM J. Comput., 23 (1994), pp. 415–436.

[Ku69] I. KUPKA, *Die numerische Bestimmung mehrfacher und nahe benachbarter Polynomnullstellen nach einem Bernoulli-Verfahren*, in Constructive Aspects of the Fundamental Theorem of Algebra, John Wiley, London, New York, 1974, pp. 181–191.

[M73] K. MADSEN, *A root-finding algorithm based on Newton's method*, BIT, 13 (1973), pp. 71–75.

[Ma54] H. MAHLEY, *Zur Iterativen Auflösung Algebraisher Gleichungen*, Z. Angew. Math. Physik, 5 (1954), pp. 260–263.

[Mi92] V. S. MILLER, *Factoring polynomials via relation-finding*, Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 606 (1992), pp. 115–121.

[MN93] J. M. MCNAMEE, *A bibliography on roots of polynomials*, J. Comput. Appl. Math., 47 (1993), pp. 391–394.

[MR75] K. MADSEN AND J. REID, *Fortran Subroutines for Finding Polynomial Zeros*, Report HL75/1172 (C.13), Computer Science and Systems Division, A. E. R. E. Harwell, Oxford, 1975.

[NAG88] *NAG Fortran Library Manual*, Mark 13, Vol. 1, 1988.

[N94] C. A. NEFF, *Specified precision polynomial root isolation is in NC*, J. Comput. System Sci., 48 (1994), pp. 429–463.

[Ne57] O. NEUGEBAUER, *The Exact Science in Antiquity*, 2nd ed., Brown University Press, Providence, RI, 1957.

[NR94] C. A. NEFF AND J. H. REIF, *An $O(n^{l+\epsilon})$ algorithm for the complex root problem*, in Proc. 35th Ann. IEEE Symp. on Foundations of Computer Science, IEEE Computer Society Press, 1994, pp. 540–547.

[O40] A. M. OSTROWSKI, *Recherches sur la Méthode de Graeffe et les Zéros des Polynomes et des Series de Laurent*, Acta Math., 72 (1940), pp. 99–257.

[P87] V. Y. PAN, *Sequential and parallel complexity of approximate evaluation of polynomial zeros*, Comput. and Math. (with Applications), 14 (1987), pp. 591–622.

[P87a] V. Y. PAN, *Algebraic complexity of computing polynomial zeros*, Comput. and Math. (with Applications), 14 (1987), pp. 285–304.

[P89] V. Y. PAN, *Fast and efficient parallel evaluation of the zeros of a polynomial having only real zeros*, Comput. and Math. (with Applications), 17 (1989), pp. 1475–1480.

[P90] V. Y. PAN, *Estimating the extremal eigenvalues of a symmetric matrix*, Comput. and Math. (with Applications), 20 (1990), pp. 17–22.

[P94] V. Y. PAN, *New techniques for approximating complex polynomial zeros*, in Proc. 5th Ann. ACM-SIAM Symp. on Discrete Algorithms, ACM Press, New York and SIAM, Philadelphia, PA, 1994, pp. 260–270.

[P94a] V. Y. PAN, *New resultant inequalities and complex polynomial factorization*, SIAM J. Comput., 23 (1994), pp. 934–950.

[P95] V. Y. PAN, *Optimal (up to polylog factors) sequential and parallel algorithms for approximating complex polynomial zeros*, in Proc. 27th Ann. ACM Symp. on Theory of Computing, ACM Press, New York, 1995, pp. 741–750.

[P95a] V. Y. PAN, *Deterministic improvement of complex polynomial factorization based on the properties of the associated resultant*, Comput. and Math. (with Applications), 30 (1995), pp. 71–94.

[P95b] V. Y. PAN, *Weyl's quadtree algorithm for the unsymmetric eigenvalue problem*, Appl. Math. Lett., 8 (1995), pp. 87–88.

[P96] V. Y. PAN, *Optimal and nearly optimal algorithms for approximating polynomial zeros*, Comput. Math. Appl., 31 (1996), pp. 97–138.

[P96a] V. Y. PAN, *On Approximating Polynomial Zeros: Modified Quadtree (Weyl's) Construction and Improved Newton's Iteration*, Research report 2894, INRIA, Sophia-Antipolis, France, 1996.

[P96b] V. Y. PAN, *Numerical Computation of a Polynomial GCD and Extensions*, Research report 2969, INRIA, Sophia-Antipolis, France, 1996.

[P96c] V. Y. PAN, *Faster computing x^n mod $p(x)$ and an application to splitting a polynomial into factors over a fixed disc*, J. Symbolic Comput., 22 (1996), pp. 377–380.

[P,a] V. Y. PAN, *Faster solution of the key equation for decoding the BCH error-correcting codes*, in Proc. 29th Ann. ACM Symp. on Theory of Computing, ACM Press, New York, 1997.

[PD93] V. Y. PAN AND J. DEMMEL, *A new algorithm for the symmetric tridiagonal eigenvalue problem*, J. Complexity, 9 (1993), pp. 387–405.

[PeS87] M. S. PETKOVIC AND L. V. STEFANOVIC, *On some iteration function for the simultaneous computation of multiple complex polynomial zeros*, BIT, 27 (1987), pp. 111–122.

[PKSHZ96] V. Y. PAN, M.-H. KIM, A. SADIKOU, X. HUANG, AND A. ZHENG, *On isolation of real and nearly real zeros of a univariate polynomial and its splitting into factors*, J. Complexity, 12 (1996), pp. 572–594.

[PT85] L. PASQUINI AND D. TRIGIANTE, *A globally convergent method for simultaneously finding polynomial roots*, Math. Comp., 44 (1985), pp. 135–149.

[Q94] M. J. QUINN, *Parallel Computing: Theory and Practice*, McGraw–Hill, New York, 1994.

[R87] J. RENEGAR, *On the worst-case arithmetic complexity of approximating zeros of polynomials*, J. Complexity, 3 (1987), pp. 90–113.

[R89] J. RENEGAR, *On the worst case arithmetic complexity of approximating zeros of systems of polynomials*, SIAM. J. Comput., 18 (1989), pp. 350–370.

[R92] J. RENEGAR, *On the computational complexity and geometry of the first order theory of reals*, J. Symbolic Comput., 13 (1992), pp. 255–352.

[RS92] J. RENEGAR AND M. SHUB, *Unified complexity analysis for Newton LP methods*, Math. Programming, 53 (1992), pp. 1–16.

[Sa84] H. SAMET, *The quadtree and related hierarchical data structures*, Comput. Surveys, 16 (1984), pp. 187–260.

[Schö82] A. SCHÖNHAGE, *The Fundamental Theorem of Algebra in Terms of Computational Complexity*, Math. Dept., University of Tübingen, Tübingen, Germany, manuscript.

[Schö84] A. SCHÖNHAGE, *Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm*, Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 172 (1984), pp. 436–447.

[SchöSt71] A. SCHÖNHAGE AND V. STRASSEN, *Schnelle Multiplikation grosse Zahlen*, Computing, 7 (1971), pp. 281–292.

[Schr57] J. SCHRÖDER, *Uber das Newtonsche Verfahren*, Arch. Rational Mech. Anal., 1 (1957), pp. 154–180.

[Se94] H. SENOUSSI, *A quadtree algorithm for template matching on pyramid computer*, Theory Comp. Sci., 136 (1994), pp. 387–417.

[SeS41] J. SEBASTIAO E SILVA, *Sur une Méthode d' Approximation Semblable à Celle de Graeffe*, Portugal Math., 2 (1941), pp. 271–279.

[Sm81] S. SMALE, *The fundamental theorem of algebra and complexity theory*, Bull. Amer. Math. Soc., 4 (1981), pp. 1–36.

[Sm85] S. SMALE, *On the efficiency of algorithms of analysis*, Bull. Amer. Math. Soc., 13 (1985), pp. 87–121.

[Sm86] S. SMALE, *Newton's method estimates from data at one point*, in The Merging of Disciplines: New Directions in Pure, Applied and Computational Mathematics, R. Ewing, K. Gross, and G. Martin, eds., Springer-Verlag, New York, 1986, pp. 185–196.

[SS93] M. SHUB AND S. SMALE, *Complexity of Bezout's theorem* I: *Geometric aspects*, J. Amer. Math. Soc., 6 (1993), pp. 459–501.

[SS93a] M. SHUB AND S. SMALE, *Complexity of Bezout's theorem* II: *Volumes and probabilities*, Computational Algebraic Geometry, F. Eyssette and A. Galligo, eds., Progress in Mathematics 109, Birkhäuser, 1993, pp. 267–285.

[SS93b] M. SHUB AND S. SMALE, *Complexity of Bezout's theorem* III: *Condition number and packing*, J. Complexity, 9 (1993), pp. 4–14.

[SS93c] M. SHUB AND S. SMALE, *Complexity of Bezout's theorem* IV: *Probability of success, extensions*, SIAM J. Numer. Anal., to appear.

[SS93d] M. SHUB AND S. SMALE, *Complexity of Bezout's theorem* V: *Polynomial time*, Theoret. Comput. Sci., 133 (1994), pp. 141–164.

[To63] A. L. TOOM, *The complexity of a scheme of functional elements realizing the multiplication of integers*, Soviet Math. Dokl., 3 (1963), pp. 714–716.

[TT94] K.-C. TOH AND L. N. TREFETHEN, *Pseudozeros of polynomials and pseudospectra of companion matrices*, Numer. Math., 68 (1994), pp. 403–425.

[Tu68] P. TURAN, *On the approximate solution of algebraic functions*, Comm. Math. Phys. Class Hung. Acad., XVIII (1968), pp. 223–236.

[Tu75] P. TURAN, *Power sum method and approximative solution of algebraic equations*, Math. Comp., 29 (1975), pp. 311–318.

[Tu84] P. TURAN, *On a New Method of Analysis and Its Applications*, John Wiley, New York, 1984.

[VdS70] A. VAN DER SLUIS, *Upper bounds for roots of polynomials*, Numer. Math., 15 (1970), pp. 250–262.

[VdW] B. L. VAN DER WAERDEN, *Modern Algebra*, Frederick Ungar Publishing Co., New York, 1953.

[W903] K. WEIERSTRASS, *Neuer Beweis des Fundamentalsatzes der Algebra*, Mathematische Werker, Tome III, Mayer und Mueller, Berlin, 1903, pp. 251–269.

[We24] H. WEYL, *Randbemerkungen zu Hauptproblemen der Mathematik*, II, *Fundamentalsatz der Algebra and Grundlagen der Mathematik*, Math. Z., 20 (1924), pp. 131–151.

[Wer82] W. WERNER, *On the simultaneous determination of polynomial roots*, Lecture Notes in Math., Springer-Verlag, Berlin, 953 (1982), pp. 188–202.

[Wi78] H. S. WILF, *A global bisection algorithm for computing the zeros of polynomials in the complex plane*, J. ACM, 25 (1978), pp. 415–420.